République algérienne démocratique et populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministére de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العلمي و البحث العلمي
Université Amar Telidji Laghouat
جامعة عمار ثليجي بالأغواط

Facuté des Sciences
كلية العلوم
Département de Mathématiques et Informatique

Mémoire de Master

Domaine : Mathématiques et Informatique
Filiére : Informatique
Option : Réseaux, systémes et applications réparties (ReSar)

Présenté par :
Amira Ben Atallah
Djamila Bellaouar

Théme :

La sécurité des communications dans les NDN (Named Data Networks)

Soutenu devant le jury composé de :

M_r.	T.Ben Douma	Président	M.C.B	UATL
M_r.	B.Brik	Examinateur	M.A.B	UATL
M_r.	N.chaib	Examinateur	M.A.A	UATL
M_r.	N.Zinelaabidine	Raporteur	M.A.A	UATL
M_r.	A.Ben Arfa	Co-Encadreur	M.A.B	UATL

Année 2016

Dédicaces

« À nos chers parents M_r Ben Atallah Abdallah, M_m Ben Atallah Souad, M_r Bellaouar Mohamed *et* M_m Bellaouar Meriem, *qui nous ont guidés durant les moments les plus pénibles de ce long chemin, et qui ont été toujours à nos côtés et qui nous ont soutenu durant toute notre vie, un grand merci à eux.*

À la mémoire de nos grandes mères et grands-pères.

À nos chers frères et soeurs.

À toute notre famille.

À toutes nos amies sans exception. »

on dédie ce modeste travail.

Remerciements

Nous remercions tout d'abord **Dieu** Tout-puissant de nous avoir permis de mener à terme ce projet.

En préambule à ce mémoire, nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Ce n'est pas typique de commencer par le Co-Encadreur mais vu l'aide immense et le support qu'il nous a apporté, nous remercions très sincèrement Mr BEN ARFA Abdelmajid qui s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de cet humble travail. Nos remerciements s'adressent également à notre encadreur Mr NADIR Zinelaabidine qui a accepté de diriger ce mémoire. Nous remercions aussi Mr LAGRAA Nasredine pour les conseils précieux qu'il nous a fait part et ses encouragements pour aller plus loin.

Enfin, Nous exprimons nos sincères remerciements à nos parents, frères et soeurs pour le soutien moral et les encouragements. Leur présence et leur patience nous ont été d'un très grand soutien.

Résumé

Dans les années 1960 et 70, lorsque les idées de base sur Internet ont été développés, la téléphonie a été le seul exemple de communication à échelle planétaire qui est réussie et efficace.

Ainsi, les solutions de la communication offerte par TCP/IP étaient uniques et révolutionnaires, le problème qu'a résolu cette architecture était la téléphonie : la conversation point à point entre deux entités. Le monde a changé depuis cela.

Alors que l'architecture IP a dépassé toutes les prévisions pour faciliter l'interconnectivité, ça a était conçu pour la conversation entre des extrémités communicatives mais utilisées massivement pour la distribution du contenu. La nature *conventionnelle* de l'architecture IP réside dans le format du paquet : les paquets peuvent contenir seulement les adresses des deux extrémités en communication(adresse source et destination), cette restriction et d'autres ont poussé à l'apparition d'une nouvelle architecture dites *Named Data Networking*.

Named Data Networking est une architecture qui peut être utilisée pour nommer un morceau de données dans une conversation, comme le transport de la signature plus du numéro de la séquence se fait dans l'architecture TCP/IP, mais ils peuvent aussi nommer un bloc de données à partir d'une vidéo Youtube directement, plutôt que de le forcer à être intégré dans une conversation entre l'hôte consommatrice et **youtube.com**.

Néanmoins une telle architecture n'est nullement prémunie contre tout type d'attaques, dans ce mémoire nous allons se focaliser sur les attaques DDoS dans les NDN et les approche offertes jusqu'à présent, mais aussi proposé notre propre méthode pour lutter contre cette attaque, les résultats de la simulation nous ont permis de faire une étude et d'évaluer les performances de cette approche.

Mots clés : Named Data Networking, DDoS, déni de service.

Abstract

In the 1960 s and 70 , When the ideas of the base on the Internet Were developed , telephony Was the only example of a global communication Who is successful and Effective.

Thus, the communication solutions offered by TCP / IP were unique and revolutionary , the problem which this architecture has solved was telephony's : a point-to-point conversation between two entities.

While IP has exceeded all expectations for facilitating ubiquitous interconnectivity, it was designed for conversations between communications endpoints, but is overwhelmingly used for content distribution.The conversational nature of IP is embodied in its datagram format : IP datagrams can only name communication endpoints (the IP destination and source addresses), this restriction and others pushed to the emergence of a new architecture called textit Named Data Networking.

Named Data Networking can be used to name a chunk of data in a conversation, as the TCP/IP transport signature plus sequence number does today, but they can also name a chunk of data from a YouTube video directly, rather than forcing it to be embedded in a conversation between the consuming host and **youtube.com**.

Nevertheless, such an architecture is not forearmed against all types of attacks, in this thesis we will focus on DDoS attacks in the NDN and countermeasures offered so far, but also offer our own method to prevent against this attack, the simulation results have allowed us to do a study and evaluate the performances of this method.

Keywords : Named Data Networking, DDoS, denial of service.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Quand les architectes de l'Internet ont commencé à interconnecter des réseaux de communication, les types d'applications dont ils avaient besoin pour prendre en charge étaient simples par rapport aux normes d'aujourd'hui : connexion à distance, e-mail, transferts de fichiers et, plus tard, d'un accès Web. Ce qui est devenu connu sous le nom TCP / IP, qui est bien adapté à la tâche [1] .

Presque immédiatement après la naissance de l'Internet dans les années 1980s, l'industrie a développé le système de résolution des noms de domaines (DNS) pour traduire les adresses IP numériques données aux extrémités en des noms comme acme.com, afin que l'échange de données soit plus facile pour les gens et les applications. Mais la confiance en TCP / IP et DNS a conduit à la création de deux espaces de noms séparés - l'un pour les numéros et un autre pour les noms - un système que certains chercheurs disent est devenu de plus en plus complexe et problématique. En effet, l'Internet d'aujourd'hui est un endroit très différent, après avoir transformé en une constellation de diffusion et le contenu de vidéo à la demande, les transactions de commerce électronique, les médias numériques et sociaux, applications Smartphones, et de logiciels Cloud.

Pour livrer un paquet à l'adresse IP de destination spécifiée dans le paquet tout en servant des applications qui fonctionnent en termes de noms, un groupe de chercheurs qui travaillent sur le projet (NDN), qui s'agit d'une nouvelle architecture d'Internet destinée à remplacer le protocole TCP / IP. Leur objectif est de créer des protocoles qui soutiendraient un seul espace de noms et d'éliminer les adresses IP numériques, permettant aux consommateurs et applications d'accéder à ces données en utilisant uniquement des noms. Cette nouvelle technologie pourrait offrir la possibilité de déplacer de gros morceaux de données sans nécessairement identifier l'adresse mais simplement pointant vers un certain sujet, ce qui permettrait d'accélérer le flux d'information.

Ce mémoire s'intéresse de cette nouvelle technologie, et surtout de son modèle de sécurité, qui est encore en cours de développement.

Dans le premier chapitre, nous allons tout d'abord définir le NDN (Named Data Networking) et de ses propriétés et ses méthodes de routage ainsi que sa sécurité. Ensuite, dans le deuxième chapitre, nous nous intéressons des attaques DDoS dans NDN et les mesures prévues pour les affaiblir.

Le chapitre 3 va présenter notre propre contribution afin d'essayer d'améliorer la façon dont NDN utilise comme mesure contre les attaques de l'inondation par paquets

de requête.

Chapitre 1

NAMED DATA NETWORKING

1.1 Introduction

L'Internet d'aujourd'hui est basé sur le TCP/IP qui fonctionne avec les adresses IP. Il était puissant et efficace depuis longtemps. Mais, il semble qu'il risque d'arriver à ses limites à cause du changement de types d'applications par rapport au temps de naissance de l'Internet.

NDN est une architecture entièrement nouvelle, mais son concept principal est dérivé du modèle actuel d'Internet, ce qui reflète notre compréhension des points forts et des limites de l'architecture actuelle de l'Internet.

Dans ce chapitre, nous allons définir le NDN et parler de son principe de fonctionnement, son architecture et routage ainsi que son concept de sécurité.

1.2 Définition

Named data Networking (NDN) est l'un des cinq projets de recherche de la fondation nationale des sciences aux États-Unis comme étant l'architecture du futur [2]. Cette architecture combine tous les avantages offerts par l'architecture d'Internet actuelle tout en remédiant au problème irrésolu jusqu'à présent comme le problème d'IP ; pour autant que le protocole lui-même fonctionne avec une grande fiabilité, garantissant la livraison des paquets de données dans l'ordre où ils ont été envoyés. Il fonctionne exactement comme il a été conçu pour faire. Le protocole a été conçu il y a environ 40 ans et il témoigne des «anciens de l'Internet» qu'il a enduré pendant si longtemps. IP continuera d'être la norme de l'Internet standard si l'Internet n'était pas une aire de jeu pour les pirates, par exemple.

La sécurité est maintenant au sommet de la plupart des agendas des entreprises du jour, et qui est là où elle devrait rester, étant donné la rapidité du changement et les techniques sophistiquées utilisées par les pirates d'aujourd'hui. La sécurité est donc un des principaux facteurs de changement.

1.3 Principe de fonctionnement

Une simple analyse du trafic d'Internet d'aujourd'hui nous montre que les utilisateurs du réseau mondial ne s'intéressent pas à l'emplacement sur lequel le contenu est présent mais plutôt ils s'intéressent beaucoup plus au contenu lui-même (vidéos, articles, messages...etc). C'est exactement ce que NDN est censé faire i.e. offrir la possibilité d'exploiter les noms des données plutôt que les adresses numériques de leurs conteneurs pour faire circuler les contenus sur Internet.

Les six principes architecturaux suivants sont utilisés pour guider la conception de l'architecture NDN grâce aux leçons apprises au cours des années [3].

1. L'architecture de sablier est ce qui rend la conception d'Internet d'origine élégante et puissante. Elle est centrée sur la couche réseau universel (IP) mettant en oeuvre la fonctionnalité minimale nécessaire pour l'interconnectivité mondiale. Cette soi-disant "taille fine" a été un facteur clé de la croissance explosive de l'Internet, en permettant aux technologies de couche inférieure et supérieure à innover sans contrainte inutile. NDN conserve la même architecture en forme de sablier comme représenté sur la figure suivante [4].

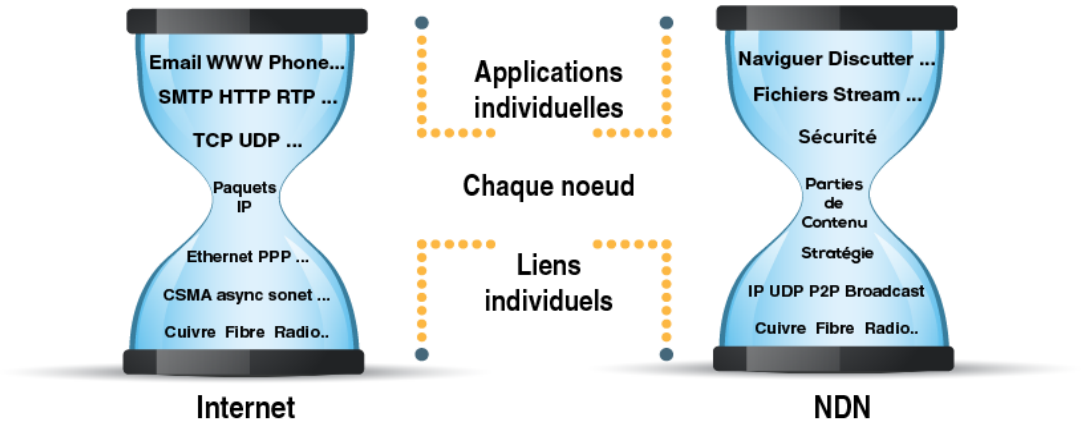

Figure 1.1 – L'architecture Sablier d'Internet : *TCP/IP* et *NDN* .

2. Le principe de bout en bout permet le développement d'applications robustes face aux défaillances du réseau. NDN conserve et élargit ce principe de conception.
3. La séparation du routage et la transmission son avéré nécessaire pour le développement de l'Internet. Elle permet le fonctionnement de la transmission alors que le système de routage continue d'évoluer au fil du temps. NDN adopte le même principe pour permettre son déploiement avec la meilleure technique de

transmission disponible pendant que des nouvelles recherches s'effectuent sur le système de routage.

4. La sécurité doit être intégrée dans l'architecture. La sécurité dans l'architecture actuelle de l'Internet est une réflexion après coup, ne répondant pas aux exigences de l'environnement de plus en plus hostile d'aujourd'hui. NDN offre des mesures de sécurité de base en signant tous les paquets de données.
5. Le trafic réseau doit bénéficier d'une autorégulation. la livraison de flux équilibré données est essentielle pour le fonctionnement du réseau stable. Puisque IP effectue la remise de données en boucle ouverte, les protocoles de transport ont été modifiés pour assurer l'équilibre du trafic unicast. NDN conçoit un flux balancé dans la taille fine.
6. L'architecture devrait faciliter le choix de l'utilisateur et de la concurrence, si possible. Bien que ce ne soit pas un facteur pertinent dans la conception d'Internet d'origine, le déploiement mondial nous a enseigné que «l'architecture n'est pas neutre". NDN fait un effort remarquable pour renforcer les utilisateurs en bout et de permettre la concurrence [5].

1.4 Architecture et principe de communication dans NDN

Dans cette section, nous donnons d'abord un bref aperçu des concepts de base de la livraison des données NDN, puis nous allons expliquer chaque élément et son rôle dans l'architecture globale. Semblable à l'architecture IP d'aujourd'hui, la taille fine est la pièce centrale de l'architecture NDN. Toutefois, en raison que la taille fine de NDN utilise des noms de données au lieu d'adresses IP pour la livraison afin d'offrir un nouvel ensemble de fonctionnalités minimales, ce changement (apparemment simple) conduit à des différences significatives entre l'IP et NDN dans leurs opérations de livraison des données.

1.4.1 Les types de paquets en NDN

On peut distinguer deux types de paquets dans NDN, les paquets de requête et les paquets de données.

1.4.1.1 Paquet de Requête (Interest Packet)

Un consommateur met le nom de la donnée désiré dans le paquet de requête et l'envoi dans le réseau. Les routeurs utilisent ce nom pour acheminer le paquet de requête vers le(s) producteur(s) de donnée [6]

Figure 1.2 – Paquet de requête

Le paquet de requête contient les champs suivant (voir figure[1.2]) :

Nom de Contenu : Nom hiérarchique de contenu NDN. Contient une séquence de composants de nom.
Sélecteur :
Contient les paramètres qui caractérisent la requête demandé.
Nonce :
Une chaine de caractères d'une longueur de 4 octets générés aléatoirement. La combinaison du nom et Nonce doit identifier de manière unique un paquet de requête. Utilisé pour détecter les boucles de requête.
Guideur :
Contient "Interest Life Time" qui indique une estimation du temps restant avant l'expiration du paquet de requête.

1.4.1.2 Paquet de Données (Data Packet)

Une fois le paquet de requête atteint un noeud qui possède les donnée demandées, le noeud va répondre avec un paquet de données, ce paquet contient le nom de contenu, avec une signature faite par la clé du producteur qui lie les deux. Le paquet de données fait une marche arrière en suivant le chemin pris par le paquet de requête pour arriver au consommateur qui a lancé la demande.

Figure 1.3 – Paquet de données

Le paquet de données contient les champs suivant (voir figure[1.3]) :

Nom de Contenu : Nom hiérarchique de contenu NDN. Contient une séquence de composants de nom.
Signature :
Les signatures de données sont obligatoires, les applications ne peuvent pas «se retirer» de la sécurité.
Informations Signées :
Contient l'ID du publieur, le localisateur de la clé, l'heure de péremption.
Données : Contient les données à transmettre.

1.4.2 Architecture du routeur

La communication dans NDN est entraînée par l'extrémité de réception, i.e : le consommateur de données. Pour recevoir des données, un consommateur envoie un paquet de requête, qui porte un nom qui identifie les données souhaitées. Un routeur se souvient de l'interface sur laquelle la demande est arrivée, puis transmet le paquet de requête en recherchant le nom dans son «forwarding information Base (FIB)», qui est remplie par un protocole de routage basé sur le nom. Une fois que la requête atteint un noeud qui possède les données demandées, un paquet de données est renvoyé, ce qui porte à la fois le nom et le contenu des données, ainsi que la signature par la clé du producteur. Ce paquet de données suit en sens inverse le chemin emprunté par la requête pour revenir au consommateur. Les paquets de requête sont acheminés vers les producteurs de données à la base des noms portés par les paquets de requête, et les paquets de données sont retournés selon les informations d'état mises en place par les requêtes à chaque saut de routeur.

Le routeur enregistre dans une table (Pending Interest Table PIT) les requêtes qui ont été envoyés mais pas encore satisfaites. Lorsque plusieurs requêtes pour les

mêmes données sont reçues en aval, seul le premier est envoyé en amont vers la source de données. Chaque entrée de PIT contient le nom de la requête et un ensemble d'interfaces à partir duquel les requêtes pour le même nom ont été reçues. Lorsqu'un paquet de données arrive, le routeur trouve l'entrée PIT correspondante et transmet les données à toutes les interfaces énumérées dans le PIT. Le routeur supprime ensuite l'entrée PIT correspondante, et met en cache les données dans le magasin de contenu (Content Store CS). Puisque les paquets de données NDN sont indépendants de l'endroit où ils viennent et où ils peuvent être transmis, le routeur peut les mettre en cache pour satisfaire les futures demandes [7].

Pour délivrer les paquets de requête et de données, chaque routeur NDN maintient trois structures, et une politique d'acheminement.

1.4.2.1 Pending Interest Table (PIT)

Dans NDN, les paquets sont transmis sur la base de résultats de recherche des noms. Dans le processus de transfert, la table des requêtes en attente (PIT) est l'un des composants de base. La table des requêtes en attente (PIT) conserve la trace des paquets de requête actuellement non satisfaits. Les paquets de requête arrivants sont transmis au prochain saut selon résultats de recherche dans le (FIB) seulement si le PIT ne trouve aucun paquet de requête en cours avec le même nom. Le PIT stocke également les informations de destination pour les paquets de données. Pour chaque paquet de données, le PIT est interrogé pour trouver l'interface d'entrée(s) qui a demandé le contenu, puis le paquet de données sera livré et son nom de contenu est supprimé du PIT. Par conséquent, le PIT nécessite des mises à jour par paquet, y compris la mémoire écrite(Voir la figure[1.4]).

1.4.2.2 Forwarding Information Base (FIB)

C'est une table de routage qui mappe le composant nom à des interfaces. Le FIB lui-même est remplie par un protocole de routage basé sur le préfixe du nom, et peut avoir des interfaces de sortie multiples pour chaque préfixe.

1.4.2.3 Content Store (CS)

Dans cette base de données sont stockés des objets physiquement. C'est un cache temporaire de paquets de données que le routeur a reçu. Parce qu'un paquet de données NDN est significativement indépendant de l'endroit où il vient ni où il est transmis, il peut être mis en cache pour satisfaire les futurs centres de requête. La stratégie de remplacement est déterminée par le routeur.

Le PIT contient la liste des requêtes préalablement transmises, mais pour lesquels il n'y a pas encore de réponse. Ce tableau stocke les interfaces à partir desquelles un paquet de requête avait été reçu, pour mettre en oeuvre le transfert de chemin inverse (reverse-path forwarding) : dès qu'un routeur reçoit un paquet de données, il :

1. Vérifie le PIT.
2. Transmet le paquet sur les mêmes interfaces à partir desquelles les paquets de requêtes de cet objet sont arrivés.
3. Supprime l'enregistrement correspondant dans le PIT.

Le content store (CS) agit comme une mémoire cache persistante pour le noeud. Lorsqu'un paquet de requête arrive, le routeur d'abord interroge le CS, en cas de succès, le routeur sera capable de servir directement le paquet de données. Sinon, en cas de défaut, le noeud va vérifier si le PIT contient un enregistrement correspondant au nom du contenu demandé. Si un tel enregistrement existe, le paquet de requête sera supprimé, et l'interface à partir de laquelle le paquet de requête a été reçu sera ajoutée au PIT. Au contraire, s'il n'y a pas des enregistrements dans le PIT pour ce contenu, le noeud va :

1. Ajouter un nouveau enregistrement pour le paquet de requête reçu dans le PIT.
2. Il va transmettre le paquet de requête vers le producteur de données. (Voir la figure [1.4])

En particulier, le prochain noeud sera choisi par le ForwardingStrategy Layer (La stratégie de routage "forwarding strategy" décide comment un paquet de requête va être transmis), selon les données disponibles dans le FIB. Les paquets de données sont transmis sur le chemin inverse par rapport aux Requêtes, et cette symétrie supprime la nécessité de fournir les données relatives à la position des hôtes. Pour des raisons similaires, les paquets de données ne peuvent pas boucler dans NDN, tandis que les boucles de requêtes sont évitées grâce à l'état du PIT. Plus en détail, chaque requête contient un Nonce généré aléatoirement qui - couplé avec le nom - identifie de manière unique le paquet. Étant donné que les noeuds maintiennent à chaque fois le nom de la requête et le Nonce dans le PIT, ils vont détecter et éliminer le bouclage des paquets de requêtes. (Voir la figure [1.4]).

1.4.3 Les noms

Les noms NDN sont abstraits dans le réseau i.e. les routeurs ne connaissent pas la signification d'un nom (bien qu'ils connaissent les limites entre les composants dans un nom). Cela permet à chaque application de choisir le schéma d'appellation qui correspond à ses besoins et permet au schéma d'appellation d'évoluer indépendamment du réseau. La conception de NDN considére que les noms soient structurés hiérarchiquement, par exemple, une vidéo produite par le lagh-univ peut avoir le nom */lagh-univ/informatique/cours.avi*, où **'/'** indique une frontière entre les composants de noms (il ne fait pas partie du nom), similaire aux URLs. Cette structure hiérarchique est utile aux applications pour représenter les relations entre les morceaux des données. Cette structure hiérarchique permet aux applications de représenter les relations entre éléments de données.Par exemple le segment 3 de la version 1 de la vidéo pourrait être nommé */lagh-univ/informatique/cours.avi/1/3*. Ça permet aussi l'agrégation des noms : Dans cet exemple lagh-univ correspond à un système autonome d'où provient vidéo.

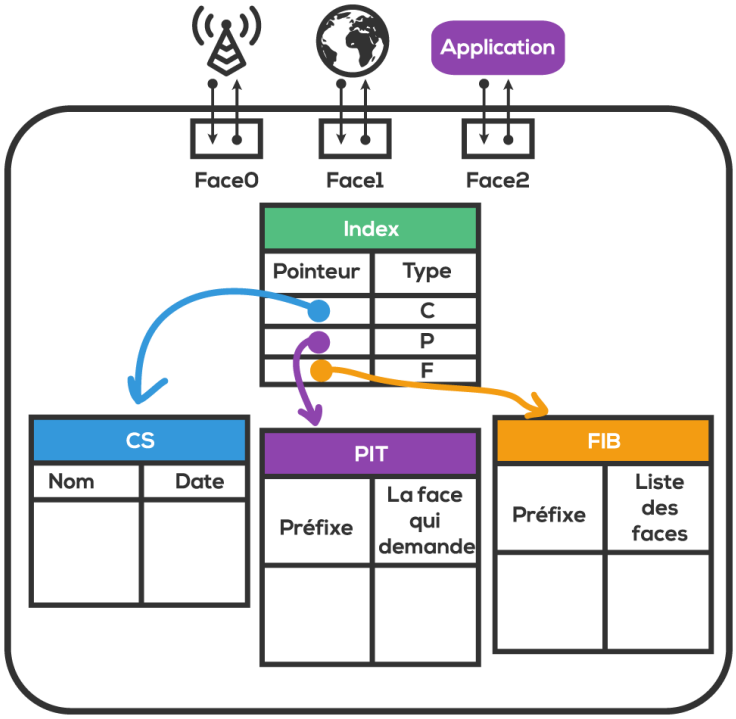

Figure 1.4 – Noeud NDN

1.5 Routage dans NDN

Aujourd'hui, il y a une variété de solutions proposées et efficaces pour la plupart des problèmes de routage. Tout système de routage qui fonctionne bien pour IP devrait aussi bien fonctionner pour NDN, parce que NDN représente un héritage du modèle IP avec la même sémantique pertinente au routage. NDN offre le transport des protocoles de routage. Puisque NDN fournit un modèle d'information sécurisé et robuste, l'utilisation de NDN comme un moyen de transport de routage peut rendre la protection de l'acheminement des infrastructures presque automatique.

En outre, il est fondamental que la nouvelle architecture de l'Internet soit capable de coexister avec le protocole TCP/IP actuel pour avoir une chance de remplacer efficacement l'architecture courante de l'Internet. En fait, il est impensable d'éteindre Internet, substituer tous les routeurs et de le rallumer après. NDN est très prudent sur ce problème et on avait pensé à l'éviter [8].

1.5.1 L'exigence de NDN à un protocole de routage

NDN exige le routage de l'information, c'est-à-dire guider des paquets de requête aux producteurs de données (via tout/tous les chemins possibles). Mais d'un autre côté, NDN n'exige pas la convergence de routage rapide.(voir figure[1.5])

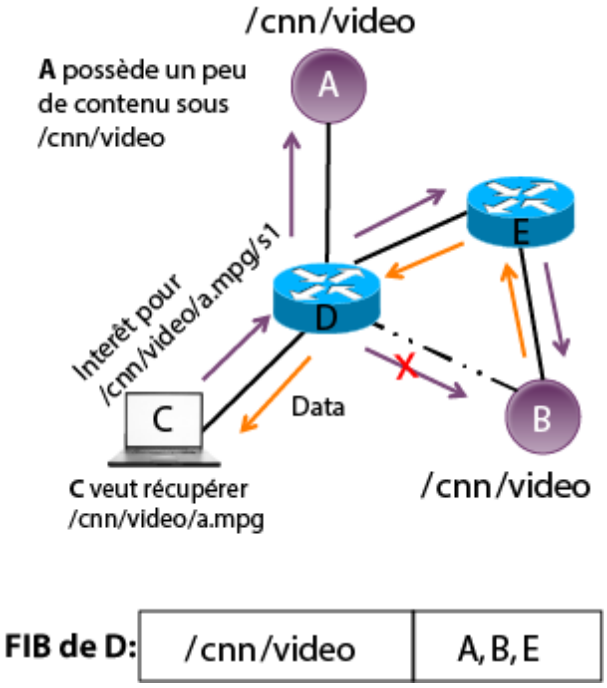

Figure 1.5 – L'exigence de routage de l'information en NDN

1.5.2 Mécanisme de routage dans NDN

Tout algorithme de calcul d'itinéraire qui fonctionne pour IP (par exemple, l'état de lien) peut être utilisé dans NDN.

1.5.2.1 Différences

Il existe des différences majeures entre l'architecture TCP/IP et NDN i.e. les NDN :

1. Remplace les adresses IP par des noms.
2. Calcule une liste des prochains sauts pour chaque nom.
3. Diffuse des mises à jour de routage utilisant des paquets de requêtes/paquets de données.

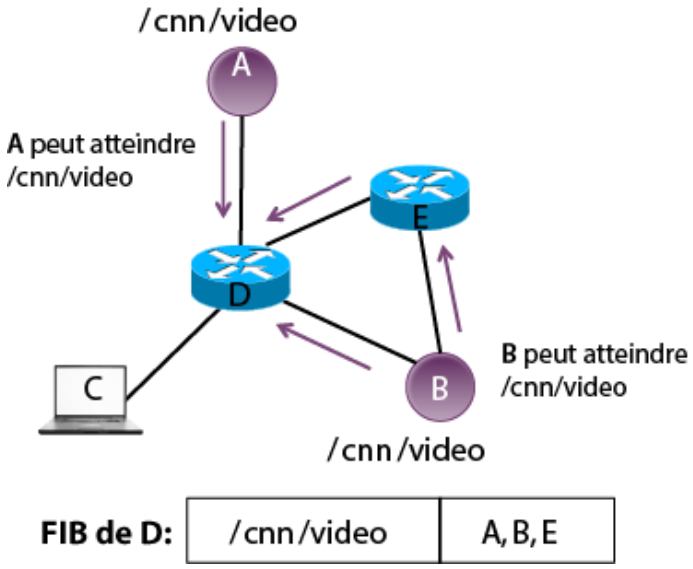

Figure 1.6 – Mécanisme de routage de l'information en NDN

1.5.2.2 La Communication entre le PIT et le CS

Pour mieux comprendre la recherche des paquets et les processus de transfert, il faut rappeler que le PIT assure le suivi des paquets de requête (interest packet) en attente, c'est-à-dire les requêtes reçues, mais pas encore répondus. Le processus spécifique de la recherche ainsi que de la transmission des paquets de requête et des données est représenté sur les figures[1.7] et [1.8].

La figure suivante montre qu'une fois un paquet de requête arrive à l'interface 'i' d'un routeur NDN 'R', ce routeur :

1. Consulte CS si le contenu désiré est présent et renvoie une copie dans le paquet de données via l'interface i.
2. Dans le cas inverse, il recherche dans le PIT pour des entrées pour cette requête. Si oui, il ajoute l'interface 'i' à cette entrée, et annule le paquet de requête.
3. Sinon, il crée une entrée pour cette requête et ajoute l'interface 'i' à cette entrée.
4. Il retransmet la requête au saut suivant en recherchant dans le FIB.

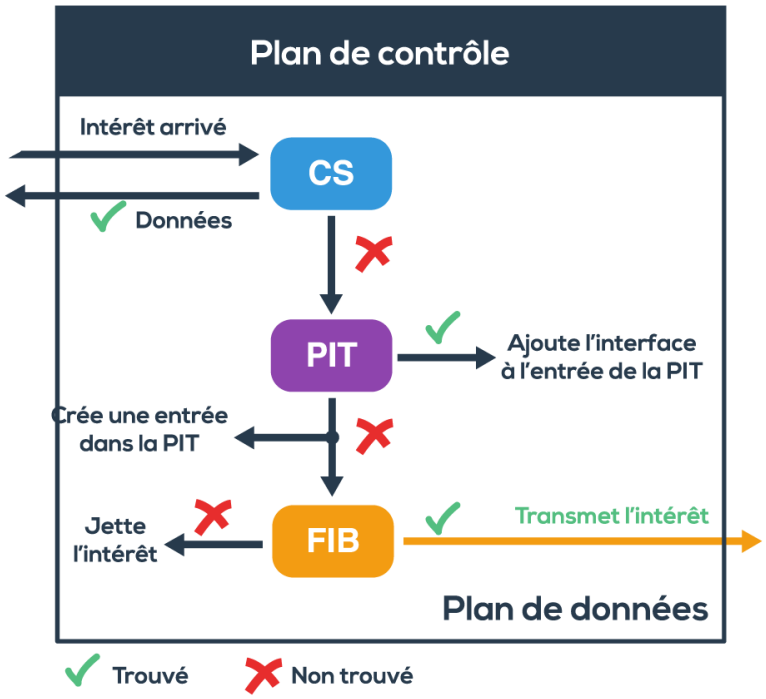

Figure 1.7 – Processus de transmission des paquets de requête

Au retour de paquets de données, la figure suivante montre que le routeur R :

1. Transmet le paquet de données sur toutes les interfaces demandantes, dans l'entrée correspondante de la table et supprime cette entrée.
2. Garde le paquet de données dans le CS selon la politique utilisée.

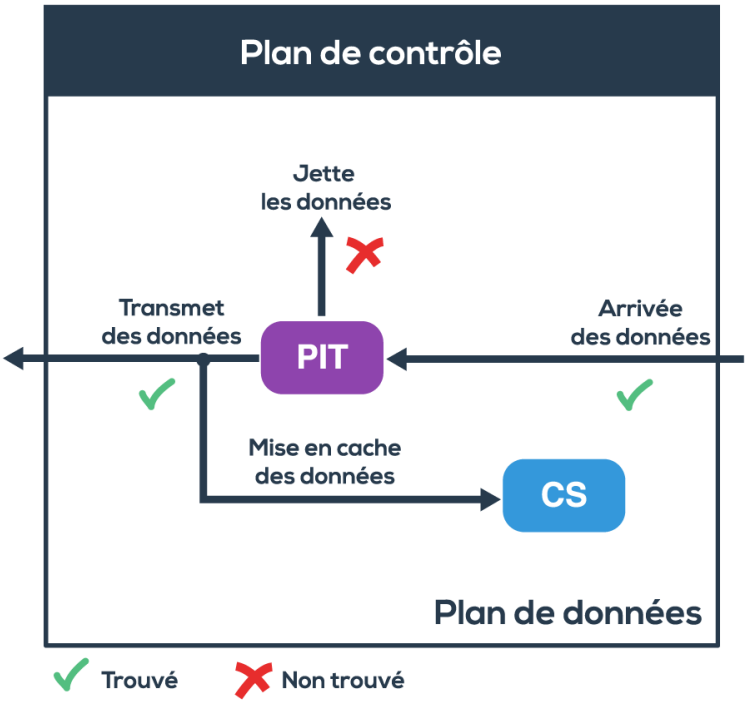

Figure 1.8 – Processus de transmission des paquets de données

1.6 Sécurité dans NDN

La sécurité est l'une des innovations incluses dans NDN. L'expérience avec TCP / IP a démontré que faire inclure un degré de sécurité dans la taille fine est très important et obligatoire. Par conséquent, NDN a été conçu pour répondre à ce besoin.

NDN est construit sur la notion de sécurité basée sur le contenu : la protection est transportée avec le contenu lui-même, plutôt que d'être une propriété de connexions sur lesquelles le contenu se transporte.

Chaque paquet de données est authentifié par des signatures numériques, et cela met en place la sécurité dans le contenu lui-même au lieu des hôtes. Actuellement dans les réseaux IP, la confiance est basée sur l'endroit et comment (quel type de pipe), le client doit récupérer le contenu directement à partir de la source lui faisant confiance. En conséquence de la théorie CCN [1](qui est une arcitecture orientée information et qui tend à remplacer l'IP. NDN et d'autres architectures sont dérivés de celle-ci), NDN ne s'interesse pas des hôtes mais il fait plutot confiance au contenu directement.

1. CCN (Content Centric Networks) est une nouvelle architecture de réseau conçu pour correspondre à la façon dont le réseau fonctionne déjà , et pour faire face aux problèmes que les gens essaient de résoudre. NCF est la prochaine étape importante dans l'héritage notable du PARC dans les réseaux.

La signature de chaque paquet de données permet à l'éditeur de contenu de faire la liaison entre le nom et le contenu. Cette simple addition pour les paquets de données apporte une amélioration importante en matière de confiance. Premièrement, la signature est de bout en bout, ce qui permet à un consommateur de vérifier l'authenticité du contenu. En outre, les données NDN sont publiquement authentifiables, la signature est une clé publique et tout le monde peut vérifier qu'une telle liaison nom-contenue a été signée par une telle clé.

L'algorithme de signature, utilisé pour signer les paquets de données, est sélectionné par le producteur et choisi pour achever les performances requises dans le processus de vérification de signature. Chaque paquet de données comprend, en outre, toutes les informations nécessaires pour vérifier la signature et récupérer la clé publique nécessaire [5].

1.7 Conclusion

Les réseaux NDN reçoivent récemment beaucoup d'attention en raison de leur manière intelligente orientée contenu pour servir les données aux utilisateurs tout en implémentant des normes de sécurité par conception. Néanmoins, leur déploiement à grande échelle nécessite la résolution de nombreux problèmes de sécurité, l'un de ces problèmes est les attaques DoS/DDoS, qui peuvent être un obstacle réel avant que ne nous puissions utiliser NDN à l'échelle mondiale. En parlant de la signature des paquets, cette vérification minime de signature peut être étonnamment utile dans la défense contre de nombreux types d'attaques réseau, mais pour les attaques DoS/DDoS, il faudrait compter sur d'autres méthodes, et c'est le but de ce mémoire. Le chapitre suivant vise à présenter les différentes méthodes prévues pour affaiblir l'impact de ces attaques sur le fonctionnement des routeurs NDN.

Chapitre 2

LES ATTAQUES DDoS DANS NDN

2.1 Introduction

NDN est principalement orienté vers la distribution efficace de contenu à grande échelle [9]. Au lieu d'établir des connexions IP directes avec un serveur de contenu, les consommateurs NDN demandent directement des éléments de contenu par leur nom, le réseau est en charge de trouver la copie la plus proche du contenu, et de la récupérer aussi efficacement que possible. L'un des objectifs clés du projet NDN est la «sécurité par design».

Pour qu'il représente une architecture Internet fiable, NDN doit être résistant contre tout type de menaces, actuelles et émergentes. Dans ce qui suit, on s'intéresse au DDoS distribué. Les attaques DDoS présentent de graves menaces pour l'Internet actuel et dont NDN ne sont pas immunisés contre [10] .

Il existe deux types d'attaques DDoS dans le domaine NDN, l'inondation par paquets de requête (IFA) et les attaques à cible unique(STA).

2.2 Attaque DoS et DDoS

Les attaques par déni de service DoS (Denial of Service en anglais) ont des attaques qui visent à rendre une machine ou un réseau indisponible durant une certaine période. En apparence une telle attaque peut sembler inoffensive si elle vise un réseau ou un ordinateur particulier, mais elle peut s'avérer redoutable lorsqu'elle vise un serveur ou des ressources matérielles appartenant à une grande société dépendante de son infrastructure réseau.

Le but d'un tel déni de service est de surcharger la bande passante du serveur cible et d'autres ressources. Cela rendra le serveur inaccessible aux autres, bloquant ainsi le site Web où tout autre chose est hébergée là.

Les attaques DDoS (Distributed Denial of Service) sont similaires aux attaques DoS, mais les résultats sont très différents. Au lieu d'un seul ordinateur et d'une seule connexion internet, les attaques DDoS utilisent plusieurs. Les ordinateurs derrière une telle attaque sont souvent distribués à travers le monde entier et fassent partie de ce qui est connu comme un botnet[]. La différence principale entre DoS et DDoS est que

le serveur cible sera en surcharge par des centaines même des milliers de requêtes avec le DDoS alors que dans le cas du DoS, le serveur est ciblé par un seul attaquant.

2.3 Attaque de l'inondation par paquets de requête (IFA)

Comme nous lavons expliqué dans le chapitre précédent, les paquets de requête dans NDN sont routés dans le réseau en se basant sur les noms de contenus des paquets et sur les tables(PIT, FIB, CS) dans les routeurs intermédiaires, cela permet aux pirates d'exploiter ces tables pour lancer des attaques DDoS dans NDN [10]. un attaquant ou bien un groupe d'attaquant peut injecter un nombre excessif de paquets de requêtes malicieux dans le but de saturer le réseau et causer un déni de service aux utilisateurs légitimes(voir figure[2.1]).

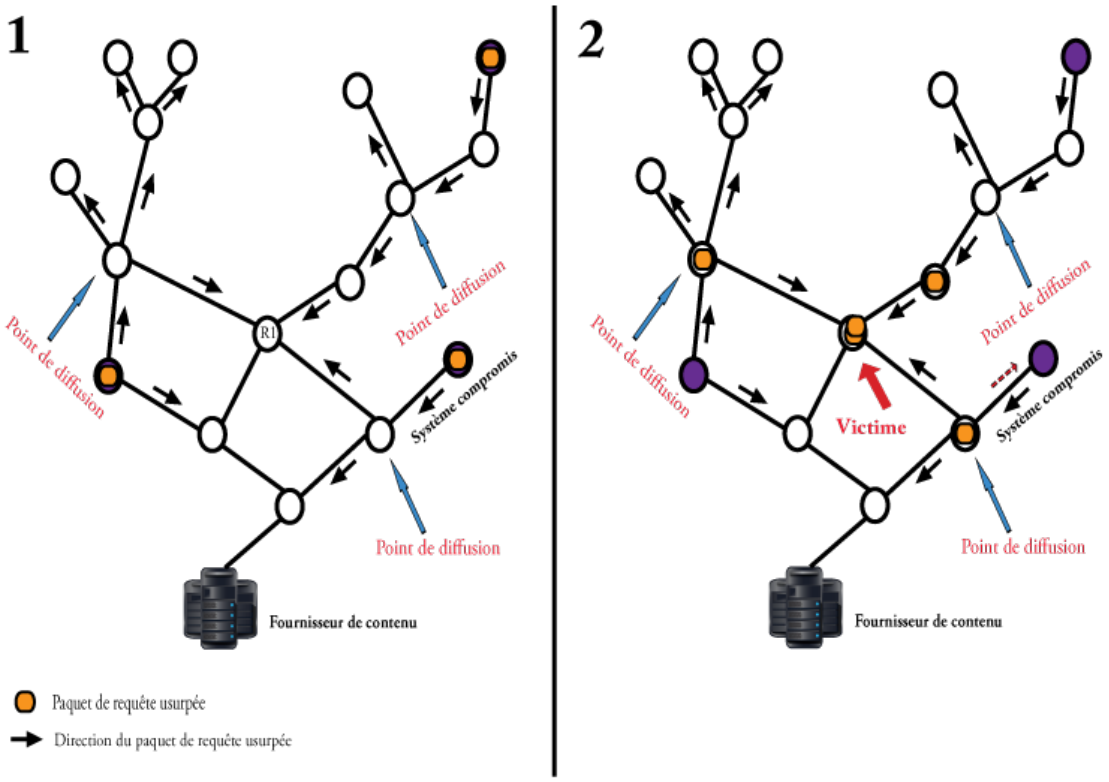

Figure 2.1 – Attaque de l'inondation par paquets de requête (IFA)

La figure[2.1] est un exemple d'une attaque d'inondation par paquets de requête, où les systèmes compromis transmettent des paquets malicieux à travers des noeuds de diffusion de façon ont saturé le PIT du routeur *R1*.

2.4 Attaque à cible unique (STA)

Ce type d'attaque ressemble à l'attaque DDoS dans les réseaux IP. Elle consiste à utiliser un préfixe de nom d'un contenu existant en encapsulant un suffixe non existant, de façon à cibler le fournisseur qui détient les paquets de données correspondant au préfixe. La composition du nom du paquet malicieux est la suivant :

La composition du nom spoofé : Préfixe existant + Suffixe forgé

La figure[2.2] est un exemple d'une attaque à cible unique, où les systèmes compromis utilisent un préfixe existant avec un suffixe forgé pour visée à saturer les PIT de *R1* et du serveur.

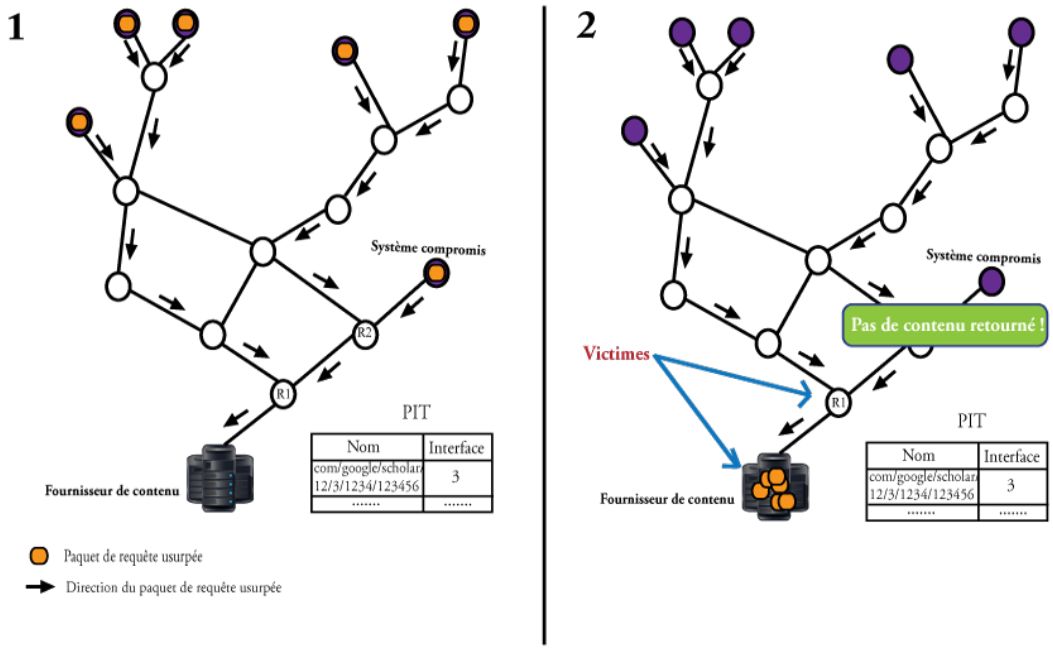

Figure 2.2 – Attaque à cible unique (STA)

2.5 Les Mesures de Défense contre les Attaques DoS/DDoS

Dans le but d'atténuer l'impact des attaques DDoS et plus spécialement l'attaque de l'inondation par paquet de requête, plusieurs méthodes ont été introduites. Parmi ces méthodes :

R	ensemble de tous les routeurs dans le réseau exécutant Poseidon
r_i	i-éme routeur,$1 \leq i \leq \|R\|$
r_i^j	j-éme interface sur le routeur r_i
t_k	k-éme intervalle de temps
$\omega(r_i^j, t_k)$	le débit entre les paquets de requête entrants et les paquets de données sortant d'une interface donnée r_i^j
$\rho(r_i^j, t_k)$	espace PIT utilisé par des paquets de requête arrivés sur l'interface r_i^j , mesurée à la fin de l'intervalle t_k
$\Omega(r_i^j)$	seuil de détection IFA pour $\omega(r_i^j, t_k)$
$P(r_i^j)$	seuil de détection IFA pour $\rho(r_i^j, t_k)$

Table 2.1 – Annotation

2.5.1 Poseidon

Proposé par Alberto Compagno [10]. Poseidon est une approche collaborative conçu pour lutter contre les attaques par inondation de paquet de requête. Cette mesure de défense est un ensemble d'algorithmes qui fonctionnent sur les routeurs, dans le but est d'identifier les anomalies du trafic (en particulier, attaque de l'inonadtion par paquet de requête) et en atténuer les effets.

Poseidon surveille en permanence chaque interface, le taux de paquets de requête non satisfaits par rapport à l'ensemble du trafic. Si ce taux change significativement entre deux intervalles de temps consécutifs, il définit un filtre sur les interfaces offensives (ce qui réduit le nombre des requêtes entrantes).

En outre, Poseidon peut émettre un message push-back "d'alerte" pour les mêmes interfaces, pour signaler l'attaque de l'innodation par paquet de requête en cours [10].

Dans les deux prochaines sous-sections, la phase de détection des attaques ainsi que la phase réactive dont la méthode Poseidon utilise seront définis, le tableau[2.1] montre les annotations utilisées dans les pseudo-codes.

2.5.1.1 Détection de l'attaque

Les attaques sont détecté en utilisant deux paramètres : $\omega(r_i^j, t_k)$ et $\rho(r_i^j, t_k)$. Le premier paramètre représente le nombre de paquets de requête arrivés divisé sur le nombre de paquets de données sorties du routeur r_i sur l'interface r_i^j avec un intervalle de temps t_k.

$$W(\underset{i}{\overset{j}{r}}, \underset{k}{t}) = \frac{\#\ de\ paquets\ de\ requête\ venant\ de\ \underset{i}{\overset{j}{r}}\ dans\ un\ intervalle\ \underset{k}{t}}{\#\ de\ paquets\ de\ données\ venant\ de\ \underset{i}{\overset{j}{r}}\ dans\ un\ intervalle\ \underset{k}{t}}$$

Le second paramètre indique la quantité d'espace(en octets)utilisé pour stocker les paquets de requête dans le PIT, venant d'une interface r_i avec un intervalle de temps t_k.

Poseidon détecte les attaques lorsque $\omega(r_i^j, t_k)$ et $\rho(r_i^j, t_k)$ dépasse leurs seuils $\Omega(r_i^j)$ et $P(r_i^j)$ respectivement. L'algorithme[1] représente la phase "Détection" de l'approche : Poseidon.

Algorithm 1: Algorithme de détection d'attaque

input : $\omega(r_i^j, t_k); \Omega(r_i^j); \rho(r_i^j, t_k); P(r_i^j)$
output: boolean
if $\omega(r_i^j, t_k) > \Omega(r_i^j)$ ***and*** $\rho(r_i^j, t_k) > P(r_i^j)$ **then**
    Output *True*
else
    Output *False*

2.5.1.2 Réaction à l'attaque

L'algorithme[2] représente la phase "réaction" de la contre-mesure : Poseidon.

Algorithm 2: Algorithme de traitement de messages

input: Incoming packet *msg* from r_i^j ;alarm_duration
$(\Omega(r_i^j))$;$(P(r_i^j))$;Scalling factor s
Alert message m from interface r_i^j

```
1  if msg is ContentObject then
2      Process msg as ContentObject
3      return
4  if msg is AlertMessage then
5      if Verify(msg.signature) and IsFresh(msg) and time from last Alert received
       from r_i^j > wait_time then
6          //push-back reaction
7          Decrease(Ω(r_i^j), s)
8          Decrease(P(r_i^j), s)
9      else
10         drop msg
11         return
12 if msg is Interest then
13     // Attack Dectection is shown in Algorithm 1
14     if AttackDetection then
15         drop msg
16         if time from last Alert received from r_i^j > wait_time then
17             // push-back alert message generation
18             send Alert to r_i^j
19     else
20         process msg
```

La table[2.2] résume les principaux avantages et inconvenants de l'approche *Poseidon*.

Avantages	Inconvénients
1. Différencie par la phase de détection entre un comportement normal et inormal des paquets de requête. 2. Impose des restrictions sur les noeuds malicieux par la phase de réaction	1. Ne supprime pas les faux paquets de requête. 2. Poseidon ne présente pas des solutions en cas où l'adversaire contrôle beaucoup de consommateurs. 3. Diminue le débit des paquets de requête qui arrivent au niveau des routeurs intermédiaires.

Table 2.2 – Les avantages et les inconvénients de la méthode *Poseidon*

2.5.2 TraceBack

La méthode « Interest traceback » proposé par HuichenDai, Yi Wang, Jindou Fan et Bin Liu [10] est très similaire à la méthode IP traceback. IP Traceback vise à trouvé la véritable origine d'un paquet IP (le trafic attaquant peut être rempli avec des adresses source usurpées), toutefois IP Traceback est jugé difficile et coûteux quand il s'agit d'implémentation.

Avec l'aide du PIT, NDN supporte le reterçage des paquets de requête (interest traceback) facilement, et cela est dû au fait que les paquets de données prennent le méme chemin que les paquets de requête.

Avec cette méthode, il sera facile de localiser l'origine du paquet de requête qui demande un contenu inexistant pour ensuite générer des paquets de données usurpés pour éliminer les paquets de requête non satisfaits depuis longtemps présents dans le PIT. Une origine suspicieuse et considérer comme une source d'attaque, la raison pour laquelle après l'avoir identifiée, une limitation du débit de l'interface connectant cet attaquant à son routeur d'accès sera imposée afin de réduire le nombre de paquets de requête entrant au réseau [11].

On peut résumer cette approche en 04 points essentiels :

- **Phase 1 :** Déclenche le processus de traceback de requête (Interest traceback) lorsque la taille du PIT augmente à un rythme alarmant ou lorsqu'elle dépasse un certain seuil.
- **Phase 2 :** Le routeur engendre des paquets de données spoofées afin de satisfaire les requêtes non satisfaites et qui ont pris beaucoup de temps dans le PIT.
- **Phase 3 :** Les paquets de données usurpés sont renvoyés à l'expéditeur en consultant le PIT des routeurs intermédiaires.
- **Phase 4 :** Affaiblir l'attaquant (e.g. limiter le débit).

La table[2.2] présente les avantages et les inconvénients sur l'approche TraceBack :

Avantages	Inconvénients
1. Permet à une victime d'identifier le chemin d'accès réseau traversé par le trafic d'attaquant. 2.Permet de supprimer toutes les fausses requêtes.	1.Les requêtes ne seront pas tous transformées en paquets de données.

Table 2.3 – Les avantages et les inconvénients de la méthode *TraceBack*

2.5.3 TokenBucket

Une des caractéristiques de base des NDN est que chaque paquet de requête n'est satisfait que par un seul paquet de données, un principe que les routeurs intermédiaires exploitent pour contrôler le trafic de données entrant, en contrôlant le nombre de paquets de requête sortants, une technique simple servirait de limiter au niveau de chaque routeur le nombre de paquets de requête transmis sur chaque interface selon la capacité physique de l'interface correspondante. Cette technique est une légère modification au célébre algorithme TokenBucket [12].

Contrairement à l'architecture des réseaux actuelle, les routeurs NDN peuvent garder la trace de la quantité de données qui peut occuper complètement le lien, et une fois la limite de capacité du lieu a été atteinte, les paquets de requêtes ne seront plus acheminés. Idéalement, le nombre de jetons pour chaque lien sera proportionnel produit bande passante-retard (BDP) [13]. Le norme de jetons peut-être formulés de tel :

$$\#\text{ de jetons} = \frac{\text{Délai * Bande Passante [Octets/s]}}{\text{Taille du paquet de données[Octets]}}$$

Où :

— **Délai :** Le temps prévu pour que le paquet de requête soit satisfait.

— **Taille de packet de données :**La taille du paquet de données de retour correspondant au paquet de requête reçu.

La table[2.4] représente les avantages et les inconvénients de l'approche TokenBucket.

Avantages	Inconvénients
1. Cette méthode est restrictive au niveau de la transmission des paquets de requêtes(rejette les paquets pour lesquels il n'y a pas de jetons).	1. TokenBucket Soutient les attaques DDoS, car si un routeur a utilisé tous ses jetons pour acheminer des paquets de requêtes malicieux, il ne pourra plus acheminer d'autres paquets de requêtes même s'ils venaient d'un utilisateur légitime, jusqu'à ce que les paquets malicieux expirent dans le PIT.

Table 2.4 – Les avantages et les inconvénients de la méthode *TokenBucket*

2.5.3.1 TokenBucket avec equité par interface

Afin de remédier à l'inconvénient mentionné précédemment, cette approche impose une équité par interface, dans le but de ne pas permettre aux paquets malicieux de consommer entièrement la bande passante d'une interface spécifique et pénaliser un utilisateur légitime [14]. Une modification a été apportée à la contre-mesure *TokenBucket* afin de corriger le manque d'équité qu'on lui reproche, cette modification consiste à assurer que les requêtes transmises par un routeur sur chaque interface représentent un mélange équitable des requêtes reçues des noeuds voisins.

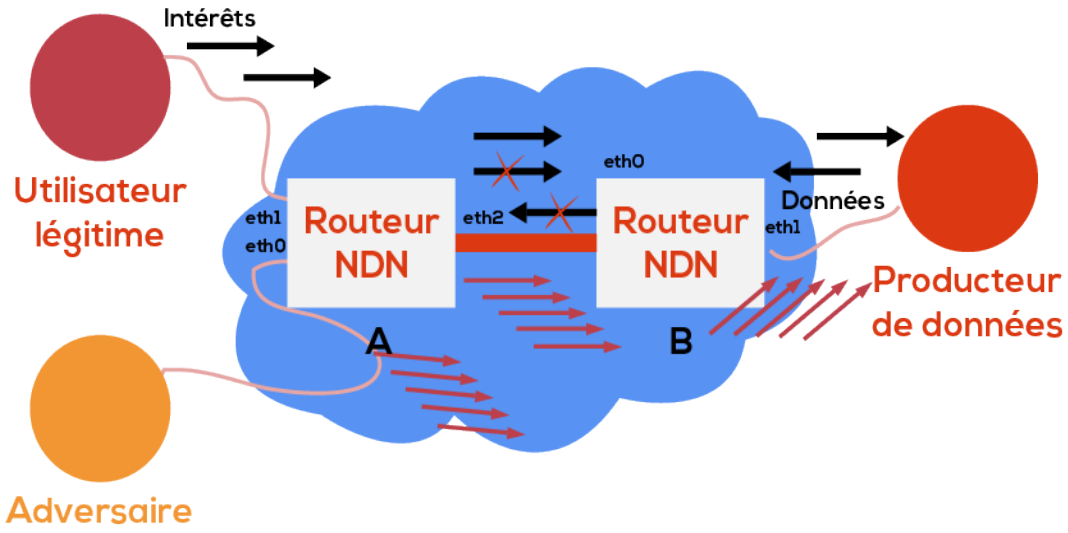

Figure 2.3 – Exemple d'une Attaque IF

Par exemple dans la figure[2.3] qui est un scénario probable d'une attaque IF, le routeur A doit assurer que les jetons associés aux paquets de requêtes envoyés sur l'interface eth2 sont équitablement distribués sur les interfaces entrantes eth0 et eth1, pour atteindre le but qui est d'assurer un mélange équitable des paquets de requête de tous les noeuds voisins, le PIT a été étendu pour supporter un drapeau pour les paquets de requêtes qui ne sont pas acheminés immédiatement et implémente une file

d'attente pour chaque interface.

Contrairement à la file d'attente habituelle, celle-là n'enregistre pas le paquet mais seulement un pointeur bidirectionnel vers l'entrée existant dans le PIT, ainsi la table sera rapidement misée à jour lorsqu'un paquet de requête sera acheminé et l'élément peut être facilement supprimé lorsqu'un paquet de requête expire.Une description formalisée de la méthode dans le pseudo-code suivant :

Algorithm 3: TokenBucket avec equité par interface

```
1  foreach Interface if do
2      L_if ← Interface Limit according to (1);
3      O_if ← 0                          ▷ Outstanding Interests on interface if
4  Function OutInterest (Interest i, InInterface in,OutInterface out)
5      if L_out − O_out > 0                          ▷ out is under limit cap then
6          O_out ← O_out+1   ▷ "borrow" a token from the bucket add out to PIT entry
           and forward i to out
7          else
8              Queue q ← out.GetSubQueue(in);
9              if Size(q) < L_out then
10                 q.PushInterest(i)
11                 add out to PIT entry, and link PIT entry with the queue
12                 else
13                     drop Interest
14                                 ▷ Whenever Lout - Oout becomes larger than Zero
15 Function TokenBecameAvailable
16     Queue q ← out.GetRoundRobinSubQueue
17     Interest ← q.PopInterest
18     update PIT entry and Forward(i, out)
19 Function InData(Data d)
20     lookup PIT entry p for Data d
21     foreach outgoing interface out in p do
22         O_out ← O_out−1;
23                                                  ▷ "return" token
24 Function TimeOut(PIT entry p)
25     foreach Interface if do
26         O_out ← O_out−1;
27                                                  ▷ "return" token
```

La table suivante présente les avantages et les inconvénients majors de la méthode TokenBucket avec équité par interface.

Avantages	Inconvénients
1. Cette approche tente de veiller à ce que chaque interface transmet sa juste part des paquets de requêtes, ce qui permet aux utilisateurs légitimes de transmettre et de recevoir leur part même dans le cas d'une attaque.	1. Dans le cas où le PIT atteint les limites, les paquets de requête qu'ils soient légitimes ou malicieux, ils seront rejetés, alors que pour que l'approche soit efficace, elle doit différencier entre les paquets légitimes et ceux qui sont malicieux.

Table 2.5 – Les avantages et les inconvénients de l'approche *TokenBucket avec equité par interface*

Pour distinguer les paquets de requête légitimes des malicieux, une fonctionnalité qui caractérise l'architecture des NDN a été mis à profit « la garantit du débit symétrique des paquets de requêtes et de données», puisque le paquet de données prend le chemin inverse du paquet de requête qui lui correspond, un routeur peut savoir si un paquet de requête a été satisfait par un paquet de données qui lui correspond ou a fini par expirer, puisque les paquets malicieux ont tendance à ne pas rapporter des paquets de données, cette information peut être utilisée pour différencier entre les paquets légitimes et malicieux.

Cette méthode de différenciation qui se base sur l'expiration des paquets est par nature réactive : il n'est pas possible de déterminer du début si le paquet de requête finira par être satisfait ou expirera. Cependant, les routeurs peuvent maintenir à jour les statistiques de la satisfaction des paquets de requêtes(nombre des paquets arrivés en fonction des paquets satisfaits) et utilisent ces statistiques pour déterminer si le paquet sera accepté ou refusé. Le pseudo-code suivant montre comment les statistiques sont calculés pour chaque interface.

Algorithm 4: Statistiques des requêtes satisfaites

foreach *Interface* ***if*** **do**
  $F_{if} \leftarrow 0$;
  ▷ forwarded Interests from interface **if**
  $\hat{F}_{if} \leftarrow 0$;
  ▷ averaged value of Fif
  $U_{if} \leftarrow 0$;
  ▷ unsatisfied Interests from interface **if**
  $\hat{U}_{if} \leftarrow 0$;
  ▷ averaged value of Uif
Function OUTINTEREST *(Interest* ***i****, InInterface* ***in****)*
  $F_{in} \leftarrow F_{in+1}$;
  record **in** in the list of incoming interfaces for **i** ;
Function INTERESTTIMEOUT*(Interest* ***i****)*
  lookup the list of incoming interfaces for **i** ;
  foreach *Interface* ***if*** *in the list* **do**
    $U_{if} \leftarrow U_{if+1}$;
    ▷ Exponentially weighted moving average smoothing
    ▷ Every second
Function EWMA
  $\alpha \leftarrow e^{-1.0/30.0}$;
  foreach *Interface* ***if*** **do**
    $\hat{U}_{if} \leftarrow \alpha.\hat{U}_{if} + (1-\alpha).U_{if}$;
    $U_{if} \leftarrow 0$;
    if $F_{if} > 0$ ▷ *To ensure decaying of ratio Uif=Fif* **then**
      $\hat{F}_{if} \leftarrow \alpha.\hat{F}_{if} + (1-\alpha).I_{if}$;
      $F_{if} \leftarrow 0$;
      ▷ Reset counters

2.5.3.2 Acceptation des requêtes basées sur les statistiques de satisfaction

Recueillir des statistiques sur les rapports de satisfaction des paquets de requête s'avère comme une astuce très importante, mais seulement en terme de détection des paquets malicieux, pour que l'approche soit plus efficace, elle doit procéder par une seconde phase réactive afin de rejeter tout paquet de requête malicieux. Cela est le prochain défi que cette approche relève.

Cette approche emploie une façon simple d'accomplir ce défi, elle utilise le rapport de satisfaction des paquets de requête comme une probabilité directe d'accepter ou de refuser les paquets au niveau de chaque routeur.

Le pseudo-code[5] montre comment les paquets sont acceptés et refusés selon les statistiques calculés par le pseudo-code[4].

Algorithm 5: Acceptation des requêtes basées sur les statistiques de satisfaction

```
1  ▷ Same init, InData and Timeout functions as in Algorithm 4
2  Function OutInterest (Interest i, InInterface in, OutInterface out)
3      ▷ Use uniform probability distribution model P(X)
4      ▷ P(X) : ∀x ∈ [0;1] ⟺ P(x) = x
5      if F_if > θ   ▷ At least some Interests were forwarded before then
6          s ← (1 − U_in/F_in) ;
7          drop interest with probability P(s) ;
8          Forward the interest, subjecting to token bucket limits ;
```

2.5.4 Le Découplage des Requêtes Malicieux par DPE

Le mécanisme DPE(Disabling PIT Exhaustion) est une approche proposée par Kai Wang, Huachun Zhou, Yajuan Qin, Jia Chen, et Hongke Zhang pour lutter contre les attaques IFA. Cette approche vise à désactiver l'épuisement du PIT causée par les paquets de requêtes malveillants. *Comme les attaques IFA visent à épuiser les ressources mémoire du PIT, l'idée principale du DPE est de détourner toutes les requêtes malicieuses en dehors du PIT, afin de le libérer de la surcharge* [15]. Ce mécanisme peut être décomposé en 03 phases :

- **Liste des paquets malicieux (m_List)**
  Dans DPE, le nombre des requêtes expirées au titre de chaque nom est surveillé à chaque fois qu'une requête arrive, cette approche impose que chaque routeur maintient une liste pour les requêtes malveillantes (appelées **m-list**)(voir la figure[2.4]) de façon a que l'orsque le nombre des paquets de requête avec un certain nom dépasse un seuil prédéfini, ce nom sera enregistré dans la *m-list*(tout nom enregistré dans cette liste est considéré comme une tentative d'attaque IF).
- **Le Marquage des Paquets**

Chaque fois qu'un paquet de requête passe par un routeur NDN, DPE lui ajoute un champ appelé *Interface_List* placé à la fin du nom du paquet de requête(voir figure[2.4]), ce champ a pour but d'enregistrer la liste des interfaces d'entrée des paquets arrivés.

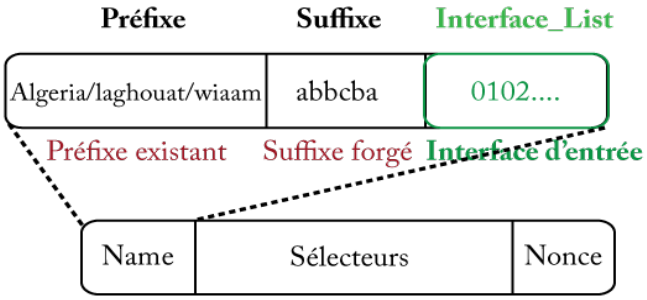

Figure 2.4 – Marquge des paquets de requête

Tout paquets dont son nom figure dans la *m_List* ne seront pas acceptés dans le PIT du routeur, mais aussi sera marqués par les interfaces d'entrées de ce routeur dans le champ *Interface_Liste*. Les paquets de requête marquée sont appelé *m_Interest*, L'orsque l'extraitm_Interest voyage vers le producteur des données, les interfaces de tous les routeurs dont il passe par seront ajoutés à *Interface_List* une par une.

— **La Transsmision des Paquets** Il existe deux types de transmission, la transmission des paquets de requête et la transmission des paquets de données. Dans ce qui suit nous allons décrire le processus de transmission des paquets lorsque l'approche DPE est implémentée dans un routeur.

1. **Transmission des paquets de requête :** DPE les paquets de requête qui ne sont pas enregistrés dans *m_List* leur transmission est la même que dans le mécanisme original de NDN, par contre pour les paquets présents dans la liste la stratégie de transmission change.

   Comme le montre la figure[2.4], chaque fois qu'un paquet de requête arrive à un routeur NDN, ce routeur regarde si il existe dans la mémoire cache (CS) un paquet de données qui correspond à cette requête, si oui le paquet de données est envoyé pour satisfaire le paquet de requête.

   Sinon le paquet de requête sera traité comme suit : le routeur regarde si le paquet est marqué par *Interface_List* ou il se trouve dans *m_List*, dans ces deux cas le paquet de requête est transmis selon la FIB du routeur sans être enregistré dans le PIT, et l'interface du routeur courant sera ajoutée dans *Interface_List*.

   Par exemple dans la figure[2.4], l'interface **1** est ajoutée à Interface_List de ce paquet de requête car ce dernier accède au routeur par cette interface. À part cela le paquet est transmis selon la stratégie originale de NDN, le routeur regarde dans le PIT pour vérifié s'il existe déjà une entrée pour le même paquet de requête, si oui la nouvelle interface d'où le paquet est

arrivé sera ajouté au PIT, mais le paquet de requête sera rejeté sans être transmis. Dans le cas contraire le paquet sera transmis au noeud suivant selon la table FIB de ce routeur, et sera aussi enregistré comme une nouvelle entrée dans le PIT.

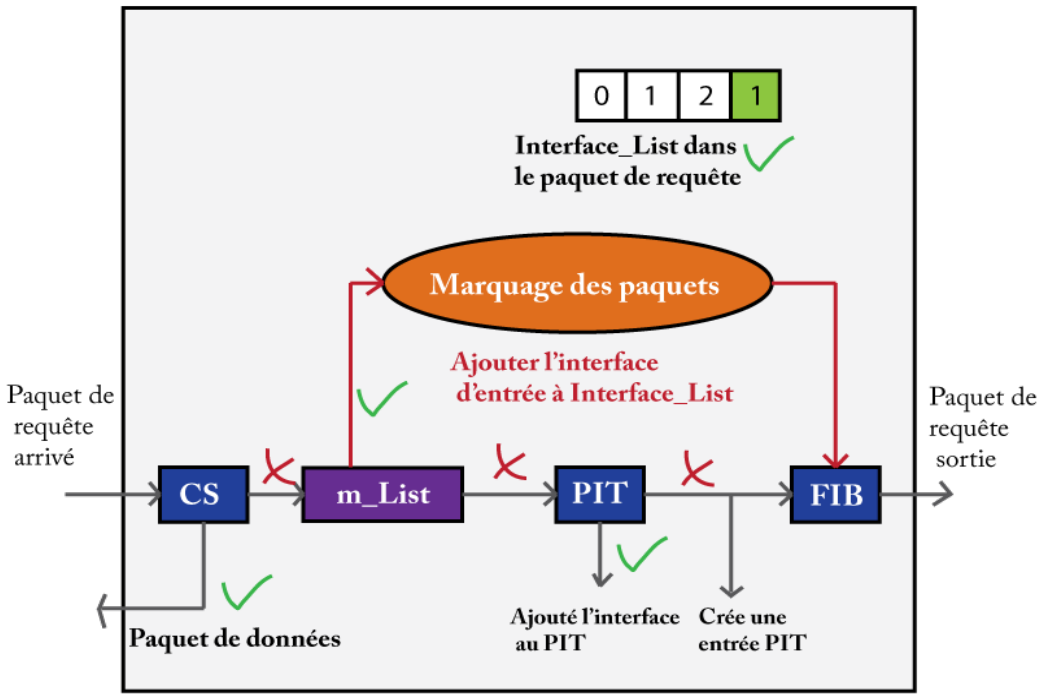

Figure 2.5 – Transmission des paquets de requête dans l'approche DPE

2. **Transmission des paquets de données :** Chaque fois qu'un paquet de données arrive à un routeur NDN, s'il existe le paquet de données dans le CS de ce routeur, ce nouveau sera rejeté sans être transmis. Sinon le routeur regarde s'il y a un champ *Interface_List* du même nom que ce paquet de données. S'il y on a, l'interface de sortie de ce paquet de données est identifié comme étant la dernière interface enregistée dans le champ *Interface_List*, et puis cette interface est supprimé du nom (du champ *Interface_List*) avant d'être transmis.

   Par exemple, dans la figure[2.5], l'interface **1** de ce routeur est supprimé de *Interface_List* du nom après être identifié comme étant l'interface de sortie de ce paquet de données. Dans un cas contraire, le paquet de données sera transmis suivant la stratégie originale de NDN. Dans ce cas le PIT du routeur est vérifié pour voir si le paquet de données a été demandé auparavant. Si le nom du paquet de données ne correspond aucune entrée dans le PIT, le paquet de données sera rejeté sans être transmis. Sinon le paquet de données est transmis selon l'interface enregistrée dans le PIT du nom qui correspond au nom de ce paquet de données, et puis cette entrée correspondante dans le PIT sera supprimé.

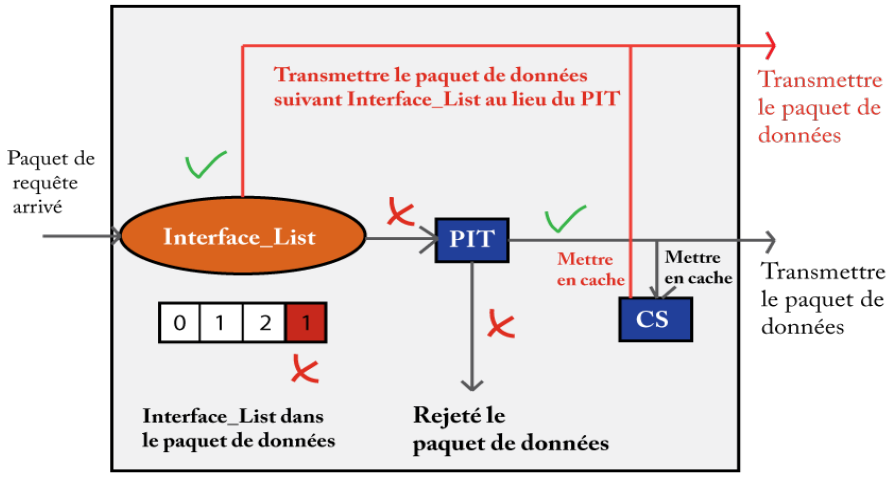

Figure 2.6 – Transmission des Paquets de données dans l'approche DPE

La table suivante résume les avantages et les inconvénients de l'approche DPE.

Avantages	Inconvénients
1. DPE permet la prévention des attaques IF. 2. DPE supprimer le faux paquet de requête.	1. Un attaquant peut visé la *m_List*

Table 2.6 – Les avantages et les inconvénients de la méthode *DPE*

Toutes les méthodes proposées jusqu'à maintenant ont l'objectif de lutter contre les attaques IFA, toutefois chacune à sa façon de détection et ça manière de réaction, le tableau suivant(voir la Table[2.7])résume les différentes méthodes mentionnées dans ce chapitre.

Approche	Type d'attaque		Type de solution		Détaction	idée	Elimination des faux paquets	Trigger	Alléviation
	IFA	STDA	Proactive	Réactive					
Poseidon2012 [5]	✓		✓		Par interface	Messages	Non	Rapport paquets de requêtes/données	Limitation du débit par interface
TraceBack 2013 [11]	✓	✓	✓		Par routeur	Taux de paquets de données	Non	Définir un seuil au PIT	Retraçage/emission de faux paquets(libérer le PIT)
TokenBucket 2013 [14]	✓		✓		Par routeur	Jetons	Non		File d'attente RR sur chaque interface
DPE 2013 [15]	✓	✓		✓	Par préfixe	Malicious List	Oui		Acheminement sans sauvgarde de paquets

Table 2.7 – Tableau comparatif des différentes approches

2.6 Conclusion

Dans ce chapitre, une partie a etait consacré à la présentation de deux différentes attaques DDoS mené dans les réseaux orientés contenu (NDN), cela a été fait de façon brève et abrégée.

Dans la seconde partie, nous avons choisi de définir quelques mesures de défense menue contre l'attaque IF avec les avantages et les inconvénients de chacune de ces méthodes.

Afin d'apporter une remédiation aux inconvénients des mesures de défense contre IFA mentionné précédemment, nous allons essayer de présenter une nouvelle approche dont ses détails seront l'objet du chapitre suivant.

Chapitre 3

NOTRE CONTRIBUTION

3.1 Introduction

Les réseaux informatiques ont connu une croissance explosive ces dernières années, spécialement en basculant d'une architecture dite TCP/IP vers une autre dites NDN. Comme mentionnée précédemment, cette architecture peut apporter beaucoup d'avantages et de corrections aux inconvénients de l'architecture TCP/IP, toutefois même les NDN ont des vulnérabilités et ne sont pas prémunis contre les attaques.

Dans le chapitre précédent, nous avons introduit les attaques STA et IFA comme étant des attaques qui exploitent la saturation du PIT, nous nous sommes focalisés sur IFA et les méthodes proposées pour lutter contre cette attaque, toutefois chacune de ces approches apporte un degré de sécurité mais aussi apporte une nouvelle vulnérabilité avec (comme la m-liste dans DPE).

Dans le même cadre de sécurité dans les réseaux orientés contenus, nous proposons dans ce chapitre notre méthode de lutte contre les attaques de l'inondation par paquets de requête, cette technique est basée fondamentalement (inspiré) sur le célèbre algorithme RED(Random Early Détection), qui repose essentiellement sur deux seuils bien définis (un seuil minimal et un autre maximal), ainsi que sur le calcul des probabilités de rejet des nouveaux paquets par rapport aux seuils (Min et Max).

Notre approche a pour but de garantir que le PIT ne se sature pas et qu'un utilisateur légitime ne soit pas pénalisé par le flot de requêtes d'un utilisateur malicieux. Plus de détails, explication et schémas feront l'objet des sections suivantes.

3.2 L'algorithme RED

L'idée de l'algorithme RED, qui a était proposé par Floyd and Jacobson [16], été la prévention du débordement du tampon par la détection précoce de la file d'attente pleine. Pour cette raison, RED utilise quatre paramètres Min_{thr}, Max_{thr}, probabilité de marquage(rejet)maximale $\mathrm{Max}_{\mathrm{pr}}$, taille de la queue Wq.

RED détecte la naissance d'un état d'encombrement par le calcul de la moyenne de la taille de la fille d'attente désigné ici par *Avr* et alertes les expéditeurs qu'une congestion s'est produite en laissant tomber les paquets délibérément à une certaine probabilité désignée ici par P(Avr), Avr et P(Avr) sont définis par les équations suivantes :

$$Avr = (1 - Wq)Avr + Wq \tag{3.1}$$

$$P(Avr) = \frac{Avr - Min_{th}}{Max_{th} - Min_{th}} * Max_p \tag{3.2}$$

Si Avr est inférieur à Min_{thr}, alors la probabilité de rejet du paquet serait nulle, par contre si Avr serait supérieur à Max_{thr}, la probabilité de rejet ici sera égale à un. Maintenant si Avr est supérieur à Min_{thr} et inférieur à Max_{thr}, dans ce cas la probabilité de rejet des paquets sera égale à $P(Avr)$.

3.3 Principe de fonctionnement

Notre contribution a pour but de garantir que le PIT ne se sature pas par des paquets qui occuperont l'espace sans jamais se satisfaire(DDoS Attack), il faudrait donc accepté ou rejeté ces paquets. L'idée est de définir trois intervalles, à la réception de chaque paquet de requête, un test sera exécuté pour déterminer à quel intervalle appartient ce paquet.

Le 1er intervalle est là où tous les paquets de requêtes reçus sont acceptés (car le PIT est loin d'être en danger de saturation), le 2éme intervalle sont celui auquel il faut rejeter n'importe quel paquet de requête reçu (car le risque de saturer le PIT est très élevé). Le dernier intervalle est là où des tests sur quelques aspects caractérisant les NDN(Préfixe, Interface, PIT, etc.) seront effectués, dans le but de définir l'opération à effectuer sur le paquet reçu[voir Figure(3.1)].

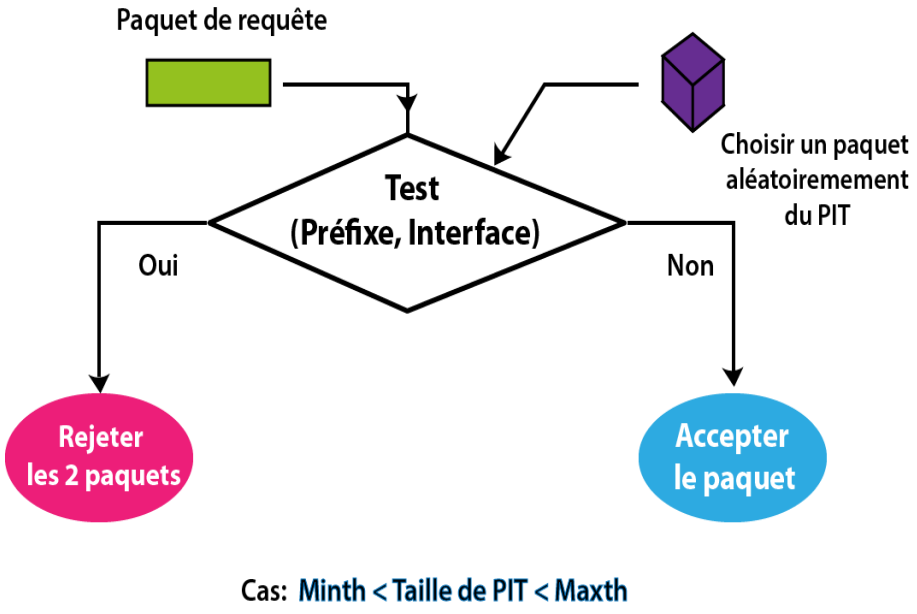

Figure 3.1 – Principe de fonctionnement

3.3.1 Détails

Dans cette section nous allons décrire notre approche qui permet d'atténuer les attaques de l'inondation par paquets de requête dans les réseaux orientés contenus, cette technique s'exécute au niveau des routeurs et vise dans un premier lieu à éviter que le PIT du routeur se sature par des paquets de requêtes, d'une autre part grâce au nom du contenu(names), l'algorithme permet de faire une distinction entre les paquets légitimes de ceux qui sont malicieux pour enfin rejeter ces derniers.

Afin de prévoir la saturation du PIT, à l'arrivée de chaque paquet l'algorithme calcule la moyenne de la taille du PIT, qui est désignée ici par *AvrPITsize*(voir étape 1 de la figure[3.2]), *AvrPITsize* sera ensuite comparée à deux seuils, minimum threshold *minth* et à maximum threshold *maxth*, Lorsque la taille moyenne du PIT est inférieure à *minth*, le paquet arrivé sera accepté, tandis que lorsqu'elle dépasse *maxth*, le paquet sera rejeté, le dernier des cas et celui où *AvrPITsize* est entre les deux seuils, dans ce cas un des paquets parmi ceux déjà présents dans le PIT sera aléatoirement sélectionné et son préfixe sera comparé avec le préfixe du nouveau paquet(voir étape 2 et 3 de la figure[3.2]).

Dans les mesures où les deux paquets ont des préfixes différents ,le nouveau paquet de requête est admis, dans le cas contraire, un test d'interface entre le paquet aléatoire du PIT et le nouveau *paquet de requête* sera effectué.

Si les deux paquets ne proviennent pas de la même interface, une probabilité de rejet au sujet du nouveau paquet sera calculé, cette probabilité est dénotée par *Pdrop* et calculée selon l'équation 3, *au cas où Pdrop est supérieure à 1/2 le paquet sera rejeté, sinon il sera accepté*(le texte italique est dénoté scénario).

En revanche si les deux paquets (paquet de requête arrivée et le paquet sélectionné du PIT) proviennent de la même interface, un rapport $paquetsarrives/paquetssatisfaits$ sera calculé et comparé à un seuil *threshold* déterminé par rapport aux deux seuils *Maxth* et *Minth*, si le rapport calculé est inférieur au seuil, la probabilité *pdrop* sera calculée et [scénario] sera à nouveau exécutée, sinon les deux paquets seront rejetés.

$$AvrPITsize = (TaillePit + AvrPITsize(courant))/2 \tag{3.3}$$

$$pdrop = Pdmax * (AvrPITsize – Minth)/(Maxth – Minth) \tag{3.4}$$

Où :

- **Pdmax :** Est une probabilité maximale de rejet de paquet.
- **AvrPITsize :** Est la moyenne pondérée de la taille du PIT.
- **Maxth :** Est le seuil maximum duquel en dessus tous les paquets doivent être rejetés.
- **Minth :** Est le seuil minimum duquel en dessous tous les paquets doivent être acceptés.

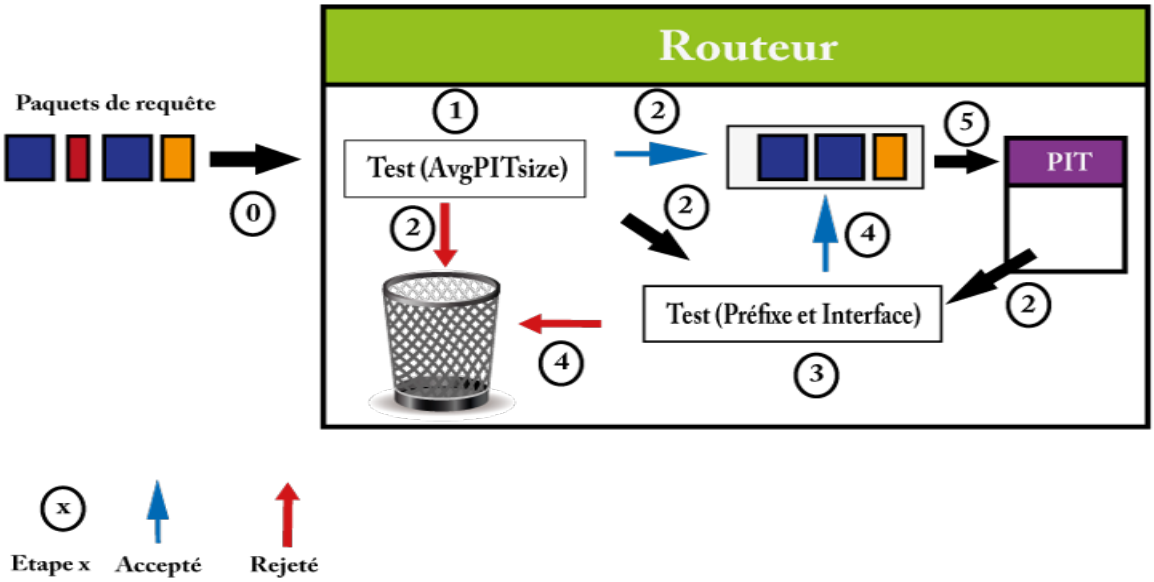

Figure 3.2 – Comportement du *Routeur* à l'arrivée des paquets

L'algorithme 1 reprend l'essentiel de notre approche proposé et du comportement du routeur à l'arrivée de chaque paquet.

Algorithm 6: Traitement du paquet arrivé

input : paquet de requête(PR),paquet aléatoire(PA),pdmax,Minth,Maxth
Output: AvrPITsize,pdrop

```
1  begin
2      if paquet = paquet de donnée then
3          Traitement du paquet de données
4      if paquet = paquet de requéte (PR) then
5          AvrPITsize = (TaillePit + AvrPITsize(courant))/2
6          if AvrPITsize > Maxth then
7              Rejeté le paquet (PR)
8          if PITsize <= Minth then
9              Accepté le paquet (PR)
10         if (AvrPITsize > Minth) And (AvrPITsize < Maxth) then
11             Choisir un paquet du PIT alétoirement (PA)
12             if (Préfixe(PR) ≠ Préfixe(PA)) then
13                 Accepté le paquet(PR)
14                 else
15                     if (Interface(PR)= Interface(PA)) then
16                         RaportRejet = # PaquetsArrivées / # PaquetsSatisfaits
17                         if RaportRejet >th then
18                             Rejet(PR,PA)
19                             else
20                                 pdrop =
                                   Pdmax ∗ (AvrPITsize−Minth)/(Maxth−Minth)
21                                 if pdrop > 0.5 then
22                                     Rejeté (PR)
23                                 else
24                                     Accepté (PR)
```

3.4 Évaluation de la méthode

Lors du développement d'une approche pour lutter contre les attaques DDoS dans les réseaux orientés contenus, la simulation par logiciel du comportement et des performances de cette approche est essentielle. Plusieurs facteurs influencent le comportement de la méthode comme la topologie, le débit, le choix des paramètres de l'algorithme(Minth, Maxth), la taille du PIT...etc. Un simulateur qui réussit à modéliser le mieux possible ces facteurs d'influence, permettra forcément une meilleure représentation du comportement réel du réseau. Le simulateur le mieux adapté pour évaluer un tel algorithme est ndnSIM qui est principalement fondé sur ns-3 et spécialisé dans la simulation des NDN.

3.4.1 ndnSIM

NdnSIM est un simulateur open source pour les NDN, il a été mis au point par les dévlopeurs de UCLA (University of California Los Angelos) en 2011, qui se sont basé également sur le simulateur NS-3 [17].
La conception de ndnSIM cible les buts suivants :

- Être open source pour permettre à la communauté de recherche d'exécuter leurs expérimentations dans une plate-forme de simulation connue.
- Être capable de simuler toutes les opérations de base des protocoles NDN.
- Maintenir au niveau du paquet l'interopérabilité avec l'implémentation de CCNx [1], pour permettre le partage de la mesure du trafic et des outils d'analyse de paquets entre CCNx et ndnSIM.
- Être capable de supporter une simulation à grande échelle des expérimentations .
- Faciliter les expérimentations au niveau de la couche réseau avec le routage, le caching des données, l'acheminement des paquets, et la gestion des congestions.

Suivant l'architecture des NDN, le simulateur ndnSIM est implémenté comme étant un nouveau modèle de protocole au niveau de la couche réseau [18], utilisant un ensemble séparé de calss C++, pour définir le comportement de chaque entité : Pending Interest table(PIT), Forwarding Information Base(FIB),Content Store(CS), les applications et les interfaces réseau, etc.[voir Figure(3.3)].

1. CCNx est l'implementation de Content Centric Networks par PARC.

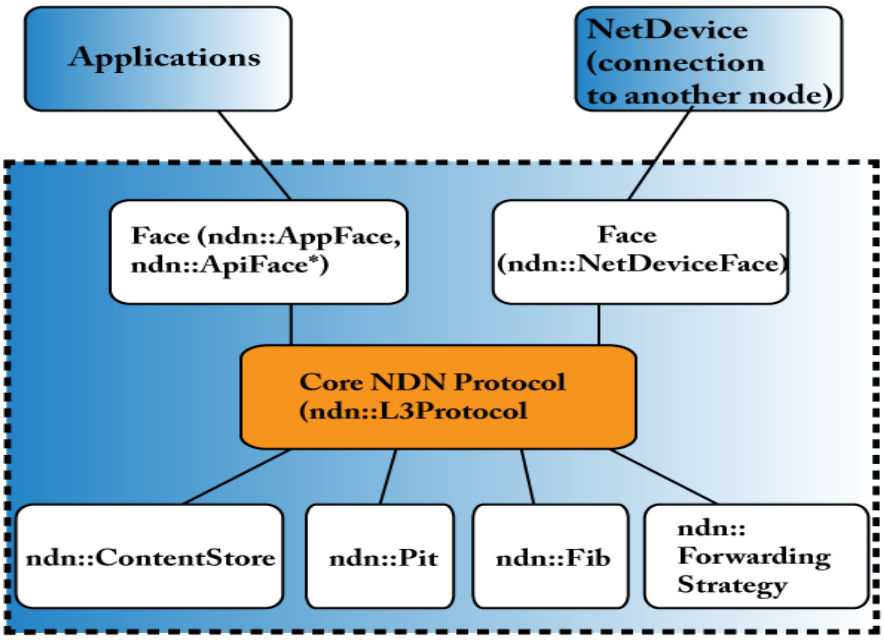

Figure 3.3 – Les composants de ndnSIM

Le design de ndnSIM suit la philosophy des simulations en NS-3, la liste suivante résume les composants abstraits implémentés en ndnSIM.

3.4.1.1 ndn : :L3Protocol

Permet de recevoir des paquets de requêtes et de données des couches supérieures et inférieures à travers ndn : :Face.

3.4.1.2 ndn : :Face

Abstraction pour permettre des communications uniformes avec des applications (NDN : :AppFace) et d'autres noeuds simulés (NDN : :NetDeviceFace) avec le soutien pluggable (et optionnel) des mesures d'évitement de la congestion au niveau de la couche liaison.

3.4.1.3 ndn : :ContentStore

Abstraction pour le stockage en réseau (par exemple : transitoire à court terme, transitoire à long terme, permanent à long terme) pour les paquets de données.

3.4.1.4 ndn : :Pit

Abstraction pour la table des requêtes en attente (PIT) qui assure le suivi (par préfixe) des interfaces sur lesquelles les paquets de requêtes ont été reçues , des interfaces sur lesquelles les paquets de requêtes ont été transmises.

3.4.1.5 ndn : :Fib

Abstraction de la base de l'acheminement des informations(FIB), qui peut être utilisée pour guider l'acheminmenet des paquets de requêtes par les stratégies d'acheminement.

3.4.1.6 ndn : :ForwardingStrategy

Abstraction et implémentation du noyau de l'acheminement des paquets de requêtes et de données. chaque étape dans le processus d'acheminement dans ndn : :ForwardingStrategy y compris la recherche dans le CS, PIT, FIB et acheminement des paquets de données suivant les entrées PIT est représenté comme un appel virtuel de fonction, qui peut être modifié dans une stratégie d'acheminement particulier.

3.4.2 Paramétres de simulation

Afin de réaliser la simulation de notre approche et de l'étudier, nous avons fixé les paramètres mentionnés dans le tableau[3.1] et cela a été fait par rapport à certains paramètres fixés par le simulateur ndnSIM(comme la taille du PIT). Pour de meilleurs résultats, la simulation a été faite sur une topologie tree(racine, 6 noeuds), avec un débit fixe de 10 Mpbs et des paquets de 1 KB.

Topologie	Arbre
Débit	10 Mbps
Taille des paquets	1 KB
Nombre de consommateurs	4
Nombre de producteurs	1
Nombre de noeuds malicieux	2
Nombre de noeuds légitimes	2
Temps de simulation	20 s
Min_{th}	200 paquets
Max_{th}	800 paquets
Threshold(th)	33.33 %
Lancement de l'algorithme	A partir de 10s

Table 3.1 – Paramétres de simulation

3.4.3 Topologie du réseau

Notre simulation a été appliquée sur une topologie de type arbre (voir figure[3.4]).

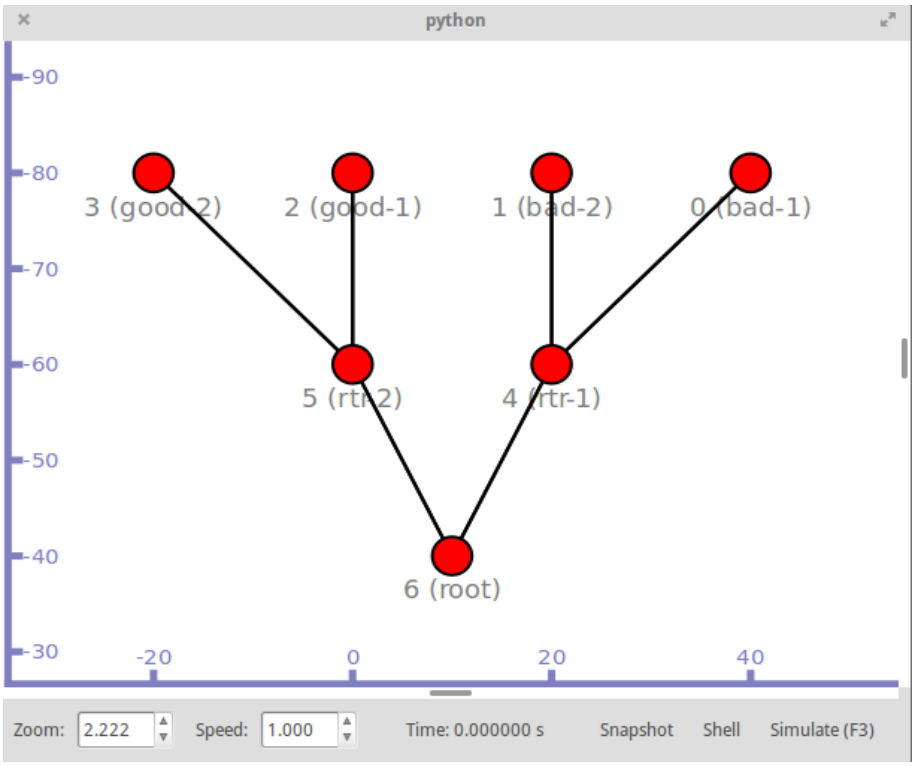

Figure 3.4 – Topologie du réseau simulé

La topologie ci-dessus a été choisis car elle met en valeur l'attaque et montre les effets de l'approche.
Une telle topologie de petite taille et simple, nous permet d'annuler des effets complexes et avoir des interprétations de résultats qui sont clairs.

Dans la figure[3.4] nous avons choisi Racine, Rtr,malicieux et légitime pour dénoter le producteur, routeur intermédiaire, noeud contrôlé par l'adversaire(produit des paquets malicieux) et consommateur, respectivement.

3.4.4 Scénario de simulation

Tout d'abord nous analysons la topologie sans trafic adverse. Chaque consommateur émet des paquets de requête pour un contenu produit par le noeud "Racine".

Les consommateurs légitimes initient le trafic avec une courte rafale de 50 paquets de requête, et les consommateurs malicieux avec une rafale de 300 paquets, espacées de 20 ms, au temps t=1s de la simulation. Nous avons établi des routeurs avec une taille de PIT à 1000 Ko, tandis que le temps d'expiration du paquet de requête a été réglé sur le délai par défaut de 4s.

Nous rapportons la moyenne des différentes pistes de la figure[3.5] montre l'utilisation de PIT (axe y) en fonction du temps de simulation(axe x), tandis que la figure[3.6] montre le nombre total de tous paquets de requêtes rejetés (axe des y) en fonction des paquets de requête malicieux (axe des x).

3.4.5 Résultats et interprétations

Avec les paramètres définis précédemment, nous allons dans cette partie présenter la simulation qui correspond à ces paramètres ainsi que l'interprétation des résultats obtenus.

3.4.5.1 Taille du PIT en fonction du temps

La figure ci-dessous, montre le nombre de paquets dans le PIT avant et après le lancement de l'approche, sachant que les noeuds 0 et 1 sont des consommateurs malicieux(génèrent des paquets de requête qui n'ont pas des paquets de données correspondants), les noeuds 2 et 3 sont des consommateurs légitimes, 4 et 5 des routeurs intermédiaires et le noeud 6 est le producteur.

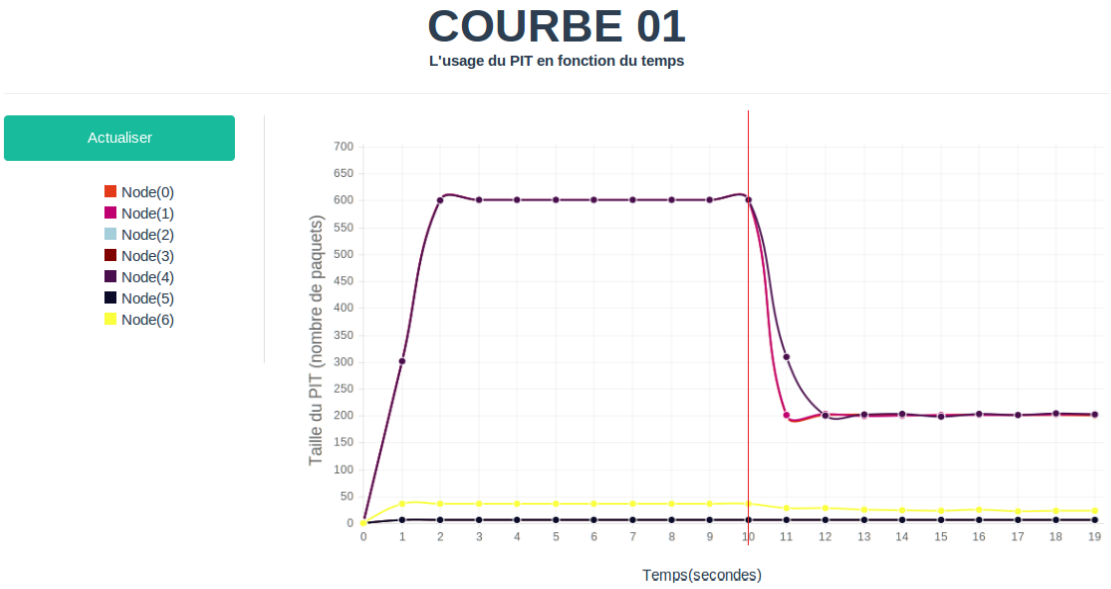

Figure 3.5 – Taille du PIT en fonction du Temps

La courbe de la figure[3.5] nous permet de remarquer que la taille du PIT était en progression continue pour les noeuds 1 et 4 (consommateur malicieux, routeur intermédiaire entre producteur et le noeud 1), et cela est avant les 10 premières secondes, dès le lancement de l'algorithme, la taille du PIT Besse et cela est dû à :

— L'élimination des paquets malicieux (grâce au test de comparaison des paquets aléatoirement sélectionnés du PIT aux paquets de requête qui arrivent).

— L'élimination de tous les paquets dès que la taille du PIT s'approche du seuil maximal.

3.4.5.2 Nombre des paquets rejetés en fonction du nombre des paquets malicieux

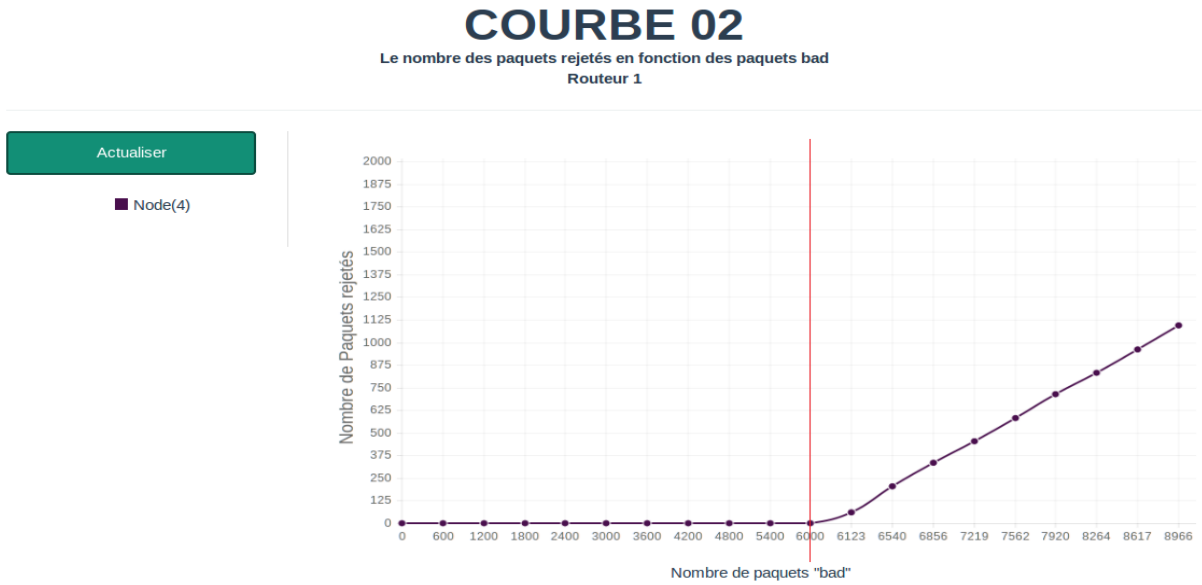

Figure 3.6 – Nombre des paquets rejetés en fonction du nombre des paquets malicieux

La courbe de la figure[3.6] montre que le nombre de paquets malicieux rejetés était null et que le PIT accepte tous les paquets sans faire distinction entre ceux qui sont légitimes et ceux qui ne le sont pas.

À partir de la 10éme seconde, ce qui coïncide la réception de 6000 paquets par le routeur intermédiaire(noeud 4), le rejet des paquets malicieux commence, nous remarquons que la courbe progresse de façon linéaire ce qui nous permet de dire que la majorité des paquets malicieux n'accèdent pas à la table PIT.

Conclusion et Perspective

Dans ce mémoire nous avons tout d'abord introduit les NDN et les aspects qui caractérisent cette nouvelle architecture, puis nous avons présenté les attaques qui peuvent viser ce type de réseaux en se focalisant essentiellement sur les attaques IF, ce qui nous a poussé à présenter et comparer les approches proposées pour lutter contre cette attaque.

Dans un cadre de dévlopement et de prévention contre IFA nous avons proposé notre solution préventive accompagnée des résultats de simulation et leur interprétation. Ce mémoire nous a permis de :

- De se plonger dans cette nouvelle architecture et d'explorer un peu ses points forts et ses vulnérabilités.
- De comprendre les attaques DDoS qui peuvent viser les NDN et les dangers qu'ils présentent.
- D'apprendre à manipuler le simulateur ndnsim et d'améliorer notre maîtrise de NS-3.

Comme travaux futurs, nous pensons à améliorer notre approche en travaillant sur le choix des paquets choisi du PIT pour être comparé avec le paquet de requête nouvellement arrivé au routeur, aussi nous pensons faire une étude sur des topologies variées et plus étendues que celle choisis dans ce mémoire.

Bibliographie

[1] David Geer. Named Data Networking project wants to retire TCP/IP. http://searchnetworking.techtarget.com/feature/Named-Data-Networking-project-want 2015. [Online ; accessed 20-Avril-2016].

[2] NDN Team. NDN Frequently Asked Questions (FAQ). http://named-data.net/project/faq/. [Online ; accessed 2-mai-2016].

[3] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3) :66–73, 2014.

[4] NDN Team. Named Data Networking :Motivation & Details. http://named-data.net/project/archoverview/. [Online ; accessed 22-Avril-2016].

[5] Alberto Compagno, Marco Conti, Paolo Gasti, and Gene Tsudik. Poseidon : Mitigating interest flooding ddos attacks in named data networking. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pages 630–638. IEEE, 2013.

[6] Van Jacobson, Jeffrey Burke, Lixia Zhang, Beichuan Zhang, Kim Claffy, Christos Papadopoulos, Tarek Abdelzaher, Lan Wang, J Alex Halderman, and Patrick Crowley. Named data networking next phase (ndn-np) project may 2014-april 2015 annual report. 2014.

[7] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, James D Thornton, Diana K Smetters, Beichuan Zhang, Gene Tsudik, Dan Massey, Christos Papadopoulos, et al. Named data networking (ndn) project. *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[8] Steven DiBenedetto, Christos Papadopoulos, and Daniel Massey. Routing policies in named data networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pages 38–43. ACM, 2011.

[9] Moti Yung. *Applied Cryptography and Network Security*. Springer, 2010.

[10] Alberto Compagno, Marco Conti, Paolo Gasti, and Gene Tsudik. Poseidon : Mitigating interest flooding ddos attacks in named data networking. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pages 630–638. IEEE, 2013.

[11] Huichen Dai, Yi Wang, Jindou Fan, and Bin Liu. Mitigate ddos attacks in ndn by interest traceback. In *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pages 381–386. IEEE, 2013.

[12] Weibin Zhao, David Olshefski, and Henning G Schulzrinne. Internet quality of service : An overview. 2000.

[13] Alexander Afanasyev, Neil Tilley, Peter Reiher, and Leonard Kleinrock. Host-to-host congestion control for tcp. *Communications Surveys & Tutorials, IEEE*, 12(3) :304–342, 2010.

[14] Alexander Afanasyev, Priya Mahadevan, Ilya Moiseenko, Ersin Uzun, and Lixia Zhang. Interest flooding attack and countermeasures in named data networking. In *IFIP Networking Conference, 2013*, pages 1–9. IEEE, 2013.

[15] Kai Wang, Huachun Zhou, Yajuan Qin, Jia Chen, and Hongke Zhang. Decoupling malicious interests from pending interest table to mitigate interest flooding attacks. In *Globecom Workshops (GC Wkshps), 2013 IEEE*, pages 963–968. IEEE, 2013.

[16] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *Networking, IEEE/ACM Transactions on*, 1(4) :397–413, 1993.

[17] Spyridon Mastorakis, Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnsim 2.0 : A new version of the ndn simulator for ns-3. Technical report, Technical Report NDN-0028, NDN, 2015.

[18] Ilya Moiseenko Alexander Afanasyev, Spyridon Mastorakis. ndnsim documentation "introduction". http://ndnsim.net/2.1/intro.html, 2011. [Online ; accessed 1-Avril-2016].