University of Padua

Departement of Mathematics
Master degree in Computer Science

Poseidon: Mitigating Interest Flooding DDoS Attacks in Named Data Networking

Supervisor:
prof. Mauro Conti
University of Padua - Italy

Author:
Alberto Compagno
Student number: 606512

Co-supervisor:
prof. Paolo Gasti
NYIT - United States

External Reviewer:
prof. Claude Castelluccia
INRIA - France

Academic Year 2011-2012

Acknowledgements

I want to thank prof. Mauro Conti for the time he has dedicated to me, his support and his collaboration in this work. My thanks go also to prof. Paolo Gasti and prof. Gene Tsudik for their collaboration and their useful advices.

I will never say enough "thank you" to my family. My parents, Sergio and Graziella, that supported me during all these seven years at the University. For the big amount of time, money and especially patience they invested in me during this 26 years. My brother Nicola that helped me in the hard times and suggested me with his experience.

Thanks to my relatives for the patience they have stood to wait the conclusion of my studies (now we can "eat"!). To my grandmother who always has a thought for me. To my cousin Cristina for all the films we have seen together (and the time we have spent to choose them).

Thanks to my "second brother" for all the evening we have spent eating "cioppette". To the "overseas" friend that has supported my insane sms in the last period. To her wherewith the last period has been less hard. To the guys of "tesisti" room for the breaks to clear up our's mind. To all of my friends, new and old, that believe in me.

Abstract

Internet has been one of the most successful inventions in the last fifty years. Thanks to its fast widespread availability has become the standard way for world communications. The emerging usage models and new access methods expose some limitations of the current Internet architecture, that was conceived back in 1970-s. Content-Centric Networking (CCN) is an emerging networking paradigm being considered as a possible replacement for the current IP-based host-centric Internet infrastructure. In CCN, content becomes a first-class entity and is directly named. CCN focuses on content distribution, which dominates current Internet traffic and is arguably not well served by IP. Named-Data Networking (NDN) is an example of CCN. NDN is also an active research project under the NSF-sponsored Future Internet Architectures (FIA) program. FIA emphasizes security and privacy from the outset and by design. To be a viable Internet architecture, NDN must be resilient against current and emerging threats. This thesis tries to give its contribution to the development of this new architecture.

This thesis focuses on distributed denial-of-service (DDoS) attacks, especially, interest flooding – an attack class that exploits key architectural features of the NDN architecture. We initially show that Interest Flooding Attacks (IFA) is a realistic threat and we demonstrate how to implement it. We show an *adversary* with limited resources can use such attacks to significantly influence network performance. We provide simulations on more than one topology to obtain better and more realistic results. We then introduce Poseidon: a toolkit for detecting and mitigating interest flooding attacks. We also report on the results of extensive simulations aimed at assessing effectiveness of our countermeasure.

Contents

List of Figures

Chapter 1

Introduction

Born from a research commissioned by the United States government, Internet is now one of the most popular and revolutionary technologies in the world. Its amazing success story is due to the worldwide spread of TCP/IP. Come to light in the early of the 70-s, TCP/IP protocol was created following the only existing communication system; telephony. This leads to base TCP/IP protocol on host-to-host communication. But, after 40 years, the way people access and utilize Internet has changed radically and host-to-host communication do not ever meet the current user requirements. Today, the Internet has to accommodate new services, new usage models and new access technologies. Smartphones, tablets, laptops allow users to be constantly on-line; sharing contents and retrieving information from Internet is a very frequent action in our life. In addition, user mobility and device heterogeneity has reached levels unthinkable during the development of TPC/IP. All these new changes have brought to light the limits of the current Internet architecture.

To this end, there are some recent research efforts [11; 32; 35; 36; 49] with the long-term goal of designing and deploying a next-generation Internet architecture. These new architectures are designed to better serve today's needs and allow the current growth rate of the Internet to continue for the foreseeable future.

Named Data Networking (NDN) is a new promising internet architecture. It is one of the five NSF-sponsored Future Internet Architectures (FIA) [17] and, like the rest, it is an on-going research effort. It is based on the principle of Content-Centric Networking, where content – rather than hosts – occupies the central role in the communication architecture. NDN is primarily oriented towards efficient large-scale content distribution. Rather than establishing direct IP connections with a host serving content, NDN consumers directly request (i.e., express *interest* in) pieces of content by name; the network is in charge of finding the closest copy of the content, and of retrieving it as efficiently as possible. This decoupling of content

and location allows NDN to efficiently implement multicast, content replication and fault tolerance.

One of the key goals of the NDN project is *"security by design"*. In contrast to today's Internet, where security problems were (and are still being) identified along the way, the NSF FIA program (for all of its projects) stresses both awareness of issues and support for features and counter-measures from the outset.

This thesis addresses distributed denial of service (DDoS) attacks in NDN. DDoS are considered to be a serious threat in the current Internet architecture and NDN do not seem to be immune to them. Generally a DDoS consist of a high number of compromised and collaborating hosts (zombie) with the purpose of preventing utilization of some service. They usually act by injecting specially crafted IP packets in order to overwhelming their victims. The effectiveness of DDoS attacks is usually proportional to the number of zombies controlled by the *adversary*. Being subject to this weakness, NDN might actually offer avenues for new DDoS attacks.

In NDN, communication is based on consumer's requests (called "interests") and producer's reply (called "contents"). Every time a consumer injects an interest on the network, NDN routers have to store a small amount of transient state. This state is flushed as the content is routed back to the consumer. This has been pointed out in previous work as a plausible attack vector – under the name of interest flooding attack (IFA) [18]. However, there has been, thus far, no evidence whether IFA is even possible. In particular, to the best of our knowledge, there is no experimental work that estimates the amount of resources (i.e., bandwidth, computing power, etc.) required to successfully carry out IFA.

1.1 Roadmap and Contributions

Motivated by the importance of addressing security in the early stages of a potential new Internet architecture, we focus on DDoS over NDN, specifically, using IFA. We believe that IFA and countermeasures deserve an in-depth investigation before NDN can be considered ready for large-scale deployment.

This thesis provides the first systematic analysis of IFA over the current NDN protocol via extensive simulations. We show strong evidence that IFA threat is not just theoretical, and that it is relatively easy to realize IFA with rather limited resources. We analyze IFA over two topologies: (1) a simple one that clearly illustrates the effects of IFA, and (2) a larger one (representing the German research network DFN [20]) which shows that effects of IFA generalize to more realistic Internet-like environments.

We discuss both *proactive* and *reactive* countermeasures designed to respectively

prevent and *react against* IFA. We then focus on reactive countermeasures. We identify techniques for early detection of IFA. (This was left as an open problem in [18].) We then describe the design and the implementation of Poseidon – a toolkit for local and distributed attack mitigation. Finally, we report on the effectiveness of proposed methods.

1.2 Organization

In Section 2 we present briefly some related works in the area. We proceed with an overview of NDN in Section 3. We then provide an overview of IFA and countermeasures in Section 4. In Section 4.3 we details our simulation environment, while in Section 4.4 we assess the impact of IFA with different setups. We show effects of IFA in different scenario and we consider the worst case as scenario to test our solution. Section 5 presents Poseidon, our countermeasure for IFA – which we evaluate in Section 6. We demonstrate that our solution works very well in our setup which is the worst case for the topologies we have considered. We conclude in Section 7 by summarizing our results and by presenting our future plans to improve Poseidon.

Chapter 2

Related Work

Content-Centric Networking (CCN) is not the first project that address the problem of designing an architecture that replace the current one. The first was Information-Centric Networking (ICN) which defines the main idea of basing communication on contents instead of hosts. CCN was born from this main idea with the purpose to detail it and to create the new internet architecture. To accomplish this aim it has been created Named Data Networking (NDN), an instantiation of CCN.

However, CCN is not the only project created from ICN. Other important architectures has been defined and two of this have influence significantly the design of CCN. In the Section 2.1 and Section 2.2 we present briefly the Data-Oriented Network Architecture [26] and TRIAD [10] respectively. In the Section 2.3 we present some preliminary work on NDN that address security,robustness and different utilization environment.

2.1 DONA

In the current Internet architecture, DNS is a fundamental part. However, DNS was developed rather late in the Internet's evolution, after many basic pieces of the architecture were in place. For instance, TCP session were already bound to IP addresses and the Berkeley Socket API referred to addresses, not name. This decision limited the extent to which DNS names could permeate the architecture. As a result, the current role of naming in the architecture is more an accident of history that the result of principled architectural design.

DONA is the first ICN approach and had significant influence on other projects. It redefines name and naming resolution to base communication on names instead of IP addresses. DONA uses "flat" self-certifying names. They are computed as the

cryptographic hash of the producer's public key, P, and a (possibly) human-readable label, L. Rather than use DNS servers, DONA rely on a new class of network entities called resolution handlers (RHs). Names resolution is accomplished by using two primitives: *FIND(P:L)* and *REGISTER(P:L)*. The first one is used to locate and retrieve content basing on name, while the last is used to register and publish a new content with a tree of trusted resolution handler. Only after the registration phase is ended the content can be successfully retrieve by users. Once the content is located, packets are exchanged with the original requester using standard IP routing.

If the utilization of standard IP routing is an interesting feature and can really help in the diffusion of DONA, DONA hides some important drawbacks. In fact, this structure prevents DONA to manage content dynamically created and introduces some delay when a piece of content changes location. Changing location to a content requires a new registration of the content and the consequent propagation through the network. NDN does not suffer from this drawbacks.

2.2 TRIAD

TRIAD names content using human-readable, location-independent names. It defines a new content layer and bases contents distribution on it. This new layer provides scalable content routing, caching, content transformation and load balancing, integrating naming, routing and transport connection setup.

Content layer does not return only contents, but in the case where the required content is large, or indeterminate in length, it can return a *network pointer*. For compatibility with the World-wide Web, contents are located with Web Uniform Resource Locator (URL), optionally augmented with cookies. Further, this *network pointer* is realized as an HTTP/TCP connection through which the content is read or written.

The content layer is implemented by: (1) content routers that direct the request towards content servers storing the content, (2) content servers that provide content services, and (3) content caches that transparently store come content nearer to the client than the servers themselves. It may also include content transformers that transform the content from one form to another in response to characteristics of the client and its network connection.

For example, in TRIAD, a client sends a lookup request with the URL for the CNN news page to its content router. The content router looks up this URL (typically just the DNS portion of the URL) and determines a nearby replica for this content, forwarding the request to a next hop accordingly. The next hop content router looks up this URL as well and may determine triadit has this content cached

locally. In this case, it returns TCP connection information to the client, allowing the client to read this content from its caches over this (HTTP/TCP) connection.

TRIAD relies on trusted directories to authenticate content lookups (but not content itself). It also implements an IP-sec-like mechanism for end-to-end security. For additional security, the authors of [10] recommend to limit the network to mutually trusting content routers.

2.3 NDN

Even if NDN is a recent idea, there are a number of works on it. Thanks to it revolutionary purpose, scientific community has invested significant resources and has looked in some part. For example, NDN caching performance optimization has been recently investigated with respect to various metrics including energy impact [27; 43; 51]. A number of works have also investigated the NDN bevhaior in different environment, as for example Telephony[23], Lighting Control systems [5] or Vehicle-to-Vehicle communication [47] while a number of works has investigated NDN security [3; 24] or privacy [15] and has proposed some improvement.

To the best of our knowledge, the work of Xie, et al. [50] is the first to address cache robustness in NDN. This work introduces CacheShield, a mechanism that helps routers to prevent caching unpopular content and therefore maximizing the use of cache for popular one.

There is lots of previous work on DoS/DDoS attacks on the current Internet infrastructure. Current literature addresses both attacks and countermeasures on the routing infrastructure [21], packet flooding [25], reflection attacks [39], DNS cache poisoning [41] and SYN flooding attacks [46]. Proposed counter-measures are based on various strategies and heuristics, including: anomaly detection [7], ingress/egress filtering [45], IP trace back [29; 44], ISP collaborative defenses [9] and user-collaborative defenses [19].

The authors of [18] present a spectrum of possible DoS and DDoS attacks in NDN. They classify those attacks in interest flooding and content/cache poisoning, and provide a high-level overview of possible countermeasures. However, the paper does not analyze specific attacks or evaluate countermeasures.

Chapter 3

NDN Overview

In this chapter we present an overview of the NDN architecture. In Section 3.1 we present CCN, the new paradigm implemented by NDN. In the remaining sections we discuss about: the basic principles that guide the design of NDN (Section 3.2), the new architecture of NDN (Section 3.3), naming (Section 3.4), routing (Section 3.5) and security (Section 3.6).

3.1 CCN model

As explained in Section 2, NDN is an implementation of a completely new paradigm called Content Centric Network. Communication of this new paradigm moves focus from hosts to contents; if in TPC/IP we have hosts as primarily entity to construct a communication, in CCN we do not matter on who there is on the other side of the communication channel. In CCN we only rely on contents; it is not important a *where* content is stored but we only take care of *what* is the content.

This fundamental change in the Internet architecture arises from an incompatibility between what TPC/IP offers - in terms of how connection is established - and how people use Internet. Nowadays, when people surf on the net, they search for contents and information, they do not care of what host contains information, they just want what they need. Instead TPC/IP bases its communication on hosts; in fact each packet of TPC/IP contains two identifiers (source address and destination address) inside it and communication is established indicating a source and a destination. This mismatching, between how TCP/IP works and how people use it, needs a number of additional "stopgap":

Availability: Fast, reliable content access requires awkward, pre-planned, application-specific mechanisms like CDNs and P2P networks, and/or imposes excessive

bandwidth costs.

Security: Trust in content is easily misplaced, relying on untrustworthy location and connection information.

Location-dependence: Mapping content to host locations complicates configuration as well as implementation of network services.

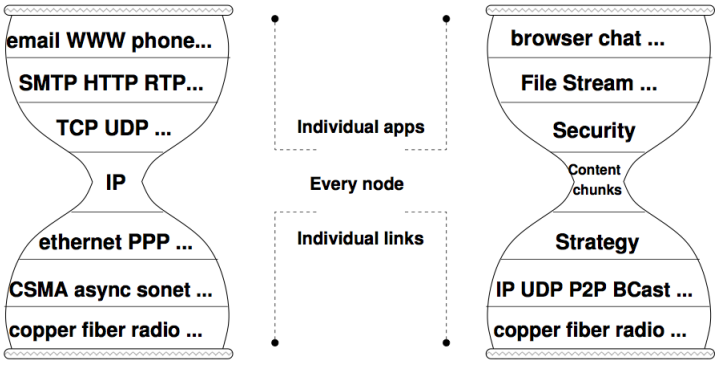


Figure 3.1: CCN moves the universal component of network stack from IP to chunks of named content (image taken from [35])

Figure 3.1 compares TCP/IP and CCN stacks. As for TCP/IP, most of the CCN layer reflect bilateral agreements: e.g. layer 2 is an agreement between two ends on a physical link while layer 4 establishes an agreement between a consumer and a producer. The only layer that needs a global agreement is layer 3. As we can see in the figure, CCN stack maintains the typical "thin waist" part of the TCP/IP stack. It is known that IP's success is due to the simplicity network layer and the weak demand it makes on layer 2, that is why CCN maintains this structure.

In the following sections we describe Named Data Networking, a project with very important participating institutions (PARC, UCI, UCLA, etc) that embrace the new ideas of CCN and implement them. Hereafter we refer to NDN instead of CCN.

3.2 NDN principles

Addressing the limitations and improving the strengths of the current Internet architecture is the main purpose of this new architecture. To accomplish this target, the design of NDN is guided by the following architectural principles:

- The hourglass architecture is what makes the original Internet design elegant and powerful. It centers on a universal network layer (IP) implementing the minimal functionality necessary for global inter-connectivity. This so-called thin waist has been a key enabler of the Internets explosive growth, by allowing lower and upper layer technologies to innovate without unnecessary constraints. NDN keeps the same hourglass-shaped architecture.

- Security must be built into the architecture. Security in the current Internet architecture is an afterthought, not meeting the demands of todays increasingly hostile environment. NDN provides a basic security building block right at the thin waist by signing all named data.

- The end-to-end principle enables development of robust applications in the face of network failures. NDN retains and expands this principle.

- Network traffic must be self-regulating. Flow-balanced data delivery is essential to stable network operation. Since IP performs open loop data delivery, transport protocols have been amended to provide unicast traffic balance. NDN designs flow-balance into the thin waist.

- Routing and forwarding plane separation has proven necessary for Internet development. It allows the forwarding plane to function while the routing system continues to evolve over time. NDN sticks to the same principle to allow the deployment of NDN with the best available forwarding technology while we carry out new routing system research in parallel.

- The architecture should facilitate user choice and competition where possible. Although not a relevant factor in the original Internet design, global deployment has taught us that architecture is not neutral. NDN makes a conscious effort to empower end users and enable competition.

3.3 Architecture

In NDN two different types of packets can be sent through the network: *interest* and *data*. Following the basic idea of CCN, a consumer asks for a content by broadcasting his interest over all the available connectivity. Any node, hearing the interest and having data that satisfies it, can respond with a data packet. This type of communication corresponds to the *producer-consumer* communication model, where a producer share some contents and consumers ask for them. Obviously, in the NDN implementation, there is not restriction on what could be the role

of a node in a communication. It can be a producer for some users and at the same time a consumer for others.

NDN packets are indicated in Figure 3.2. The most important part of an interest packet is the `Content Name`. It contains the name that identify the wanted content and it corresponds with the `Content Name` in the corresponding data Packet. The other two fields in the interest packet indicate: a series of fields useful to refine content searching (`Selector`) and information to locate equal interest packets (`Nonce`).

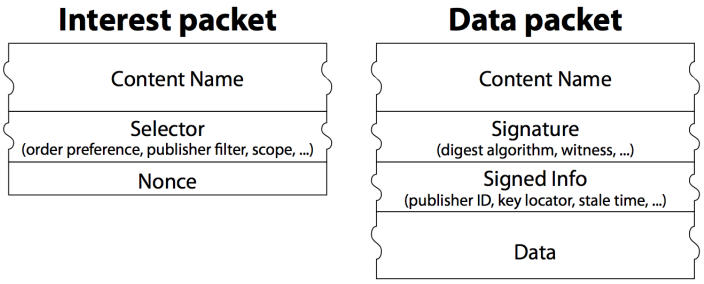


Figure 3.2: NDN packet types (image taken from [35])

In the data Packet, besides `Content Name`, we can find other tree fields: `Signature`, `Signed Info` and `data`. The first contains a signature of the entire packet (excluding signature field), the second contains useful information to verify the correctness of the signature and the last the data required.

Figure 3.3 shows the structure of an NDN node[1]. An NDN node contains 3 structures (`Content store`, `Pending Interest Table`, `Forwarding Information Base`) and a number of faces to manage interest and data packets. While faces are quite the same of interfaces in TCP/IP, so their purpose is to connect a node to the rest of the network, the other 3 structures are completely new:

Content Store, is a buffer memory where it is saved every content received from a face. Its presence allows a node, when an interest arrives, to immediately reply to the interest if the related data packet is in the Content Store.

Pending Interest Table, hereafter PIT, traces interests arrived from a face and forwarded up towards the source of the content until the interests are satisfied.

Forwarding Interest Table is used to forward interest packets toward potential source of matching data. It is almost identical to an IP FIB excepts it allows for a list of outgoing faces rather than a single one.

[1] At this level we do not make any distinction between the type of the node. It represents any devices connect to the network (routers, mobile devices, PCs, etc).

Even if faces have the same purpose of interface, there are some difference between them. A very important difference depends directly on the fundaments of NDN and allows NDN to take advantage on multiple faces. Because NDN bases its communication on content and it does not include host information on its packets, it does not need to obtain or bind layer 2 identity (MAC address) with layer 3 identity (IP address). This allows NDN to change connectivity faster than TCP/IP, in fact NDN can exchange data as soon as it is physically possible to do so.

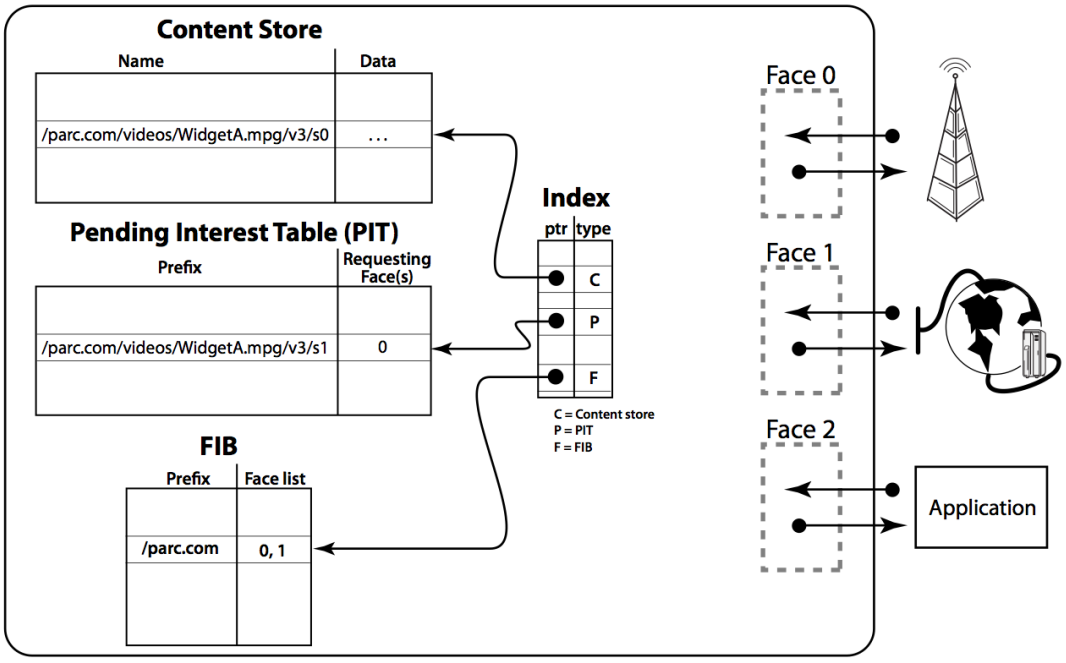


Figure 3.3: NDN forwarding engine model (image taken from [35])

These 3 structures are the ground for the communication in NDN. As explained before, communication is based only on content and not on host; in fact, inside interest and data packets we can not find any trace that refers to host. So, how interest arrives to node that contains the related data packet? And how data packets are forwarded back to the consumer? To answer to these questions we analyze how Content Store, PIT and FIB are used in the communication process.

When an interest arrives to a node, from a face, a lookup is performed to the Content Store. If Content Store contains the data packet associated to the interest, the data is sent back through the arrival face and interest is discarded. Otherwise,

if a match is not found, a second lookup is performed in the PIT. As shows in Figure 3.3, PIT entries associate unsatisfied interest with its arrival faces. It has to be noted that there are not reasons that prevent the arrival of the same interest from different faces; PIT has to trace all arrival faces. However when a new interest arrives and an entry PIT matches with the arrived interest, the arrival face is added to the entry PIT and interest is not forwarded. In this case, a previous interest requiring the same data has been sent toward the producer and sending another interest will be completely useless. However, if no matches occurs with the entries of the PIT, a third lookup is performed in the FIB and a match in this structure means that node knows how to retrieve the desired data. After that, a new entry in the PIT is created and interest are sent out on the face indicated in the FIB.

It should be clear that, on the path from consumer to producer, interest packets leave a breadcrumb (entry in PIT) to the reached nodes. In this way, data packet processing is relatively simple since data is not routed but simply follows the chain of PIT entries back to the original requester(s). A longest-match lookup of a data packets ContentName is done upon arrival. A ContentStore match means the data is a duplicate so it is discarded. A PIT match means the data was solicited by interest sent by this node. The data is (optionally) validated then added to the ContentStore. Then a list is created that is the union of the RequestingFaces list of each PIT match minus the arrival face of the data packet. The data packet is then sent out each face on this list.

The mechanism we have just described allows a number of improvement from the known TCP/IP control flow system. First of all we have to note that, on each hop, for a sent interest at most one data packet is forwarded back. This feature enable in NDN a flow control hop-by-hop that comes for free. Moreover, if a node receives the same interest more times it forwards only the first; this permits NDN to use the minimum number of packets necessary. Remembering that in TCP flow control is done end-by-end, in NDN we have fine-grained flow control without any additional structure, as TCP windows, which is a clear improvement.

3.4 Names

Basing communication on contents instead of on hosts means that we need to identify content in some way. Names are used to disambiguate one content from another. NDN uses *opaque* names and leaves to applications to choose the scheme they prefer that fits they needs. This allow to get an independent naming scheme from the network so that their evolution can growth distinctly.

NDN design assumes hierarchically structured names; a name is composed of a

number of *components* that have no meaning to NDN transport but they should have for applications. For example a document shared by the University of Padua could have the following name `/unipd/documents/documentA.pdf`. This hierarchical structure, in addition to be human readable, allows routing to scale. In particular, as for IP where aggregation is essential in routing scale, this naming scheme encourages the utilization of aggregation (for example we can use `/unipd/documents` to refer to all the document published by University of Padua). A user that does not know the entire name of a content can ask for a list of the contents in a namespace and afterwards ask for a specific one.

This naming structure bring with it a number of features very useful in the common network utilization. For example an application can use its own sequencing system to locate different parts of the same content. It is sufficient that consumer and producer agreed on how different parts are named, and they can extend the name of content with the name of the part.

Furthermore, a name has not to be for sure globally unique; it depends on the visibility of the content. A local content, used only for a local communication, can collide with other name of contents local to other node. A global content, on the other hand, has to use a globally unique name. Manage of the dynamic content is also easy with this name structure. Again, an agreement between producer and consumer on how name a content is sufficient to identify it (for example using the same algorithm to create names).

Because NDN names are opaque, we can improve privacy in communication using encrypted component of the name without interfere with routing system. In fact, when an interest arrives to an NDN node, a simple match is done between the name of the interest and the names store in the FIB. If we substitute the name with the encrypted name in the interest and in the FIB, forwarding mechanism still works.

It is clear that naming system is a very important component in the NDN architecture and not all the part of naming have been defined exhaustively. A simple example is the name clashing of the global content, but also other areas where naming are essential are not completely defined (e.g. identify an old version of a content in a node) and can influence naming system. In fact NDN is a work-in-progess project and scientific community is currently investigating all its parts.

3.5 Routing

Today there are a variety of interesting and effective candidate solutions for most routing problems. Any routing scheme that works well for IP should also work well for NDN, because NDNs forwarding model is a strict superset of the IP model with

fewer restrictions (no restriction on multi-source, multi-destination to avoid looping) and the same semantics relevant to routing (hierarchical name aggregation with longest-match lookup). NDN provides an excellent vehicle to implement a routing protocols transport: at the heart of most routing transport protocols is something very similar to NDNs information-oriented guided-diffusion flooding model since they have to function in the pre-topology phase of networking where peer identities and locations are not known. Since NDN provides a robust information security model, using NDN as a routing transport can make routing infrastructure protection almost automatic.

Furthermore, it is fundamental that the new internet architecture is able to coexist with the current TCP/IP to have a chance to effectively replace the current internet architecture. In fact, it is unthinkable to switch off internet, substitute all routers and turn on internet. NDN is very careful on this problem and it has been thought to avoid it.

We now present an example of how NDN can coexists with the current routing protocols. Both IP forwarding and NDN forwarding are almost identical. They both use prefix-based longest match lookups to find local neighbors "closer" to the identifier matched.

NDN prefixes are very different from IP prefixes, so the main question is whether it is possible to express them in some particular routing protocol. Fortunately, both IS-IS (Internet System to Internet System) and OSPF (Open Shortest Path First) can describe directly connected resources via a general TLV (Type Label Value) scheme that is suitable for distributing NDN content prefixes. The specification says that unrecognized types should be ignored, which means that content routers, implementing the full NDN forwarding model, can be attached to an existing IS-IS or OSPF network with no modifications to the network or its routers. The content routers learn the physical network topology and announce their place in that topology via the adjacency protocol and flood their prefixes in prefix announcements using a NDN TLV.

For example, Figure 3.4 shows an IGP (Interior Gateway Protocol) domain with some IP only routers (single circles) and some IP+NDN routers. The repository next to `A` is announcing (via a NDN broadcast in a local network management namespace) that it can serve interests matching the prefix `/unipd.it/students/documents`. A routing application on `A` hears this announcement (since it has expressed interest in the namespace where such announcements are made), installs a local NDN `FIB` entry for the prefix pointing at the face where it heard the announcement, and packages the prefix into IGP LSA (Interior Gateway Protocol Link-State Advertisement) which is flooded to all nodes. When the routing application on `E`, for example, initially gets this LSA, it creates a NDN face to `A` then adds a

prefix entry for `/unipd.it/students/documents` via that face to the local NDN `FIB`. When a different repository adjacent to `B` announces `/unipd.it/students` and `/unipd.it/students/documents`, `B` floods an IGP LSA for these two prefixes with the result that `E`'s NDN `FIB` is as shown in the figure. An interest in `/unipd.it/students/documents/calendar.pdf` expressed by a client adjacent to `E` will be forwarded to both `A` and `B`, who each forward it to their adjacent repository.

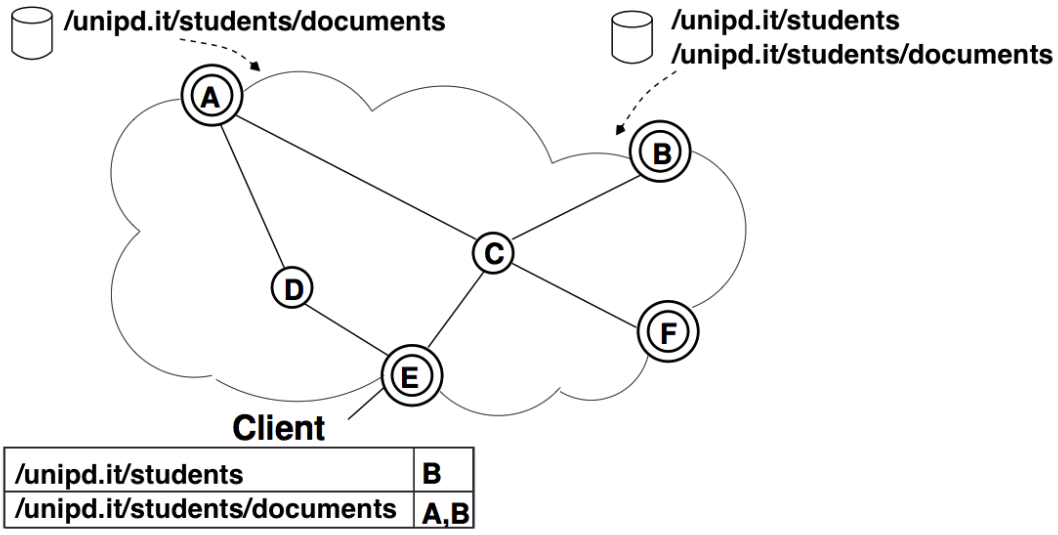


Figure 3.4: Routing interests to a domains students content

NDN dynamically constructs topologies that are close to optimal for both bandwidth and delay (i.e., data goes only where there is interest, over the shortest path, and at most one copy of any piece of data goes over any link). But this delivery topology is clearly non-optimal since a client adjacent to `F` interested in the same movie would result in a second copy of the content crossing the `A-C` or `B-C` link. This happens when an incremental NDN deployment leaves some parts of the physical topology inaccessible to NDN (`C` is not a content router so it cannot cache). As soon as `C` gets the NDN software upgrade, `E` and `F` will forward their interests via it and the distribution will be optimal.

If on Link-state Intra-domain routing level NDN can coexist with not relevant problem, the same thing is not possible on inter-domain routing level. When some customers start to use NDN, it is in the ISP's best interest to deploy content routers to reduce peering costs. However, since costumers are connected to their ISP, it is

trivial for them to learn about the ISP's content router via a service discovery protocol run over the customer ISP peering links. The central problem is to bridge the gap between domain that have content routers but are separated by ISPs that do not. If two ISPs both support content routing but they are connected via ISPs that do not, there is no way for the first ISP to learn of the relevant content router in the second ISP so it will forward interests directly to it. Thus without additional mechanism, ISP routers benefit inbound content (content requested by their customers) but not outbound (content created by their customers). This partially negates a major long term NDN advantage of making traffic near the root of a content distribution tree independent of the popularity of the content.

This problem can be fixed by integrating domain-level content prefixes into BGP. Current BGP inter-domain routing has the equivalent of the IGP TLV mechanism that would allow domains to advertise their customers content prefixes. The BGP AS-path information also lets each domain construct a topology map equivalent to the one constructed in the IGP case, but at the Autonomous System (AS) rather than network prefix level. This map is functionally equivalent to the IGP case (one learns which domains serve interests in some prefix and what is the closest CCN-capable domain on the paths to those domains) so the same algorithms apply.

3.6 Security

Security is one of the innovations included in the NDN stack. Experience in TCP/IP has demonstrated that including a degree of security in the thin-waist layer is a must and NDN has been designed with this requirement.

NDN is built on the notion of content-based security: protection and trust travel with the content itself, rather than being a property of the connections over which it travels. Every data packet is authenticated via digital signatures, and this moves security from hosts to content itself. Currently in IP networks, trust are based on where (from what host) and how (over what sort of pipe); client has to retrieve content directly from the source to trust it. Accordingly whit the CCN theory, NDN forgets about hosts and moves trust directly to the contents.

Signing every data packet allows content publisher to securely bind name to content. This simple addition to a data packet brings an important improvement in trust. First of all a signature verification can be done end-to-end enabling a consumer to check the authenticity of the content. In addition NDN data is publicly authenticatable, signature are standard public key signature and anyone, not just the endpoints, can verify that name-content binding was signed by a particular key.

The signature algorithm, used to sign data packet, is selected by the producer

from a large scale of fixed set and chosen to meet required performance in the signature/verification process. Every data packet moreover include all the information necessary to verify the signature and retrieve the public key necessary. It can include cryptographic digest, or fingerprint of that public key; a shorthand identifier for the publisher; and a key locator to indicate where the key can be obtained, or containing the key itself.

This minimal verification can be surprisingly useful in defending against many types of network attacks but it needs a further refinement. It has not jet defined how use signature to completely trust to a producer. The current solution used in TCP/IP to trust key, and so producer, is to use a PKI. The same infrastructure can be replied in NDN in a simple way; we can create a simple certificate from a data packet. Adding the public key to the name of the content let signature to bind public key with the private key used, that is exactly the main purpose of a certificate.

Other solutions are under investigation to extend the described solution (for example SDSI/SPKI [1; 6; 16]). Anyway, security, privacy and trust area has not been completely covered and this thesis try to help in this purpose addressing a security problem.

Chapter 4

Interest Flooding Attack and Countermeasures

We are now presenting how a particular DDoS attack can be implemented in a NDN network and, we are showing how NDN is no resilient to this kind of attack.

NDN routers differ from their IP counterparts in maintaining data structures used to store state. In particular, CS and PIT are unique to NDN routers and if there are not any mechanism to protect them, an *adversary* could abuse them to implement DoS/DDoS attacks. In this thesis, we focus on attacks using PIT, in particular, how to construct interests that saturate the victim router's PIT.

As mentioned in Section 3, PIT contains information about pending interests. This information is used to route content back to consumers. If the PIT is completely full, all incoming (un-collapsible) interests are dropped. A router with a completely full PIT cannot accept new interests, until the pending ones are either satisfied or expire.

Flooding a router with many distinct interests allows the *adversary* to saturate the PIT. If the rate of incoming interests is higher than the rate at which entries are removed from PIT (either due to returning content or expiration), the router eventually stops forwarding interests. This is precisely the goal of Interest Flooding Attacks (IFA).

As observed in [18], there are at least three ways to perform this attack. The *adversary* can issue closely-spaced interests for: (1) existing static content; (2) dynamically-generated content; or (3) non-existent content. In the rest of this documents, we refer to interests for non existing content as *fake interests*.

In strategy (1), the *adversary* requests distinct content to avoid interest collapsing. These interests are flushed from the victim's PIT when content is returned. However, if the *adversary*'s flooding rate is sufficiently high, the producer (or pos-

sibly a router past the victim) will start dropping packets. This causes interests to linger in the victim's PIT until they expire; usually, after a few seconds, i.e., significantly longer than expected round-trip time. If enough interests are left to expire, the *adversary* can fill up the victim's PIT. However, depending on the victim's ability to satisfy interests quickly (and flush them from the PIT), this strategy may be very expensive. Also, router caches might lower the impact of this attack, satisfying *adversary*'s requests before they reach the victim.

Strategy (2) is similar to (1), except that content is never returned from caches. Also, (2) may impose more load on the producer, due to the increased number of requests for content that can not be precomputed. This could cause higher round-trip latency and higher rate of dropped packets, which forces *adversary*'s interests to remain in the victim's PIT longer.

Strategy (3) allows the *adversary* to create entries in the victim's PIT without receiving any content packets in return. This has several effects:

- Since no content is returned, adversarial interests are stored in the victim's PIT until they expire.
- The maximum rate of adversarial interests does not depend on the bandwidth allocated by the victim to content packets, or on the *adversary*'s ability to receive content.
- Adversarial interests cannot be satisfied by caches, since they request non-existing content.
- Similarly, when constructed properly interests adversarial interests are never collapsed. In particular, adding a random component at the end of each name is enough to prevent any two interests from being aggregated.

These effects allow the *adversary* to efficiently fill up the victim router's PIT. Adversarial interests make efficient use of available bandwidth. Within the (relatively large) PIT expiration window, the *adversary* consumes one PIT entry for each interest it issues. Because of the comparatively higher impact of attack (3), in the rest of this thesis, we focus on IFA via fake interests.

Constructing fake interests that are routed through the victim is pretty straightforward. Let R be the router advertising namespace `/nsf/fia/`. An *adversary* uses name `/nsf/fia/`*rnd* (where "*rnd*" is a random string) to construct a fake interest. Such interests are forwarded through R, however, no content is forwarded back, leaving useless entries in R's PIT.

We now overview some possible IFA countermeasures. We divide them into *proactive* and *reactive*. We then propose a specific reactive countermeasure, Poseidon, discussed in Chapter 5.

4.1 Proactive Countermeasures

As name suggests, the main feature of proactive countermeasures is to anticipate the occurrence of an attack. In this way, when malicious users start to perform an attack, the attack has no effects (or slight effects) on the network. In the following sections we discuss pros and cons of some proactive countermeasures.

4.1.1 Signed Interests

As mentioned in Section 3, interests are not signed and do not carry any consumer address.[1] This allows the *adversary* to implement IFA without being traceable.

Introducing signatures on interests allows routers to implement effective countermeasures once they identify an attack. For example, routers can keep statistics over expiring interests in their PITs. If a large number of such interests corresponds to a relatively small number of sources, routers can identify such sources as malicious and rate-limit (or completely discard) their interests.

There are, however, several significant drawbacks:

- **Privacy:** unsigned interests hide the identity of the consumers. Introducing signatures would lower consumer privacy.
- **Signing Cost:** the cost of signature generation is non-negligible, both in terms of computation and power consumption (especially on mobile devices).
- **Verifying Cost:** more importantly, forcing routers to verify interest signatures would significantly lower their performance. Note that this is different from content signatures which are optional for routers to verify.
- **Aggregating interests:** signatures would prevent routers from aggregating similar interest.
- **Key management:** signed interests are effective against IFA only if it expensive for the *adversary* to generate a large number of *trusted* keys.

4.1.2 Resource Allocation

Let mpx (maximum PIT usage) be the upper bound for the amount of PIT space required to route interests. In particular, mpx is defined as:

$$mpx = rate \cdot timeout,$$

[1] The first hop may be able to identify the consumer through its layer-2 address or, if NDN runs as an IP overlay, IP address.

where *rate* is the sum of maximum rates of incoming interests on all interfaces, and *timeout* is the interest timeout value. If PIT size is at least *mpx*, it cannot be completely filled up, even if all incoming interests are not satisfied.

However, large and fast PITs are difficult and expensive to implement [48]. For this reason, assuming that most content is returned before the corresponding interest expires, PITs are usually smaller than *mpx*. Clearly, a large PIT (close to *mpx*), makes the attack expensive; even though, it does not prevent it.

Alternatively, for a given PIT size, routers could limit the maximum incoming rate for interests so that the product $rate \cdot timeout$ equals PIT size. This would prevent the *adversary* from filling up the PIT. However, it would also reduce the amount of non-malicious traffic that each router can forward.

Finally, a router could set a small PIT timeout value, so that fake interests are removed more promptly. Unfortunately, this would increase the likelihood of content being dropped (whenever it arrives after the corresponding PIT entry is expired), thus affecting overall network performance.

4.1.3 Error Messages

When an interest cannot be satisfied, no error message is returned to the consumer. We could extend the NDN architecture to allow producers and routers to issue error messages when content cannot be returned. Such messages could be routed exactly like content, and flush PIT state of the routers they traverse. This would force the *adversary* to implement IFA using interests that always return content (possibly in the form of error messages), making the attack less effective.

However, this would also have several drawbacks. There is a tradeoff between space usage in PITs and bandwidth usage. In particular, for each PIT entry removed by an error message, one extra packet – i.e., the error message – must be transported by the network.

It is not obvious whether error messages should be signed by the entity who issues them. If they are, signature generation cost may turn out to be prohibitive. If they are not, they can themselves be used to implement attacks. Also, in case of multipath forwarding (i.e., multiple copies of the same interest concurrently forwarded to different interfaces), error messages can be handled in a variety of ways with different results.

We consider error messages a viable but ultimately too flawed approach to mitigate IFA. In the rest of the paper we focus on reactive countermeasures.

4.2 Reactive Countermeasures

We now discuss possible reactive countermeasures. They are characterized by a detection and a reaction phase. Detection can be local or distributed. In the first case, routers rely only on local metrics (e.g., PIT usage, rate of unsatisfied interests, amount of bandwidth used to forward content) to identify an attack. In the second case, nearby routers collaborate to determine whether an attack is in progress and how to react.

In case of successful IFA, the victim router can easily identify the attack by observing whether its PIT is full or whether the bandwidth allocated to forwarding of content is very small. However, it may not be possible for a router on the path to the victim to detect an attack in progress. Collaborative detection mechanisms allow routers to exchange information about their state, with the goal of detecting an attack in progress as soon as possible.

Once an attack has been identified, routers must react to mitigate it. Ideally, routers should drop all interests from the *adversary*, while forwarding interests from honest users. Alternatively, routers could drop all interests corresponding to non-existent content. Clearly, routers can implement neither of these countermeasures: a smart *adversary* may be able to hide malicious traffic within legitimate data streams; additionally, routers cannot determine *a-priori* which interests will return content. Feasible strategies include:

1. Identifying a (set of) namespace(s) under attack, and rate limit the corresponding interests. While this may not provide significant relief to the victim router, it could reduce the effects of the attack over other close-by routers.

2. Identifying a set of interfaces to which the attack is directed, or from which it is coming, and rate-limit them. If fake interests are spread over many different namespaces but are concentrated over a limited set of interfaces, this approach may turn to be more effective.

3. In addition to strategies 1. and 2., routers could identify PIT entries corresponding to (likely) fake interests and remove them before they time out.

When routers rely on collaborative detection mechanisms they can not only exchange information about the *existence* of an attack, but also the (locally detected) properties of such attack. This way, the strategies described above can be augmented to take into account feedback from multiple routers.

In this thesis, we consider a particular collaborative approach – known as push-back [18] – to counter IFA. For each interface over which an attack has been detected,

the router informs its neighbors. These routers might then react without waiting to detect the attack themselves.

The amount of traffic generated by push-back messages is negligible, in particular with respect to that of error messages overviewed in Section 4.1.3. In fact, when a router receives a push-back messages it does not forward it any further. Moreover, the number of unsatisfiable interests – and therefore the number of corresponding hypothetical error messages – is expected to be significantly higher than the number of push-back messages sent under normal conditions.

Before presenting Poseidon (deferred to Chapter 5), we introduce the evaluation environment we use to assess the feasibility of IFA and the effectiveness of our countermeasures.

4.3 Evaluation Environment

In this we describe the topologies used in our experiments and the behavior of honest users. Then, we analyze network traffic and routers' PIT usage under such conditions. This provides us with a baseline that we use to determine the effects of IFA and the benefits of Poseidon.

4.3.1 Simulation Environment

We rely on simulations to quantify the effects of the attack and determine the benefits of our countermeasures. In particular, we use NS-3 DCE implementation for CCNx.

CCNx [8] is the official implementation of NDN. Originally developed by PARC, it has been released as open-source project in 2009.

NS-3 [37] is a well-known open-source discrete-event network simulator that supports wired and wireless networks based on IP or other protocols. Our setup relies on wired networking; we run NDN as an overlay, over IP. We underline that running NDN as an overlay over IP reflects the status of the current NDN implementation. In fact, even in the official in this testbed, links between routers are essentially GRE (Generic Routing Encapsulation) tunnels [35].

New protocols in NS-3 are defined using C++ or Python; such code cannot be reused directly for actual (non-simulated) implementations. For this reason, Inria has been developing DCE [12], or Direct Code Execution. DCE allows regular applications to access a network environment simulated using NS-3. This way, the same code does not need to be rewritten for NS-3.

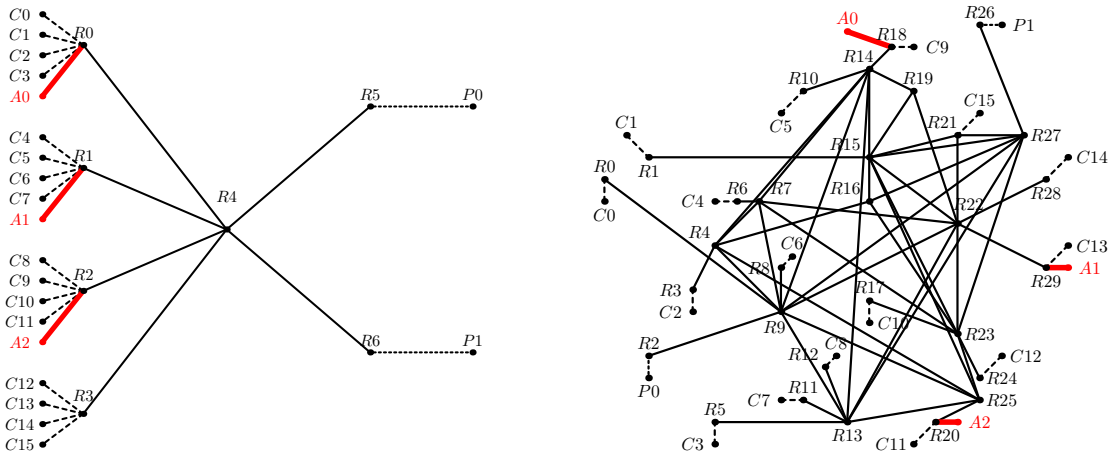
We used this setup to simulate multiple CCNx nodes over an IP network. The

advantages of this approach compared to the use of an implementation of NDN specific for NS-3 are numerous. In fact, we can obtain results that are very close to how non-simulated implementation of NDN would behave, since they are sharing the same code.

Unfortunately, DCE adds a significant overhead on top of NS-3. For this reason, we run our simulation over a cluster computer composed of 32 nodes with heterogeneous resources (a representative node – widely used for our simulation – has two CPUs Intel Xeon E5-2680 at 2.70GHz, and 256GB of RAM). Nodes are connected via Infiniband/Ethernet.

4.3.2 Experimental Setup

Our experiments are performed using the architectures in Figure 4.1.



(a) Simple architecture

(b) DFN architecture

Figure 4.1: Considered architectures: topologies

In particular, the network in Figure 4.1(a) is designed to emphasize the effects of attack and countermeasures. The small size and simple topology allows us to rule out complex effects and provide a clear interpretation of our results. We then consider a more complex topology, represented in Figure 4.1(b), which corresponds to the "Deutsches ForschungsNetz" – DFN, the German Research Network [13; 14; 20]. This topology allows us to confirm the observations made in our simple topology and substantiates the generality of our claims. The DFN topology has been proposed before as a meaningful architecture for network simulations [20].

In the rest of the document, we use the notation RX, CX, PX, and AX to represent the X-th router, consumer, producer, and *adversary*-controlled node, respectively. Continuous lines in Figure 4.1 indicate connections between routers, dashed lines denote connections between consumers and routers, while dotted lines show connections between producers and routers. In both topologies, we considered 16 (honest) consumers and 2 producers.

We first analyze these topologies without any adversarial traffic. This provides us with a series of values that represent a baseline, which we compare with the analogous values obtained measuring the effects of IFA and of our countermeasure. Each consumer issues interests for existent content produced by P0 and P1. Each interest retrieves a distinct piece of content, so that different interests cannot be collapsed and do not take advantage of routers caches. Consumers send a short burst of 30 interests, spaced by 2 ms, at time t =1,000 ms of the simulation. Starting from t = 1,200 ms, consumers switch to a rate of 1 interest every 10.7 ms. In our configurations, such interest spacing allows routers to forward interests roughly at the same rate at which they receive content packets. We set routers' PIT size to 120,000 B, while the interest expiration time was set to 4,000 ms.

We run each simulation 20 times and we report the average of the various runs in figures 4.2 and 4.3. In particular, figures 4.2(a) and 4.2(b) show the total number of content packets (y-axis) received by the different routers (x-axis), for the simple architecture and the DFN one, respectively. Figures 4.3(a) and 4.3(b) show the PIT usage (y-axis) as a function of the simulation time (x-axis) for the two topologies. We omit PIT usage plots for R1 and R2 in Figure 4.3(a), since they are very similar to that of R0. The maximum value on the y-axis for figures 4.3(a) and 4.3(b) corresponds to the total space available in the PITs (120,000 B). Also, the two vertical lines (at 1,000 ms and 26,000 ms) indicate the instant when consumers start and stop sending interests. (The same notation is used in all graphs that refer to PIT usage reported in this document.)

We can see that throughput's figures (4.2(a) and 4.2(b)) and PIT usage's figures (4.3(a) and 4.3(b)) reflect some architecture proprieties. Considering simple architecture we can see how R4 has a higher load than other routers. In fact, looking at the simple architecture topology 4.1(a) we can see that all contents collapse to R4; R5 and R6 receive half of all contents and R0, R1, R2, R3 receive a quarter of all contents. This distribution of contents is clearly visible in both charts 4.2(a) and 4.3(a). We can also observe another important feature. In chart 4.3(a) we can see that R5 and R6 from time 1,000 ms up to 1,500 ms have a PIT usage higher than the PIT usage of R0, R1, R2 and R3. Starting from 1,500 ms R5 and R6 have a PIT usage lower than the other four routers. This phenomenon can be explained by looking at the topology of the network and the way we send interests. The first 30 interests

are sent in a really short time, this involves PIT to be filled because producers are not able to reply to consumers requests at the same rate. But, when the first burst of interests is ended, interests are sent with a lower frequency hence producer are able to reply to the received and receiving interests. This entails that R5 and R6 are able to flush almost all the interests they have in PIT.

The same behavior is replicated the routers R0, R1, R2, R3, R4 but if we look at the chart we can see that the correspondent curves have less steep slopes. Because of R4 has to manage a higher number of content, it is not able to flush interest at the same rate of R5 and R6 and introduces some lateness in the network. Due to this lateness, R0, R1, R2, R3 can not for sure flush interests with a rate higher than R4 and this is why their slope of curves are slighter than R4 curve's slope.

Looking at the chart 4.3(b) we can identify quite the same behavior of the curves in the chart 4.3(a). In fact there is similar behavior between routers in simple architecture and routers in DFN. This similarity is due to the number connection of a router and the load it has to manage in the simulations. For example, R9 and R27 of DFN are very similar to R4 of simple architecture. In some sense they are a fulcrum of the networks; all interests and contents pass through them. We have another similarity between R5 and R6 of simple architecture and R26 and R2; they are the nodes directly connected with the two producers. Routers R0, R1, R2 in simple topology are connected to the *adversary* as routers R14, R29, R20 in the DFN. Finally, all other routers in DFN do not have a particular role in the network; they are neither directly connected to producers or fulcrums. Because of their "generality", their behavior are closed to router R3 of simple architecture.

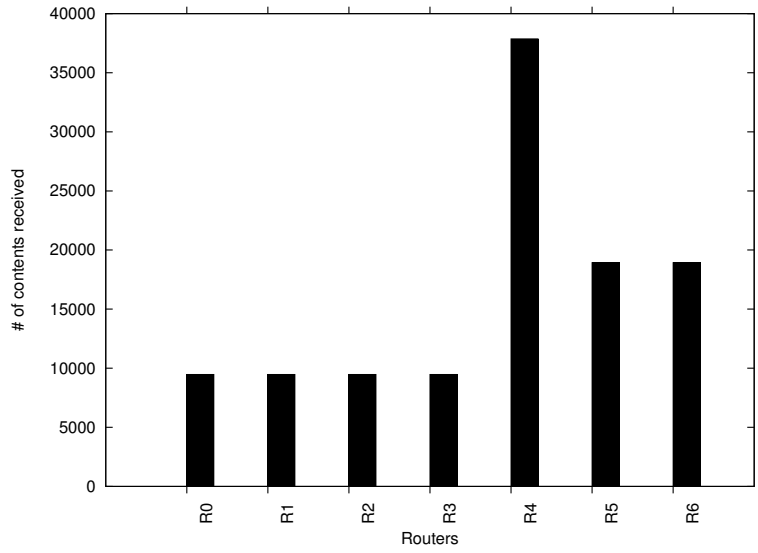
We summarize similarities between router of simple topology and DFN in Table 4.1

Simple topology's routers	**DFN's routers**
R0, R1, R2	R14, R29, R20
R3	R0, R1, R3, R5, R6, R7, R8, R10, R11, R12, R13, R15, R16, R17, R18, R19, R21, R23, R24, R28
R4	R9,R27
R5	R2
R6	R26

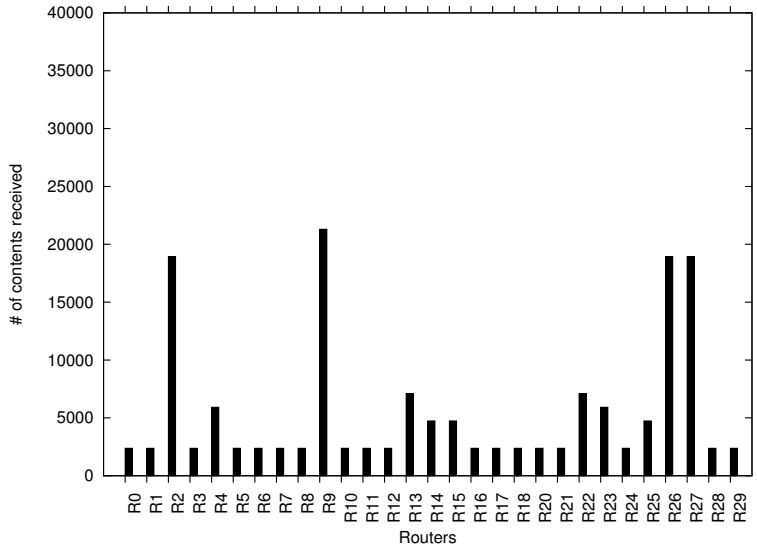
Table 4.1: Similarities between routers of simple architecture and DFN

All these similarity between these two architectures means that also simple ar-

chitecture carry with it proprieties of implemented topologies and confirms that this architecture is a really good start point in observing effects of IFA.

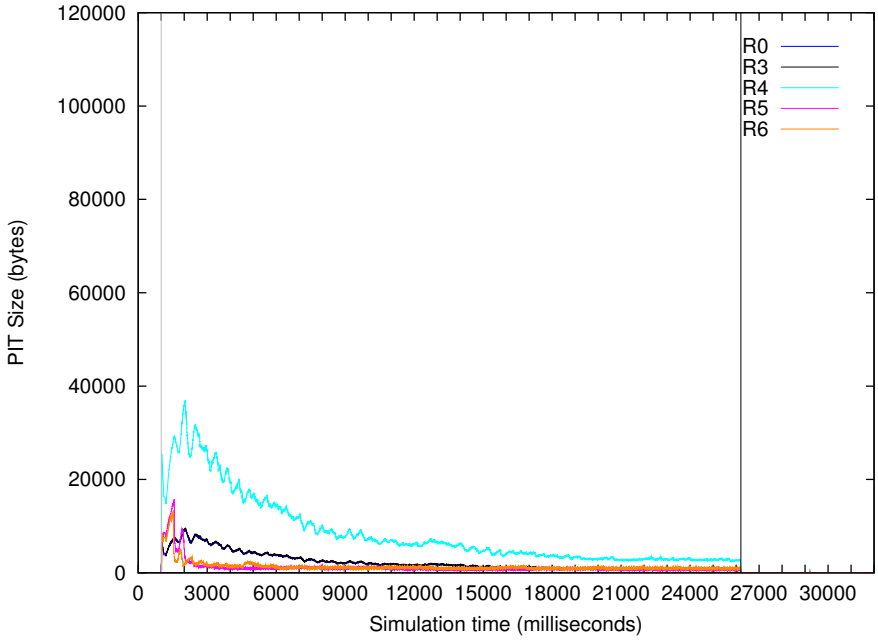


(a) Simple architecture

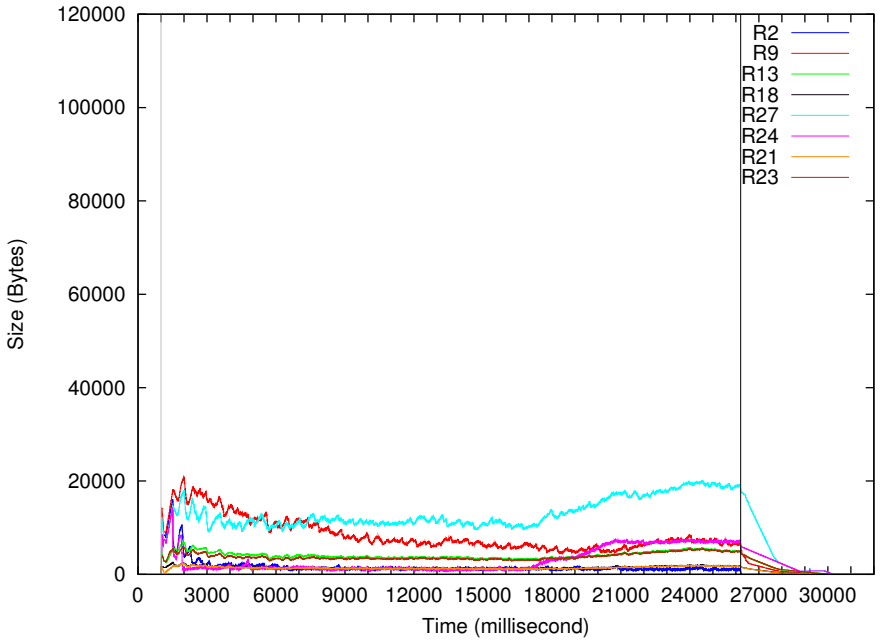


(b) DFN architecture

Figure 4.2: Baseline behavior (no attack): Throughput (absolute values)



(a) Simple architecture



(b) DFN architecture

Figure 4.3: Baseline behavior (no attack): PIT usage

4.4 Attacks

In this section we describe the implementation of IFA attacks. We focus to the different effects an IFA generates in a network.

We assume that the *adversary* is able to control a portion of consumers, through which it issues fake interests. However, the *adversary* is not allowed to control routers. We believe that this restriction is realistic and well represents the current scenario of DDoS attacks (e.g. [22]). While we do not exclude, in principle, that attacks might come from internal routers, we leave the investigation of this as future work.

We identify three parameters that could modify effects of IFA:

1. Size of an interest.
2. Rate of sending of fake interests.
3. Number of consumers controlled by the adversary in the network.

Recalling fields in interests (Section 3.3), we can use name, selector and nonce components to enlarge it. Even if, at this moment the NDN specification does not limits the size of selector component, it is reasonable to think that a complete specification defines a finite size for that component. In the two remaining component, while nonce is calculated from NDN implementation, user has complete control over the name component. Therefore, the only way a user have to enlarge an interest is to set a long name. Nevertheless, neither using a large name is a good solution. If it is true that the largest interest size the earlier a PIT is completely filled, it is also true that a very long name could easily be identified from a router and considered fake. For this reason, in our simulation, we use fake interest with the same size of honest interest.

We think that modifying the other two parameters, rate of sending and number of consumers controlled by the adversary, are a good way to study different behavior of IFA in the network. While we investigate in the rate of sending fake interest (Section 4.4.1), we leave as future work the investigation in the number of controlled consumers.

4.4.1 Attack Effectiveness

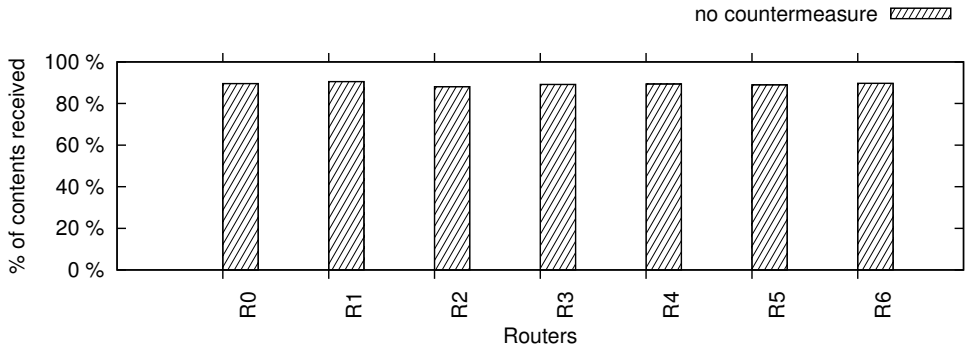
We study the effectiveness of the attack using three different rates to generate fake interests. Compared to the rate used by honest user the rate of generation of fake interest is double, 4 times and 8 times. This means that every *adversary* sends one fake interest every 5.350 ms, 2.675 ms and 1.337 ms respectively.

We have noted that the rate at which the *adversary* injects interests in the network is limited by the capacity of its next-hop routers. In our simulations we observe that successful instantiation of IFA requires very small amount of bandwidth. The *adversary* controls the nodes connected through a red solid line in Figure 4.1. The three adversarial nodes (`A0`, `A1`, `A2`) send interests for non-existent content for the namespace registered by `P0` – i.e., all fake interests are routed to `P0`. Similarly to the honest nodes, the *adversary* starts sending interests at t =1,000 ms. The behavior of the honest consumers is unchanged from the base scenario, i.e., with no *adversary*.

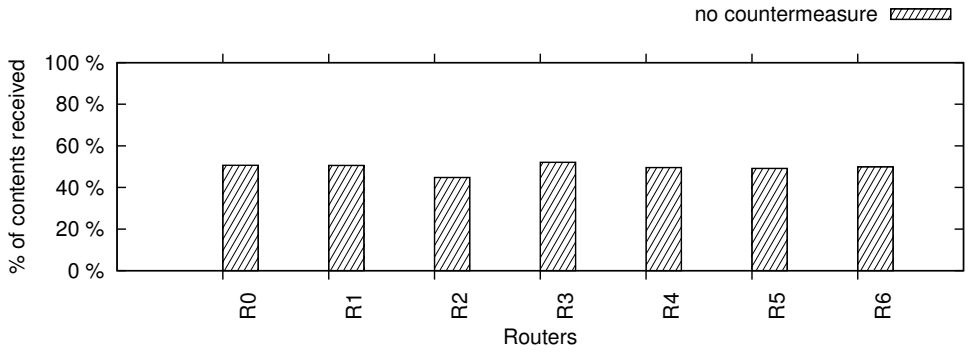
Results of the attack are plotted in Figure 4.4 (for simple architecture) and Figure 4.5 (for DFN). In particular, PIT usage are shown when fake interests are generated every 5.350 ms (figures 4.6) and 4.11, 2.675 ms (figures 4.7 and 4.12) and 1.337 ms (figures 4.6 and 4.13).

Figure 4.4 shows the percentage of content packets forwarded by the routers when an IFA is performed on the simple topology. Results are shown in percentage with respect to the simple topology baseline with no *adversary*. We can observe how IFA attack is more devastating with a higher rate of fake interests generation. In particular we can see really evident effects with a rate, for generating fake interests, 4 and 8 times higher than the rate used to require existing contents. While in Figure 4.4(a) we can note that 80% of contents are forwarded, in Figure 4.4(b) contents forwarded from routers drop to 50% and in Figure 4.4(c) fall down to 20% and 30% of the original traffic. This suggest that increasing the rate of the generation of fake interests improves the damaged that IFA does.

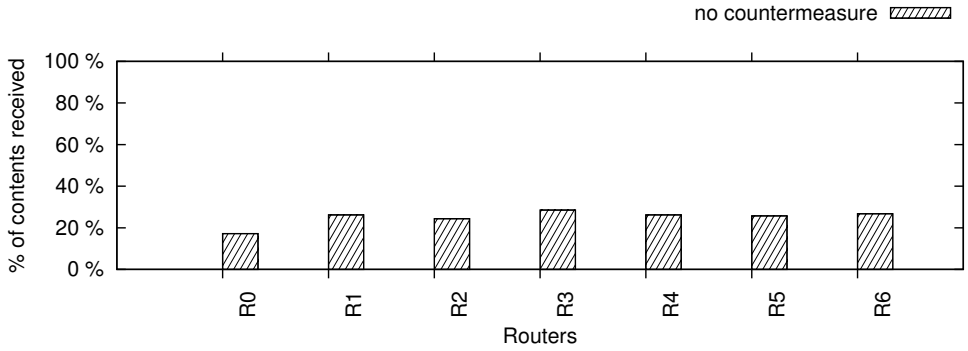
Figure 4.5 shows the impact of IFA in the DFN topology. It is evident how impact of IFA is less pronounced in the DFN architecture. Figure 4.5(a) shows how with double rate attack produces a not so important effects. Figures 4.5(b) and 4.5(c) reveal a more noticeable consequences (in some routers contents returned fall to 25 %). It is also interesting to note how, routers not directly involved in the attack (e.g. `R26` and `R27`), are less influenced than the correspondent router, `R5`, in simple architecture. However, also in this architecture we note that increasing the rate of the generation on fake interests the attack is more efficient.



(a) *Adversary*'s rate 2x

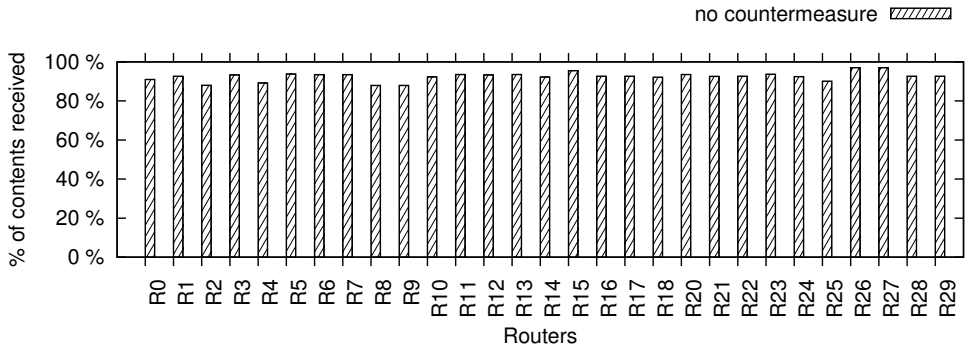


(b) *Adversary*'s rate 4x

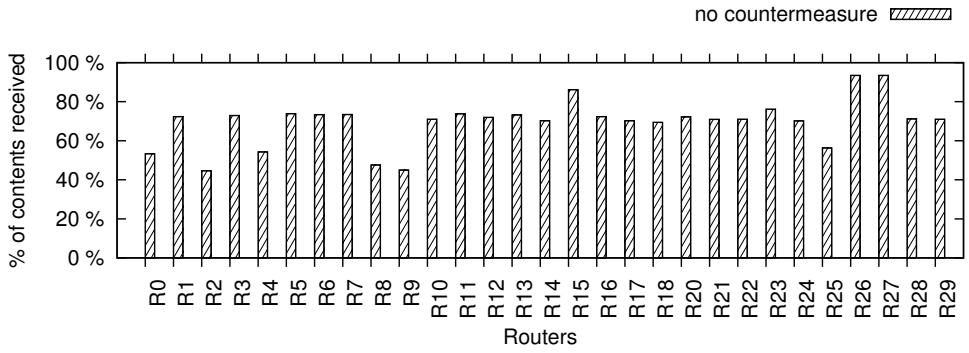


(c) *Adversary*'s rate 8x

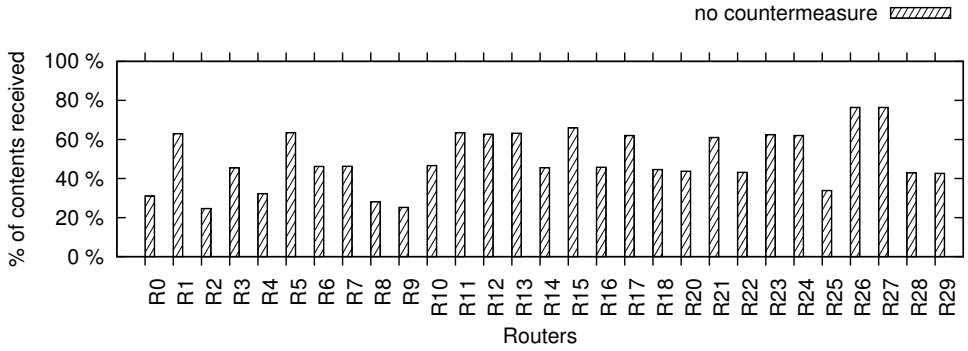
Figure 4.4: IFA: throughput relative to baseline simple topology (percentage)



(a) *Adversary*'s rate 2x



(b) *Adversary*'s rate 4x



(c) *Adversary*'s rate 8x

Figure 4.5: IFA: throughput relative to baseline DFN (percentage)

It is now interesting to see what are the effect IFA produces in the PIT of each router. Figures 4.6, 4.7, 4.8 and 4.11, 4.12, 4.13 show the PIT occupation over the time. The first refers to routers in the simple topology while the second represents

PIT occupation of routers of DFN topology. As indicated, figures 4.6 and 4.11 report results for the scenario with the rate of generation of fake interests equals to 5.350 ms, figures 4.7 and 4.12 refer to the scenario with a rate of generation of fake interests equals to 2.675 ms while Figures 4.4(c) and 4.13 refer to rate equals to 1.337 ms. These results are very interesting and they need an explanation for the behavior shown.

Looking at Figure 4.6 we can see how no one PIT reaches it bound. Router `R4` is very close to have a PIT completely filled, in fact it reaches all interests generated (good and fake). Also `R5` is very close to have its PIT completely filled, but it remains always under the `R4` curve. While `R5` receives only interests directed to `P0` (all fake interests and a half of good interests are directed to `P0`), `R4` receives all (fake and good) interests generated. This means that the difference of number of interests stored in PIT from `R4` and `R5` are exactly the interests directed to `P1` and not yet satisfied. However, what Figure 4.6 suggest is that fake interests and honest interest are not enough to completely fill a PIT.

Looking at Figure 4.7 we can see how, it this case, the PIT of router `R4`, `R0` and `R5` reach their bound. This means that using a rate for generating fake interests equals to 2.675 ms we are able to generate enough interest to fill the PIT of `R0` and so for sure we fill PIT of `R4` and `R5`. So this rate is sufficient to produce some visible effect in the network.

Comparing curves of `R3` in Figure 4.6 and 4.7 we can see how, in the second case, PIT of `R3` contains more interests than in the first case. We have to remember that `R3` is not directly involved in the attack, it does not receive any fake interest, and this means that IFA produces effects on router not directed involved in the attack. It can be explained as follows: R3 is connected to R4, which is on a path involved in the attack; therefore, some interests forwarded by R3 are dropped by R4. Hence a number of interests in the PIT of R3 are never satisfied, even though technically none of them are fake.

Going further we can observe Figure 4.8. At first glance, it seems that effects of IFA in this scenario are less pronounced than effects in Figure 4.7. In Figure 4.8 PIT of `R5` is, on the average, less filled than in Figure 4.7. In addition, with rate generation of fake interests equals to 1.337 ms, curve of `R5` achieves a step trend that it does not have in the Figure 4.6. Actually this behavior regards almost all the routers, not only `R5` and it is generated from a combination of factors. However, if we understand the motivation of this phenomenon for `R5` we can apply it to other routers. To understand this phenomenon we have investigate on what happens in the PIT of `R4`.

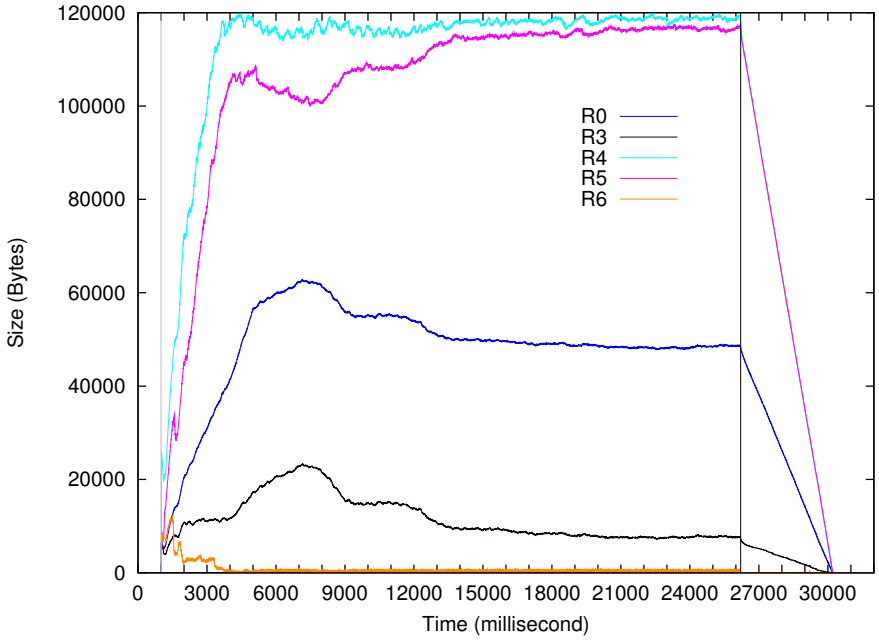


Figure 4.6: IFA: PIT usage in simple topology. *Adversary*'s rate 2x

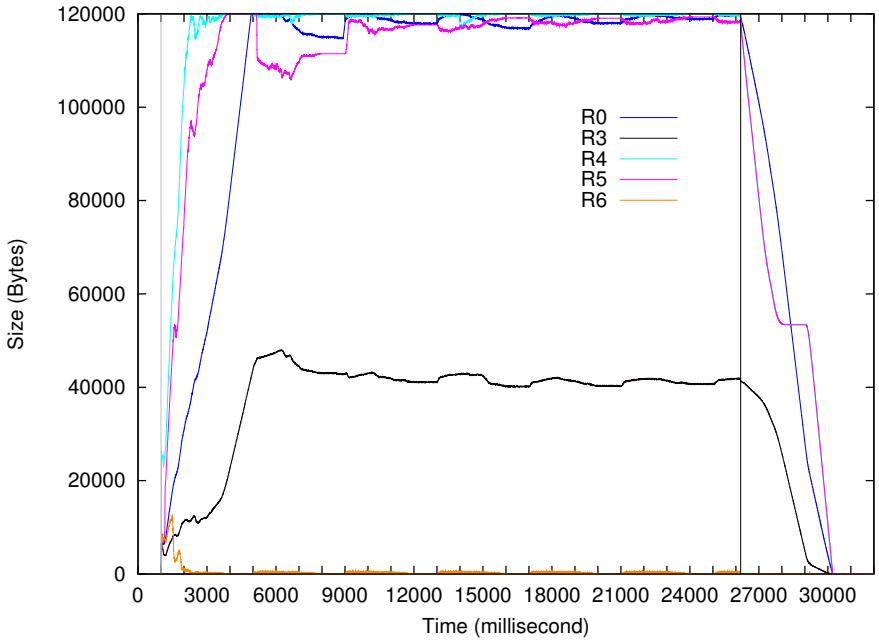


Figure 4.7: IFA: PIT usage in simple topology. *Adversary*'s rate 4x

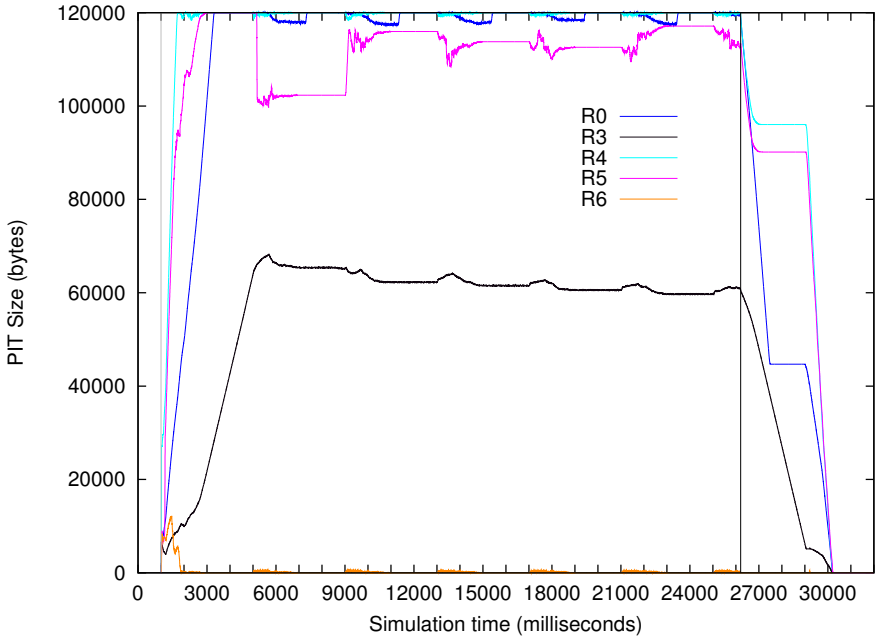


Figure 4.8: IFA: PIT usage in simple topology. *Adversary*'s rate 8x

Looking at simple topology (Figure 4.1) we see that `R4` receives interests from routers `R0`, `R1` `R2` `R3`. As we can see in Figure 4.8, about at time 3,000 ms routers `R0`, `R1`, `R2` has its PIT completely filled. This means that, starting from that time, `R0`, `R1` and `R2`, do not forward any other interest to `R4`. To start to forward interests again, they have to wait that some interest in their PIT is flushed. This happens only when interests are satisfies or they expires their life time. So, what we expect, is that starting from some time, the sending of interest is not continuously but intermittent. We can observe this phenomenon by looking in the Figure 4.9 where we can see that interests arrives in burst to `R4`. This is exactly the phenomenon we have just describe. When the PIT of a router is completely filled, the router stops to send interests until some space in their PIT is freed. At this point we expect that, when routers `R0`, `R1`, `R2` stop sending interests, the usage of PIT of `R4` decrease. This should happen because PIT of `R4` does not contain only fake interests but also good interests that can be satisfied. However, it does not happen and the motivation is explained in Figure 4.9. In this chart we can see that router `R3` never stops to send interests to `R4`. In fact PIT of `R3` is never filled so `R3` does not suffer of the same problem of `R0`, `R1`, `R2` so all interests coming from consumers are continuously forwarded to `R4` and that is why usage of PIT of `R4` does not decrease.

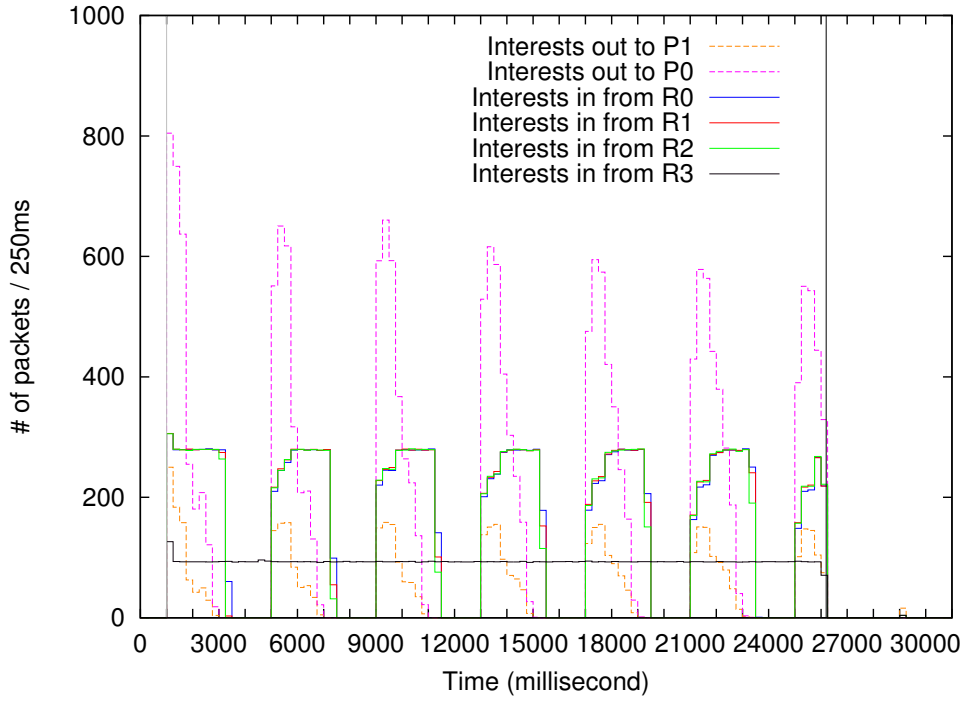


Figure 4.9: IFA: R4 throughput in simple topology

Considering R5, we can see in Figure 4.8 how its PIT is not always filled and PIT usage follow a step trend. This behavior is due to R4 that acts as a filter for R5. In fact, when PIT of R4 is completely filled it starts to drop new arrived interests and so it does not forward any other interest to R5 and R6. Because of R5 only receives interests from R4, if R4 stops to send interests and PIT of R5 is not filled, there is not any other way to fill it. In fact this is what happens to R5 at time 5,000 ms in the Figure 4.8. At time 5,000 ms interests start to expire their life time and this involved in a fall of PIT usage. At that time, really a bit before, also in R4 interests start to expire their time life, so R4 starts to send interests to R5. So, interests expired in R5 are immediately replaced by interest coming from R4 and the result is the little fluctuation we see in the first second of each step. It is also interesting to note how interests coming from R4 are not able to fill the PIT of R5. This is due to the fact that R4 starts to send interests before interests in R5 expire and also to the fact that not all the interest that R4 forwards are directed to R4. In fact, a little part of them are interests forwarded to R6.

Moving focus from R5 to R6 we can see how attack influences R6 too. Looking the curve of R6 in Figures 4.6, 4.7, 4.8 we can see how the number of interests in its PIT decreases. As for R3, influence to R6, is due to the directly connection between with

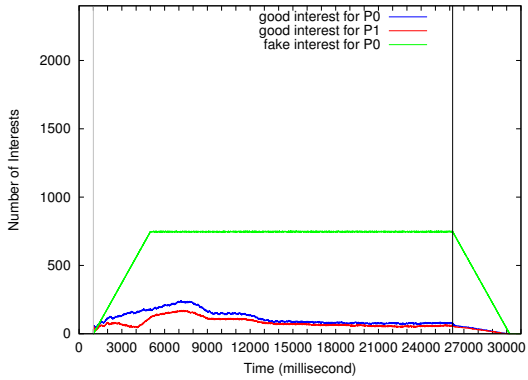
R4 which is a single point of failure for the network. When its PIT is completely filled no one interest is accepted from R4 (and sent out) and so some of the interests directed to R6 are dropped by R4.

Figure 4.10 illustrates the number of interest stored in PIT categorized in fake interests and good interest directed to P0 and P1. Figures 4.10(a), 4.10(c), 4.10(e) show how increasing the rate of generation of fake interest, the number of fake interests stored in PIT of R0 augment.

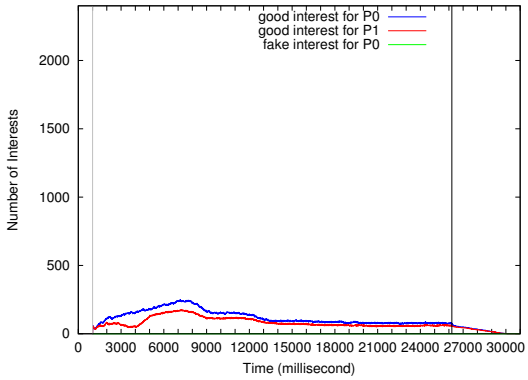
Instead, if we consider good interests, we can see how between figure 4.10(a) and figure 4.10(c) there is an incrementation in the number of good interests stored in PIT. This increasing is due to the effect of IFA in the neighbor routers, in particular R4. The high number of interest that R4 receives leads R4 to drop some interest coming from R0 and, in fact, these interests remain in the PIT of R0 until their life time expires increasing PIT occupation.

Comparing figure 4.10(c) and figure 4.10(e) we can see a reverse trend. Increasing the rate of generation of fake interests decrease the number of good interests in PIT. In this case, the guilty of this trends has to be looked for inside R0. A rate equals to 1.337 ms produces more fake interests per second than a rate equals to 2.675 ms and so we need less time to reach the PIT occupation bound. Having less time to reach bound lets a router to accept less good interests and this produces the result in Figure 4.10(e). In fact, we can observe this phenomenon looking the green line in Figure 4.10(e) and Figure 4.10(e). In the first chart the green curve has an initial slight slope than green curve in the second and adding the number of interests in PIT we can see that with a rate equals to 2.675 ms bound is reached between time 4,000 ms and 5,000 ms while with a rate equals to 1.337 ms the bound of PIT is reached about at time 3,000 ms.

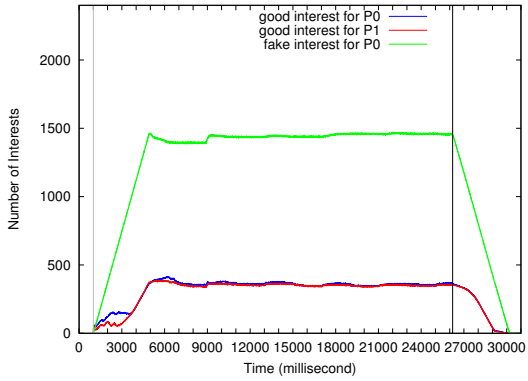
Figures 4.10(b), 4.10(d), 4.10(f) represent the number of good interest in router R3. In this case we can see an unique trend, increasing the rate of generation of fake interest, increase the number of interests in PIT. In fact R3 does not receive fake interests and so it only suffers for effect of IFA deriving from neighbor routers; in this case R4.



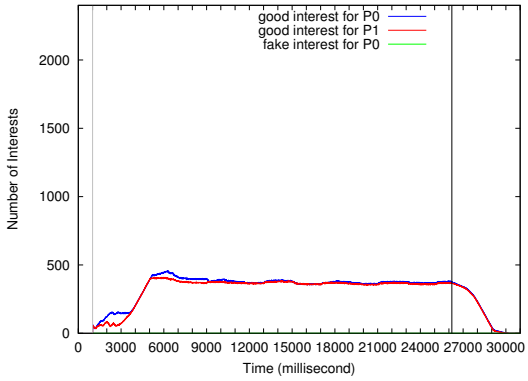
(a) PIT usage of `R0`: *adversary*'s rate 2x



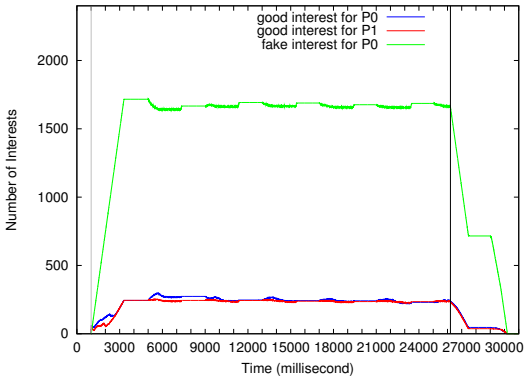
(b) PIT usage of `R3`: *adversary*'s rate 2x



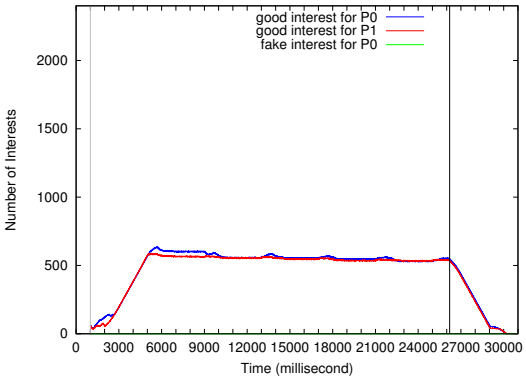
(c) PIT usage of `R0`: *adversary*'s rate 4x



(d) PIT usage of `R3`: *adversary*'s rate 4x



(e) PIT usage of `R0`: *adversary*'s rate 8x



(f) PIT usage of `R3`: *adversary*'s rate 8x

Figure 4.10: IFA: interests in PIT of simple topology

Figures 4.11, 4.12, 4.13 show results for PIT usage of routers of DFN. Phenomenon we have described for simple topology are presents also in DFN. As describe in Table 4.1 we can define an association, between some router of simple topology and DFN, basing on routers similarities. In fact, this similarities reflect also in Figures 4.6, 4.7, 4.8 and 4.11, 4.12, 4.13 and lead to match consideration we have done for simple topology in the DFN. We do not present other charts for DFN because all relevant aspect of attack has been presented in the description of simple topology.

Results we have proposed show that increasing the rate of the attack improve the malicious effects in the network. However, results also suggest that increasing the rate of the fake interests leads to have the PIT of routers connected to the producer less filled. This suggest that an attack with more *adversary* with a lower rate of interest sending could probably be more devastating for router connected to producers. We leave a further investigation for a future work. In the next sections countermeasure are run in top of the scenario where rate of fake interest generation is equals to 1.337 ms.

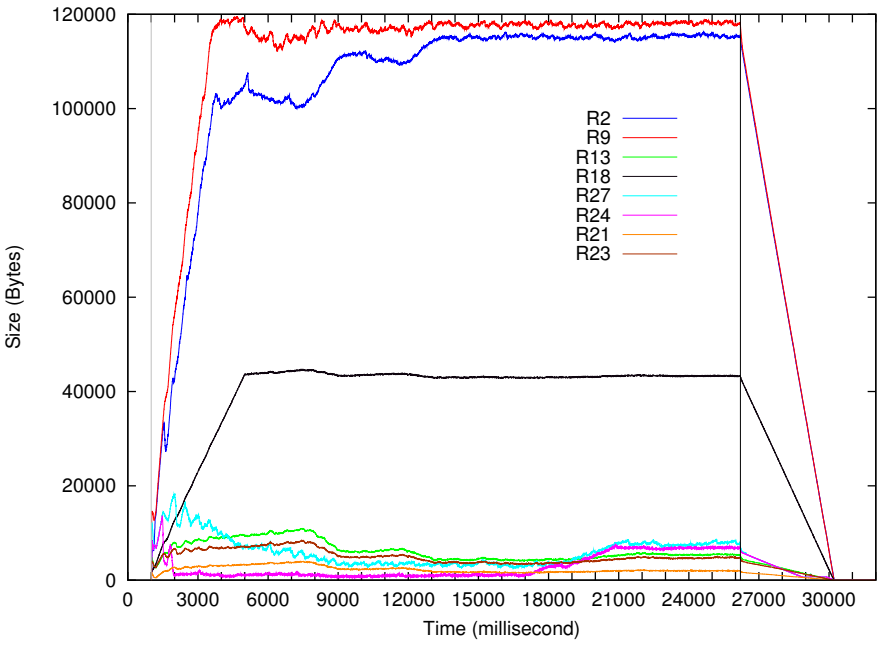


Figure 4.11: IFA: PIT usage in DFN. *Adversary*'s rate 2x

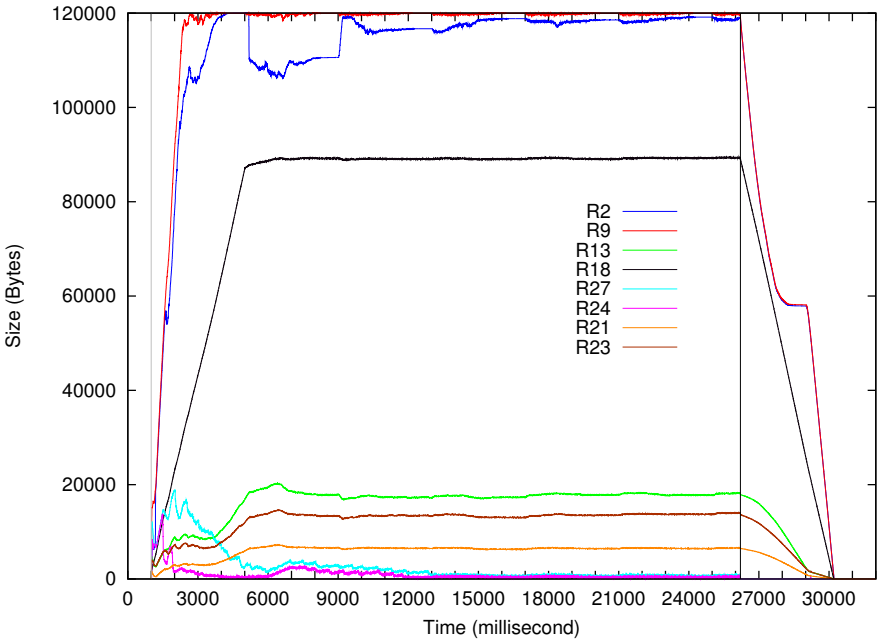


Figure 4.12: IFA: PIT usage in DFN. *Adversary*'s rate 4x

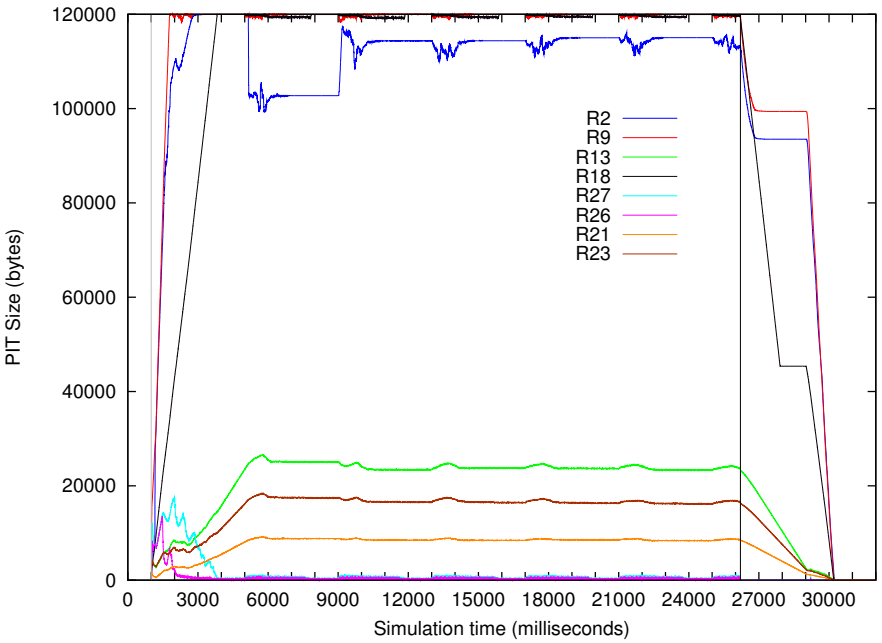


Figure 4.13: IFA: PIT usage in DFN. *Adversary*'s rate 8x

Chapter 5

Our Countermeasure: Poseidon

As mentioned in Chapter 4, the primary goal of an IFA is to make a node unavailable. In fact, this is the main purpose of a DoS attack; in our case, because IFA uses as more as possible zombies we refers to DDoS attack.

In the literature there is a large body of research on DDoS attacks [31; 33; 40] but, because of we are considering a new paradigm of communication, we can not directly apply existing solutions. However, some peculiarity of IFA is reflected in the DDoS attacks conducted to the TCP/IP. Knowing existing solutions allow us to get some working ideas so that we can create a more robust and incisive countermeasure.

When a DDoS attack is performed, for example in a router, we can identify two types of traffic directed to the attacked router. Malicious traffic that are generated from adversary and honest traffic generated from common users. A good countermeasure to DDoS attack should block the former and satisfy the latter as quickly as possible. To do this, the first step to thwart a DDoS attack is detect when an attack is in progress while the second step consist of blocking only malicious traffic to reduce the effect of the attack.

Detection of DDoS attack on the current internet structure can be performed in different ways. It is possible to use database of known signatures by recognizing anomalies in system behaviors or using third party. Signature based approach ([38], [42]) employs a priori knowledge of attack signatures. The signatures are manually constructed by security experts analyzing previous attacks and used to match with incoming traffic to detect intrusions. SNORT [42]) and Bro [38] are the two widely used signature based detection approaches.

Anomaly detection [4; 28; 30; 34] relies on detecting behaviors that are abnormal with respect to some normal standard. Detecting DDoS attacks involves first knowing normal behavior of our system and then to find deviations from that behavior. Mirkovic et al. [30] proposed D-WARD defense system that does DDoS attack de-

tection at source, based on the idea that DDoS attacks should be stopped as close to the source as possible. Cheng et al. [34] proposed to use spectral analysis to identify DoS attack flows. Lee and Stolfo [28] used data mining techniques to discover patterns of system features that describe program and user behavior and implement a classifier that can recognize anomalies and intrusions. Bencsath et al. [4] have given a volume based approach, in which incoming traffic is monitored continuously and dangerous traffic intensity rises are detected. This approach is suitable for detecting high rate attacks, but ineffective to detect low rate degrading attacks. Mechanisms that deploy third-party detection do not handle the detection process themselves, but rely on an external third party that signals the occurrence of the attack [31].

We now present our solution. Section 5.1 briefly introduce Poseidon while in the sections 5.2 and 5.3 we describe in details the detection phase and reaction phase.

5.1 Overview

In this section we introduce Poseidon, our approach for mitigating Interest flooding. Poseidon is a set of algorithms that run on routers with the goal of identifying traffic anomalies and react to mitigate their effects.

The approach, that Poseidon uses to identify traffic anomalies, is based on the evaluation of some network parameter. Since Poseidon has to run on NDN we have to investigate on what information we can rely on to identify anomalies. Recalling Section 3.3 we know that NDN packets do not transport any information about the source of an interest. This loss of information can affect the identification of anomalies but, it affects the identification of the source of the attack. While in the TCP/IP we can use source information to identify the source of the attack and so build a better countermeasure, in NDN we can not leverage on this. Unfortunately, being not able to identify source of requests means to lose an efficient way to discriminate correctly between malicious requests and honest requests. If we can not use source to discriminate we can refer to the destination (content referred) of the interest. Also using the destination as discriminant factor does not help us. In fact, a good IFA instantiation uses interests that require non-existing content of existing producer. While a router could detect request for non-existing router, it can not for sure detect requests for non-existing content. This kind of detection requires that a router knows the existence of every content in the network but equally it can have problems with contents generated dynamically.

Because interest does not contain information on the user that has created it, Poseidon uses interfaces of a node to discriminate the origin of an interest and metrics, used to detect attacks, are related to interface. In fact, it continuously

monitors per-interface rates of unsatisfied interests with respect to the overall traffic. If these rates change significantly between two consecutive time intervals, Poseidon sets a filter on the offending interface(s) (which reduces the number of incoming interests). Additionally, Poseidon can issue a push-back "alert" message to the offending interfaces, to signal that an IFA is in progress. Upon receiving an alert message, Poseidon reacts by altering its IFA detection thresholds. Once an IFA is detected, Reaction Phase comes into action and interests are dropped in accordance with some rule.

Poseidon keeps several statistics on expired interests. In particular, for each of them it records namespace and incoming/outgoing interfaces information. These statistics are periodically consolidated: only the most significant values (i.e., corresponding to the few largest cluster of expired interests) are saved.

Relatively common network phenomenon (e.g., packet loss) and regular applications behavior usually account for only a (relatively) small amount of expiring interests in routers' PITs. In order to limit false positive in attack detection and minimize the reaction time, Poseidon monitors multiple metrics for the detection of IFA.

In the next sections we present the detection and reaction phases of Poseidon. Notation used is shown in Table 5.1.

R	set of all routers in the network running Poseidon
r_i	i-th router, $1 \leq i \leq \lvert\mathsf{R}\rvert$
r_i^j	j-th interface on router r_i
t_k	k-th time interval
$\omega(r_i^j, t_k)$	rate between incoming interest and outgoing content for a given interface r_i^j
$\rho(r_i^j, t_k)$	PIT space used by interests arrived on interface r_i^j, measured at the end of interval t_k
$\Omega(r_i^j)$	IFA detection threshold for $\omega(r_i^j, t_k)$
$\mathrm{P}(r_i^j)$	IFA detection threshold for $\rho(r_i^j, t_k)$

Table 5.1: Notation.

5.2 Detection Phase

Attacks are detected using two parameters: $\omega(r_i^j, t_k)$, and $\rho(r_i^j, t_k)$. The former represents 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:

$$\omega(r_i^j, t_k) = \frac{(\#\text{ of interests from } r_i^j \text{ at interval } t_k)}{(\#\text{ of content packets to } r_i^j \text{ at interval } t_k)}.$$

The latter indicates the amount of space (in bytes) used to store interests in PIT, coming from interface r_i^j within time interval t_k.

Poseidon detects an attack when both $\omega(r_i^j, t_k)$ and $\rho(r_i^j, t_k)$ exceed their respective thresholds $\Omega(r_i^j)$ and $\mathrm{P}(r_i^j)$. The detection algorithm (Algorithm 1) is executed at fixed time intervals – typically every 60 ms – and in the presence of particular events, as detailed below.

Algorithm 1: AttackDetection

input : $\omega(r_i^j, t_k)$; $\Omega(r_i^j)$; $\rho(r_i^j, t_k)$; $\mathrm{P}(r_i^j)$
output: boolean
1: **if** $\omega(r_i^j, t_k) > \Omega(r_i^j)$ **and** $\rho(r_i^j, t_k) > \mathrm{P}(r_i^j)$ **then**
2: output true
3: **else**
4: output false
5: **end if**

The parameter $\omega(r_i^j, t_k)$ is a good representation of the ability of routers to satisfy incoming interests, in a particular time interval. This is also confirmed by our experiments, detailed in Chapter 6. In particular, $\omega(r_i^j, t_k) > 1$ indicates that the number of content packets forwarded to r_i^j is smaller than the number of interests coming from the same interface. However, a small bursts of (either regular or non-satisfiable) interests may not be caused by an attack. Hence, taking into account only $\omega(r_i^j, t_k)$ can cause the detection algorithm to report a large number of false positives. Engaging in a countermeasure may, in this case, produce only negative effects to the overall performance of the network.

We argue that neither increasing $\Omega(r_i^j)$, nor computing $\omega(r_i^j, t_k)$ over longer intervals, produces the indented effects in solving this problem. In fact, in the first case the bound must be set high enough to avoid classification of short burst of interests as an attacks; however this could inevitably lead to late or mis-detection of real attacks. Increasing the size of the interval over which $\Omega(r_i^j)$ is computed may reduce the sensitivity of Poseidon to short burst of interests. An interval length similar or longer than the average round-trip time of interest/content packet, in fact, may allow (part of) the content requested by the burst to be forwarded back, reducing $\omega(r_i^j, t_k)$ to a value closer to 1. However this could significantly increase the detection time in case of attack.

Instead, to improve detection accuracy (distinguishing naturally occurring burst of interests from attacks), Poseidon takes into account also $\rho(r_i^j, t_k)$. This value measures the PIT space used by interests coming from a particular interface. This allows Poseidon to maintain the number of false positives low – when compared to considering solely $\omega(r_i^j, t_k)$ – while allowing it to detect low-rate IFA. In a low rate IFA, the *adversary* limits the rate of fake interests to keep $\omega(r_i^j, t_k)$ below its thresholds. Monitoring the content of the PIT allows Poseidon to observe the *effects* of the attack, rather than just its *causes*.

To sum up, different parameters monitored by Poseidon act as weights and counterweights for IFA detection. When a router is unable to satisfy incoming interests for short amounts of time, $\rho(r_i^j, t_k)$ may exceed a preset limit, but $\omega(r_i^j, t_k)$ will not; when it receives a short bursts of interests, $\omega(r_i^j, t_k)$ may become bigger than $\Omega(r_i^j)$ but the PIT usage will likely be within correct parameters. To stay undetected, an *adversary* willing to perform IFA must therefore: (1) reduce the rate at which it sends interests, which limits the effects of the attack; and/or (2) restrict the attack to short burst, which makes the attack fairly ineffective.

5.3 Reaction Phase

Poseidon has been designed with two different reaction phases that can be used. In the first one, Section 5.3.1 routers do not collaborate in the detection of an IFA; in the second reaction phase, Section 5.3.2, routers collaborate to try to improve the detection of IFA.

5.3.1 Rate-Based (Local) Countermeasure

Once an IFA from interface r_i^j of router r_i has been identified, Poseidon limits the rate of incoming interests from that interface. The original rate is restored once all detection parameters fall again below their corresponding thresholds.

5.3.2 Push-Back (Collaborative) Countermeasure

When collaborative countermeasures are in place, once a router detects adversarial traffic from a set of interfaces it limits their rate and issues an alert message on each of them. An alert message is an unsolicited content packet belonging a reserved namespace ("`/pushback/alerts/`" in our implementation), used to convey information about IFA in progress.

There are two reasons for using content packets rather than interests for carrying push-back information: (1) during an attack, the PIT of the next hop connected

to the offending interface may be full, and therefore the alert message may be discarded; and (2) content packets are signed, while interests are not – this allows routers receiving alert messages to determine whether the information they carry is legitimate.

Routers running Poseidon do not process alert messages as regular content: alerts are not checked against PIT content and are not forwarded any further. The payload of an alert packet contains: the timestamp corresponding to the alert generation time; the new (reduced) rate at which offending interests will be accepted on the incoming interface; and detailed information about the attack – such as the namespaces used referred in malicious interests.

Router r_i receiving a message msg processes it as follows (and as detailed in Algorithm 2):

1. If msg is a content packet, r_i processes msg according to the NDN protocol (see Section 3.3).

2. If msg is an alert message, r_i checks its signature and discards it if verification fails; if the message is fresh (i.e., it has been signed within the last few seconds – Line 6 of Algorithm 2) and r_i has not received an alert from the same interface within $wait\_time$, the detection thresholds are decreased according to a scaling parameter s.

3. If msg is an interest, r_i checks whether there is an attack in progress (Line 17). If so, the interest is dropped and an alert is sent to the router connected on interface r_i^j – unless similar alert has been sent within $alarm\_duration$ (Line 19); otherwise r_i process the interest according to NDN protocol (see Section 3.3).

A persistent IFA on router r_i causes it to send multiple alert messages towards the source(s) of the attack. Such sources will decrease their thresholds $\Omega(r_i^j)$ and $\mathrm{P}(r_i^j)$ until they detect the attack and implement rate-limiting on the malicious interests.

This push-back mechanism allows routers that are not the target of the attack, but are unwittingly forwarding malicious interests, to detect the attack early. Alert messages allow routers to detect IFA even when they are far away from the intended victim – i.e., close to nodes controlled by the *adversary*, where countermeasures are most effective.

Routers periodically check whether $\omega(r_i^j, t_k)$ and $\rho(r_i^j, t_k)$ are below their threshold. If they are – and the router has not received alert messages on the corresponding interfaces for longer than $wait\_time$ – then thresholds $\Omega(r_i^j)$ and $\mathrm{P}(r_i^j)$ are increased.

Algorithm 2: MessageProcessing

input : Incoming packet msg from r_i^j; $alarm\_duration$
$\Omega(r_i^j)$; $\mathrm{P}(r_i^j)$; Scaling factor s;
Alert message m from interface r_i^j

1: **if** msg is ContentObject **then**
2: process msg as ContentObject
3: return
4: **end if**
5: **if** msg is AlertMessage **then**
6: **if** Verify($msg.signature$) **and** IsFresh(msg) **and**
time from last Alert received from $r_i^j > wait\_time$ **then**
7: `// Push-back reaction`
8: Decrease($\Omega(r_i^j), s$)
9: Decrease($\mathrm{P}(r_i^j), s$)
10: **else**
11: drop msg
12: return
13: **end if**
14: **end if**
15: **if** msg is Interest **then**
16: `// AttackDetection is shown in Algorithm 1`
17: **if** AttackDetection **then**
18: drop msg
19: **if** time from last Alert sent on interface $r_i^j > wait\_time$ **then**
20: `// Push-back alert message generation`
21: send Alert to r_i^j
22: **end if**
23: **else**
24: process msg as Interest
25: **end if**
26: **end if**

Chapter 6

Evaluation

In this section we report on experimental evaluation of countermeasures presented in Section 5. Our countermeasures are tested over the same topologies used in previous experiments and detailed in Figure 4.1. Each router implements detection techniques from Section 5.2 (Algorithm 1) and countermeasures from Section 5.3.

As for the parameters used in our experiments, we considered these initial values: for each router r_i, interface r_i^j, $\Omega(r_i^j) = 3$ and $\mathrm{P}(r_i^j) = 1/8$ of the PIT size. Furthermore, we set scaling factor $s = 2$ and *wait_time* (both used in Algorithm 2) to 60 ms.

The Decrease function divides the threshold in input by s at each invocation. Similarly, the Increase function increases its input by 1/8 of the current value.

Consumers request the same content at the same rate as in the previous simulations. Similarly, the nodes controlled by the *adversary* implement IFA as in the simulation in Section 4.4.1. We have used the worst scenario (*adversary* sends a fake interest every 1.337 ms form every controlled consumer) to test Poseidon. In the next two sections we present the results with the local countermeasure (Section 6.1) and the distributed countermeasure (Section 6.2)

6.1 Local Countermeasures

Figures 6.1 and 6.2 shows the result of applying local countermeasures. Values shown represent the average of 20 executions.

Figure 6.1(a) and Figure 6.1(b) report the ratio of content packets received with respect to the scenario with no *adversary*. (For comparison purposes, in the same figure we also report the same value with no countermeasures in place.)

Our results show that the rate-based (local) countermeasure – while simple – is very effective. In the smaller topology the impact of the *adversary* is now very

limited: the attack only reduces the traffic by *at most* 10%. In contrast, without any countermeasure the *adversary* was able to reduce content traffic by about 80%.

The same countermeasure is also very effective in the DFN topology. Most routers, in fact, forward virtually the same amount of content they otherwise forward without attack – up from about 60% with no countermeasures. Moreover, with Poseidon no router forwarded less than 80% of the original traffic in our experiments.

Figures 6.2(a) and 6.2(b) report PIT usage over the same experiments. For clarity, we omit results for some of the routers. In particular in Figure 6.2(a), we do not report measurement for R1 and R2 – which are virtually identical to those of R0 (the three routers are connected to nodes behaving identically).

Figure 6.2(b) presents our results for the DFN architecture; we do not report measurements for nodes `R4`, `R29`, `R22`, `R25`, `R20`, `R14` – almost identical to those of `R18`. Results for routers `R3`, `R5`, `R6`, `R7`, `R8`, `R11`, `R12`, `R15`, `R17`, `R19`, `R26`, and `R28` are also not reported being very similar to those of `R13`, `R21`, and `R23`.

Our results also show that this countermeasure significantly reduces the PIT usage in presence of an *adversary*. The effects of the rate-based approach are evident at $t = 6000$ ms, when fake interests corresponding to the initial phase of the attack – those that triggered the detection – expire. If we investigate more deeply in the results, we can see that time needed by Poseidon to detect an attack is about of 600 ms in every router. In fact, when attack is detected, Poseidon stops to forward interests coming from faces where an attack has been detected. It has to be noted that, in our simulations, malicious users start the attack at the same time the honest users start to require existing contents. Because its design, Poseidon needs a short time to collect data before being able to measure the defined metrics. During this time IFA is able to fill PIT as shows in Figure 6.2(a). We are aware that, in this manner, we have no verification on what is the real detection time but this represent the worst case.

In Figure 6.2(a) we can see this phenomenon by looking `R0`, `R4` and `R5` PIT usage. Table 6.1 reports the average of the detection time in the router `R0`, `R4` and `R5`. We can see that, at indicated time, the curve of `R0`, `R4` and `R5` changes its slope. This happens because Poseidon has detected the attack and it starts to drop interests coming from the attacked faces.

Routers	**Detection point (ms)**
`R0`	1,630
`R4`	1,832
`R5`	1,854

Table 6.1: Detection time in simple topology

It is also interesting to note that, when Poseidon detects an attack, the occupation on PIT of router R0 and R5 is still incrementing. This is due to design of Poseidon; in fact, when interests coming from attacked faces are dropped, ω of that faces decreases. When ω goes under Ω, Poseidon restarts to forward interests and to store them in the PIT. But, because of interests coming from *adversary* arrivers with a high rate, ω quickly exceed Ω again leading Poseidon to stop to accept fake interests. This phenomenon produces the step trend in R0 and R5 we can see from detection time to time 6,000 ms.

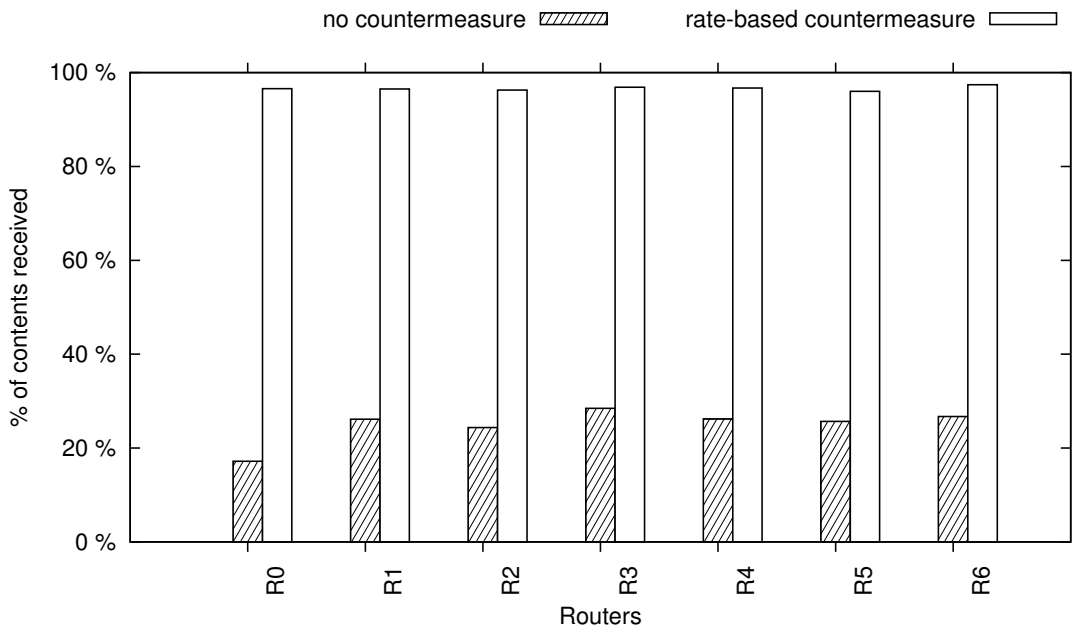
Regarding R4, Figure 6.2(a) shows that, after IFA is detected, its PIT usage has a fluctuating trend. R4 is connected to routers R0, R1, R2, R3, and Poseidon detects an attack on faces that connect R4 to R0, R1, R2. In this way, starting from detection time up to about 6,000 ms, Poseidon limits interests coming from that faces but accepts all interests coming from R3. The fluctuation is due to: (1) satisfied interests coming from R3, (2) satisfied interests coming from R0, R1, R2.

It should also be noted a different behavior between the PIT usage of R4 and the PIT usage of R5. Because of all interests that R5 receives comes from R4, we expect that if R5 occupation of PIT is increased, also R4 occupation of PIT is increased too. This does not occur because not all interests that R4 receives are forwarded to R5, in particular interests directed to P2 are forwarded to R6 not to R5. The gap between the occupation of PIT in R4 and R5 are exactly the interests directed to P2 and they produce this fluctuation in R4 curve.

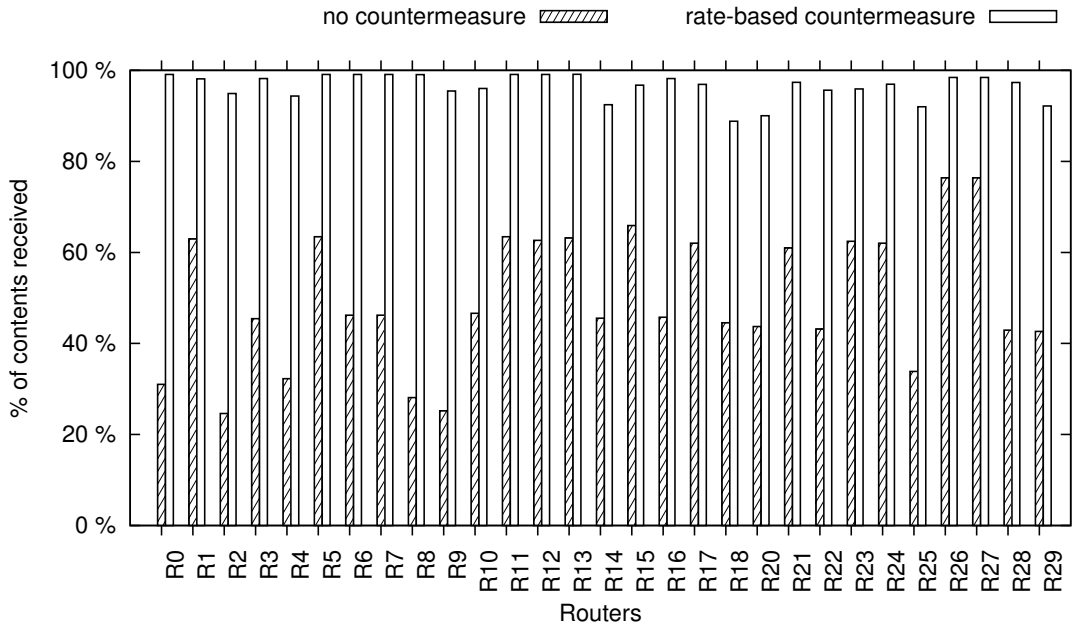
At time 6,000 ms all initial fake interests included, before the detection time, in the PIT are flushed because their life time is expired. From time 6,000 ms we can see the real effect of Poseidon in a running environment. In this case, we can note clear improvements in the occupation of PIT compared to the attack scenario (Figure 4.8). It is interesting to note an advantage of Poseidon also in R3. Because of R3 is not directly involved in the attack, Poseidon do no come into action in R3, but benefits produced in other routers involve in a collateral effect in R3.

Figure 6.2(a) reports PIT usage of routers of DFN. It is clear the correspondent between routers in the simple architecture and DFN architecture reported in Table 4.1. In fact, we can note the same phenomenon we have described for simple architecture in the DFN architecture. Every consideration is still valid in the DFN architecture.

So far we have considered cumulative results for throughput; however, it is interesting to analyze also how the content packets throughput varies over time in different scenarios. Figures 6.3 and 6.4 show the effect of our rate-based countermeasure on some routers, which we deem notable in our topologies. This figures clearly shows that the local (rate-based) countermeasure is able to guarantee roughly the same throughput measured without IFA.

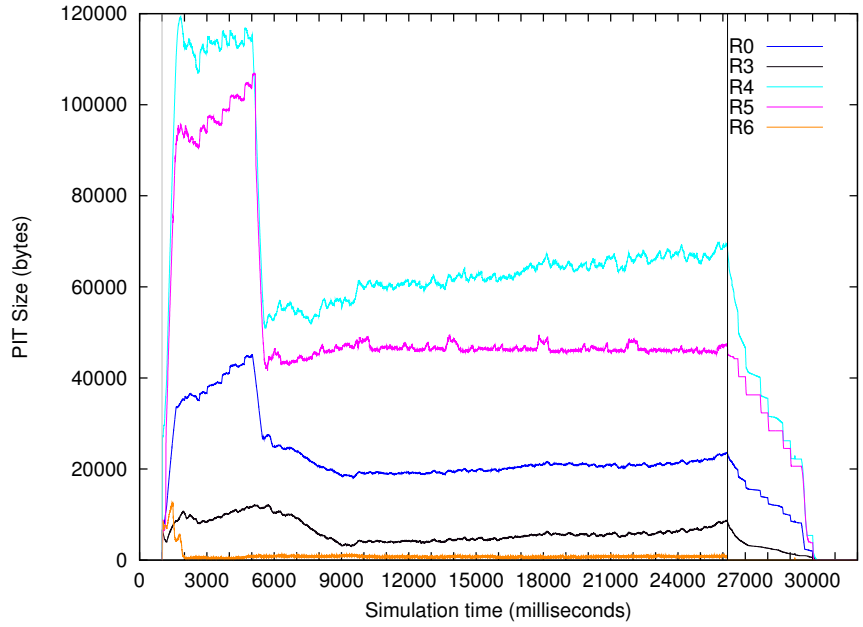


(a) Simple architecture

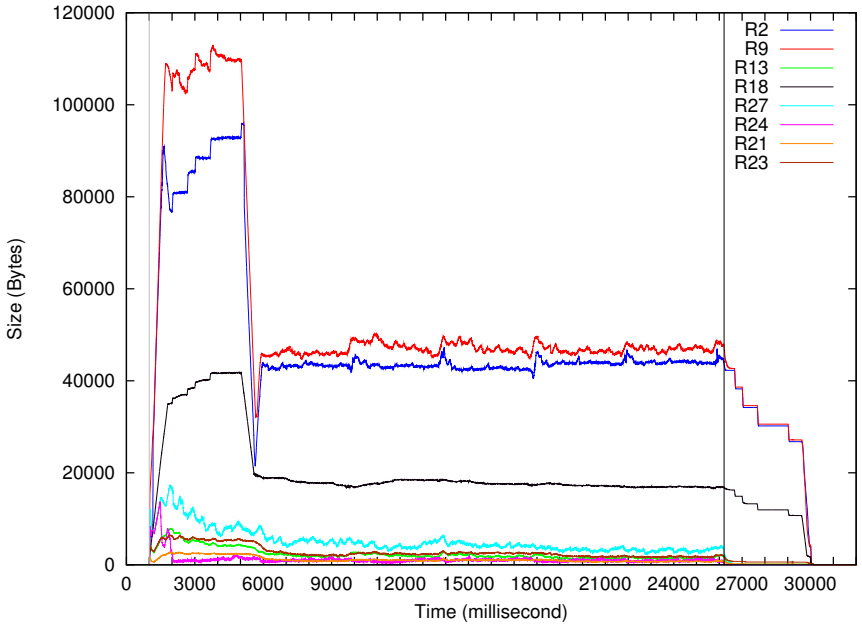


(b) DFN architecture

Figure 6.1: Rate-based (local) countermeasure: throughput relative to baseline (percentage)



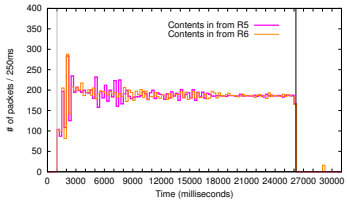
(a) Simple architecture



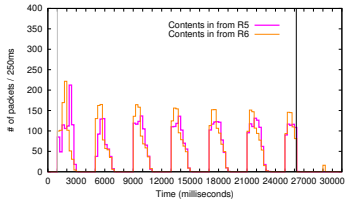
(b) DFN architecture

Figure 6.2: Rate-based (local) countermeasure: PIT usage

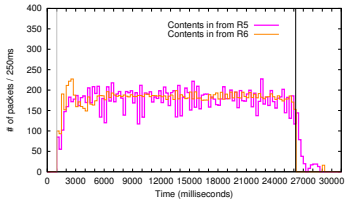
An interesting phenomenon highlighted by figures 6.3(b), 6.3(h) and 6.3(n), is the cyclic behavior of the bandwidth available to content in routers `R4`, `R0`, and `R5`, respectively. This pattern can be explained as follows. As soon as the PITs of these routers are filled up with fake interests, no legitimate interests are forwarded and therefore no content is routed back. After four seconds – i.e., the lifetime of an unsatisfied interest in our setting – fake interests are removed from the PITs, allowing routers to forward new (legitimate) requests for content. However, the *adversary* is quickly able to fill up PITs again. This process continues indefinitely for the whole duration of the attack. The same phenomenon can be noted in the DFN topology in Figure 6.4.



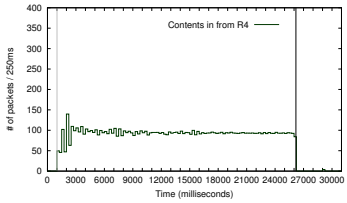
(a) R4: baseline



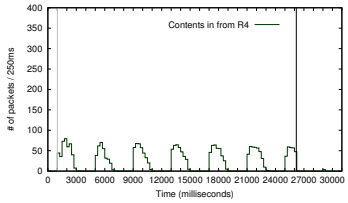
(b) R4: attack



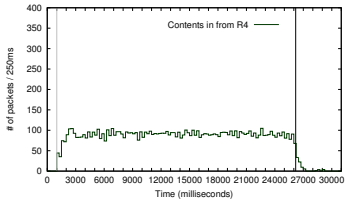
(c) R4: rate-based countermeasure



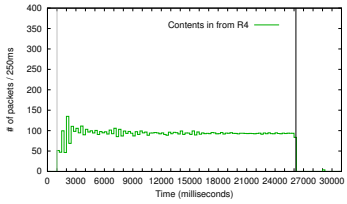
(d) R0: baseline



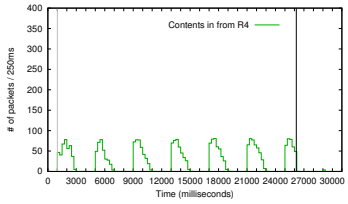
(e) R0: attack



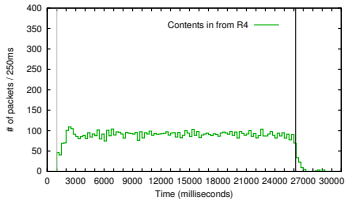
(f) R0: rate-based countermeasure



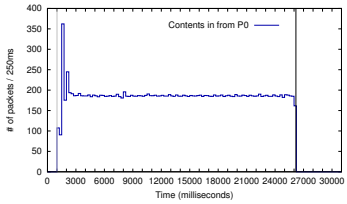
(g) R3: baseline



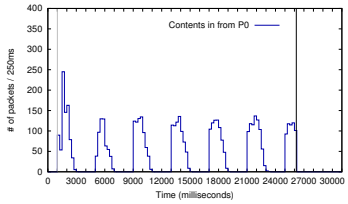
(h) R3: attack



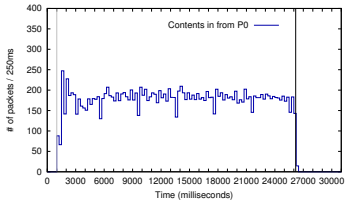
(i) R3: rate-based countermeasure



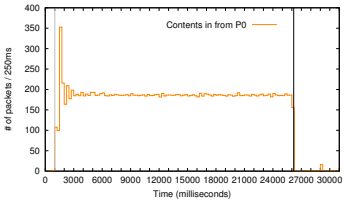
(j) R5: baseline



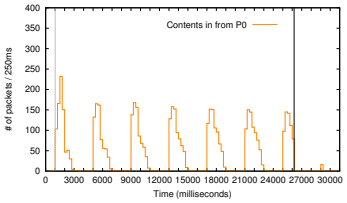
(k) R5: attack



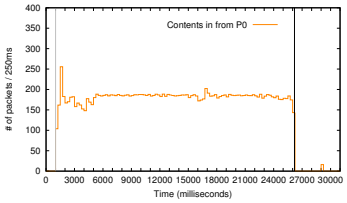
(l) R5: rate-based countermeasure



(m) R6: baseline

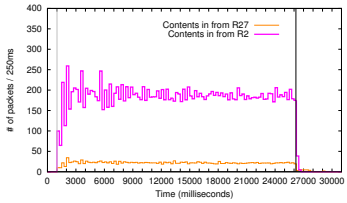


(n) R6: attack

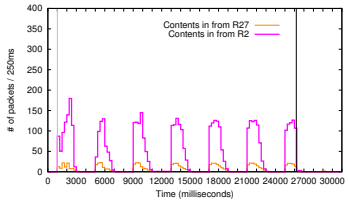


(o) R6: rate-based countermeasure

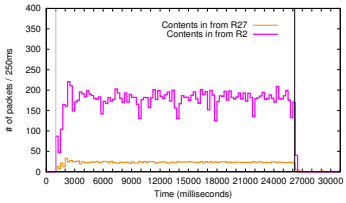
Figure 6.3: Simple architecture, representative routers: content throughput (absolute values)



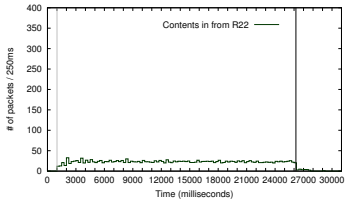
(a) R9: baseline



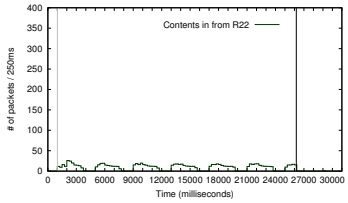
(b) R9: attack



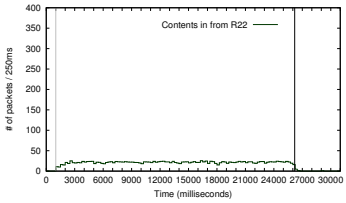
(c) R9: rate-based countermeasure



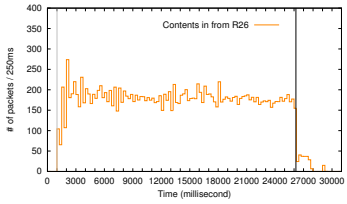
(d) R29: baseline



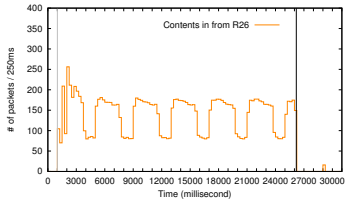
(e) R29: attack



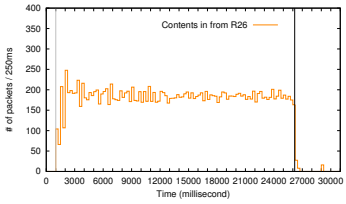
(f) R29: rate-based countermeasure



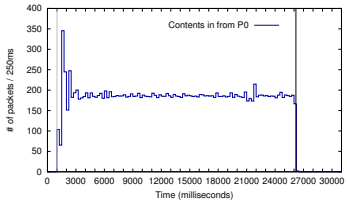
(g) R27: baseline



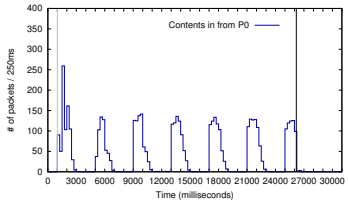
(h) R27: attack



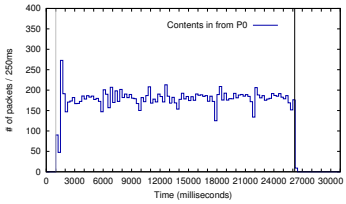
(i) R27: rate-based countermeasure



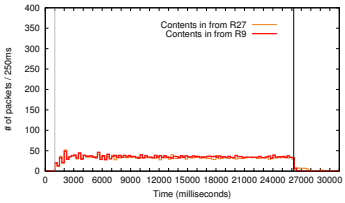
(j) R2: baseline



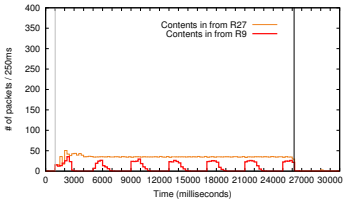
(k) R2: attack



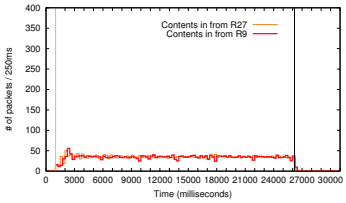
(l) R2: rate-based countermeasure



(m) R13: baseline



(n) R13: attack



(o) R13: rate-based countermeasure

Figure 6.4: DFN architecture, representative routers: content throughput (absolute values)

6.2 Distributed Countermeasures

We now consider our distributed (push-back) mechanism and we present results.

Figures 6.5(a) and 6.5(b) show the ratio of content packets received under attack with the the push-back countermeasure in place. To simplify comparison, we report the results of the simulations of the attack without countermeasures and with the previous (local) countermeasure.

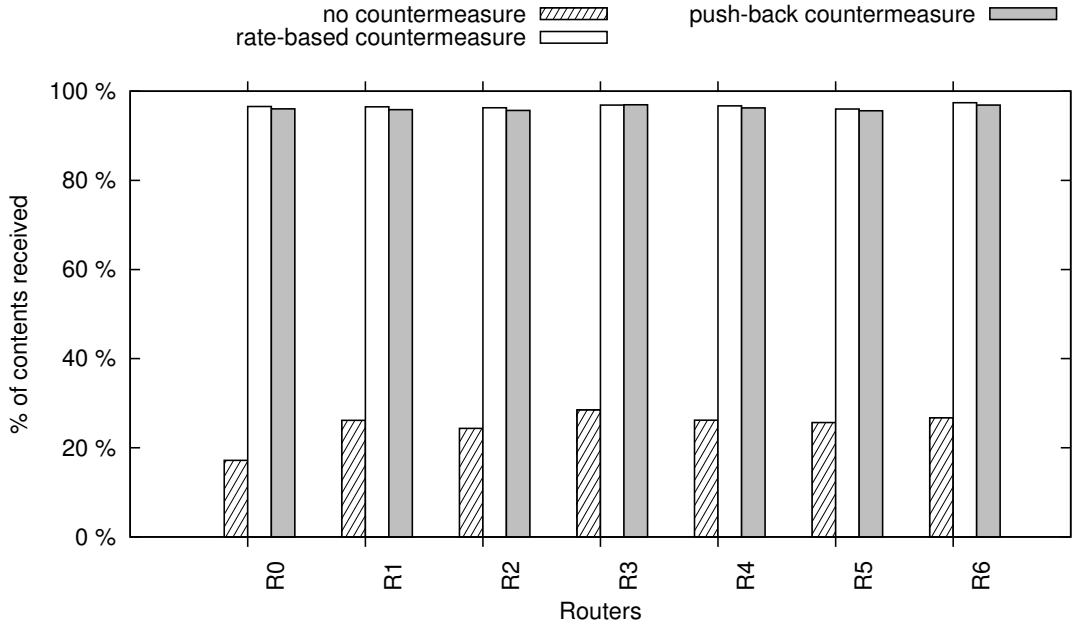
In the simple architecture, the push-back approach does not provide better results compared to the rate-based one; instead, we highlight a slight decrease in routed traffic. This is due to the goodness of the local countermeasure in a simple architecture. In fact, in simple topology the advantage of distributed countermeasure is not exploited. Distributed countermeasure has been design with a clear purpose in mind, reduce the time needed to detect an IFA by exchanging information between router. In fact, as described in Section 6.1, local countermeasure is able to detect IFA in a very short time, nearly to the best time.

Table 6.2 reports detection point of IFA in the routers `R0`, `R4` and `R5`. Comparing tables 6.1 and 6.2 we can see that there are no evident differences on the detection time.

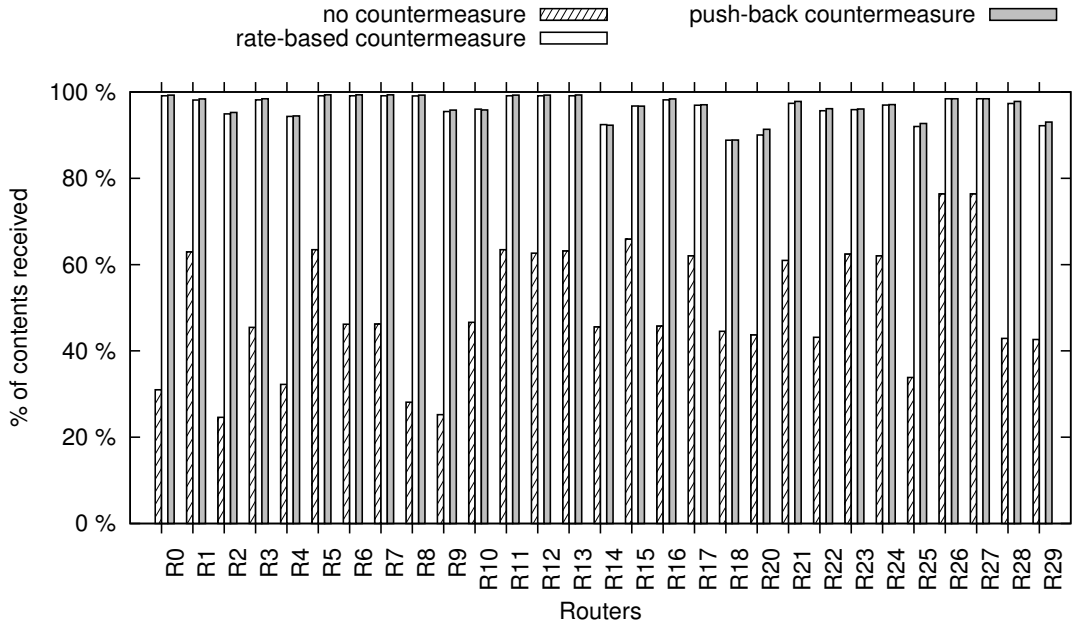
Routers	**Detection time (ms)**
`R0`	1,631
`R4`	1,830
`R5`	1,851

Table 6.2: Detection time of distributed countermeasure in simple topology

Detection time in Table 6.2 also reveals why distributed countermeasure are not effective in simple topology. The first router that detects IFA is `R0`, this means that when `R4` detects the attack and sends an alert to `R0`, `R0` has already applied its countermeasure so alert is useless. The same phenomenon occurs between `R5` and `R4`.



(a) Simple architecture



(b) DFN architecture

Figure 6.5: Push-back (distributed) countermeasure: throughput relative to baseline (percentage)

However, it is interesting to note how push-back countermeasure do not lower the performance than rate-based countemeasure in a significant manner. In fact, Figure 6.8 compares throughput of routers with rate-based countermeasure and push-back countermeasure. Results are very similar confirming that push-back do not lower the performance of the countermeasure.

The more complex DFN architecture exhibits a more interesting behavior. There, the push-back mechanism offers visibly better performance compared to the rate-based countermeasure. In particular, `R28`, `R20`, `R29`, and `R25` forward 16%, 13%, 11%, and 8% more content packets than in their counterparts in using the rate-based countermeasure.

A similar conclusion should apply also to the PIT usage – which is another measure for attach effectiveness. Instead, Figure 6.2(b) shows virtually no improvement over Figure 6.7 as Figure 6.2(a) shows virtually no improvement over Figure 6.6. In fact, improvements of distributed countermeasure are spread all over the simulation time and they appear not noticeable.

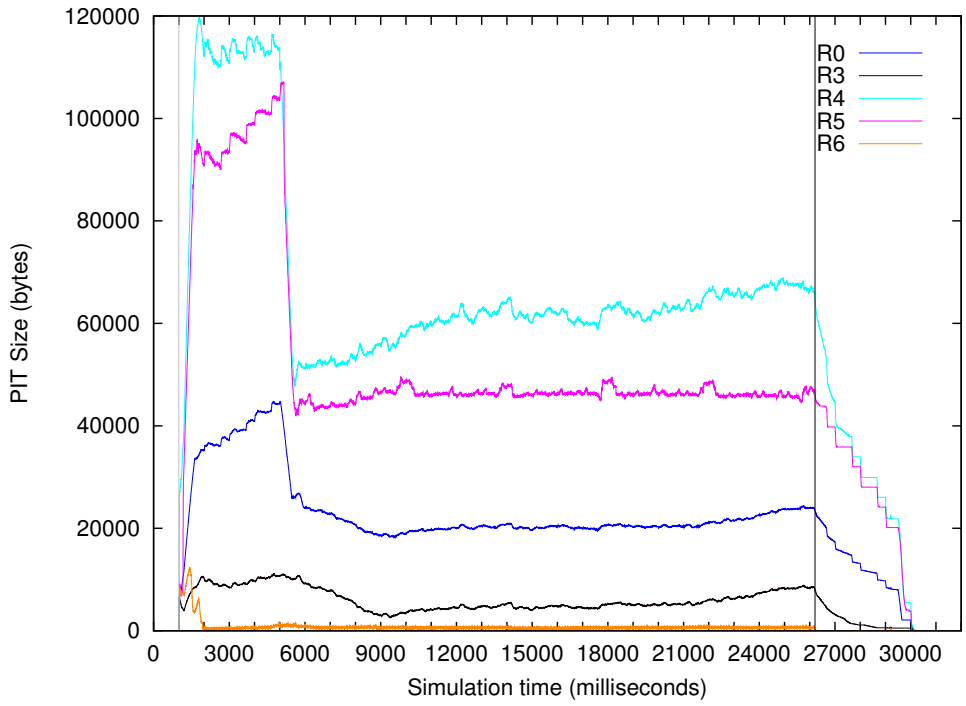


Figure 6.6: Push-back (distributed) countermeasure: PIT usage simple architecture

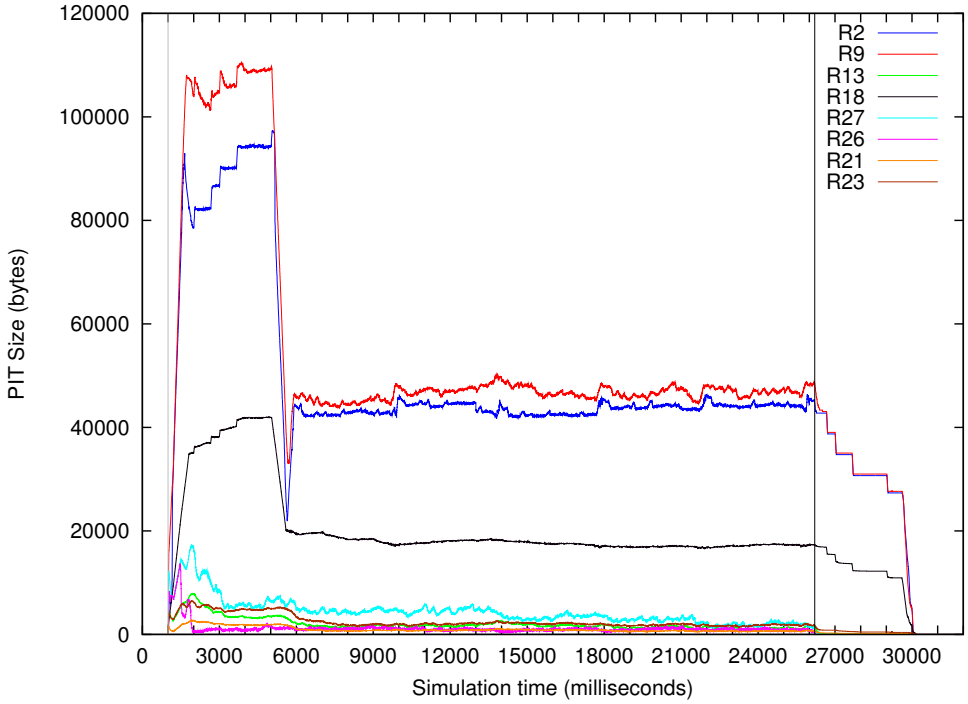
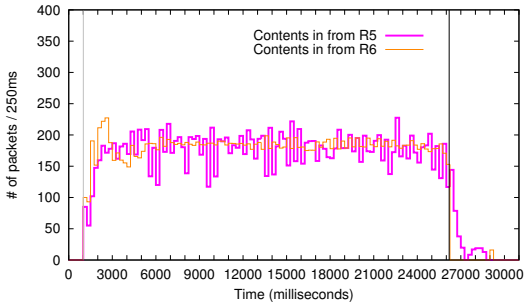
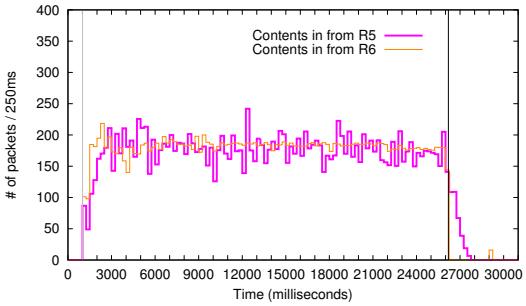


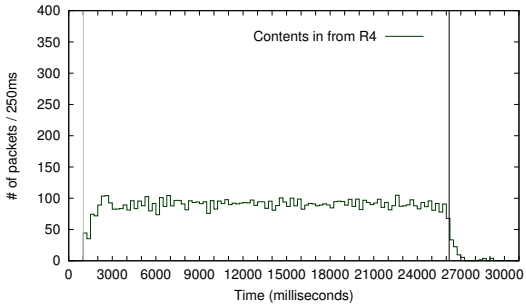
Figure 6.7: Push-back (distributed) countermeasure: PIT usage DFN architecture



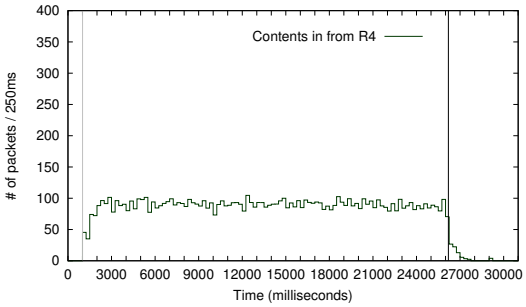
(a) R4: rate-based countermeasure



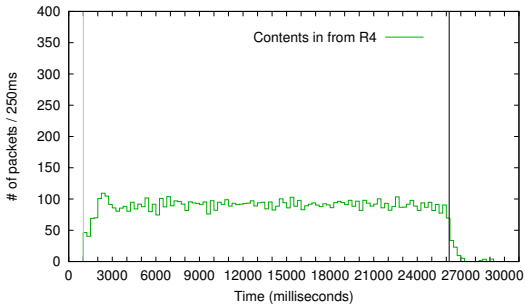
(b) R4: push-back countermeasure



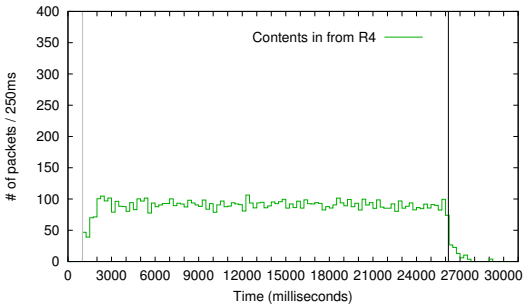
(c) R0: rate-based countermeasure



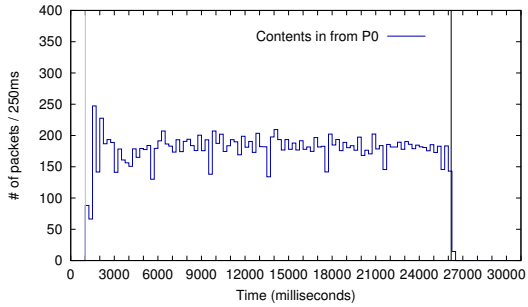
(d) R0: push-back countermeasure



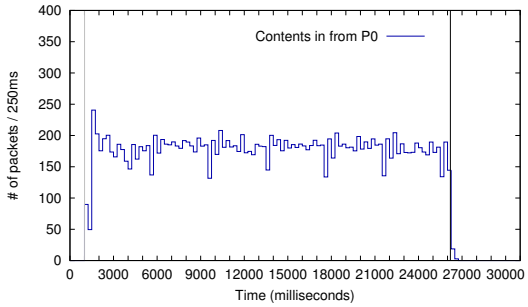
(e) R3: rate-based countermeasure



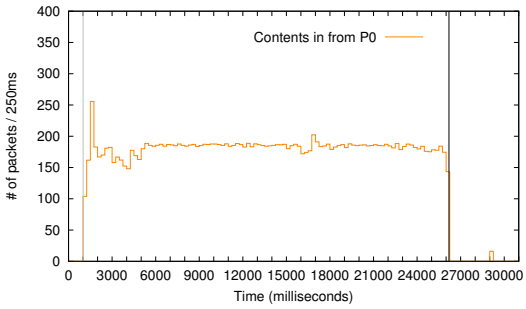
(f) R3: push-back countermeasure



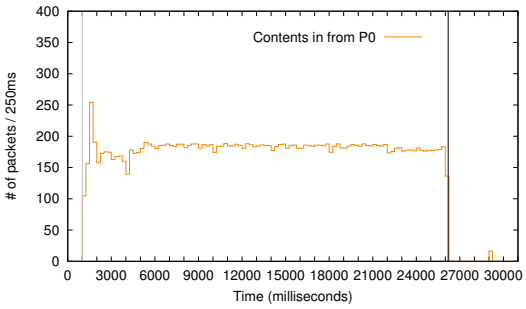
(g) R5: rate-based countermeasure



(h) R5: push-back countermeasure



(i) R6: rate-based countermeasure



(j) R6: push-back countermeasure

Figure 6.8: Rate-based/Push-back comparison: content throughput (absolute values)

Chapter 7

Conclusions and Future Works

In this thesis we discussed IFA-based DDoS over NDN. We provided, to the best of our knowledge, the first experimental evaluation of the attack. We demonstrated that IFA is a real and feasible threat for NDN. In particular, with a very low amount of resources, an adversary can obtain an impressive decreasing in network performance. In this chapter we summarize results we have obtained and we demonstrate the effectiveness of our countermeasure (Section 7.1). Moreover, in Section 7.2 we present some ideas and improvements that can be applied to Poseidon in future works.

7.1 Conclusions

In this thesis we have focused our attention on the effects of IFA and we have provided a working solution to counter this type of attacks. We have conducted experiments on the official NDN implementation codebase; we have argued that this setup provides reliable results, and closely mimics the behavior of physical (non-simulated) networks. We have used two topologies to understand and illustrate effects of IFA on the network. A simple topology has been used as a testing topology to understand how we can perform a devastating IFA and then we have verified results on DFN, the German Research Network.

We have demonstrated, in the both topologies, that IFA is a realistic threat; in particular, we have shown that, in the simple topology, an *adversary* with limited resources can reduce the amount of bandwidth allocated for content objects to 15-25% of the total bandwidth. Moreover, we have demonstrated that effects of IFA depends on the topology; an *adversary* with the same resources produces less malicious effects on the DFN topology.

Then we have introduced Poseidon, a new mechanism for detecting and mitigat-

ing IFA. Poseidon implements two mechanism to counter an attack: a local and a distributed countermeasure. While the first count only on local metrics to detect and thwart an attack, the second exploit a collaboration between routers to obtain better performance, in particular to detect as soon as possible an IFA.

We have shown that the benefits of Poseidon are significant. Our experiments have suggested that local countermeasure provides a very interesting results. In the simple topology, the worst case restores more than 95% of the available bandwidth during the attack. In the DFN topology, results have shown that Poseidon restore more than 80% of the bandwidth. Simulation with collaborative countermeasure have revealed a very important proprieties. Even if collaborative countermeasure is not able to improve performance, they do not decrease it. In fact, in our simulation on simple topology, results of collaborative countermeasure are comparable with results of local countermeasure.

Experiments with collaborative countermeasure reveal that in the DFN topology we obtain a better performance that local countermeasure. While some router have the same performance than local countermeasure, some other is able to forward up to 16% more content packets to the consumers. In fact, distributed countermeasure are more effective in a large topology and this confirm that our solutions should perform very well on the world wide network.

Poseidon is a first effort to counter IFA in NDN. Even if it has revealed very good performances we are aware that it needs a further refinement to improve its performance and to adapt it to all network topology. In the following section we present our plans for future works.

7.2 Future Works

To the best of our knowledge this thesis is a first work that address IFA on NDN. Even if Poseidon is working solution, we need more experiment and test to improve and generalize it. In the following we describe our future steps to obtain a full working solution, with good performance in all topologies and with different attacks (more *adversary* and different rate). Future works will dedicate to the following areas:

Extend IFA investigation: we want to conduct more experiment to study more deeply the effect of IFA with different parameters (e.g. *adversary* controls more consumers) on different topologies. If primary results tell us that IFA has very similar effects on the two used topology, we can not exclude a priory that for some other (larger) topology IFA can revels new different effects.

New metrics: we want to improve the effectiveness of Poseidon by introducing new

metrics for early detection of IFA. In particular we are studying solutions that consider also outgoing metrics and namespace of interest to better identify fake interests.

Reaction algorithm: Poseidon do not consider network topology on his reaction algorithm. We plan to design more sophisticated reaction algorithms that explicitly take into account the topology of the network and improve Poseidon performance.

Remove fake interests: fake interests stored in PIT are useless and counterproductive. Currently Poseidon do not remove them from PIT but it waits until they expire. We plan to investigate new techniques for identifying and removing fake interests from PITs before they expire. In this way we think we can even further reduce the effects of IFA.

New simulator: in our simulation we have used DCE module to run NDN code over NS-3 simulator. This enable us to obtain the results that are very close to how non-simulated implementation of NDN would behave. But, DCE add a significant overhead on top of NS-3 so that simulations on large topology are not possible. We intend to perform additional experiments using the NDN simulator developed at UCLA [2]. This simulator does not rely on DCE [12] and therefore scales significantly better than the one used in this work. This will allow us to run experiment over larger topologies.

Bibliography

[1] Martin Abadi. On SDSI's Linked Local Name Spaces. *Journal of Computer Security*, **6**[1-2]:3–21, October 1998.

[2] A. Afanasyev, I. Moiseenko, and Lixia Zhang. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005, 2012, University of California, Los Angeles, June 2012.

[3] Fabio Angius, Mario Gerla, and Giovanni Pau. Bloogo: Bloom filter based gossip algorithm for wireless ndn. In *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, pages 25–30, New York, NY, USA, 2012. ACM.

[4] Boldizsar Bencsath and Istvan Vajda. Protection against ddos attacks based on traffic level measurements, 2004.

[5] J. Burke, P. Gasti, N. Nathan, and G. Tsudik. Securing Instrumented Environments over Content-Centric Networking: the Case of Lighting Control. *ArXiv e-prints*, 2012.

[6] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. SPKI Certificate Theory. *RCF2693*, September 1999.

[7] Glenn Carl, George Kesidis, Richard R. Brooks, and Suresh Rai. Denial-of-service attack-detection techniques. *IEEE Internet Computing*, **10**[1]:82–89, January 2006.

[8] Content centric networking (CCNx) project. http://www.ccnx.org.

[9] Yu Chen, Kai Hwang, and Wei-Shinn Ku. Collaborative detection of ddos attacks over multiple network domains. *IEEE Trans. Parallel Distrib. Syst.*, **18**[12]:1649–1662, 2007.

[10] D.R. Cheriton and M. Gritter. Triad: A new next-generation internet architecture, 2000.

[11] ChoiceNet - Evolution through Choice. https://code.renci.org/gf/project/choicenet/. Retrieved July 2012.

[12] NS3 DCE CCNx. http://www-sop.inria.fr/members/Frederic.Urbani/ns3dceccnx/getting-started.html. Retrieved July 2012.

[13] DFN-Verein: DFN-NOC. http://www.dfn.de/dienstleistungen/dfninternet/noc/. Retrieved July 2012.

[14] Germany - DFN/WiN. http://lit.jinr.ru/LCTA/txt/Reports_probl_net/icfa/pswg_20.htm. Retrieved July 2012.

[15] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. ANDaNA: Anonymous named data networking application. In *NDSS*, 2012.

[16] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. *SPKI Certificate Theory*, September 1999. RFC2693.

[17] National science foundation (NSF) future of internet architecture (FIA) program. http://www.nets-fia.net/.

[18] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. DoS & DDoS in named-data networking. Technical report, University of California, Irvine, 2012.

[19] C. Gkantsidis and P. Rodriguez. Cooperative security for network coding file distribution. In *INFOCOM 2006*, 2006.

[20] Oliver Heckmann, Michael Piringer, Jens Schmitt, and Ralf Steinmetz. On realistic network topologies for simulation. In *ACM SIGCOMM MoMeTools*, pages 28–32. ACM Press, 2003.

[21] John Ioannidis and Steven M. Bellovin. Implementing pushback: Router-based defense against ddos attacks. In *NDSS*, 2002.

[22] Low Orbit ION Cannon project on SourceForge. http://sourceforge.net/projects/loic/. Retrieved July 2012.

[23] V. Jacobson, D. Smetters, N. Briggs, M. Plass, J. Thornton, and R. Braynard. Voccn: Voice-over content centric networks. In *The 2009 Workshop on Re-architecting the Internet*, 2009.

[24] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca Braynard. Networking named content. *Commun. ACM*, **55**[1]:117–124, 2012.

[25] Jae-Hyun Jun, Hyunju Oh, and Sung-Ho Kim. Ddos flooding attack detection through a step-by-step investigation. *Networked Embedded Systems for Enterprise Applications*, **0**:1–5, 2011.

[26] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *SIGCOMM '07*, pages 181–192, New York, NY, USA, 2007. ACM.

[27] Uichin Lee, Ivica Rimac, and Volker Hilt. Greening the internet with content-centric networking. In *e-Energy*, pages 179–182, 2010.

[28] Zenghui Liu and Yingxu Lai. A data mining framework for building intrusion detection models based on ipv6. In *Proceedings of the 3rd International Conference and Workshops on Advances in Information Security and Assurance*, ISA '09, pages 608–618, Berlin, Heidelberg, 2009. Springer-Verlag.

[29] Liming Lu, Mun Choon Chan, and Ee-Chien Chang. A general model of probabilistic packet marking for ip traceback. In *ASIACCS*, ASIACCS '08, pages 179–188. ACM, 2008.

[30] Jelena Mirkovic, Gregory Prier, and Peter L. Reiher. Attacking ddos at the source. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, ICNP '02, pages 312–321, Washington, DC, USA, 2002. IEEE Computer Society.

[31] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, **34**[2]:39–53, 2004.

[32] MobilityFirst FIA Overview. http://mobilityfirst.winlab.rutgers.edu. Retrieved July 2012.

[33] Jarmo Mölsä. Mitigating denial of service attacks: a tutorial. *J. Comput. Secur.*, **13**[6]:807–837, 2005.

[34] Chen mou Cheng, H. T. Kung, and Koan sin Tan. Use of spectral analysis in defense against dos attacks. In *In Proceedings of the IEEE GLOBECOM*, pages 2143–2148, 2002.

[35] Named data networking project (NDN). http://named-data.org.

[36] Nebula. http://nebula.cis.upenn.edu. Retrieved July 2012.

[37] NS-3. http://www.nsnam.org. Retrieved July 2012.

[38] Vern Paxson. Bro: a system for detecting network intruders in real-time. In *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*, SSYM'98, pages 3–3, Berkeley, CA, USA, 1998. USENIX Association.

[39] Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, **31**[3]:38–47, 2001.

[40] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, **39**[1], 2007.

[41] V. Ramasubramanian and E. Sirer. Perils of transitive trust in the domain name system. In *Proc. International Measurement Conference*, 2005.

[42] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, LISA '99, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association.

[43] Elisha J. Rosensweig, Jim Kurose, and Don Towsley. Approximate models for general cache networks. In *INFOCOM'10*, pages 1100–1108, Piscataway, NJ, USA, 2010. IEEE Press.

[44] Robert Stone. Centertrack: an ip overlay network for tracking dos floods. In *USENIX '00*, SSYM'00, pages 15–15, Berkeley, CA, USA, 2000. USENIX Association.

[45] Udaya Tupakula and Vijay Varadharajan. A practical method to counteract denial of service attacks. In *ACSC '03*, pages 275–284. Australian Computer Society, Inc., 2003.

[46] Haining Wang, Danlu Zhang, and Kang G. Shin. Detecting syn flooding attacks. In *INFOCOM*, pages 1530–1539. IEEE, 2002.

[47] Lucas Wang, Ryuji Wakikawa, Romain Kuntz, Rama Vuyyuru, and Lixia Zhang. Data naming in vehicle-to-vehicle communications. In *In Proceedings of INFOCOM 2012 Workshop on Emerging Design Choices in Name-Oriented Networking*, March 2012.

[48] Y. Wei, B. Mathieu, P. Truong, G. Simon, and J. Peltier. Realistic Storage of Pending Requests in Content-Centric Network Routers. In *ICCC 2012*, 2012.

[49] XIA - eXpressive Internet Architecture. http://www.cs.cmu.edu/~xia/. Retrieved July 2012.

[50] M. Xie, I. Widjaja, and H. Wang. Enhancing cache robustness for content-centric networks. In *INFOCOM*, 2012.

[51] Haowei Yuan and Patrick Crowley. Performance measurement of name-centric content distribution methods. In *Proceedings of the 2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*, ANCS '11, pages 223–224, Washington, DC, USA, 2011. IEEE Computer Society.

Appendix A

External Review

INVENTEURS DU MONDE NUMÉRIQUE

September 20th, 2012.

Dear Madam/Sir,

Please find in this letter my evaluation of the MSc thesis of Alberto Compagno.

The thesis is about security issues of Named Data Networking (NDN), which is one of the possible implementations of the Content-Centric Networking (CCN). NDN is an emerging networking paradigm being considered as a possible replacement for the current IP-based host-centric Internet infrastructure. This thesis addresses a specific Distributed Denial of Service (DDoS) attack to NDN, the Interest Flooding Attack (IFA). In NDN, communication is based on requests (interests) sent out by consumers, and replies (contents) generated by producers. Every time a consumer injects an interest on the network, NDN routers have to store a small amount of transient state. The IFA attack basically exploits in a malicious way the resources allocated to store this transient state.

The thesis starts with a very clear description of the problem, an extensive discussion of related work, and a clear overview of NDN. Subsequently, the thesis shows that IFA is a realistic threat: demonstrating how to implement it and evaluating its impact. The reported results show that an adversary with limited resources can use such attack to significantly influence the network performance. This would have a significant impact on the deployment of the new Internet architecture based on NDN (or similar implementations of CCN), and this strongly motivates this research project.

This thesis presents and fully evaluates Poseidon, a solution to mitigate

CENTRE DE RECHERCHE
GRENOBLE - RHÔNE-ALPES
Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex France
Tél. : +33 (0)4 76 61 52 00
Fax : +33 (0)4 76 61 52 52

www.inria.fr

IFA. Poseidon is a tool that includes two countermeasures to mitigate IFA: rate based (local) countermeasures, and push-back (collaborative) countermeasures.

The subject addressed by this thesis is timely: the NDN project is currently funded by the NSF and involves several top institutions in the US. Also, similar projects are currently ongoing both in the US and in Europe. The problem addressed in this thesis has a strong motivation: the IFA attack illustrated would have a significant impact on NDN performances. The idea proposed with Poseidon is original. The thesis is well organized and presents solid results. Furthermore, the thesis combines theory and practice, and contains several innovative ideas that deserve further investigation. Overall, this thesis is of high quality, and I would rank it among the best 5% of the thesis I have reviewed so far.

Finally, I would recommend the publication of the result of this work. Indeed, the work is currently under submission at a top security conference (the thesis supervisor confidentially shared the submitted version with me), and an extract of the work is already accepted for presentation as a poster the next December at the 2012 Annual Computer Security Applications Conference (ACSAC).

Claude Castelluccia
Senior Research Scientist,
Head of the Inria Security Research Group,
INRIA, France