MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Saad Dahleb de Blida 1
Faculté des sciences

Département d'informatique

Mémoire de fin d'étude

Pour l'obtention du diplôme de master en informatique

Spécialité : Sécurité des systèmes d'informations

THEME :
Développement d'une plateforme CCN-SOA-SECURE

Réalisé par

KANTÉ Coumba Cheicknè
DIARRA Amadou

Promotrice : Mme Arkam Meriem

Année universitaire : 2020-2021

Résumé

Les réseaux centrés contenus, soit Content-Centric-Network CCN) en anglais, sont de nouveaux types de réseau émergeant tentant avec de nouveaux protocoles de routage de transmettre les données le plus rapidement possible sans se soucier de leur emplacement, ce dernier n'intéressant aucunement les utilisateurs finaux.

Notre travail consiste à adapter ces réseaux aux architectures orientées service. Ces derniers ayant pour fonctions la résolution de problèmes particuliers en organisant les systèmes d'information en composants logiciels nommés services pouvant inter-opérer pour offrir plus de solutions, problèmes que peuvent rencontrer les CCN.

Dans ce mémoire, nous proposons une architecture que nous avons nommée : CCN-SOA. Cette approche est basée sur l'utilisation des concepts de base des réseaux CCN en les adaptant de telle sorte qu'ils puissent supporter de la manière la plus légère possible les concepts de base de SOA.

Tout système informatique a ses failles et ses vulnérabilités, le CCN étant une architecture émergeante, n'y fait pas exception. C'est dans cette optique que nous proposerons aussi une solution pour sécuriser notre architecture d'où le nom : CCN-SOA-SECURE contre l'attaque de l'homme du milieu passif.

Mots clés : réseaux centrés contenus (CCN), réseaux centrés informations (ICN), architecture orientée service (SOA), cryptographie homomorphe, RSA.

Abstract

Content-centric networks, or Content-Centric-Network CCN), are new types of networks emerging that attempt with new routing protocols to transmit data as quickly as possible without worrying about where it is, the latter of no interest to end users.

Our job is to adapt these networks to service oriented architectures. The latter's functions are to solve particular problems by organizing information systems into software components called services that can interoperate to offer more solutions, problems that may be encountered by the CCNs.

In this thesis, we propose an architecture that we have named: CCN-SOA. This approach is based on using the basic concepts of CCN networks by adapting them so that they can support the basic concepts of SOA in the lightest possible way.

Every computer system has its flaws and vulnerabilities, and CCN being an emerging architecture is no exception. It is with this in mind that we will also propose a solution to secure our architecture, hence the name: CCN-SOA-SECURE against the passive attack of Man-in-the-Middle.

Keywords: content-centric networks (CCN), information-centric networks (ICN), service-oriented architecture (SOA), homomorphic cryptography, RSA..

Remerciements

Avant toute chose, nous remercions Allah, le tout miséricordieux, le très miséricordieux de nous avoir donné la force et la volonté pour mener ce travail jusqu'au bout.

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes auxquelles nous voudrions témoigner toute notre gratitude.

On remercie du fond du cœur Madame ARKAM Meriem pour son suivi et son apport scientifique, sa disponibilité et ses conseils qui nous ont été extrêmement utiles lors de l'élaboration de ce mémoire.

Nos remerciements vont à l'endroit de toute l'équipe pédagogique de l'université SAAD Dahleb de Blida pour leurs enseignements.

Un grand merci aux membres du jury qui ont pris le temps de juger notre modeste travail, nous leur exprimons l'expression de notre profonde gratitude.

Nous remercions tout naturellement nos parents, qui durant toutes ces années n'ont pas arrêtés nous de soutenir que ce soit moralement ou financièrement, merci à eux pour tout l'amour qu'ils nous donnent, c'est une source de motivation intarissable.

Enfin, nous remercions toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce travail, plus particulièrement nos frères et sœurs des cités universitaires 3, 5, 6 et 7.

Merci à toutes et à tous !!!

Table des matières

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

Acronyme	***Intitulé***
CCN	*Content Centric NetworK*
ICN	*Information Centric Network*
DONA	*Data Oriented Network Architecture*
NDN	*Named Data Network*
NETINF	*Network of Information*
PSIRP	*Publication Subscribe Internet Routing*
PURSUIT	*Publish Subscribe Internet Routing Paradigm*
SAIL	*Software Architecture Integration Library*
RN	*Rendez vous Node*
RENE	*Réseau de rendez vous*
TM	*Topology Manager*
DHT	*Destributed hash table*
CONET	*Cooperative Wireless Communications and Networking*
EN	*End Node*
SN	*Serving Node*
BN	*Border Node*
IN	*Internal Node*
NSN	*Named System Node*
IPSEC	*Internet Protocol Security*
FIB	*Forwarding Information Base*
PIT	*Pending Interest Table*
CS	*Content Store*

DDOS	*Distributed Denial of Service*
PARC	*Palo Alto Research Center*
MONET	*Mobile and Network Laboratory*
KNU	*Université nationale de Kyungpook*
SOA	*Service Oriented Architecture*
IT	*Information Technology*
SOAP	*Simple Object Access Protocol*
WSDL	*Web Service Description Language*
UDDI	*Universal Description, Discovery and Integration*
XML	*Extensible Markup Language*
http	*Hypertext Transfer Protocol*
RPC	*Remote Procedure Call*
SMTP	*Simple Mail Transfer Protocol*
W3C	*World Wide Web Consortium*
FTP	*File Transfer Protocol*
URL	*Uniform Resource Locator*
IBM	*International Business Machines*
API	*Application Programming Interface*
HP	*Hewlett Packard*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
ADS	*Advertisement and Discovery of Services Protocol*
ebXML	*Electronic Business-Extensible Markup Language*
Daml-S	*DARPA Agent Markup Language for Services*
SAWSDL	*Semantic Annotation for WSDL*
SOAP-RP	*Simple Objet Access Protocol-Rounting Protocol*

SAML	*Security Assertion Markup Langage*
BTP	*Business Transaction Protocol*
XACML	*eXtensible Access Control Markup Language*
XPDL	*XML Process Definition Language*
WSFL	*Web Service Flow Langage*
WSCL	*Web Service Conversation Langage*
TPAML	*Trading Partner Agreement Langage*
WSBPEL	*Web Service Business Process Execution Language*
REST	*RestFul*
JSON	*JavaScript Objet Notation*
CSV	*Comma Separated Values*
RSS	*Really Simple Syndication*
WADL	*Web Application Description Language*
IOT	*Internet of Things*
SCN	*Service Centric Network*
INET	*Institut National des Etudes Territoriales*
RSA	*Rivest-Shamir-Adleman*
IBC	*Identity-Based Cryptographic*
PKG	*Public Key Generator*
IBS	*Identity-Based Signature*
IBE	*Identity-Based Encryption*

Introduction générale

Depuis les années 2010, l'internet a connu une importante croissance de connectivité partout dans le monde. Et ce taux de connectivité augmente de jour en jour, augmentant ainsi la quantité de données à transporter dans le réseau.

Comme le remarquait Van Jacobson, l'un des pères de l'internet et celui qui a proposé l'architecture CCN, le concept ICN propose un basculement fondamental par rapport au fonctionnement actuel de l'internet : on passerait d'une connectivité permanente à une connectivité plus dynamique, s'adaptant aux changements de l'utilisateur et du contenu **[39].**

Contrairement aux réseaux TCP/IP qui accordent une importance particulière à la localisation des hôtes et des contenus, les CCN font une abstraction de cette localisation en partant du postulat selon lequel elle n'a pas de nécessité pour les utilisateurs parce que tout ce qui intéresse ces derniers est le contenu. Les protocoles de routage des CCN se basent sur les contenus afin de les livrer en des délais très courts.

En plus d'une forte demande de connectivité, demandes d'autant plus fortes pour les entreprises, subsistait une problématique d'interopérabilité entre les différents systèmes d'information de chacune d'elles parce qu'aucune entreprise ne peut seule mettre en place tous les logiciels informatiques dont elle a besoin pour son fonctionnement quotidien. Même si elle est plus expressive pour les entreprises, cette problématique s'impose également aux utilisateurs particuliers notamment au niveau des couts d'accès et d'utilisation d'un logiciel.

Pour résoudre ce problème, les experts en système d'information proposèrent au début des années 2000, les architectures orientées service (SERVICE ORIENTED ARCHITECTURE) basés sur des composants logiciels appelés services qui en plus de résoudre les problèmes d'interopérabilité, les résolvent d'une manière dynamique et permettent également de combiner ces services pour avoir des services plus évolués avec plus de fonctionnalités.

Entre les demandes de service de streaming vidéo qui représentent aujourd'hui les contenus les plus demandés et les autres demandes de service, les SOA font face à un problème de transport de quantité importante de contenus ce qui explique que les travaux ayant pour but de faire fonctionner ensemble les SOA et les réseaux CCN ont commencé assez tôt après le lancement des ICN.

Les littératures sur cette question, d'adapter les CCN aux SOA sont nombreuses mais les propositions faites soulèvement également d'autres problématiques. Ainsi dans ce mémoire nous

ferons une proposition sur comment adapter les CCN aux SOA, que nous nommons : l'architecture CCN-SOA.

Nous avons comme objectif majeur de rendre notre architecture facile d'implémentation, légère et sure.

Nous mettons un accent particulièrement important sur la sécurité parce que les CCN ont des failles non négligeables mettant en danger la disponibilité, la confidentialité et l'intégrité des données. Les CCN étant le soubassement de notre architecture, traiter cette question de sécurité est nécessaire.

Pour atteindre notre objectif, qui est d'adapter le CCN à l'architecture SOA tout en assurant la sécurité des données, nous avons divisé ce mémoire en quatre chapitres :

- Le chapitre I portera sur une étude des réseaux centrés sur l'information et plus particulièrement des réseaux CCN.
- Le deuxième chapitre sera des architectures SOA, nous parlerons des éléments essentiels de ces architectures notamment ce qui nous sont indispensables pour notre travail.
- Le chapitre III, quant à lui, portera sur notre architecture CCN-SOA où on formulera en premier les éléments théoriques pour son fonctionnement et on mettra en pratique ces théories à la fin du chapitre grâce à la simulation.
- Le chapitre IV qui très technique contient les éléments sur la sécurité de notre architecture mais nous y parlons également de concepts comme la cryptographie qui a nous d'ailleurs permis de sécuriser notre architecture.

Chapitre I: Architectures des réseaux centrés sur l'Information

I.1. Introduction

L'ICN est une approche ayant pour but de faire évoluer le concept sur lequel est basé l'internet actuel. Le concept de l'internet d'aujourd'hui est basé sur l'emplacement du l'information alors que celui des réseaux ICN est basé sur l'information elle-même indépendamment de son emplacement. Après la proposition du concept ICN, les chercheurs ont proposé plusieurs architectures comme la DONA, LE NDN/CCN, PURSUIT, NETINF, etc. Chacune de ses différentes architectures a ses propriétés et caractéristiques mais aussi le même but, celui de satisfaire les demandes plus rapidement et efficacement en ne se basant que sur l'information.

Dans ce chapitre, nous parlerons d'abord des réseaux ICN, ensuite nous parlerons des architectures ICN les plus connues, et faire une comparaison entre celles-ci, pour finir, nous détaillerons un peu plus les réseaux CCN.

I.2. Réseaux centrés sur l'information (ICN)

L'accroissement de l'utilisation de l'internet d'aujourd'hui a apporté certains besoins que les fonctions de base de celui-ci ne pourront peut-être plus combler efficacement car les données transportées seront beaucoup trop grandes.

Le concept sur lequel sont basés les réseaux centrés sur l'information (ICN) est que le plus important n'est pas l'endroit où peut se trouver l'information mais bien l'information elle-même.

Contrairement à l'adressage IP, les réseaux ICN (réseaux centrés sur l'information) mettent l'accent sur l'information en le rendant directement adressable et routable. L'information, elle-même devient le cœur de la communication.

I.3. Architecture des réseaux centrés sur l'information

Dans cette partie, nous donnons un aperçu de cinq approches ICN en réseau, à savoir CCN, NDN, DONA, NETINF et PURSUIT comme présenté dans la **figure 1**. Pour pouvoir fournir aux lecteurs une compréhension plus générale, une description plus détaillée de ces architectures y est illustrée plus bas.

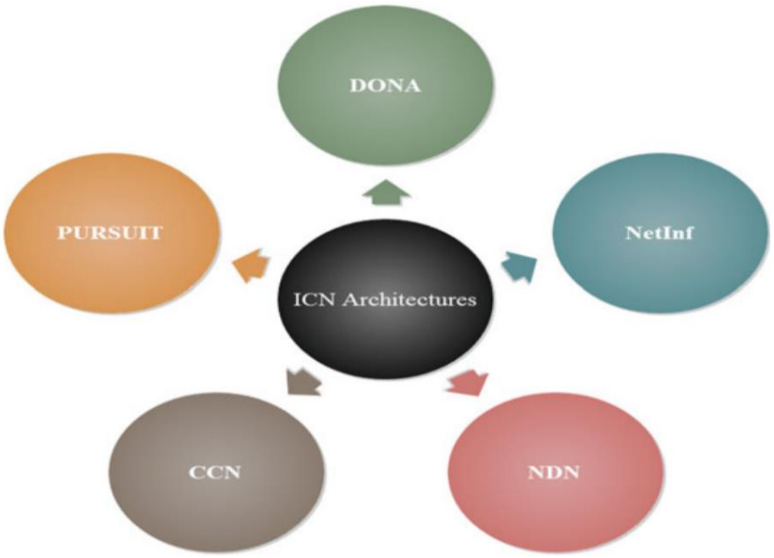


Figure 1:Les différentes architectures ICN [1]

I.3.1. Les réseaux centrés sur le contenu : CCN/NDN

Les deux architectures dont nous allons parler ici sont presque semblables, ils représentent dans cette partie les architectures basées sur le contenu.

I.3.1.1. Content-Centric-Networking

L'architecture de réseau centré sur le contenu (CCN) est un projet initialement proposé par Van Jacobson. Différent du protocole TCP/IP habituel, les réseaux CCN identifient les données par le nom. En effet, la transmission des données dans le réseau se fait en suivant un nommage hiérarchique. Les types d'objets transitant dans un réseau CCN sont des paquets (intérêt et donnée). Chaque nœud d'un réseau CCN a une cache dans lequel les contenus sont stockés. Lorsqu'un paquet d'intérêt est envoyé dans un réseau, le fournisseur ou tout autre nœud du réseau possédant une copie du contenu demandé, achemine le contenu suivant un protocole de routage basé sur le nom, vers l'interface demandeuse. Une explication plus explicite de ce scénario est expliquée dans la figure 2**.**

Fondamentalement, CCN fait partie de l'une des futures architectures Internet ayant pour but de fournir une communication indépendante de l'emplacement des données, contenus, éléments d'information, contenu en streaming (vidéo/audio)**[1].**

I.3.1.2. Named-Data-Networking

La mise en réseau de données nommées (NDN) est une version améliorée de l'architecture CCN. Semblable à CCN, NDN suit également le concept intérêt/paquet de données pour l'obtention de données particulières. Il existe cependant certaines différences d'architecture incorporées dans NDN, ce qui réduit le temps de recherche d'intérêt/de données ainsi que le problème de boucle d'intérêt. Les figures 2 et 3 illustrent les opérations de base CCN et NDN qui sont identiques. **[1]**

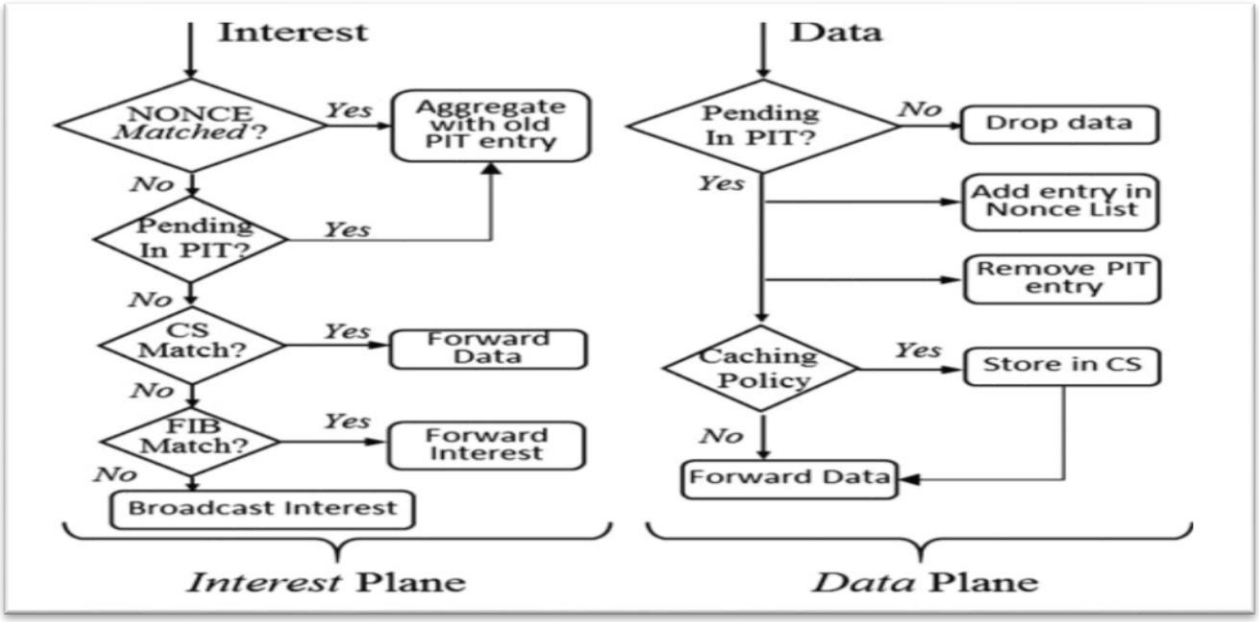


Figure 2: Déroulement CCN [1]

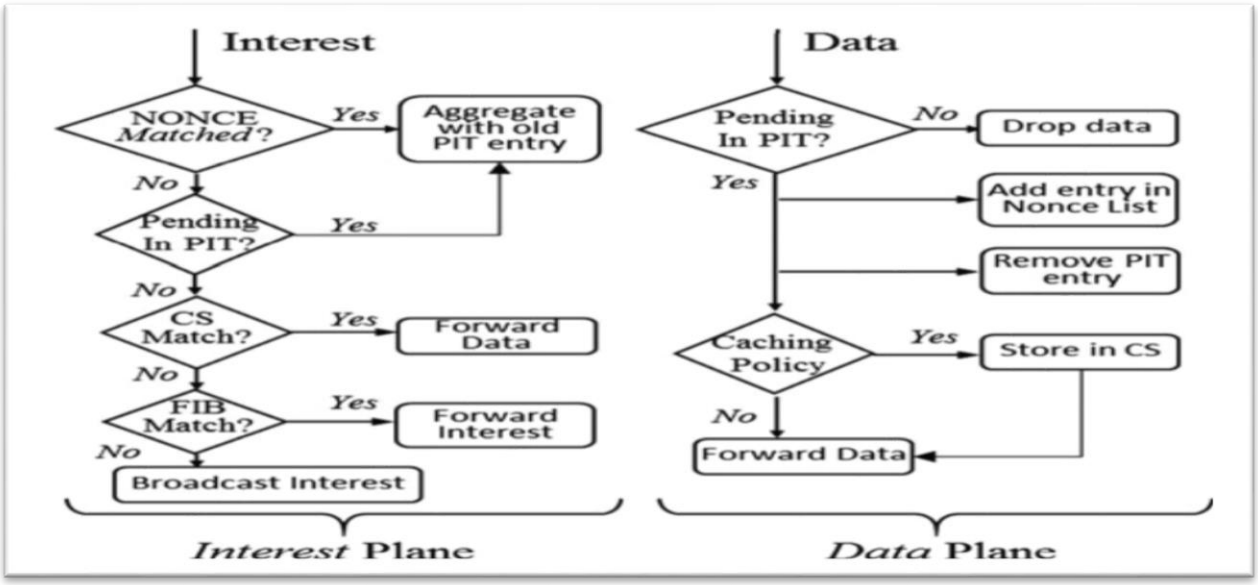


Figure 3:Déroulement NDN [1]

I.3.2. Netinf (Network of Information)

NetInf **[38]** est une architecture développée par le projet SAIL (Scalable and Adaptive Internet Solution) **[37]**. L'architecture NETINF est une combinaison de deux autres architectures ICN : CCN et PURSUIT. Etant un mélange de ces derniers, Netinf (intra-domaine) utilise un schéma de nommage plat comme PURSUIT et Netinf (inter-domaines) utilise un schéma de nommage hiérarchique comme CCN. L'objectif de Netinf est de permettre l'interconnexion de plusieurs technologies.

Pour la résolution des noms, l'architecture Netinf est basée sur des résolveurs globaux de noms qui permettent d'identifier :

- la première partie du nom (représentant l'entité ayant créé l'objet) afin de savoir où envoyer la requête pour pouvoir récupérer le contenu demandé.
- la seconde partie du nom (décrivant l'objet de manière hiérarchique) est résolue par un résolveur local, celui-ci ayant pour but de trouver quel est le contenu demandé. Ces résolveurs globaux se comportent comme des routeurs IP, envoyant les requêtes dans le réseau approprié. En plus de toutes ces fonctions, chaque nœud d'un réseau Netinf possède un cache permettant la mise en cache des contenus qui y transitent**. [2]**

I.3.3. Publish-Subscribe Internet Technologies (PURSUIT) :

PURSUIT est un projet financé par l'UE et a pour prédécesseur l'architecture PSIRP (Publication-Subscribe Internet Routing). Cette architecture possède une pile de protocoles de publication-Subscribe complète, au lieu de la pile de protocoles IP traditionnelle. PURSUIT a trois fonctions principales : rendez-vous, gestion de la topologie et transfert. Rendezvous fait référence à une collection de nœuds de rendez-vous (RN) connue sous le nom de réseau de rendez-vous (RENE). La gestion de la topologie est composée de gestionnaires de topologie (TM), tandis que la fonction de transfert est contrôlée par les nœuds de transfert (FN). **[3]**

Avec un système de nommage presque similaire à celui de DONA, PURSUIT dispose d'un schéma de nommage composé d'un identifiant de portée, qui regroupe des informations étroitement liées et un identifiant de rendez-vous qui assure l'unicité de chaque objet dans son groupe.

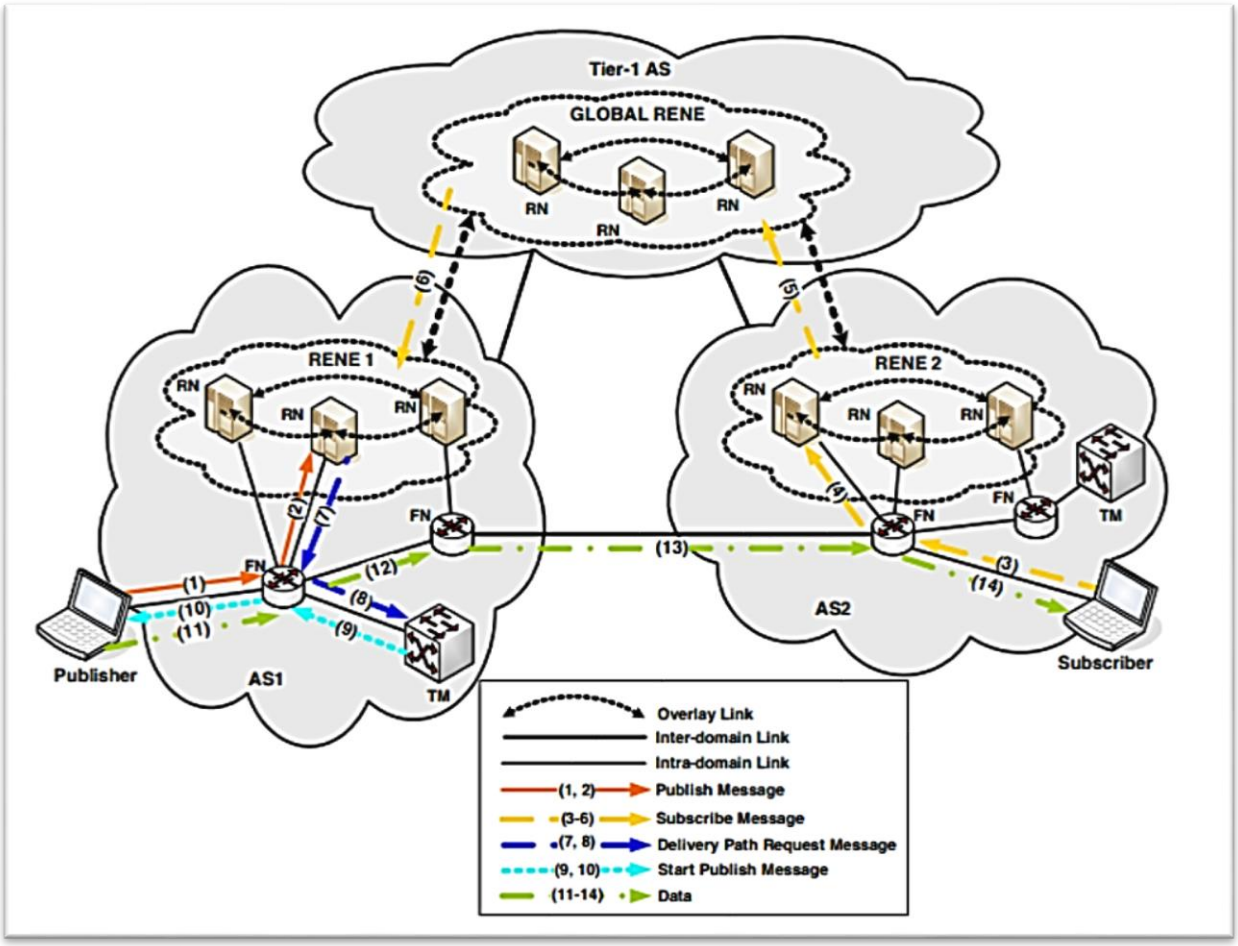


Figure 4: Aperçu de PURSUIT [3]

Comme illustré dans la figure 4 :

(1,2) Un éditeur (Publisher) envoie son contenu à son RN local par le biais d'un message de publication (Publish Message) qui y est acheminé en fonction de l'identifiant de portée par le DHT;

(3-6) Un abonné (Subscriber) envoie un message d'abonnement (Publish Message) pour demander le contenu publié par l'éditeur à son RN local qui utilisera DHT pour acheminer le message vers le RN local de l'éditeur.

(7,8) Pour l'acheminement du contenu demandé, le RN envoie une demande du chemin de livraison (Delivery Path Request Message) au TM qui va créer une route entre l'éditeur et l'abonné.

(9,10) Le TM retourne alors à l'éditeur un message de début de publication (Start Publish Message) avec la route demandée.

(11-14) L'éditeur utilise la route envoyée par le TM pour renvoyer les données via le FN de l'abonné.

I.3.4. Architecture DONA (Data-Oriented Network Architecture)

L'architecture DONA a été proposée en 2007 et est l'une des premières architectures DONA à avoir été achevée.

DONA utilise un schéma de nommage plat donc chaque nom est constitué de deux parties :

- La première partie consiste en un hachage cryptographique de la clé de l'éditeur,
- Et l'autre partie est un identifiant d'objet attribué par l'éditeur.

Le nom est unique pour chaque éditeur. Comme les RN dans PURSUIT, les gestionnaires de résolution RH sont la base de DONA, ayant pour taches de récupérer et publier des objets dans le réseau. Dans la figure 5, le fonctionnement de DONA est brièvement expliqué.

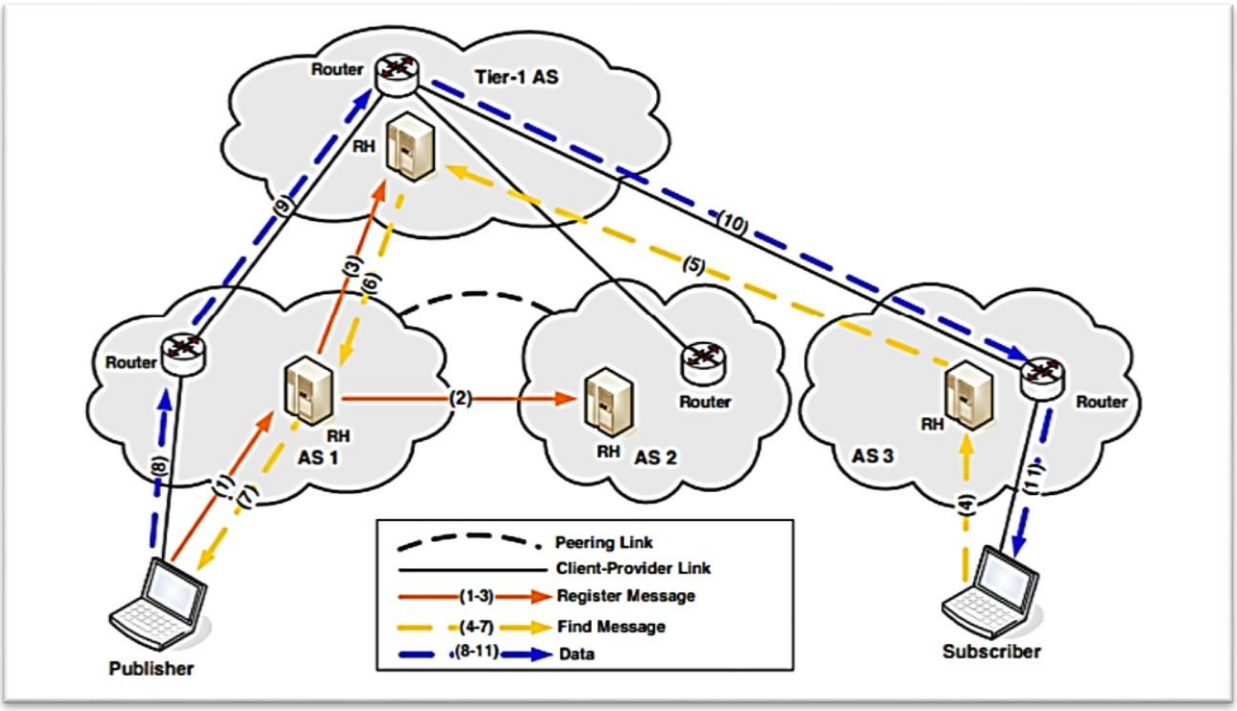


Figure 5: Aperçu de l'architecture DONA [3]

Comme illustré à la figure 5 :

(1-3) L'éditeur(Publisher) envoie un message d'inscription (Register Message) à son RH local qui va transmettre le message à ses RH proches qui stocker un schéma reliant le contenu au RH local,

(4-7) Un abonné (Subscriber) intéressé par le contenu de l'éditeur, il envoie une demande de recherche (Find Message). Ce Find message sera envoyé au RH local de l'abonné. Une fois le schéma trouvé reliant le contenu au RH local, les requêtes seront acheminées vers ce RH, celui de l'éditeur.

(8-11) Les messages FIND sont répondus. Le contenu demandé est acheminé de l'éditeur vers l'abonné, soit à l'aide d'un routage ou d'un transfert IP normal, soit les données sont envoyées à l'aide d'adresses IP locales.

I.3.5. Autres architectures ICN

Présentés ci-dessus, les architectures dont nous avons parlés sont les principaux projets ICN, les plus connus mais il y en a d'autres comme :

- **CONET**

CONET est une architecture conçue pour connecter plusieurs sous-systèmes CONET (CSS), qui peuvent être l'ensemble du réseau Internet ou quelques composants connectés au réseau. Les principaux composants de la conception de CONET, illustrés à la figure 6, sont les suivants :

End-Node (EN), Serving-Node (SN), Border-Node (BN), Internal-Node (IN) et Name-System-Node (NSN).

Un EN demande des données nommées en émettant un intérêt acheminé par les BN, qui sont situés à la frontière des CSS. Le processus de routage par nom identifie l'adresse CSS du prochain BN qui est le plus proche du SN dès que le CSS approprié est atteint. Puis, les IN transmettent le paquet en utilisant le routage sous-CONET. L'adresse CSS de EN et les adresses CSS des nœuds traversés sont ajoutés au paquet. Aussitôt qu'un nœud CONET s'avère être en mesure de fournir les données, celles-ci sont renvoyées sur le chemin inverse pour servir la demande. Tous les BN et IN le long du chemin parcouru peuvent mettre le contenu en cache **[4]**.

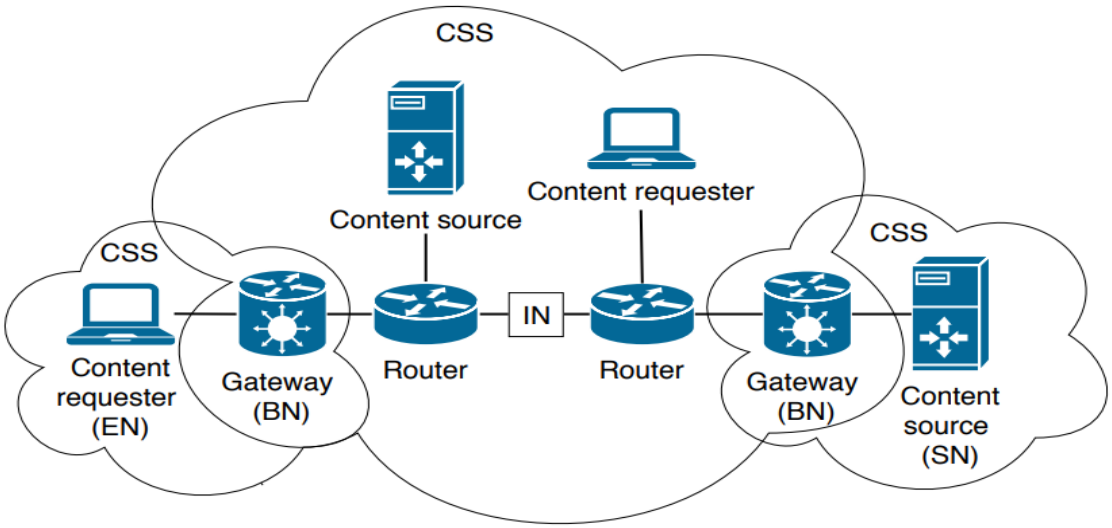


Figure 6: Aperçu de CONET [4]

- **DOCTOR**

Le déploiement et la sécurisation de nouvelles fonctionnalités dans les environnements de réseaux virtualisés (DOCTOR) est un projet en cours financé par l'Agence Nationale de la Recherche.

Le projet accompagne l'adoption de nouvelles normes en développant une utilisation sécurisée des équipements réseaux virtualisés. Cela permet de faciliter le déploiement de nouvelles architectures de réseau, permettant ainsi la coexistence d'IP et de piles émergentes, telles que NDN, ainsi que la migration progressive du trafic d'une pile à l'autre. DOCTOR propose l'utilisation de l'infrastructure NFV pour réaliser le déploiement incrémental de NDN à faible coût. Le projet propose une passerelle HTTP/NDN pour interconnecter des « îlots » ICN au monde IP, et une architecture expérimentale capable de traiter le trafic web transitant par un réseau NDN virtualisé. **[4]**

I.3.6. Comparaison des différents projets ICN

Après avoir détaillé les différentes architectures, nous allons les comparer en tenant compte de leurs spécificités :

	CCN/NDN	NETINF	PURSUIT	DONA
Nommage	Hiérarchique	Utilise le nommage plat et hiérarchique	Plat	Plat
Résolution des noms	Les CCN/NDN n'ont pas besoin d'une entité pour la résolution de nom.	NetInf dispose de résolveurs locaux.	La résolution des noms dans PURSUIT est basée sur le Rendez-Vous.	La résolution des noms dans DONA est basée sur des résolveurs hiérarchisés.
Types de Données	Paquet	Objet	Objet	Objet
Routage	Pour retrouver un contenu dans le réseau, CCN/NDN utilise le schéma de nommage hiérarchique afin de localiser les données dans le réseau.	Le routage est basée est basé sur une entité permettant de calculer la route entre un client et un serveur pour assurer la transmission de la donnée.	Etant un nommage plat, il a le même processus de routage que NetInf (intra-domaine)	Le routage se fait en utilisant le résolveur racine du réseau. Le routage IP est utilisé pour transmettre la donnée à l'interface demandeuse.
Récupération dans la cache	La récupération dans le cache est possible dans tous les nœuds du réseau dispo-	La récupération dans le cache est possible dans tous les nœuds du réseau dispo-	La récupération des données en cache n'est pas possible ici.	La récupération dans le cache est possible dans tous les nœuds du réseau dispo-

	sant du contenu demandé.	sant du contenu demandé.		sant du contenu demandé.

Tableau 1: Comparaison des différentes architectures ICN

Parmi les nombreuses architectures ICN, nous avons présenté les plus populaires d'entre elles. Nous avons également comparé ces architectures entre elles et détaillés leurs propriétés face à ces principes : le nommage des données, le type de données transportées, la résolution des noms, le routage ou l'utilisation des données stockées en cache.

L'étude de ces architectures nous a permis de faire un choix et celui-ci s'est porté sur le CCN.

En effet, cette architecture suscite beaucoup d'intérêt et est beaucoup utilisée actuellement créant ainsi une grande communauté de chercheurs autour d'elle. Avec l'intérêt qu'elle suscite, nous avons aussi pu constater du nombre de codes sources disponibles qui pourraient nous aider dans l'implémentation de notre projet. De plus, le système de nommage utilisé par CCN nous intéresse beaucoup pour l'adaptation des réseaux CCN aux services SOA. C'est pour ces raisons que nous avons choisi l'architecture CCN pour la mise en place de notre projet. Ceci fait, nous détaillerons les propriétés et fonctions des réseaux centrés sur le contenu(CCN) plus bas.

I.4. Les réseaux centrés sur le contenu (CCN)

Ayant déjà introduit les CCN un peu plus haut sans entrer dans les détails, dans cette partie, nous allons parler de comment fonctionne cette architecture, des problèmes de sécurité qu'elle peut rencontrer et de ses différents caractéristiques.

I.4.1. Les paquets des réseaux centrés sur le contenu

Il est difficile de bien présenter les réseaux centrés sur le contenu sans présenter les types de données qui y sont transportés et de leurs utilités. Donc dans cette partie, nous présenterons :

- **Le paquet d'intérêt (Interest)** :

Le paquet d'intérêt est utilisé pour l'envoi de requêtes, pour demander du contenu. Le paquet d'intérêt contient le nom du contenu souhaité (Content Name), des informations de filtrage (Selector) et enfin d'un détecteur d'intérêts en double (Nonce) comme le montre la figure 7.

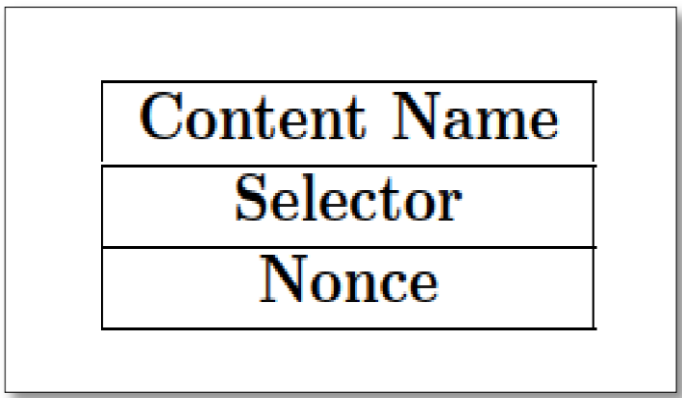


Figure 7:Structure d'un paquet d'intérêt [6]

- **Le paquet de données (Data):**

Un paquet de donnée a pour but de répondre à un intérêt. Le paquet de données contient le nom du contenu souhaité(ContentName), une signature numérique (Signature), des informations signées (Signed Info) et enfin la donnée demandée (Data) comme le montre la figure 8.

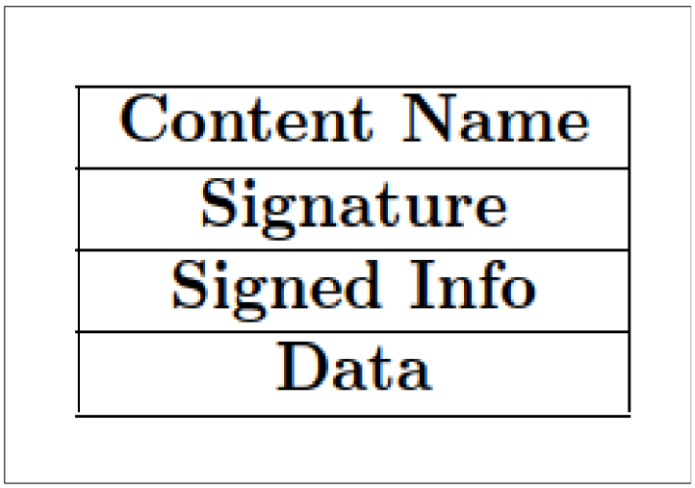


Figure 8:Contenu d'un paquet de données [6]

I.4.2. Routage dans le CCN

Pour parler du routage dans les réseaux CCN, il faut parler de ce que contient un routeur CCN. Dans ce qui suit, seront détaillées les différentes tables qui composent un routeur CCN :

- **Content Store (CS)** : le CS est un cache que possède chaque nœud du réseau CCN. Lorsqu'un nœud du réseau reçoit des données, sur le chemin du retour, celui-ci peut garder une copie de la donnée dans son CS pour pouvoir rapidement répondre aux futurs messages d'intérêts.

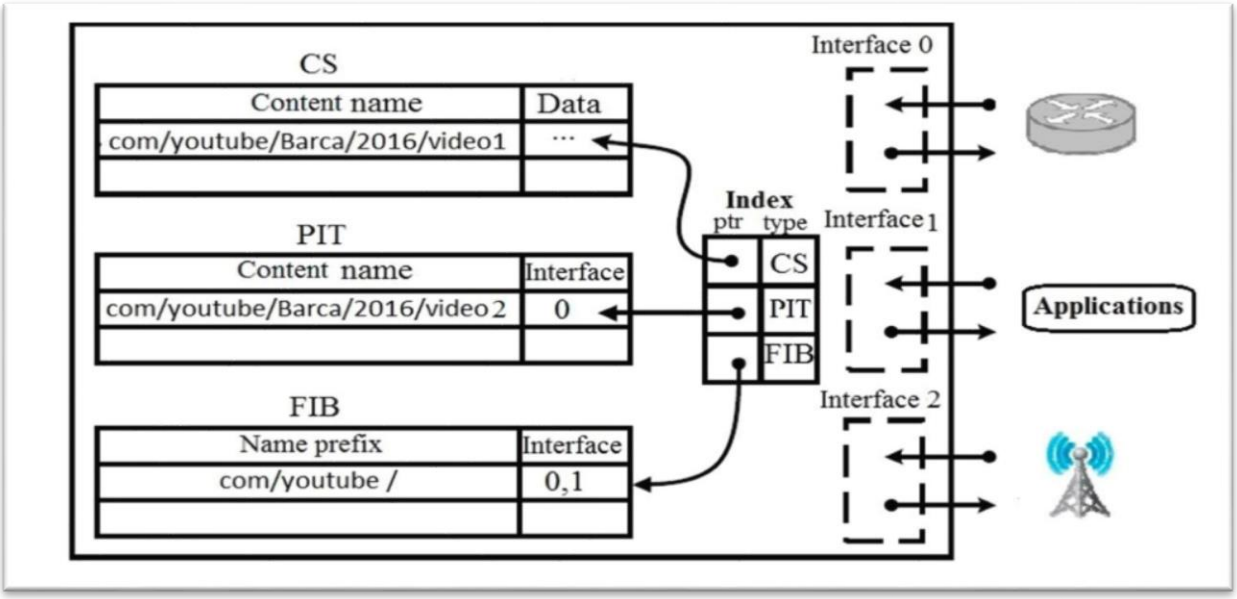


Figure 9: Structure des tables de routage d'un CCN [7]

- **Pending Intérêt Table (PIT)** :

  La table PIT d'un routeur CCN permet de conserver temporairement des messages d'intérêts reçus par le nœud afin de les transmettre au nœud suivant. Cette fonctionnalité de sauvegarde d'intérêts permet aux paquets de données (Data) de suivre le chemin inverse et livrer la donnée à l'interface demandeuse.

  Lorsqu'un nœud reçoit plusieurs paquets d'intérêts demandant le même contenu alors seul est le premier est envoyé chercher le contenu demandé et les autres mis en attente. Une fois, les données reçues, celles-ci seront distribuées aux interfaces en attente.

CCN Pending Interest Table	
Nom de contenu	**Interfaces**
ccnx:/youtube.com/news/baby_born/video1	face308, face321
ccn:/google.fr/	face103
ccn:/orange.fr/news/meteo/page.html	face201
. . .	. . .

Figure 10: Structure d'une table PIT [8]

- **Forwarding Information Base (FIB)**

La table FIB ressemble à la table de routage IP. Elle permet de gérer l'envoi des paquets du réseau vers les interfaces demandeuses.

CCN Forwarding Information Base	
Nom de domaine /prefixes	**Interface**
ccn:/youtube.com/	face101, face102
ccn:/google.fr	face103
ccn:/orange.fr/news/meteo/	face201
. . .	. . .

Figure 11: Fonctionnement d'un nœud CCN [8]

En résumé, le routage dans un réseau CCN se déroule comme suit :

- Lorsqu'un client a besoin de certains contenus ou données, il envoie un paquet intérêt à tous les nœuds du réseau auxquels il est directement lié ;
- A la réception de celui-ci, chaque nœud vérifie dans sa table CS si le contenu demandé y existe, si c'est le cas alors le contenu est retourné au client ou à l'interface demandeuse alors la demande est satisfaite. Dans le cas contraire, les nœuds vérifient dans leur table PIT. Si le contenu demandé y existe alors l'interface demandeuse sera ajoutée au PIT.
- Si les deux tables citées précédemment ne retournent aucune information alors la recherche se fera dans la table FIB alors le paquet Intérêt sera acheminé vers les interfaces conduisant au contenu demandé. Le contenu trouvé, il sera acheminé vers l'interface demandeuse. Sur le chemin de retour, une copie du contenu sera fait au niveau de tous les nœuds acheminant pour pouvoir bien satisfaire les demandes futures du même contenu.

I.4.3. Nommage dans les réseaux CCN

Le concept de nommage des données est un concept clé pour les réseaux ICN car c'est grâce à celui-ci que le routage dans le réseau est possible. Le nommage permet de retrouver les objets indépendamment de leur emplacement dans le réseau.

Le réseau CCN utilise un schéma de nommage hiérarchique dans lequel chaque segment du nom est délimité par un « / » comme dans la représentation des URI permettant ainsi la transmission des informations dans le réseau. Le nom du contenu est défini par le fournisseur de service, celui-ci devant respecter la structure hiérarchique. L'architecture adopte la structure des URL comme structure hiérarchique pour nommer un contenu. Par exemple, le nom d'un étudiant inscrit en 2016 de la spécialité SSI du département d'informatique de l'université de Blida peut-être représenté comme suit : **/CCN/Blida/Informatique/SSI/nom étudiant.**

Chaque partie du nom délimitée comporte des informations permettant de router les informations vers les interfaces appropriées.

I.4.4. Mise en cache dans les réseaux CCN

L'un des principaux avantages des architectures ICN est la mise en cache. Le cache d'un nœud CCN ressemble à la mémoire tampon dans les routeurs IP. La récupération des données dans un réseau CCN se fait dans le cache du nœud le plus proche de l'interface demandeuse contenant la donnée. L'avantage de CCN par rapport aux routeurs IP est que ces derniers ne peuvent pas réutiliser les paquets de données après transmission dans le réseau tandis que dans un réseau CCN, la récupération des données est possible, celle-ci permettant de répondre rapidement aux requêtes.

I.4.5. Transport des données dans les réseaux CCN

Dans un réseau CCN, la fonctionnalité de transport n'est pas encore disponible. Les fonctionnalités de la couche transport sont fournies par l'application dans laquelle il est utilisé, par exemple les logiciels de simulation, etc …

Le système de nommage hiérarchique de CCN permet d'ajouter des informations pour le transport des données. Ce fait lui permet de se passer du besoin d'informations sur les numéros de ports, séquences, etc. …

I.4.6. Attaques dans les réseaux centrés sur le contenu

Dans cette section, nous parlerons des trois principales types d'attaques faces aux quelles les réseaux CCN sont les plus vulnérables.

I.4.6.1. Empoisonnement de contenu

L'attaque par empoisonnement de contenu a pour but de remplir les caches des routeurs avec un contenu invalide. Le contenu inséré dans le réseau par l'attaquant peut être confondu avec un intérêt légitime car il aura le nom de celui-ci mais une signature invalide. Dans cette attaque, une personne malhonnête profite des vulnérabilités du réseau en s'accaparant des routeurs intermédiaires afin de pouvoir y injecter le contenu invalide.

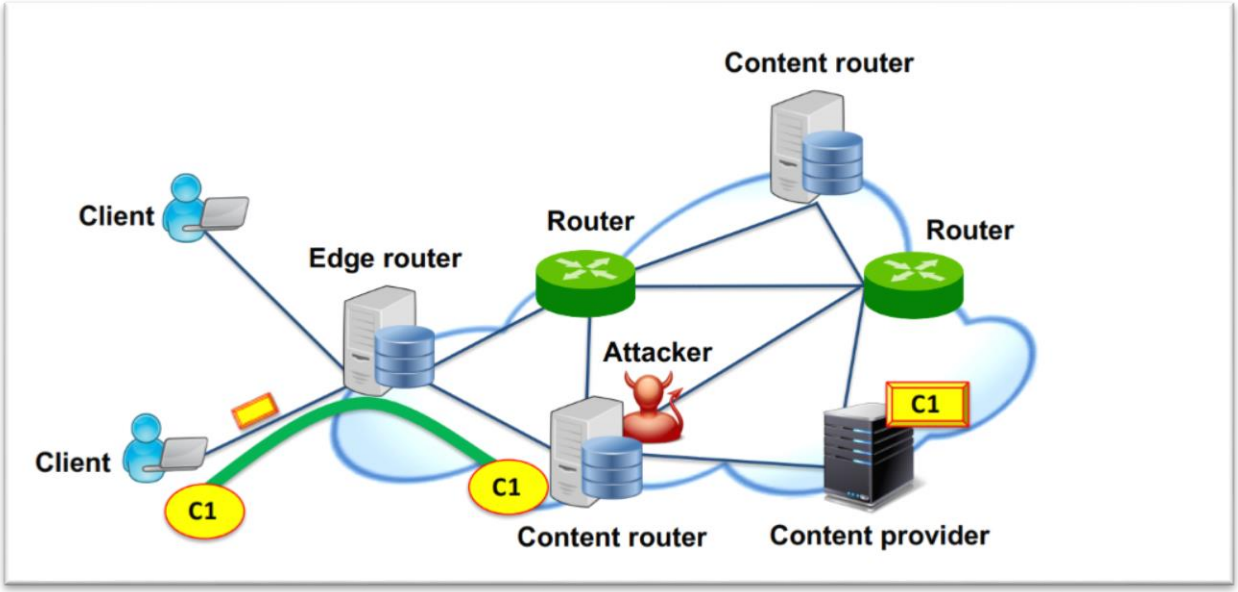


Figure 12: Scénario de l'empoisonnement de contenu [5]

Dans la figure 12, l'attaque par empoisonnement de contenu y est illustrée :

- Le client demande le contenu C1 (Rectangle à double bordure), celui-ci existant dans le serveur (Content Provider) ;
- L'attaquant (Attacker) est un l'un des routeurs existants entre le client et le serveur (Content Router). Celui-ci retourne un contenu invalide C1 (Ovale) correspondant au nom du contenu demandé au client au lieu du contenu authentique demandé C1 (Rectangle à double bordure).

Cette attaque peut avoir des effets potentiellement dévastateurs et conséquents: à moins que le contenu ne soit validé, un attaquant peut remplir le réseau d'objets de contenu empoisonnés, de telle sorte que les contenus utiles n'auront plus leur place dans le cache. **[5]**

I.4.6.2. Attaque de pollution de cache

La mise en cache dans CCN est efficace, surtout si l'univers d'Internet suit une distribution de popularité, où un petit nombre de contenus populaires sont demandés fréquemment, tandis que le reste du contenu est demandé avec parcimonie. Les contenus populaires (fréquemment demandés) peuvent être mis en cache dans le réseau, réduisant ainsi la latence des requêtes et la charge de travail du réseau. Cependant, un attaquant peut miner cette mise en cache basée sur la popularité du contenu en demandant le moins demandé des contenus populaires plus fréquemment. C'est la pollution des caches **[5].**

I.4.6.3. Attaque DDoS

L'objectif des attaques DDos est de submerger les services réseau en les inondant de demandes. Cette attaque affecte principalement la disponibilité des données. Dans les réseaux CCN, les attaques DDoS ciblent généralement la table PIT d'un nœud, qui est inondée d'intérêts pour des contenus potentiellement inexistants dans le réseau. L'attaque DDos consiste à envoyer des intérêts pour une variété d'objets de contenu qui sont peu susceptibles d'exister dans les caches des nœuds ciblés dans le but de ralentir le réseau.

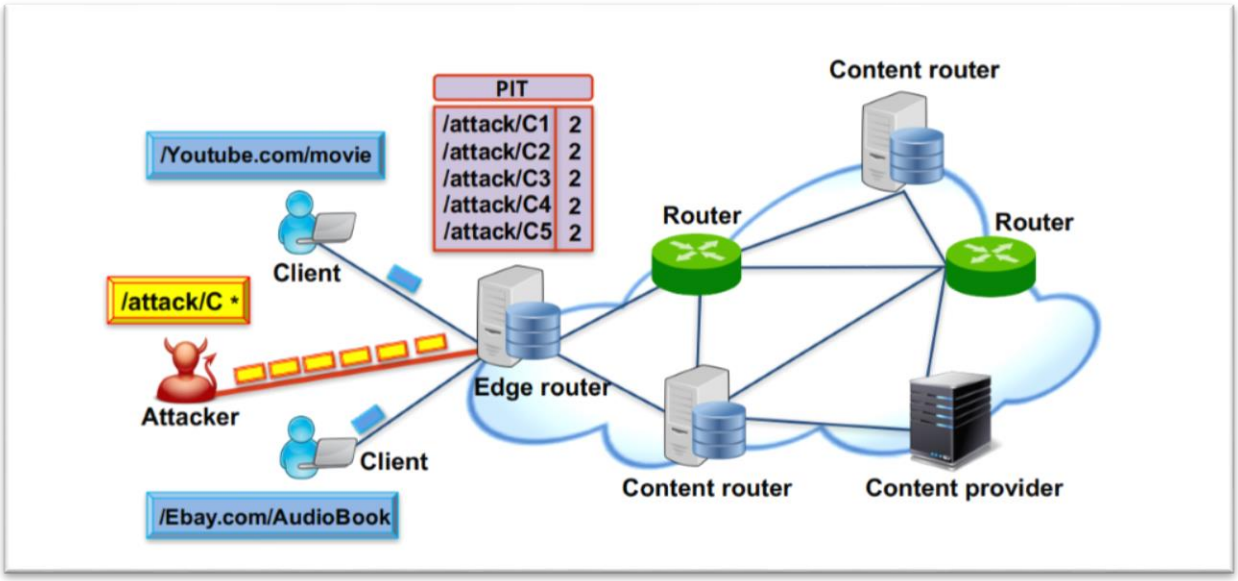


Figure 13: Scénario d'attaque DDoS [5]

Dans la figure 13, le scénario de l'attaque DDos y est illustré :

Un client et un attaquant sont connectés à un routeur de bord, qui peut mettre en cache le contenu. Le réseau est composé d'un fournisseur de contenu (Content Provider) et de deux routeurs sans contenus en cache (Content router) et des routeurs avec des contenus en cache

(Router). Dans ce scénario, le routeur de bord connecté à l'attaquant ainsi qu'au client légitime a son PIT rempli par les intérêts de l'attaquant. Le nom de l'intérêt /attack/C* fait référence à un nom de contenu non défini qui n'existe pas, est inexact ou est une demande de contenu dynamique créé à la volée. **[5]**

I.5. Conclusion

Dans ce chapitre, nous avons d'abord présenté quelques architectures des réseaux centrés sur l'information, les fonctions de base de ceux-ci et expliquer comme se passe le routage dans le CCN. Comparer les différentes architectures ICN nous a permis de faire un choix. Comme conclusion à cette recherche, nous nous sommes tournés vers les réseaux centrés sur le contenu pour la mise en place de notre projet. Nous avons détaillé cette architecture avec ses failles, ses particularités et ses atouts. Cette description des réseaux CCN aidera le lecteur à mieux comprendre le but de notre projet. Pour cela, nous passons au chapitre suivant, dans lequel nous parlerons de l'architecture logicielle SOA.

Chapitre II : L’architecture orientée service- SOA

II.1. Introduction

Service Oriented Architecture (SOA) **[11]** qui se traduit en français par Architecture Orientée Service est une architecture de conception logicielle qui a émergé dans les années 80, cette émergence est due à une nécessité d'interopérabilité et de communication entre les systèmes d'information.

Ces architectures tournent essentiellement autour de la notion de service, elles sont une abstraction de différents composants et technologies informatiques ainsi différentes applications peuvent s'invoquer et interagir mutuellement, indépendamment de leurs plateformes sous-jacentes, de leurs localisations géographiques et aussi des langages dans lesquels elles ont étés développées **[10].**

Les SOA offrent des services réutilisables avec la possibilité de découvrir et de composer dynamiquement avec un couplage faible ces services pour avoir d'autres services appelés services composés.

Cette flexibilité et cette homogénéité offertes par l'architecture orientée service, ont poussé les développeurs et les spécialistes des IT (Information Technology ou Technologie de l'Information) à se tourner de plus en plus vers cette architecture dont le champ d'application est très divers actuellement notamment dans les secteurs économiques tels que la banque, la santé, les transports **[11].**

Ce chapitre parlera dans un premier temps de la notion de service ainsi que de ses composants, ensuite nous donnerons différentes définitions des SOA, les avantages de ces architectures et également quelques inconvénients. Pour finir, dans la dernière partie nous parlerons des services web et RESTFul qui sont les implémentations les plus connues et les plus utilisées de ces architectures aujourd'hui.

II.2. Notion de service

Dans cette section on définit un service et décrit brièvement ses composants.

II.2.1. Définition d'un service

En se basant sur **[10], [11]** et **[12]** on peut dire que, dans une SOA, les services sont les éléments centraux, un service représente une entité de traitement dont les fonctionnalités sont bien définies dans un annuaire de service.

Un service peut être utilisé par un autre service, de ce fait nous pouvons composer les services pour avoir des fonctionnalités de plus haut niveau. Cette composition se fait en ayant un couplage faible entre les services.

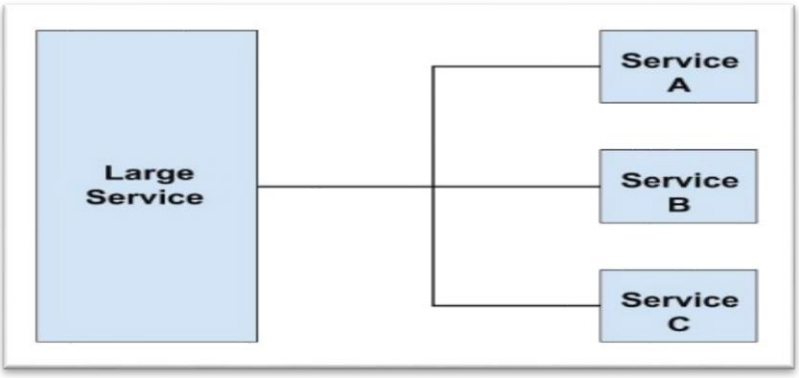


Figure 14: Composition de service [16]

II.2.2. Les composants d'un service

Un service est composé de :

- L'interface : qui est l'encapsulation de l'implémentation et l'exécution physique du service définissant aussi la manière d'accéder au service (nom, données d'entrée et de sorties) ;
- Le contrat : détaille les conditions d'utilisation du service ;
- La logique d'affaire : comprend les règles de service, les exécutions et les résultats ;
- Les données : les éventuelles données du service ;
- L'implémentation : l'exécution physique du service.

II.3. Architecture orientée service

L'architecture orientée service tourne autour de beaucoup de concepts et d'éléments dont la compréhension est essentielle pour une bonne implémentation de cette architecture ainsi dans cette partie nous parlerons de ces éléments.

II.3.1. Définition de l'architecture SOA

Plusieurs définitions selon différents points de vue ont été données aux SOA par différents chercheurs et spécialistes des technologies de l'information (IT) mais qui vont plus ou moins dans le même sens.

Le groupe OASIS définit l'architecture orientée service en trois points comme suit : le SOA lie les besoins des consommateurs aux capacités des fournisseurs ainsi les SOA fournissent un

mécanisme faisant correspondre les besoins des consommateurs avec la capacité des fournisseurs de services combinant la capacité d'effectuer un travail pour un autre, la spécification du travail offert par un autre et l'offre exécutant un travail pour un autre **[15]**.

Teo **[13]** avec ses collaborateurs définissent les SOA comme une architecture d'application au sein de laquelle toutes les fonctions sont définies comme des services indépendants avec des interfaces invocables bien définies qui peuvent être appelées dans des séquences définies pour former des affaires processus.**[13][14]**

Pour Mark et Bell **[12]**, SOA est une architecture métier conceptuel où les fonctionnalités métiers, la logique de l'application, est mise à disposition des utilisateurs SOA, en tant que services réutilisables et partagés dans un environnement informatique. **[12]**

D'une manière générale, nous pouvons affirmer que l'architecture orientée service est une méthode de conception, de développement de briques logicielles appelées service offerts par des fournisseurs pour des utilisateurs ou consommateurs qui peuvent même être des services permettant une interopérabilité entre différents système d'informations.

II.3.2. Les composants de la SOA

Une architecture SOA comprend trois composants qui sont :

Composants	Description
Le fournisseur de service	Correspond au propriétaire du service, il publie et décrit le service.
Le client	Correspond au demandeur du service pouvant lui-même être un service
L'annuaire de services	Correspond à un registre de description de service facilitant la publication de service pour les fournisseurs et les recherches de services pour les clients

Tableau 2: Les composants d'une SOA

Les interactions de base entre ces trois entités incluent les opérations de publication, de recherche et de lien d'opérations comme à la figure suivante :

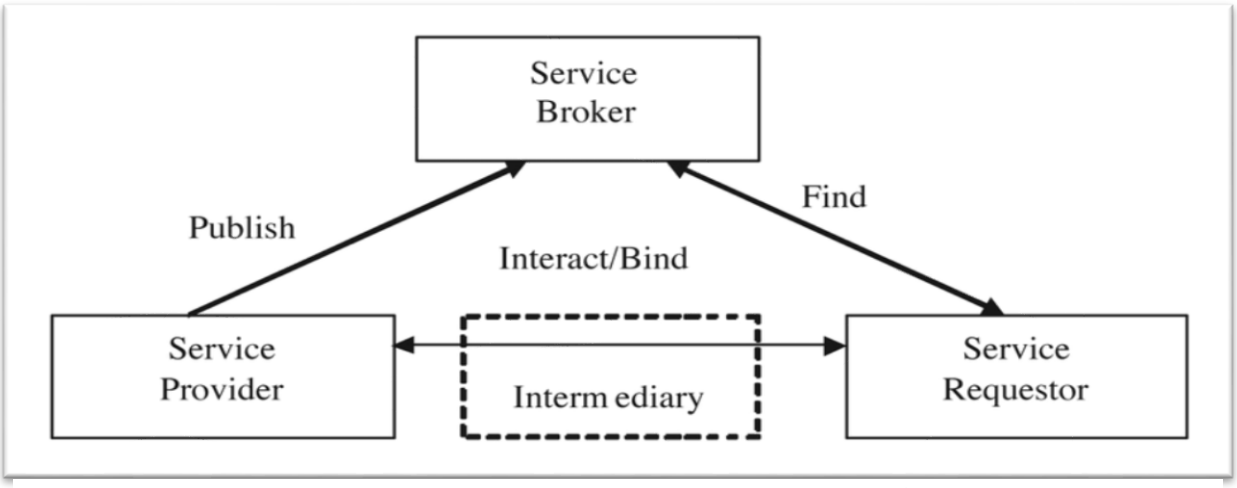


Figure 15: Interaction entre composants d'une SOA [17]

II.3.3. Le principe de fonctionnement de SOA

L'architecture SOA fonctionne comme suit :

Le fournisseur de service définit la description de son service dans un document de type standard ensuite il la publie dans un annuaire de service. Le client se servira de l'annuaire pour rechercher un service. Il examine ensuite la description du service sélectionné pour récupérer les informations nécessaires lui permettant de se connecter au fournisseur de service. Une fois cela fait, il fait invoque l'opération désirée par le biais d'une requête contenant les paramètres d'entrée de l'opération.

Le fournisseur reçoit et traite la requête et renvoie la réponse.

II.3.4. Les propriétés d'un service

L'architecture orientée service a été construit autour de principes bien précis et qui font également son succès, nous pouvons citer entre autres **[10]** :

- L'interopérabilité : elle permet à des applications de communiquer indépendamment de leurs plateformes sous-jacentes ;
- Le couplage faible : le niveau d'abstraction des services étant élevé, les interactions entre eux doivent être réduits au maximum pour que la modification d'un entre eux n'ait pas de conséquence sur les autres ;

- La réutilisation : les services doivent être conçus de manière à ce qu'ils s'adaptent rapidement à des fins et utilisateurs différents ;
- La granularité des services : les services communiquent entre eux et le nombre d'opérations diffère par service, cette propriété permet de trouver un compromis entre les services qui englobe beaucoup d'opérations et ceux qui en ont peu.
- La découverte : les services doivent être implémentés dans des standards afin d'être découvrables, la découverte se focalise sur la définition de contrat de service, cette dernière doit être claire pour éviter des redondances et favoriser la réutilisation.
- La composition : les services doivent être capables de s'invoquer afin de donner d'autres services ainsi un service peut être un service composant, un service composé ou les deux à la fois.

II.3.5. Avantages et inconvénients

Nous parlons très souvent des avantages des architectures SOA mais elles ont des inconvénients qui peuvent être entravant, nous en parlons dans cette section.

II.3.5.1. Avantages

Les SOA ont plusieurs avantages dont :

- Une modularité permettant de remplacer facilement un service par un autre ;
- Une réutilisation des composants ;
- Une tolérance aux pannes ;
- Une réduction de cout pour les particuliers (consommateurs) ;
- Une évolution rapide grâce à la composition.

II.3.5.2. Inconvénients

Les SOA ont énormément d'avantages mais ils ont aussi des inconvénients qui peuvent être :

- Une non-adaptabilité aux systèmes nécessitant le transport de données élevées ;
- Un cout de développement élevé pour les fournisseurs de service ;
- Difficulté de migration d'une architecture monolithique à une architecture SOA.

Le fait que les SOA ne soient pas adaptées aux systèmes nécessitant un transport de quantité importante de données est plus dû aux protocoles de transport réseaux sur lesquels ils fonctionnent plutôt que les SOA eux-mêmes comme G. Dugué le prouve d'ailleurs dans sa thèse de doctorat **[24].**

Donc cet inconvénient découle plus du réseau TCP/IP que des SOA, ceci et d'autres nécessités ont d'ailleurs conduit à la naissance du style architectural basé sur les micro-services qui permettant de développer, de déployer et de gérer opérationnellement des applications distribuées, constituées de services aux fonctionnalités complémentaires, potentiellement hétérogènes et interopérables.

Donc nous pouvons très naturellement se dire que les SOA pallient à cet inconvénient s'ils fonctionnaient sur une architecture réseau mieux adaptée au transport de quantité de données importante, c'est qu'interviennent les réseaux CCN.

II.4. Les services web

Pour implémenter une architecture SOA, les services web représentent l'implémentation la plus utilisée, de ce fait nous parlerons de ces derniers dans cette partie.

II.4.1. Définitions

Un service web désigne une application mise à disposition sur internet par un fournisseur de service et accessible via son interface par des clients à travers des protocoles Internet standard. **[10]**

Les services web représentent l'implémentation la plus utilisée de l'architecture SOA, ils reposent sur un modèle en couches avec trois opérations fondamentales qui sont l'invocation, la description et la découverte.

L'implémentation de ces couches est assurée par une collection de normes et de protocoles appelée *WEB SERVICE PROTOCOLE STACK* qui comprend entre autre :

- SOAP qui assure la communication via des messages entre le client et le fournisseur ;
- WSDL, un document contenant la description du service ainsi que la manière et les informations à fournir pour y accéder ;
- UDDI, représente l'annuaire des services web, il fournit l'infrastructure pour la publication et la découverte des services web.

Ces trois protocoles se basent sur le standard XML.

II.4.2. Les architectures des services web

L'architecture des services web définit des éléments généraux assurant l'interopérabilité. Nous parlerons de deux architectures :

- L'architecture de référence : pour les services simples ou isolés
- L'architecture en pile : pour la composition de services.

II.4.2.1. Architecture de référence

Cette architecture a trois objectifs principaux :

- Identification des composants fonctionnels ;
- Définitions des relations entre composants ;
- Etablissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés de l'architecture.

Pour atteindre ces objectifs, cette architecture comporte trois composants qui sont :

- Le fournisseur de service : représente le propriétaire du service, il assure l'hébergement et l'exécution du service. Il est lui-même constitué de différentes couches dont la couche de données, pouvant être une ou plusieurs bases de données, la couche applicative qui est la plateforme d'accueil du service et la couche de description contenant la description des fonctionnalités du service dans un document standard.
- Le client : représente le demandeur ou consommateur du service, il est constitué techniquement de l'application de recherche du service et celle invoquant le service, il peut être un autre service.
- L'annuaire de service : est un registre de description de service facilitant la publication et l'indexation aux fournisseurs d'un côté et la recherche aux clients de l'autre.

Les interactions de base entre ces trois composants peuvent être formalisées par les 6 étapes de la (figure 16) :

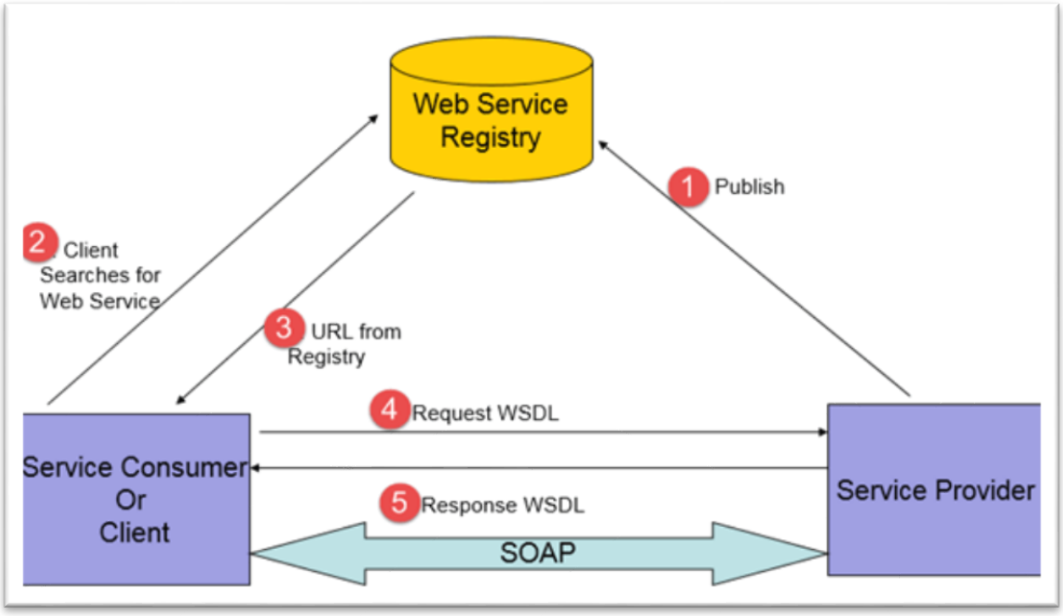


Figure 16: Fonctionnement de l'architecture de base d'une SOA [18]

- Etape 1 : Description du service

  Le fournisseur définit et décrit son service dans un document WSDL.

- Etape 2 : Publication du service dans un annuaire.

  Le fournisseur publie son document WSDL dans un annuaire de service UDDI pour le rendre accessible sur le réseau et faciliter sa découverte.

- Etape 3 : Recherche du service web

  Le client se connecte à l'annuaire UDDI et effectue une recherche via un message SOAP.

- Etape 4 : Récupération des informations et description du service

  Une fois le service trouvé, l'annuaire fournit au client le document WSDL et ce dernier s'identifie auprès du fournisseur montrant ainsi sa volonté d'utiliser le service web.

- Etape 5 : Connexion et invocation du service

  Le client extrait les informations de connexion au service du document WSDL et utilise ces dernières pour invoquer les opérations désirées par le biais d'une requête SOAP porté par le protocole HTTP.

- Etape 6 : Réponse du serveur

Le serveur qui représente le fournisseur envoie sa réponse au client sous formant XML encapsulé dans une enveloppe SOAP.

II.4.2.2. Architecture en pile

Dans cette architecture, les différents éléments sont disposés en couches superposées reposant les uns sur les autres en partant du haut (figure 17) :

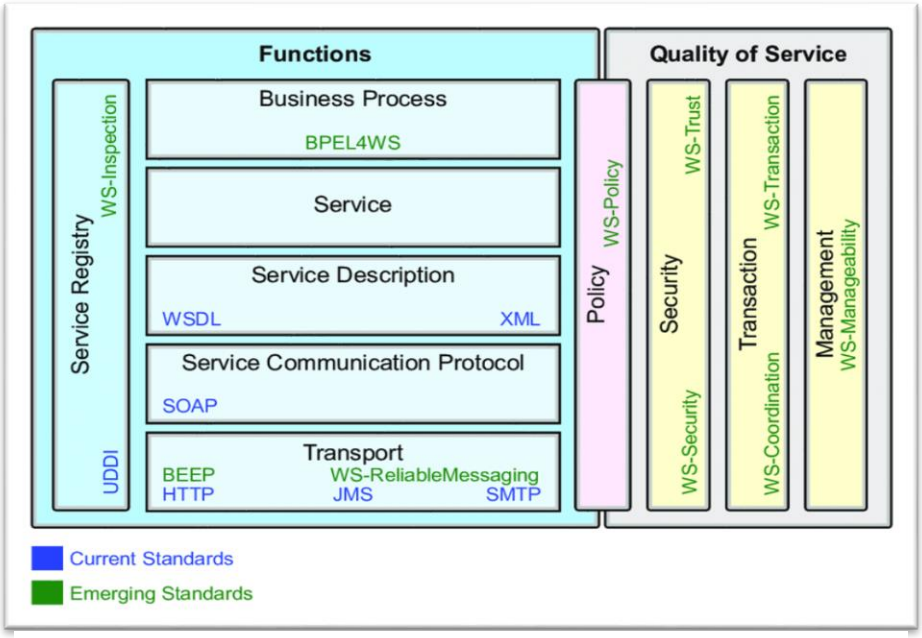


Figure 17: Eléments d'une architecture en pile [19]

- La couche transport : c'est la couche de base, elle assure le transport des messages avec des protocoles comme HTTP, RPC, SMTP, etc.
- La couche message : elle repose essentiellement sur le standard SOAP dont la structure est conçue indépendamment de tout modèle de programmation ;
- La couche description : cette couche comprend la description avec le langage WSDL et la découverte avec le protocole UDDI.
- La couche processus : cette couche permet l'intégration et la composition de services existants pour former de nouveaux services.
- Les couches transversales : ces couches rendent viable l'utilisation effective des services Web dans le monde industriel en augmentant, par exemple, la sécurité.

II.5. Langages et protocoles utilisés par les services web

Pour l'implémentation d'un service web, il y'a un certain de nombre de technologies qui est utilisé que nous décrirons brièvement dans cette section.

II.5.1. Le langage XML

XML (Extended Markup Langage) est un langage de balisage promulgué par le World Wide Web Consortium (W3C) **[20]**. Il s'oriente texte et décrit la manière de structuration des données.

XML a été développé pour le stockage et le transport de données, cela en séparant les données de la manière dont elles sont représentées.

L'hétérogénéité des systèmes d'information a permis au langage XML d'être omniprésent.

Ces dernières années, beaucoup de technologies et de langages se sont développés autour du XML dont :

- XLink et XPointer : ils permettent d'établir des liens entre documents ;
- XQuery : permet l'extraction d'information à partir de documents XML et synthétise celles-ci pour former de nouvelles informations ;
- XPath : permet de sélectionner des éléments en créant des chemins de localisations.

Les standards des services web utilisent énormément le langage XML.

II.5.2. Le protocole SOAP

SOAP (Simple Objet Architecture Protocole) ou (Simple Objet Access Protocole) ou encore (Service Oriented Architecture Protocole) est un protocole qui permet d'échanger des informations structurées dans un environnement reparti et décentralisé. Il est basé sur le langage XML.

Il se base sur des protocoles standards de communication (HTTP, FTP, SMTP) pour transmettre des documents XML et cela avec une abstraction des technologies d'implémentation et du système d'exploitation.

SOAP définit un ensemble de règles pour structurer des messages qui peuvent être utilisées dans de simples transmissions unidirectionnelles mais il est particulièrement utile pour exécuter des dialogues requête-réponse ou RTP (Remote Procedure Call). **[10]**

Un message SOAP a une structure normalisée comprenant :

- Une enveloppe (SOAP-ENV) : racine du message, il contient un en-tête (SOAP header) optionnel et un corps obligatoire (SOAP body) ;
- L'en-tête SOAP (SOAP header) : est optionnel et est utilisé généralement pour transmettre des données d'authentification et la gestion des transactions.
- Le corps SOAP (SOAP body) : contrairement à l'en-tête, il est obligatoire et représente le corps du message.

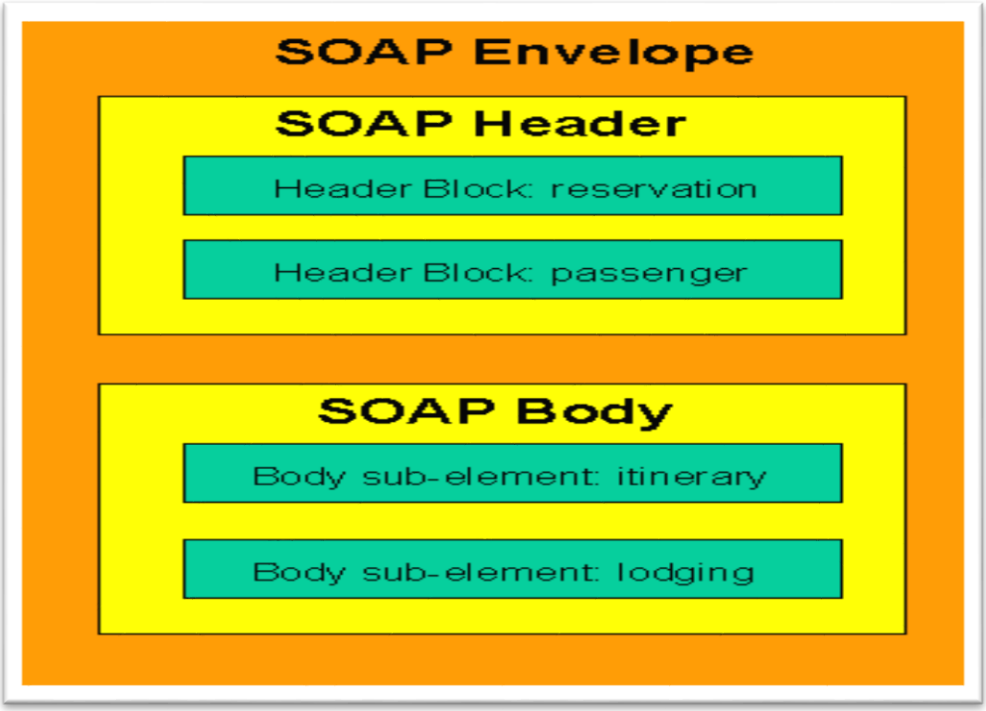


Figure 18: Enveloppe SOAP [20]

II.5.3. WSDL-La description

WSDL (Web Service Description Langage) est un standard W3C basé sur XML pour la description des services web.

Il définit les paramètres nécessaires pour invoquer un service, le format et le type des données retournées. **[12]**

Un document WSDL décrit un service, la manière d'y accéder en incluant des détails sur le protocole à utiliser, l'URL, les opérations pouvant être effectuées et les formats des messages SOAP. **[10]**

Les éléments constitutifs d'un document WSDL sont :

- La racine marquée par une entité ‘<definitions>’, elle contient à son tour cinq éléments qui sont :
  - <type> : contient les types des données échangées sous format XML-SCHEMA ;
  - <message> : définit les données échangées entre le service, le nom et le type de ces données ;
  - <porttype> : définit de manière abstraite les opérations du service web, chaque opération est déclenchée par une requête ;
  - <binding> : permet de faire le lien entre les <message> et les <porttype> ainsi que la définition du protocole utilisé pour la communication.

- L’implémentation de service : est la partie qui décrit comment une interface particulière d’un service est implémentée par un fournisseur. **[10]**
  Elle comprend :
  - Services : décrivent les adresses permettant d’invoquer les services ;
  - Ports : définissent un point de communication unique avec l’adresse réseau à laquelle elle est liée.

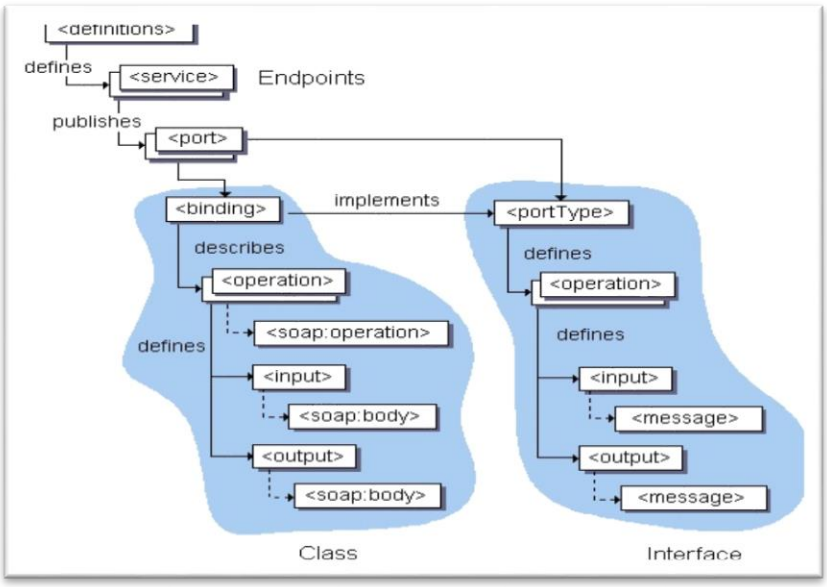


Figure 19: Exemple d’une description WSDL [21]

II.5.4. UDDI-Enregistrement/Découverte

UDDI (Universal Description and Discovery Integration) **[10]** est une norme pour la publication et la découverte des services web qui a vu le jour suite à un accord industriel entre les géants des technologies de l'information dont Microsofts, Oracle, HP, IBM, Sun Microsystem, etc…

C'est un registre fournissant une API aux clients pour la découverte des services web et une autre aux fournisseurs pour la publication de leurs documents WSDL.

UDDI utilise les protocoles SOAP et HTPP pour les spécifications. Chaque registre UDDI comprend trois structures :

- Les pages blanches : ces pages contiennent les données des fournisseurs, nom, adresse, contact, identifiant. Ces données sont décrites dans des entités de type '<BussinessEntities>'.
- Les pages jaunes : celles-ci contiennent des données sur l'activité ou le service métier du fournisseur et sont décrites dans des entités de type '<BusinessService>'.
- Les pages vertes : ces dernières contiennent des informations techniques de chaque service en plus des spécifications sur l'utilisation du service. Ces informations sont encapsulées dans des documents '<BindingTemplates>' et '<TModel>'.

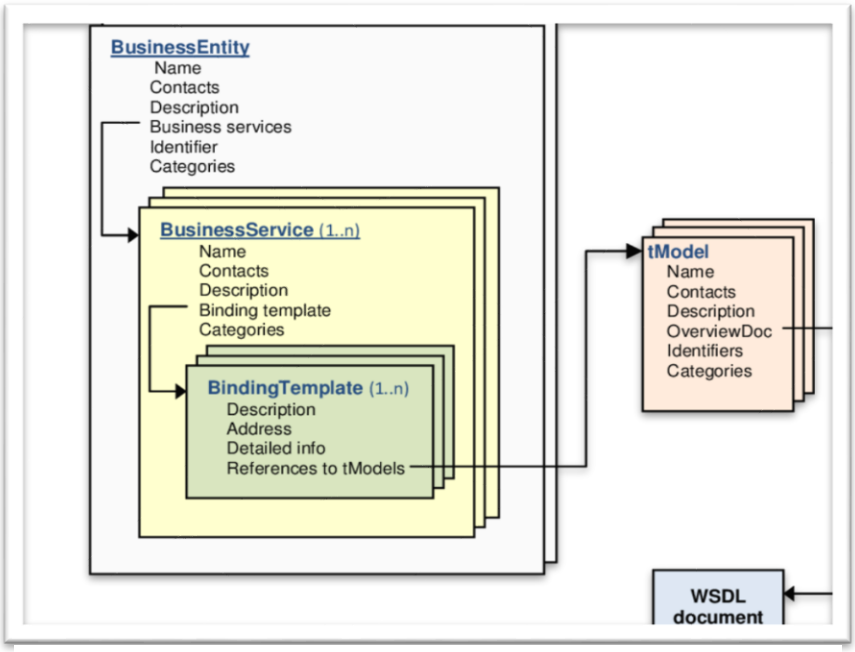


Figure 20: Exemple d'un annuaire UDDI [22]

II.5.5. Standard HTTP et SMTP

Tous les échanges dans les services web sont effectués en utilisant des enveloppes SOAP qui sont encapsulées dans des structures HTTP ou SMTP qui est des protocoles de la couche application du modèle TCP/IP, comme illustré a la figure 21.

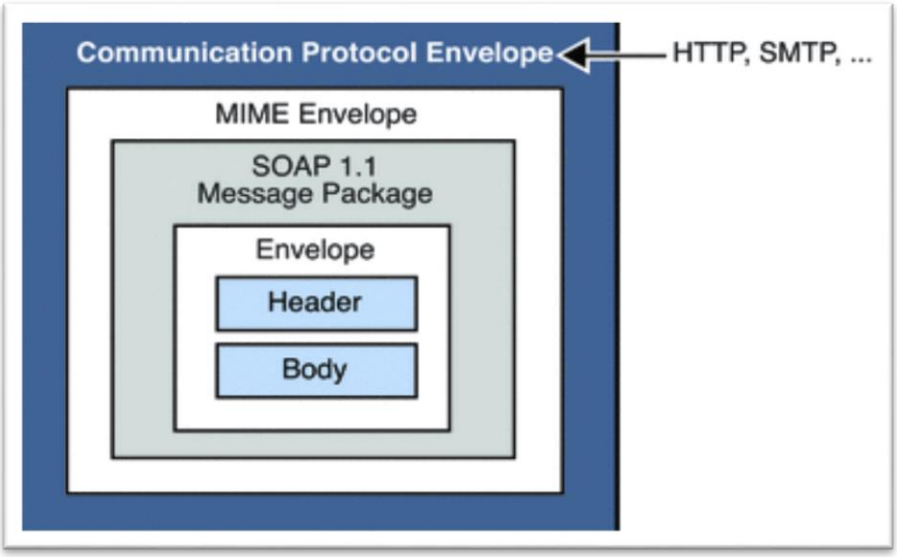


Figure 21: Message SOAP dans une enveloppe http [23]

II.5.6. Autres protocoles pour les services web

D'autres protocoles sont utilisés par les services web en plus des standards de bases décrits ci-dessus. Quelques-uns d'entre eux sont cités avec une brève description dans le tableau ci-dessous **[10]**:

Fonctions	**Standards**
Découverte	UDDI, WSIL, ADS (Advertisement and Discovery of Services Protocol), ebXML.
Description	WSDL, Daml-S, OWL-S, SAWSDL,
Echange	SOAP, DIME (Direct Internet Message Encapsulation), SWAT (SOAP With Attachment).
Transaction	SOAP-RP (SOAP-Routing Protocol), WS-Transaction (XML Web Services Transaction Language) BTP (Business Transaction Protocol)
Sécurité	SAML (Security Assertion Markup Language), XACML (XML Access Control Markup Language), WS- Security (Web Services Security)
Gestion des processus	ebXML, XLang, XPDL (XML Pipeline Definition Language), WSFL (Web Services Flow Language), WSCL (Web Services Conversation Language) , tpaML (Trading Partner Agreement Markup Language), BPWS (Business Process Execution Language)
Orchestration des processus	BPML (Business Process Management Language), BPEL4WS (Business Process Execution Language for Web Services)WSBPEL,

Tableau 3: Autres protocoles utilisés pour l'implémentation d'une SOA

II.5.7. Les services RESTFul

REST, acronyme de 'Representational State Transfer', appelé souvent RESTFul, est un style architectural contrairement aux services web basés sur SOAP qui est un standard.

Les services REST ont été mis en place afin de résoudre les problèmes liés à l'utilisation de SOAP avec lequel tous les échanges se font à l'aide de documents XML dans une enveloppe HTTP ce qui peut extrêmement encombrant avec certains langages de programmation.

Les échanges entre composants de REST peuvent se faire dans des formats de fichiers différents du XML comme JSON, CSV, RSS ce qui les rend beaucoup plus souples que SOAP en plus du fait que les requêtes ne soient pas encapsulées. **[10]**

Analogiquement au WSDL les services RESTFul sont décrits par le langage WADL (Web Application Description Langage), cette description fournit les informations sur les ressources déployées, les types de supports, les méthodes HTTP.

L'accès à un service RESTFul se fait avec un simple URL, pour effectuer une action nous utilisons les méthodes du protocole HTPP **[10]** :

- POST pour créer une ressource sur le serveur ;
- GET pour accéder à une ressource ;
- PUT pour modifier l'état d'une ressource ;
- DELETE pour supprimer une ressource.

Les principales différences entre SOAP et RESTFul sont résumées dans le tableau suivant :

SOAP	**REST**
SOAP est un protocole	REST est un style architectural
Les transferts se font avec HTTP, SMTP, FTP, etc…	Les transferts se font uniquement avec http
XML est le seul format de données autorisé	D'autres formats de données comme CSV, JSON sont autorisés
SOAP définit les normes à suivre strictement	REST ne définit pas trop de normes comme SOAP
SOAP ne peut pas utiliser REST	REST peut utiliser SOAP
Les interfaces de services sont utilisées pour exposer la logique métier	La logique métier est exposée par des URL

Les performances sont moins élevées que pour REST	Les performances sont plus élevées que pour SOAP

Tableau 4: Comparaison SOAP-RESTFul

Contrairement au service RESTFul, le SOAP prend en charge tous les protocoles de transfert de la couche application particulièrement le protocole UDP qui nous intéresse ici parce que les réseaux CCN le prennent en compte.

II.6. Conclusion

Bien vrai qu'ils existent depuis longtemps, les architectures orientées services ont une grande importance et nécessité dans le monde d'aujourd'hui et cela ne fera que s'accentuer avec le temps parce que, de jour en jour, les utilisateurs d'internet augmentent, avec eux, les informations à transporter et de nouveaux services doivent émerger pour répondre à ces nouveaux besoins.

Aucune entreprise n'est en mesure, aujourd'hui, de développer toutes les applications dont elle a besoin donc il faut forcément une interopérabilité entre leurs systèmes d'informations, les SOA répondent à cela. Pour les particuliers, les SOA sont aussi nécessaires à cause de leur cout d'utilisation peu élevé.

Avec un nombre constant de service de demande de contenu comme les streaming vidéo et audio, penser à faire fonctionner les SOA sur un type de réseau mieux adapté à cela, comme le CCN est un point non négligeable.

Dans les lignes qui suivent, nous ferons notre proposition pour faire fonctionner ensemble une architecture SOA et un réseau CCN.

Chapitre III : Architecture CCN-SOA

III.1. Introduction

L'internet actuel et les systèmes d'information en général se tournent vers des réseaux adaptés au transport de quantité importante de données donc l'architecture réseau CCN est peut-être l'avenir d'internet et l'architecture SOA est et continuera probablement d'être l'architecture logicielle la plus utilisée de surcroit avec l'expansion incroyable d'internet. Cette conjecture est d'autant plus crédible à cause de l'interopérabilité offerte par le SOA donc la combinaison des deux pour tirer le meilleur de l'un comme de l'autre deviendra éventuellement une nécessité.

Bien qu'extrêmement plébiscités, les SOA soulèvent néanmoins des problématiques, l'une d'elle qui n'est pas des moindres étant leur inadaptation au transport de quantité importante de données pour les services de streaming par exemple mais cette problématique est intrinsèque au support réseau utilisé qui est le TCP/IP ainsi nous voudrons surmonter cela en proposant une architecture reliant le SOA et les CCN.

Après avoir présenté les deux architectures dans les chapitres précédents, dans celui-ci, nous présenterons quelques approches existantes sur la mise en place d'architectures reliant les ICN et les SOA souvent appelées SERVICE CENTRIC NETWORK ensuite, nous parlerons de l'architecture que nous proposons, pour finir nous simulerons notre architecture.

III.2. Quelques approches existantes sur la mise en place d'architecture reliant CCN (ICN) et SOA

De nombreuses littératures existent sur la mise en commun des réseaux ICN avec les architectures basées sur les services, prouvant tout l'intérêt que suscite cette méthode. Ici nous parlerons de trois de ces approches.

III.2.1. Service-Centric Networking (SCN)[25]

Cette approche proposée par T. Braun **[25]**, étend les mécanismes architecturaux des réseaux CCN pour qu'elles prennent en compte les services.

Les services sont des éléments logiciels situés dans l'ensemble de l'infrastructure réseau et sont hébergés sur du matériel dédié, placé à côté de l'infrastructure de routage avec une intégration de la recherche au transfert des paquets. **[25]**

Cette approche se base sur le paradigme de programmation orienté objet utilisant des objets pour l'interaction des différentes entités sur l'architecture, même les paquets sont des objets.

Il y'a trois types d'objets :

- Les objets de contenus purs représentent les données réelles comme des images, des fichiers audio/vidéos et ne prenant en compte que les méthodes de lectures. Ces dernières sont les méthodes appelées par défaut si aucune autre n'est appelée spécifiquement. **[25]**
- Les objets de services purs ne contiennent aucune donnée mais des fonctions pouvant être invoquées par les clients pour traiter leurs données.
- Les objets de contenu et de service combinés intègrent à la fois les objets de services et de contenus. Une demande de service est envoyée en utilisant le nom de l'objet comme adresse et est ensuite acheminée vers l'objet stockant les objets de contenus **[25].**

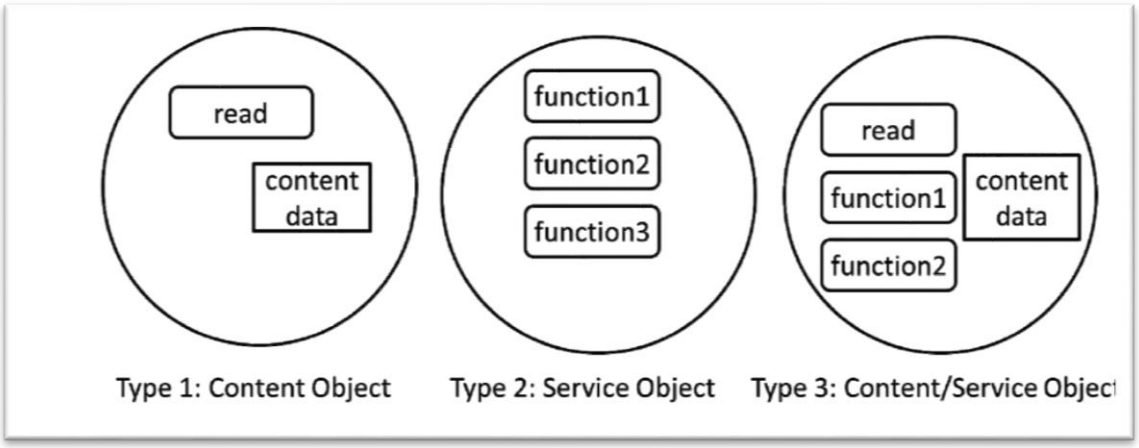


Figure 22: Types d'objets dans un SCN [25]

Comme nous pouvons le constater, cette approche nécessite une profonde modification des mécanismes de fonctionnement des réseaux CCN, ce qui peut être fastidieux car elle nécessite d'énormes moyens pour le déploiement des infrastructures dédiées au niveau des FAI, en plus de ceux au niveau des fournisseurs de services.

III.2.2. SOCCER

SOCCER est une approche basée sur ACO (ANT COLONY OPTIMIZATION), ce dernier est utilisé dans les réseaux informatiques pour résoudre des problèmes d'optimisation. **[26]**

Il modifie les messages de données et d'intérêt CCN pour activer une couche de contrôle qui est installée sur tous les nœuds et diffuse périodiquement des messages Ant-interest. Ces derniers sont utilisés envoyer des demandes pour des services distincts. Un message peut traverser l'ensemble du réseau jusqu'à arriver à un nœud desservant le service correspondant. Le nœud qui gère le service répond aux messages Ant-Interest avec un message Ant-Data, qui contient des informations sur l'état du serveur telles que l'utilisation du processeur ou la consommation de mémoire. Le message Ant-Data utilise les informations de fil d'Ariane laissées par les messages Ant-Interest pour arriver au demandeur de service avec succès. **[26]**

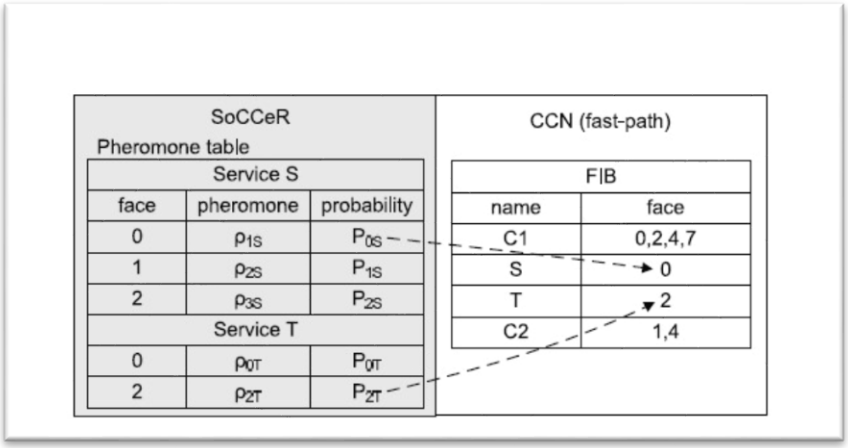


Figure 23: Nœuds dans SOCCER [26]

En ajoutant une couche supplémentaire au réseau CCN de base, cette architecture peut causer des problèmes de surcharge dans le réseau.

III.2.3. CCNxServ

CCNxServ est une implémentation des CCN qui fournit NetServ, un composant logiciel étendant le CCNx en prenant en charge les services.

CCNxServ nous permet de déployer des services de manière dynamique. Un nœud CCNxServ doit récupérer et déployer l'application de service à partir du réseau avant d'exécuter la demande de service. **[27]**

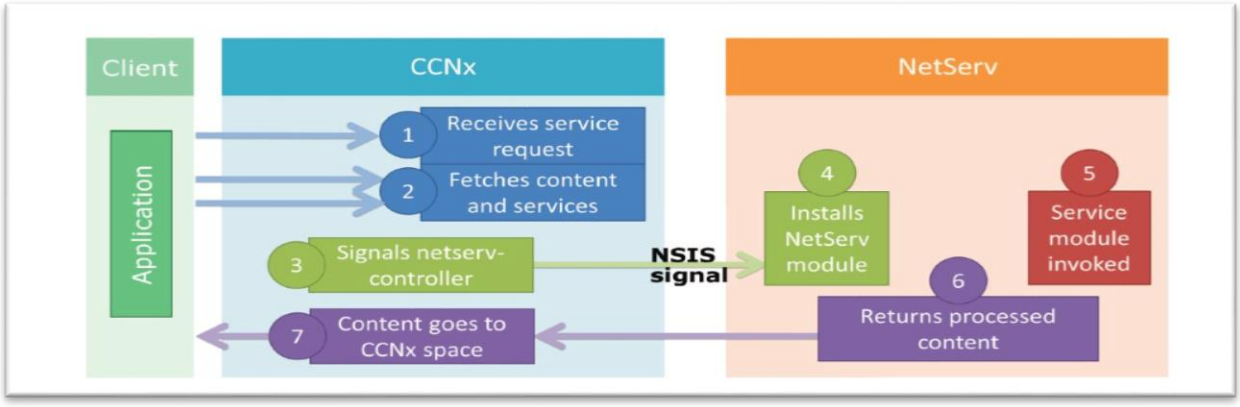


Figure 24:Intégration CCN et NetServ [27]

CCNxServ a de nombreuses limitations qui le rend presque non utilisable dans un environnement hétérogène comme internet par exemple, le fait qu'il ne prend en charge que les services fournis sous format de fichier JAR unique et implémentés exclusivement en Java, ce qui est contraire au principe d'interopérabilité des architectures orientées service.**[27]**

III.3. Présentation de notre architecture:

Nous proposons, dans cette section, une architecture permettant d'adapter les CCN à l'architecture SOA que nous avons nommé CCN-SOA et qui se veut légère. Nous prenons les éléments et concepts qui font la force de l'un et de l'autre, également les éléments nécessaires au bon fonctionnement des deux architectures afin de rendre l'architecture la moins surchargée possible et permettre également une facile mise en place.

Pour cette adaptation, les étapes qui suivent sont nécessaires :

- La mise en place d'une méthode de nommage de service ;
- L'adaptation des paquets intérêts/données à l'architecture ;
- La transformation des routeurs CCN en annuaire UDDI pour le routage vers les fournisseurs de service et l'accès aux documents WSDL ainsi qu'aux données des services ;
- Une méthode d'encapsulation des données SOA(SOAP) afin d'assurer leur transmission sur le réseau CCN.

III.3.1. Nommages des services

Dans les réseaux CCN, le routage se fait suivant les noms ainsi que la table de routage qui est représentée par la table FIB. Cette dernière ressemble fortement à une table composée d'un ensemble d'URL et cette méthode n'est pas totalement inconnue de l'architecture SOA vue que les accès aux services RESTFul aussi se font via des URL.

Dans ce sens il n'y aura pas énormément de transformation pour cette adaptation mais néanmoins nous proposons une hiérarchie spécifique pour le nommage des services comme la figure 25 le montre :

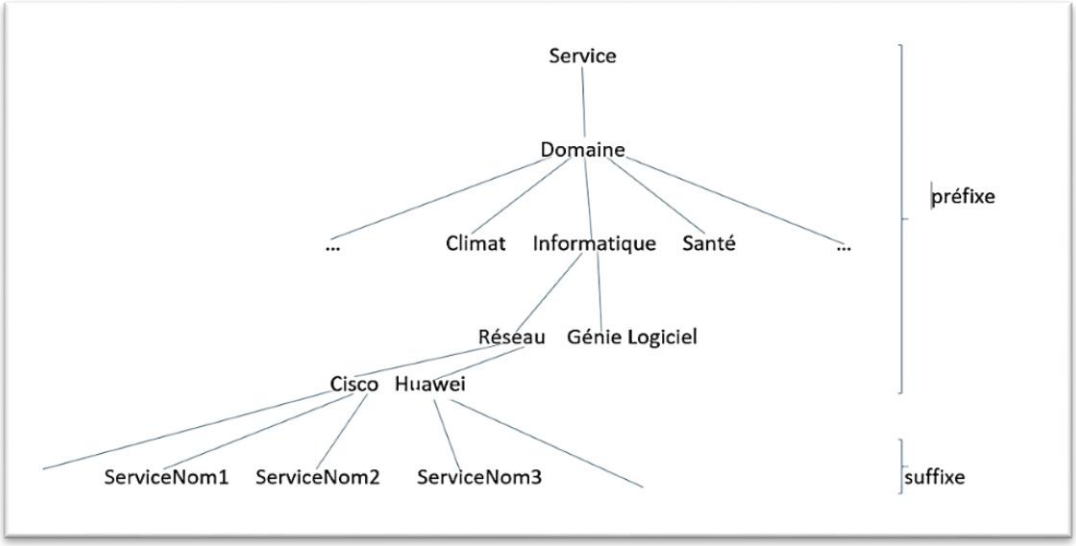


Figure 25: Préfixes et suffixes des services

Dans la figure 25 représentant une hiérarchie de noms, nous voyons que jusqu'à un certain niveau, nous sommes dans une généralité ce qui allège beaucoup le travail des routeurs.

La partie analysée par les routeurs dans un premier temps étant les préfixes, ceux-ci doivent être explicitement précisés dans la table de routage ou dans notre cas l'annuaire UDDI, nous détaillons cette partie dans la section III.3.2.

Le soin est laissé aux fournisseurs de nommer eux-mêmes leurs services mais ils doivent respecter la structure hiérarchique sous-jacente afin de garder une certaine cohérence.

III.3.2. Transformation (adaptation) des routeurs CCN en annuaire UDDI

Dans les réseaux CCN, chaque routeur contient un ContentStore, une table PIT et une table FIB pour le routage comme précisé dans le chapitre I.

Dans notre architecture, nous utiliserons les tables FIB et PIT comme annuaire UDDI ainsi grâce celles-ci, nous pourrons localiser chaque service de manière précise et accéder plus rapidement aux documents WSDL des services.

III.4. Fonctionnement de l'architecture

Suivant le même principe d'échange que les réseaux CCN classiques, l'architecture CCN-SOA fonctionne avec des échanges de deux types de paquets, les paquets intérêts qui sont également de deux types, les premiers serviront exclusivement pour la recherche d'un service et l'acquisition de son document WDSL et les seconds types sont utilisés pour les échanges entre client et serveur après établissement du lien entre eux. Quant aux paquets de données, ils ne changent pas vraiment, ils sont utilisés dans un premier temps pour transmettre le document WSDL des services qui sont des données aussi et sont utilisés après pour transmettre les autres données aux clients.

Au niveau d'un routeur, la recherche d'un service et la transmission de sa description WSDL se fait comme à la figure 26, où les parties encadrées en rouge se passent au niveau d'un routeur et la partie verte au niveau d'un serveur.

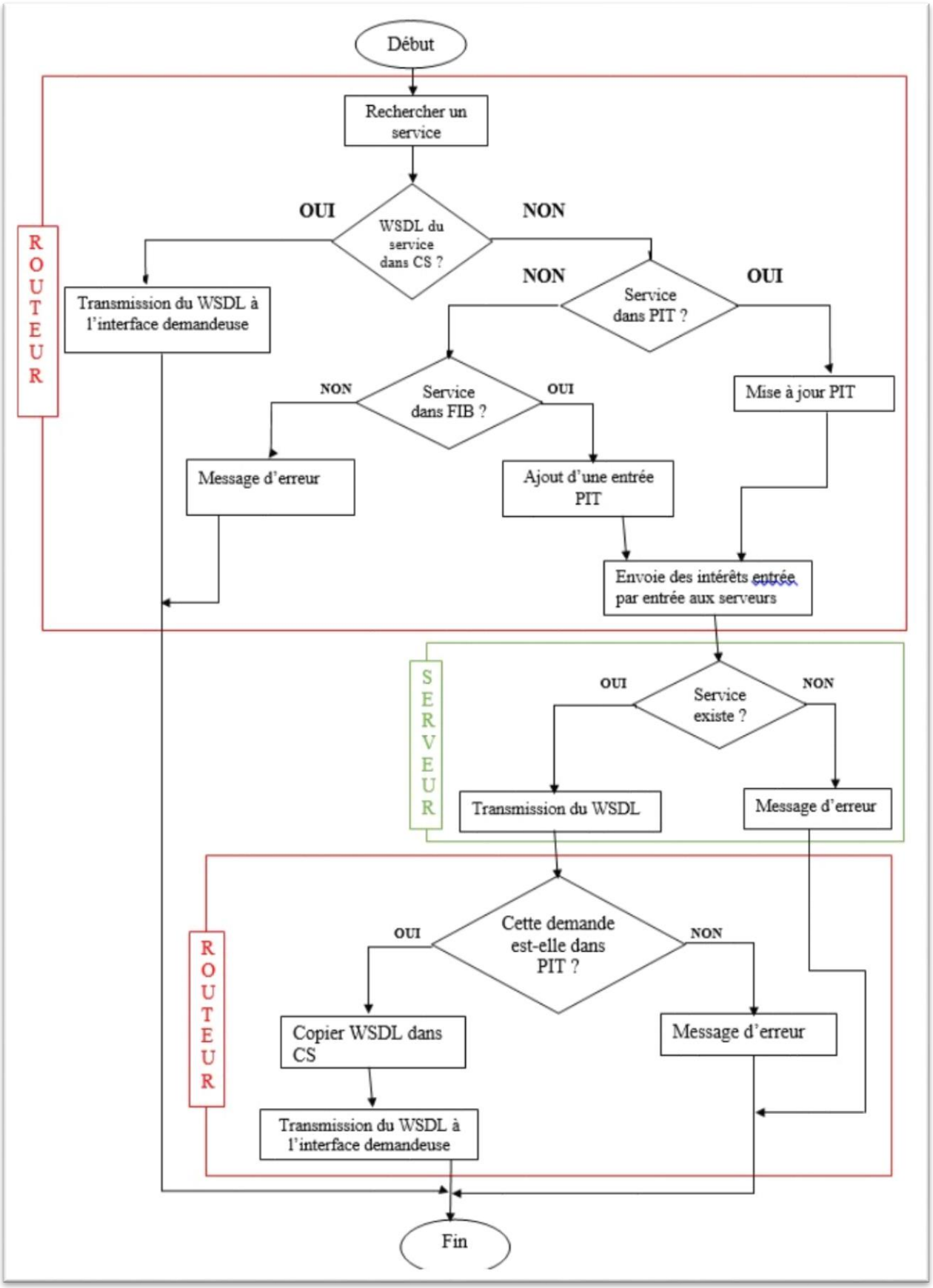


Figure 26: Organigramme de transmission d'un document WSDL

Lorsqu'un client effectue une recherche pour un service, une fois au niveau d'un routeur les choses se passent comme suit :

- Si le document WSDL du service se trouve dans son content Store, il l'envoie à l'interface d'arrivée sinon il vérifie dans sa table PIT, si il y'a déjà une demande pour ce même service, la table PIT est mise à jour en y ajoutant l'interface d'arrivée de la demande. Si la demande n'est pas dans la table PIT, le routeur vérifie sa table FIB si le service y est, une entrée est créée au niveau de la table PIT pour ce service, dans le cas échéant, la demande est rejetée car le service recherché n'existe pas.
- La table PIT contient alors les différentes demandes de service, elle les satisfait l'une après l'autre en les envoyant aux serveurs concernés qui répondront par des documents WSDL.
- Une fois qu'un serveur transmet un document WSDL, le routeur vérifie encore dans sa table PIT s'il y avait réellement une demande pour le service en question, si c'est le cas il copie le WSDL dans ContentStore et transmet le WSDL à l'interface demandeuse dans ce cas il rejette le document.

Ces documents contiennent la description des services ainsi que les informations à fournir par les clients pour établir un lien avec le fournisseur. En fournissant ces informations, les clients transmettent un second paquet intérêt pour se mettre en contact avec les serveurs de manière continue donc les routeurs ont juste à vérifier un nouveau champ que nous ajouterons aux paquets (figure 27) pour savoir s'ils sont du premier ou du deuxième type. S'ils sont du premier type, la recherche expliquée ci-dessus sera effectuée, s'ils sont au contraire du deuxième type, le routeur transmet juste le paquet à l'interface suivante pour atteindre le serveur.

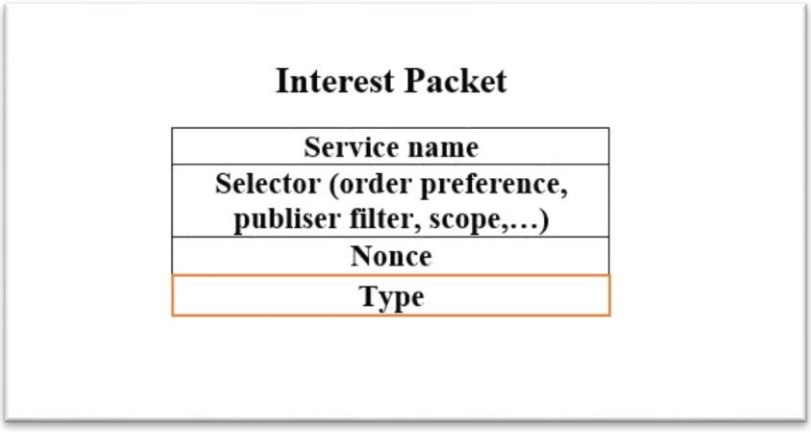


Figure 27: Paquet intérêt de notre architecture CCN-SOA

III.5. Encapsulation des données

Tous les échanges dans une architecture SOA (SOAP) se font via des documents XML encapsulés dans des enveloppes SOAP, qui à leur tour, sont encapsulées dans des enveloppes HTTP pour le transport, tout ce processus est spécifique au réseau TCP/IP.

Pour les réseaux CCN, au lieu d'être encapsulés dans les enveloppes HTTP, les enveloppes SOAP seront encapsulées dans des paquets d'intérêts ou de données.

Donc nos données seront encapsulées dans des enveloppes SOAP.

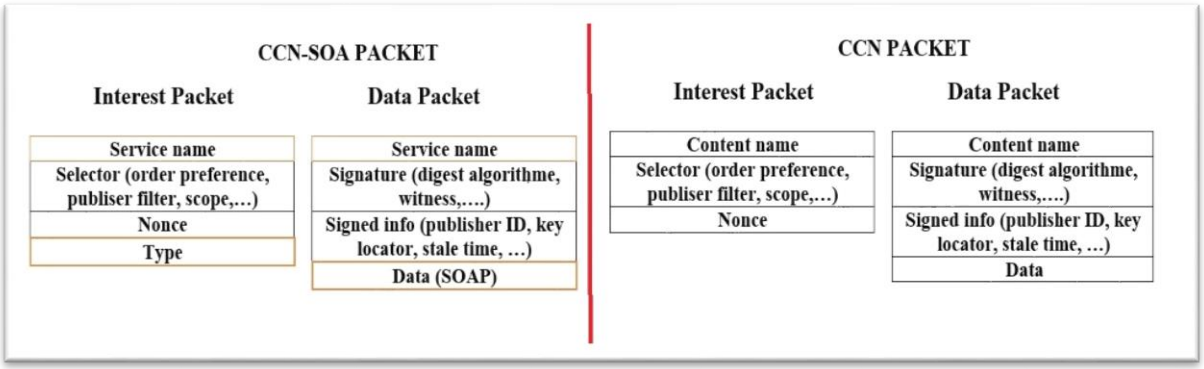


Figure 28: Comparaison des paquets CCN-SOA et CCN

III.6. Environnement logiciel de travail pour la simulation

Pour tester notre architecture, nous nous sommes naturellement dirigés vers la simulation qui est une solution plus facile et moins couteux.

Dans cette section, nous parlerons du simulateur OMNET++ qui est utilisé principalement pour les simulations réseaux et qui, en dépit de la complexité de la simulation d'une SOA, nous permettra de simuler notre architecture avec une petite topologie.

Nous parlerons également de CCN-lite qui est une implémentation légère des réseaux CCN et du Framework INET essentiel dans l'utilisation d'OMNeT++.

III.6.1. OMNeT++

OMNeT++ est une bibliothèque et un framework de simulation C++ extensible, modulaire et basé sur des composants, principalement pour la construction de simulateurs de réseau. « Réseau » est dit dans un sens plus large ce qui inclut les réseaux de communication câblés et sans fil, les réseaux sur puce, les réseaux de file d'attente, etc. **[40]**

OMNeT++ est principalement utilisé gratuitement et à des fins de simulations académiques ou dans le cadre de l'enseignement.

Le framework INET peut être considéré comme la bibliothèque de modèles de protocole standard d'OMNeT++. INET contient des modèles pour la pile Internet et de nombreux autres protocoles et composants. Le framework INET est maintenu par l'équipe OMNeT++ pour la communauté, en utilisant des correctifs et de nouveaux modèles fournis par les membres de la communauté. **[40]**

Parmi les fichiers dans OMNET++ nous pouvons trouver entre autre :

1- Fichiers **.ned :** ce sont des fichiers de description de topologie / réseau
2- Fichiers **.ini** : ce sont des fichiers d'initialisation
3- Fichiers de code : ce sont les fichiers de code **.cc** et **.h**

III.6.2. CCN-LITE

CCN-lite est une implémentation réduite et légère des protocoles CCN. Il est disponible sur plusieurs plates-formes, notamment :

- Espace utilisateur Linux et OS X
- Noyau Linux
- Android
- RIOT OS

CCN-lite a été conçu comme une base de code pour le travail en classe, les extensions expérimentales et les expériences de simulation. Il est aussi un excellent point de départ pour les produits commerciaux. **[41]**

III.6.3. Installation de CCN-Lite et INET dans OMNeT++

Pour installer CCN-Lite dans OMNeT++, nous avons suivi les étapes suivantes :

- Importer le Framework INET après son téléchargement dans le Workspace d'OMNeT++ en cliquant ***File*** **>** ***Import*** **>** ***Existing Projects into Workspace*** ;
- Clic droit sur INET puis sur ***Build Project*** ;
- Faire de même avec l'importation du fichier **ccn-lite-omnet ;**
- Clic droit sur le dossier ***ccn-lite-omnet*** **>** ***Clean Local***
- Clic droit sur le dossier ***ccn-lite-omnet*** **>** ***Clean Project***

III.6.4. Description des paquets Ccn-Lite dans OMNeT++

Le paquet CCN-Lite dans Omnet contient plusieurs fichiers, chacun ayant un rôle bien défini dans la simulation d'un réseau CCN. Nous avons:

- **CcnCore.{cc,h} :** fournit une interface C++ à Ccn-lite
- **Ccn.{cc,h,ned}** : implémente toutes les fonctionnalités liées à Ccn et hérite directement de CcnInet.
- **CcnInet.{cc,h,ned}** Intégration du cadre OMNet++ INET
- **CcnAdmin.{cc,h,ned}** Administrateur de scénario
- **CcnPacket_m.{cc,h}** Paquets échangés via les nœuds CCN dans OMNet++
- **CcnAppMessage_m.{cc/h/msg}** - Passage de messages qui sert à la communication entre les couches.

III.7. Matériels utilisés dans l'implémentation :

Le déploiement de ce genre d'architecture demande de grands moyens et d'infrastructures. Mais avec des moyens assez limités, nous nous sommes contentés d'un simulateur réseau utilisé dans des simulations académiques. Dans le tableau ci-dessous, nous présentons l'ordinateur utilisé ainsi que ses caractéristiques :

CPU	RAM	DISQUE DUR	SYSTEME D'EXPLOTATION	OMNeT++ Version	Framework INET Version	Paquet CCN-LITE
Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz 2.90 GHz	16GB	256 GB	Ubuntu 16.04.4 desk-top amd64	OMNeT++ 4.4.1	INET 2.6	CCN-Lite 0.3.0

Tableau 5: Caractéristiques physiques des matériels utilisés

III.8. Simulation

Si une chose est commune aux différentes littératures sur les architectures ayant pour but de combiner le CCN et le SOA, c'est bien la complexité de leur simulation, ce qui peut expliquer pourquoi beaucoup d'entre elles n'ont pas eu lieu.

Cette difficulté est due au fait que SOA est une architecture logicielle, c'est-à-dire qu'elle se trouve au niveau des dernières couches d'un standard de communication réseau et les simulateurs actuels CCN sont basés sur les couches basses.

Après avoir présenté les environnements matériels et logiciels, nous passerons à la simulation pour laquelle nous avons choisi un petit cas d'étude.

III.8.1. Cas d'étude : Services d'Acquisition de Documents en Ligne

Notre cas d'étude porte sur les services offerts par une bibliothèque, sur l'acquisition de documents en ligne pour les clients afin de leur permettre d'accéder rapidement aux documents demandés. Afin de mettre en œuvre notre service, nous proposons la hiérarchie de nommage illustrée à la **figure 29**.

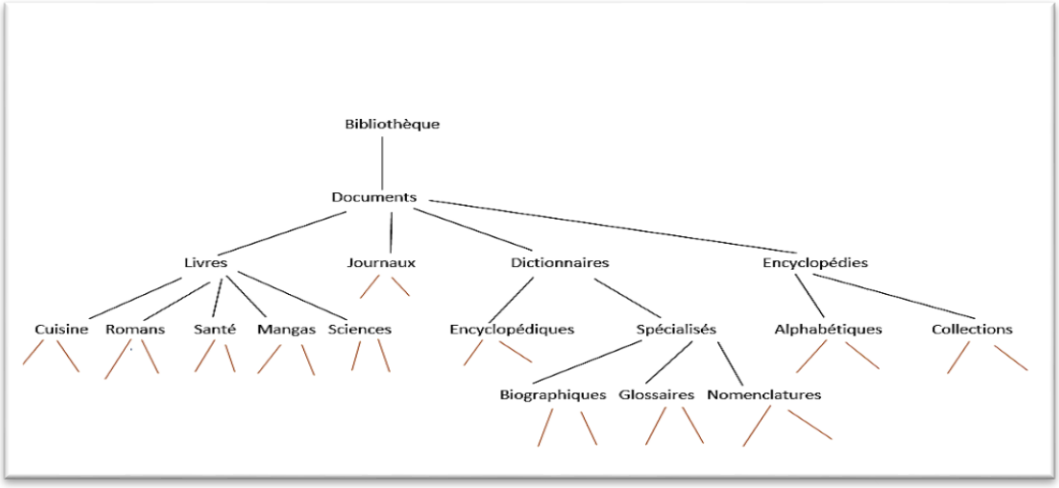


Figure 29: Services de la bibliothèque

Comme illustré dans la figure 29, la hiérarchie de nommage des services permet de fournir aux clients différents services ayant pour but de :

- Fournir des livres de cuisine, de science, de santé, des mangas et des romans ;
- Fournir des journaux ;
- Fournir des dictionnaires encyclopédiques ;
- Fournir des dictionnaires spécialisés : biographiques, glossaires et des nomenclatures ;
- Fournir des encyclopédies alphabétiques et de collection ;

III.8.2. Topologie utilisée

Notre topologie est composée de 6 clients, 4 routeurs et d'un serveur comme l'illustre la figure 30, les différentes étapes de la mise en place de cette topologie sont :

- **Etape1 : Mise en place de la topologie réseau**

Dans notre exemple, nous avons créé une topologie contenant 6 clients, 4 routeurs et un serveur comme illustré dans la figure 30.

Pour ce faire, nous avons utilisé le fichier « **ccn-lite-omnet/topology/CCNoEther_Tau_2cli_2rtr_1svr.ned »** se situant dans le fichier« ccn-lite-omnet » importé dans le simulateur OMNet++.

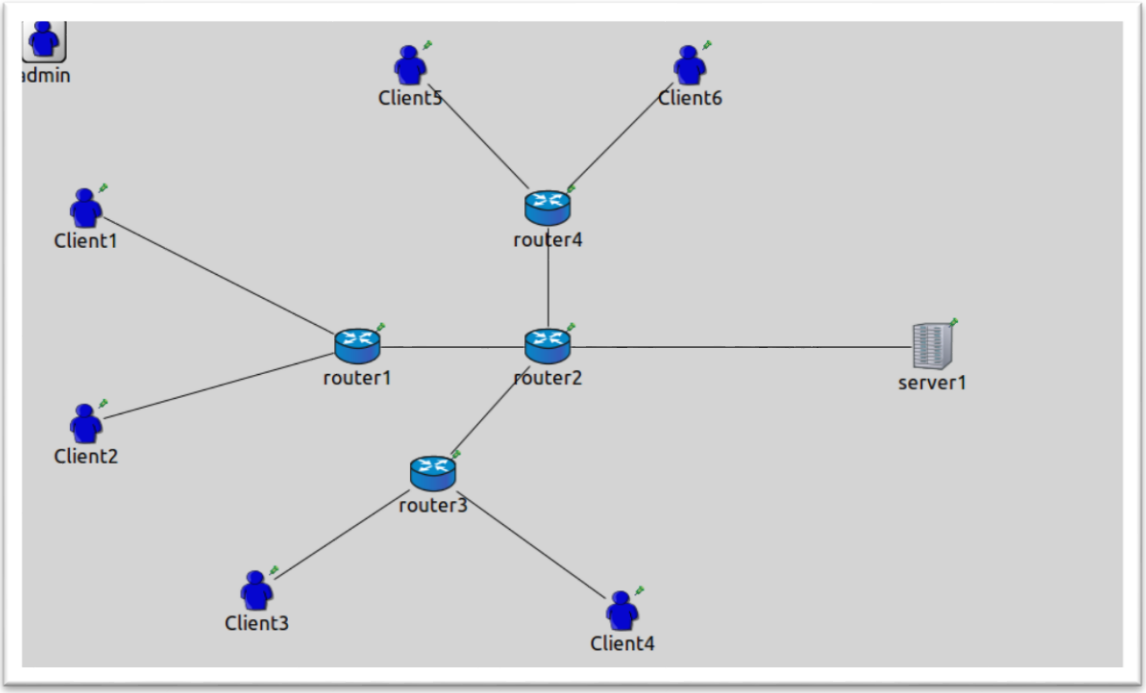


Figure 30: Topologie utilisée

Etape 2 : Connexion des nœuds

Dans cette étape, nous connectons les différents nœuds dans le fichier source de **ccn-lite-omnet/topology/CCNoEther_Tau_2cli_2rtr_1svr.ned** comme indiqué dans la figure 31 :

```
connections:

    Client1.ethg[0] <--> fastEthernet <--> router1.ethg[0];
    Client2.ethg[0] <--> fastEthernet <--> router1.ethg[1];

    router1.ethg[2] <--> fastEthernet <--> router2.ethg[0];
    router2.ethg[1] <--> fastEthernet <--> server1.ethg[0];
    Client3.ethg[0] <--> fastEthernet <--> router3.ethg[0];
    Client4.ethg[0] <--> fastEthernet <--> router3.ethg[1];
    router3.ethg[2] <--> fastEthernet <--> router2.ethg[2];
    Client5.ethg[0] <--> fastEthernet <--> router4.ethg[0];
    Client6.ethg[0] <--> fastEthernet <--> router4.ethg[1];
    router4.ethg[2] <--> fastEthernet <--> router2.ethg[3];
```

Figure 31: Connexion entre les nœuds

Etape3 : Paramétrage du réseau

Dans cette étape, nous connectons les nœuds aux fichiers de rôles portant l'extension **.cfg** qui y correspondent. Les fichiers **.cfg** existent dans le répertoire « **ccn-lite-omnet/simulations/example1 »** du réseau et sont connectés en utilisant le fichier source omnetpp.ini comme l'indique la figure 32 :

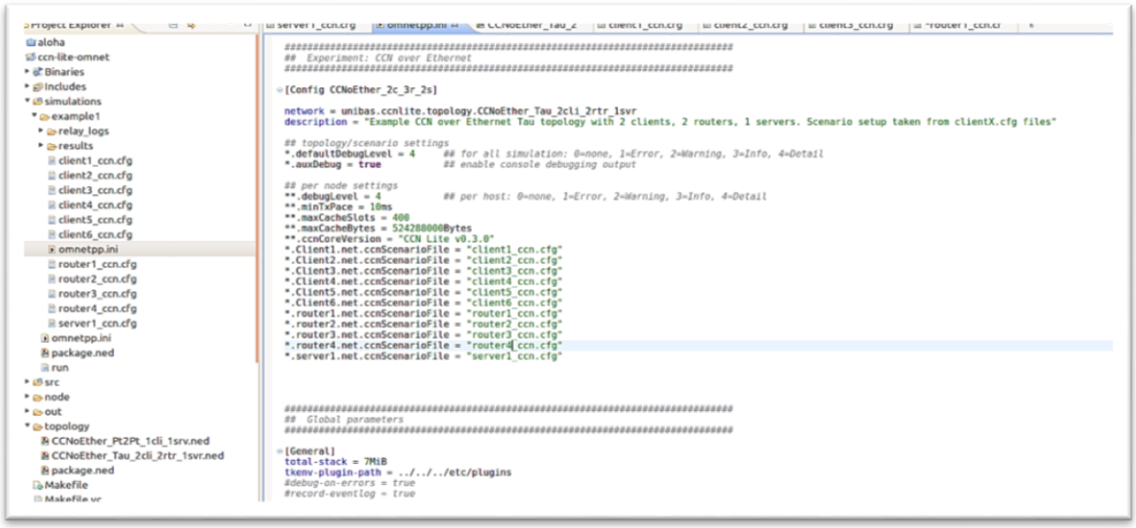


Figure 32: Connexion des nœuds aux fichiers .cfg

III.8.3. Mise en place des services

Comme indiqué dans la topologie représentée dans la figure 30, nous avons utilisé le serveur «server1 » pour stocker les différents services, comme illustrés dans la figure 33.

Pour cela, nous avons modifié le fichier **server_ccn.cfg** dans le dossier « **ccn-lite-omnet/simulations/example1 »** pour mettre en place la hiérarchie des services fournis par notre bibliothèque. Le fichier **server.cfg** contient tous les services fournis par la bibliothèque comme indiqué dans la figure ci-dessous :

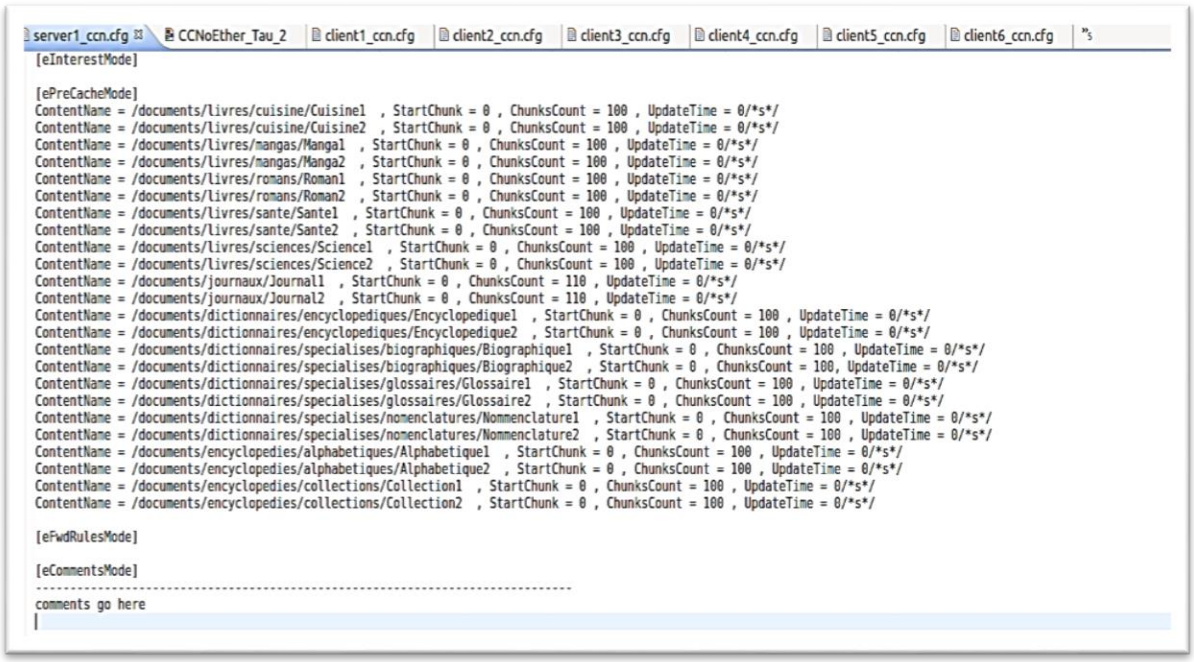


Figure 33: Contenu du serveur « server1_ccn.cfg»

Après, nous avons modifié la configuration des routeurs pour que les clients puissent envoyer des intérêts et recevoir des réponses sans entrave, comme l'indique la figure 34 représentant le contenu du routeur1.

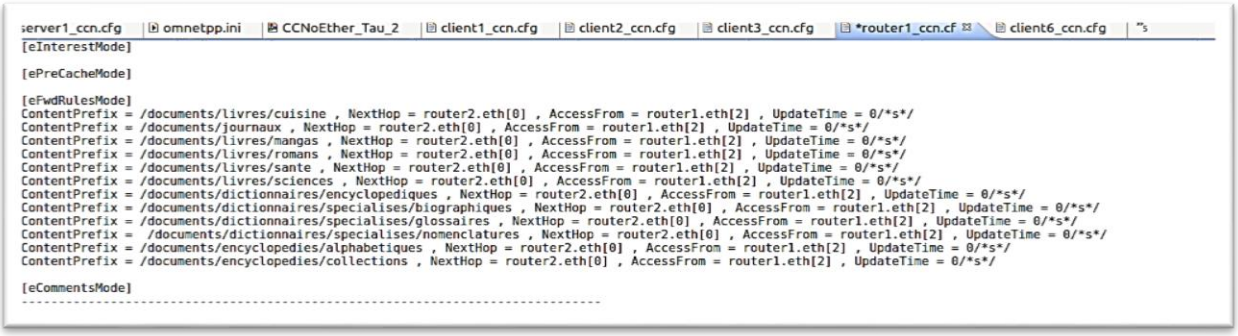


Figure 34: Contenu du routeur « router1_ccn.cfg»

- **Explication du contenu des fichiers «.cfg » :**

Chaque nœud CCN dans la topologie est relié à un fichier de configuration **.cfg**, le fichier est composé de trois parties principales et d'une partie pour les commentaires.

- **[eInterestMode]** : il représente la table PIT du nœud CCN et contient les attributs **ContentName** représentant le nom du contenu, **StartChunk** représentant le début du segment, ChunksCount représentant le nombre des segments et le RequestTime représentant le temps de réponse.
- **[ePreCacheMode]** : il représente le Content-Store du nœud CCN, il permet le stockage des données et a les mêmes attributs que le ContentName.
- **[eFwdRulesMode]** : il représente la table FIB, elle contient les informations de routage, et les attributs **ContentPrefix** définissant le préfixe du nom de contenu, **NextHop** définissant l'interface du nœud suivant, **AccessFrom** représentant l'interface de sortie et **UpdateTime** représentant le temps de mise à jour.
- **[eCommentsMode]** : il permet l'insertion des commentaires.

-Exemple de requête pour un service :

La figure suivante représente les services que le client1 recherche, nous constatons qu'il cherche deux services qui sont : cuisine1 et science2

```
eInterestMode]
ontentName = /documents/livres/cuisine/Cuisine1 , StartChunk = 0 , ChunksCount = 1 , RequestTime = 1/*s*/
ontentName = /documents/livres/sciences/Science2 , StartChunk = 0 , ChunksCount = 2 , RequestTime = 2/*s*/

ePreCacheMode]

eFwdRulesMode]

ontentPrefix = /documents/livres/cuisine , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/livres/sciences , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/


eCommentsMode]
--------------------------------------------------------------------------------
omments go here


ontentPrefix = /documents/livres/cuisine , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/livres/romans , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/livres/sante , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/livres/mangas , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/livres/sciences , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/journaux , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/dictionnaires/encyclopediques , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/dictionnaires/specialises/biographiques , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/dictionnaires/specialises/glossaires , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix =  /documents/dictionnaires/specialises/nomenclatures , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/encyclopedies/alphabetiques , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
ontentPrefix = /documents/encyclopedies/collections , NextHop = router1.eth[0] , AccessFrom = Client1.eth[0] , UpdateTime = 0/*s*/
```

Figure 35: Recherche de service par le client1

Quant au client4 il recherche un seul service qui est le journal1, la figure ci-dessous l'illustre :

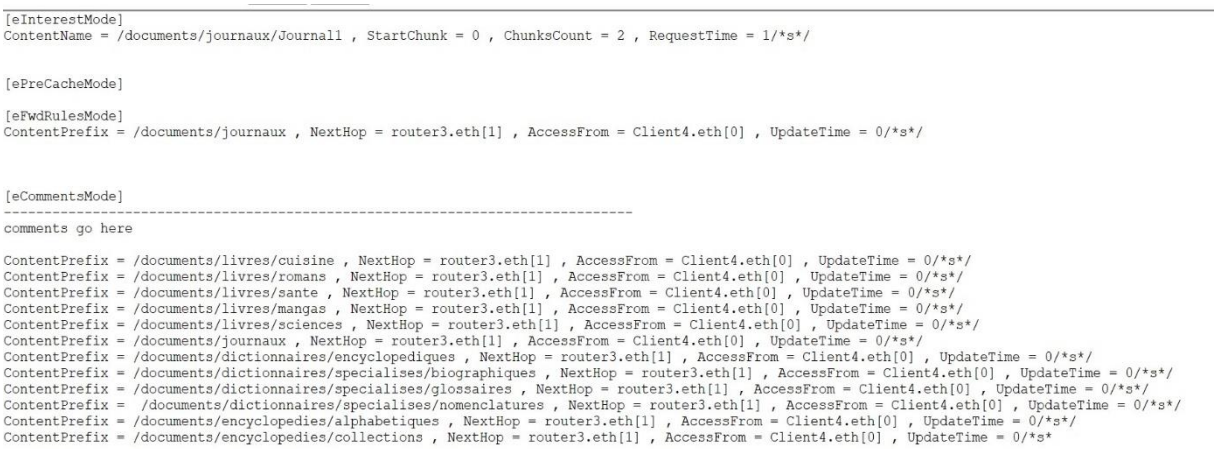


Figure 36: Recherche de service par le client4

III.8.4. Résultat de la simulation

Apres la simulation les résultats obtenus sont consignés dans le tableau suivant :

Client	**Service demandé**	**Moment d'envoi de la requête en seconde**	**Moment de réception de la description WSDL en seconde**
Client1	Cuisine1	1	1,0000
	Scienc2	1	2.0002
Client2	Encyclopedique1	1	1,0003
Client3	Collection1	1	1,0004
Client4	Journal1	1	1,0004
Client5	Glossaire1	1	1,0002
Client6	Science2	1	2,002

Tableau 6: Résultat de la simulation

Ce tableau contient le résultat de la simulation, il montre que la transmission des demandes ont été un succès. La transmission des descriptions WSDL qui est une réponse à ces de-

mandes, est très rapide donc nous pouvons conclure que l'architecture marche et offre une satisfaisante rapidité de transmission des données.

III.9. Conclusion

Les résultats de la simulation montrent qu'en dépit d'une mise en place difficile, il est bel et bien possible, avec une petite topologie d'avoir les résultats escomptés notamment avec une transmission rapide des données.

Une fois l'architecture mise en place, nous essayerons dans le chapitre suivant de renforcer notre solution par un mécanisme de sécurité, sachant que plusieurs failles de sécurité existent dans les CCN malgré tous leurs avantages, ceux-ci rendant vulnérable l'architecture proposée.

Chapitre IV : Implémentation d’une méthode de protection cryptographique

IV.1. Introduction

En dépit de tout ce qu'ils apportent, les réseaux CCN ont leurs lots de vulnérabilités aux attaques notamment celle du déni de service et de l'homme du milieu donc notre architecture CCN-SOA s'en trouve forcément affectée.

Beaucoup de solutions ont été proposé pour régler ces problèmes et chacune de ces solutions a ses forces et faiblesses.

La cryptographie étant l'une de ces solutions, elle consiste à crypter les données en vue de respecter leur confidentialité pour ne permettre qu'aux personnes autorisées d'avoir accès aux données non cryptées. Le problème lié aux solutions cryptographiques est très souvent l'échange des clés de cryptage et décryptage.

La cryptographie homomorphe est une nouvelle forme de cryptographie tentant bien que difficilement de résoudre ce problème, ainsi nous nous sommes orientés vers le crypto système RSA, qui est l'un des trois types de crypto-système homomorphe existants, pour crypter les données échangées sur notre architecture.

Dans ce chapitre, nous parlons en premier de l'attaque homme du milieu, ensuite de quelques solutions cryptographiques existantes pour crypter les paquets échangés sur un réseau, puis, de la cryptographie homomorphe et de ce qu'elle pourrait apporter. Pour finir nous parlerons de notre solution basée sur le crypto-système RSA pour crypter les données sur notre architecture CCN-SOA et de sa simulation.

IV.2. Attaque de l'homme du milieu

Le principe de l'attaque de l'homme du milieu est l'interception de données cryptées échangées entre deux entités afin d'en déchiffrer le contenu. L'attaquant se place au milieu afin de recevoir toutes les informations envoyées par les deux parties, et ainsi pouvoir se faire passer pour l'un deux. Cette attaque affecte surtout la confidentialité, et dans certains cas à l'intégrité des données car l'objectif de ce type d'attaque est l'espionnage, ou la modification des données. Les réseaux CCN sont vulnérables à ce genre d'attaques, car un attaquant peut intercepter le paquet de donnée au retour affectant ainsi la confidentialité des données. Pour protéger contre ce genre d'attaque, la solution la plus sûre est la mise en place d'une solution cryptographique afin que la confidentialité des données reste intacte. C'est dans cette optique que nous nous sommes tournés vers cette solution afin de protéger notre architecture.

IV.3. Solutions cryptographiques existantes

Contrairement aux réseaux modernes, CCN promeut la sécurité des données plutôt que la sécurité des transports. Cela signifie que les données sensibles ou confidentielles doivent être cryptées. **[28]** Donc même si un attaquant parvient à avoir accès aux données, elles lui seront inutiles étant donné qu'il ne possède pas la clé de décryptage.

Dans la section qui suit nous parlerons de deux de ces solutions que nous avons jugé très pertinentes.

IV.3.1. Secure Content Delivery in Information-Centric Networks [30]

Un Framework de cryptage de données sur les réseaux CCN a été proposé par S.Misra et al., et il est basé sur le système de partage de Shamir **[29]**.

Le processus de cryptage des données se déroule en trois phases :

- Le serveur génère pour chaque nouvel utilisateur connecté un couple de clé, une partie de la clé est gardée par le serveur pour crypter les données qu'il transmettra ensuite à cet utilisateur, il transmet l'autre partie de la clé à l'utilisateur en cryptant cette partie de la clé avec la signature et le délai d'attente afin que l'utilisateur puisse déchiffrer la clé secrète **[30]** ;
- Ensuite, pour communiquer avec chaque utilisateur, les paquets sont disposées en bloc qui sont cryptés en utilisant la clé au niveau du serveur et ces blocs contiennent une autre clé secrète qui décrypte les blocs assemblés
- Une fois que l'utilisateur en possession de ces blocs, il les décrypte avec sa clé, il les assemble et il utilise la seconde clé secrète contenue dans les blocs pour décrypter les blocs une fois assemblés.

IV.3.2. Towards Name-based Trust and Security for Content-centric Network[34]

Cette approche est basée sur la cryptographie basée sur l'identité (Identity-Base Cryptographic ou IBC en anglais), elle crypte les données avec l'identité du client.
Le système IBC comprend un générateur de clés privées (PKG) qui génère une clé secrète principale (MSK) et des paramètres principaux du système (SP).
Le MSK est gardé secret et utilisé par le PKG uniquement pour générer les clés privées correspondantes pour les utilisateurs individuellement et SP est publié. Tout utilisateur peut utili-

ser le SP publié et son identité pour générer des clés publiques pour les autres utilisateurs et fournisseurs. **[34]**

IBC comprend deux crypto-systèmes : IBS **[35]** (Identity-based signature) qui utilise l'identité du fournisseur de donnée pour signer la donnée et IBE **[36]**(Identity-based encryption) qui, quant à lui, utilise l'identité des utilisateurs pour crypter la donnée. Cette approche combine ces deux crypto-systèmes pour, d'un côté, signer les données par les fournisseurs et, d'un autre côté, les crypter par les identités des clients.

Mais ce qui nous intéresse concrètement ici, c'est le cryptage des données donc nous allons nous concentrer sur l'IBE.
L'IBE est un schéma de cryptage composé de quatre algorithmes : setup, extract, encrypt, decrypt.

- Setup (Lk) : PKG exécute cet algorithme prenant un paramètre Lk et génère le MSK et le SP. MSK est secret et SP est publié ;
- Extract (MSK, ID) : génère une clé privée SK pour l'utilisateur ID avec les paramètres MSK et ID ;
- Encrypt (SP, ID, M) : encrypte le message M avec l'identité ID d'un utilisateur pour générer un texte de Cipher C ;
- Decrypt (SK, C): décrypte le texte de Cipher pour donner le message M.

Tout ce processus est résumé à la figure 37**:**

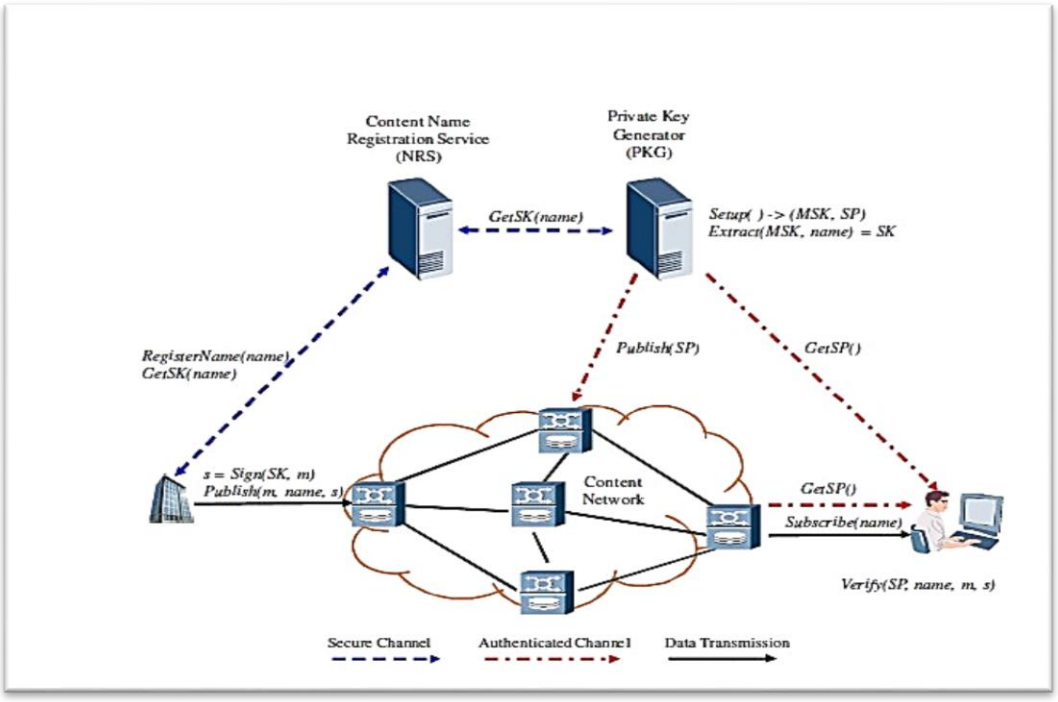


Figure 37: Fonctionnement Towards Name-based Trust and Security for Content-centric Network [7]

L'expéditeur peut utiliser l'identité du récepteur pour chiffrer les données et les publier.
Une fois reçues, les données peuvent être décryptées par le récepteur s'il peut obtenir le SK de la PKG. **[34]**

IV.3.3. Problèmes liés à ces solutions

Les méthodes de cryptographie présentées ci-dessus pour crypter les données sur le réseau CCN utilisent des crypto-systèmes symétriques donc elles ont principalement comme problème le fait qu'il devrait y'avoir au préalable un échange de clé entre le client et le fournisseur, malheureusement, ces clés sont échangées en clair, dès qu'un attaquant accède à cette clé il pourra décrypter les données.

En plus de ce problème, les architectures SOA étant beaucoup utilisées pour les services de Cloud Computing et IoT (Internet of Things) pour des fins de conservation et de traitement, un autre problème essentiel subsiste au niveau de la confidentialité entre le client et le fournisseur, le client est obligé d'utiliser des données non cryptées pour effectuer des opérations fournies pour ce fournisseur. De ce fait les informations du client seront lisibles pour le fournisseur même s'il s'y oppose.

Pour endiguer ces problèmes, les cryptographies homomorphes offrent des solutions très satisfaisantes.

IV.4. Cryptographie Homomorphe

En plus de la confidentialité des données vis-à-vis des attaquants, la démocratisation des services de Cloud Computing et Internet of Thing dont les traitements de données sont effectués par des fournisseurs ayant accès à des informations potentielles des clients, la confidentialité vis-à-vis des fournisseurs s'impose, donc une évolution des méthodes de cryptographie étaient fondamentalement nécessaires afin de protéger la vie privée des clients.

La cryptographie homomorphe répond justement à ce besoin car elle permet d'effectuer des opérations sur des données cryptées, donc elle offre la possibilité aux utilisateurs d'envoyer des données cryptées aux fournisseurs, toutes les opérations effectuées sur ces données cryptées donnent le même résultat que si elles ont été effectuées sur les données non cryptées.

L'idée de la cryptographie homomorphe est née pour la première fois en 1978 grâce aux chercheurs Rivest, Adleman et Dertouzos **[31]** qui voyaient sa nécessité dès cette époque et où il était question d'effectuer des opérations mathématiques sur des données chiffrées.

Trente ans plus tard en 2009, Craig Gentry publia sa thèse **[32]** ce qui donna un nouveau souffle à ce domaine si prometteur et posant les bases des crypto-systèmes homomorphes actuels.

Un schéma de chiffrement homomorphe est constitué formellement de quatre algorithmes (KeyGen, Enc, Dec, Eval) probabilistes s'exécutant en temps polynomial et qui fonctionnent comme suit **[33]** :

- Un algorithme de génération de clé KeyGen prend en entrée un paramètre de sécurité 1^{λ}, et retourne des paramètres publics PP, une clé secrète *sk*, une clé publique *pk*.

- Un algorithme de chiffrement Enc qui prend en entrées les paramètres publics PP, un message clair *m*, la clé publique *pk* et retourne un chiffré *c*, *c*= Enc (*pp*, *pk*, *m*).

- Un algorithme de déchiffrement Dec qui prend en entrées les paramètres publics PP, un chiffré *c*, la clé secrète *sk* et retourne un message M' = Dec (*pp*, *sk*, *c*).

- Un algorithme d'évaluation Eval qui prend en entrées les paramètres publics PP, la clé publique *pk*, un circuit C défini sur *n*, des chiffrés (*c*1, *c*2, . . ., *cn*) portant sur les mes-

sages ($m1$, $m2$, . . ., mn) respectivement. Il retourne un chiffré c = Enc (pp, pk, C ($m1$, $m2$, . . ., mn)).

Il existe trois grandes classes de schémas homomorphe :

- Les schémas partiellement homomorphes (Partially homomorphic scheme ou PHE en anglais) ;
- Les schémas presque homomorphes (somewhat homomorphic scheme ou SHE en anglais) ;
- Et pour les schémas totalement homomorphes (Fully homomorphic scheme en anglais)

Le schéma PHE dont l'illustration a été faite par Rivest et al., est un crypto-système à clé publique RSA. Pour un module n et un exposant publique e, nous remarquons qu'à partir de deux messages chiffrés $m1$ et $m2$, il est possible d'obtenir celui de leur produit $m1.m2$ **[33]**:

E (*m*1) .E (*m*2) = $m1^e.m2^e$ **mod *n* =** $a(m1.m2)^2$ **mod *n* = E (*m*1. *m*2).**

La mise en place de schémas entièrement homomorphes qui est l'objectif réel de la cryptographie homomorphe, veut que le nombre et type d'opérations sur les données chiffrées ne soit pas limité, ce que RSA ne donne pas clairement, d'ailleurs c'est pour cette raison que RSA est classé parmi les schémas partiellement homomorphe car le nombre d'opérations pouvant être effectuées sur les données cryptées est limité.

La mise en œuvre d'algorithmes de cryptographie entièrement homomorphes est—extrêmement difficile et dépend surtout de la puissance de calcul de l'ordinateur, donc nécessite des moyens colossaux, alors pour sécuriser notre architecture CCN-SOA sans alourdir le système, nous avons choisi le chiffrement à base de RSA qui est un schéma partiellement homomorphe.

IV.5. Méthode de chiffrement RSA

L'algorithme de cryptage RSA est l'algorithme cryptographique à clé publique le plus utilisé. La difficulté de factorisation de grands entiers dans l'algorithme RSA fait sa force en termes de sécurité. C'est pourquoi, plus les entiers sont grands, plus RSA est sécurisé. La difficulté de factoriser n pour trouver les nombres premiers originaux p, q définit la force de RSA.

L’algorithme de RSA possède une clé publique pour le chiffrement (e) et une clé privée pour le déchiffrement (d). La mise en place de l’algorithme RSA se résume à trois étapes principales.

IV.5.1. Génération de clés

Dans cette étape, les deux clés (publique et privée) sont générées afin de pouvoir être utilisé pour le chiffrement et le déchiffrement.

Ainsi, voici les différentes étapes à suivre pour la génération des deux clés :

- Choisir deux grands nombres premiers p et q.
- Calculer le nombre de modules n = p x q.
- Calculer la fonction d'Euler (n) = (p-1) x (q-1).
- Sélectionnez un nombre entier e au hasard comme clé publique. Il doit satisfaire le plus grand diviseur commun PGCD(e, φ(n)) = 1, 1< e < φ(n).
- Calculer la clé privée d telle que d x e =1(mod φ(n)).

IV.5.2. Cryptage

C'est la transformation des données en une forme qui devient aussi difficile que possible à lire sans les connaissances appropriées (une clé). Dans l'algorithme RSA, le texte chiffré est généré par l'équation ci-dessous.

$$C=Me \bmod n$$

IV.5.3. Décryptage

C'est la transformation d’un texte clair en une forme impossible à lire sans les éléments nécessaires. Dans l'algorithme RSA, le texte chiffré est généré par

$$M=Cd \bmod n$$

Ainsi, l’utilisateur peut utiliser cette formule pour décrypter la donnée dédiée à lui seul.

IV.6. Implémentation de la méthode de chiffrement RSA

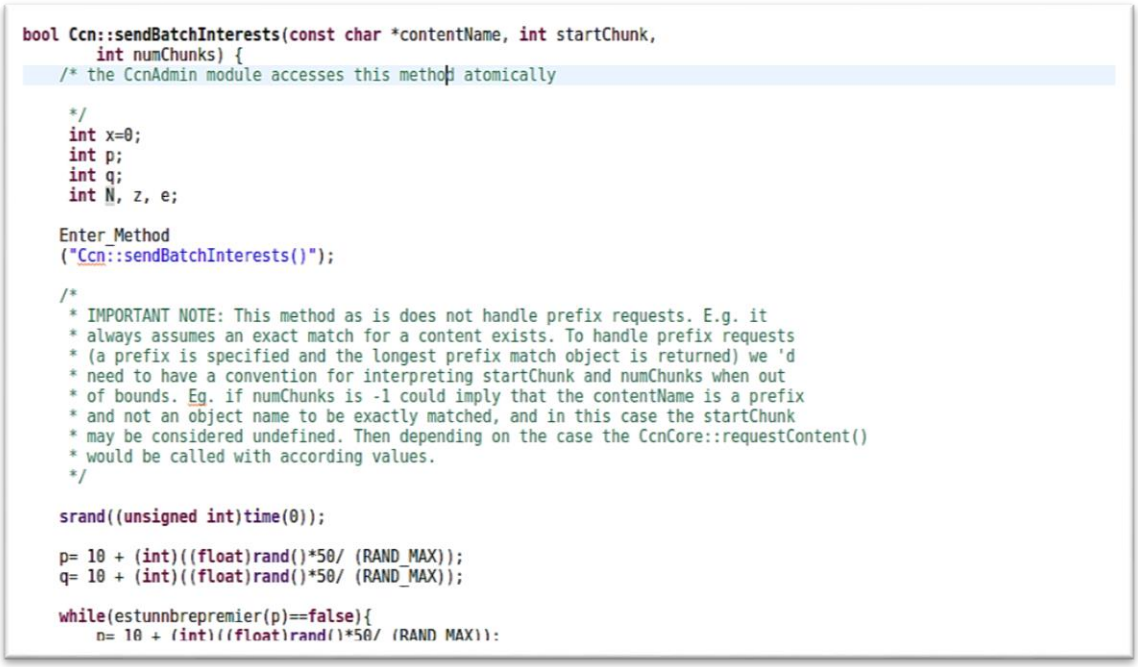
Avant de présenter le travail effectué, nous allons détailler le fonctionnement de la solution que nous avons mis en place et les étapes suivies :

- Pour crypter et décrypter nos données, la première des choses est l'échange des clés. Pour se faire nous avons ajouté un nouveau champ aux paquets qui transitent dans le réseau pour y insérer la clé publique des clients.
- Lorsqu'un client doit envoyer un paquet intérêt pour un service, les deux clés (publique et privé) de notre système sont automatiquement générées pour le client. Il garde la clé privée et la clé publique est ajoutée à son paquet intérêt.
- Une fois, un service trouvé pour sa demande, au niveau du serveur, la donnée est cryptée avec la clé publique contenue dans le paquet intérêt avant d'être d'encapsuler dans le paquet de données et retransmis au client qui pourra le décrypter en utilisant sa clé privée.

Nous avons pris pour but de ne crypter que la donnée car elle est l'élément le plus important à protéger contre les attaques, par exemple l'attaque de l'empoisonnement du contenu, pour ainsi pouvoir garder l'intégrité et la confidentialité des données.

Pour l'implémentation, nous avons utilisés les classes :

- Ccn.cc et Ccn.h, plus précisément la fonction sendBatchInterests pour la génération des deux clés.



```
bool Ccn::sendBatchInterests(const char *contentName, int startChunk,
      int numChunks) {
    /* the CcnAdmin module accesses this method atomically

     */
     int x=0;
     int p;
     int q;
     int N, z, e;

    Enter_Method
    ("Ccn::sendBatchInterests()");

    /*
     * IMPORTANT NOTE: This method as is does not handle prefix requests. E.g. it
     * always assumes an exact match for a content exists. To handle prefix requests
     * (a prefix is specified and the longest prefix match object is returned) we 'd
     * need to have a convention for interpreting startChunk and numChunks when out
     * of bounds. Eg. if numChunks is -1 could imply that the contentName is a prefix
     * and not an object name to be exactly matched, and in this case the startChunk
     * may be considered undefined. Then depending on the case the CcnCore::requestContent()
     * would be called with according values.
     */

    srand((unsigned int)time(0));

    p= 10 + (int)((float)rand()*50/ (RAND_MAX));
    q= 10 + (int)((float)rand()*50/ (RAND_MAX));

    while(estunnbrepremier(p)==false){
        p= 10 + (int)((float)rand()*50/ (RAND_MAX));
```

Figure 38: Aperçu du code dans Ccn.cc

- CcnPacket_m.cc et CcnPacket_m.h : pour l'ajout des attributs au paquet CcnPacket ;
- CcnCore.cc et CcnCore.h : pour l'envoi des clés et le cryptage des données dans le réseau. Dans cette classe, nous avons étudié les fonctions qui sont déjà existantes et qui pourraient nous aider telles :

- La fonction requestContent utilisé pour la récupération des clés;
- La fonction opp_deliver pour la livraison des clés ;
- La fonction toMacFace pour le cryptage des données avant l'encapsulation.
- La fonction toMacFace pour l'envoi du double intérêt ;

En plus d'avoir utilisé les fonctionnalités offertes par ces différentes fonctions, nous avons créé certaines fonctions indispensables au processus, comme le montre la figure ci-dessous :

```
    ~CcnCore();

    CcnCore(Ccn *owner, const char *nodeName, const char *coreVer);

    /**
     * API: Ccn omnet module --> CCN Lite core
     */
    /** Some text info about the interface the module creates */
    std::string info();

    /** @brief Update relay configuration (e.g. cache store size, cache policy, tx pace etc) */
    bool updateRelayConfig(void *);

    /** Add to cache a buffer of content chunks of equal size. Return the number of chunks successfully added to C
    int addToCacheFixedSizeChunks(const char *contentName,
            const int seqNumStart, const int numChunks, const char *chunkPtrs[],
            const int chunkSize);

    /** Add a FIB rule using L2 ID (MAC address). Return success/fail */
    bool addL2FwdRule(const char *contentName, const char *peerAddr,
            int localNetifIndex);

    /** Express Interest for named content. Return 1/true on success or 0/false on failure */
    bool requestContent(const char *contentName, const int seqNum, int publickey, int N);

    /** Pass packet received from the MAC layer to the relay. Return 1/true on success or 0/false on failure */
    bool fromMACFace(CcnContext *ccnCtx, int arrNetIf, CcnPacket *ccnPkt);

    int* transform_into_ascii(char* T);
    int taille(char * T);
    int powermodulo(int x, int y, int p);
    int* Cryptage(int* T, int e, int n);
    int* Deryptage(int* T, int d, int n);

    /**
     * API: Ccn omnet module <-- CCN Lite core
     *
     * these methods are accessed by the CCN lite code through C wrappers in CcnCore.cc
```

Figure 39: Aperçu des fonctions utilisées dans CcnCore.h

- ccnl-uapi.cc, ccnl-uapi.h et ccnl-core.h: pour la liaison de l'envoi des données dans le réseau.

IV.7. Résultat de l'implémentation

En exécutant le code que nous avons implémenté et en utilisant la fonction d'affichage dans la console afin de constater le résultat du travail, nous remarquons une génération automatique de la clé pour chaque intérêt :

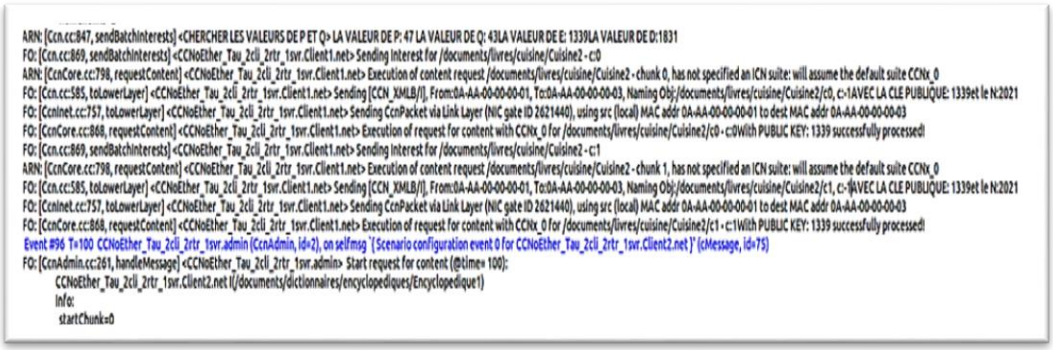


Figure 40: Aperçu de la génération de la clé

Après la génération, nous avons crypté l'attribut data de CcnPacket en lui affectant une valeur pour pouvoir vérifier le résultat de la simulation car OMNet++ étant un simulateur, le paquet data est vide. Après cryptage, nous obtenons le résultat suivant :

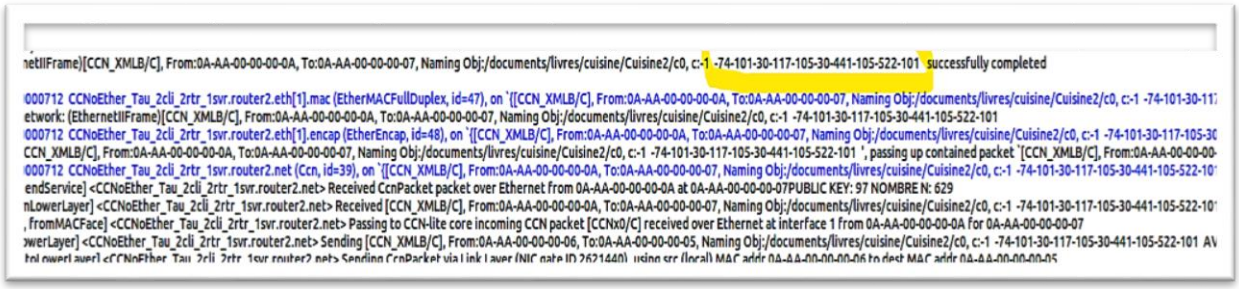


Figure 41: Aperçu de la donnée cryptée

Dans le résultat ci-dessus, au retour, les paquets de données [CCN_XMLB/C] sont cryptés. A l'arrivée des paquets de données vers l'interface demandeuse, dans notre cas les clients, il pourra être décrypté avec leur clé privée (d, N).

IV.8. Conclusion

Dans ce chapitre, nous avons procédé à la sécurisation de notre architecture CCN-SOA. Pour arriver à cette fin, nous avons crypté nos données grâce à l'algorithme de cryptographie asy-

métrique et partiellement homomorphe RSA, les différentes simulations nous ont donné des résultats satisfaisants qui nécessitent quand même quelques ajustements et améliorations.

Conclusion générale

L'accroissement de la demande de connectivité a vu internet se développer très rapidement, des années 70's à maintenant son utilisation s'est très diversifiée du web classique à internet des objets, en passant par les streaming audio et vidéos, cet accroissement et cette diversité se sont faits accompagnés par l'explosion de la quantité de données à transporter donc une amélioration des protocoles de transport s'est imposée. Les réseaux CCN proposent justement des protocoles pour répondre à ces problématiques.

En plus de ça, s'ajoute la nécessité d'interopération entre les systèmes informatiques ainsi qu'une composition entre eux pour d'une part, favoriser la réutilisation et d'autre part réduire les couts de développement notamment dans les entreprises pour se faire il fallait une architecture de développement répondant à ces besoins précis. De cette nécessité est sont nées les architectures orientées service, créées autour de propriétés pour palier à ces problèmes.

À l'apparition des réseaux CCN en 2008 soit 8 ans après les SOA (2000-2001), les scientifiques ont vite compris qu'il fallait une adaptions de ces réseaux pour supporter l'architecture SOA car ayant fait l'objet de beaucoup de recherches et de solutions proposées comme les quelques-unes citées dans ce mémoire.

Dans ce mémoire, nous avons proposé une solution pour cette adaptation que nous avons nommé architecture CCN-SOA, nous avons proposé une solution qui se différencie des autres par sa simplicité et sa sécurité. Pour se faire nous avons choisi une solution de bout-à-bout c'est-à-dire allant d'une méthode spécifique de nommage des services jusqu'à une adaptation des paquets de données et d'intérêt qui transitent sur notre réseau pour une architecture aussi légère que possible.

Notre architecture a naturellement hérité des failles des réseaux CCN donc vulnérable à certaines attaques comme les attaques sur le cache et celle de l'homme du milieu contre laquelle nous proposons une solution dans la deuxième partie de ce travail.

Pour simuler notre architecture dans un premier temps nous avons mis en place une petite topologie qui nous a fournis des résultats satisfaisants.

Pour la sécurité, nous nous sommes orientés vers la cryptographie homomorphe mais à défaut d'appliquer un crypto-système entièrement homomorphe pour des raisons liées au coût et à la légèreté, nous avons choisi la méthode de cryptographie RSA qui est partiellement homomorphe pour crypter nos paquets.

Ainsi dans un second temps on a implémenté des méthodes de générations automatiques de clés publiques et privées pour RSA et crypté nos données pour enfin appliquer ceci à notre topologie, avec une fois de plus des résultats satisfaisants.

Ayant au cours de ce travail, toucher à des domaines comme le développement logiciel avec l'architecture SOA, les réseaux TCP//IP notamment leurs limites pour les demandes présentes et futures des utilisateurs, les réseaux ICN plus précisément les réseaux CCN et la cryptographie homomorphe nous a permis d'améliorer considérablement notre niveau dans les domaines cités et également repérer les parties de notre architecture qui nécessitent d'éventuelles améliorations.

Bien que nécessaire, nous n'avons pas pu tester notre architecture de manière exhaustive, cela dû à une indisponibilité d'environnement de simulation pouvant simuler celle-ci. La difficulté de la simulation d'une telle architecture réside dans le fait que l'architecture SOA est une architecture logicielle donc l'exécution de ces parties liées à la connectivité réseau s'effectue au niveau des dernières couches des standards de communication réseaux notamment la couche application et les outils de simulation des réseaux CCN sont basés sur les premières couches c'est-à-dire de la couche physique à la couche réseau.

Comme perspectives, nous pouvons principalement nous tourner vers ces deux problématiques :

- En premier lieu, implémenter une SOA entière et simuler l'architecture CCN-SOA de manière exhaustive avec un outil de simulation réseau pouvant effectuer ce travail ;
- En second lieu, utiliser la signature numérique pour vérifier l'identité des différentes entités dans le réseau afin de protéger l'intégrité des données ou utiliser un cryptosystème entièrement homomorphe et cette solution nécessite une amélioration extrême de la légèreté des algorithmes de cryptographie homomorphe qui ne sont actuellement à la portée que de quelques entreprises comme IBM et Microsoft à cause de leur cout.

Bibliographie

[1] Hassan Syed Ahmed, Safdar Hussain, Bouk Dongkyun Kim, Content-Centric Networks an Overview, Applications and Research Challenges, Kyungpook National University Research Fund, 2015.

[2] Elian Aubry. Protocole de routage pour l'architecture NDN, Réseaux et télécommunications [cs.NI], Université de Lorraine, 2017, Français. ffNNT : 2017LORR0267ff. fftel-01699127.

[3] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulo, X. Vasilakos, K. Katsaros, and G. Polyzos, A Survey of Information-Centric Networking Research, IEEE Communications Surveys & Tutorials, vol. 16, no. 2, pp. 1024-1049, https://ieeexplore.ieee.org/document/6563278 , 2014

[4] Conti, Mauro & Gangwal, Ankit & Hassan, Muhammad & Lal, Chhagan & Losiouk, Eleonora, The Road Ahead for Networking: A Survey on ICN-IP Coexistence Solutions, IEEE Communications Surveys & Tutorials. PP. 10.1109/COMST.2020.2994526, 2020.

[5] Tourani, Reza & Mick, Travis & Misra, Satyajayant & Panwar, Gaurav, Security, Privacy, and Access Control in Information-Centric Networking: A Survey, IEEE Communications Surveys & Tutorials (in submission, available at arXiv). PP. 10.1109/COMST.2017.2749508, 2016.

[6] Fabian Ohlman, 'Content-Centric Networking ', Faculty for Informatics, Technische Universität München, 2013.

[7] Majed, Al-qutwani & Wang, Xingwei & Yi, Bo, Name Lookup in Named Data Networking: A Review, Information. 10. 85. 10.3390/info10030085, 2019.

[8] Wei You, A Content-Centric Networking Node for a Realistic Efficient Implementation and Deployment, Networking and Internet Architecture, Télécom Bretagne, Université de Rennes 1, 2014.

[9] : Yu, Keping & Eum, Suyong & Kurita, Toshihiko & Hua, Qiaozhi & Sato, Takuro & Nakazato, Hidenori & Asami, Tohru & Kafle, Ved, Information-Centric Networking: Research and Standardization Status. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2938586, 2019.

[10] Dr. DEHIMI Nour El Houda. Architecture orientées services. Université Larbi Ben M'hidi – Oum El Bouaghi, 2020.

[11] Naghmeh Niknejad, Waidah Ismail, Imran Ghani, Behzad Nazari, Mahadi Bahari, Ab Razak Bin Che Hussin, Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation.

[12] Sabri Mohamed, These de doctorat, Plate-forme de Data Mining Orientée Services, these de doctorat de l'université des sciences et de la technologie d'Oran, 2018.

[13]Alexandre Beaupré, étude des architecture orientées service, université de Sherbrook, 2012.

[14] Teo, L. K. Y., Teh, D. W., Corbitt, B.; Service Oriented Architecture (SOA), Implications for Australian university information systems curriculum, Pacific-Asia Conférence on Information Systems Proceedings, 2010.

[15] OASIS, Reference Model for Service Oriented Architecture 1.0, https://oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf (consulté Juillet 2021)

[16] SOA-Service composition, www.tutorialspoint.com/soa/soa_service_composition.html (consulté Aout 2021).

[17] Georgiadis, Christos and Fouliras, Panayotis and Mavridis, Ioannis and Manitsaris, Athanasios, Web services and multimedia in m-business applications: Opportunities and concerns, in IJWIS, vol. 2, pp. 51-62, February 2006.

[18] What is SOA Testing? Tutorial with Example, https://www.guru99.com/learn-soa-testing.html (consulté Juillet 2021)

[19] Endrei, M. & Ang, J. and Arsanjani, Ali & Chua, S. & Comte, Philippe & Krogdahl, P. & Luo, Min & Newling, T., IBM: Patterns: service-oriented architecture and web services, in IBM Redbook, p. 345, January 2004.

[20] World wide web consortium, SOAP Version 1.2, https://www.w3.org/2002/07/soap-translation/soap12-part0.html (consulté Juillet 2021)

[21]Oracle, Structure of a WSDL document, https://download.oracle.com/otn_hosted_doc/jdeveloper/1012/web_services/ws_wsdlstructure.html (consulté Juillet 2021)

[22] Boudaa, Boudjemaa, Les communautés pour une haute disponibilité des services Web maintenant la médiation sémantique dans les compositions, 2009.

[23] Oracle, Working with SOAP messages, docs.oracle.com/cd/E19340-01/820-6767/6ni2reild/index.html (consulté Aout 2021)

[24] G. Dugué, Modélisation d'une architecture orientée service et basée composant pour une couche de Transport autonome, dynamique et hautement configurable, Réseaux et télécommunications [cs.NI], Institut National des Sciences Appliquées de Toulouse (INSA Toulouse), 2014.

[25] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner, and M. Varvello, Service-centric networking, *IEEE International Conference on Communications Workshops (ICC)*, June 2011.

[26] S. Shanbhag, N. Schwa, I. Rimac, and M. Varvello, SoCCeR: Services over Content-Centric Routing," *ACM Workshop on Information-Centric Networking (ICN)*, August 2011.

[27] S. Srinivasan, A. Singh, D. Batni, J. Lee, H. Schulzrinne, V. Hilt, G. Kunzmann, Ccnx-serv: Dynamic service scalability in information centric networks, in Communications (ICC), 2012 IEEE International Conference on, pp. 2617–2622, 2012.

[28] Wood, Christopher Alphonse, Security and Privacy Challenges in Content-Centric Networks, p. 338, 2017.

[29] A. Shamir, How to share a secret, Communications of the ACM, 22(11):612{613, 1979.

[30] S. Misra, R. Tourani, & N. E. Majd, Secure content delivery in information-centric networks: Design, implementation, and analyses, In Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking, pages 73{78. ACM, 2013.

[31] R.L. Rivest, L. Adleman, M.L. Dertouzos, on data banks and privacy Homomorphisms, 1978.

[32] C. Gentry, A fully homomorphic encryption scheme, phd, 2009.

[33] Donald Nokam Kuate, Cryptographie homomorphe et transcodage d'image/video dans le domaine chiffré, Cryptographie et sécurité [cs.CR], Université Paris Saclay (COmUE), 2018. Français. NNT : 2018SACLS575. tel-02150082

[34] X. Zhang, K. Chang, H. Xiong, Y. Wen, G. Shi, and G. Wang. Towards name-based trust and security for content-centric network. In Network Protocols (ICNP), 2011 19th IEEE International Conference on, pages 1{6. IEEE, 2011.

[35] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of CRYPTO*, 1985.

[36] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proce. of CRYPTO*, 2001.

[37] RISE. Scalable and adaptive internet solutions, https://www.sics.se/projects/ scalable-and-adaptive-internet-solutions.

[38] Tahir, Ari. (2015), Design and Implementation of RSA Algorithm using FPGA, International Journal of Computers & Technology, Vol 14. 6361-6367. 10.24297/ijct.v14i12.1737.

[39] L'Information-Centric Networking : l'infrastructure de demain? , https://gblogs.cisco.com/fr/innovation/symposium-2017-3-linformation-centric-networking-linfrastructure-de-demain/ (Consulté Septembre 2021)

[40] What is OMNeT ?, https://omnetpp.org/intro/ (consulté Septembre 2021)

[41] Ccn-lite on Github, https://github.com/cn-uofbasel/ccn-lite (consulté Septembre 2021)