الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire
Abdelhafid Boussouf Mila

Institut des Sciences et Technologie
Département de Mathématiques et Informatique

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique

Spécialité: Sciences et Technologies de l'Information et de la Communication (STIC)

Thème

(Route-Translator)
Une solution pour le déploiement du routage IP dans les réseaux SDN

Préparé par : - **MAZZI mohammed salah**
- **DERBOUCHE yaaqoub**

Soutenue devant le jury

Président:	**Mr DJAABOUB Salim**	**Grade: MCB**
Examinateur :	**Mme BOUMASSATA Meryem**	**Grade: MAA**
Encadreur:	**Mr BENCHEIKH ELHOCINE madjed**	**Grade: MCB**

Année Universitaire : 2018/2019

DEDICACES

Au Début et avant tous, je veux remercier le dieu qui à permet le courage à faire et finir ce modeste travail.

C'est avec un grand plaisir et une réelle joie de fierté que je dédie ce travail.

À celle qui m'a soutenu durant tout mes années d'étude, qui mérité mon amour éternel pour ses conseils précieuse, sa tâchasse, sa patience à ma mère.

À mon Père HACHEMI qui a fait de moi ce que je suis, j'espère que trouveras dans modeste travail toute la fierté que peut éprouver un père pour son fils.

À tous mes frères AYYOUB, AYMEN ;

À ma sœur IMANE;

À mes oncles et tantes, ainsi que mes cousins et cousins ;

À mes chère amis: MED-SALAH, ADEL, ISMAIL, TAREK, KHALED, FATEH ;

Et à tous les étudiants du STIC 2

À la fin, nous remercions tous ceux qui ont aidé de prés ou de loin à réaliser notre travail.

YAAQOUB

DEDICACES

Au Début et avant tous, je veux remercier le dieu qui à permet le courage à faire et finir ce modeste travail.

C'est avec un grand plaisir et une réelle joie de fierté que je dédie ce travail.

À celle qui m'a soutenu durant tout mes années d'étude, qui mérité mon amour éternel pour ses conseils précieuse, sa tâchasse, sa patience à ma mère .

À mon Père qui a fait de moi ce que je suis, j'espère que trouveras dans modeste travail toute la fierté que peut éprouver un père pour son fils.

À mes trois chéres sœurs ;

À mes oncles et tantes, ainsi que mes cousins et cousins ;

À mes chère amis ;

Et à tous mes colegues du lycée ainsi que les étudiants du STIC 2

À la fin, nous remercions tous ceux qui ont aidé de prés ou de loin à réaliser notre travail.

MOHAMMED SALAH

REMERCIEMENTS

Au terme de ce travail, nous tenons vivement à remercier toutes les personnes qui ont contribué de prés ou de loin à la réalisation de ce présent travail.

En premier lieu, nous exprimons toute nos gratitudes pour notre promoteur, Mr ***BENCHIEKH EL HOCINE MADJED*** *pour ses précieux conseils, sa disponibilité, la confiance qu'il nous 'a toujours témoigné et la sollicitude dont il nous 'a entouré, et ce tout au long de l'élaboration du présent travail.*

Un grand merci à tous nos enseignants du centre universitaire

ABDELHAFID BOUSSOUF - MILA,

pour toutes les connaissances qu'ils nous 'ont inculquées.

Résumé

Le SDN (Software Defined Networking) est un sujet d'actualité qui agite le monde du réseau depuis ces dernières années, la technologie SDN a déclenché un changement radical à long terme dans la conception des réseaux. Pratiquement, le marché a rapidement adopté le SDN comme un ensemble de solutions permettant de supprimer les frontières entre les mondes des applications et du réseau, tout en assurant un déploiement dynamique et aisé des applications, notamment grâce à la virtualisation et au Cloud.

Contrairement aux réseaux informatiques traditionnels, la nouvelle technologie a permis de faciliter la gestion des réseaux en offrant la possibilité de définir les politiques sous forme de programmes de contrôle. Ceci décharge les administrateurs réseaux de la tache de configuration individuelle des équipements afin d'implanter une politique donnée.

Le programme de contrôle s'exécute sur un contrôleur qui est chargé de communiquer les configurations nécessaires aux équipements SDN afin de mettre en œuvre les politiques désirées.

Dans le cadre de ce projet nous avons développé une application permettant d'assister les administrateurs dans la tache de migration des réseaux traditionnels vers les réseaux SDN. La solution proposée assure une migration sure et fiable dans le but de bien profiter des services offerts par la nouvelle technologie SDN.

ملخص

تعد تكنولوجيا الشبكات المبرمجة (SDN) الموضوع الراهن الذي يحرك عالم شبكات الاتصال في السنوات الأخيرة، وقد أحدثت تقنية الشبكات المبرمجة (SDN) تغييرا جذريا في تصميم الشبكات.حاليا تعتبر تكنولوجيا الشبكات المبرمجة من الحلول الفعالة التي تهدف لإزالة الحدود بين عالم التطبيقات وعالم الشبكات مع ضمان التثبيت السهل و الاستغلال الأمثل لهذه التطبيقات.

على عكس شبكات الاتصال التقليدية، سهلت التكنولوجيا الجديدة إدارة الشبكات من خلال توفير إمكانية لتحديد سياساتها في شكل برامج تحكم. هذا يعفي مسؤولي الشبكة من مهمة إدارة كل جهاز على حدة من أجل تنفيذ سياسة معينة.

برامج التحكم تُثبت على مستوى وحدة التحكم المكلفة بترجمة و إيصال الإعدادات اللازمة لأجهزة الشبكة المبرمجة من أجل تحقيق السياسات المطلوبة والخدمات المرجوة.

في إطار هذا المشروع، قمنا بتطوير تطبيق يعمل على مساعدة مسؤولي الشبكة في مهمة الترقية من الشبكات التقليدية إلى الشبكات المبرمجة (SDN).

النهج المقترح في إطار هذا المشروع يضمن لنا ترقية آمنة ومضمونة نحو الشبكات المبرمجة، بغرض الاستفادة من الخدمات التي توفرها هته التكنولوجيا الجديدة.

Abstract

The SDN (Software Define Networking) is the current topic that has been reshaping the network world for the past few years; SDN technology has triggered a radical change over long term in network design. Practically, the market has quickly adopted the SDN as a set of solutions to remove the boundaries between the worlds of applications and the network, while ensuring a dynamic and easy deployment of applications, especially with virtualization and the Cloud Computing.

Unlike traditional networks, the new technology has simplified the network management by offering the possibility of defining the policies in the form of control programs. This relieves network administrators from the task of configuring individually all the devices in order to implement a given policy.

The control program runs on top of a controller who is responsible for communicating the configurations to the SDN devices in order to apply the desired policies.

For this project, we have developed an application that assists the administrators in migrating from traditional networks to SDN networks. The proposed solution ensures a safe and reliable migration to fully take advantage of the services provided by the new technology SDN.

Sommaire

CHAPTRE II : LA MIGRATION VERS LES RESEAUX SDN

CHAPTRE III : CONCEPTION ET REALISATION

Liste des figures

Liste des tableaux

Introduction générale

INTRODUCTION GÉNÉRALE

1. Contexte

Les réseaux informatiques représentent le noyau de communication pour notre vie personnelle ou professionnelle, c'est un système qui permet, à partir de l'interconnexion de différents équipements, de transporter l'information quel que soit son type (Texte, audio ou vidéo). Les réseaux permettent principalement la communication entre des personnes, des processus, des périphériques et même des objets.

L'évolution de l'Internet de nos jours et son expansion vers de nouveaux besoins, caractérisés par la complexité, l'hétérogénéité et la mobilité, ont exposé ses faiblesses architecturales. La gestion d'un tel réseau est devenue un fardeau pour les opérateurs, nécessitant des compétences avancées et une longue expérience.

Récemment, le paradigme du Software Defined Networking (SDN) est devenu un domaine de recherche attrayant, visant à rénover l'architecture des réseaux et à contourner les obstacles existants. Ce nouveau paradigme propose des concepts permettant de subvenir aux besoins récents de l'entreprise ou bien ceux de l'Internet

2. Problématique

Bien que les réseaux SDN présente beaucoup d'avantages par rapport aux réseaux classiques, l'interaction des administrateurs avec l'unité de contrôle SDN reste le maillon faible dans la gestion de ces réseaux, dans le sens ou l'administrateur est chargé d'alimenter manuellement les applications SDN avec les paramètres nécessaires pour le déploiement du réseau. Cette interaction manuelle est toujours sujette à l'erreur.

3. Objectif

L'objectif de notre projet consiste à automatiser le processus de migration des réseaux traditionnels vers les réseaux SDN. La solution proposée assiste les administrateurs afin de garantir une migration fiable avec le minimum d'interaction humaine.

4. Organisation du mémoire

Les chapitres de ce mémoire sont organisés comme suit :

- ✓ Le premier chapitre met l'accent sur les concepts de base de la technologie SDN, permettant aux lecteurs d'avoir une vision générale sur cette nouvelle technologie à travers la présentation de son architecture, ses avantages et ses applications.

- ✓ Le deuxième chapitre présente le principe de la migration des réseaux traditionnels vers les réseaux SDN. Ensuite, il expose, à travers une simulation détaillée, les différentes étapes de l'approche proposée dans le cadre de ce projet, pour une migration fiable du plan d'adressage et du routage à partir d'un réseau traditionnel ver un réseau SDN.

- ✓ En fin, le troisième chapitre est consacré à la conception de la solution ***Route-Translator***, suivi par une présentation de l'environnement de développement ainsi que les outils exploités pour la réalisation de notre application.

Chapitre I

La technologie SDN

Introduction :

L'évolution des appareils mobiles, la virtualisation des serveurs, et l'avènement de services de Cloud sont parmi les tendances qui ont conduit l'industrie des réseaux à réexaminer les architectures de réseau traditionnelles. L'état du réseau change de façon permanente, ce qui inclut un ajustement manuel de la configuration afin de répondre aux changements. Impliquant par-conséquent l'utilisation des scripts pour reconfigurer les équipements quand un changement se produit, y compris ainsi un grand pourcentage d'erreur de configuration.

L'apparition d'un nouveau paradigme réseau SDN (Software Defined Networking), qui introduit des nouvelles méthodes afin d'améliorer différents aspects de gestion et de configuration des réseaux. SDN est un concept qui sépare le plan de données et le plan contrôle, faisant des Switchs des éléments de commutation de paquet dont l'intelligence a été extraite pour devenir un programme placé dans une entité réseau appelé contrôleur.

Dans ce chapitre, nous allons décrire la technologie SDN, en identifiant ses différents modèles. Ensuite, nous allons présenter son architecture, ses avantages et ses domaines d'applications.

1. Définition du SDN

Le SDN (Software Defined Networking) ; En français le réseau défini par logiciel, est un modèle d'architecture réseau qui permet aux administrateurs réseaux de gérer les services de réseaux par abstraction de fonctionnalités .

Dans ce modèle, les équipements réseau se contentent d'implémenter des règles, injectées par les applications, de traitement des flux de données. Une entité intelligente, appelée « contrôleur » voit le réseau dans sa globalité et injecte directement les règles de traitement des données sur chaque équipement du réseau [1].

2. Le but de SDN

Le premier but du SDN est de simplifier l'administration du réseau et à l'instar de ce que la virtualisation a réalisé dans le monde des serveurs, de rendre la consommation des ressources réseaux par les applications plus flexible.

Le SDN permet de rendre les réseaux programmables, par le biais d'un contrôleur centralisé. Aujourd'hui les périphériques réseau prennent leurs propres décisions en interne sur la meilleure façon d'aiguiller le trafic. Ces décisions s'appuient sur les informations distribuées collectées par des protocoles de routage comme OSPF (Open Shortest Path First) [2] et BGP (Border Gateway Protocol) [3] ou des protocoles de vérification des redondances de la topologie comme STP (Spanning Tree Protocol) [4]. Tous les équipements du réseau doivent suivre les règles définies par les standards et permettre ces protocoles de fonctionner ensemble.

Le SDN nous permet d'établir une séparation claire entre le plan de contrôle (qui définit comment un équipement achemine le trafic) et le plan de données (la partie des commutateurs et routeurs qui assure effectivement le transfère des données) [5].

3. Architecture du SDN

Le SDN présente une architecture réseau composée de trois couches (comme l'illustre la **Figure 1.1**) :

- **La couche infrastructure :** Cette couche constituée les commutateurs physiques du réseau.
- **La couche de contrôle :** Représente le logiciel qui agit comme cerveau du SDN. Ce contrôleur réside sur le serveur et gère les règles et le flux de trafic sur le réseau.
- **La couche application :** Permet aux administrateurs de configurer, gérer, sécuriser et optimiser les ressources du réseau via des programmes.

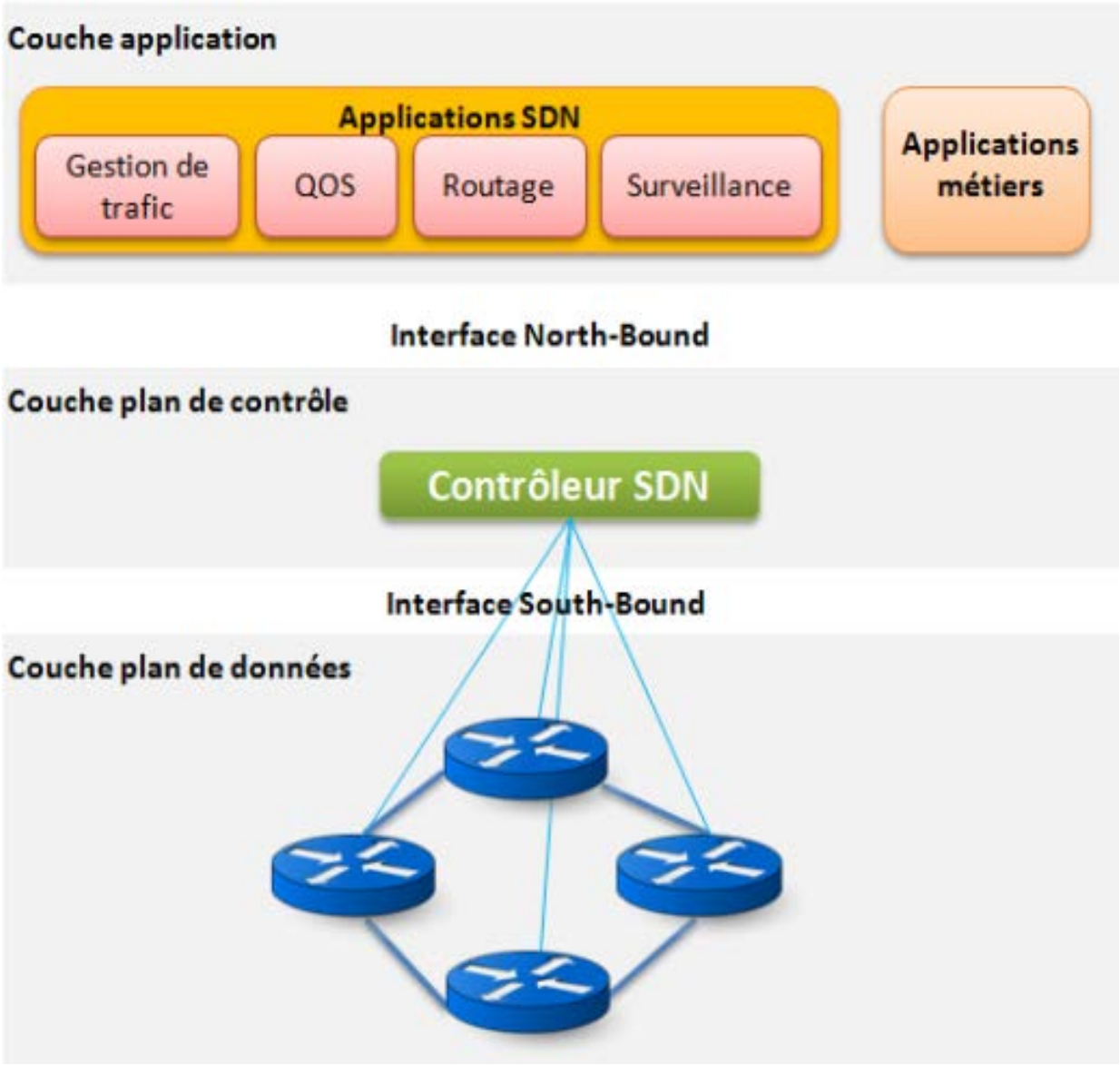

Figure 1.1: Architecture du SDN

4. La logique du SDN

Comme la montre la **Figure 1.2**, les différents plans qui composent les équipements réseaux sont :

4.1. Le plan de données ou (data plane)

C'est la partie qui gère le cœur de notre équipement, son rôle est d'acheminer des paquets depuis un point A vers un point B. Autrement dit "commuter" et/ou "router" en se basant sur des informations telles que la FIB (Forwarding Information Base) ou CAM (Contenent Addressable Memory).

On peut également définir le plan de données comme la gestion du trafic qui n'est pas à destination de l'équipement lui-même.

4.2. Le plan de contrôle ou (control plane)

Ce plan permet de contrôler le plan de données en établissant les règles qu'il devra suivre (Ex. la table de routage). Parmi les protocoles qui participent à se plan, on peut citer : OSPF, BGP, ARP (Address Resolution Protocol), ou STP.

Dans les reseux actuels, chaque équipement opère de façon distribuée chacune de ces fonctions. Le SDN propose donc de créer un point central qui gère le plan de contrôle, tandis que les équipements n'auraient plus qu'à s'occuper du plan de données. Le protocole le plus utilisé pour la communication entre le plan contrôle et le plan de données est OpenFlow .

OpenFlow est un protocole standard pour transmettre des instructions ("entrées des flux " dans le jargon OpenFlow), sont des règles avec un pattern (IP source, IP destination, et les ports) et une action correspondante (rejeter le paquet, transmettre sur port 6633, ajouter un entête VLAN …).

4.3. Le plan de gestion ou (management plane)

Ce plan concerne l'administration de l'équipement. Il s'agit donc des flux provenant des protocoles de gestion tels que: SSH (Secure SHell) [6] ou SNMP (Simple Network Management Protocol). Il est considéré comme un sous-ensemble du plan de contrôle [7].

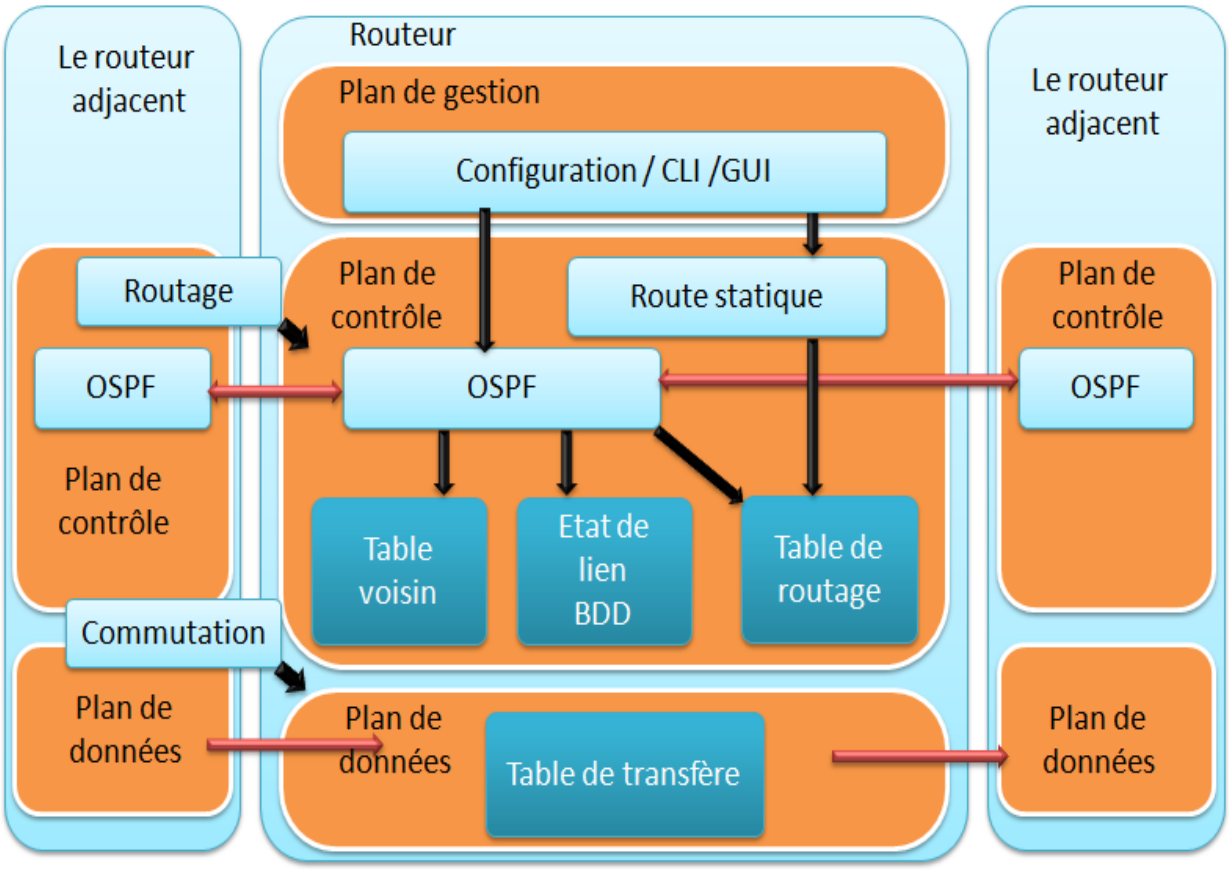

Figure 1.2 : Les plans qui composent les équipements réseaux

5. Le protocole OpenFlow dans l'architecture SDN

OpenFlow c'est le protocole qui lie le plan de contrôle avec le plan de données. L'échange des messages se fait au cours d'une session TCP (Transmission Control Protocol) établie via le port 6633 du serveur contrôleur. Openflow est une composante du SDN, délivré par l'université de Stanford et l'université de Californie à Berkeley, un Switch OpenFlow contient une ou plusieurs tables de flux (flow-table). Lorsqu'un paquet parvient au Switch, les valeurs contenues dans ses en-têtes sont comparées aux différentes règles enregistrées dans la table de flux du Switch [8].

OpenFlow a été initié comme un projet à l'université de Stanford lorsqu'un groupe de chercheurs exploraient la manière de tester de nouveaux protocoles dans le monde IP (créer un réseau expérimental avec le réseau de production) mais sans arrêter le trafic du réseau de production lors des tests.

C'est dans cet environnement que les chercheurs à Stanford ont trouvé un moyen de séparer le trafic de recherche du trafic du réseau de production qui utilise le même réseau IP.

Le résultat de l'équipe de recherche à Stanford a été OpenFlow qui fournit un protocole ouvert dans un but de séparer le trafic de recherche du trafic de production, chacun avec son ensemble de fonctionnalités et caractéristiques de flux [9].

6. Différents modèles pour le SDN

Le SDN englobe toutes les solutions permettant une programmation du réseau, afin de mieux interagir avec les applications. Plusieurs solutions coexistent, adaptées selon les besoins des utilisateurs. On comprend que les solutions permettant de simplifier le déploiement d'une application dans un centre de données ne sont pas les mêmes que celles qui permettront de mieux contrôler l'éclairage d'une ville à travers le réseau. On peut noter différents modèles du SDN :

6.1. Programmabilité individuelle de chaque équipement

Dans ce modèle une application interagit directement avec chaque équipement via des API. L'application communique directement avec les nœuds du réseau pour réaliser des tâches spécifiques.

6.2. Programmabilité via un contrôleur

Dans ce modèle, une application donne un ordre abstrait et global à un contrôleur, qui à son tour traduit cette requête en une suite d'ordres auprès des équipements du réseau concerné. Ce modèle est certainement le plus populaire puisqu'il permet de simplifier le réseau. Le contrôleur masque la complexité du réseau.

6.3. Création d'un réseau virtuel au dessus du réseau physique

Dans ce modèle, les applications créent leur propre réseau « Overlay », s'affranchissant des contraintes du réseau physique sous jacent. Ce dernier n'a pour mission que la simple connectivité entre les nœuds d'extrémité des tunnels, et le réseau d'Overlay assure l'intégralité des services [1].

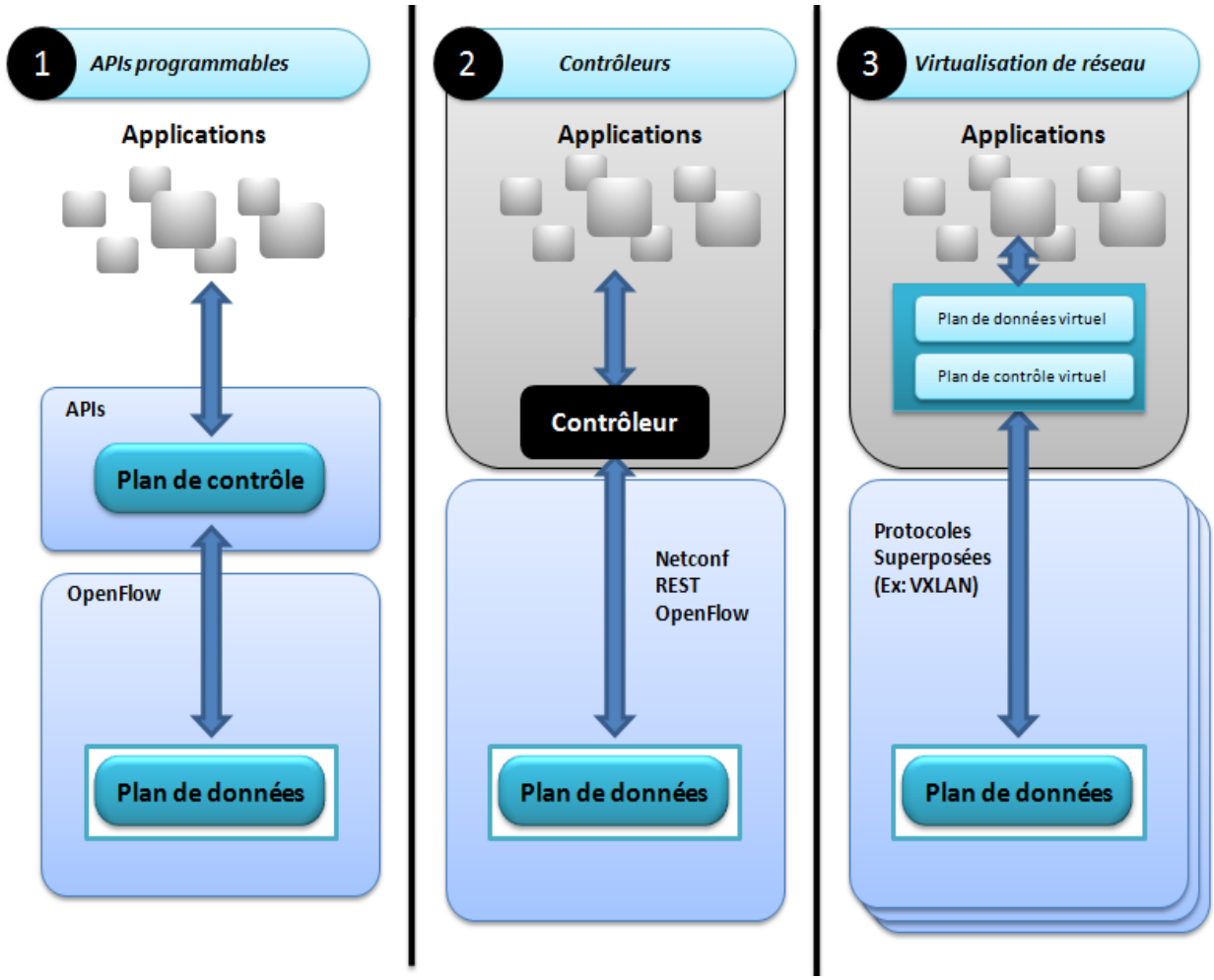

Figure 1.3: Les modèles du SDN [1]

7. Les contrôleurs SDN

7.1. Définition du contrôleur

Un contrôleur est une application dans l'architecture SDN qui gère le contrôle de flux pour améliorer la gestion du réseau et les performances des applications [10].

Le contrôleur SDN permet d'implémenter rapidement un changement sur le réseau en traduisant une demande globale (Ex. prioriser l'application X) en une suite d'opérations sur les équipements réseau [1].

Le contrôleur communique avec les équipements via une ou plusieurs API dites Southbound ou interface sud. Openflow se positionne comme une API sud agissant directement sur le plan de données.

7.2. Le contrôleur SDN et le protocole OpenFlow

Dans les réseaux SDN, l'interface (southbound) est la spécification de protocole OpenFlow, sa fonction principale est de permettre la communication entre le contrôleur SDN et les nœuds du réseau, afin que le routeur définir les flux et exécuter les requêtes qu'il a transférés via des API northbound.

L'interface (northbound) décrit la zone de communication couverte par le protocole entre le contrôleur et les applications ou les programmes de commande de la couche supérieure [11].

7.3. Interfaces de communications

7.3.1. L'interface sud (Southbound)

C'est une interface de communication permettant au contrôleur d'interagir avec les nœuds de la couche d'infrastructure.

Un Switch qui supporte l'OpenFlow contient des tables spéciales appelées tables de flux, ces tables sont similaires à la table MAC sauf qu'elles stockent et gèrent des règles appropriées à chaque flux au lieu des adresses MAC.

7.3.2. L'interface nord (Northbound)

Les interfaces Nord servent à programmer les éléments de la transmission en exploitant l'abstraction du réseau fourni par le plan de contrôle. Elles sont considérées davantage comme des API que comme protocole de programmation et de gestion de réseau.

Il n'existe aucun standard intervenant entre la couche de contrôle et celle d'application du côté d'ONF ou d'autres organisations. Selon l'ONF, plusieurs niveaux d'abstraction et différents cas d'utilisation peuvent être caractérisés, ce qui signifie qu'il peut y avoir plusieurs interfaces Nord pour servir tous les cas d'utilisation. Parmi les propositions des industriels, nous trouvons une API basée sur REST API (REpresentational State Transfer), pour fournir une interface programmable utilisable par les applications du commerce [12].

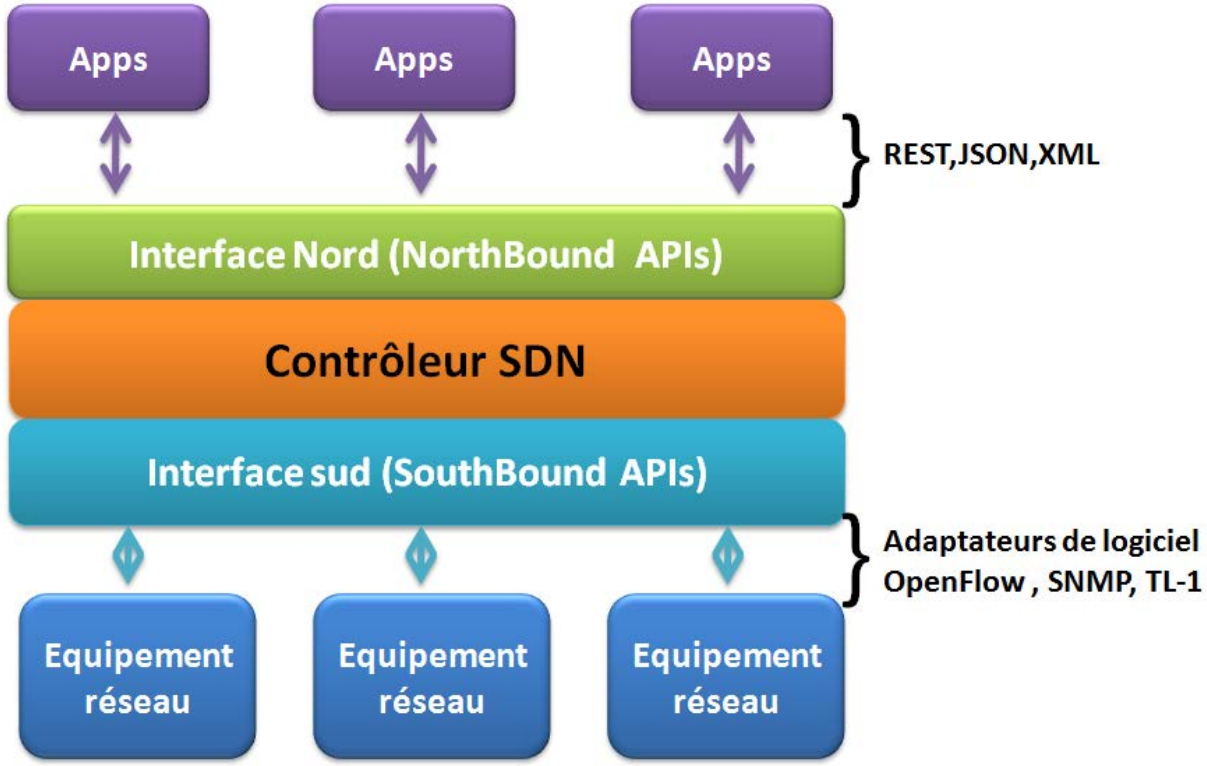

Figure 1.4: Interface-sud (SouthBound)/Interface-nord (NorthBound) du SDN

7.4. Exemples des contrôleurs SDN

Il existe plusieurs contrôleurs SDN, tel que :

7.4.1. NOX

Initialement développé chez Nicira, NOX est le premier contrôleur OpenFlow. Ecrit en C ++ [13].

7.4.2. POX

POX est le plus jeune frère de NOX. C'est un contrôleur de source ouverte (open-source) écrit en Python, et comme NOX, fournit un cadre pour le développement et le test d'un contrôleur OpenFlow, mais les performances POX sont nettement inférieures à celles des autres contrôleurs et ne convient donc pas au déploiement d'entreprise [14].

7.4.3. Beacon

Beacon est un contrôleur Java connu par sa stabilité. Il a été créé en 2010 et est toujours maintenu, il a été utilisé dans plusieurs projets de recherche. En raison de ses performances,

c'est une solution fiable pour l'utilisation dans des conditions réelles. Ce contrôleur a également été utilisé dans d'autres projets tels que Floodlight ou OpenDaylight [15].

7.4.4. Floodlight

Floodlight est un contrôleur OpenFlow basé sur Java, pris en charge par BigSwitch Networks, il est sous licence Apache. C'est un contrôleur qui est facile à configurer. Avec ses fonctionnalités, Floodlight est considéré comme une solution compléte [16].

7.4.5. OpenDaylight

OpenDaylight est un projet de la fondation linux pris en charge par l'industrie. C'est un framework de source ouverte (open-source). Comme Floodlight, il peut également être considéré comme une solution compléte [17].

7 .4.6. Ryu

Ryu est un contrôleur SDN appelé aussi un 'Framework' SDN fournissant des composants qui facilitent la gestion des réseaux et la création des applications de contrôle. Ryu est basé sur Python et supporte la majorité des versions d'OpenFlow (voir la **figure 1.5**) [18].

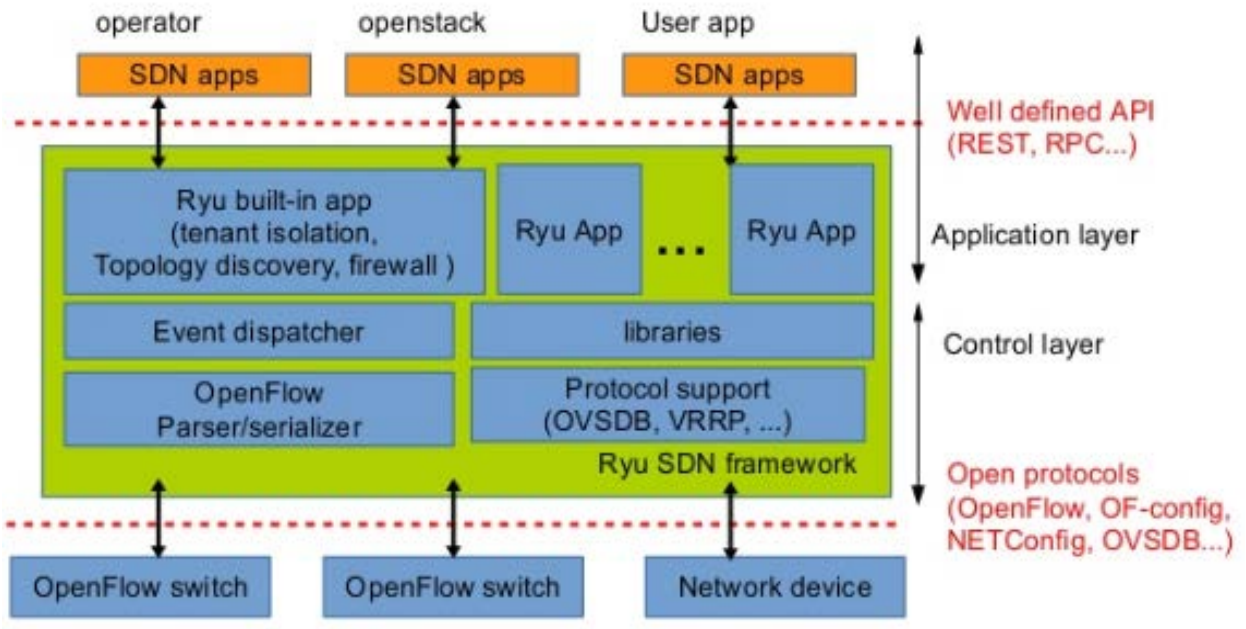

Figure 1.5 : L'architecture du contrôleur Ryu [19]

8. Les outils de simulation du réseau SDN

Plusieurs outils de simulation et d'émulation ont été développés pour implémenter les réseaux basés sur OpenFlow dans une seule machine, et aussi pour tester les nouvelles applications.

8.1. Mininet :

Un émulateur d'environnement SDN, développé par l'université de Stanford, et peut être utilisé pour construire des réseaux virtuels et estimer les métriques de performance pour différentes topologies avec des configurations différentes [20].

8.2. NS3 :

Une autre option est d'utiliser le simulateur NS dans sa version 3 qui supporte la version 0.8.9 du protocole OpenFlow dans son environnement [21].

8.3. EstiNet :

Un simulateur propriétaire supportant la version 1.3.2 du protocole OpenFlow. Il peut simuler des milliers de Switchs. EstiNet consomme moins de ressources [22].

8.4. OMNET++:

Un projet de source ouverte (open-source) dont le développement en C++ a commencé en 1992 par Andras Vargas à l'université de Budapest. En 2013, Omnet++ support le protocole OpenFlow v1.0 [23].

9. La programmation des équipements

La programmation des équipements réseau nécessite sur ces derniers la capacité de recevoir des directives de l'extérieur. Pour cela des interfaces de programmation sont nécessaires : des API (Application Programming Interface).

Il existe plusieurs API, standards ou propriétaires, Il est communément admis qu'une seule API universelle ne suffira pas pour résoudre toutes les problématiques réseau. Aussi les équipements modernes implémentent souvent plusieurs API. On décrit quelques API les plus communément supportées sur les équipements réseau :

9.1. OpenFlow

OpenFlow est une API permettant la programmation du plan de données. Des actions sont programmées au niveau des flux (un ensemble de critères sur les paquets/trames : par exemple IP source, MAC source, etc…) tout le flux correspondant à une entrée dans la table OpenFlow de l'équipement sera traité selon les actions demandées [24].

9.2. Netconf/YANG

Netconf est un standard IETF (Internet Engineering Task Force) (RFC 6241) visant à standardiser les directives réseau auprès des équipements.

YANG est un langage permettant la modélisation des services réseau. Une directive Netconf pourra contenir un modèle YANG et ainsi configurer de la même manière des équipements de constructeurs différents, ce modèle standardisé est le plus souvent privilégié par le fournisseur de service qui cherche au maximum à être indépendants des constructeurs, sans aucun probléme sur le contrôle du réseau [25].

9.3. RESTful API

Une API REST (REpresentational State Transfer) ou (RESTful) est une interface de programmation d'application qui fait appel à des requêtes HTTP (HyperText Transfer Protocol) pour obtenir (GET), placer (PUT), publier (POST) et supprimer (DELETE) des données [26].

9.4. Opflex

Un protocole permettant à un référentiel de règles de dialoguer avec les équipements pour leurs demander de réaliser une action abstraite. Il est adapté à un modèle de programmation déclaratif puisqu'on va se focaliser ici sur un objectif. Opflex intègre une boucle de feedback permettant à un équipement réseau de fournir des informations clés au référentiel de règles afin de l'aider à prendre les bonnes décisions.

Ce protocole est en cours de standardisation à l'IETF. Il est davantage appliqué au niveau du centre de données puisque c'est aujourd'hui sur cet écosystème qu'il a été déployé par divers constructeurs [27].

9.5. Autres API

Il existe autres interfaces de programmation, plus ou moins ouvertes et adaptées à des environnements spécifiques parmi lesquelles: CLI, OVSDB, PCEP, BGP-LS, Python API, SNMP, LISP, CAPWAP, HTTP, OVSDB .etc.

10. Applications SDN

Les réseaux SDN ont une large variété d'applications dans les environnements réseaux, ils simplifient également le développement et le déploiement de nouveaux services et protocoles réseaux. Selon IDC (International Data Corporation), les équipements d'infrastructure, des logiciels de virtualisation, des applications de réseau et de sécurité, sera évalué à 12.5 milliards de dollars en 2020 dans le marché.

10.1. Centres des données et Cloud computing

Le concept de SDN a été également une source d'inspiration pour l'amélioration du rendement des réseaux domicile et des grands centres de données. Ce dernier est gourmand en ressources puisqu'il collecte et traite des volumes très larges des utilisateurs. Pour éviter la consommation de la bande passante, il faut avoir une infrastructure flexible et dimensionnée. Le SDN répond à ces exigences et adapte en temps réel le réseau à l'évolution des besoins en optimisant la centralisation des données, accélérant le traitement et simplifiant la gestion des commutateurs.

10.2. Multimédia et QOS

Les applications multimédias comme le streaming vidéo, la vidéo à la demande, la vidéoconférence, la WebTV, etc. nécessitent des ressources réseau stables et tolèrent les erreurs et les retards de transmission. En se basant sur la vue centralisée du réseau offerte par SDN, on peut sélectionner selon le débit, des chemins différents pour les divers flux de trafic. Les développeurs fournissent le VSDN (Vidéo over Software-Defined Networking), une architecture de réseau qui détermine la trajectoire optimale en utilisant une vue d'ensemble du réseau.

10.3. Sécurité

Grâce à l'architecture centralisée, le contrôleur peut détecter les attaques rapidement et limiter leurs effets. Plusieurs chercheurs ont présenté des mécanismes pour détecter les attaques par

déni de service (Dos et DDos). Par exemple, les développeurs proposent d'isoler le périphérique réseau des attaques intra-LAN à l'aide d'OpenFlow. Dans l'architecture suggérée, le composant IDS (Intrusion Detection System) est chargé de détecter les attaques en reconnaissant les équipements non sécurisés, et d'informer le contrôleur. Les résultats ont montré que cette architecture est assez efficace sur un réseau LAN réel. Un nouveau plan de sécurité a été proposé, leur but est d'inspecter tous les échanges de flux entre les switchs et le contrôleur pour prévenir les attaques de type (IP spoofing). Les résultats de simulation montrent que l'introduction de ce plan permet de renforcer, en temps réel, les différents niveaux de sécurité avec un minimum de configuration tout en gardant une surcharge très faible sur le contrôleur, puisque tout le traitement est placé en dehors du contrôleur. Cela a permis de bloquer rapidement les attaques au niveau du contrôleur et des switchs et de tracer leur source.

10.4. Réseaux sans fil

Le SDWN (Software Defined Wireless Network) a été proposé comme solution d'intégration du SDN pour les réseaux sans fil, pour faciliter la gestion et supporter le déploiement de nouvelles applications dans l'infrastructure réseau. De nouvelles architectures basées sur la séparation du plan de contrôle et de données ont été proposées, notamment le CSDN (CellularSDN) et le SoftAir. Ces architectures collectent les données des utilisateurs et les conditions du réseau, et les font remonter au contrôleur afin d'optimiser la consommation d'énergie, l'utilisation des ressources et la personnalisation des services aux utilisateurs [28].

11. Avantages du SDN

11.1. Réseaux programmables

Avec SDN, il est plus simple de modifier les stratégies réseau car il suffit de changer une politique de haut niveau et non de multiples règles dans divers équipements de réseau.

De plus, la centralisation de la logique avec des connaissances globales et une grande puissance de calcul, simplifient le développement des fonctions plus sophistiquées. Cette aptitude à programmer le réseau est l'élément clé du SDN.

11.2. Flexibilité

SDN apporte également une grande flexibilité dans la gestion du réseau. Il devient facile de rediriger le trafic, d'inspecter des flux particuliers, de tester de nouvelles stratégies ou de découvrir des flux inattendus.

11.3. Réduire les couts d'exploitation

L'efficacité administrative, amélioration de l'utilisation du serveur, le meilleur contrôle de la virtualisation, et d'autres avantages devraient produire en des économies d'exploitation.

Le SDN devrait réduire les couts d'exploitation globaux et générer des économies administratives, car plusieurs problèmes d'administration du réseau courants peuvent être centralisés et automatisés.

11.4. Routage

SDN peut également être utilisée pour gérer les informations de routage de manière centralisée.

11.5. Gestion du Cloud

SDN permet également une gestion simple d'une plateforme Cloud. En effet, la dynamique apportée par SDN traite des problèmes spécifiques aux Clouds tels que l'évolutivité, l'adaptation, ou des mouvements de machines virtuelles.

11.6. Simplification matérielle

SDN a tendance d'utiliser des technologies standard et de base pour contrôler les équipements du réseau, tandis que la puissance de calcul n'est requise qu'au niveau du contrôleur. Ainsi, les équipements de réseau deviendront des produits à bas prix offrant des interfaces standard. Avec ce type de matériel, il serait également simple d'ajouter de nouveaux périphériques, puisqu'ils ne sont pas spécialisés, de les connecter au réseau et de laisser le contrôleur les gérer conformément à la politique définie. Ainsi, le réseau devient facilement évolutif [29].

Conclusion

Dans ce chapitre nous avons fourni une base théorique sur la nouvelle technologie SDN qui élimine la nature complexe et statique des architectures de réseau distribuées traditionnelles, nous avons présenté ses déférents modèles, son architecture et ses avantages ce qui nous donne une bonne compréhension de cette nouvelle technologie. Dans ce qui suit nous nous intéressant à présenter les étapes à suivre afin de mettre cette technologie en réalité.

Chapitre II

La migration vers les réseaux SDN

Introduction :

Le réseau SDN a été bien accepté par l'industrie des réseaux comme un moyen pour optimiser les réseaux d'entreprise, de centre de données, des fournisseurs de services et de campus. L'objectif de cette transformation SDN est de permettre la simplification du réseau tout en assurant un déploiement rapide des nouveaux services différenciés. La programmabilité et l'ouverture constituent les caractéristiques clés des réseaux SDN.

Dans ce chapitre, nous allons présenter d'une façon générale le principe de la migration vers les réseaux SDN, ensuite nous allons détailler l'approche proposée, dans le cadre de ce projet, pour une migration d'un réseau traditionnel vers un réseau SDN.

1. Les considérations de la transition vers Les réseaux SDN

Bien que le déploiement du SDN dans un nouveau centre de données soit relativement simple, la plupart des opérateurs n'ont pas le luxe d'avoir un nouvel environnement. Par conséquent, la planification de la migration est essentielle pour ouvrir la voie au SDN. Un certain nombre de défis doivent être affrontés tout au long du processus, notamment les couts, les performances, la disponibilité des services, la gestion et la sécurité. La résolution des problèmes de sécurité figure parmi les priorités les plus importantes des opérateurs de réseau. La fondation des réseaux ouverts reconnait ces préoccupations et a mis en place un groupe de discussion sur la sécurité qui se concentre sur les considérations de sécurité du réseau SDN, qui sont également abordées par plusieurs autres groupes de travail de la fondation. Pour commencer la transition vers le SDN, certaines questions peuvent naturellement se poser, notamment:

- Quels sont les buts de la transition vers le SDN ?
- Quelles sont les étapes de la transition vers le SDN ?

La transition vers SDN peut être une tâche difficile. Cependant, plus on consacre de temps à répondre aux questions ci-dessus, plus les préoccupations fondamentales sont atténuées [30].

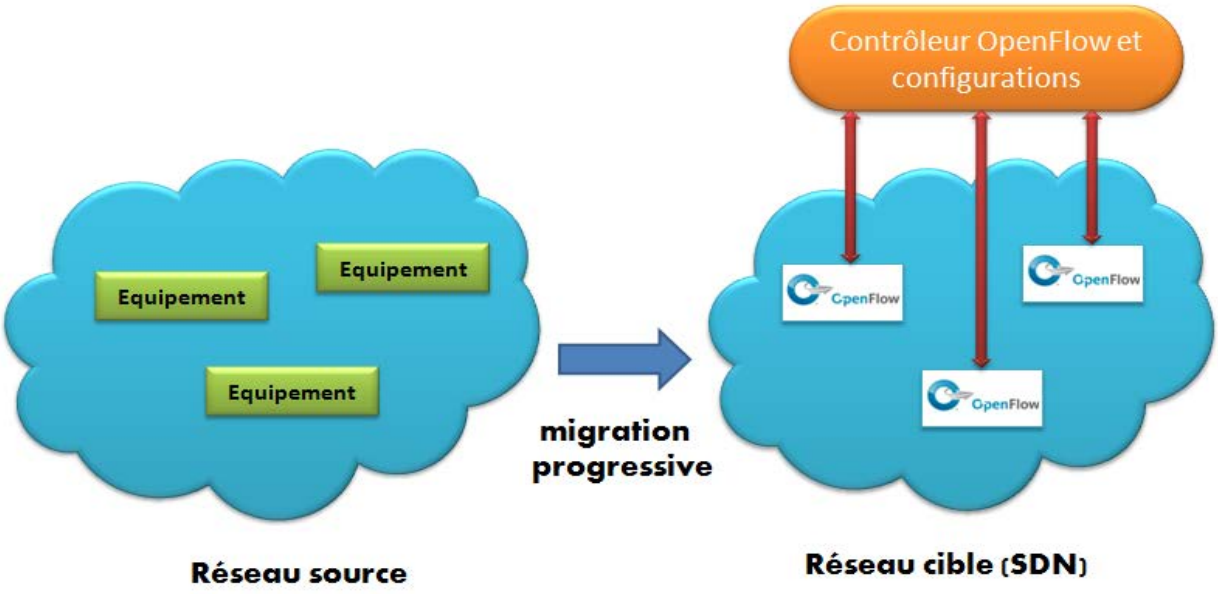

Figure 2.1: **La migration du réseau traditionnel vers les réseaux SDN**

2. Les cas d'utilisation de la transition vers les réseaux SDN

Les cas d'utilisation de la transition vers SDN se répartissent en trois catégories principales : d'un réseau traditionnel vers un réseau SDN pur, d'un réseau traditionnel vers un réseau mixte et d'un réseau traditionnel vers un réseau hybride (voir la **Figure 2.2**).

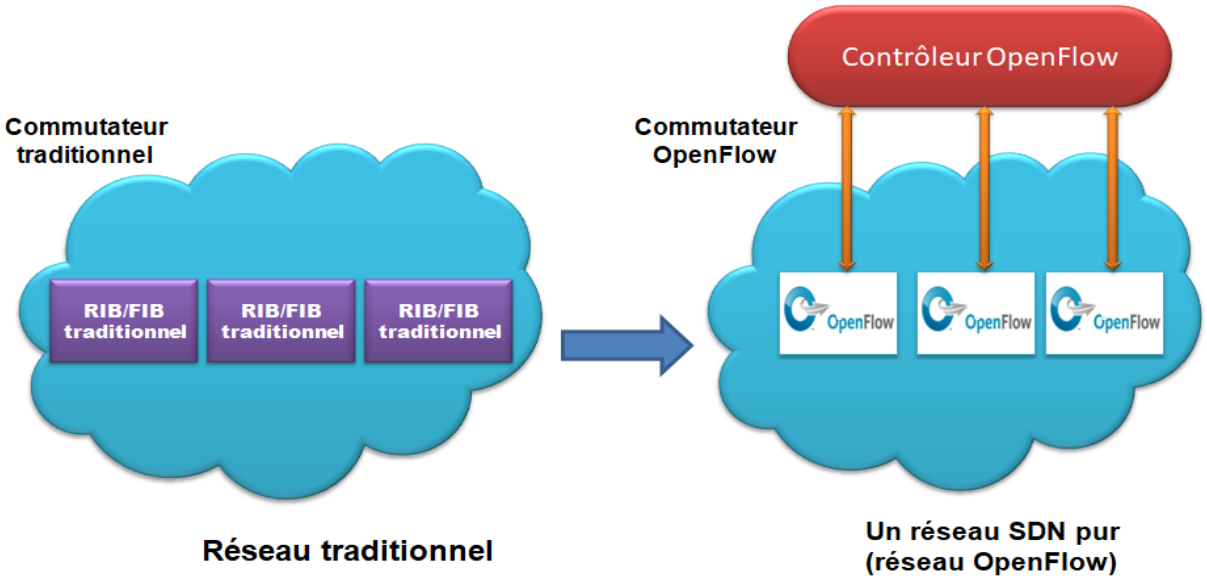

Figure 2.2.A: **La transition vers un réseau SDN pur**

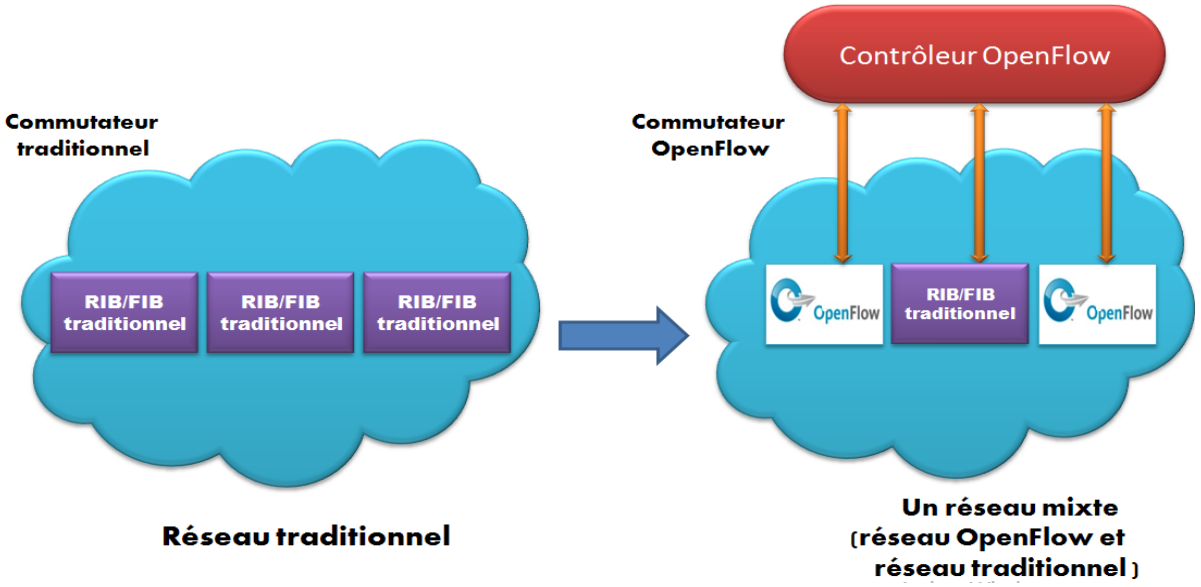

Figure 2.2.B: La transition vers un réseau SDN mixte

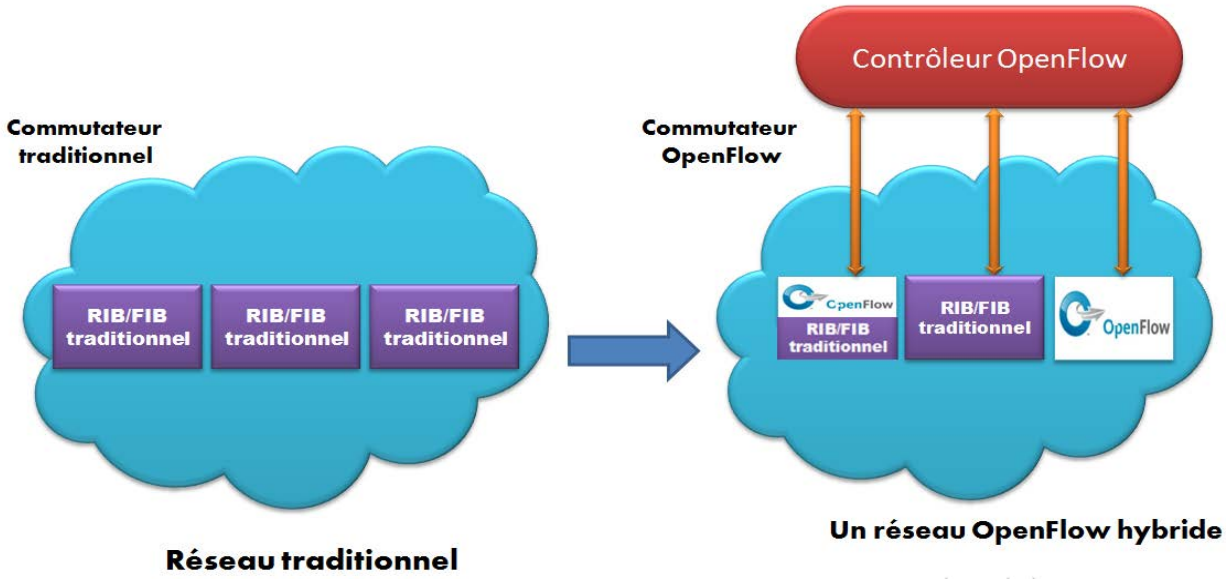

Figure 2.2.C: La transition vers un réseau SDN hybride [30]

Les scénarios de transition vers un réseau SDN pur sont les moins complexes car il n'est pas nécessaire de prendre en charge l'intégration ou l'interopérabilité avec une infrastructure réseau existante non basée sur Open-Flow, contrairement aux scénarios de transition vers un réseau mixte et hybride.

Avec le système de transition d'un réseau traditionnel vers un mixte, les nœuds Open-Flow sont déployés et coexistent avec les commutateurs, routeurs traditionnels et s'interfacent avec les plans de contrôle traditionnel. Les contrôleurs Open-Flow et les périphériques

traditionnels doivent échanger des informations de routage via le plan de contrôle traditionnel. Avec un déploiement de réseau hybride, les périphériques hybrides interfacent avec les contrôleurs Open-Flow et le plan de contrôle traditionnel.

Dans le reste de cette section, nous présentons brièvement deux fameux projets de migration vers le SDN, établis respectivement par l'université de Stanford et par Google.

La conception et le déploiement de ces deux scénarios (transition vers un réseau hybride) sont plus difficiles à concevoir comparé à une transition vers un réseau SDN pur [30].

2.1. Transition de l'université de Stanford (traditionnel vers hybride)

L'une des motivations principales de la transition de cette université était de mieux comprendre et de vérifier Open-Flow en tant que nouvelle technologie. L'université a déployé un réseau SDN entièrement fonctionnel utilisant des contrôleurs OpenFlow sur une partie de l'université. Cette transition était initialement axée sur les utilisateurs sans fil, suivie de certains utilisateurs filaires couvrant deux constructions indépendants.

Le déploiement de l'université de Stanford spécifiait une disponibilité du réseau supérieure à 99,9% ainsi qu'un système de sécurité permettant de revenir au réseau traditionnel en cas de pannes majeures. En outre, les performances du nouveau réseau ont été spécifiées pour être les mêmes que les performances du réseau traditionnel, sans aucun impact sur l'expérience d'utilisateur [31].

2.2. La transition de Google (traditionnel vers hybride)

Le réseau WAN Open-Flow de Google est organisé en deux catégories distinctes, l'un transportant le trafic des utilisateurs d'internet et un autre acheminant le trafic interne des centres de données mondiaux de Google. L'ampleur des besoins des utilisateurs et l'ampleur du projet ont fait de la transition vers SDN Open-Flow de Google un cas d'utilisation unique et stimulant, démontrant la flexibilité d'Open-Flow.

L'objectif de la transition de réseau étendu de Google était d'améliorer l'évolutivité, la flexibilité et l'agilité dans la gestion de la structure de réseau étendu internet afin d'optimiser les services Google basés sur les utilisateurs, notamment Google +, Gmail, YouTube, Google maps et autres.

Le réseau interne de Google qui connecte plusieurs centres de données est aujourd'hui un réseau Open-Flow et un cas d'utilisation bien connu du SDN.

Ce réseau inter-centres de données reposait sur une architecture à trois couches: couche matérielle de commutateur, couche de contrôleur de site et couche de contrôle global [32].

3. Approche proposée pour la migration vers les réseaux SDN

Dans le contexte de ce projet, nous proposons le système Route-Translator qui permet aux administrateurs réseaux de migrer progressivement et en toute sécurité leurs réseaux traditionnels vers des réseaux SDN. Notre système se focalise sur la duplication du plan de routage incluant le schéma d'adressage IP (autrement dit les informations de couche trois).

Le processus global est basé principalement sur les trois étapes suivantes:

• Etape 1 : Restaurez automatiquement le plan de routage à partir des réseaux traditionnels. Ceci est obtenu en utilisant les résultats de l'application SpiderNet qui est basé sur le protocole de gestion SNMP (Simple Network Management Protocol). Cette application a été réalisée au niveau du centre universitaire de Mila dans le cadre d'un projet MASTER STIC 2014-2015.

• Etape 2 : Assister l'administrateur pour faire correspondre la topologie source du réseau traditionnel à la topologie cible du réseau SDN.

• Etape 3 : Configurer automatiquement les applications de routage SDN via les API REST exposées, en se basant sur les informations collectées dans la première étape.

Vu que la première étape de cette approche représente un travail réalisé, notre application vise à automatiser la deuxième et la troisième étape tout en exploitant les résultats de la première étape.

Avant de simuler la logique de notre application, il est important de décrire les concepts fondamentaux, liés à la réalisation de la première étape (SpiderNet).

3.1. Système de gestion basée sur le protocole SNMP :

3.1.1. Définition de protocole SNMP

Simple Network Management Protocol est un protocole de gestion réseau traditionnel. Il s'agit d'un protocole de couche application utilisé pour la communication avec des dispositifs de réseau, pour recueillir des informations ou transmettre la configuration. SNMP est un

protocole très utilisé et est mis en œuvre dans la plupart des équipements de réseau disponible (par exemple, commutateurs, routeurs et serveurs…).

3.1.2. L'architecture du protocole SNMP

Un environnement SNMP contient généralement trois éléments principaux :

- **Agent**

Les agents de SNMP sont des programmes installés dans les périphériques de réseau. Un agent SNMP est essentiellement constitué de la pile de protocoles nécessaire pour assurer la l'interaction du NMS (Network Management Station) avec la base de données MIB (Management Information Base).

- **NMS**

Le NMS désigne le périphérique utilisé par l'administrateur pour gérer son réseau. Celui-ci doit obligatoirement posséder :

- Des applications spécifiques à l'administration.
- Une interface avec l'administrateur.

La station NMS peut envoyer des requêtes à un périphérique afin d'obtenir ses informations. L'agent du périphérique reçoit la requête et renvoie les informations demandées. La station NMS peut utiliser les informations de configuration du périphérique afin de déterminer les opérations à entreprendre en fonction de son état.

- **MIB**

Elle est considérée comme un magasin virtuel de l'information de gestion sous forme d'arborescence, c'est la structure qui répertorie les objets gérés. Ces objets gérés pourraient être des paramètres de configuration, de performance [33].

3.2. La découverte automatique de la topologie du réseau traditionnel

La découverte automatique de la topologie réseau est un mécanisme pour obtenir des informations sur les nœuds existants dans le réseau, ainsi que les informations sur l'interconnexion entre ces nœuds. La découverte de la topologie réseau permet d'améliorer la planification, la gestion des ressources et pour obtenir des avantages essentiels dans la gestion

des fautes. La découverte de la topologie peut être effectuée à deux niveaux d'abstractions : physique et logique [33].

3.2.1. Découverte au niveau physique

La topologie physique prend en compte les nœuds physiques actifs du réseau qui interviennent dans le fonctionnement et la transmission d'information au niveau de la couche liaison, tels que les commutateurs et les terminaux.

Dans ce niveau, il existe trois techniques de découverte:

- ***Technique basé sur le protocole CDP et SNMP***

Cisco Discovery Protocol est un protocole propriétaire développé par Cisco ; L'avantage d'utiliser CDP pour découvrir la topologie est que CDP fournit des informations détaillées sur la connectivité des nœuds.

- ***Technique basé sur le protocole LLDP et SNMP***

LLDP (Link Layer Discovery Protocol) est un protocole de découverte standard défini par IEEE 802.1. Comme son nom l'indique, il fonctionne sur la couche liaison.

Ce protocole est très similaire à CDP (Cisco Discovery Protocole), il est basé sur l'échange de trames spécifiques permettant à des équipements réseaux (Switch, routeur) à annoncer et partager leurs informations avec leurs voisins.

Les informations diffusées par ce protocole sont stockée par les destinataires dans le MIB standard, ce qui permet à l'information d'être accessible par un système de gestion de réseau (NMS) utilisant le protocole SNMP.

- ***Technique basé sur le protocole ARP et SNMP***

ARP (Address Resolution Protocol) l'un des protocoles de la suite TCP/IP, a été conçu pour fournir la correspondance entre les adresses logique IP et les adresses physiques MAC.

3.2.2. Découverte au niveau logique

La topologie logique représente la cartographie IP, les nœuds actifs intervenant dans cette topologie sont des équipements capables d'effectuer le routage des paquets IP. Dans ce niveau il existe trois techniques de découverte.

- ***Technique basé sur le protocole SNMP***

C'est une technique passive, la découverte de la topologie des réseaux IPv4 s'appuie sur les informations enregistrées dans les MIBs des nœuds, et plus particulièrement l'objet « ipRouteTable » de MIB-II, il est représenté sous forme d'un tableau bidimensionnel contenant les informations sur l'ensemble de routes installées au niveau du nœud.

- ***Technique basée sur le protocole ICMP***

ICMP est l'un des protocoles de base d'internet. Il est principalement utilisé pour envoyer des messages de contrôle entre des ordinateurs et des routeurs dans le réseau afin de tester l'état d'un nœud. La technique de découverte basée sur ICMP est une technique active, elle est basé sur les deux outils : le Ping et le Traceroute.

- ✓ ***Le Ping :***

Son fonctionnement de base est le suivant : il envoie vers les nœuds de réseau des paquets ICMP Echo-Request, qui demande à la destination de répondre par un ICMP Echo-Reply.

- ✓ ***Traceroute :***

Cet outil utilise la spécification de la durée de vie d'un paquet IP dans le réseau (champ TTL : Time-To-Live) pour découvrir le chemin parcouru entre l'émetteur et le destinataire. Traceroute envoie une séquence des paquets ICMP Echo-Request adressée à chaque nœud, avec l'augmentation progressive de TTL [33].

3.2.3. L'algorithme de découverte de la topologie traditionnel utilisé par SpideNet

L'algorithme utilisé par le système SpiderNet permet de découvrir automatiquement les réseaux au niveau logique. Cet algorithme est entièrement basé sur le protocole SNMP.

Les différentes étapes du processus de découverte utilisé par SpiderNet sont expliquées en détail dans l'algorithme suivant.

Algorithme de découverte utilisé par SpiderNet

1. Auto-découverte des routeurs connectés au segment local.
2. **Si** le résultat de la phase N°1 est nul **Alors** Fin de l'algorithme, **Sinon** Ajouter les adresses IP récupérées dans la phase N°1 à la liste ***Non-Parcourue*s.**
3. Récupération des tables de routage de toutes les adresses IP de la liste ***Non-Parcourue*s**.
4. A partir des tables de routages récupérées dans la phase N°3, Ajouter les Next-Hop des routes directement connectées à la liste ***Visitées.***
5. **Si** la liste ***Non-Parcourue*s** n'est pas vide **Alors** configurer la première adresse de cette liste comme Gateway au niveau du serveur hébergeant **«SpiderNet»**. **Sinon** Fin de l'algorithme.
6. A partir des tables de routages récupérées dans la phase N°3, cibler la table de routage du routeur identifié par l'adresse sélectionnée dans la phase N°5. Ajouter les Next-Hop des routes non-directement connectées qui ne figurent pas dans la liste ***Visitées*** à la liste ***Non-Visitées.***
7. **Si** la liste ***Non-Visitées*** est vide **Alors** Retirer l'adresse IP sélectionnée dans la phase N°5 de la liste ***Non-Parcourue*s**. Aller à la phase N°5. **Sinon** Récupérer la table de routage à partir du routeur identifié par la première adresse IP de la liste ***Non-Visitées.***
8. **Si** la récupération de la table de routage échoue **Alors** Ajouter le couple (adresse sélectionnée dans la phase N°7, adresse sélectionnée dans la phase N°5) à la liste ***Bloquées.*** Retirer l'adresse IP sélectionnée dans la phase N°7 de la liste ***Non-Visitées***. Aller à la phase N°7. **Sinon** Aller à la phase N°9.
9. A partir des tables de routages récupérées dans la phase N°7 :
    - Les Next-Hop des routes directement connectées sont considérées des adresses visitées. Mettre à jour les listes ***Visitées, Non-Visitées, Bloquées.***
    - Pour les Next-Hop des routes non-directement connectées, Ajouter les adresses qui ne figurent pas dans la liste ***Visitées*** à la liste ***Non-Visitées.***

Aller à la phase N°7.

4. Simulation de Route-Translator

Cette section met l'accent sur l'apport de ce projet en expliquant les deux dernières étapes de l'approche proposée pour la migration vers les réseaux SDN :

- Assister l'administrateur pour faire correspondre les deux topologies.
- Configurer automatiquement les applications de routage SDN via les API REST.

4.1. Assister l'administrateur pour faire correspondre les deux topologies

La correspondance des topologies est une tache complexe, elle doit être précédée par la récupération des informations relatives aux topologies des deux réseaux : source (traditionnel) et cible (SDN).

La topologie du réseau source a été récupérée durant la première étape (voir la **Figure 2.3**).

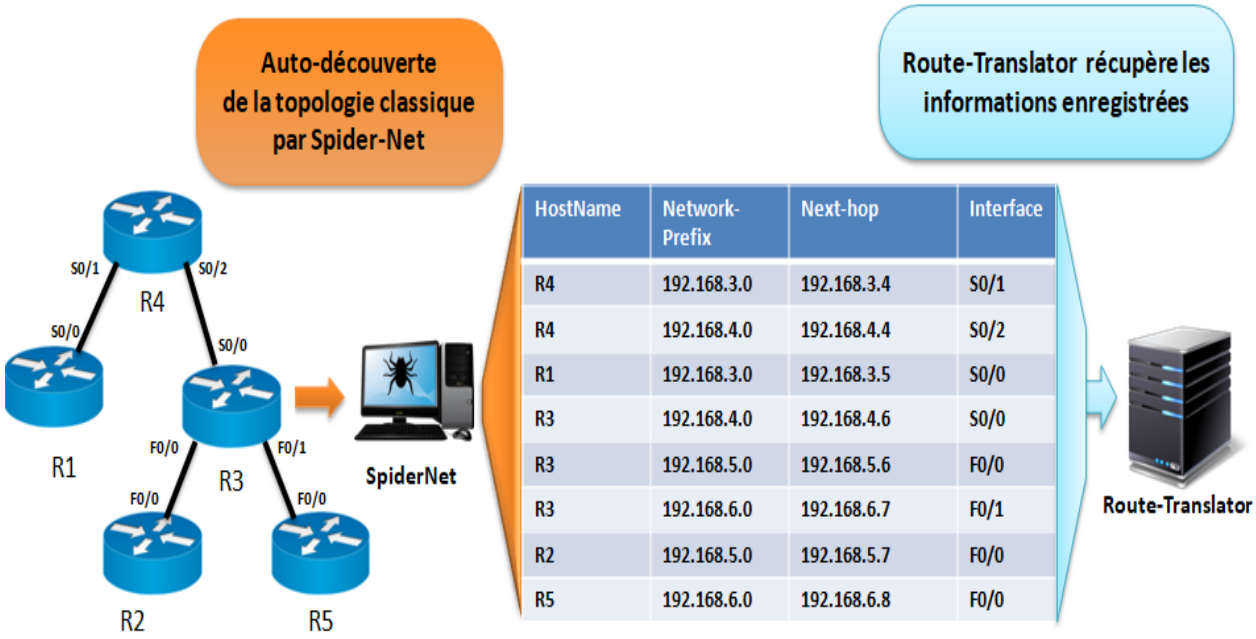

Figure 2.3: La récupération des informations du réseau source

Durant cette étape le système Route-translator doit assurer ce qui suit :

1. Récupérer d'abord les informations relatives au réseau cible **(**voir la **Figure 2.4**).
2. Dérouler un processus permettant d'automatiser (au maximum possible) la correspondance des deux topologies.

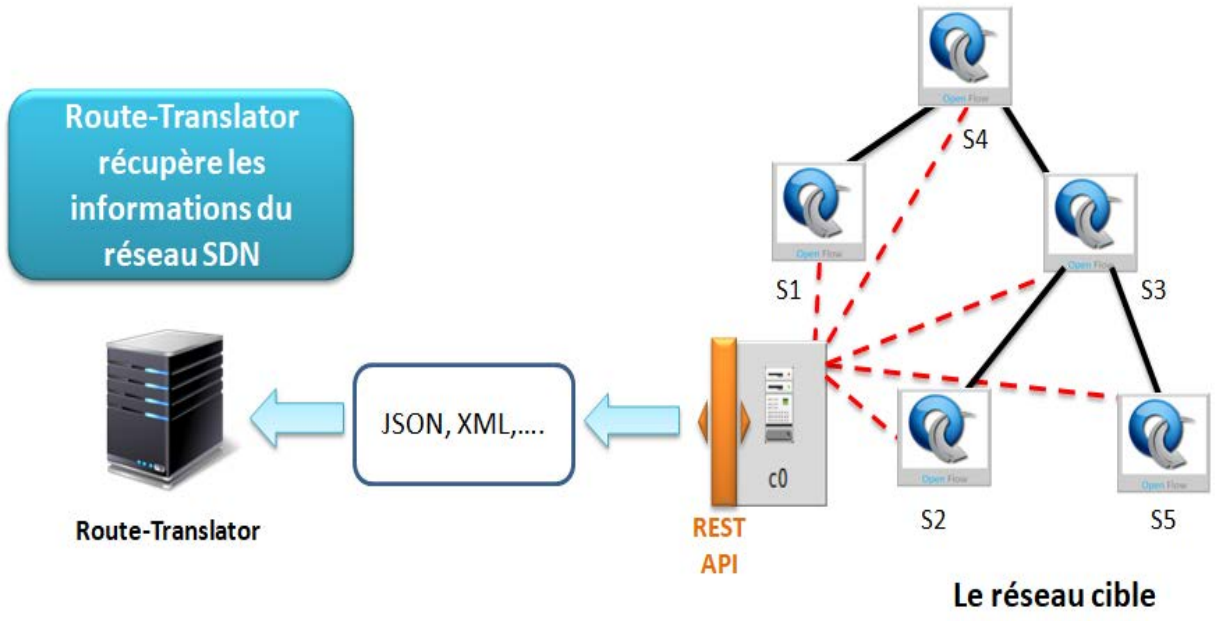

Figure 2.4: La récupération des informations du réseau cible

4.1.1. La découverte automatique de la topologie dans les réseaux SDN

Les contrôleurs OpenFlow ou bien SDN implémentent le protocole de découverte nommé OFDP (OpenFlow Discovery Protocol).

Comme le montre la **Figure 2.5**, le protocole OFDP suit les étapes suivantes afin de découvrir le lien unidirectionnel connectant Switch S1 avec Switch S2 :

1. Le contrôleur envoi au Switch S1 un message de sortie contenant un paquet LLDP. Ce paquet contient des instructions pour que S1 envoie le paquet LLDP via le port P1.
2. En recevant le paquet LLDP via le port P2, le Switch S2 le renvoie au contrôleur dans un message d'entrée.
3. Le contrôleur reçoit le paquet LLDP et confirme l'existence d'un lien unidirectionnel de S1 à S2.

Le même processus est effectué pour découvrir le sens opposé, de S2 vers S1 ainsi que tous les autres liens du réseau [34s].

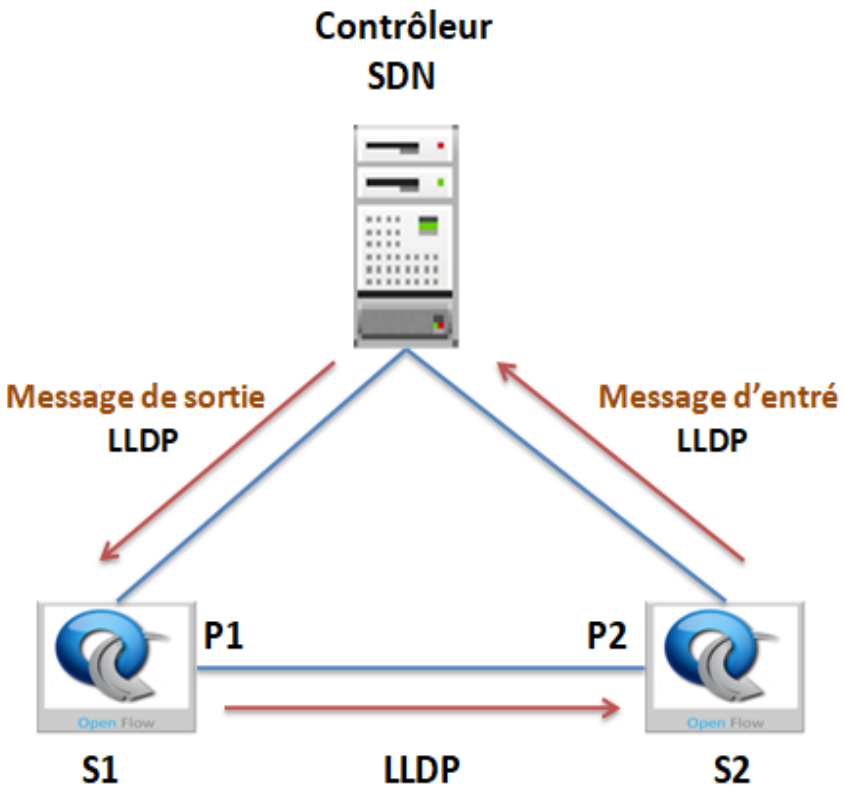

Figure 2.5 : La découverte d'un lien unidirectionnel avec OFDP [34]

4.1.2. La correspondance des topologies source et cible

Au lieu que l'administrateur effectue la correspondre des topologies manuellement, l'application Route-Translator assure une assistance pour la réalisation de cette tache.

Au début, l'administrateur doit faire correspondre juste les premiers nœuds (nœud racine du réseau source et celui du réseau cible). En se basant sur ces nœuds racines le système déroule le processus de correspondance d'une manière automatique pour le reste des nœuds tout en garantissant la fiabilité de cette opération.

Pendant le déroulement du processus de correspondance, si le système n'arrive pas à faire correspondre de façon unique un nœud source à un nœud cible, car il existe plusieurs possibilités, le processus de correspondance s'arrête à cause du conflit. Le cas du conflit exige l'intervention de l'administrateur pour le résoudre.

Dans ce qui suit nous allons décrire pas à pas le fonctionnement du processus de la correspondance des topologies.

4.1.2.1. La correspondance des nœuds racines

L'administrateur choisi un nœud du réseau source comme un nœud de départ et le fait correspondre avec un nœud du réseau cible (voir la **Figure 2.6**), afin de permettre le lancement du processus de la correspondance.

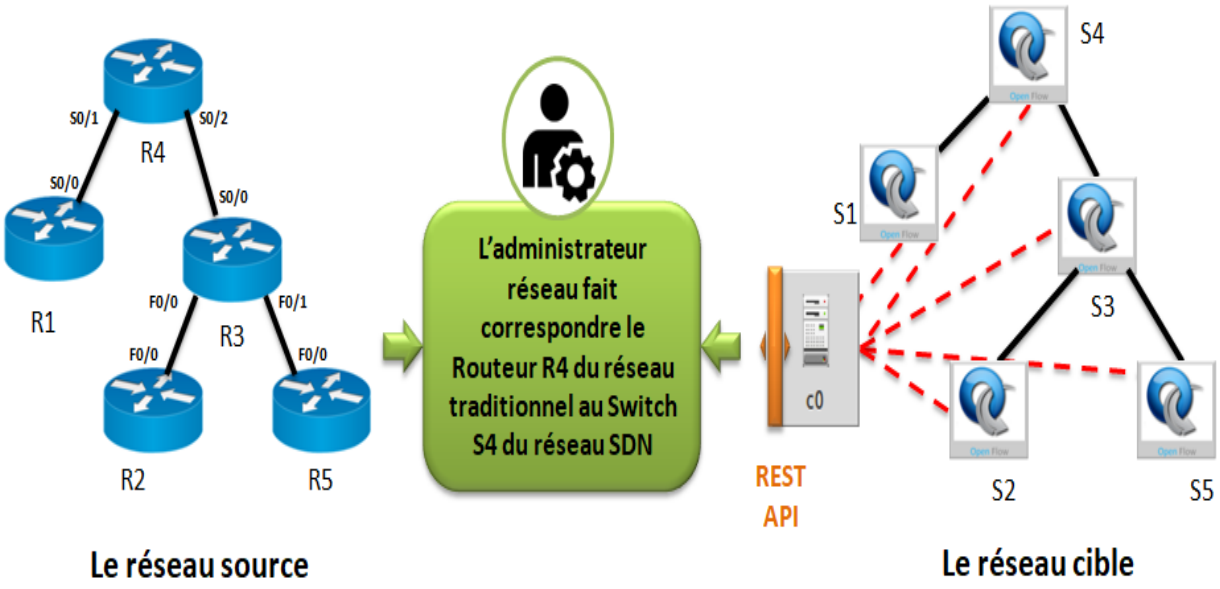

Figure 2.6: La correspondance des nœuds racines

4.1.2.2. Le processus de correspondance des topologies

En se basant sur les nœuds racines, le système commence à faire correspondre automatiquement le reste des nœuds (voir la **Figure 2.7**). Le processus de correspondance est basé essentiellement sur le calcul du nombre des voisins pour chaque nœud dans les deux topologies. Les étapes de déroulement de l'algorithme correspondant (appliqué aux topologies simulées dans la figure 2.7) sont détaillées dans la sous-section **4.1.2.4**.

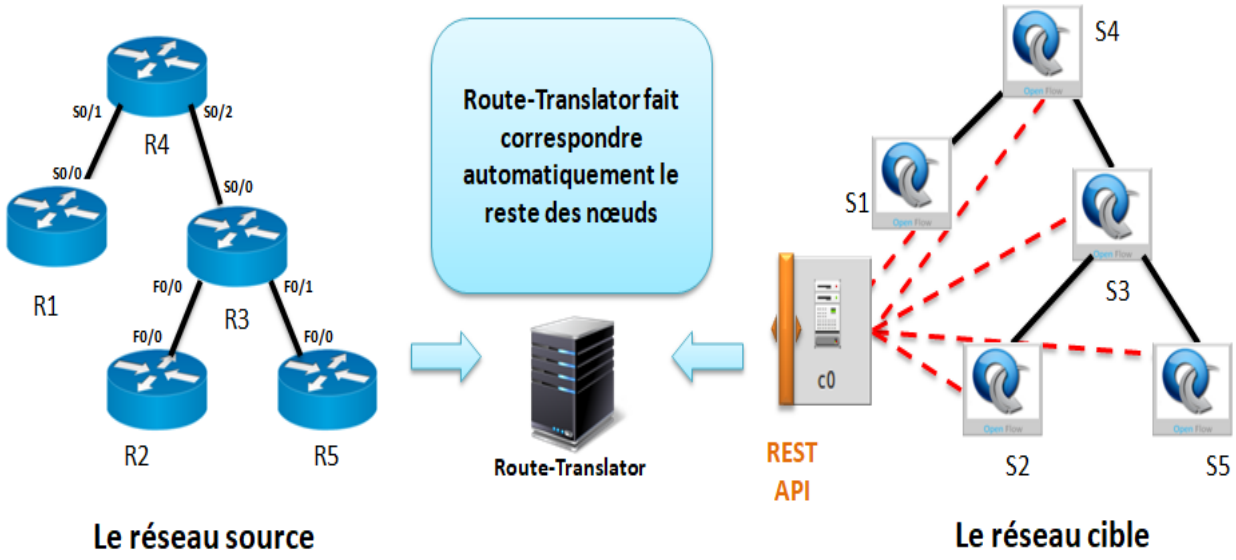

Figure 2.7: La correspondance automatique des topologies

Comme le montre la **Figure 2.8**, en ce qui concerne le processus de correspondance il existe deux types des topologies réseaux :

- Des topologies réseaux sans conflit : Dans le sens ou le processus de correspondance est complètement automatisé pour toute la topologie sans intervention humaine (Voir la **Figure 2.8**.A).
- Des topologies réseaux comportant des conflits : Ceci est montré dans la **Figure 2.8.B** coté gauche, ou le système Route-Translator rencontre un conflit pendant la correspondance des nœuds R2 et R5 avec les nœuds du réseau cible.

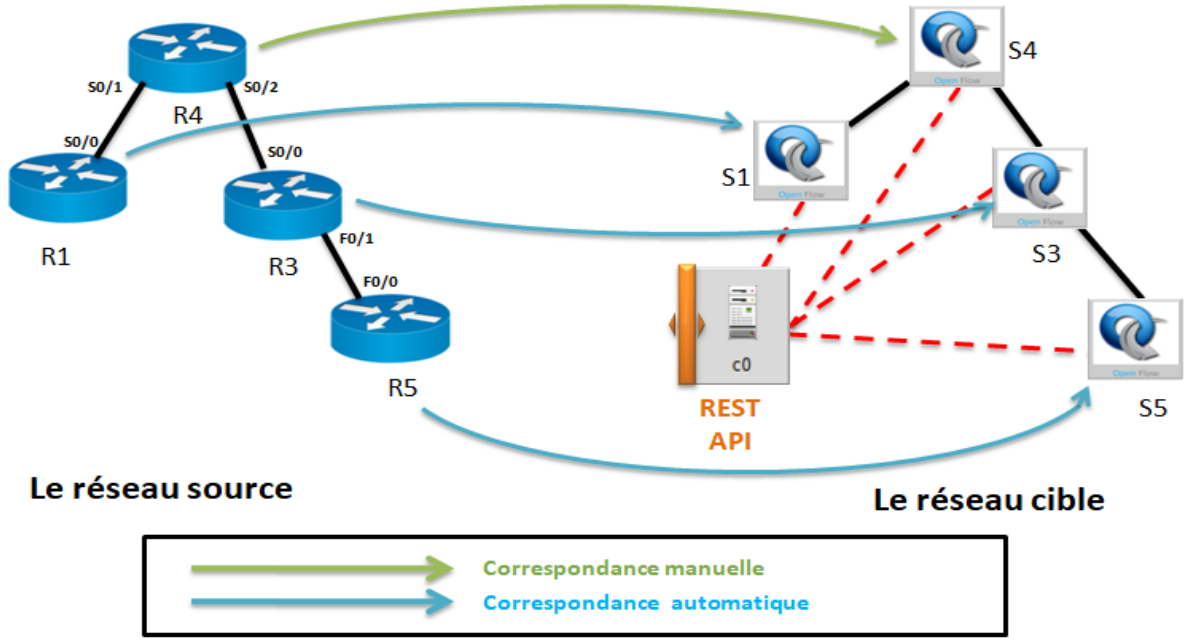

Figure 2.8.A: La correspondance des topologies sans conflits

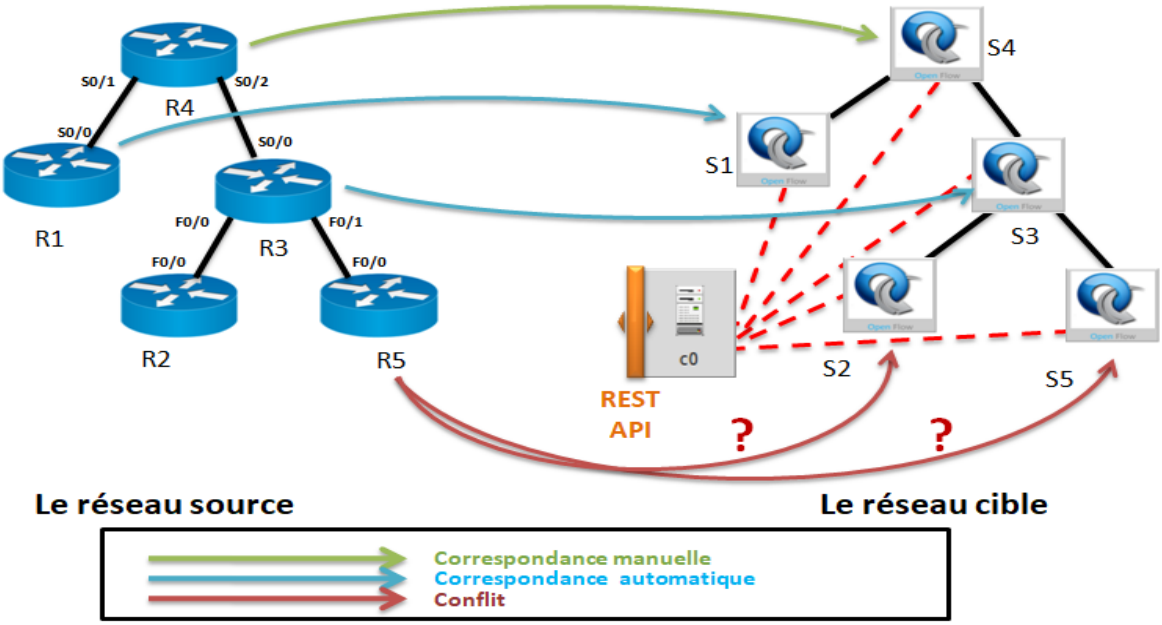

Figure 2.8.B: La correspondance des topologies comportant des conflits

4.1.2.3. La résolution des conflits

Dans le cas de la correspondance des topologies comportant des conflits, l'intervention de l'administrateur réseau est obligatoire. Il doit effectuer une correspondance d'un ou plusieurs nœuds manuellement afin de permettre au système de reprendre le processus automatique pour le reste des nœuds.

Dans la **Figure 2.9**, le fait que l'administrateur correspond manuellement le nœud R2 du réseau source au Switch S2 de réseau cible, permet au système de déduire la correspondance de R5 avec S5.

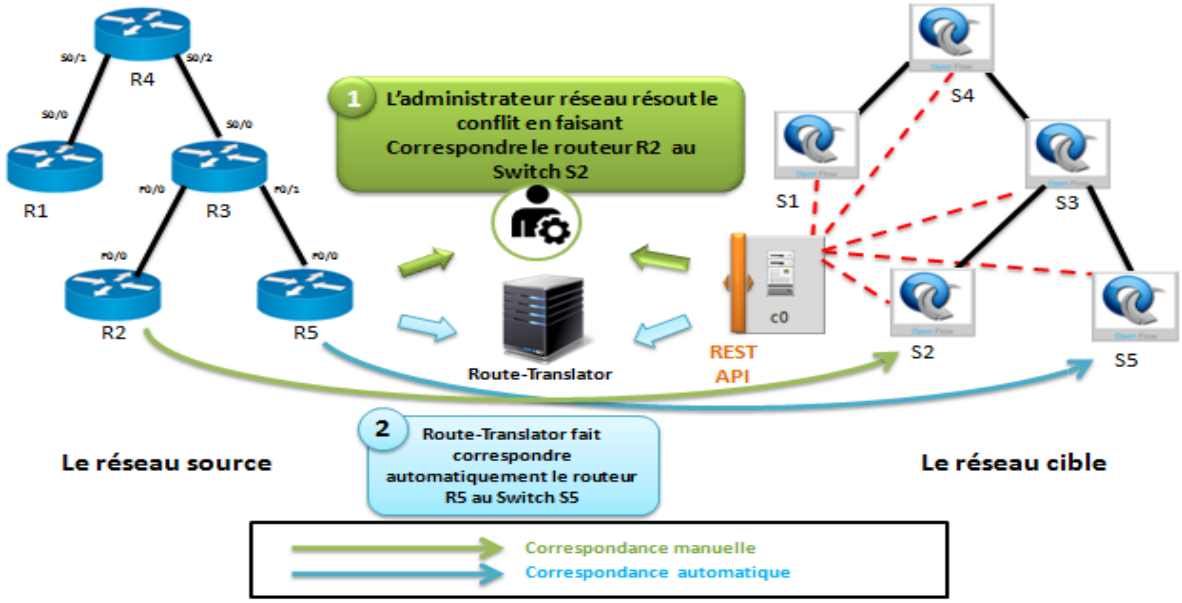

Figure 2.9: La résolution du conflit dans la correspondance des topologies

4.1.2.4. Algorithme appliqué pour la correspondance automatique des topologies

Après la détermination manuelle des deux nœuds racines (source et cible), l'algorithme de correspondance, proposé pour Route-Translator, fonctionne en se basant sur les trois règles suivantes :

1. Un nœud source qui a un nombre unique de voisins par rapport à ses siblings est automatiquement associé au commutateur SDN correspondant (ayant le même nombre de voisin).
2. Le conflit entre les nœuds siblings, ayant le même nombre de voisins, pourrait être résolu en vérifiant, itérativement, le nombre de leurs voisins.
3. Après avoir analysé toutes les branches, les conflits non résolus nécessitent une correspondance manuelle, ce qui permet au système de garantir un processus automatique fiable, respectant le plan conçu par l'administrateur pour les deux topologies.

Afin de simplifier la compréhension de l'algorithme proposé, nous allons décrire, dans le tableau suivant, les différentes étapes du déroulement de cet algorithme appliqué aux deux topologies simulées dans la Figure 2.8. Les mots-clés utilisés sont définis comme suit :

- Les Siblings (nœuds frères): les nœuds du même niveau dans une topologie réseau.
- Le Nbr-Voisins : représente le nombre des nœuds, non traités par le processus de correspondance, qui sont connectés directement au nœud en cous du traitement.

Etapes	Taches		Résultats
Etape N°1	• Faire correspondre manuellement le routeur R4 au Switch S4.		
	• Lister les nœuds voisins du R4.		**R1, R3**
	• Lister les nœuds voisins du S4.		**S1, S3**
Etape N°2	R1	✓ Identifier le Nbr-Voisins du R1.	**0**
		✓ Lister les nœuds voisins du R1.	/
	R3	✓ Identifier le Nbr-Voisins du R3.	**2**
		✓ Lister les nœuds voisins du R3.	**R2, R5**
	S1	✓ Identifier le Nbr-Voisins du S1.	**0**
		✓ Lister les nœuds voisins du S1.	/
	S3	✓ Identifier le Nbr-Voisins du S3.	**2**
		✓ Lister les nœuds voisins du S3.	**S2, S5**
	• Faire correspondre automatiquement R1 au Switch S1 et R3 au Switch S3.		
Etape N°3	R2	✓ Identifier Nbr-Voisins du R2.	**0**
		✓ Lister les nœuds voisins du R2.	/
	R5	✓ Identifier Nbr-Voisins du R5.	**0**
		✓ Lister les nœuds voisins du R5.	/
	S2	✓ Identifier Nbr-Voisins du S2.	**0**
		✓ Lister les nœuds voisins du S2.	/
	S5	✓ Identifier Nbr-Voisins du S5.	**0**
		✓ Lister les nœuds voisins du S5.	/
	• Faire correspondre manuellement R2 au Switch S2. • Faire correspondre automatiquement R5 au Switch S5.		

Tableau 2.1 : Etapes d'application de l'algorithme auto-correspondance des topologies

4.2. La configuration de l'application de routage SDN

Afin que Route-Translator puisse configurer automatiquement l'application de routage SDN, l'étape de la correspondance des topologies doit être totalement achevée, c'est-à-dire tous les nœuds de la topologie source ont un nœud correspondant dans la topologie cible.

Cette étape consiste à configurer l'application de routage SDN d'une façon complètement automatique sans aucune intervention de l'administrateur réseau, en se basant sur les données collectées dans la première étape ainsi que la correspondance des topologies effectuée dans la deuxième étape. Ceci est réalisé via les API REST exposées par l'application de routage, dans le but de garantir un traitement des paquets, au niveau du réseau SDN, similaire à celui de réseau traditionnel (Voir la **Figure 2.10**).

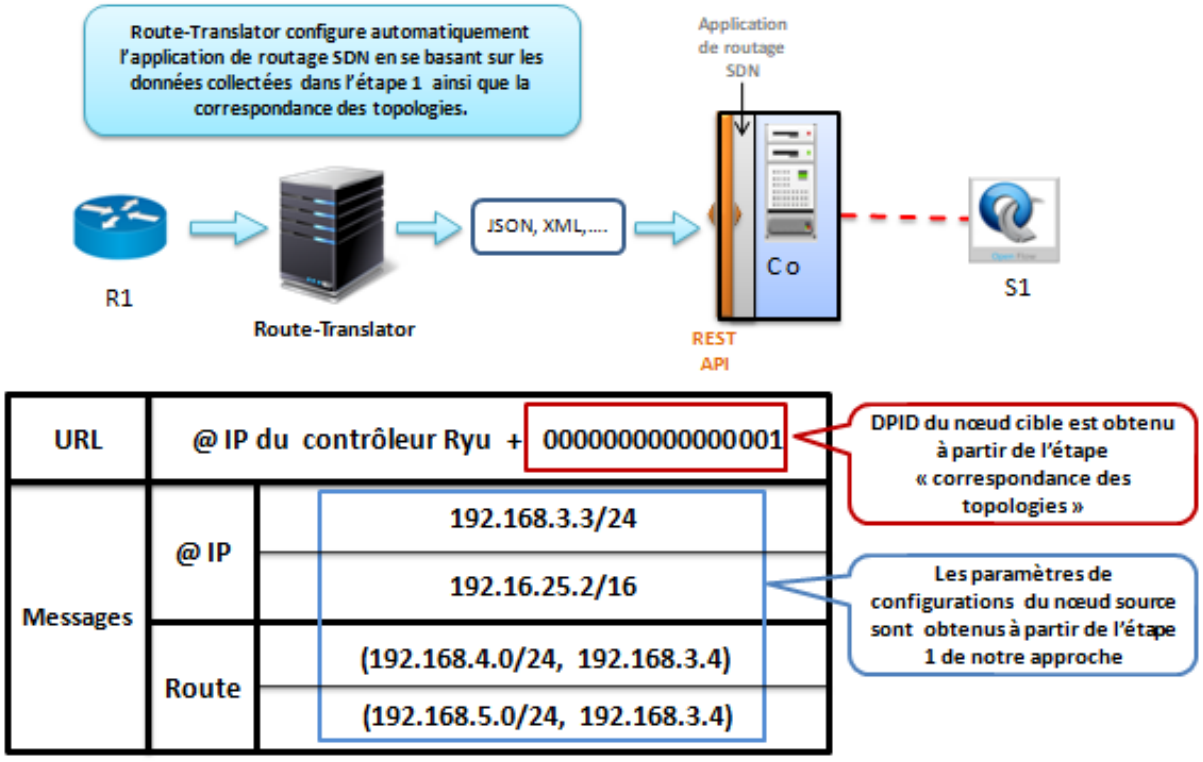

Figure 2.10: La configuration automatique de l'application de routage SDN

Conclusion

Le processus global du passage d'un réseau traditionnel vers un réseau SDN doit se faire d'une manière bien étudiée et bien organisée, ce processus se déroule en trois étapes, la restauration des informations du réseau traditionnel, la correspondance des topologies et en fin la configuration de l'application de routage SDN.

L'approche de migration proposée dans ce chapitre permet d'éviter les problèmes liés à l'erreur humaine et assure une transition fiable vers les réseaux SDN.

Dans le chapitre suivant nous allons détailler la conception de notre système Route-Translator, ensuite nous allons présenter l'environnement de travail qui nous a permis de réaliser ce projet.

Chapitre III

Conception et réalisation

Introduction :

La modélisation d'une application représente une démarche très importante dans le développement de n'importe quel projet logiciel. Cette démarche permet de bien définir l'aspect fonctionnel du système.

Nous présentons dans ce chapitre la modélisation de notre solution ***Route-Translator*** utilisant le langage UML qui s'est imposé comme une norme standard dans la conception orientée objets. Ensuite, nous allons spécifier l'environnement de développement (Langages de programmation, bibliothèques, utilitaires …) qui nous a permis de réaliser ce projet.

1. Conception de Route Translator

Le développement d'une application ou un logiciel nécessite préalablement le passage par un ensemble d'étapes de modélisation. Actuellement il ya plusieurs méthodes de conception, parmi ces méthodes on trouve l'UML.

1.1. Définition UML :

UML (Unified Modeling Language) est un langage de modélisation graphique, basé sur les diagrammes. Il est développé par l'OMG (Object Management Group) dans le but de définir la notation standard pour la modélisation des applications construites à l'aide d'objets.

Les principaux auteurs de la notation UML sont Grady Booch, Ivar Jacobson et Jim Rumbaugh. UML permet une couverture continue de toutes les étapes du processus logiciel **(**voir la **Figure 3.1)** [35]**.**

Figure 3.1 : Logo UML [35]

1.2. Les vues et les diagrammes UML

UML dans sa version 2 propose treize diagrammes, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel. Ces types de diagrammes sont répartis en trois vues classiques **(**voir la **Figure 3.2)**.

- **Vue fonctionnelle :** permet de spécifier les tâches et les services fournis par le système.
- **Vue structurelle (statique) :** permet de visualiser, spécifier, construire et documenter l'aspect statique ou structurel du système informatisé.
- **Vue dynamique :** modélise l'aspect dynamique du système, qui montre les interactions entre le système et ses différents acteurs, ainsi que la façon dont les différents objets contenus dans le système communiquent entre eux [36].

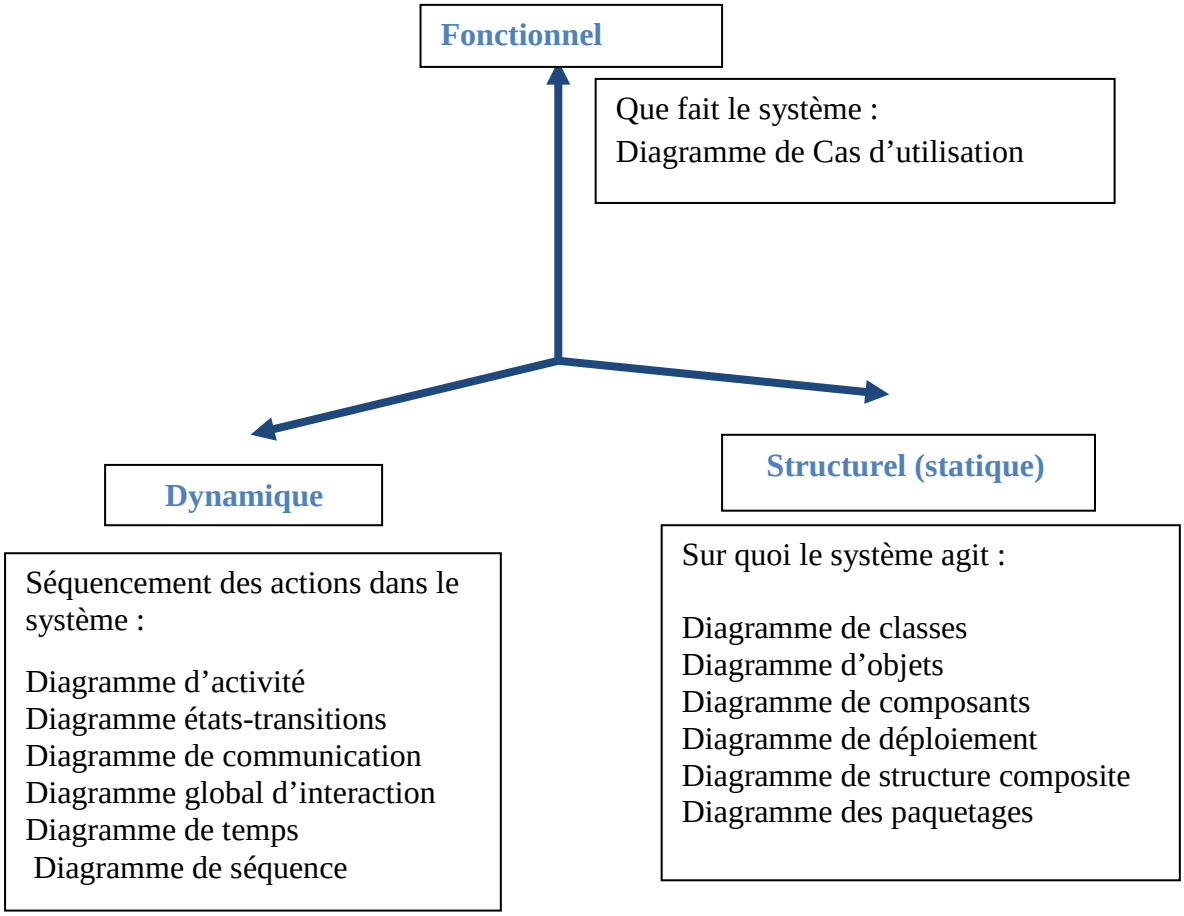

Figure 3.2 : Les vues classiques de modélisation

1.3. Le processus de développement :

Le processus de développement définit une séquence d'étapes, en partie ordonnée, qui concoure à l'obtention d'un système logiciel ou à l'évolution d'un système existant, pour produire des logiciels de qualité.

UML n'est qu'un langage de modélisation, ce n'est pas une méthode. Pour cela, les développeurs d'UML ont créé le processus unifié UP (Unified Process) [37].

Un processus unifié est un processus de développement logiciel basé sur UML, il est itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques.

Dans la modélisation de notre système, nous suivrons une démarche simple. Cette démarche, inspirée du processus UP, répond aux exigences de notre système à travers les trois phases suivantes :

❖ **Identification des besoins :**

- Diagramme de cas d'utilisation

- Diagramme de séquence système

❖ **Phase d'analyse :**

- Diagramme d'activités

❖ **Phase de conception :**

- Diagramme de classes

1.3.1. Identification des besoins

L'objectif de cette phase est de préciser les tâches du système :

➢ Diagramme de cas d'utilisation

A - Définition

Le diagramme de cas d'utilisation est le diagramme qui identifie les grandes fonctionnalités fournies par le système, ainsi que les utilisateurs ou bien les acteurs agissant sur le système. Le diagramme de cas d'utilisation décrit la succession des opérations réalisées par un acteur, les interactions entre les acteurs et les fonctionnalités [38].

B - Les éléments de diagramme de cas d'utilisation

- **Acteur :**

  Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié. La représentation graphique standard de l'acteur en UML est l'icône appelée stick man avec le nom de l'acteur sous le dessin. Il y'a deux types d'acteur :

Un acteur principal :

- ✓ Directement concerné par le cas d'utilisation décrit.
- ✓ Sollicite le système pour obtenir un résultat perceptible.

Un acteur secondaire :

- ✓ Il est sollicité pour des informations complémentaires.
- ✓ Nécessaire au déroulement du cas d'utilisation décrit.

- **Cas d'utilisation**

Un cas d'utilisation représente une fonctionnalité fournie par le système, modélise le service rendu par le système sans imposer le mode de réalisation. Les cas d'utilisation sont représentés par une ellipse contenant leur nom.

Pour détailler la dynamique du cas d'utilisation, la procédure la plus évidente consiste à recenser de façon textuelle toutes les interactions entre les acteurs et le système. Le cas d'utilisation doit avoir un début et une fin clairement identifiés.

- **Les relations :**
  - Relation d'association : C'est un lien de communication entre un acteur et un cas d'utilisation. Elle est représentée par un trait continu.
  - Relation d'inclusion : La relation d'inclusion spécifie qu'un cas d'utilisation est nécessairement une partie d'un autre cas d'utilisation. Elle est représentée par une flèche discontinue stéréotypée « inclusion ».
  - Relation d'extension : La relation d'extension spécifie qu'un cas d'utilisation est éventuellement une partie d'un autre cas d'utilisation. Elle est représentée par une flèche discontinue stéréotypée « extension ».
  - Relation de généralisation : La relation de généralisation/spécialisation est la transposition de la notion d'héritage dans le paradigme objet. Elle est représentée par une flèche dont la pointe (un triangle fermé) est dirigée vers l'élément le plus général.

Dans notre système, l'acteur principal qui est le déclencheur de tous les cas d'utilisation et qui interagit avec l'application c'est l'administrateur réseau.

Les acteurs secondaires sont le contrôleur SDN et le système Spider-Net qui sont nécessaires pour le déroulement des cas d'utilisations.

Les services offerts par ***Route-Translator*** sont résumés dans les cas d'utilisations suivants :

- Introduire l'adresse du contrôleur SDN.
- Lister les nœuds cibles non affectés.
- Lister les nœuds sources non affectés.
- Faire correspondre les nœuds racines.
- Lancer l'auto-correspondance des topologies.
- Résoudre le conflit.
- Réinitialisation de la correspondance.
- Migrer vers le réseau cible.
- Réinitialisation réseau cible.

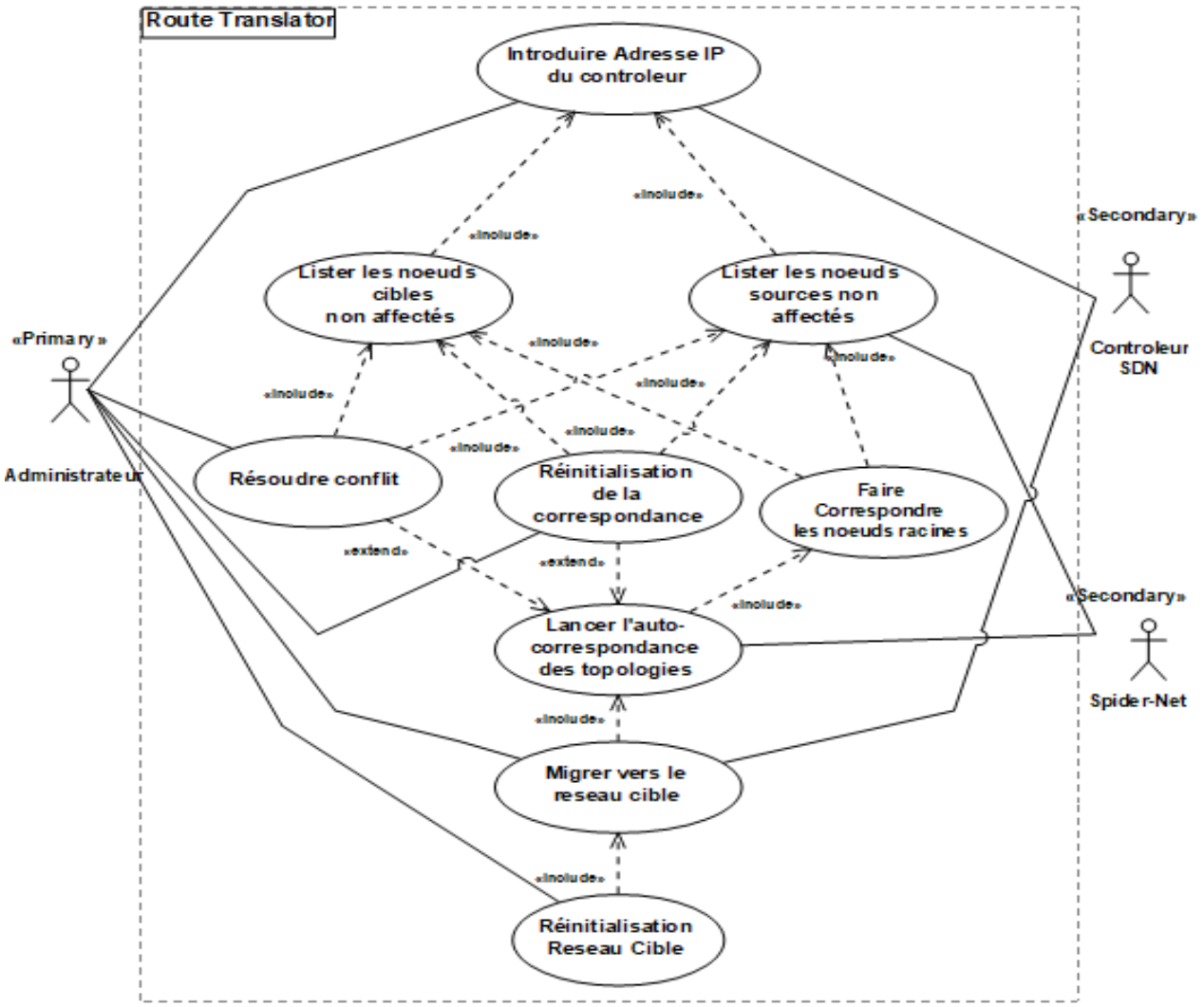

Figure 3.3 : Diagramme de cas d'utilisation du système Route Translator

➢ Description textuelle d'un cas d'utilisation :

À chaque cas d'utilisation doit être associée une description textuelle des interactions entre l'acteur et le système et les actions que le système doit réaliser en vue de produire les résultats attendus par les acteurs.

La description textuelle d'un cas d'utilisation est articulée dans les points suivants :

• **Objectif :** Décrire le contexte et les résultats attendus du cas d'utilisation.

• **Acteurs concernés** : le ou les acteurs concernés par le cas doivent être identifiés en précisant globalement leur rôle (acteur primaire et/ou secondaire).

• **Pré-conditions** : Si certaines conditions particulières sont requises avant l'exécution du cas, elles sont à exprimer à ce niveau.

• **Post - conditions** : Elles indiquent dans quel état se trouve le système après le déroulement de la séquence nominale.

• **Scénario nominal** : Le scénario principal qui doit se dérouler sans incident et qui permet d'aboutir au résultat souhaité.

• **Scénarios alternatifs** : Les autres scénarios, secondaires ou correspondants à la résolution d'anomalies, sont à décrire à ce niveau. Le lien avec le scénario principal se fait à l'aide d'une numérotation hiérarchisée (1.1, 1.2,…) rappelant le numéro de l'action concernée.

Cas d'utilisation « Introduire Adresse IP du contrôleur »

Acteur principal	L'administrateur réseau.
But	Tester la connectivité du contrôleur SDN.
Pré condition	- Le contrôleur SDN est joignable à partir du réseau. - Le système est connecté à la base de données.
Scénario nominal	1- Le système affiche le champ de saisie relatif à l'adresse IP du contrôleur SDN. 2- L'administrateur introduit l'adresse IP du contrôleur SDN. 3- L'administrateur valide la saisie. 4- Le système vérifie la validité de l'adresse IP, ensuite confirme la connectivité de l'adresse IP introduite. 5- Le système lister les nœuds du réseau source ainsi que ceux du réseau cible.
Scénario alternatif	4. A- L'adresse IP introduite est erronée ou le système n'arrive pas à se connecter au contrôleur SDN. 4. B - Le système affiche un message d'erreur et retourne au scénario nominal l'étape 3.
Scénario exceptionnel	Le système ne parvient pas à se connecter à la base de données ou la connectivité au contrôleur SDN a échoué : le système affiche un message d'erreur.

Tableau 3.1 : scénario du cas d'utilisation « Introduire adresse IP du contrôleur »

❖ **Cas d'utilisation « Lister les nœuds sources non affectés »**

Acteur principal	L'administrateur réseau.
But	Lister les nœuds du réseau source.
Pré condition	- Administrateur réseau a validé l'introduction de l'adresse IP du contrôleur. - La connectivité au contrôleur SDN est confirmée. - Le système est connecté à la base de données.
Scénario nominal	1- Le système affiche la liste des nœuds du réseau source à partir de la base de données.
Scénario exceptionnel	Le système ne parvient pas à se connecter à la base de données : le système affiche un message d'erreur.

Tableau 3.2 : Scénario du cas d'utilisation « Lister les nœuds sources non affectées »

❖ **Cas d'utilisation « Lister les nœuds cibles non affectés »**

Acteur principal	L'administrateur réseau.
But	Lister les nœuds du réseau cible.
Pré condition	- Administrateur réseau a validé l'introduction de l'adresse IP du contrôleur. - La connectivité au contrôleur SDN est confirmée.
Scénario nominal	1- Le système affiche la liste des nœuds du réseau cible à partir du contrôleur SDN.
Scénario exceptionnel	La connectivité au contrôleur SDN a échoué : le système affiche un message d'erreur.

Tableau 3.3 : Scénario du cas d'utilisation « Lister les nœuds cibles non affectées »

❖ **Cas d'utilisation « Faire correspondre les nœuds racines »**

Acteur principal	L'administrateur réseau.
But	Faire correspondre manuellement les nœuds racines.
Pré condition	- Les nœuds du réseau source sont listés. - Les nœuds du réseau cible sont listés.
Scénario nominal	1- L'administrateur choisit un nœud du réseau source. 2- L'administrateur sélectionne le nœud du réseau cible qui correspond au nœud source choisi. 3- Le système met à jour les listes des nœuds. 4- Le bouton « Lancer l'auto correspondance des topologies » est activé.

Tableau 3.4 : Scénario du cas d'utilisation « Faire correspondre les nœuds racines »

❖ Cas d'utilisation « Lancer l'auto-correspondance des topologies »

Acteur principal	L'administrateur réseau.
But	Faire correspondre automatiquement les deux topologies
Pré condition	- Le système connecté à la base de données. - Les nœuds racines sont correspondus.
Scénario nominal	1 - L'administrateur lance « l'auto-correspondance des topologies ». 2- Le système applique l'algorithme d'auto-correspondance des topologies expliqué dans la sous-section (4.1.2.4) du chapitre 2. 3 - Le système met à jour les listes des nœuds. 4 - Le système affiche une notification « Fin de la correspondance des topologies ». 5 - Le bouton « Migrer vers le réseau SDN » est activé.
Scénario alternatif	2. A- Le système rencontre un conflit pendant la correspondance automatique des topologies. 2. B- Le système affiche une notification « un conflit s'est produit » informant l'administrateur de passer a la tache : « Résoudre conflit ».
Scénario exceptionnel	Le système ne parvient pas à se connecter à la base de données : le système affiche un message d'erreur.

Tableau 3.5 : Scénario du cas d'utilisation « Lancer l'auto correspondance des topologies »

❖ Cas d'utilisation « Réinitialisation de la correspondance »

Acteur principal	L'administrateur réseau.
But	Réinitialiser la correspondance des topologies
Pré condition	L'auto-correspondance des topologies a déjà été lancée.
Scénario nominal	1 - L'administrateur lance « Réinitialiser la correspondance ». 2 - Le système met à jour les listes des nœuds. 3- Le système retourne à la tache « Faire correspondre les nœuds racines ».
Scénario exceptionnel	Le système ne parvient pas à se connecter à la base de données : le système affiche un message d'erreur.

Tableau 3.6 : Scénario du cas d'utilisation «Réinitialisation de la correspondance »

❖ Cas d'utilisation « Résoudre conflit »

Acteur principal	L'administrateur réseau.
But	L'administrateur résout manuellement le conflit.
Pré condition	- Le système rencontre un conflit pondant la correspondance des topologies.
Scénario nominal	1- L'administrateur choisit un nœud du réseau source. 2- L'administrateur sélectionne le nœud du réseau cible qui correspond au nœud source choisi. 3- Le système met a jour les listes des nœuds.

Tableau 3.7 : Scénario du cas d'utilisation «Résoudre conflit »

❖ **Cas d'utilisation « Migrer vers le réseau cible »**

Acteur principal	L'administrateur réseau.
But	La migration du réseau source vers le réseau SDN.
Pré condition	- La connectivité au contrôleur SDN est confirmée. - Le système est connecté à la base de données. - L'étape de la correspondance des topologies est accomplie.
Scénario nominal	1- L'administrateur lance la migration vers le réseau cible 2- Le système configure l'application du routage SDN selon les étapes suivantes : 2.1- La configuration des adresses IP. 2.2- La configuration des routes. 3- Le système affiche une notification « la migration vers le réseau SDN est terminée ». 4- Le bouton « Réinitialiser le réseau cible » est activé.
Scénario exceptionnel	Le système ne parvient pas à se connecter à la base de données ou la connectivité au contrôleur SDN a échoué : le système affiche un message d'erreur.

Tableau 3.8 : Scénario du cas d'utilisation « Migrer vers le réseau cible »

❖ **Cas d'utilisation « Réinitialisation du réseau cible »**

Acteur principal	L'administrateur réseau.
But	La réinitialisation du réseau SDN.
Pré condition	- Le processus de la migration vers le réseau SDN est accompli .
Scénario nominal	1- l'administrateur lance « réinitialiser le réseau SDN ». 2- Le système supprime toutes les configurations du réseau SDN. 3-Le système affiche une notification « Le réseau SDN est réinitialisé ».
Scénario exceptionnel	La connectivité au contrôleur SDN a échoué: le système affiche un message d'erreur.

Tableau 3.9 : Scénario du cas d'utilisation « Réinitialisation du réseau cible »

➢ Diagramme de séquence système

A-Définition

Le diagramme de séquence système représente la succession chronologique des opérations réalisées par un acteur à savoir : saisir une donnée, consulter une donnée, lancer un traitement… etc. Il montre les interactions entre les objets selon un point de vue temporel [36,38].

Les objets communiquent en échangeant des messages représentés au moyen de flèches horizontales, orientées de l'émetteur du message vers le destinataire. L'ordre d'envoi des messages en fonction du temps est donné par la position sur l'axe vertical.

B- Les composants d'un diagramme de séquence :

Dans un diagramme des séquences, les classes et les acteurs sont énumérés en colonnes, avec leurs lignes de vie verticales indiquant la durée de vie de l'objet.

- **Les objets :** sont des instances des classes, et sont rangés horizontalement. La représentation graphique pour un objet est similaire à une classe (un rectangle) précédée du nom d'objet (facultatif) et deux-points (:).
- **Les lignes de vie** : identifient l'existence de l'objet par rapport au temps. La notation utilisée pour une ligne de vie est une ligne pointillée verticale partant de l'objet.
- **Les activations :** sont modélisées par des boîtes rectangulaires sur la ligne de vie. Elles indiquent quand l'objet effectue une action.
- **Message :** modélisés par des flèches horizontales entre les activations, indiquent une communication entre des lignes de vie. Ainsi, une communication d'un objet vers un autre objet. La réception d'un message est considérée par l'objet récepteur comme un événement qu'il faut traiter (ou pas). Plusieurs types de messages existent, les plus communs sont : message synchrone, et message asynchrone.

❖ **Cas d'utilisation « Introduire adresse IP du contrôleur »**

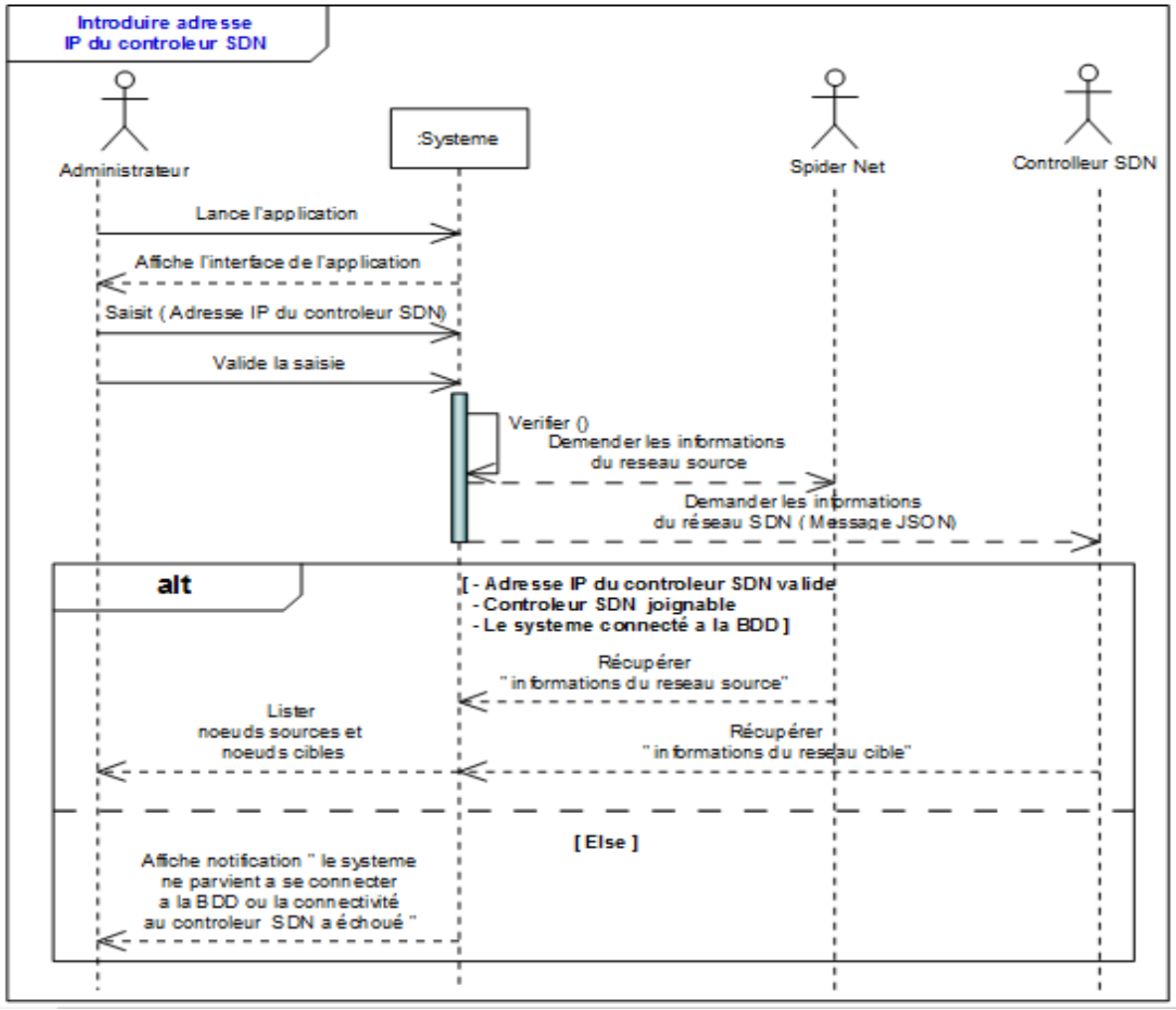

Figure 3.4 : Diagramme de séquence cas d'utilisation «Introduire adresse IP du contrôleur»

❖ **Cas d'utilisation «Faire correspondre les nœuds racines »**

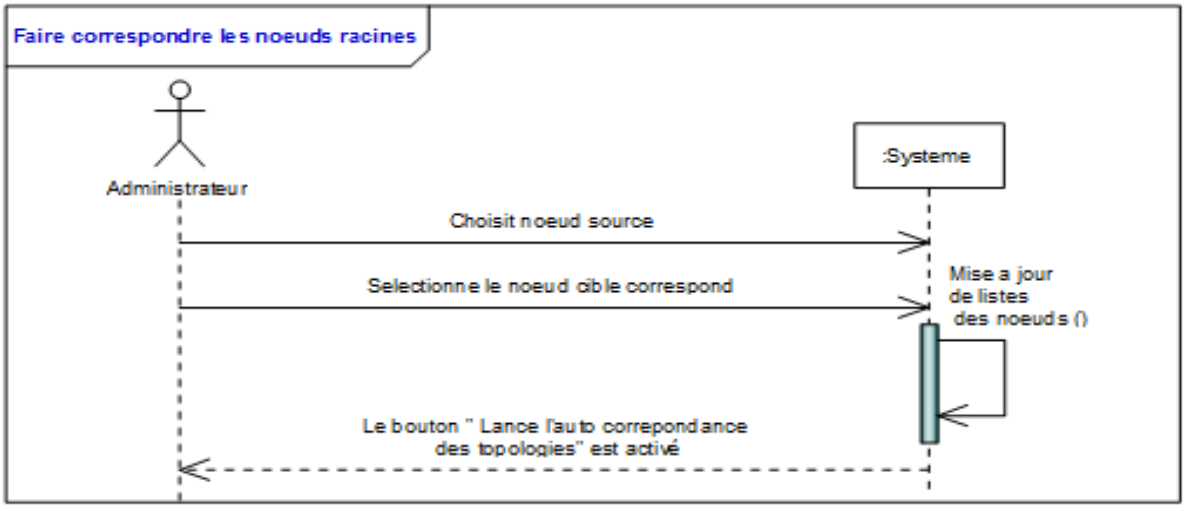

Figure 3.5 : Diagramme de séquence cas d'utilisation « Faire correspondre les nœuds racines »

❖ **Cas d'utilisation « Lancer l'auto correspondance des topologies»**

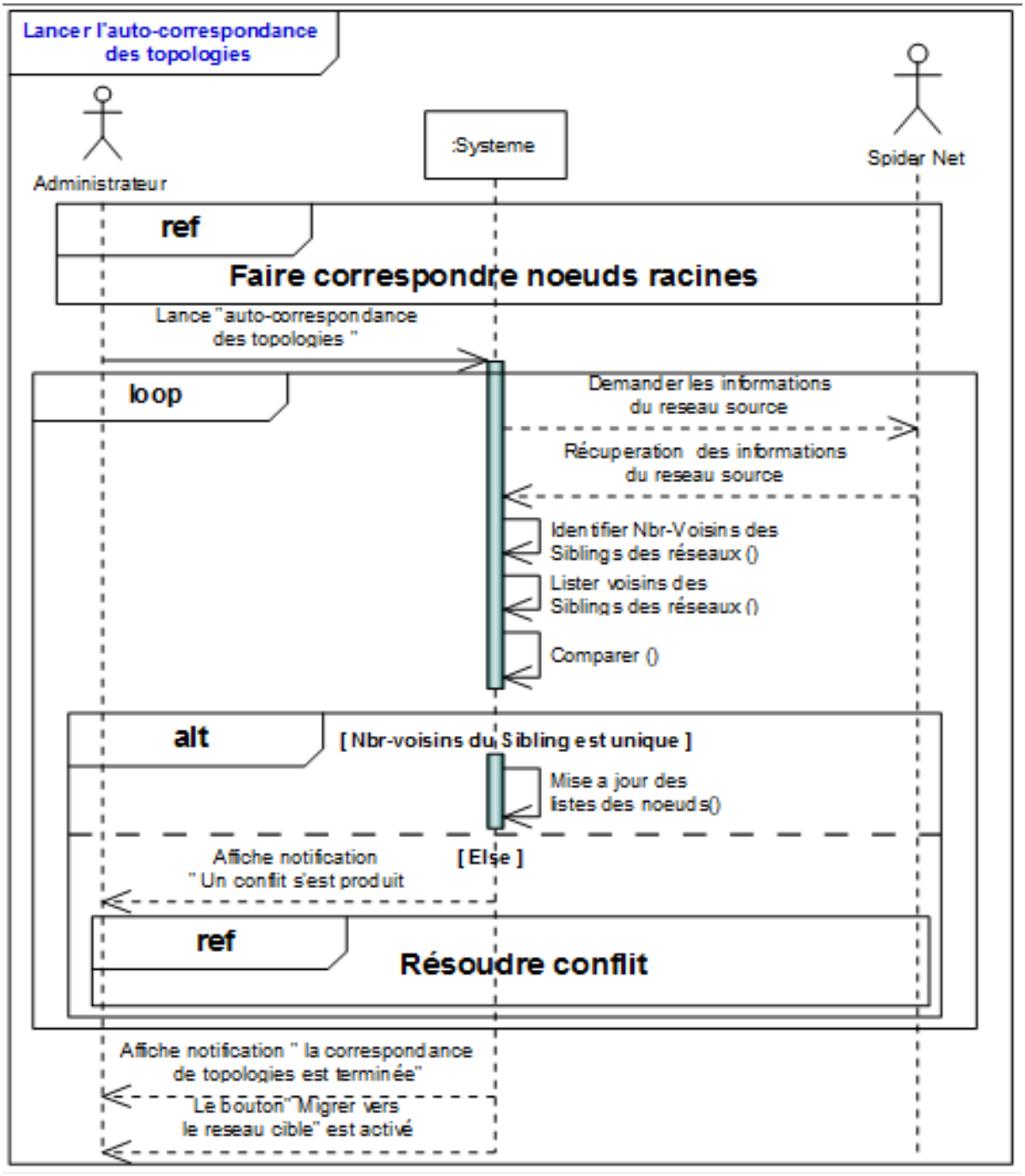

Figure 3.6 : Diagramme de séquence cas d'utilisation « Lancer l'auto correspondance des topologies»

❖ **Cas d'utilisation « Résoudre conflit»**

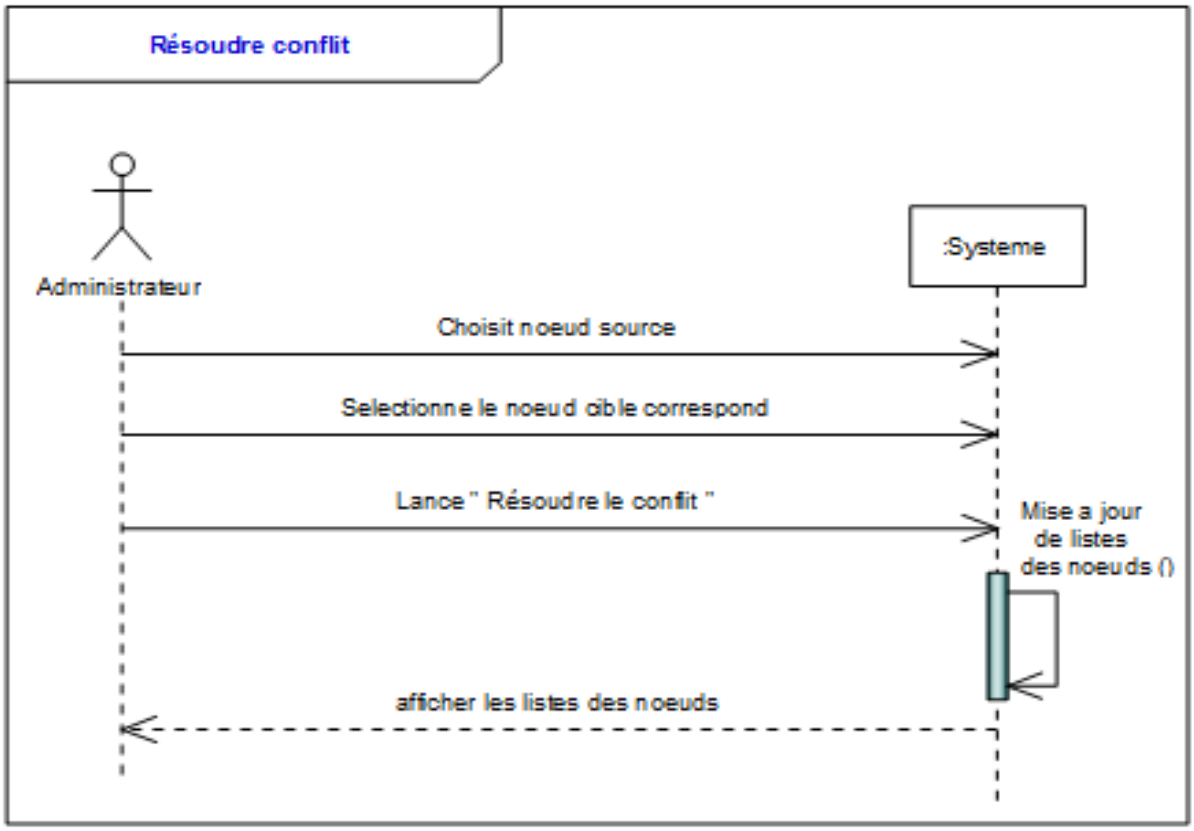

Figure 3.7 : Diagramme de séquence cas d'utilisation « Résoudre conflit»

❖ **Cas d'utilisation « Réinitialisation de la correspondance»**

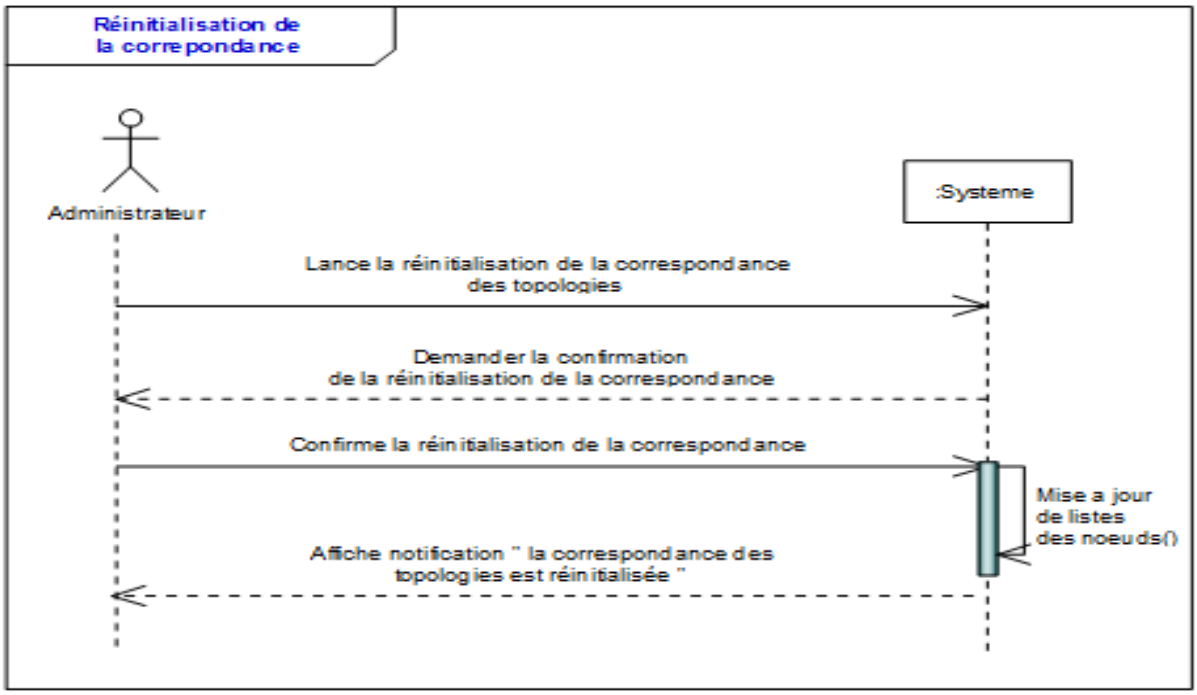

Figure 3.8 : Diagramme de séquence cas d'utilisation « Réinitialisation de la correspondance»

❖ **Cas d'utilisation « Migrer vers le réseau cible »**

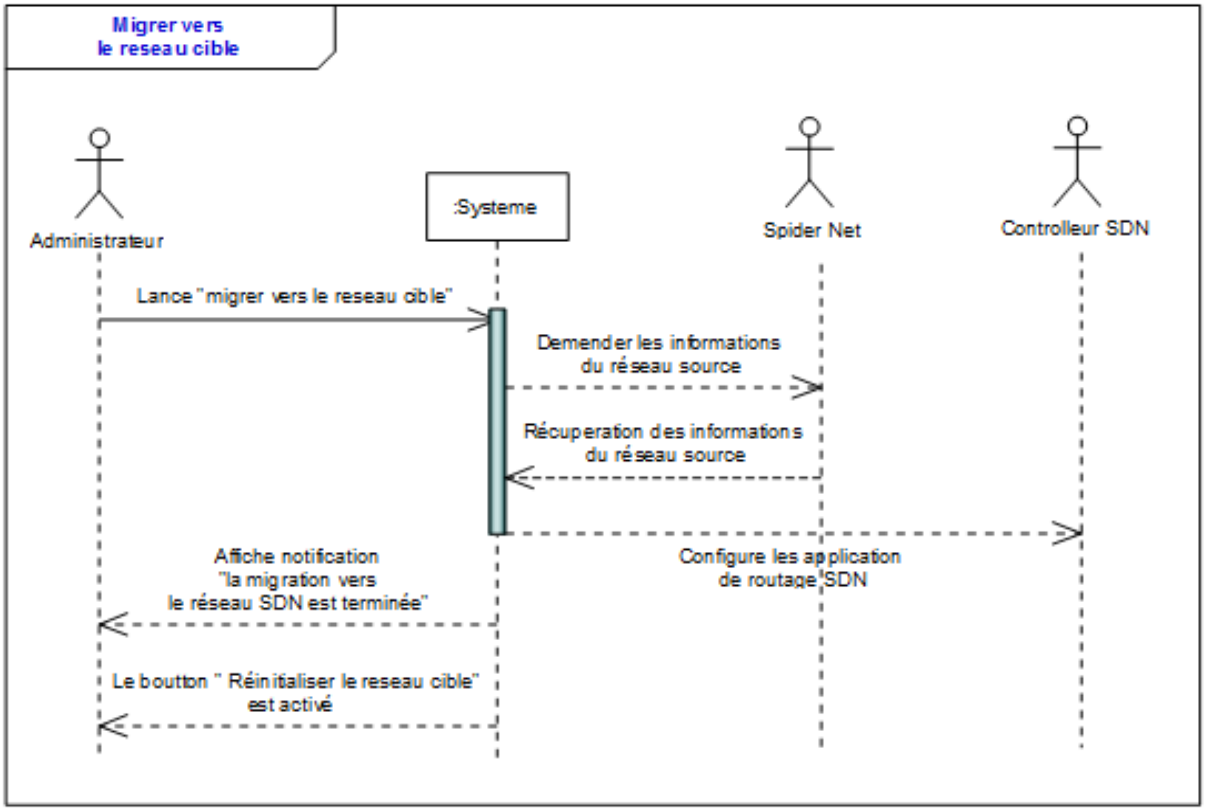

Figure 3.9 : Diagramme de séquence cas d'utilisation « Migrer vers le réseau cible »

❖ **Cas d'utilisation « Réinitialisation du réseau cible»**

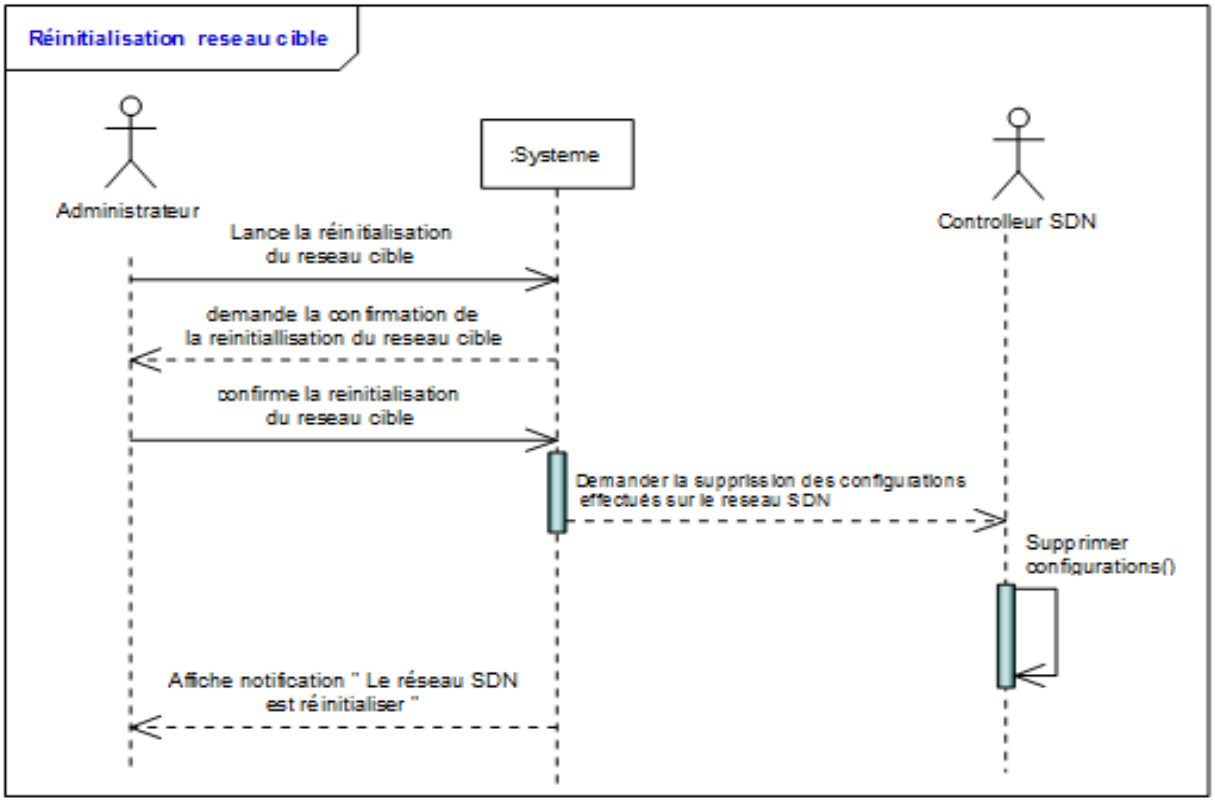

Figure 3.10 : Diagramme de séquence cas d'utilisation « Réinitialisation du réseau cible»

1.3.2. Phase d'analyse

La phase d'analyse permet de décrire le comportement du système :

➢ Diagramme d'activité

A-Définition

Le diagramme d'activité représente le déroulement d'un cas d'utilisation réalisé par le système, avec tous les branchements conditionnels et toutes les boucles possibles. C'est un graphe orienté d'actions et de transitions. Les transitions sont franchies lors de la fin des actions, des étapes peuvent être réalisées en parallèle ou en séquence. Les diagrammes d'activités permettent de mettre l'accent sur les traitements. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle et de flots de données. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation [39].

B- Les composants de base de diagramme d'activité :

- Nœud initial : Il indique le début de déroulement de cas d'utilisation modélisé, un nœud initial est un nœud de contrôle à partir duquel le flot débute lorsque l'activité enveloppante est invoquée. Graphiquement, un nœud initial est représenté par un petit cercle plein.

- Nœud final : Il indique la fin de déroulement de cas d'utilisation modélisé, un nœud final est un nœud de contrôle possédant un ou plusieurs arcs entrants et aucun arc sortant. Graphiquement, un nœud final est représenté par un cercle plein entouré d'un autre cercle.

- Nœud de décision : Un nœud de décision est un nœud de contrôle qui permet de faire un choix entre plusieurs flots sortants. Il possède un arc entrant et plusieurs arcs sortants. Ces derniers sont généralement accompagnés de conditions de garde pour conditionner le choix. Graphiquement, on représente un nœud de décision par un losange.

- Le nœud d'action : Un nœud d'action est un état d'activité exécutable qui constitue l'unité fondamentale de fonctionnalité exécutable dans une activité.

- La transition : Quand un état d'activité est accompli, le traitement passe à un autre état d'activité. Les transitions sont utilisées pour marquer ce passage. Les transitions sont modélisées par des flèches.

❖ **Cas d'utilisation « Introduire adresse IP du contrôleur»**

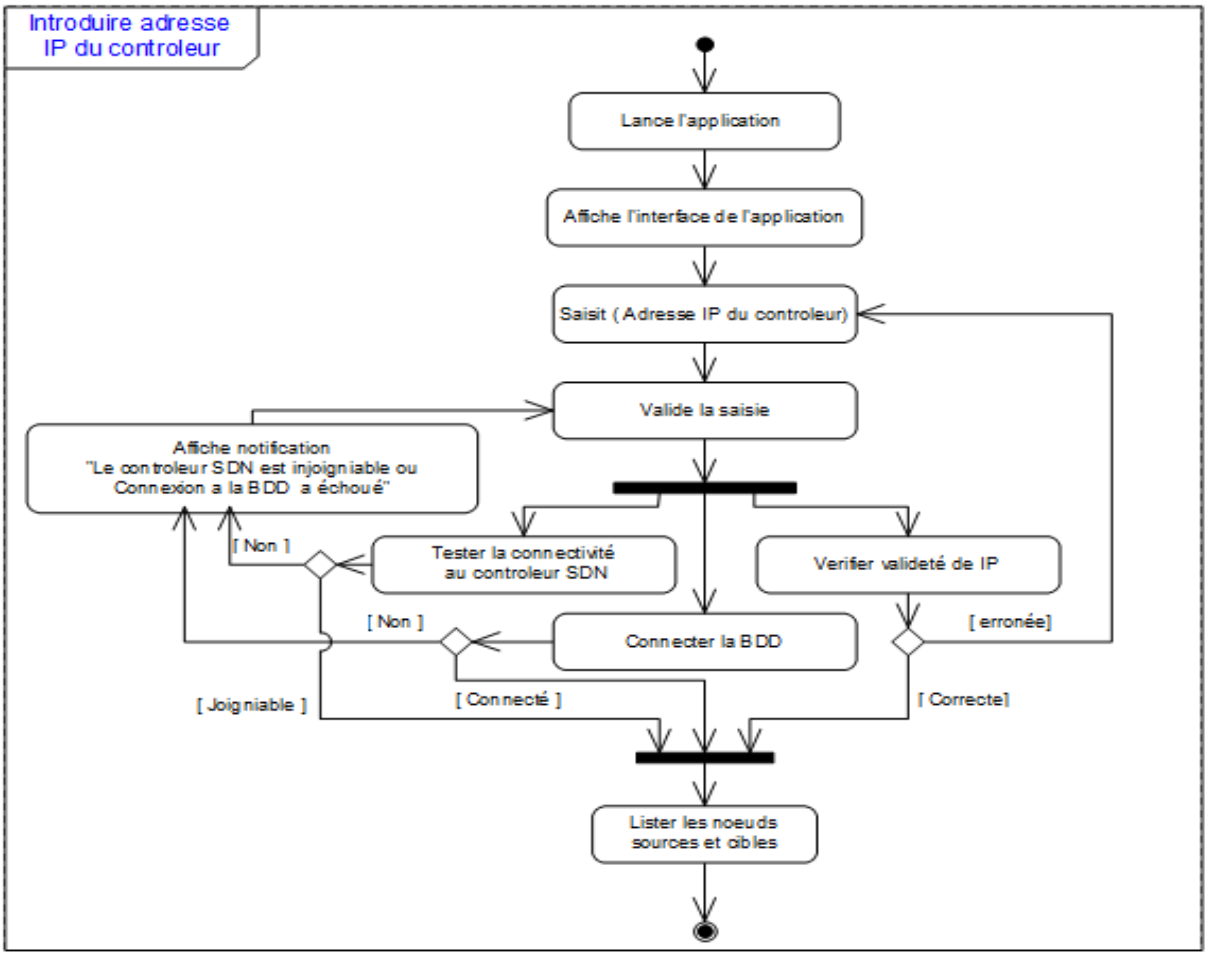

Figure 3.11 : Diagramme d'activité du cas d'utilisation « Introduire adresse IP du contrôleur »

❖ **Cas d'utilisation « Faire correspondre les nœuds racines»**

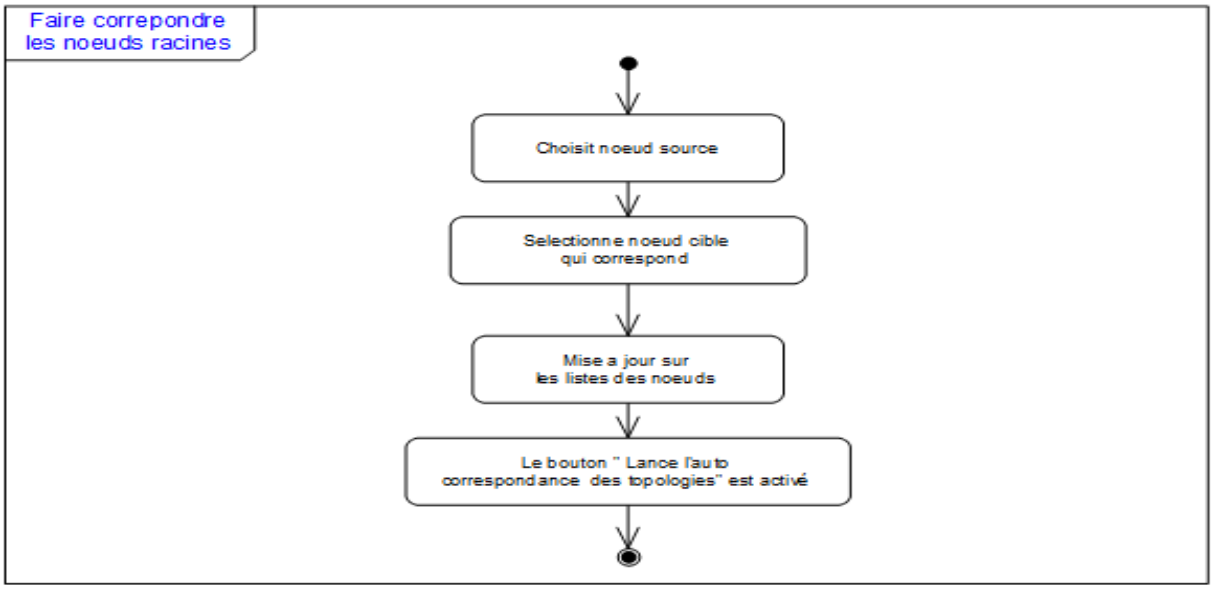

Figure 3.12 : Diagramme d'activité du cas d'utilisation « Faire correspondre les nœuds racines »

❖ **Cas d'utilisation « Lance l'auto- correspondance des topologies»**

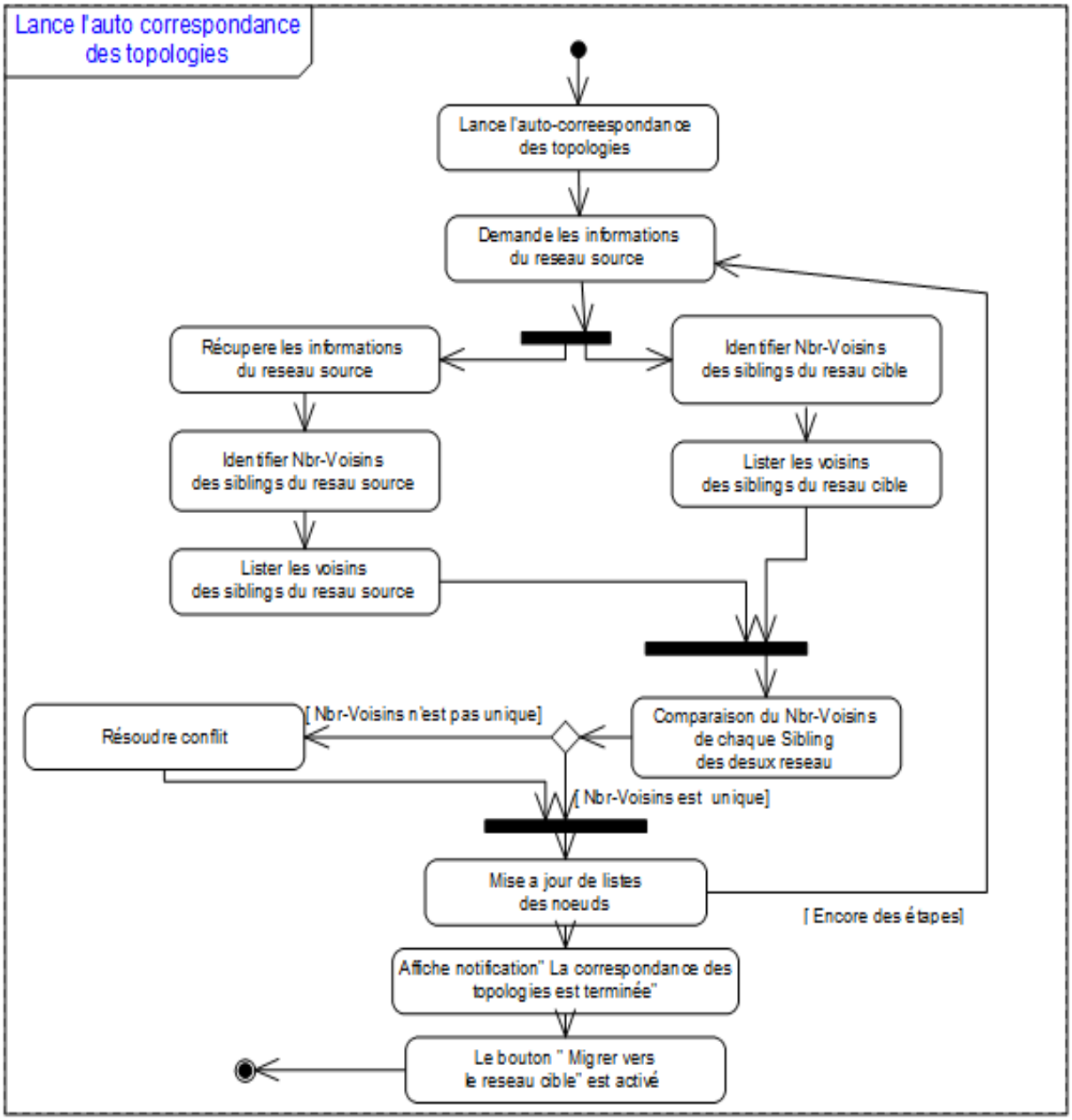

Figure 3.13 : Diagramme d'activité du cas d'utilisation « Lance l'auto correspondance des topologies »

❖ **Cas d'utilisation « Résoudre conflit»**

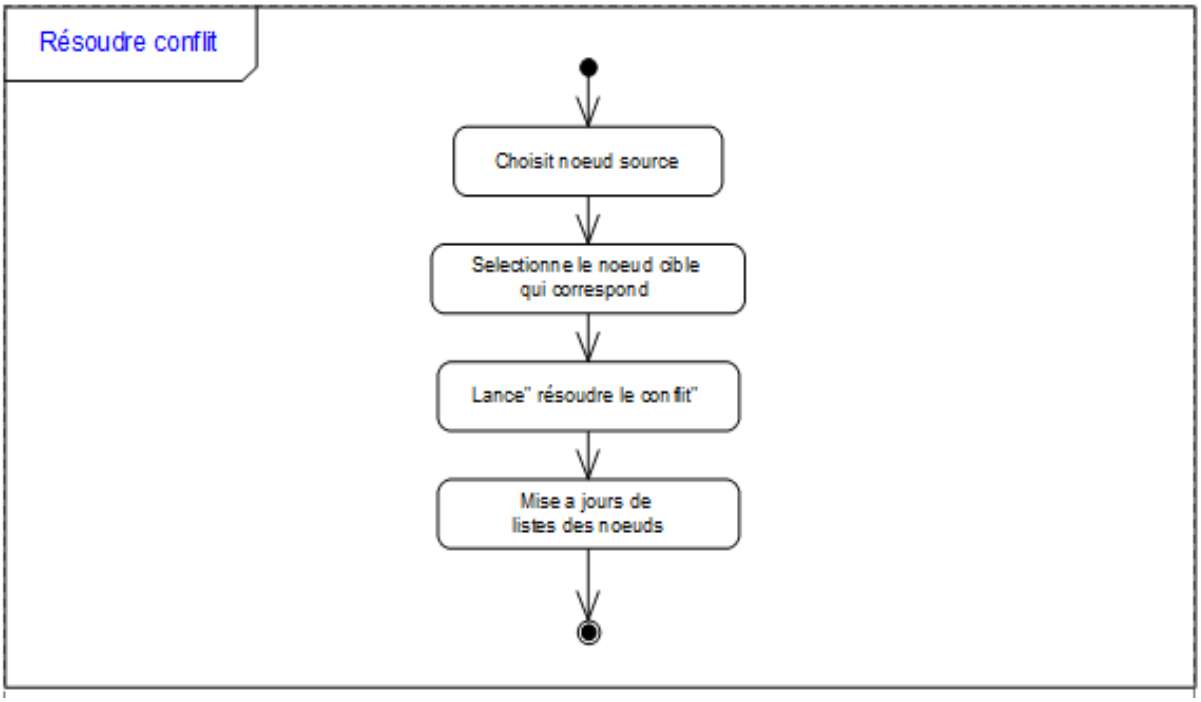

Figure 3.14 : Diagramme d'activité du cas d'utilisation « Résoudre conflit »

❖ **Cas d'utilisation « Réinitialisation de la correspondance»**

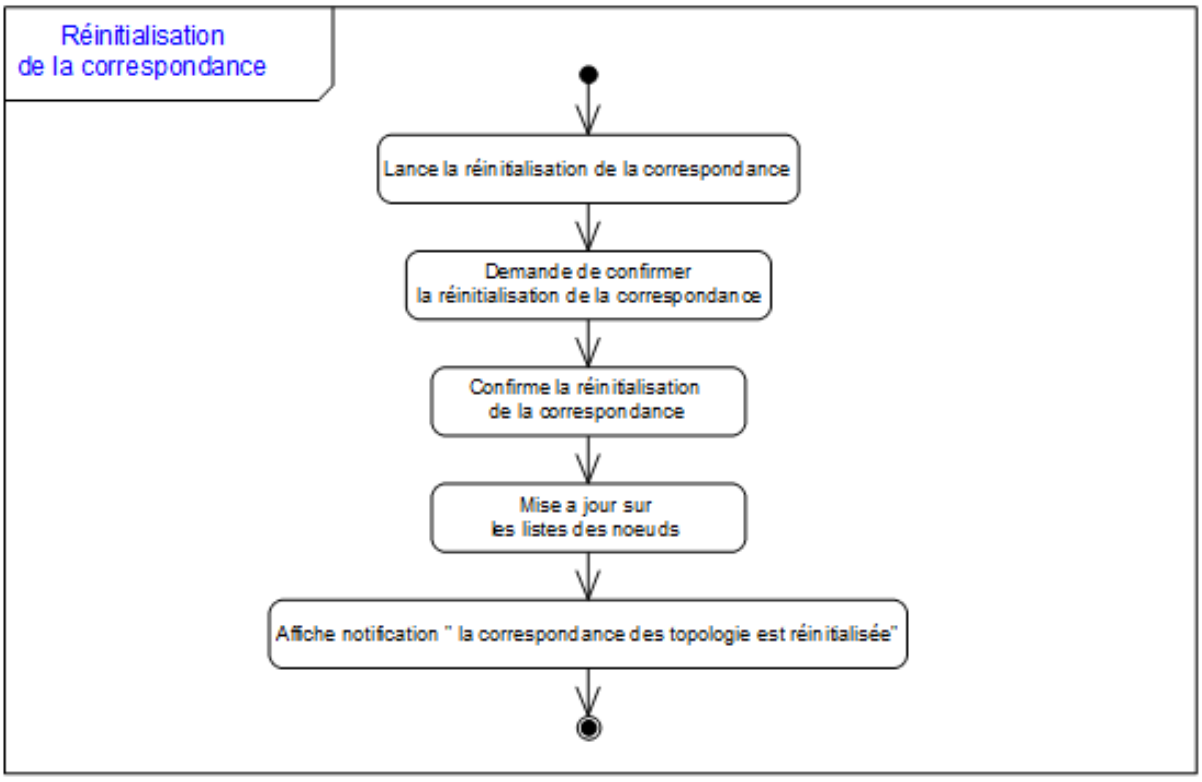

Figure 3.15 : Diagramme d'activité du cas d'utilisation « Réinitialisation de la correspondance »

❖ **Cas d'utilisation « Migrer vers le réseau cible »**

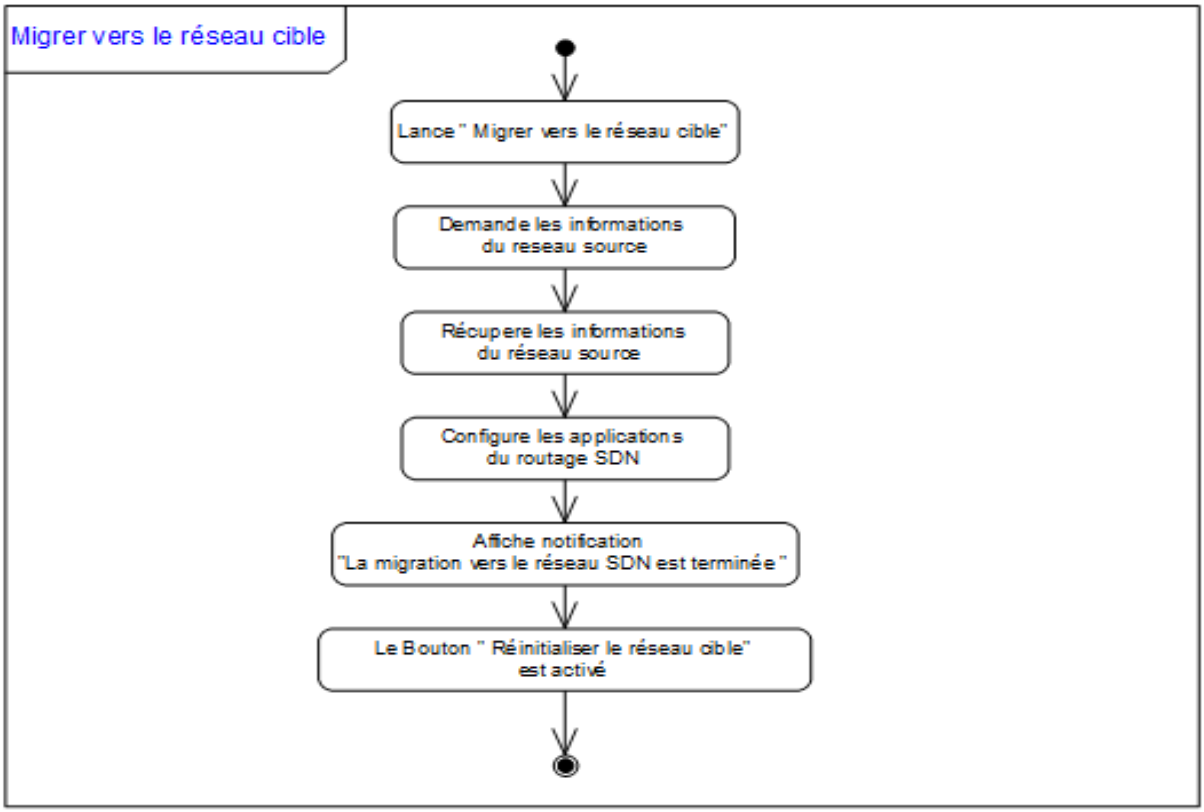

Figure 3.16 : Diagramme d'activité du cas d'utilisation « Migrer vers le réseau cible

❖ **Cas d'utilisation « Réinitialisation réseau cible»**

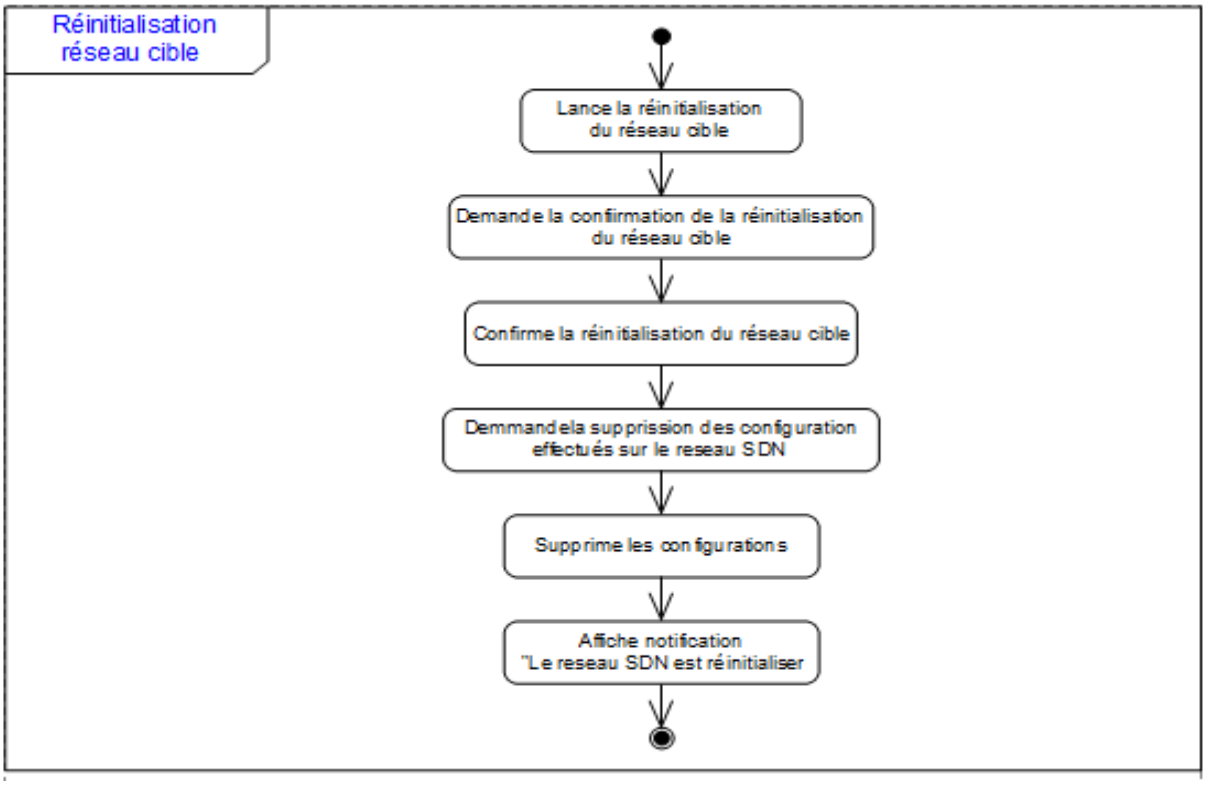

Figure 3.17 : Diagramme d'activité du cas d'utilisation « Réinitialisation réseau cible»

1.3.3. Phase de conception

- **Diagramme de classe**

A-Définition

Il est considéré comme le plus important de la modélisation orientée objet, ce diagramme représente la vue statique du système, le diagramme de classe est un diagramme entité association qui identifie la structure des classes d'un système, y compris les propriétés et les méthodes de chaque classe ainsi que les différentes relations entre celles-ci.

B- Les composants de base de diagramme de classe :

- Les classes : Sont les modules de base de la programmation orientée objet. Une classe est représentée en utilisant un rectangle divisé en trois sections. La section supérieure est le nom de la classe. La section centrale définit les propriétés de la classe. La section du bas énumère les méthodes de la classe.

- L'association : représente une relation sémantique durable entre deux classes. Elle est modélisée par une ligne reliant les deux classes. Cette ligne peut être qualifiée avec le type de relation, et peut également comporter des règles de multiplicité (par exemple un à un, un à plusieurs, plusieurs à plusieurs) pour la relation.

-La composition : Si une classe ne peut pas exister par elle-même, mais doit être un membre d'une autre classe, alors elle possède une relation de composition avec la classe contenant. Une relation de composition est indiquée par une ligne avec un "diamant" rempli. Une composition est une agrégation plus forte impliquant que :

- Un élément ne peut appartenir qu'à un seul agrégat composite (agrégation non partagée)

- La destruction de l'agrégat composite entraîne la destruction de tous ses éléments(le composite est responsable du cycle de vie des parties).

-L'agrégation : Indique une relation de contenance, Elle décrite par une relation "possède". Une relation d'agrégation est représentée par une ligne avec un "diamant" creux.

-La généralisation : Est l'équivalent d'une relation d'héritage en terme orienté objet. Une relation de généralisation est indiquée par une flèche creuse se dirigeant vers la classe "parent".

-La dépendance : La dépendance entre deux classes permet de représenter l'existence d'un lien sémantique. Une classe B est en dépendance de la classe A si des éléments de la classe A sont nécessaires pour construire la classe B. La relation de dépendance se représente par une flèche en pointillé.

En se qui concerne notre projet, le système Route-Translator utilise une base de données existante (déjà remplie) qui respecte la structure de données proposée pour l'application Spider-Net. Cette base de données contient les informations relatives à la topologie réseau traditionnel (réseau source) collectées par la solution de découverte automatique Spider Net.

Donc, le diagramme de classe de notre système est inspiré du diagramme de classe du système Spider Net, à l'exception de la section des méthodes de chaque classe qui a été remplacée afin de refléter les méthodes utilisées par le système Route-Translator.

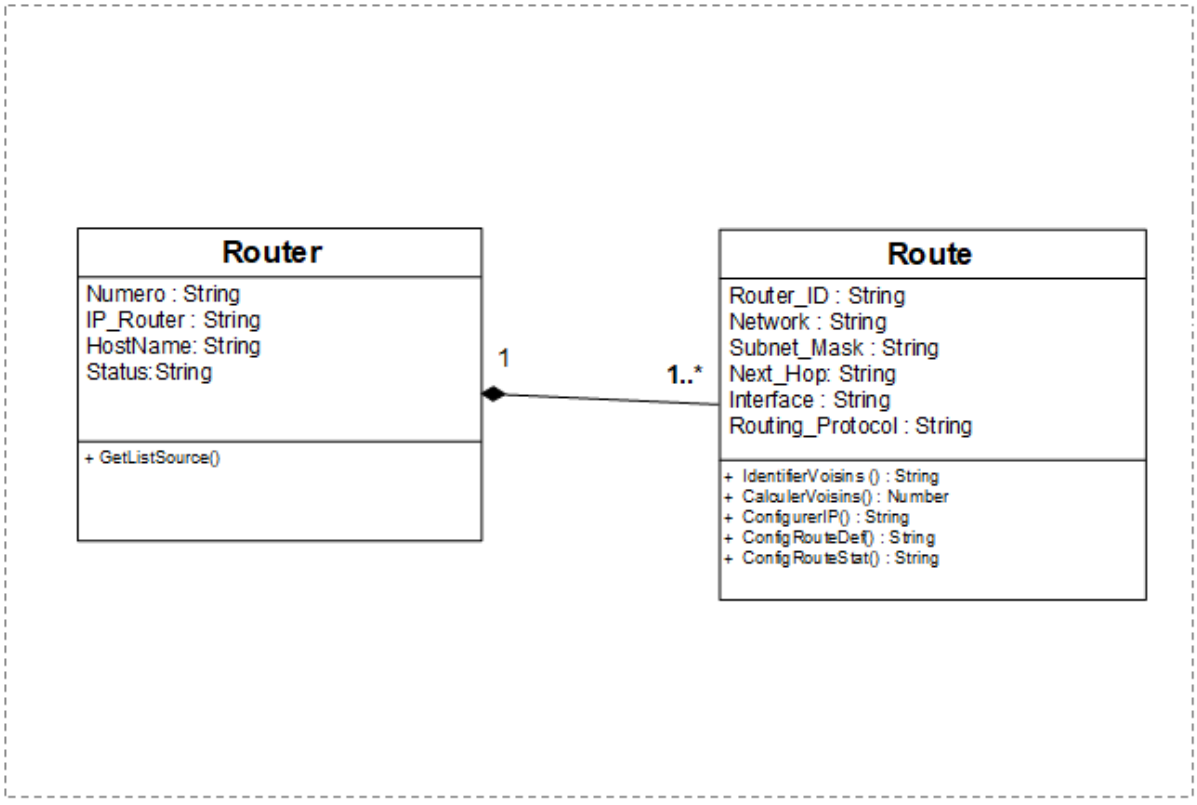

Figure 3.18 : Diagramme de classe du système Route-Translator

- **les règles de gestion:**

  - Chaque routeur contient plusieurs routes dans sa table de routage.
  - Chaque route dans la table de routage n'appartient qu'à un seul routeur.

2. Réalisation de Route Translator

Dans cette partie, nous présentons l'environnement de développement de notre système.

2.1. Machine virtuelle (VirtualBox)

La VirtualBox est un package de virtualisation logicielle qui s'installe sur un système d'exploitation en tant que une application. La VirtualBox permet l'installation des systèmes d'exploitation supplémentaires sur celui-ci, en tant que système d'exploitation invité, et son exécution dans un environnement virtuel [40].

2.2. Ryu

Nous avons opté pour le contrôleur SDN Ryu car il présente des caractéristiques équitables, c'est le bon choix pour les applications de recherche et les petites entreprises. Étant codé en Python, ce contrôleur présente des fonctionnalités pour le développement d'applications et de modules (voir **chapitre 1 sous-section 7.4.6)**.

2.3. Mininet

Pour la simulation de notre réseau SDN, nous avons utilisé mininet, cet outil est facile à utiliser parce qu'il fournit une simplicité dans la construction des topologies SDN, il est conçu pour supporter la recherche, le développement et l'apprentissage dans les technologies SDN. En outre, cet outil est très utile pour ceux qui possèdent un contrôleur SDN et utilisent mininet pour vérifier son comportement et ses performances (voir **chapitre 1 sous-section 8.1)**.

2.4. Netbeans

NetBeans est un environnement de développement intégré (EDI), open source, très utile qui permet de développer en java. NetBeans permet également de supporter une large variété de langages de programmation telle que C, C++, JavaScript, XML, PHP et HTML.

Cet outil comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, éditeur graphique d'interfaces et de pages Web).

NetBeans constitue par ailleurs une plate-forme qui permet le développement d'applications spécifiques (bibliothèque Swing [Java]) [41].

2.5. Serveur XAMPP

(X : Un des systèmes d'exploitation, Apache http server, MySQL data base, PHP and Perl) Un ensemble de logiciels permettant de facilement créer une interface web interagissant avec une base de données SQL, avec le MySQL XAMPP se compose de l'un des systèmes de gestion de base de données relationnelle les plus populaires du monde, MySQL sert à l'enregistrement de données pour des services [42].

2.6. Le langage de programmation JAVA

Java est un langage de programmation orienté objet. Il est très utilisé dans le domaine de développement créé par James Gosling et Patrick Naughton, employés de Sun Microsystems. Il est fourni avec un ensemble d'outils (JDK Java Development Kit) et un ensemble de packages : ensemble de classes. Ces différentes classes de base couvrent beaucoup de domaines (entrées/sorties, interface graphique, réseau, etc.) Cette richesse en "librairies standards" explique sûrement en partie le succès de Java.

La particularité principale de java est que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels que : Unix, Microsoft Windows, Mac OS ou Linux avec peu ou pas de modifications. C'est la plate-forme qui garantit la portabilité des applications développées en Java [43].

2.7. Présentation de JDK

JDK (Java Développent Kit) est nécessaire pour développer des applications Java. Le JDK contient le compilateur Java et les bibliothèques de programmation nécessaires à l'élaboration de programmes Java [44].

2.8. JDBC

JDBC (Java Database Connectivity) est une interface de programmation d'application (API) pour le langage de programmation Java, qui définit la manière dont un client peut accéder à tout type de données, en particulier de base de données relationnelle. Il fait partie de la plate-forme Java Standard Edition d'Oracle Corporation. Il agit comme une interface de couche intermédiaire entre les applications Java et la base de données [45].

2.9. JSON-simple

JSON-simple est une bibliothèque pour le traitement simple des messages de format JSON cette bibliothèque peut être utilisée pour la lecture ainsi que l'écriture d'un fichier de format JSON [46].

3. Simulation de l'utilisation du système Route Translator

Cette section représente un guide d'utilisation pour notre solution Route-Translator. Afin de migrer vers le réseau SDN, l'utilisateur de Route Translator doit respecter les points suivants :

Premièrement, il faut créer un réseau SDN physique/virtuel similaire au réseau traditionnel qui représente la source de migration. Dans notre cas le réseau SDN est simulé utilisant Mininet. Initialement le réseau SDN n'a aucun plan d'adressage ou bien de routage (voir **la Figure 3.19**).

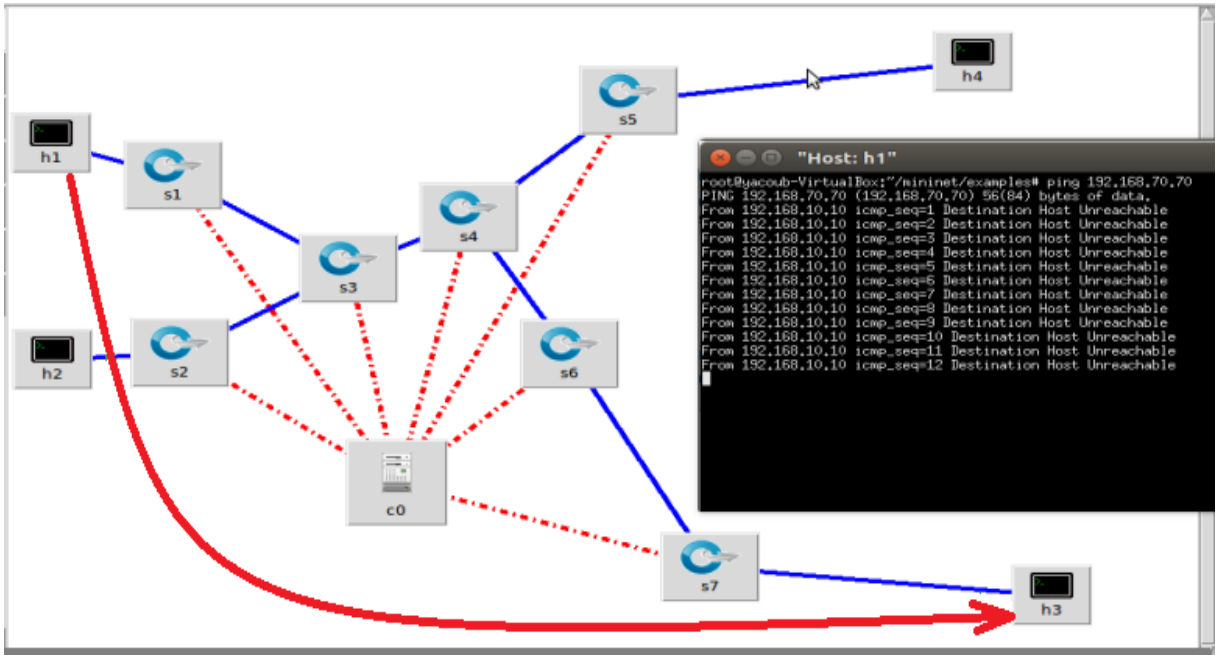

Figure 3.19 : Le réseau SDN cible sans déploiement du routage

Une fois que le réseau SDN est conçu, l'administrateur lance Route-Translator. La première des choses à faire c'est d'introduire l'adresse IP du contrôleur SDN dans le champ de saisie qui se trouve en haut. Une fois validé, le système récupère la liste des nœuds se trouvant dans les deux réseaux source et cible (voir **la Figure 3.20**).

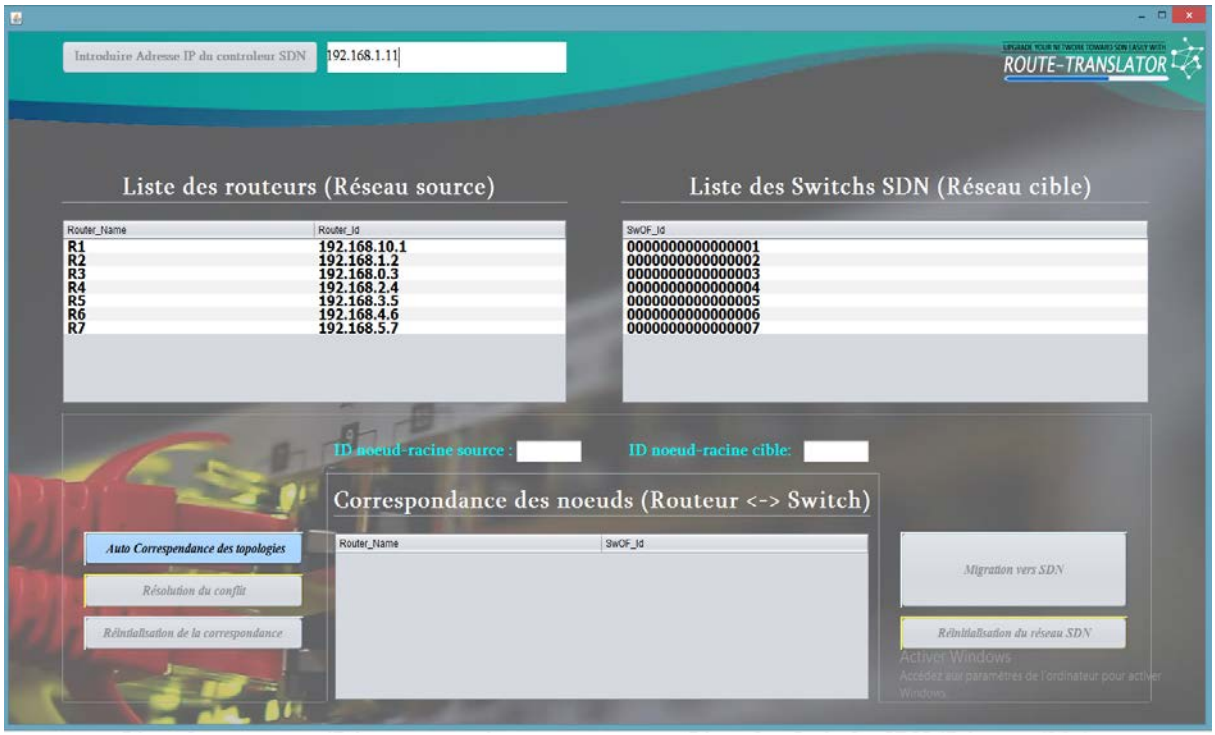

Figure 3.20 : Introduction de l'adresse IP du contrôleur SDN.

Juste après, l'administrateur doit introduire les nœuds racines (source et cible) afin qu'il puisse lancer l'auto-correspondance des topologies (voir **la Figure 3.21**). Lorsque la tache de la correspondance des topologies est terminée avec succès, le bouton « Migrer vers le SDN » s'active (voir **la Figure 3.23**).

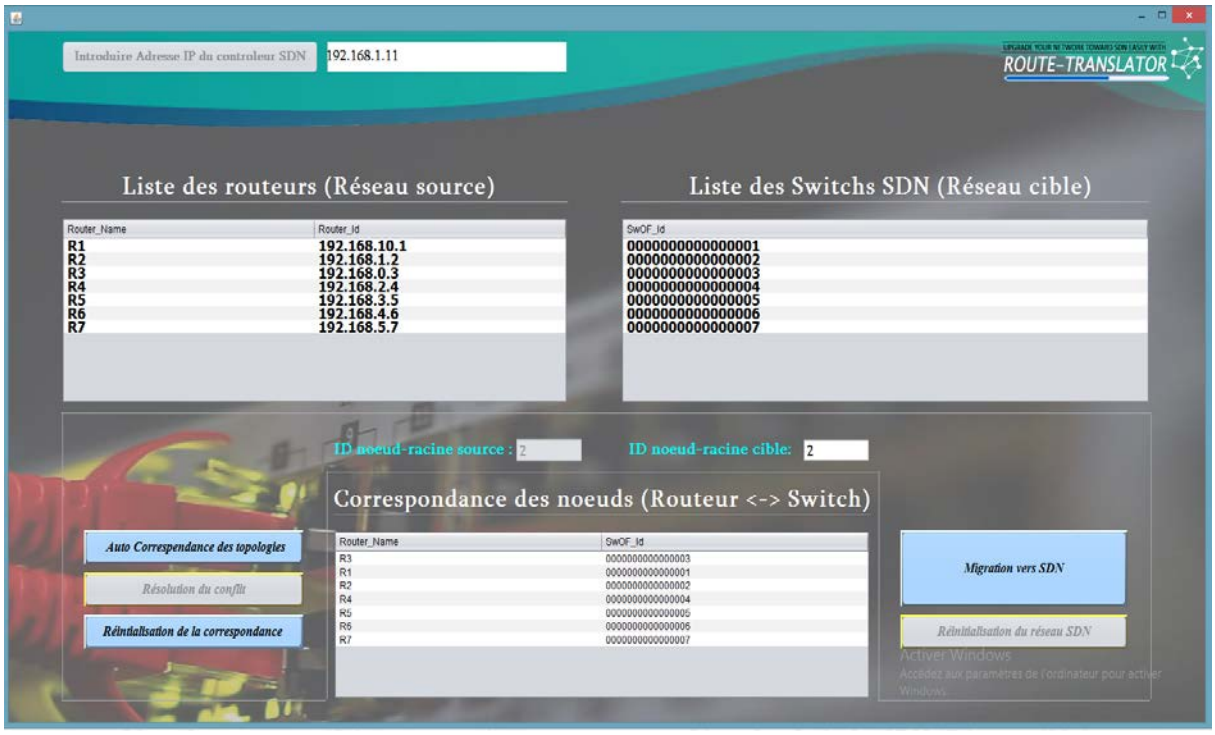

Figure 3.21 : La correspondance des topologies

Dans le cas ou la correspondance des topologies échoue, le système signale à l'administrateur (via une boite de dialogue) cet échec afin de lui permettre de réinitialiser cette tache a partir du bouton « Réinitialiser la correspondance » (voir **la Figure 3.22**).

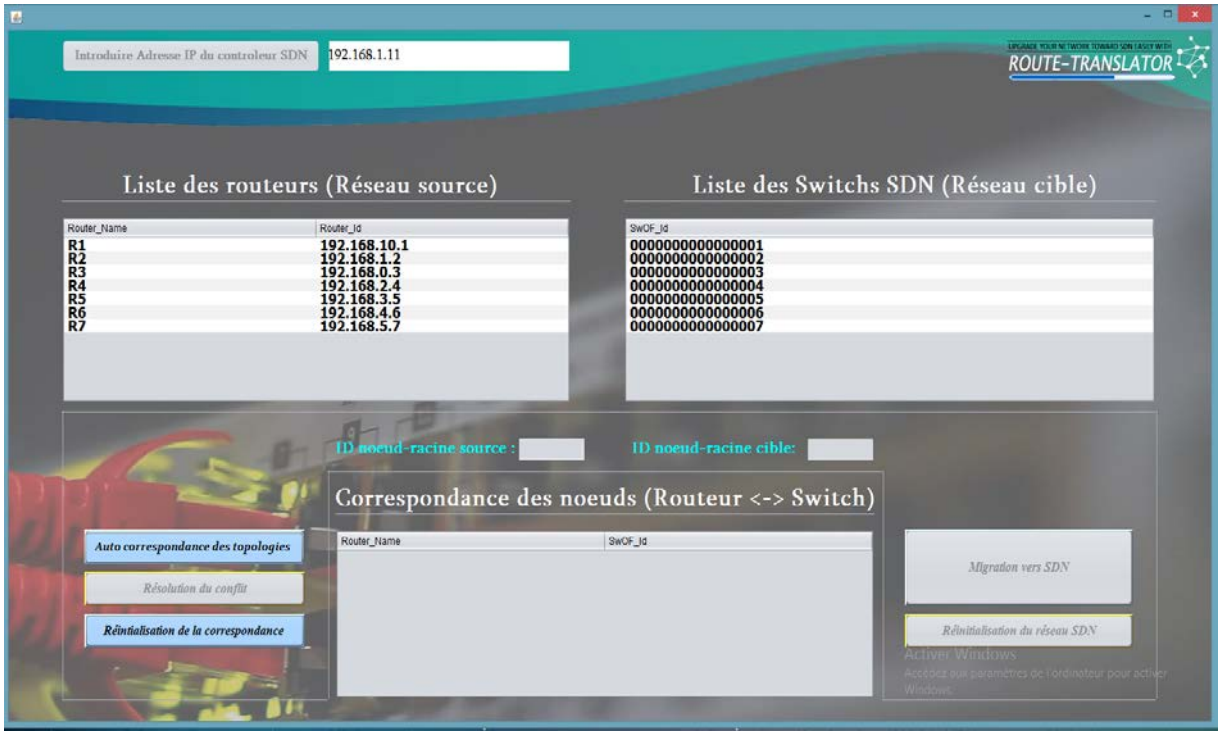

Figure 3.22 : La réinitialisation de la correspondance des topologies suite à un échec

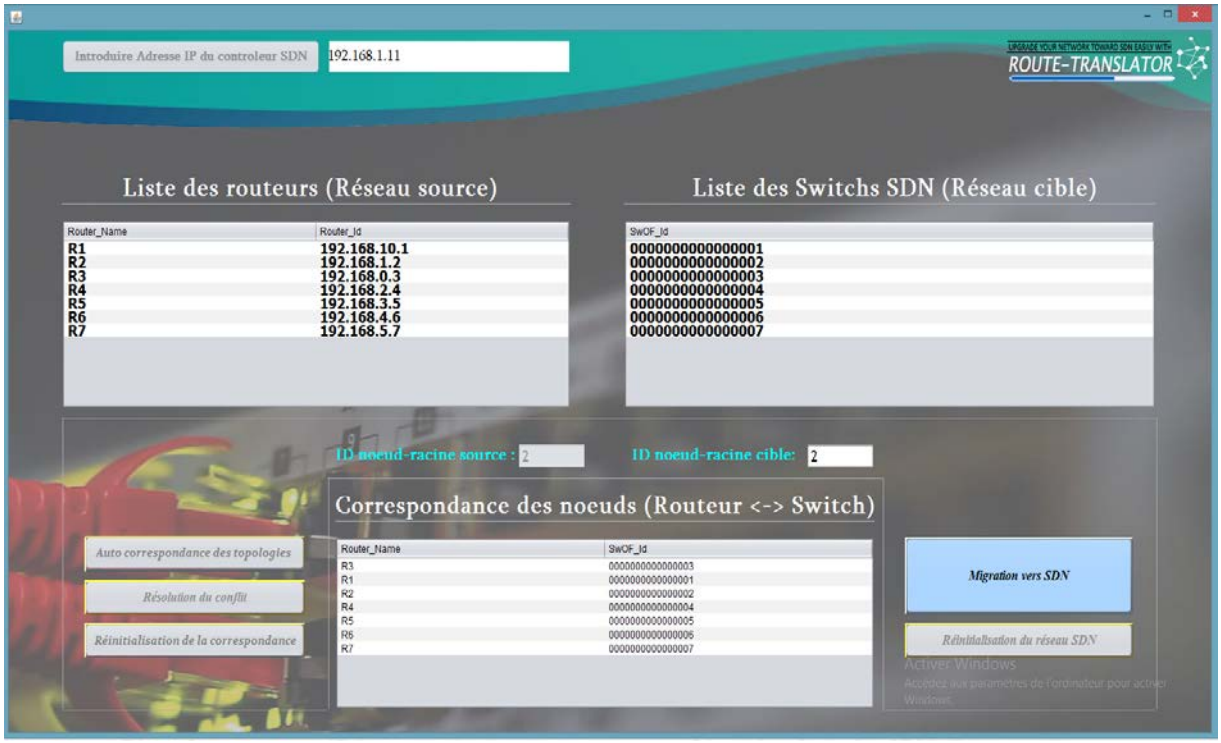

Figure 3.23 : Activation du bouton « Migrer vers le SDN »

Si la correspondance des deux topologies réussit, l'administrateur du réseau lance la migration vers le réseau SDN. Le résultat de cette opération est la configuration de l'adressage et du routage IP au niveau de tout le réseau SDN, assurant ainsi la connectivité entre les différents équipements du réseau (voir **la Figure 3.24**).

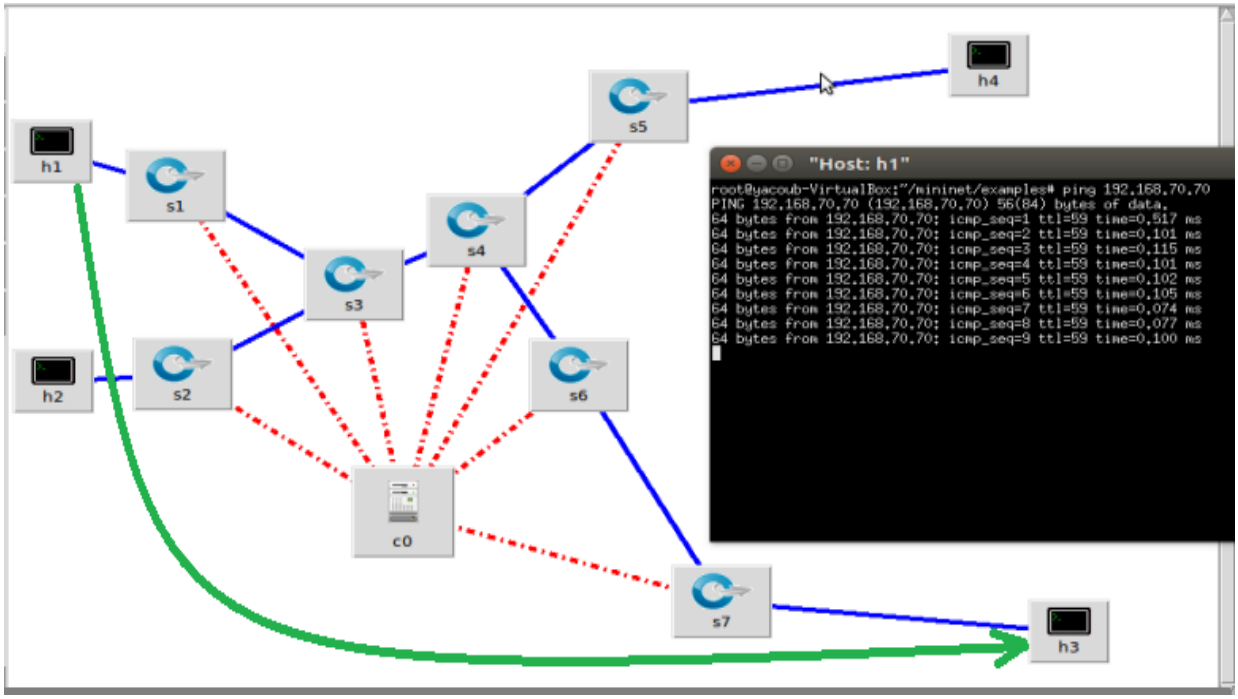

Figure 3.24: Confirmation du test de connectivité dans le réseau SDN après la migration

Conclusion

Dans ce chapitre nous avons présenté une conception détaillée de notre projet à travers un ensemble de diagramme UML, modélisant le comportement de Route Translator**.** Cette phase nous a permis de concrétiser la partie importante de l'étude théorique par un logiciel pouvant être pratiquement exploité.

Vu la restriction des délais, certains taches optionnelles, mais couteuses en matière de temps, non pas pu être achevées ouvrant la possibilité d'une extension d'un autre projet afin d'aboutir à une solution pouvant être commercialisée, répondant aux besoins des administrateurs et assurant une transition fiable des réseaux traditionnels vers les réseaux SDN.

Conclusion générale

Conclusion générale

L'objectif de notre projet était la réalisation d'un système permettant d'assister les administrateurs dans la tache de migration des réseaux traditionnels existants (réseau d'entreprise ou d'université) vers des réseaux SDN, afin de bien profiter de cette nouvelle technologie qui offre un niveau de contrôle personnalisable et centralisé.

Notre application ***Route-Translator*** a été conçue afin de répondre aux besoins des administrateurs quelque soit la taille de la topologie réseau, c'est-à-dire qu'elle couvre les réseaux de petite taille ou bien les réseaux déployés à large échelle.

Route-Translator permet la possibilité de migrer progressivement le réseau traditionnel vers le réseau SDN selon le besoin du conseil d'administration ou bien selon le budget dédié à cette opération (achat des équipements SDN), tout en minimisant au maximum le taux d'intervention humaine dans le processus de migration.

Dans le contexte de ce projet nous avons travaillé sur la migration du niveau trois c'est-à-dire le plan d'adressage et le plan de routage. Vu l'importance et la complexité de ce travail nous avons pu assurer le fonctionnement de base du processus de migration aboutissant à une connectivité garantie entre les différents équipements du réseau SDN. Nous projetons dans le futur à développer une solution plus complète en couvrant les taches optionnelles (non encore achevées dans Route-Translator) et en plus travailler sur la migration du niveau deux incluant les concepts suivants : VLAN, domaines de diffusion et la gestion de la redondance.

Liste des abreviations

SDN: Software Defined Networking

OSPF: Open Shortest Path First

BGP: Border Gateway Protocol

STP : Spanning Tree Protocol

FIB: Forwarding Information Base

CAM : Contenent Addressable Memory

ARP : Address Resolution Protocol

SSH: Secure Shell

SNMP : Simple Network Management Protocol

REST API : REpresentational State Transfer

CDP : Cisco Discovery Protocol

LLDP: Link Layer Discovery Protocol

ICMP : Internet Control Message Protocol

MIB : Management Information Base

NMS : Network Management Station

OFDP : OpenFlow Discovery Protocol

JDK : Java Développent Kit

UML : Unified Modeling Language

ODBC : Open DataBase Connectivit

References bibliographiques

[1] D. Jérome, 'Le SDN pour les nuls', Cisco Systems, 11 rue Camille Desmoulins, 92130 Issy-les-Moulineaux [Document en ligne] (2015,Déc,15) Dispoinble : https://gblogs.cisco.com/fr/uncategorized/le-sdn-pour-les-nuls/ [Accédé Mars. 2019]

[2] Simple Network Management Protocol (SNMP). Disponible: http://searchnetworking.teletarget.com/definition/SNMP [Accédé Fév. 2019]

[3] Border Gateway Protocol (BGP). Disponible: https://www.cisco.com/c/en/us/products/ios-nx-os-software/border-gateway-protocol-bgp/index.html [Accédé Mars. 2019]

[4] Spaning Tree Protocol (STP). Disponible: https://www.cisco.com/c/fr_ca/support/docs/lan-switching/spanning-tree-protocol/5234-5.pdf [Accédé Mars. 2019]

[5] Le but de SDN. Disponible : https://www.lemagit.fr/definition/SDN-Software-Defined-Networking [Accédé Mars. 2019]

[6] Le protocole SSH. Disponible: http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-fr-4/ch-ssh.html [Accédé Avril. 2019]

[7] Le Protocole OpenFlow. Disponible : https://www.randco/OpenFlow [Accédé Avril. 2019]

[8] Le Protocole OpenFlow dans l'architecture SDN. Disponible: http://supinfo.com/article/single/1010-protocole-openflow [Accédé Avril. 2019]

[9] La génese d'OpenFlow. Disponible : http://www.efort.com/r_tutoriels/OPENFLOW_EFORT [Accédé Avril. 2019]

[10] Les controleurs SDN. Disponible : https://searchnetworking.techtarget.com/definition/SDN-controller-software-defined networking-controller [Accédé Mai. 2019]

[11] Interface SouthBound. Disponible : https://whatis.techtarget.com/fr/definition/interface-northbound-southbound [Accédé Mai. 2019]

[12] F .Benamran, 'Etude des performances des architectures du plan de contrôle des réseaux Software-Defined Networks', Université Mohammed IV- Rabat ,N° d'ordre 29521, (janvier 2017).

[13] Sridhar Rao, SDN Series Part Three: NOX, the Original OpenFlow Controller 15 dec 2014. Disponible : https://thenewstack.io/sdn-series-part-iii-nox-the-original-openflow-controller [Accédé Mai. 2019]

[14] Pox Controller Tutorial. Disponible : http://sdnhub.org/tutorials/pox [Accédé Mai. 2019]

[15] David E.The Beacon OpenFlow Controller. Disponible http://yuba.stanford.edu/~derickso/docs/hotsdn15-erickson.pdf [Accédé Mai. 2019]

[16] Flooflight Controller. Disponible : http://www.projectfloodlight.org/floodlight [Accédé Mai. 2019]

[17] OpenDaylight. Disponible: https://www.opendaylight.org [Accédé Mai. 2019]

[18] Ryu Controller. Disponible: https://ryu.readthedocs.io/en/latest/getting_started.html [Accédé Mai. 2019]

[19] Isaku Yamahata. Ryu:SDN framwork and Python experience. 2013,Sep 14. Disponible https://www.slideshare.net/yamahata/ryu-sdnframeworkupload [Accédé Mai. 2019]

[20] Get Started With Mininet. Disponible: http://mininet.org/download [Accédé Mai. 2019]

[21] NS3- Network Simulator. Disponible: https://www.nsnam.org [Accédé Mai. 2019]

[22] Estinet. Disponible :https://www.estinet.com/ns [Accédé Mai. 2019]

[23] OMNET ++ Discrete Event Simulator. Disponible: https://omnetpp.org [Accédé Mai. 2019]

[24] OpenFlow Switch Fondation March 26, 2015 . Disponible : https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.3.5.pdf [Accédé Mai. 2019]

[25] NetConf/Yang. Disponible: https://wapiti.telecom-lille.fr/commun/ens/peda/options/st/rio/pub/exposes/exposesser2010-ttnfa2011/lelaouenan-zuccarelli/intro.html [Accédé Mai. 2019]

[26] REST APIs in SDN: An introduction for network engineers. Disponible : https://searchnetworking.techtarget.com/tip/REST-APIs-in-SDN-An-introduction-for-network-engineers [Accédé Mai. 2019]

[27] Opflex. Disponible : https://ipwithease.com/opflex [Accédé Mai. 2019]

[28] H. BOUIDA, 'Etude et mise en oeuvre d'une solution SDN : Application de Gestion de VLANs' , (2017, Janvier,18) Faculté des Sciences, 4 Avenue Ibn Battouta B.P. 1014 RP, Rabat – Maroc. Disponible : http://dspace.univtlemcen.dz/bitstream/112/12600/1/Etude_mise_en_oeuvre_solution_SDN.pdf

[**29]** 7 Advantages of Software Defined Networking. 2017,Aug ,8. Disponible : https://imaginenext.ingrammicro.com/data-center/7-advantages-of-software-defined-networking [Accédé Mai. 2019]

[30] Open Networking Foundation ,SDN migration considerations and use cases, (2014,Nov,21) 2275 E. Bayshore Road, Suite 103, Palo Alto, CA 94303 Disponible : www.opennetworking.org

[31] Mohammed A, Mohammed, Mohammed O and Mohammed AL-Hassan A. Research Submitted in Partial fulfillment for the Requirements of the Degree of B.Sc. (Honors) in Electronics Engineering 'Implementation of SDN in a Campus NAC Use Case' (2016 Oct) .

[32] Inside Google's Software-Defined Networking. Disponible : https://www.networkcomputing.com/networking/inside-googles-software-defined-network [Accédé Mai. 2019]

[33] M. Belhain, Auto-Découverte des topologies réseaux. Mémoire master en informatique (Option STIC), Centre universitaire Mila, 2015

[34] A. Azzouni, T. Nguyen and R. Boutaba. Limitations of OpenFlow Topology Discovery Protocol. 2012,Mai,10. LIP6 / UPMC ; Paris , France. Université de waterloo ; CANADA Disponible : https://arxiv.org/pdf/1705.00706.pdf [Accédé Mai. 2019]

[35] Unified Modelling language. Disponible : https://www.uml.org [Accédé Juin. 2019]

[36] R. Pascal, 'UML 2 par la pratique', N° :11770,2006,236. N° d'editeur : 7280 (Aout 2006). France, Pascale

[37] Unified Process. Disponible : https://sabricole.developpez.com/uml/tutoriel/unifiedProcess [Accédé Juin. 2019]

[38] Olivier, G. (Edition 2005-2006). Introduction à la modélisation orientée objets avec UML

[39] G. Joseph, G. David. (2008), 'UML 2 Analyse et conception' ISBN 978-2-10-053567-5 Université de Paris-Dauphine.

[40] Virtual Box. Disponible : https://www.computerhope.com/jargon/v/virtualbox.htm [Accédé Aout. 2019]

[41] Netbeans. Disponible :https://www.techno-science.net/definition/5346.html [Accédé Aout. 2019]

[42] Xampp Apache. Disponible : https://www.apachefriends.org/fr/index.html [Accédé Aout. 2019]

[43] Jean Michel DOUDOUX .Développons en Java. Version 2.20. (2019, Jan, 28) Disponible : https// :www.jmdoudoux.fr

[44] Java SE Development Kit 8 Downloads. Disponible : https://www.oracle.com/technetwork/java/jdk8-downloads-2133151.html [Accédé Aout. 2019]

[45] JDBC Drivers. Disponible : https://www.geeksforgeeks.org/jdbc-drivers [Accédé Aout. 2019]

[46] Introducing JSON. Disponible : https://www.json.org [Accédé Aout. 2019]