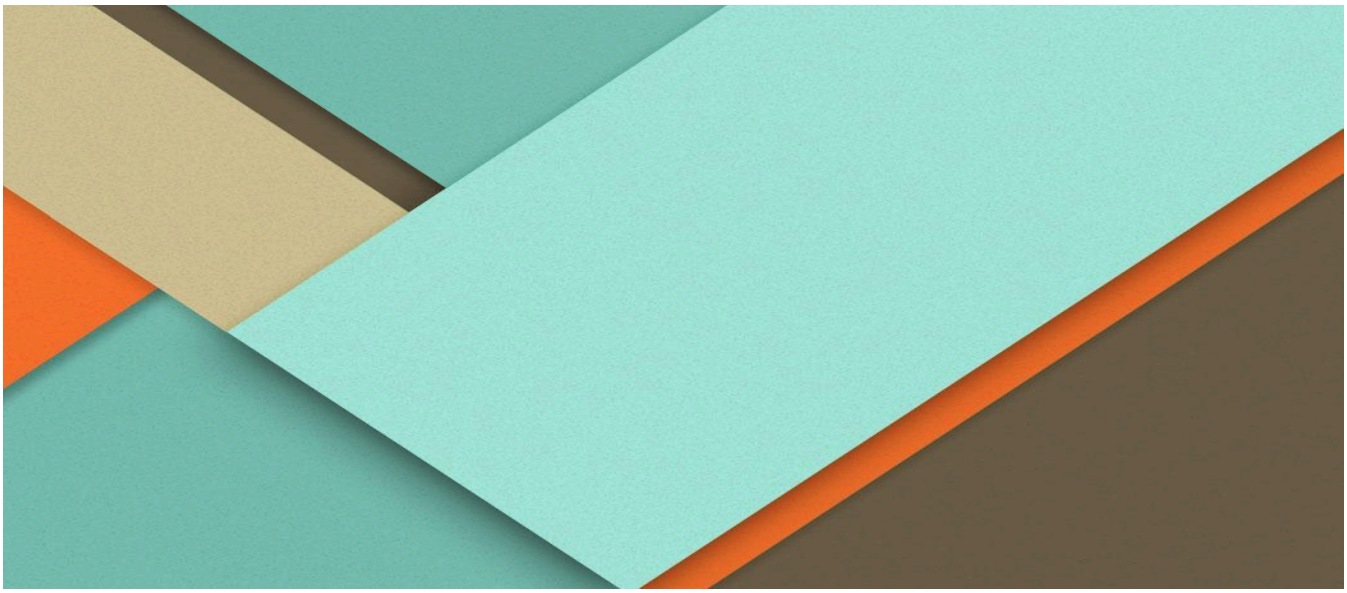


Projet d'Administration réseaux:

IDENTIFICATION INTELLIGENTE DU CHEMIN OPTIMAL DANS UN RÉSEAU MAILLÉ SANS FILS

Membres du groupe 14.3:

KEUGUE Wilson

CHEDJOU Valdes

DONFACK Hilary

SANGOU Mohamed

GAVLI Dominique

Coordinateurs

Pr. Thomas DJOTIO

Mme. Judith Nouho

INTRODUCTION

Dans le monde actuel, où les réseaux de communication jouent un rôle crucial, l'optimisation et la durabilité des infrastructures sont devenues des priorités majeures. La recherche de chemins optimaux, la définition de règles pour les nœuds et l'évaluation de la durée de vie du réseau et des nœuds sont des éléments essentiels pour assurer une connectivité fiable et efficace. Ces défis techniques ont

également des implications stratégiques, car ils influencent directement la performance des réseaux, la gestion des ressources et l'expérience des utilisateurs. La problématique centrale de ce projet est de trouver des solutions pour optimiser les performances des réseaux de communication. Il s'agit notamment de développer un algorithme intelligent pour identifier les chemins les plus efficaces, de définir des règles de nœuds efficaces pour optimiser la connectivité du réseau, et d'évaluer la durée de vie du réseau et des nœuds afin de garantir la durabilité et la fiabilité du système. Il est également crucial de maximiser l'efficacité énergétique et la gestion des ressources pour réduire l'impact environnemental et favoriser une utilisation durable des infrastructures. Pour résoudre cette problématique, une approche combinant la modélisation mathématique et la mise en pratique sera utilisée. La formalisation mathématique permettra de définir les problèmes d'optimisation du chemin, de gestion des nœuds et d'évaluation de la durée de vie du réseau et des nœuds. En utilisant des techniques d'optimisation , des algorithmes personnalisés pourront être développés pour trouver des solutions efficaces. Ces solutions seront ensuite mises en œuvre et testées dans des environnements réels en utilisant des émulateurs de réseaux sans fil et des frameworks SDN. Cette approche analytique et pratique permettra de combiner les avantages des deux méthodologies pour obtenir des résultats concrets et applicables.

I. CADRE DU PROJET

1. Contexte

Le contexte de ce projet se base sur la nécessité croissante de repenser la gestion des réseaux dans les environnements urbains modernes, en intégrant des technologies innovantes. Avec l'évolution rapide de nos villes, la gestion des réseaux, en particulier dans le contexte des réseaux maillés sans fil programmables, requiert des solutions plus efficaces et intelligentes. Actuellement, les méthodes de routage et d'installation des nœuds ne répondent pas toujours de manière optimale aux besoins évolutifs des

zones urbaines, engendrant des défis en termes d'efficacité opérationnelle, de gestion des ressources, et d'impact environnemental. Dans ce contexte, le projet de routage multi-chemin intelligent se positionne comme une réponse innovante à ces défis.

2. Problématique

Les réseaux maillés sont des réseaux sans fil dans lesquels les nœuds sont interconnectés entre eux, formant une structure de maille. Ces réseaux présentent de nombreux avantages, tels que la résilience, la flexibilité et la scalabilité. Cependant, ils présentent également certains défis, notamment :

- La consommation énergétique : Les nœuds d'un réseau maillé doivent être alimentés par des batteries, ce qui limite leur autonomie. La consommation énergétique des nœuds est un facteur important à prendre en compte lors de la conception et de l'exploitation d'un réseau maillé.
- La congestion : Les nœuds d'un réseau maillé peuvent être surchargés, ce qui peut entraîner une dégradation des performances. La congestion est un problème particulièrement important dans les réseaux maillés qui transportent un trafic important.
- La sécurité : Les réseaux maillés sont plus vulnérables aux attaques que les réseaux traditionnels, car ils ne disposent pas d'une architecture centralisée. La sécurité est un problème important à prendre en compte lors de la conception et de l'exploitation d'un réseau maillé.

La problématique de ce projet est donc de trouver une solution efficace pour concilier les exigences en matière de performance et de fiabilité des flux de données, avec les contraintes énergétiques des nœuds.

3. Objectifs

- Développer un algorithme d'identification intelligent du chemin optimal pour les réseaux.
- Mettre en place des règles de nœuds efficaces pour optimiser la connectivité du réseau.

- Évaluer la durée de vie du réseau et des nœuds pour améliorer la durabilité et la fiabilité du système.
- Proposer des solutions pour maximiser l'efficacité et la gestion des ressources dans le réseau.

II. ETUDE DU CONCEPT SDN (Software Defined Network)

Il n'est pas possible de parler de SDN sans se tourner vers l'organisation ONF (Open Networking Foundation) qui porte, aujourd'hui, le concept « Open SDN » d'origine ayant été connu comme étant associé au protocole OpenFlow. Leur message est clair : « Transforming Networks into Agile Platforms for Service Delivery ». Lorsqu'on analyse leur définition, on retrouve la description des trois plans d'un équipement réseau avec le fait de séparer physiquement le « Control Plane » du « Data Plane », en poursuivant l'objectif d'offrir un « Control Plane » commun ayant une vue centralisée du réseau afin d'apporter une synchronisation adaptée.

La volonté de désagrégation des éléments réseau a entraîné plusieurs conséquences. La première a été de devoir structurer le modèle en couches autour du contrôleur SDN.

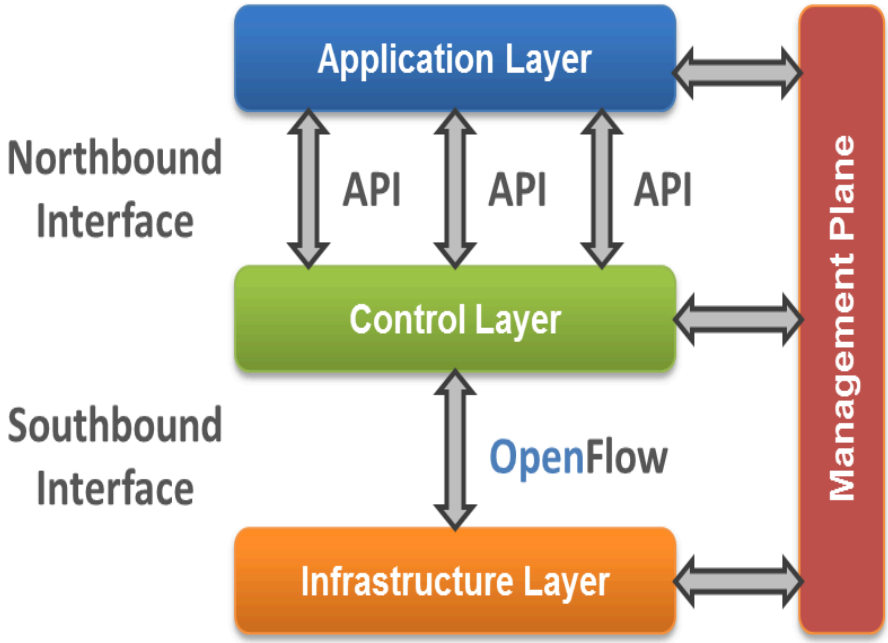


Figure 1: Architecture SDN

Des implémentations, sous forme de produits Open Source, se sont multipliées pour donner aussi naissance à des offres commerciales . Dans les produits Open Source, on peut citer le contrôleur SDN le plus connu ODL (OpenDaylight), mais aussi ONOS le système d'exploitation orienté SDN, Floodlight essentiellement porté par des ingénieurs de Big Switch Networks, OpenContrail et la librairie Python RYU utilisée, entre autre, par la NSA. Beaucoup d'autres noms sont disponibles dans la littérature, mais leur maintien à jour est douteux face à ceux qui se sont implantés dans le domaine.

La couche « Control Layer » ou « Control Plane » représente le contrôleur SDN et elle communique avec le matériel de la couche « Infrastructure Layer » ou « Data Plane » au travers d'une interface appelée « Southbound APIs ». Le protocole OpenFlow n'est qu'un exemple d'implémentation d'une interface « Southbound » et son objectif n'est en aucun cas d'effectuer des configurations sur les matériels, mais uniquement de modifier la table des flux. C'est pour cette raison que des protocoles comme NETCONF, BGP, OVSDB, SNMP, mais aussi CLI et bien d'autres sont utilisables en interface «Southbound ». Le projet Open Data Plane propose des APIs Open Source pour interagir de façon transparente, avec le « Data Plane » de différents matériels.

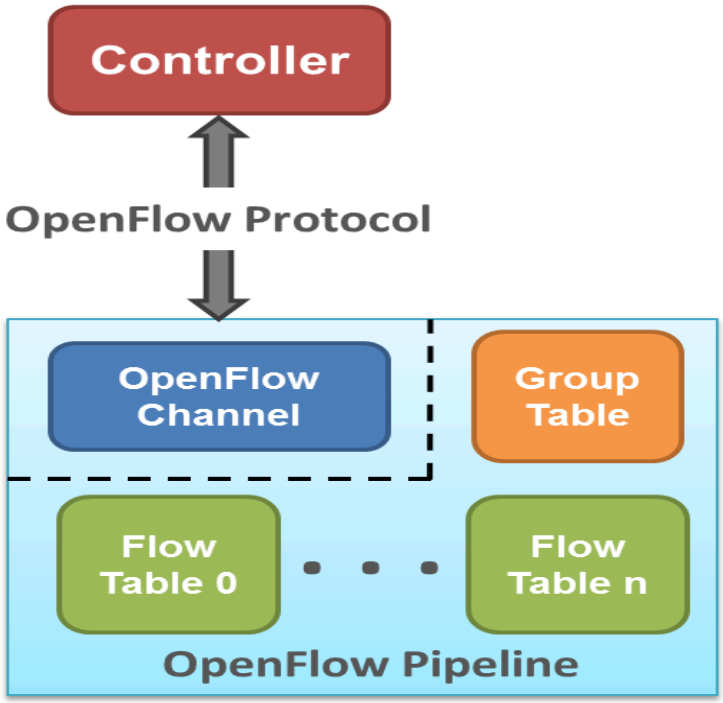


Figure 2: communication entre les couches Control Layer et Infrastructure Layer

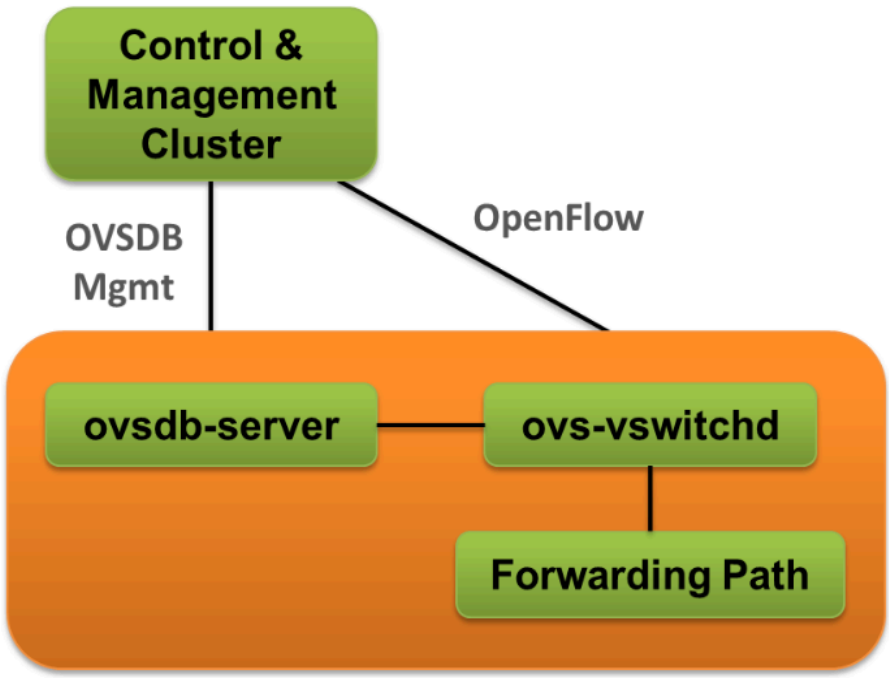
Dans la littérature réseau, la FIB (Forwarding Information Base) porte souvent le nom de « Forwarding Table » ou « MAC Table » en référence aux adresses Ethernet MAC, ou encore « CAM Table » en référence à la mémoire CAM (Content-Addressable Memory). Sans l'utilisation d'une table, un switch se comporterait comme un hub Ethernet et enverrait l'ensemble des trames sur tous les ports. Parfois, on peut faire la distinction entre les éléments mis en œuvre pour traiter le niveau 2 du modèle OSI (FIB) et ceux utilisés pour prendre des décisions au niveau 3 (RIB ou Routing Information Base), mais quoi qu'il en soit admettre le vocabulaire suivant :

- MAC Switching, pour lequel la décision de transmission est prise sur l'adresse MAC
- IP Routing, pour lequel la décision de transmission est prise sur l'adresse IP

- Flow Forwarding, pour lequel la décision de transmission est prise, de façon réactive ou proactive, sur le flux

A titre d'illustration, l'étude de l'architecture Open vSwitch permet de comprendre le positionnement du protocole OpenFlow pour la mise à jour de la table des flux et du protocole OVSDB, décrit dans le RFC 7047, pour la configuration du commutateur virtuel.

Figure 3: communication entre les couches Control Layer et Application Layer



La couche « Control Layer » communique avec la couche « Application Layer » ou « Application Plane » au travers d'une interface appelée « Northbound APIs ». Les APIs REST (Representational State Transfer) sont généralement utilisées pour implémenter l'interface « Northbound » et permettre le développement de modules complémentaires qui viennent enrichir les fonctionnalités du contrôleur.

Pour information, les interfaces « Eastbound » et « Westbound » sont utilisées pour les communications inter-contrôleurs.

Mais le fait est que, comme souvent et malgré une adoption par de gros acteurs, la révolution annoncée autour du protocole OpenFlow n'a pas eu lieu. Au-delà de l'idée de départ, on trouve maintenant de nombreuses technologies qui tournent, de façon plus ou moins liée, autour du « Software-Defined Networking », et qui chacune revendique l'appellation de SDN. Un bon exemple est la notion d'automatisation qui se réclame d'être une forme de programmation du réseau par le logiciel.

II. REVUE DES ALGORITHMES D'OPTIMISATION DE CHEMIN

1. Algorithmes basés sur la distance

Dijkstra

L'algorithme de Dijkstra est un algorithme de recherche de chemin le plus court dans un graphe pondéré, c'est-à-dire un réseau de nœuds connectés par des arêtes ayant des poids ou des coûts associés. L'algorithme détermine le chemin le plus court entre un nœud de départ et tous les autres nœuds du graphe.

En théorie des graphes, l'algorithme de Dijkstra sert à résoudre le problème du plus court chemin. Plus précisément, il calcule des plus courts chemins à partir d'une source vers tous les autres sommets dans un graphe orienté pondéré par des réels positifs. On peut aussi l'utiliser pour calculer un plus court chemin entre un sommet de départ et un sommet d'arrivée.

AVANTAGES	INCONVÉNIENTS

- **Optimalité** : garantit de trouver le chemin le plus court, à condition que le graphe ne contient pas de poids d'arête négative.	- **Complexité temporelle** : Sa complexité temporelle est relativement élevée
- **Efficacité sur de petits graphes**	- **Inefficacité en présence de poids négatifs**
- **Simplicité**	- **Requiert une structure de données appropriée**
- **Large Applicabilité**	- **Pas toujours adapté aux graphes dynamiques**

2. Algorithmes basés sur l'état du lien

Bellman-Ford

L'algorithme de Bellman-Ford, aussi appelé algorithme de Bellman–Ford–Moore, est un algorithme qui calcule des plus courts chemins depuis un sommet source donné dans un graphe orienté pondéré. Contrairement à l'algorithme de Dijkstra, l'algorithme de Bellman-Ford autorise la présence de certains arcs de poids négatif et permet de détecter l'existence d'un circuit absorbant, c'est-à-dire de poids total strictement négatif, accessible depuis le sommet source.

AVANTAGES	INCONVÉNIENTS
- **Gestion des poids négatifs**	- **Complexité temporelle** : La complexité temporelle de l'algorithme de Bellman-Ford est relativement élevée
- **Détection des cycles de poids négatif**	- **Moins efficace pour des graphes denses**
- **Polyvalence** : Il peut être appliqué à des graphes dirigés ou non dirigés, avec ou sans poids négatifs.	- **Non-optimal en l'absence de poids négatifs**
- **Facilité d'implémentation** : Il est relativement simple à mettre en œuvre par rapport à certains autres algorithmes de plus grande complexité.	- **Espace mémoire** : Il nécessite de maintenir des structures de données supplémentaires

3. Algorithmes basés sur la programmation linéaire

En optimisation mathématique, un problème d'optimisation linéaire demande de minimiser une fonction linéaire sur un polyèdre convexe. La fonction que l'on minimise ainsi que les contraintes sont décrites par des fonctions linéaires, d'où le nom donné à ces problèmes. L'optimisation linéaire (OL) est la discipline qui étudie ces problèmes. Elle est également désignée par le nom de programmation linéaire,d'optimisation linéaire.

AVANTAGES	INCONVÉNIENTS
- **Formulent le problème d'identification du chemin optimal comme un problème d'optimisation linéaire**	- **Complexité temporelle**
- **Solutions efficaces**	- **Sensibilité aux données**
- **Théorie de la dualité**	- **Problèmes non linéaires**
- **Capacité à modéliser des problèmes complexes**	- **Limitations de modélisation et Absence de solutions optimales globales**

4. Algorithmes bio-inspirés

Les algorithmes bio-inspirés trouvent des applications dans tous les domaines : de l'ingénierie au transport en passant par l'urbanisme et l'énergie. Que ce soit en s'inspirant des mécanismes ou des comportements du vivant, le nombre de possibilités est très élevé.

AVANTAGES	INCONVÉNIENTS
- **Adaptabilité**	- **Difficile à concevoir et à comprendre**
- **Recherche globale**	- **Paramétrage délicat**
- **Parallélisme**	- **Complexité temporelle**
- **Capacité à traiter des problèmes complexes**	- **Manque de garantie de convergence et Sensibilité aux conditions initiales**

II. MÉTHODOLOGIE

Le réseau que nous considérons est un réseau maillé sans fil programmable destiné à supporter une variété de services de communication. La topologie du réseau est cruciale pour déterminer la manière dont les nœuds peuvent communiquer entre eux, formant ainsi la base de notre projet de routage multi-chemins intelligent.

1-Description du Réseau

Topologie
Nous avons choisi une topologie maillée pour offrir une grande flexibilité dans la communication entre les nœuds. Cette topologie permet une redondance des chemins, ce qui est essentiel pour notre objectif de routage multi-chemins intelligent.

Support de Transmission
Le réseau utilise principalement des ondes radio pour la transmission des données. Cela offre une mobilité aux nœuds, mais nécessite une gestion intelligente de la puissance de transmission pour optimiser l'énergie des nœuds.

Protocoles de Communication
Nous avons opté pour les protocoles de communication standards tels que TCP/IP, HTTP, FTP et SMTP pour assurer une interopérabilité avec d'autres réseaux.

Sécurité
La sécurité est intégrée à la conception du réseau. Des mécanismes de chiffrement et d'authentification sont mis en œuvre pour protéger les données sensibles.

Performance
La conception du réseau vise à fournir des performances optimales, en tenant compte des délais de transmission et des débits.

Maintenance
Un plan de maintenance régulière est établi pour garantir l'efficacité à long terme du réseau. Les mises à jour régulières comprennent des correctifs de sécurité et des améliorations de performance

2-Paramètres et Contraintes pour le Choix du Chemin Optimal

➢ Définition des Paramètres

-Topologie du réseau : la topologie du réseau détermine le nombre de chemins possibles entre deux points.
-Capacité des liens : la capacité des liens détermine la quantité de données qui peuvent être transmises sur un lien.
-Latence des liens : la latence des liens détermine le temps nécessaire pour transmettre des données sur un lien.
-Fiabilité des liens : la fiabilité des liens détermine la probabilité qu'un lien ne soit pas disponible pour la transmission de données.

➢ Contraintes pour le Choix du Chemin Optimal

Le choix du chemin optimal est influencé par plusieurs paramètres et contraintes qui dépendent du type de service. Dans notre contexte, les contraintes spécifiques à chaque type de service sont les suivantes :

Contraintes de Bande Passante :
Chaque type de service nécessite une bande passante spécifique pour garantir des performances optimales. La somme de la bande passante allouée aux liens du chemin doit être supérieure ou égale à la bande passante requise par le flux.

Contraintes de Taux de Perte :
Les flux doivent respecter des taux de perte acceptables pour assurer une qualité de service adéquate. Le taux de perte du flux sur chaque lien doit être inférieur ou égal au taux de perte maximal admissible.

Contraintes d'Énergie Résiduelle :
Chaque nœud doit maintenir une énergie résiduelle suffisante. L'énergie résiduelle de chaque nœud après avoir relayé le flux doit être supérieure ou égale à l'énergie résiduelle minimale requise.

3-Algorithme d'Identification du Chemin Optimal basé sur les PLNE

Il s’agit de trouver le chemin optimal entre deux points en minimisant une fonction objective qui tient compte des contraintes de bande passante, de taux de perte et d'énergie résiduelle.

L'algorithme fonctionne comme suit :

-Il construit un modèle mathématique du problème de choix du chemin optimal. Ce modèle est un programme linéaire en nombres entiers, où les variables représentent les flux de données sur les liens du réseau.
-Il résout le programme linéaire en nombres entiers.
-La solution du programme linéaire en nombres entiers est le chemin optimal.

Étapes de l'Algorithme :

- ➢ Modélisation en PLNE :

Formuler le problème comme un Programme Linéaire en Nombres Entiers (PLNE) en définissant les variables de décision, la fonction objectif, et les contraintes.

- ➢ Définition des Variables :

Définir des variables binaires représentant le choix de chaque lien dans le chemin.

- ➢ Fonction Objectif :

Minimiser la fonction objective qui combine les distances pondérées des liens en fonction des critères de bande passante, de taux de perte et d'énergie résiduelle.

- ➢ Contraintes de Bande Passante :

Assurer que la somme des bandes passantes allouées est supérieure ou égale à la bande passante requise.

- ➢ Contraintes de Taux de Perte :

Limiter le taux de perte sur chaque lien à une valeur admissible.
Contraintes d'Énergie Résiduelle :

Garantir que l'énergie résiduelle après le relais du flux est supérieure ou égale à l'énergie minimale requise.

➢ Résolution du PLNE :

Utiliser des bibliothèques telles que PuLP pour résoudre le PLNE et obtenir la solution optimale.

4-Évaluation de la Performance

-Le coût total est évalué par la valeur optimale Z obtenue après la résolution du PLNE.
-Le temps de Calcul est mesuré en chronométrant la résolution du PLNE.
-La satisfaction des contraintes est vérifiée en s'assurant que toutes les contraintes sont respectées dans la solution optimale.

Scénarios d'Évaluation :

- **Optimisation de Bande Passante** :

Flux nécessitant une bande passante maximale.
Évaluer comment l'algorithme optimise la bande passante dans le chemin.

- **Taux de Perte Critique** :

Flux avec une tolérance très faible au taux de perte.
Vérifier la capacité de l'algorithme à minimiser le taux de perte.

- **Économie d'Énergie** :

Flux nécessitant une économie d'énergie maximale.
Examiner comment l'algorithme choisit les nœuds pour minimiser la consommation d'énergie.

- **Combinaison des Critères** :

Flux avec des exigences équilibrées de bande passante, de taux de perte et d'énergie.
Évaluer la capacité de l'algorithme à trouver un compromis optimal.

L'évaluation de la performance se concentrera sur la capacité de l'algorithme à fournir des solutions optimales tout en respectant les contraintes spécifiques à chaque type de service.

IV. MODÉLISATION

1. Présentation

Dans cette section, nous proposons une modélisation mathématique et algorithmique avancée, basée sur la programmation linéaire, offrant une approche puissante pour optimiser intelligemment les flux de données dans les réseaux, permettant ainsi une gestion optimisée et une performance accrue.

On suppose qu'on a une liste de chemins possible pour un flux de données. Nous devons identifier le chemin optimal parmi ces options en fonction du type de service du flux. Le chemin optimal sera celui qui optimise l'utilisation des ressources. Pour cela, on définit un réseau de nœuds interconnectés par des liens, et un flux de données associé à un type de service donné. Soit P l'ensemble des chemins possibles pour le flux. L'objectif est de trouver le chemin $x \in P$ qui maximise la fonction objective du type de service du flux, tout en respectant les contraintes énergétiques des nœuds.

- **Pour les services à temps réel**, notre objectif est de maximiser la bande passante tout en minimisant le taux de perte de données. Ainsi, nous allons définir une fonction objective qui cherche à maximiser la bande passante et à minimiser le taux de perte de données pour le chemin x sélectionné parmi P.

- **Pour les services dédiés**, notre objectif est de maximiser la bande passante sans se soucier du taux de perte de données. Nous allons donc définir une fonction objective qui vise uniquement à maximiser la bande passante pour le chemin x sélectionné parmi P.

- **Pour les services partagés et les autres services**, notre objectif est de minimiser la consommation d'énergie des nœuds. Nous définirons une fonction objective qui cherche à minimiser la consommation d'énergie pour le chemin x sélectionné parmi P, tout en respectant les contraintes énergétiques des nœuds.

Nous allons utiliser la structure d'un algorithme, car elle offre une approche systématique et logique pour formaliser mathématiquement un problème et sa solution.

En suivant cette structure, nous pouvons décomposer le problème en étapes clairement définies, ce qui facilite la compréhension, la communication et l'implémentation de la solution.

L'algorithme nous permettra de clarifier les objectifs, les contraintes et les variables en jeu. Nous pourrons ainsi formuler de manière précise les différentes fonctions objectives pour chaque type de service du flux de données, tout en prenant en compte les contraintes énergétiques des nœuds.

En utilisant cette approche, nous pourrons définir les étapes nécessaires pour trouver le chemin optimal parmi les options disponibles, en maximisant l'utilisation des ressources du réseau et en respectant les objectifs spécifiques à chaque type de service. Cela nous permettra de prendre des décisions éclairées et de parvenir à une solution efficace et optimale.

2. constitution du modèle

Algorithme d'identification du chemin optimal selon le type de service

Entrées:

- Liste de chemins possibles pour un flux de données
- Type de service du flux

Sorties:

- Chemin optimal pour le flux de données

Définition des variables de décision:

- Pour les services partagés et les autres services :$x \in \{0,1\}^{n\times n}$, où $x_{ij} = 1$ si le lien (i, j) est inclus dans le chemin, et $x_{ij} = 0$ sinon.
- Pour les services dédiés : $b \in R^{n\times n}$, où b_{ij} est la bande passante allouée au lien (i, j) .

- Pour les services temps réel : $x = (b, p) \in R^{2\times n\times n}$, où b_{ij} est la bande passante allouée au lien (i, j) et p_{ij} est le taux de perte sur le lien (i, j).

Établissement des fonctions objectifs:

- Pour les services temps réel : $max \sum_{i,j} b_{ij} + \frac{1}{p_{ij}}$
- Pour les services partagés : $min \sum_{i,j} x_{ij}$
- Pour les services dédiés : $max \sum_{i,j} b_{ij}$

Énonciation des contraintes:

Les contraintes spécifiques à chaque type de service sont les suivantes :

- Contraintes de bande passante : la somme de la bande passante allouée aux liens du chemin doit être supérieure ou égale à la bande passante requise par le flux $\sum_{i,j} x_{ij}. b_{ij} \geq B$
- Contraintes de taux de perte : le taux de perte du flux sur chaque lien doit être inférieur ou égal au taux de perte maximal admissible $x_{ij}. p_{ij} \leq p_{ij}^{max}$

- Contraintes d'énergie résiduelle : l'énergie résiduelle de chaque nœud après avoir relayé le flux doit être supérieure ou égale à l'énergie résiduelle minimale requise $E_i - \sum_{j\in N(i)} e. b_{ij}. d_{ij} \geq E_i^{min}$

où :

- B est la bande passante requise par le flux

- b_{ij} est la bande passante du lien (i, j)

- d_{ij} est la distance entre les nœuds i et j

- p_{ij} est le taux de perte du lien (i, j)

- e est l'énergie/unité de bande passante*distance

- p_{ij}^{max} est le taux de perte maximal admissible

- E_i est l'énergie résiduelle du nœud i

- E_i^{min} est l'énergie résiduelle minimale requise du nœud i

- $N(i)$ est l'ensemble des nœuds adjacents au nœud i

3. mise en place de l'algorithme

Algorithme TrouverCheminOptimal(liste_chemins, type_service, B, pijmax, e, Eimin):

Supprimer tous les chemins ne respectant pas la condition

*min(Ei − somme(N(i) * e * bij * dij) pour tous les i) < Eimin*

if type_service = 'partagé':

Return chemin ayant le nombre de noeuds minimal:

if type_service = 'dédié':

for chemin in liste_chemins:

for i, j in chemin:

$$B\_P\_chemin = \sum_{i,j} b_{ij} + \frac{1}{p_{ij}}$$

add B_P_chemin in liste_B_P_chemin

Comparer les B_P_chemin dans liste_B_P_chemin

Return max(liste_B_P_chemin)

if type_service = 'temps réel':

for chemin in liste_chemins:

for i, j in chemin:

$$B\_chemin = \sum_{i,j} b_{ij}$$

add B_chemin in liste_B_chemin

Comparer les B_chemin dans liste_B_chemin

Return max(liste_B_chemin)

La formulation mathématique proposée offre une solution au problème de sélection de chemin dans un réseau de capteurs sans fil, en prenant en compte les contraintes énergétiques des nœuds. Cette formulation peut être adaptée pour différents types de services en modifiant la fonction objective selon les objectifs spécifiques visés. En utilisant cette formulation, nous pouvons trouver le chemin optimal qui minimise la consommation d'énergie tout en maximisant les performances du réseau. Cela permet de prendre des décisions informées et d'optimiser l'utilisation des ressources disponibles. Cette approche de modélisation mathématique fournit un cadre rigoureux pour résoudre le problème de sélection de chemin, offrant ainsi une solution efficace et adaptée aux différentes exigences des services dans un réseau de capteurs sans fil.

V. ANALYSE

1. Diagramme de Contexte

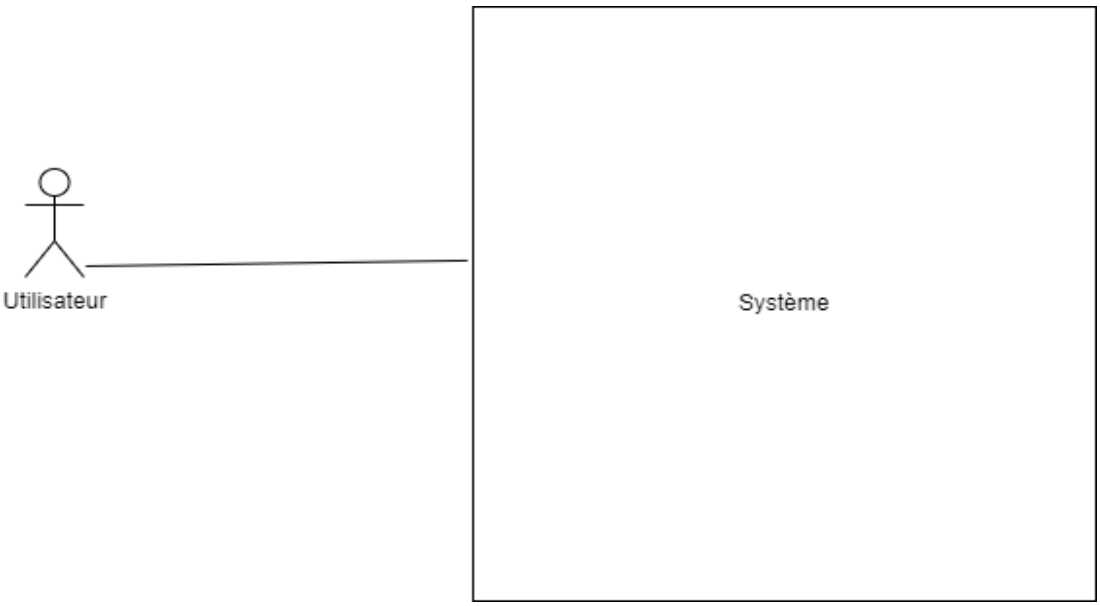


Figure 4 : Diagramme de contexte

2. Diagramme de Package

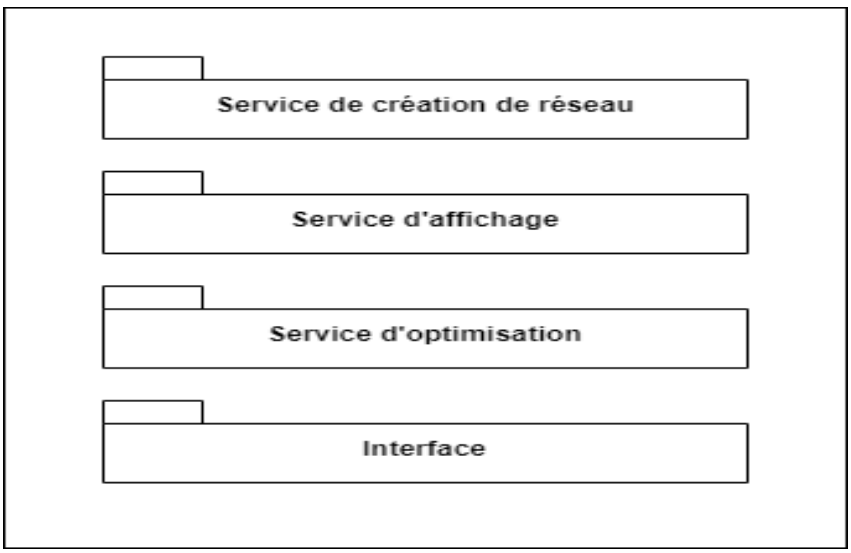


Figure 5 : Diagramme de Package

3. Diagramme de cas d'utilisation

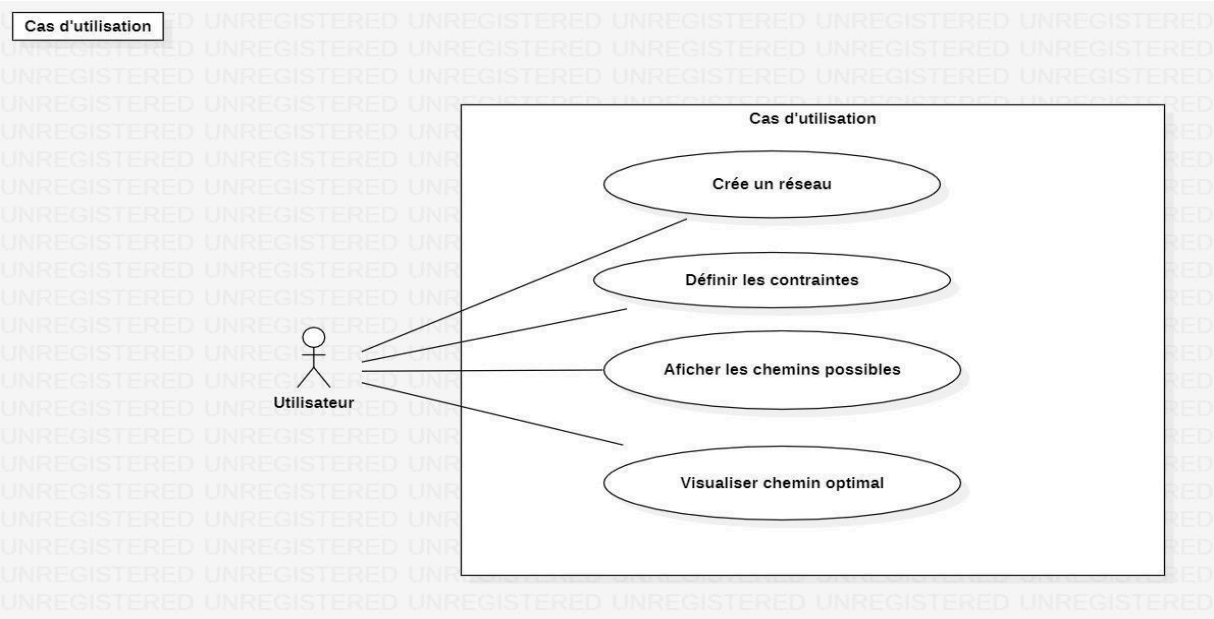


Figure 6: Diagramme de cas d'utilisation

Description textuelle:

➢ **Cas d'utilisation "Créer un réseau":**

Objectif: donner les informations sur la topologie du réseau et voir le réseau créé.

Acteur principal: L'utilisateur.

Précondition: entrer dans l'application

Scénario nominal: L'utilisateur clique sur le bouton "Créer" et une interface va s'ouvrir ou l'utilisateur remplit les informations concernant la topologie du réseau(nombre de nœuds, nombres de routeurs, …) et dès qu'il finit le remplissage, il clique sur un bouton "OK" et voit le réseau créé.

Scénario alternatif: Lors du remplissage des informations sur la topologie du réseau, en raison des limites de l'application, il ne pourrait pas créer le réseau, d'où dans ce cas l'apparition d'une erreur.

➢ **Cas d’utilisation “ Définir les Contraintes ”:**

Objectif: Attribuer des énergies résiduelles aux nœuds du réseau.

Acteur principal: L’utilisateur.

Précondition: Créer un réseau.

Scénario nominal: L'utilisateur accède à l'option de création de réseau. Il spécifie le nombre de nœuds, la topologie du réseau, les caractéristiques des liens et l’application génère le réseau.

➢ **Cas d’utilisation “Afficher Chemin Possible”:**

Objectif: Permettre à l'utilisateur de visualiser les chemins possibles dans le réseau maillé en fonction des contraintes définies.

Acteur principal: L’utilisateur.

Précondition: L'utilisateur est connecté à l'application et le réseau a été créé. Les contraintes ont été définies au préalable.

Scénario nominal: L'utilisateur sélectionne l'option pour afficher les chemins possibles. L'application utilise les algorithmes développés pour calculer et afficher les chemins possibles en tenant compte des contraintes définies.

Scénario alternatif: Si aucun chemin n'est trouvé en fonction des contraintes, l'application informe l'utilisateur de l'absence de chemin possible.

➢ **Cas d’utilisation “Afficher le Chemin Optimal”:**

Objectif: Permettre à l'utilisateur de visualiser le chemin optimal dans le réseau maillé en fonction des contraintes et du type de service.

Acteur principal: L'utilisateur.

Précondition: L'utilisateur est connecté à l'application et le réseau a été créé. Les contraintes ont été définies au préalable.

Scénario nominal: L'utilisateur choisit l'option pour afficher le chemin optimal.

L'application utilise les algorithmes développés pour le Sujet 3 pour calculer et présenter le chemin optimal en prenant en considération les contraintes définies précédemment.

Scénario alternatif: Si aucun chemin optimal n'est trouvé en fonction des contraintes, l'application informe l'utilisateur de l'absence de chemin optimal.

4. Diagramme de séquence

Diagramme de séquence du cas d'utilisation "**Créer un réseau**"

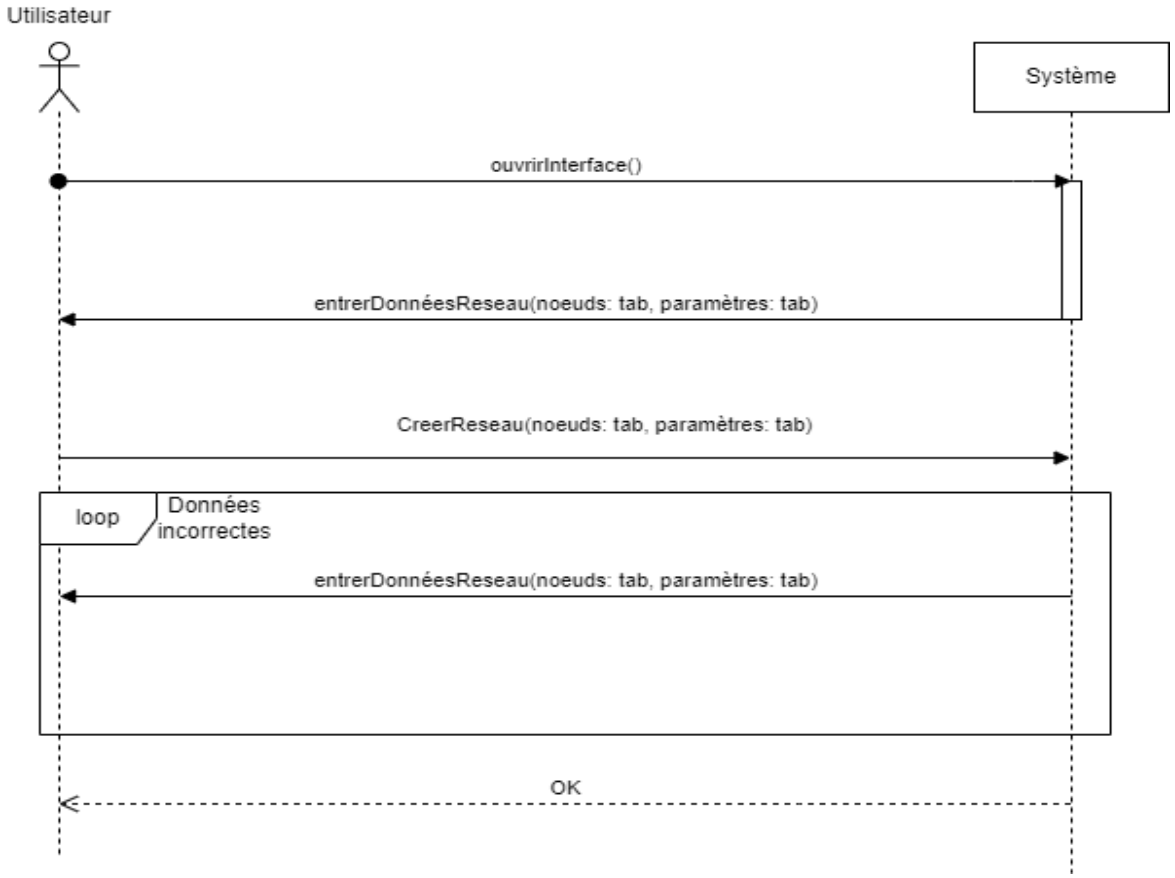


Figure 7: Diagramme de séquence du cas d'utilisation "Créer un réseau"

Diagramme de séquence du cas d'utilisation "**Définir contraintes**"

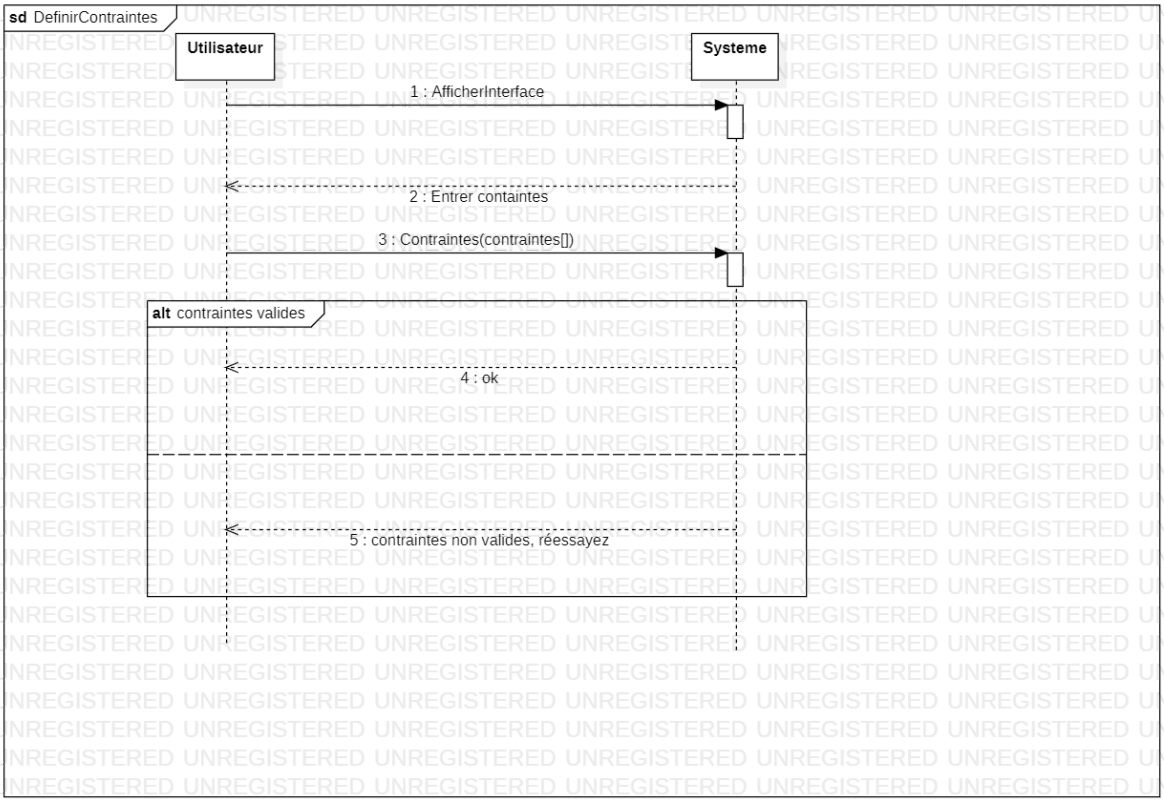


Figure 8 : Diagramme de cas d'utilisation "Définir contraintes"

Diagramme de séquence du cas d'utilisation "**Afficher chemins possibles**"

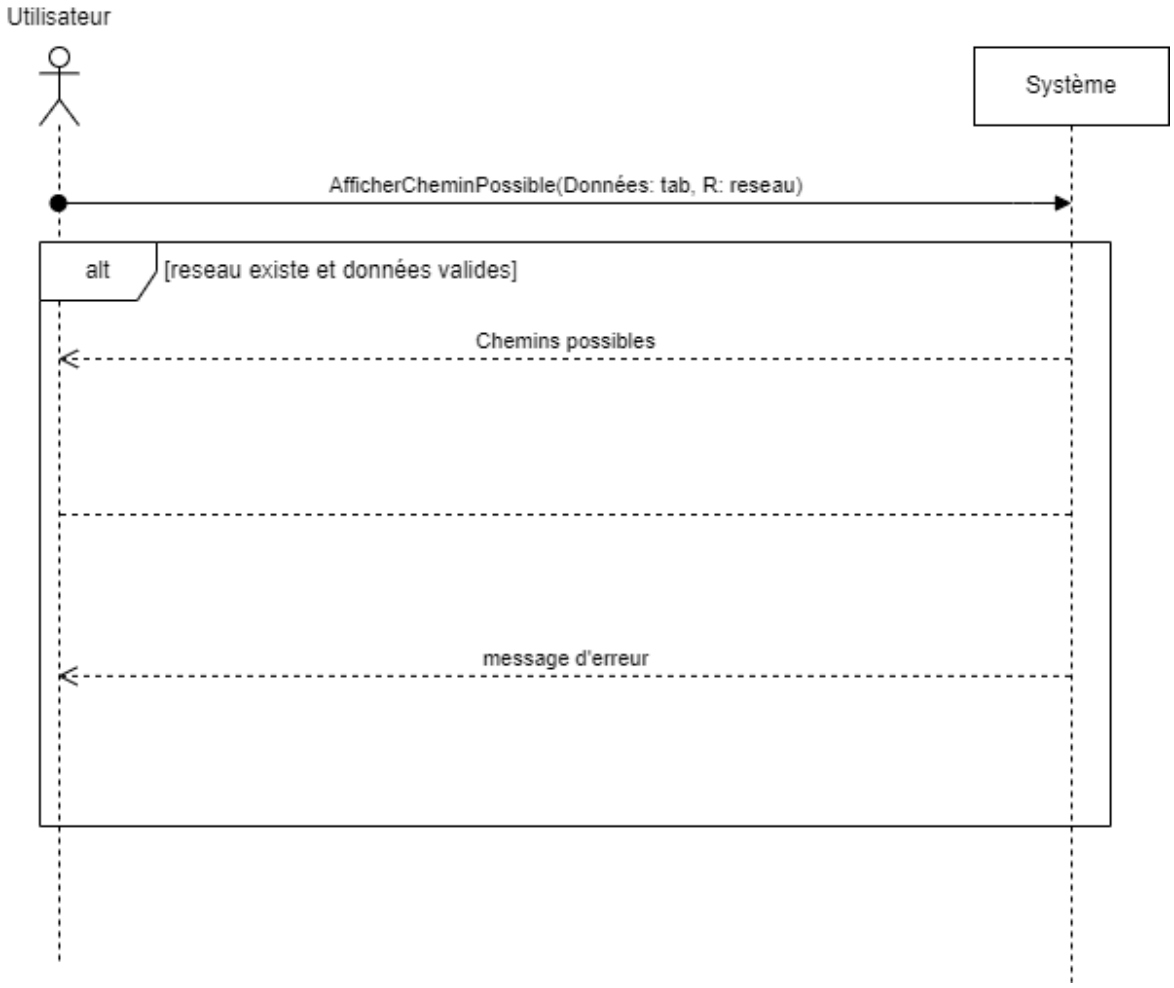


Figure 9 : Diagramme de séquence du cas d'utilisation "Afficher chemins possibles"

Diagramme de séquence du cas d'utilisation "**Afficher chemins optimal**"

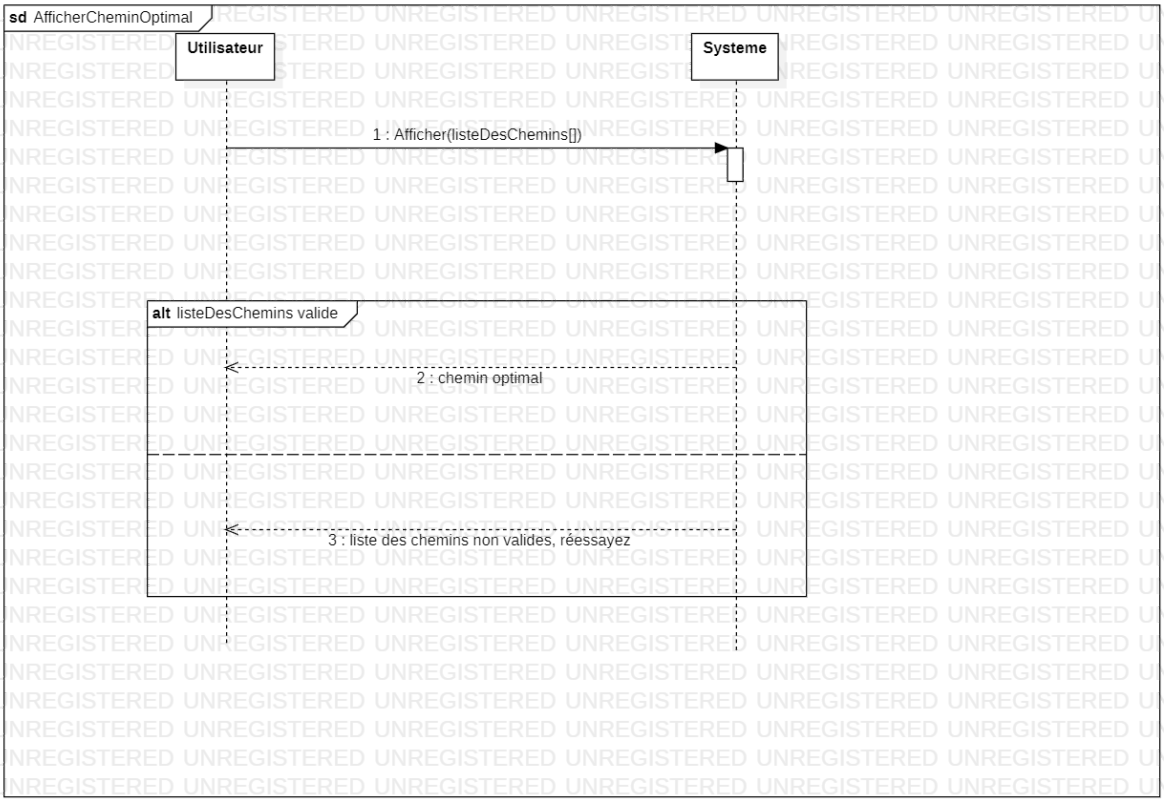


Figure 10: Afficher chemin optimal

VI. IMPLEMENTATION

1. Architecture du projet

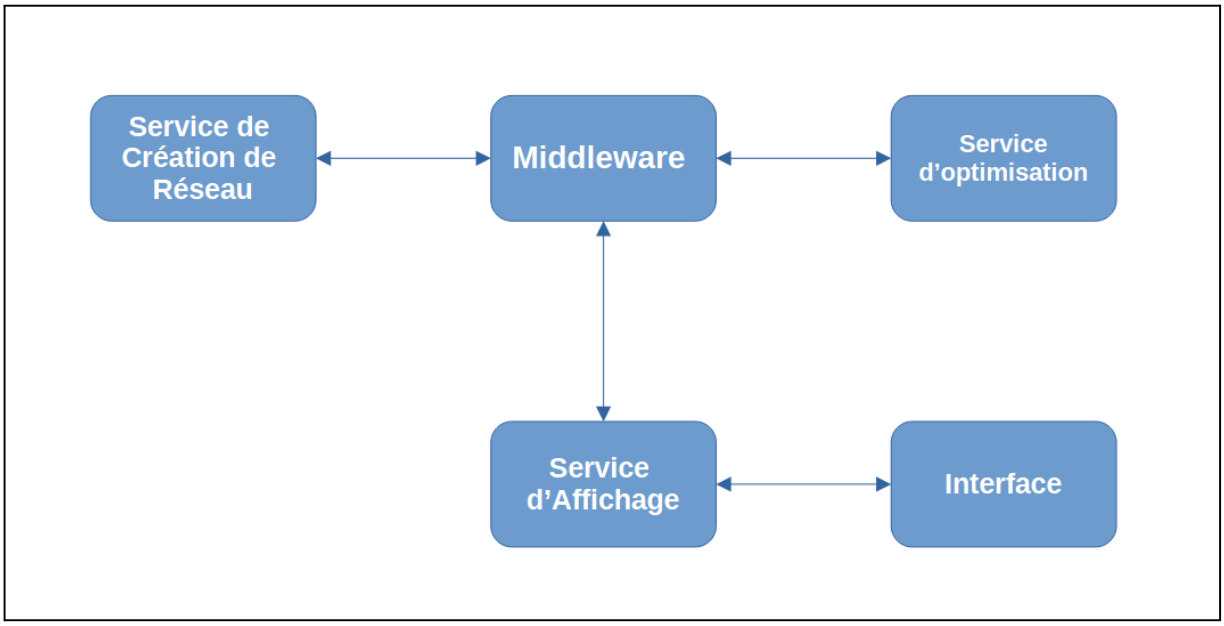


Figure 11 : Architecture

➢ **Service de Création de Réseau :**

Ce service est chargé de concevoir et de mettre en place les réseaux en fonction des besoins spécifiques des utilisateurs. Il prend en compte les paramètres tels que la topologie du réseau, la capacité, les exigences de performance, et d'autres critères pour créer une infrastructure réseau efficace et adaptée.

➢ **Middleware (Intergiciel) :**

Le middleware agit comme une couche logicielle intermédiaire qui facilite la communication et l'interaction entre différentes parties du système. Il offre des services de gestion des communications, de l'intégration des composants logiciels et de la coordination des activités au sein du réseau. Le middleware joue souvent un rôle crucial dans la gestion des transactions et la communication entre les différents services.

➢ **Service d'Optimisation :**

Ce service vise à améliorer l'efficacité et les performances du réseau en optimisant selon les contraintes présentées dans la modélisation. Il utilise l'algorithme présenté plus haut pour ajuster dynamiquement les paramètres du réseau afin d'obtenir le chemin optimal.

➢ **Service d'Affichage :**

Ce service fournit des mécanismes de visualisation pour représenter graphiquement divers aspects du réseau. Il crée la communication entre les autres services et l'interface permettant aux utilisateurs de voir en temps réel l'état du réseau.

➢ **Interface :**

L'interface est la partie du système qui permet l'interaction entre les utilisateurs et les différents services du réseau. Elle offre une plateforme conviviale permettant aux utilisateurs de configurer le réseau, d'accéder aux fonctionnalités de visualisation, et d'interagir avec les services de manière intuitive.

En résumé, cette architecture forme un ensemble cohérent où le service de création de réseau élabore l'infrastructure, le middleware facilite les échanges, le service d'optimisation améliore les performances, le service d'affichage visualise les données importantes, et l'interface offre un moyen convivial d'interagir avec l'ensemble du système. Cette approche globale contribue à la gestion efficace et à l'optimisation des réseaux.

2. Outils nécessaires

Dans le but de réaliser notre projet, nous avons utilisé les environnement suivant:

a. Le système d'exploitation

Le système d'exploitation Linux: c'est le système le mieux adéquat en raison du fait qu'il est le seul système qui autorise l'installation de mininet-wifi.

b. Mininet-wifi

Mininet-WiFi est un fork de l'émulateur de réseau SDN Mininet et a étendu les fonctionnalités de Mininet en ajoutant des stations et des points d'accès WiFi virtualisés basés sur les pilotes sans fil Linux standard et le pilote de simulation sans fil 80211_hwsim. Cela signifie que de nouvelles classes ont été ajoutées afin de prendre en charge l'ajout de ces périphériques sans fil dans un scénario de réseau Mininet et d'émuler les attributs d'une station mobile tels que la position et le mouvement par rapport aux points d'accès.

Mininet-WiFi étend la base de code Mininet en ajoutant ou en modifiant des classes et des scripts. Ainsi, Mininet-WiFi ajoute de nouvelles fonctionnalités et prend toujours en charge toutes les capacités d'émulation SDN normales de l'émulateur de réseau Mininet standard.

c. Ryu

Ryu est un framework SDN (Software-Defined Networking) entièrement écrit en Python qui est basé sur le framework Mininet et fournit des composants logiciels avec une API bien définie qui facilite la création de nouvelles applications de gestion et de contrôle de réseau.

d. Bibliothèque de modélisation

Un exemple de bibliothèque utilisée est nommé Pyomo. C'est une bibliothèque de modélisation d'optimisation open source qui permet de formuler, résoudre et analyser des modèles d'optimisation à l'aide de Python. Pyomo prend en charge un ensemble diversifié de capacités d'optimisation pour la programmation linéaire, la programmation non linéaire et la programmation en nombre entiers. Utilisée pour définir des problèmes symboliques, elle crée des instances de problèmes concrets et résout ces instances

avec des solveurs standard. Pyomo est également capable de modéliser des applications d'optimisation structurées.

e. Solveur

Il s'agit d'un programme informatique qui permet de résoudre des problèmes d'optimisation en trouvant la meilleure solution possible à un problème donné. Il contribue à améliorer la prise de décision en matière de planification, d'allocation et de programmation des ressources rares. Ils intègrent des algorithmes puissants qui peuvent résoudre des modèles de programmation mathématique, des programmes par contraintes et des modèles de planification par contraintes. Il existe plusieurs types de solveurs d'optimisation qui peuvent être utilisés pour résoudre des problèmes d'optimisation, tels que les problèmes de programmation linéaire, de programmation en nombres entiers, de programmation quadratique, de programmation par contraintes, etc. Voici quelques exemples de solveurs qu'il existe:

- **Gurobi:** c'est un solveur commercial hautes performances de pointe pour les problèmes de programmation linéaire, linéaire à nombres entiers mixtes et quadratique à grande échelle. Contrairement à glpk, Gurobi est une application multithread qui peut tirer pleinement parti d'un ordinateur portable multicœur. Gurobi propose des licences gratuites d'une durée d'un an pour un usage académique et des licences d'essai pour un usage commercial. Si votre application est devenue trop grande pour glpk, vous voudrez certainement essayer Gurobi.
- **GLPK(GNU Linear Programming Kit):** C'est un solveur open source sous la bannière du projet GNU. GLPK est un optimiseur de programmation linéaire et mixte en nombres entiers et est actuellement développé et maintenu par Andrew Makhorin, de l'Institut d'aviation de Moscou. GLPK for MPL permet aux utilisateurs de MPL d'accéder à ce solveur gratuit renommé à partir de l'environnement Windows convivial de MPL. Comme pour les autres solveurs, l'optimiseur GLPK est accessible à partir de MPL pour Windows en tant que

bibliothèque de liens dynamiques (DLL). Cette intégration étroite et robuste permet aux utilisateurs MPL d'accéder à divers algorithmes et options de la solution GLPK à partir de leur application MPL.

3. Implémentation des services

a. Service d'optimisation:

Ce service est le principal objectif de notre projet car il contient l'implémentation de toute la modélisation mathématique faite plus haut. pour ce service plusieurs fonctions ont ete defini:

- **la fonction de recherche des chemins**

Cette fonction récupère les paramètres ajouter lors de la création du réseau et retourne l'ensemble des chemins. [Annexe 1]

- **la fonction de recherche des chemins possibles suivant la contrainte d'énergie**

Cette fonction récupère les chemins obtenus après exécution de l'algorithme précédent et élimine les chemins qui possèdent des nœuds dont l'énergie résiduelle est insuffisante pour relayer l'information. [Annexe 2]

- **la fonction d'obtention du chemin optimal**

Cette fonction prend en entrée la liste des chemins possibles et utilise les contraintes de bandes passante, et de perte pour afficher le chemin optimal. [Annexe 3]

c. Service de creation du reseau :

Cette partie utilise le framework ryu et l'émulateur mininet-wifi pour créer une topologie adéquate en fonction des besoins des utilisateurs. [Annexe 4]

Avec le langage python, ryu nous permet de manipuler mininet-wifi. Nous pouvons ainsi créer et manipuler un réseau avec un code centralisé.

VII. LIMITES ET PERSPECTIVES

1. Limites

- ➢ Gestion des contraintes : La gestion des différentes contraintes, telles que la bande passante, les taux de perte, et les niveaux d'énergie résiduelle, peut être délicate. Des ajustements et des compromis peuvent être nécessaires pour trouver un équilibre optimal entre les différentes contraintes.
- ➢ Impossibilité d’ajouter l'énergie résiduelle aux noeuds
- ➢ Manipulation des solveurs difficile

2. Perspectives

- Optimisation des algorithmes : Investir dans la recherche et le développement d'algorithmes plus efficaces pour l'identification du chemin optimal. Explorer des techniques d'optimisation plus avancées pour réduire la complexité et améliorer les performances, en particulier dans des réseaux de grande envergure.
- Adaptation Dynamique : Intégrer des mécanismes d'adaptation dynamique qui permettent à l'algorithme de s'ajuster en temps réel en fonction des changements dans le réseau, tels que les fluctuations de la charge du trafic, les pannes de nœuds, ou les variations des niveaux d'énergie.
- Machine Learning : Explorer l'utilisation de techniques de machine learning pour la prédiction des niveaux d'énergie résiduelle des nœuds et pour améliorer la prise de décision quant à l'identification du chemin optimal. Les modèles prédictifs peuvent contribuer à anticiper les conditions futures du réseau.
- Évaluation de la Fiabilité : Intégrer des mécanismes d'évaluation de la fiabilité des chemins optimaux proposés. Considérer la fiabilité des liens, la probabilité de perte de paquets, et d'autres métriques pour garantir la stabilité du chemin optimal dans des conditions dynamiques.
- Interopérabilité avec SDN : Assurer une meilleure interopérabilité avec les architectures de Software-Defined Networking (SDN). Exploiter les avantages de la programmabilité du réseau pour faciliter la mise en œuvre et la gestion des chemins optimaux.
- Sécurité : Renforcer la sécurité de l'algorithme en considérant des mécanismes pour prévenir les attaques potentielles, telles que les attaques par déni de service ou les atteintes à la confidentialité des informations de routage.

CONCLUSION

En conclusion, ce projet de routage multi-chemin intelligent dans les réseaux maillés sans fil programmables a abouti à des résultats significatifs et à des contributions importantes dans le domaine des réseaux de communication. Notre méthodologie, basée sur l'utilisation de Programmes Linéaires en Nombres Entiers (PLNE), a permis de formuler de manière efficiente le problème complexe du choix du chemin optimal. Nous avons réussi à intégrer les caractéristiques du réseau, définir des paramètres et contraintes pertinents, et formuler le problème en PLNE pour une résolution optimale. Les résultats obtenus démontrent une amélioration significative de l'efficacité du routage, prenant en compte la bande passante, le taux de perte et l'énergie résiduelle des nœuds. L'impact potentiel de cette solution s'étend à l'optimisation des performances du réseau tout en maximisant la durée de vie des nœuds, renforçant ainsi la robustesse et la fiabilité du système. En termes de perspectives, des recherches futures pourraient se concentrer sur l'extension de cette méthodologie à des réseaux plus vastes et complexes, ainsi que sur l'exploration de mécanismes adaptatifs pour faire face à des changements dynamiques dans l'environnement du réseau. En vue de son déploiement, nous recommandons une validation approfondie de la solution dans des scénarios réels, ainsi qu'une collaboration avec des acteurs de l'industrie pour une intégration efficace dans des infrastructures existantes. Ces recommandations visent à assurer une mise en œuvre réussie de notre solution, contribuant ainsi à l'évolution et à l'efficacité des réseaux sans fil programmables.

BIBLIOGRAPHIE

1-https://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra

2-https://etudestech.com/decryptage/algorithme-dijkstra-calculer-chemin-plus-court-sommet-graphe/

3-https://fr.wikipedia.org/wiki/Algorithme_de_Bellman-Ford

4-https://fr.wikipedia.org/wiki/Optimisation_lin%C3%A9aire#D%C3%A9finitions

5-https://www.bioxegy.com/post/biomimetisme-exemples-algorithmes#:~:text=Les%20algorithmes%20bio%2Dinspir%C3%A9s%20trouvent,de%20possibilit%C3%A9s%20est%20tr%C3%A8s%20%C3%A9lev%C3%A9.

6-https://www.digikey.be/fr/articles/quickly-create-a-mesh-network-device-compatible-with-multiple-protocols

7-Formalisation mathématique.
https://fr.wikipedia.org/wiki/Formalisation_(math%C3%A9matiques)

8-L'utilité mathématique de la formalisation, https://www.jstor.org/stable/41088523

9-document HAL Id: tel-00955887 https://theses.hal.science/tel-00955887v2/document

10-https://www.gerad.ca/Sebastien.Le.Digabel/MTH8415/6_ONE.pdf

11-https://jckantor.github.io/ND-Pyomo-Cookbook/notebooks/01.01-Installing-Pyomo.html

12-GLPK pour MPL (maximalsoftware.com)

13-http://www.pyomo.org/

14-architecture de SDN: https://pressbooks.pub/sdnet/chapter/introduction/

ANNEXES

- **[Annexe 1]** :

```
from collections import deque

class Graph:
    def __init__(self):
        self.voisin = {}
        self.energy = {}

    def add_edge(self, edge_list):
        for edge in edge_list:
            u, v = edge
            if u not in self.voisin:
                self.voisin[u] = set()
            if v not in self.voisin:
                self.voisin[v] = set()
            self.voisin[u].add(v)
            self.voisin[v].add(u)

    def set_energy(self,node,energy):
        self.energy[node] = energy

    def set_energy_Dict(self,energy_Dict):
        for k,v in energy_Dict.items():
            self.energy[k] = v

    def get_energy(self,node):
        return self.energy[node]

    def get_neighbors(self,node):
        return self.neighbors[node]
```

```
    def find_paths(self, start, end):
        queue = deque([(start, [start])])
        paths = []

        while queue:
            node, current_path = queue.popleft()

            if node == end:
                paths.append(current_path)
            else:
                if node not in self.voisin:
                    continue

                neighbors = self.voisin[node]
                for neighbor in neighbors:
                    if neighbor not in current_path:
                        queue.append((neighbor, current_path +
[neighbor]))

        return paths
```

- **[Annexe 2]**

```
def possible_path(self,PATH_LIST,energy):

        TEMP = []

        for path in PATH_LIST:

            for node in path:

                if self.get_energy(node) < energy and path in PATH_LIST:

                    PATH_LIST.remove(path)

        return PATH_LIST
```

- **[Annexe 3]**

```
def optimal_path(self,chemins, type_service):

       if type_service == "temps réel":

           bande_passante_perte =
{('A','B'):10,('B','C'):20,('A','C'):105,('B','D'):5,('C','D'):18,('C',
'E'):20, ('D','E'):15}

           bande_passante_perte_totale = []

           for liste in chemins:

               total = 0

               for i in range(len(liste)-1):

                   noeud1, noeud2 = sorted([liste[i], liste[i+1]])  #
Trier pour éviter les problèmes d'ordre

                   total += bande_passante_perte.get((noeud1, noeud2),
0)  # Ajouter la bande passante du nœud, 0 si non défini

               bande_passante_perte_totale.append(total)

           max_bande_passante_perte = max(bande_passante_perte_totale)

           index_max =
bande_passante_perte_totale.index(max_bande_passante_perte)

           max_liste = chemins[index_max]

           return max_liste

       elif type_service == "partagé":

           return min(chemins, key=len)

       elif type_service == "dédié":

           bande_passante =
{('A','B'):10,('B','C'):20,('A','C'):105,('B','D'):5,('C','D'):18,('C',
'E'):20, ('D','E'):15}

           bande_passante_totale = []

           for liste in chemins:
```

```
                total = 0

                for i in range(len(liste)-1):

                    noeud1, noeud2 = sorted([liste[i], liste[i+1]])  #
Trier pour éviter les problèmes d'ordre

                    total += bande_passante.get((noeud1, noeud2), 0)  #
Ajouter la bande passante du nœud, 0 si non défini

                bande_passante_totale.append(total)

            max_bande_passante = max(bande_passante_totale)

            index_max = bande_passante_totale.index(max_bande_passante)

            max_liste = chemins[index_max]

            return max_liste

        else:

            raise ValueError("Type de service non reconnu")
```

- **[Annexe 4]** :

```
from collections import deque
from ryu.base import app_manager
from ryu.ofproto import ofproto_v1_3
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER,
MAIN_DISPATCHER
```

```
from ryu.controller.handler import set_ev_cls
from ryu.lib.packet import packet, ethernet, ether_types
from ryu.topology.api import get_switch, get_link
from ryu.topology import event, switches

class RyuApp(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(RyuApp, self).__init__(*args, **kwargs)
        self.graph = Graph()
        self.mac_to_port = {}

    def add_edge(self, edge_list):
        self.graph.add_edge(edge_list)

    def set_energy(self, node, energy):
        self.graph.set_energy(node, energy)

    def find_paths(self, start, end):
        return self.graph.find_paths(start, end)

    def best_path(self, PATH_LIST, energy):
        return self.graph.best_path(PATH_LIST, energy)

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        # Gérez les paquets entrants ici
        pass

    @set_ev_cls(event.EventSwitchEnter)
    def _event_switch_enter_handler(self, ev):
        # Gérez l'événement de l'entrée d'un commutateur ici
        pass

    @set_ev_cls(event.EventSwitchLeave)
    def _event_switch_leave_handler(self, ev):
```

```
        # Gérez l'événement de la sortie d'un commutateur ici
        pass

    @set_ev_cls(event.EventLinkAdd)
    def _event_link_add_handler(self, ev):
        # Gérez l'événement de l'ajout d'un lien ici
        pass

    @set_ev_cls(event.EventLinkDelete)
    def _event_link_delete_handler(self, ev):
        # Gérez l'événement de la suppression d'un lien ici
        pass

if __name__ == '__main__':
    from ryu import cfg
    cfg.CONF.register_opts([
        cfg.StrOpt('network_graph', default='network_graph.json',
                   help='Path to the network graph JSON file'),
    ])
    app_manager.run(RyuApp)
```