Projet ANR- 13-ASTR-0024
Application Driven Networking in Dynamic Environments

Livrable 3

Cas d'application communications dans un théâtre d'opérations : implémentation, validation et évaluation

Référence document	ANR-13-ASTR-0024-L1.1
Liste des auteurs	S. Abdellatif, P. Berthou, K. B. Nougnanke
Version	1.0
Date de livraison	1/10/2017

Résumé

Ce document présente la mise en œuvre d'un démonstrateur de l'architecture définie dans ta tache 1.3, pour les applications dynamiques dans un contexte de réseau dynamique. L'application visée correspond à un scénario de communications sur un théâtre d'opérations. La plateforme repose sur un ou plusieurs contrôleurs Openflow commandant des bornes wifi sur lesquelles un portage logiciel de la pil Openflow linux a été réalisé. Ce document présente les limites du déploiement d'une solution SDN actuelle et standard dans un environnement sans fil et propose un ensemble d'extensions permettant un déploiement qui permet l'exploitation des concepts SDN dans ce cadre.

Mots-clés

Software Defined Networking (SDN), virtualisation réseau, Network Functions Virtualization (NFV), cloud networking, Data Distribution Service (DDS)

Résumé.
Les réseaux multi-sauts sans fil constituent une classe des réseaux sans fil où des noeuds mobiles et/ou fixes forment à la volée un réseau dynamique et temporaire sans une administration centralisée. Ces réseaux présentent plusieurs caractéristiques dont une topologie très dynamique qui contraignent les techniques de routage et la fourniture de qualité de service dans ces réseaux. L'application de l'approche SDN (Software Defined Networking) aux réseaux multi-sauts sans fil permettraient de répondre efficacement à ces contraintes. C'est dans ce cadre que s'inscrit cette tâche dont l'objectif est triple. Premièrement la mise en place d'une plateforme sans fil SDN constituée de noeuds sans fil pilotables par un contrôleur SDN à travers le protocole OpenFlow. Les noeuds considérés reposent sur un système d'exploitation de type Linux et embarquent le switch logiciel OpenFlow « OpenVSwitch ». Deuxièmement il s'agit de définir un service de découverte de topologie qui maintient une base d'information riche et à jour sur la topologie du réseau (noeuds, liens) et la qualité des liens. Et enfin des expérimentations ont été menées sur la plateforme développée dans le but de caractériser les communications entre le contrôleur et les noeuds sans fil. En effet, l'efficacité d'un contrôle SDN est dépendante de la latence et de la fiabilité des communications entre contrôleur et noeuds. Cette caractérisation permettra d'analyser puis valider expérimentalement l'applicabilité d'une approche SDN aux principales fonctions réseau : routage, contrôle d'accès(scheduling), contrôle de topologie.

Mots clés : SDN, Réseaux multi-sauts sans fil, OpenFlow, OpenVSwitch, OFDP, LLDP, AvilaGetwork, BananaPi, RaspberryPi

Abstract.
Multi-hop wireless networks are a class of wireless networks where mobile and/or fixed nodes form a dynamic network on the fly and temporary without a centralized administration. These networks have several characteristics, including a very dynamic topology that constrain routing techniques and the provision of quality of service in these networks. Application of the Software Defined Networking (SDN) approach to multi-hop networks would effectively address these constraints. This is the purpose of this task whose goal is triple. Firstly the implementation of an SDN wireless platform made up of wireless nodes that can be controlled by an SDN controller through the OpenFlow protocol. The nodes considered are based on a Linux operating system and embed OpenFlow software switch "OpenVSwitch". Secondly, we define a topology discovery service that maintains a rich and up-to-date informations on network topology (nodes, links) and link quality. And finally, experiments were carried out on the platform in order to characterize the communications between the controller and the wireless nodes. Indeed, the effectiveness of an SDN control is dependent on the latency and the reliability of communications between controller and nodes. This characterization will allow to analyze and then experimentally validate the applicability of an SDN approach to the main network functions : routing, access control (scheduling) topology control.

keywords : SDN, Wireless multi hop networks, OpenFlow, OpenVSwitch, OFDP, LLDP,AvilaGetwork, BananaPi, RaspberryPi

Table des matières

Résumé **ii**

Abstract **ii**

Introduction générale **1**

1 Contexte et Problématique **3**
1.1 Contexte . . . 3
1.2 Problématique . . . 3

2 SDN et Réseaux sans fil **5**
2.1 Le sans-fil . . . 5
2.1.1 Généralités sur les réseaux sans fil . . . 5
2.1.2 Les réseaux multi-sauts sans fil . . . 6
2.1.3 La Qos dans les réseaux muti-sauts sans fil . . . 7
2.1.4 Le routage dans les réseaux muti-sauts sans fil . . . 8
2.2 Précisions sur le protocole OpenFlow . . . 8
Les messages OpenFlow . . . 8
Interaction Contrôleur et commutateur OpenFlow . . . 9
2.3 Architecture SDN et réseaux sans fil : Réseaux multi-sauts sans-fil SDN 10
2.3.1 Intérêts et difficultés de l'approche . . . 12

3 Mise en place d'une plateforme réseau multi-sauts sans fil SDN **13**
3.1 Besoins . . . 13
3.2 Mise en place de la plateforme . . . 14
3.2.1 Choix techniques . . . 14
Type de connexion . . . 14
Type de cartes . . . 15
Type de système d'exploitation . . . 17
3.2.2 OpenVSwitch . . . 17
Structure d'OpenvSwitch : les différents composants . . . 17
3.2.3 Le contrôleur SDN : Ryu . . . 19
Ryu Controller . . . 19
3.2.4 Déploiement des cartes . . . 19
Plan du déploiement . . . 19
Type de contrôle (In-band ou Out-of-band) . . . 20
Photos des cartes déployées et poste de contrôle . . . 22

4 Vers un service de découverte de topologie générique pour les Réseaux Multi-saut Sans-fil **25**
4.1 Objectifs du service de découverte de topologie . . . 25
4.2 Le service de découverte du contrôleur Ryu . . . 26

4.2.1 Quelques éléments sur la plateforme utilisée pour décrire le service de découverte . . . 27
4.2.2 Découverte des noeuds . . . 27
4.2.3 Suivi de la présence et des caractéristiques des noeuds . . . 30
4.2.4 Découverte et suivi des liens . . . 32
4.2.5 LLDP . . . 32
4.2.6 OFDP . . . 33
4.3 Analyse des limites du service de découverte de Ryu pour les réseaux multi-saut sans-fil . . . 36
4.4 Propositions d'extensions du service de découverte de RYU . . . 37
4.4.1 Prise en compte des attributs de noeuds et de ports . . . 37
4.4.2 Prise en compte de liens multipoints sans-fil dans le service de découverte de RYU . . . 39
Algorithme actuellement utilisé par RYU . . . 39
Algorithme modifié . . . 40
4.4.3 Prise en compte de métriques de qualité du lien . . . 40
Extension de OFDP : LLDP et agents OVS . . . 40
SNMP . . . 41
OF messages . . . 42
4.5 Positionnement par rapport à l'existant . . . 44
4.5.1 En termes de découverte de topologie . . . 44
4.5.2 En termes de réduction du surdébit (overhead) . . . 44
4.5.3 Conclusion . . . 45

5 Analyse expérimentale des communications contrôleur SDN à noeud sans-fil 47
5.1 Préambule . . . 47
5.2 Objectifs . . . 48
5.3 Evaluation du débit du trafic de contrôle . . . 48
5.3.1 Scénario considéré . . . 48
5.3.2 Résultats expérimentaux . . . 49
5.3.3 Conclusion . . . 50
5.4 Evaluation de l'impact de la charge du réseau sur les messages OFDP . 51
5.4.1 Conclusion . . . 52

Conclusion générale 53

A Annexe 55
A.1 Quelques commandes utiles . . . 55
A.2 Mise en marche de la plateforme . . . 55
A.2.1 Cas du contrôle out-of-band . . . 55
A.2.2 Cas du contrôle in-band . . . 56
A.2.3 Exemples d'applications à lancer sur le contrôleur . . . 56

Table des figures

2.1 Modes infrastructure et ad hoc 5
2.2 multi hop communication 6
2.3 Diagramme de séquence communication CTRL-OVS 11
2.4 Exemple d'application du SDN aux réseaux multi-saut sans fil 12

3.1 Limitation du mode ad hoc simple 15
3.2 Exemple de configuration mesh 16
3.3 Carte Avila GateWork 2348 16
3.4 Carte Banana Pi R1 16
3.5 Carte Raspberry Pi 3B 17
3.6 Vision générale d'OpenvSwitch 18
3.7 Les principaux composants d'OpenvSwitch 18
3.8 Plan de déploiement des cartes dans le laboratoire 20
3.9 Contrôle Out-of-band 21
3.10 Contrôle In-band 21
3.11 Avila 23
3.12 Banana Pi 23
3.13 Raspberry Pi 23
3.14 Contrôleur 23

4.1 Visualisation de la plateforme déploiyée 27
4.2 Requête OF_FEATRES_REQUEST (Ctrl vers OVS 17) 28
4.3 Réponse OFPT_FEATRES_RPLY (OVS 17 vers Ctrl) 28
4.4 Requête OFPMP_PORT_DESC_STATS_REQUEST (Vers OVS17) 29
4.5 Réponse OFPMP_PORT_DESC_STATS_REPLY (d'OVS17). 29
4.6 Paquet OFPT_ECHO_REQUEST (de l'OVS17 vers le Ctrl) 31
4.7 Paquet OFPT_ECHO_REPLY (du Ctrl vers OVS17) 31
4.8 paquet OFPT_PORT_STATUS (de l'OVS17 vers le Ctrl) 31
4.9 Trame LLDP 32
4.10 Exemple de la circulation de la trame LLDP entre OVS17 et OVS11 . . 34
4.11 Paquet 'Packet_out LLDP' (du Ctrl vers OVS17) 35
4.12 Capture de la trame LLDP au niveau de l'OVS17 35
4.13 Paquet 'LLDP' (de l'OVS17 vers OVS11) 35
4.14 Paquet 'Packet_IN LLDP' (de l'OVS17 vers le Ctrl) 36
4.15 Illustration du problème des connexions multipoints 37
4.16 Interactions entre manager et agent SNMP 41
4.17 Exemple d'arbre OID 42
4.18 L'OID experimental 43
4.19 Metriques de lien recueillies par SNMP 43

5.1 Débit du trafic(messages) OpenFlow 49
5.2 Le trafic de contrôle au niveau du contrôleur (séparé) 50

5.3 Délai des messages OFDP du packet-out au Packet-in pour la détection d'un lien 51
5.4 Contrôle out-of-band 52
5.5 Contrôle in-band 52

Liste des tableaux

2.1 Différences entre réseaux cellulaires et réseaux ad hoc. 6

3.1 Quelques éléments de la plateforme 23

4.1 Données Techniques des OVS 11 et 17. 27
4.2 Spécification des champs de la trame LLDP. 33

5.1 Débits individuels. 50
5.2 Réseau expérimental - delai OFDP 52

Abréviations

ADN	**A**pplication **D**riven **N**etwork
AODV	**A**d-hoc **O**n-demand **D**istance **V**ector
BSC	**B**ase **S**tation **C**ontroller
CSMA	**C**arrier **S**ense **M**ultiple **A**ccess
ETX	**E**xpected **T**ransmission Count
ETT	**E**xpected **T**ransmission **T**ime
GSM	**G**lobal **S**ystem for **M**obile Communications
HLR	**H**ome **L**ocation **R**egister
IP	**I**nternet **P**rotocol
ISM	**I**ndustrial, **S**cientific and **M**edical
LLDP	**L**ink **L**ayer **D**iscovery **P**rotocol
LP-WAN	**L**ow **P**ower **W**ide **A**rea **N**etwork
MAC	**M**edium **A**ccess **C**ontrol
MANET	**M**obile **A**d hoc **NET**work
MSC	**M**obile **S**witching **C**enter
OID	**O**bject **ID**entifier
OFDP	**O**pen**F**low **D**iscovery **P**rotocol
OF	**O**pen**F**low
OLSR	**O**ptimized **L**ink **S**tate **R**outing **P**rotocol
OSI	**O**pen **S**ystems **I**nterconnection
OVS	**O**pen**VS**witch
PDU	**P**rotocol **D**ata **U**nit
QoS	**Q**uality **o**f **S**ervice
RSSI	**R**eceived **S**ignal **S**trength **I**ndication
SDN	**S**oftware **D**efined **N**etwork
SINR	**S**ignal to **I**nterference plus **N**oise **R**atio
SNR	**S**ignal **N**oise **R**atio
SMI	**S**tructure of **M**anagement **I**nformation
SNMP	**S**imple **N**etwork **M**anagement **P**rotocol
TCP	**T**ransmission **C**ontrol **P**rotocol
TDMA	**T**ime **D**ivision **M**ultiple **A**ccess
TLV	**T**ype **L**ength **V**alue
U-NII	**U**nlicensed **N**ational **I**nformation **I**Infrastructure
WISE	**WI**reless **SE**nsor network
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WMN	**W**ireless **M**esh **N**etwork
WSN	**W**ireless **S**ensor **N**etwork
ZRP	**Z**one **R**outing **P**rotocol

Projet ANR- 13-ASTR-0024
Application Driven Networking in Dynamic Environments

Livrable 3

Cas d'application communications dans un théâtre d'opérations : implémentation, validation et évaluation

Référence document	ANR-13-ASTR-0024-L1.1
Liste des auteurs	S. Abdellatif, P. Berthou, K. B. Nougnanke
Version	1.0
Date de livraison	1/10/2017

Résumé

Ce document présente la mise en œuvre d'un démonstrateur de l'architecture définie dans ta tache 1.3, pour les applications dynamiques dans un contexte de réseau dynamique. L'application visée correspond à un scénario de communications sur un théâtre d'opérations. La plateforme repose sur un ou plusieurs contrôleurs Openflow commandant des bornes wifi sur lesquelles un portage logiciel de la pil Openflow linux a été réalisé. Ce document présente les limites du déploiement d'une solution SDN actuelle et standard dans un environnement sans fil et propose un ensemble d'extensions permettant un déploiement qui permet l'exploitation des concepts SDN dans ce cadre.

Mots-clés

Software Defined Networking (SDN), virtualisation réseau, Network Functions Virtualization (NFV), cloud networking, Data Distribution Service (DDS)

Introduction générale

De plus en plus d'applications sont des applications dynamiques dans la mesure où les flux de données qu'elles échangent et les besoins en qualité de service évoluent dans le temps. De plus, elles peuvent potentiellement évoluer dans un environnement dynamique (environnement impliquant des noeuds mobiles et hétérogènes qui communiquent via des liens sans-fil). Ce caractère dynamique des applications et de leur environnement constitue l'un des défis majeurs qui est posé aux réseaux de communication sous-jacents dont la mission est de leur offrir un service réseau capable de suivre leurs attentes. En effet, cela suppose que le réseau soit suffisamment flexible pour être reprogrammé selon la situation. C'est malheureusement le talon d'Achille des réseaux actuels.

Les réseaux SDN (Software Defined Networks ou réseaux définis par logiciel) sont un nouveau type de réseaux qui prônent la séparation des fonctions de contrôle et des fonctions d'acheminement au sein des équipements réseau (les fonctions de contrôle sont centralisées au sein d'équipements contrôleurs), et le pilotage à distance de ces fonctions d'acheminement par du logiciel. Cela permet de construire des réseaux flexibles dont le fonctionnement peut être reprogrammé à la demande des applications. Ils constituent une solution prometteuse pour supporter efficacement des applications dynamiques citées ci-avant.

Cette dernière tâche vise à développer une plateforme permettant la mise en place de solutions SDN sur un réseau de communication sans fil et dans le but d'offrir des services de communication à qualité de service. Notre première contribution a porté sur la mise en place d'une plateforme réseau multi-sauts sans-fil conforme au modèle SDN. Cette dernière a servi de base pour effectuer une évaluation expérimentale des communications entre le contrôleur et les noeuds sans-fil qu'il contrôle. Une telle étude est primordiale pour appréhender les possibilités et limites de l'application du paradigme SDN aux réseaux multi-sauts sans-fil. Notre troisième contribution a porté sur le développement d'un service de découverte de topologie (qu'implémente le contrôleur) adapté aux réseaux multi-sauts sans-fil.

Le présent rapport s'organise en 5 chapitres. Le chapitre 1 présente le contexte du workpackage.
Dans le chapitre 2 nous présentons et les réseaux multi-sauts sans fil et le concept SDN, leurs principales caractéristiques ainsi que le bien fondé d'une hybridation de ces deux concepts. Les trois derniers chapitres entrent dans le vif du sujet en présentant les différentes réalisations.
Le chapitre 3 présente les détails techniques relatifs aux divers choix techniques qui ont été adoptés pour la mise en place de la plateforme réseau SDN sans-fil, et les résultats de ce déploiement. Il détaille les différentes technologies utilisées (contrôleurs, cartes, système d'exploitation, etc.).
Le chapitre 4 concerne le service de découverte de topologie implémenté par le contrôleur SDN. Il expose les insuffisances du service de découverte de topologie utilisé par les contrôleurs actuels pour une mise en application aux réseaux multi-sauts sans-fil, et nos propositions d'amélioration et d'extension.
Et avant de conclure ce rapport, vient le chapitre 5 qui présente les expérimentations menées sur la plateforme ainsi que les analyses des résultats obtenus.

Chapitre 1

Contexte et Problématique

1.1 Contexte

L'objectif général de la tâche 3 du projet ADN est de définir, prototyper et évaluer un système de communication basé sur les réseaux de type SDN (Software Defined Networks) capable de supporter efficacement des applications distribuées dynamiques évoluant dans des environnements dynamiques. L'idée principale est de déployer pour chaque application, un réseau virtuel supportant les flux de données applicatifs et dont le comportement peut être reprogrammé dynamiquement en fonction de l'évolution des échanges de l'application et de l'évolution de l'environnement dans lequel elle opère.

L'un des cas d'application du projet ADN met l'accent sur la dimension « dynamique de l'environnement » et concerne le cas des communications sur un théâtre d'opération ou en situation d'intervention d'urgence avec des noeuds hétérogènes, mobiles qui communiquent via des liens sans fil. Les réseaux de données correspondant à cet environnement dynamique s'assimilent aux réseaux multi-sauts sans fil.

1.2 Problématique

Les réseaux multi-sauts sans fil présentent principalement deux types de contraintes : celles liées aux techniques de routage et celles relatives à la fourniture de qualité de service. Ceci pour plusieurs raisons : le caractère aléatoire du support de communication sans fil, canaux partagés, problèmes de bande passante, **mobilité des noeuds**, **topologie dynamique**, imprécision sur l'estimation des canaux, **contraintes énergétiques** etc.

Ces réseaux auto-organisés reposent généralement sur des protocoles de routage décentralisés où tous les noeuds prennent part à la décision d'acheminement des données de bout en bout. Parmi les protocoles les plus connus on a AODV et OLSR. AODV (Ad hoc On Demand Distance Vector) est un protocole de routage réactif du type «On-demand Driven », le chemin étant calculé à la demande. De ce fait, il est auto-démarrant peu gourmand en énergie et ne nécessite pas de grande puissance de calcul, gènère cependant un retard au démarrage (lors de la définition

des routes). A l'inverse, OLSR (Optimized Link State Routing Protocol) est un protocole proactif qui échange régulièrement des informations de topologie avec tous les noeuds. Les routes sont ainsi disponibles à tout moment, mais il y a une surcharge causée par l'échange de messages périodiques. L'on trouve également dans la littérature des protocoles hybrides (réactifs et proactifs) comme le ZRP (Zone Routing Protocol) qui établissent la topologie de manière pro-active et la maintienent de manière réactive.

Toutefois, cette vision totalement décentralisée n'est pas forcément la meilleure et des solutions optimales différentes peuvent être trouvées dès lors que l'on connaît la topologie ou les caractéristiques du réseau. L'approche proposée ici repose sur l'utilisation du paradigme de réseau SDN (Software Defined Networking) qui centralise le contrôle du réseau afin d'en optimiser l'exploitation.

Chapitre 2

SDN et Réseaux sans fil

2.1 Le sans-fil

.

2.1.1 Généralités sur les réseaux sans fil

Dans le domaine des réseaux sans fil, on peut faire une distinction entre les réseaux à infrastructure et les réseaux sans infrastructure. Dans le cas des réseaux à infrastructure, une infrastructure fixe, généralement filaire est disponible, et toutes les communications sont dirigées sur cette épine dorsale. Cette approche est utilisée dans les réseaux cellulaires (GSM, 3G, 4G) où l'infrastructure est constituée par les stations de base, le Core Network (BSC, MSC, HLR) et les réseaux locaux sans fil (WLAN) où l'on a un point d'accès. Quant aux réseaux sans infrastructure, pas d'épine dorsale ni d'administration centralisée, les périphériques sans fil communiquent directement entre eux par des connexions point-à-point(figure 2.1). Les réseaux sans infrastructure sont également appelés réseaux ad hoc, systèmes autonomes de noeuds mobiles, réseaux peuvant être déployés à la volée, sans avoir besoin d'une planification préalable. Le tableau 2.1 [**Reference15**] présentent les différences entre réseaux cellulaires et réseaux ad hoc.

Bien que les communications sans fil aient apporté de nombreux avantages (connectivité élevée, mobilité, ...) depuis leur apparition, elles présentent certains inconvénients par rapport aux communications filaires. Les communications sans-fil souffrent d'interférences, d'une faible disponibilité de bande passante, de faibles débits de données par rapport au filaire et aussi du phénomène d'évanouissement du signal. De tels inconvénients entrainent des limitations dans les transmissions sans-fil. Par exemple, pour qu'un noeud envoie un paquet à une destination hors de sa portée de transmission, il devrait dépendre de certains noeuds intermédiaires

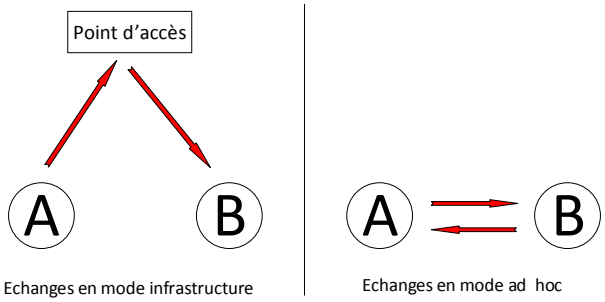


FIGURE 2.1: Modes infrastructure et ad hoc

TABLE 2.1: Différences entre réseaux cellulaires et réseaux ad hoc.

Réseaux Cellulaires	**Réseaux Ad Hoc**
Réseaux à infrastructure	Réseaux sans infrastructure
Stations de base fixes	Pas de stations de base et déploiement rapide
Topologie du réseau coeur(Backbone) statique	Topologie du réseau fortement dynamique et communication multihop
Environnement et connetivité stable	Environnment hostile(bruit, pertes) et connectivité instable
Une bonne planification au préalable avant déploiement des stations de base	Réseaux crées à la volée et facilement extensible
Couts d'installation élevés	Déploiement à faible cout

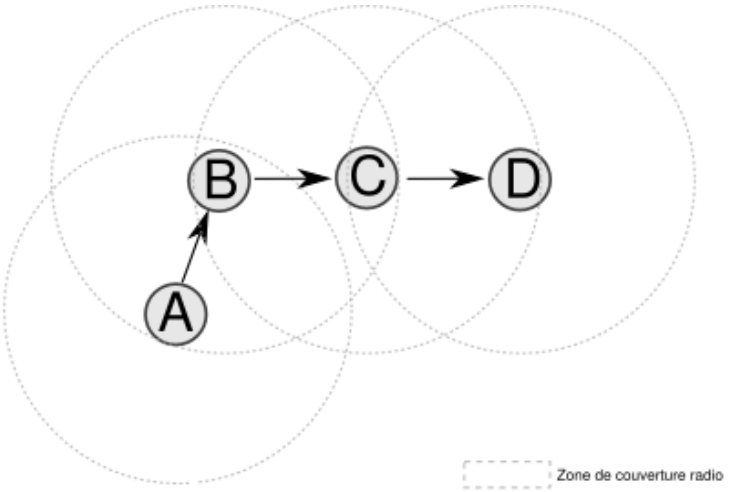


FIGURE 2.2: Communication multi hop.

pour relayer le paquet. Un tel paradigme est connu sous le nom de communication multi-hop et les réseaux sans fil adoptant ce paradigme de communication sont connus sous le nom de réseaux multi-hop sans fil. Et ceci fait partie intégrante des réseaux sans fil sans infrastructure (réseaux ad hoc) : puisque les connexions point-à-point direct ne sont possibles qu'entre noeuds sans fil qui se situent en zone radio immédiate les uns des autres, la communication entre les noeuds qui sont éloignés les uns des autres doivent être pris en charge par d'autres noeuds dans le réseau, qui fonctionnent comme des points relais. Ainsi dans cette configuration chaque noeud joue le rôle d'un récepteur et d'un routeur.Nous avons une illustration du fonctionnement multi-sauts sur la figure 2.2.

2.1.2 Les réseaux multi-sauts sans fil

Les réseaux ad hoc multi-sauts sans fil (ad hoc wireless multi-hop networks) sont des réseaux capables de s'organiser sans infrastructure définie préalablement où tous les noeuds sont en mesure de se joindre les uns aux autres de manière transparente. Dans de tels réseaux, les noeuds distants qui ne peuvent pas communiquer directement utilisent des noeuds intermédiaires pour relayer les informations. Cette capacité de couvrir une large zone pour un faible coût rend ce type de réseaux très intéressants et populaires. L'on peut identifier trois (3) grands types de réseaux

multi-sauts sans fil : Réseaux mobiles ad hoc (MANET : Mobile Ad hoc NETworks), Réseaux maillées sans fil (WMN : Wireless Mesh Networks) et les Réseaux de capteurs sans fil (WSN : Wireless Sensor Networks).

Les réseaux maillés sans fil se distiguent des MANETs par l'absence ou le peu de mobilité et généralement ne présentent pas trop de contraintes d'énergie comme les MANETs et les réseaux de capteurs. En effet dans les réseaux maillées l'on peut noter la présence d'un coeur de réseau (dit fédérateur ou backbone) constitué de routeurs fixes et alimentés.

Le projet ADN adresse les réseaux multi-sauts sans fil dont les grands défis sont le maintient de la connectivité par le routage et la gestion de la qualité de service.

2.1.3 La Qos dans les réseaux muti-sauts sans fil

La fourniture de la qualité de service dans les réseaux sans fil est problématique pour plusieurs raisons : le caractère aléatoire du support de communication sans fil, canaux partagés, problèmes de bande passante, mobilité des noeuds, topologie dynamique, imprécision sur l'estimation des canaux, contraintes énergétiques etc.

Les réseaux muti-sauts sans fil sont généralement basés sur la technologie IEEE 802.11, ou du moins ceux considérés dans ce travail. Cette technologie utilise une technique d'accès de type "Carrier Sense Multiple Access (CSMA)" qui souffre des problèmes de stations cachées, stations exposées, collisions etc [**Reference1**]. Et fournir de la qualité de service pour ce type de réseaux n'est pas chose évidente.Le standard IEEE 802.11e a été proposé afin d'intégrer de la qualité de service dans les couches MAC et physique. IEEE 802.11e introduit des classes de services avec différentiation à la priorité d'accès. Mais cette solution est plutôt spécifique aux réseaux locaux sans fil et ne fonctionne pas nécessairement pour les réseaux multi-sauts. La thechnique "Accès multiple par répartition dans le temps (TDMA)" pourraient fournir une certaine qualité de service dû à son caractère déterministe. Cependant, les solutions basées sur TDMA sont plus adaptées aux WLAN à un saut et assurer une bonne synchronisation entre noeuds distribués d'un réseau multi-sauts peut être très problématique.

Une deuxième famille de solutions se concentre sur la couche réseau. Une catégorie de ces solutions propose la réservation des ressources, notamment la bande passante le long d'un chemin afin de garantir la bande passante de bout-en-bout. Chose pas facile non plus à cause de la faible disponibilité de bande passante dont disposent les réseaux sans fil. Une autre classe cherche à contrôler l'admission sans réservation préalable de ressources (DiffServ - Differentiated Services).

Enfin, des solutions fonctionnant au niveau de plusieurs couches du modèle OSI ont également été proposés afin de fournir de la qualité de service. Ces dernières supposent l'utilisation dans les couches inférieures d'un protocole de routage réactif qui couple la phase de découverture de chemin au contrôle d'admission et la réservation de ressource.

2.1.4 Le routage dans les réseaux muti-sauts sans fil

L'objectif principal des protocoles de routage est l'établissement et la maintenance de la meillleure route entre une paire de noeuds afin que les messages puissent être livrés de manière fiable et en temps opportun. Les caractéristiques des réseaux multi-sauts ne rendent pas cette tâche facile. Les protocoles de routage devront fonctionner sur des réseaux avec des topologies très dynamiques où les algorithmes de routage s'exécutent sur des périphériques à ressources limitées.

On distingue deux grandes familles de protocoles de routages dans les réseaux multi-sauts sans fil :

Les protocoles de routage réactifs agissent sur le principe de demande c.à.d. la construction de routes en cas de besoin. Ils permettent de réduire considérablement la surcharge de paquets de routage dans le réseau mais génère une latence considérable dans le processus de routage plus précisement lors de la définition des routes. Le plus connu est AODV (Ad-hoc On-demand Distance Vector)

Les protocoles proactifs suivent une approche dans laquelle l'information de routage est régulièrement diffusée à travers le réseau afin que les tables de routage de chaque noeud soient mises à jour. Par conséquent, les routes sont disponibles là tout moment, mais il y a une surcharge causée par l'échange de messages périodiques. Le plus connu et le plus adopté est OLSR (Optimized Link State Routing Protocol)

L'on distingue également des protocoles dits hybrides qui essayent de tirer des avantages des protocoles réactifs et proactifs selon les situations. Par exemple on a le protocole ZRP (Zone Routing Protocol) résolvent une partie du problème en établissant la topologie de manière pro-active et en la maintenant de manière réactive.

2.2 Précisions sur le protocole OpenFlow

L'architecture ADN repose en parti sur le protocole OpenFlow comme interface entre le matériel et le contrôleur de réseau. Sans revenir sur les spécificités du protocole déjà décrites dans les précédents rapports, nous avons souhaité décrire plus précisément les messages liés à la découverte du réseau par le contrôleur. En effet, les extensions qui ont été nécessaires à la mise en place de la plateforme sont principalement liées à cet aspect du protocole.

Les messages OpenFlow

Ces interactions sont caractérisées par des messages échangés à travers le canal OpenFlow. On distingue 3 types de messages : messages contrôleur-vers-switch, messages asynchrones et les messages symétriques

1. Les messages contrôleur-vers-switch
   Ce sont des messages initiés par le contrôleur et qui peuvent nécessiter des réponses de la part du switch ou non. Quelques-uns de ces messages sont les suivants :

- Features :
  A travers ces messages (OFPT_FEATURES_REQUEST) le contrôleur peut demander l'identité d'un switch et ses fonctionnalités de bases.
- Read-State :
  Pour récupérer des informations sur le switch telles que sa configuration actuelle, les statistiques ...
- Configuration :
  Pour opérer une certaine configuration sur le switch
- Packet-out :
  Le contrôleur utilise ces messages pour demander au switch d'envoyer un paquet sur un port spécifique. Généralement ce paquet est envoyé au contrôleur par le switch à travers un paquet-in. Le message paquet-out contient un paquet entier ou un buffer ID identifiant un paquet stocké par le switch et une liste d'actions à appliquer au paquet en question

2. Les messages asynchrones
   Ils sont envoyés par le switch au contrôleur pour lui indiquer un changement d'état ou l'arrivée d'un paquet :
   - Packet-in : Pour transférer au contrôleur des informations sur un paquet dont il ne sait comment le traiter. Le message packet-in peut contenir tout le paquet en question ou juste un buffer ID si le switch a la capacité de bufferisé le paquet.
   - Flow-Removed : Pour informer le contrôleur du retrait d'une entrée de la table de flux
   - Port-status : pour donner des informations sur un quelconque changement survenu sur un port
3. Les messages symétriques
   Ce sont des messages envoyés sans sollicitation ni du switch ni du contrôleur :
   - Hello : Messages utilisés au début de la connexion pour négocier la version d'OpenFlow à utiliser
   - Echo : Les messages echo request/reply sont utilisés pour s'assurer du maintien de la connexion contrôleur-switch, et aussi pour mesurer les paramètres de cette connexion (latence, bande passante ')
   - Error : Utilisés pour notifier des problèmes de connexion. Généralement utilisés par le switch pour indiquer l'échec d'une requête initiée par le contrôleur.

Interaction Contrôleur et commutateur OpenFlow

L'établissement effectif de la connexion entre le contrôleur et le switch se fait en 3 étapes comme suit :

1. Etablissement d'une session TCP entre le switch et le contrôleur sur l'initiative du switch, ce dernier étant au préalable configuré avec l'adresse IP du contrôleur. Il est réalisé par handshaking en trois temps avec les messages SYNC, SYNC-ACK et ACK

2. La négociation de la version d'OpenFlow par les massages OFPT_HELLO initiés par le contrôleur :
   Chacune des parties (Le contrôleur et le switch) envoie un message OFPT_HELLO dans lequel elle précise la dernière version du protocole OpenFlow supportée. La version la plus récente supportée par les deux parties sera adoptée (si y'a un échec lors de la négociation un message OFP-ERROR avec un type OFPET_HELLO_FAILED est envoyé par l'élément en difficulté, et la connexion sera interrompue
3. Les messages Feature (Request/Reply) également initiés par le contrôleur pour identifier les principales caractéristiques techniques du switch.
   Le message OFPT-FEATURES-REPLY (envoyé par le switch (comme réponse au message REQUEST)) contient :
   - Datapath_Id : l'identifiant du Switch
   - n_buffers : La capacité de bufferisation
   - n_tables : La capacité de la table des flux
   - Capabilities : Ensemble des fonctionnalités que le Switche supporte ou pas.

Les informations suivantes récupérées par le contrôleur concernent les ports du switch, elle s'effectue à travers les messages OF-PORT-DESC-STATS-REQUEST et OF-PORT-DESC-STATS-REPLY, et permet au contrôleur d'avoir les informations suivantes pour chaque port du bridge (switch) :

- port : chaque information de port est envoyée dans une trame séparée.
- Port no : Le numéro de port (important pour l'installation des flux)
- Hw addr : l'adresse Mac du port
- Name : Le nom du port
- Config : Config du port (Down/up, NO_RECV, NO_FWD,...)
- State : Information sur (BLOCKED, LIVE, DOWN)

Après ces échanges de messages, le contrôleur (après avoir récupéré toutes les informations du switch) peut maintenant ajouter, supprimer et modifier les tables de flux de celui-ci, pour ce faire, il utilise les messages OF_FLOW_MOD. Le switch est ainsi prêt à traiter les paquets par le protocole OpenFlow traitement qui se base essentiellement sur les tables de flux et/ou l'envoi des messages Packet-in au contrôleur.

La Figure 2.3 montre un diagramme de séquence d'interactions entre un contrôleur et un switch OpenFlow(OpenVswitch

2.3 Architecture SDN et réseaux sans fil : Réseaux multi-sauts sans-fil SDN

L'approche proposée par la tâche 3 du projet ADN consiste à porter les techniques des réseaux SDN « software defined network » pour les réseaux multi-sauts sans-fil. Avec cette architecture Figure 2.4, les réseaux multi-sauts sans fil pourront profiter des services proposés par la technologie SDN et ainsi proposer des solutions nouvelles, adaptées, aux problèmes de routage et de QoS que rencontrent ces réseaux.

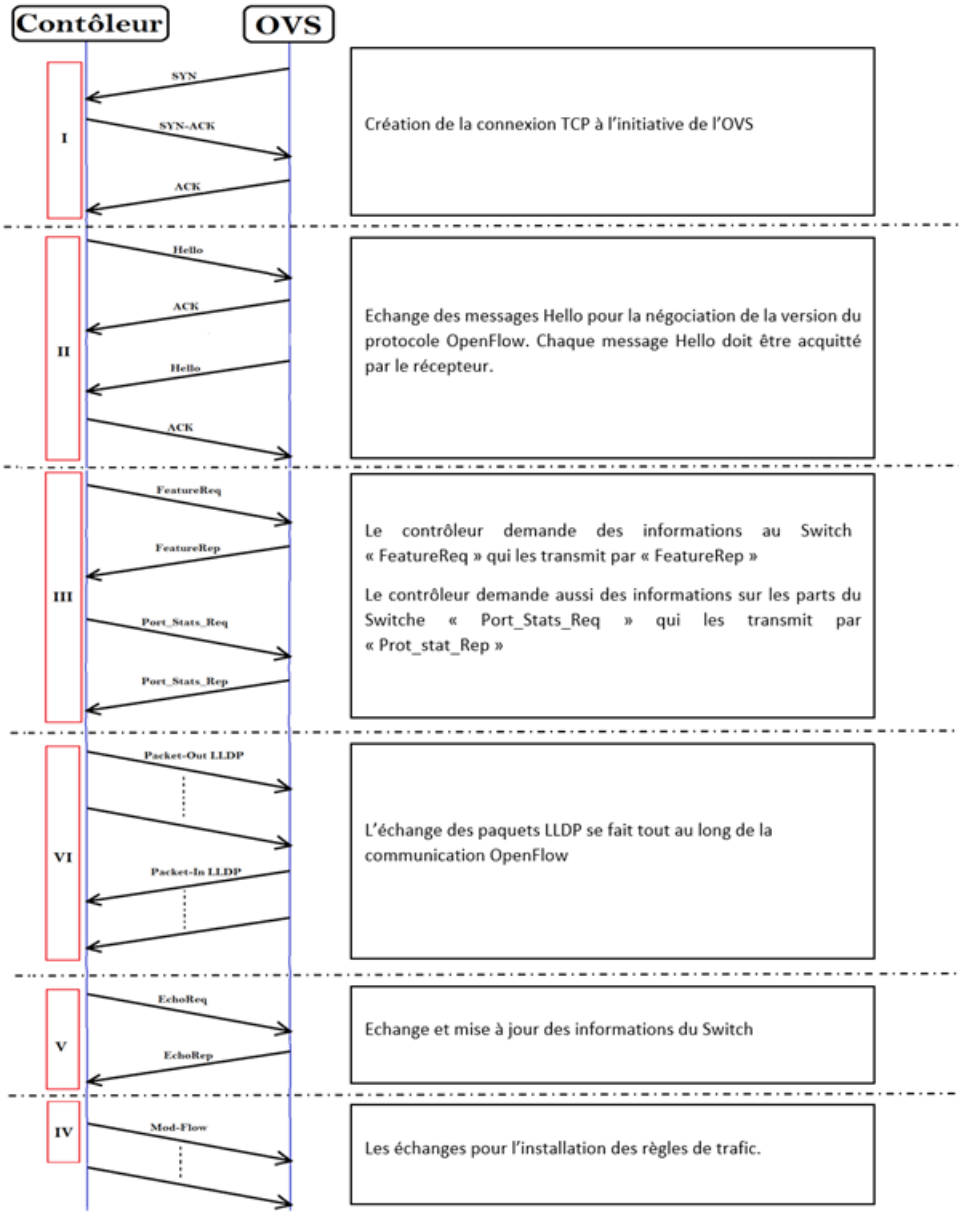


FIGURE 2.3: Diagramme de séquence communication CTRL-OVS.

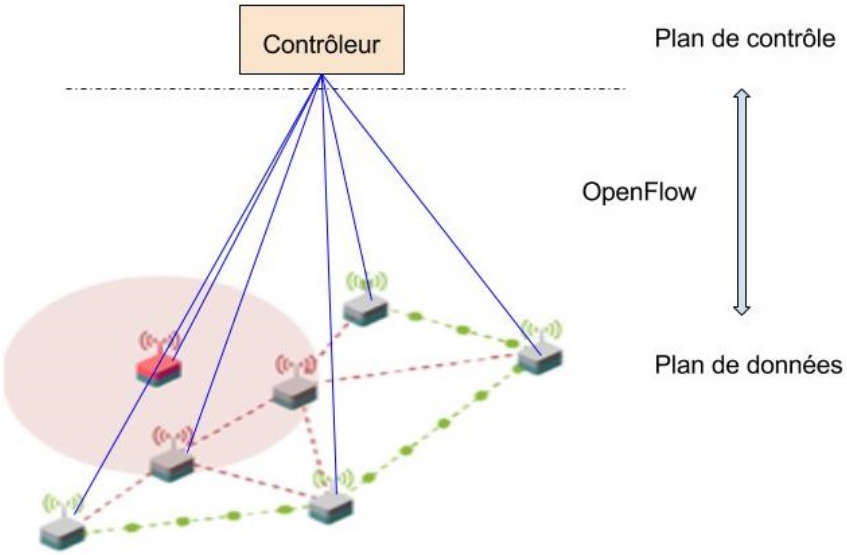


FIGURE 2.4: Exemple d'application du SDN aux réseaux multi-saut sans fil.

2.3.1 Intérêts et difficultés de l'approche

— Intérêts :
L'application du paradigme SDN aux réseaux multi-sauts sans-fil, ouvre le champ à l'élaboration de nouveaux algorithmes de contrôle réseau (routage, contrôle de topologie, gestion de la mobilité, etc.) grâce à la vue globale exposée par le contrôleur, et permet d'avoir un acheminement basé flux.

— Difficultés :
Pour arriver à ce fonctionnement, nous devons faire face à de nombreuses difficultés ; parmi ces difficultés, nous avons les performances modestes des communications entre contrôleur et noeud réseau et aussi un service de Découverte de topologie inadapté aux réseaux multi-sauts sans-fil.

Chapitre 3

Mise en place d'une plateforme réseau multi-sauts sans fil SDN

Ce chapitre présente les détails techniques relatifs aux divers choix techniques qui ont été adoptés pour la mise en place de la plateforme réseau, constituée de noeuds sans fil et d'un contrôleur SDN, et les résultats de ce déploiement. Il détaille encore les différentes technologies utilisées (contrôleurs, cartes, etc.).

3.1 Besoins

Le projet ADN vise le développement d'un système de communication offrant des services avec de la Qualité de Service (QdS) à des applications dynamiques évoluant dans un environnement dynamique. L'un des cas d'application considéré par le projet est un scénario d'intervention d'urgence tel qu'une troupe de pompiers qui interviennent pour porter secours suite à une catastrophe naturelle ayant détruits les moyens de communication conventionnels. Dans ce genre de situations, le réseau déployé pour supporter les communications entre secouristes et leur chef de groupe est un réseau sans-fil dédié. Il est typiquement constitué d'une vingtaine de noeuds (chacun transporté par un secouriste). Souvent, la couverture globale de ce réseau est plutôt réduite car les noeuds sont proches et se déplacent ensemble avec une mobilité plutôt déterministe. En dépit de la proximité des noeuds, la présence d'obstacles ou les performances rédhibitoires de certains liens sans-fil font que tous les noeuds ne sont pas à couverture radio mutuelle. Cela confère au réseau le caractère multi-sauts sans-fil. Les noeuds (ou routeurs) sont très souvent hétérogènes, multi-interfaces et multi-technologies. Les techniques d'accès TDMA (Time Division Multiple Access) sont privilégiées, même si les techniques CSMA (Carrier Sense Multiple Access) sont également considérées. Enfin, certains noeuds peuvent jouer le rôle de passerelle vers des réseaux avec infrastructure terrestres ou par satellite. Deux types de trafic sont supportés par le réseau :

— Un trafic de contrôle qui est plutôt périodique et qui est traité par le réseau de manière prioritaire

— Un trafic applicatif de différents types :

  — Phonie : très cadrée (ordres, etc.)

  — Informations d'état (position, etc.) plutôt périodiques avec des exigences de délai de livraison de l'ordre de la centaine de ms

  — Data : tel que du « SMS » ou du « chat ».

Le but de la plateforme à déployer est de reproduire le plus fidèlement possible le type de réseau décrit ci-avant en y inculquant un fonctionnement selon l'approche

SDN. Les contraintes matérielles et logistiques nous ont conduit au réseau dont les principales caractéristiques sont décrites ci-après.

Le réseau a été déployé en intérieur (« in-door ») avec une alimentation sur secteur. Il est composé de 17 noeuds statiques, chacun équipé de plusieurs interfaces Wi-Fi (« Wireless-Fidelity' ou 802.11) pouvant opérer dans les bandes ISM 2,4 GHz et 5 GHz. Afin de prendre en compte le caractère hétérogène des noeuds, trois types de noeuds ont été considérés : ceux basés sur des cartes Avila GATEWORK 2348 et ceux basés sur des cartes Banana PI R1 routeur et Raspberry pi 3 B. Deux types de systèmes opératoires ont également été considérés : OpenWRT et LEDE.

Le comportement SDN/OpenFlow de la plateforme a été mis en place en embarquant au niveau de chaque noeud un switch logiciel OpenVswitch (OVS). Ce switch relie les interfaces Wi-Fi du noeud et peut être contrôlé à distance via le protocole OpenFlow. Deux contrôleurs OpenFlow ont été considérés et testés : Opendaylight et Ryu. Enfin, tous les noeuds sans-fil du réseau sont rattachés à une infrastructure filaire via un VLAN d'entreprise pour permettre leur configuration à distance, la collecte de données expérimentales et la mise en place d'une horloge globale via le protocole NTP (Network Time Protocol). Comme expliqué dans la suite de ce chapitre, il peut également être utilisé pour émuler le transport hors-bande (Out-of-band) des messages OpenFlow. Nous détaillons dans les sections suivantes les composants et caractéristiques cités ci-avant.

3.2 Mise en place de la plateforme

3.2.1 Choix techniques

Type de connexion

Le choix du type de connexion devait répondre au critère de mobilité des noeuds qu'imposait le cahier des charges (topologie variable et environnement hautement dynamique), tout en assurant une connectivité multi-sauts (chaque noeud doit jouer le rôle d'un hôte et d'un pont(relais)). Trois modes de fonctionnement sont définis dans la norme 802.11 :

1. Point d'accès / infrastructure : Ce mode n'est pas un mode « Ad hoc », il limite la mobilité des noeuds par rapport à un noeud (ou plus) maitre (s) et ne permet pas l'utilisation du concept du multi-saut entre les noeuds du réseau. C'est pourquoi ce mode de fonctionnement n'a pas été testé et a été rapidement ignoré.

2. Ad-hoc : Ce mode permet une connectivité qui répond en partie aux besoins (pas d'infrastructure réseau, pas de réseau préalablement défini et gère une certaine mobilité des noeuds) mais, comme spécifié dans le standard IEEE 802.11, ce mode suppose une connectivité radio mutuelle entre les différents noeuds. Ainsi, deux noeuds qui ne sont pas en connectivité radio directe ne peuvent disposer d'une connectivité de niveau 2 même en présence d'autres noeuds via lesquels les trames pourraient transiter pour atteindre le noeud pair. En d'autres termes, avec ce mode, les noeuds ne peuvent jouer le rôle de relai. A titre illustratif, considérant l'exemple des 3 noeuds de la Figure 3.1, où OVS2 n'a pas de connexion directe avec OVS3 et les deux ont une connexion

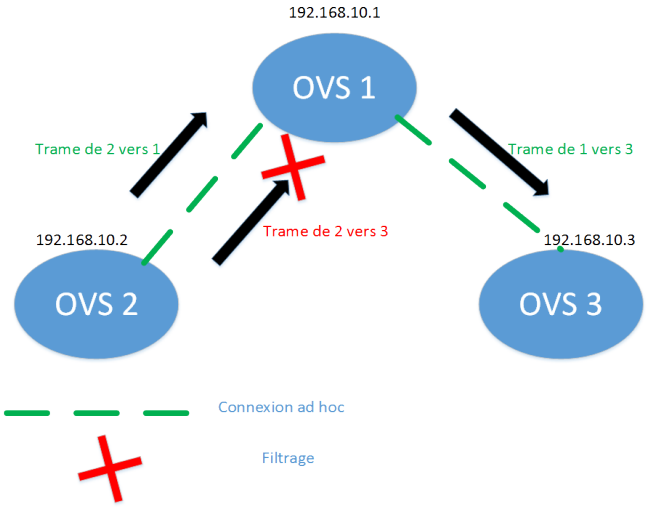


FIGURE 3.1: Limitation du mode ad hoc simple.

avec OVS1.

Ce mode Ad-hoc seul ne permet pas d'avoir une connectivité entre les deux OVS qui ne sont pas à une couverture radio mutuelle (OVS 2 et OVs 3). Le problème est que configuré en mode ad hoc "simple" un noeud ne peut relayer de paquet, la couche physique ne "fait pas suivre" les paquets à d'autres noeuds. Ainsi OVS1 ne laisse passer que les trames qui lui sont destinées et rejette toutes les autres

3. Mesh 802.11s : Les réseaux Mesh (802.11s) ou réseaux maillés sont une topologie du réseau où tous les noeuds sont connectés pair à pair sans hiérarchie centrale avec une topologie avec un maillage important. Chaque noeud doit être capable de recevoir, envoyer et relayer les données.
   Ainsi, le comportement par défaut des noeuds qui utilisent ce mode de fonctionnement est de relayer au niveau 2 toutes les trames reçues sur un port et qui ne lui sont pas destinées vers d'autre(s) port(s). De ce fait, il n'y a plus de filtrage par rapport à une adresse donnée et tout le trafic peut transiter vers d'autres noeuds. Et c'est ce mode qui est activé sur les noeuds. Un protocole de routage de niveau 2 est toutefois nécessaire pour proposer des routes vers chaque destination du réseau. Pour ce qui nous concerne, c'est une application SDN qui serait en charge de proposer les routes à suivre vers chaque destination. De ce fait, aucun protocole de routage n'a été activé au niveau des noeuds.

Type de cartes

Les cartes ont été choisies en respectant l'objectif d'hétérogénéité des noeuds imposé par le cahier des charges. Trois types de cartes ont été utilisés : Avila GateWork 'GW2348' , Banana Pi R1 et Raspberry Pi qui sont des cartes de petite taille (ordinateur à carte unique), pouvant être équipées de plusieurs interfaces sans-fil et qui supportent une large gamme de systèmes d'exploitation.

1. Avila GATEWORK 2348

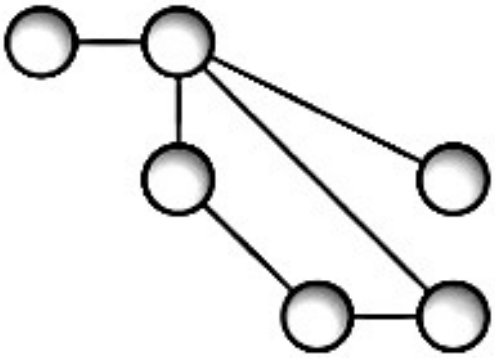


FIGURE 3.2: Exemple de configuration mesh.



FIGURE 3.3: .Carte Avila GateWork 2348

La GW2348 [1] (figure 3.3) est un membre de la famille des Gateworks Avila Network, elle répond aux exigences du résidentiel et d'entreprise en termes d'applications réseau. Elle est multi-interface sans-fil via ses quatre ports mini-PCI (pour branchement des cartes sans-fil, dont WiFi), et multi-interfaces filaires via ses deux ports Ethernet. Avec ses 16 Mbytes de mémoire flash, extensible par une carte externe, elle supporte une large gamme de systèmes d'exploitation. Elle tourne avec un processeur Intel Xscale à 533Mhz et une mémoire SDRAM de 64 Mbytes.

2. Banana PI R1
   De la famille des Banana Pi [2] (figure 3.4), les BPI-R1 sont des routeurs « Open-Hardware » qui supportent une large gamme de système d'exploitation, et sont multi-interface sans fil via leurs ports USB (auxquels il est possible de connecter des dongles WiFi) et tournent avec une CPU de 1.0Ghz et une mémoire DDR3 de 1 Go. Ils n'ont pas de mémoire flash intégrée mais un emplacement de carte mémoire pouvant aller jusqu'à 64 Go et un emplacement pour Disque dur.

1. http ://www.gateworks.com/product/item/avila-gw2348-4-network-processor
2. http ://www.banana-pi.org/r1.html



FIGURE 3.4: Carte Banana Pi R1.

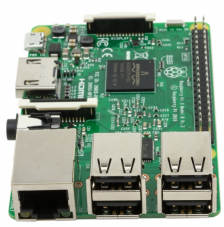


FIGURE 3.5: Carte Raspberry Pi 3B.

3. Raspberry pi 3 b
   De la famille des Raspberry Pi (figure 3.5), les RPI-3B [3] sont des ordinateurs à carte unique « OpenHardware » qui supportent une large gamme de système d'exploitation, et sont multi-interface sans-fil via leurs port USB (par des dongles WiFi) et tournent avec une CPU de 900MHz et une mémoire DDR3 de 1 Go. Ils n'ont pas de mémoire flash intégrée mais un emplacement de carte mémoire pouvant aller jusqu'à 64 Go.

Type de système d'exploitation

Le choix des systèmes d'exploitation s'est fait par rapport au mode de connexion choisie, Il a fallu trouver un système d'exploitation qui peut supporter le fonctionnement en mode Mesh 802.11s de ces interfaces Wifi et aussi supporter des drivers WiFi (pour les dongles WiFi USB) fonctionnant en 802.11s.
Parmi les nombreux systèmes existants (Linux ubuntu, rasbpian, ...), deux systèmes capables de supporter un driver fonctionnel pour le mode Mesh ont été identifiés. Il s'agit de OpenWrt et LEDE. Ces deux systèmes ont été adoptés et installés sur les noeuds de la plateforme.

OpenWrt et LEDE
Les systèmes OpenWrt et LEDE sont des distributions minimalistes GNU/Linux, ils se basent sur un noyau Linux, idéal pour les systèmes embarqués

3.2.2 OpenVSwitch

Après avoir choisi les cartes et leurs systèmes d'exploitation, il reste à donner un comportement 'Commutateur OpenFlow' aux cartes, les switchs OpenFlow virtuel 'OpenvSwitch' [**Reference5**] ont été utilsés pour cela.
L'objectif de OpenvSwitch est d'obtenir un commutateur OpenFlow virtuel ayant les mêmes fonctionnalités qu'un vrai switch OpenFlow, son code source est distribué sous licence Apache 2, sauf la partie spécifique au noyau Linux qui est sous GPL. Il est écrit en langage C, avec le souci d'être le plus indépendant possible de la plateforme sous-jacente. OpenVswitch est a la particularité d'être multicouches : elle travaille au niveau de la couche 2 du modèle OSI mais il peut également supporter la couche 3 comme la couche 4.Il présente ou supporte une multitude de fonctionnalités comme indiquée sur la Figure 3.6 ci-dessous.

Structure d'OpenvSwitch : les différents composants

On distingue trois principaux composants (comme indiquée sur la : Figure 3.7)

3. https ://www.raspberrypi.org/products/raspberry-pi-3-model-b/

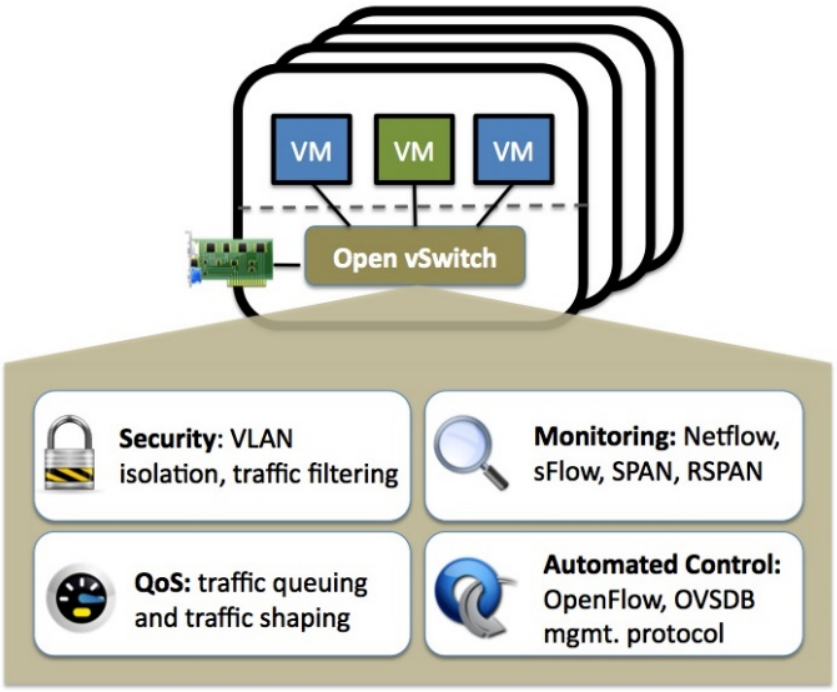


FIGURE 3.6: Vision générale d'OpenvSwitch.

— ovs-vswitchd (dans l'espace utilisateur)
— ovsdb-server (également dans l'espace utilisateur)
— OVS Kerner Module (dans le noyau)

OVS-VSWITCHD

C'est le daemon(processus s'exécutant en arrière-plan) principal du commutateur. Il est le coeur du fonctionnement du commutateur virtuel.Il commmunique avec les 2 autres entités : la base de données OVSDB-SERVER et le module noyau OVS KERNEL MODULE.
→ Avec l'ovsdb-server soit pour stocker la configuration du switch soit pour la consulter. L'importance de cette base de données (stockée sur un disque non volatile)

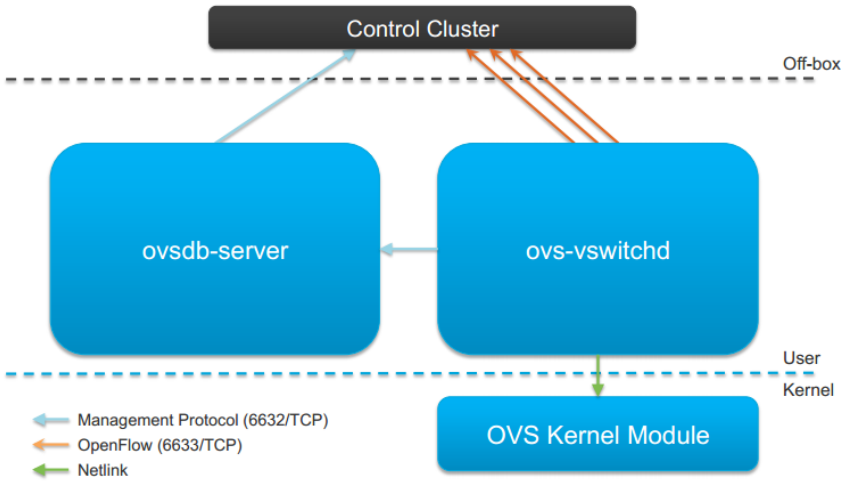


FIGURE 3.7: Les principaux composants d'OpenvSwitch.

est que l'on garde la configuration du switch même après un reboot La configuration du switch concerne les bridge, ports , interfaces, adresses de contrôleurs ..
→ Avec le module noyau à travers Netlink, famille de sockets des interfaces du noyau linux permettant d'assurer les communications entres processus de l'espace utilisateur et ceux de l'espace noyau. C'est le module noyau qui est responsable de l'acheminement des paquets.

3.2.3 Le contrôleur SDN : Ryu

Un contrôleur SDN a pour rôle d'ajouter, supprimer et modifier d'une façon dynamique les entrées de flux dans la 'table des flux' de chaque switch OpenFlow (Virtuel ou pas) pour ainsi 'contrôler' le comportement du réseau.

Ryu Controller

Ryu [**Reference6**] [**Reference16**], faisant partie des contrôleurs SDN, est un framework qui fournit un ensemble de composants logiciels ainsi que des API bien définis permettant de développer très facilement de nouvelles applications dans de le plan de contrôle, de gestion et de la configuration des équipements réseaux. Il supporte plusieurs protocoles parmi les lesquels on peut retenir : OpenFlow, Netconf, OF-Config. Entièrement écrit en python, Ryu a son code source disponible gratuitement sous la licence Apache 2.0. Ryu est un mot japonais qui signifie "flux". C'est le contrôleur Ryu que nous avons adopté pour le projet.

3.2.4 Déploiement des cartes

17 noeuds fixes ont été déployés au sein du LAAS/CNRS (où chaque carte représente un noeud) à une distance moyenne de 20-30 mètres les unes des autres. Le déploiement de noeud mobile n'a pas été effectué pour l'instant mais reste à l'étude. Afin d'être capable d'intervenir (via SSH) sur les cartes, pour modifier des configurations, un VLAN a été configuré sur un réseau filaire ethernet. Il facilite la gestion des cartes, mais servira également comme réseau de contrôle dans le cas d'un réseau SDN avec un contrôle en « Out-of-band »). Les canaux WiFi ont été sélectionnés afin de ne pas trop perturber le réseau sans fil existant au LAAS. De par la nature des expérimentations que nous étions amenés à réaliser, il était important de s'éloigner des bandes de fréquences utilisées par le réseau WiFi existant.

Plan du déploiement

La Figure 3.8 montre le déploiement des cartes (au sein du LAAS) et un jeu de canaux utilisés.

A titre d'exemple, le noeud OVS 12 au centre de la figure, est équipé de deux interfaces Wi-Fi : une première opérant sur le canal 36 de la bande UNII et une deuxième dans le canal 11 de la bande ISM. L'une de ses interfaces Ethernet (en rose) est connectée à un point de rattachement de l'infrastructure réseau du LAAS qui la place dans le VLAN correspondant au réseau de gestion de la plateforme. Comme indiqué dans la Figure 3.8, le contrôleur SDN est à son tour relié à ce réseau de gestion et pourrait, au besoin, exploiter ce réseau de gestion pour communiquer directement avec les noeuds de la plateforme.

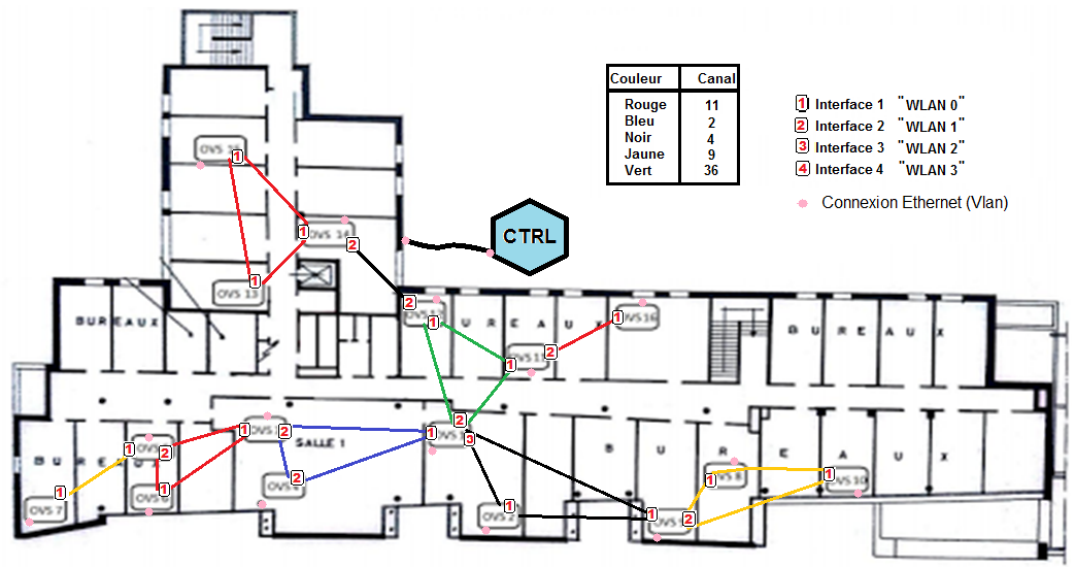


FIGURE 3.8: Plan de déploiement des cartes dans le laboratoire.

Type de contrôle (In-band ou Out-of-band)

Un switch (commutateur) OpenFlow doit établir et maintenir une connexion TCP avec son contrôleur. Deux solutions peuvent être adoptées pour transporter sur le réseau les messages OpenFlow (dits messages de contrôle) de cette connexion : Une première consiste à les transporter sur un réseau dédié complétement distinct du réseau utilisé pour le transport des données « usager , on parle alors d'une contrôle hors-bande(Out-of-band) ; La deuxième consiste à faire transiter les messages de contrôle sur les mêmes chemins de données que les données des utilisateurs, on parle alors d'un contrôle « dans-la-bande » (In-band).

Un contrôle Out-of-band

Dans ce type de configuration, notre plateforme comporte deux réseaux séparés un pour le trafic de contrôle (OpenFlow), en rouge sur la Figure 3.9 et qui se confond avec le réseau de gestion, et l'autre est le réseau multi-sauts sans-fil (en bleu sur la Figure 3.9) qui transporte le trafic de données (Vidéo, Voix, ...). L'avantage d'une telle configuration est que le trafic de données n'entrera pas en concurrence avec le trafic de contrôle et n'aura aucune influence sur le fonctionnement du protocole OpenFlow et vice versa.

Les avantages qu'offre cette méthode se résument comme suit :

— Simplicité : Simplifie légèrement l'implémentation des switchs.
— Fiabilité : le volume excessif du trafic des données n'interférera pas avec le trafic de contrôle.
— Sécurité : L'authentification et le contrôle d'accès des noeuds au VLAN réseau de gestion limite les menaces relatives à la présence de noeuds illégitimes.

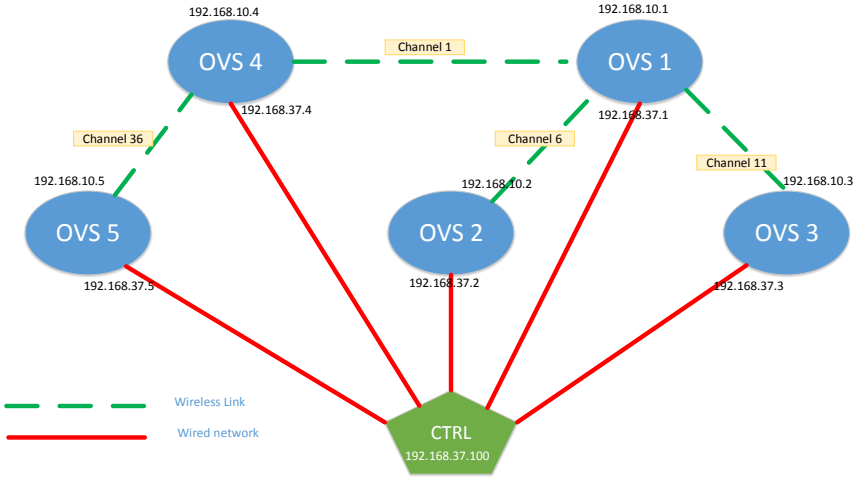


FIGURE 3.9: Contrôle Out-of-band.

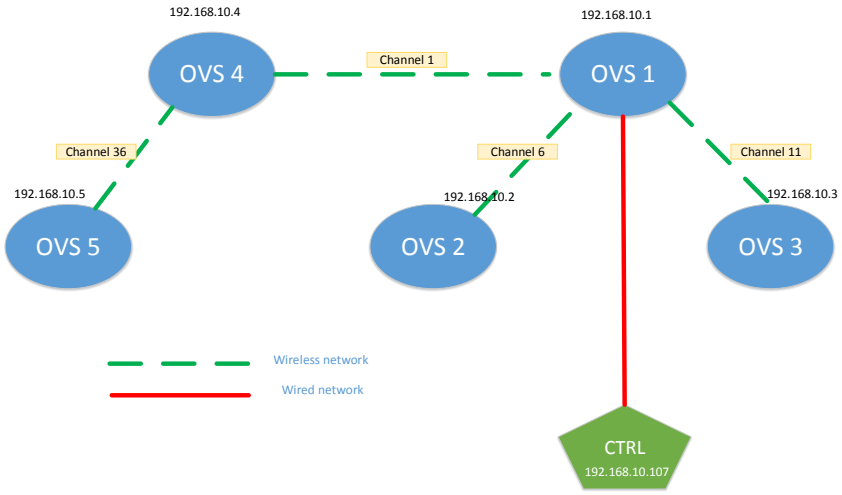


FIGURE 3.10: Contrôle In-band.

Un contrôle In-band

Dans ce type de configuration le trafic de contrôle (OpenFlow) et le trafic des données (Vidéo, Voix, ...) passent dans le même et seul réseau. Quoique cette configuration expose notre trafic de contrôle à des risques de pertes et des délais de livraison accrus, elle nous évite le déploiement et le maintien de deux réseaux (Figure 3.10). Le contrôleur peut être connecté à un noeud, qui va jouer le rôle d'une liaison entre ce contrôleur et tous les autres noeuds (comme décrit dans l'exemple de la Figure 3.10) ou être installé directement sur un noeud ou plusieurs de la plateforme.

Contrairement au contrôle « out-of-band », la mise en place de la connexion « In-band » demande plus de configurations et de mise en point au niveau de chaque noeud. L'on doit assurer une connectivité IP entre les commutateurs et le contrôleur fonctionnant sur TCP.

En effet chaque noeud du réseau doit pouvoir établir un chemin (route) vers le contrôleur à travers le noeud qui est directement relié à ce dernier. Les spécifications d'OpenFlow ne rentrent pas dans les détails du fonctionnement in-band qui est

laissé libre aux divers implémentations. On trouve plusieurs solutions :

Il est possible d'activer un protocole de routage ad-hoc (OLSR par exemple) qui se chargerait de l'établissement des chemins et l'acheminement du trafic de contrôle. Les routes du trafic de données seront bien évidemment défines par le contrôleur après l'établissement de la connexion IP et en cas de besoin.

Il est également possible d'établir un arbre de connectivité IP de tous les noeuds du réseaux vers le contrôleurs en installant des règles spécifiques sur ces noeuds.

Une autre solution consiste à configurer les cartes en mode "standalone" où chaque noeud se comporte comme un switch traditionnel ("NORMAL") en l'absence de connexion avec un contrôleur. En activant le mode in-band sur nos cartes des règles sont installées et des sessions IP sont établies entre chaque noeud et le contrôleur au dessus de réseaux de données de niveau 2 composés par les switch OpenFlow.

Les avantages qu'offre la configuration In-band se résument comme suit :

— Aucun port dédié : Pas besoin de dédier un port physique pour le contrôle, ce qui est important sur les noeuds qui ont peu de ports.
— Aucun réseau dédié : Pas besoin de construire et de maintenir un réseau de commande séparé. Ceci est important dans de nombreux environnements.

Photos des cartes déployées et poste de contrôle

Les figures suivantes (regroupées dans le tableau 3.1) présentent quelques photos des noeuds déployés au LAAS. La figure 3.11 décrit un noeud de la plateforme de type Avila Gatework. Les deux interfaces sans-fil sont localisées sur le haut de la carte et sont reliées à deux antennes omnidirectionnelles. Le câble jaune relie la carte au VLAN de gestion suscité. Il est également utilisé pour alimenter la carte avec le boitier d'alimentation placé dans le haut de la figure. La figure 3.12 décrit un autre noeud qui est de type Banana Pi R1, il n'a qu'une seule interface (carte wifi intégré dans la carte) connecté à une antenne omnidirectionnelle à la droite de la photo, la gestion de cette carte est assurée par le VLAN auquel elle est connectée à l'aide du câble Ethernet bleu, et son alimentation est assurée par le câble noir (mini USB). La figure 3.13 c'est un noeud de type Raspberry Pi 3, ça connectivité Wifi est assurée par le Wifi USB (avec le LED bleu sur la photo), et la connectivité au VLAN est assurée par le câble Ethernet en bleu. La figure 3.14 représente le poste de contrôle (la machine ou le contrôleur est installé), en-dessous de la table, il a deux interfaces Ethernet, une connectée au VLAN pour un contrôle de type « out-band » et l'autre a une carte (ici OVS11) dans le cas d'un contrôle de type « in-band ».

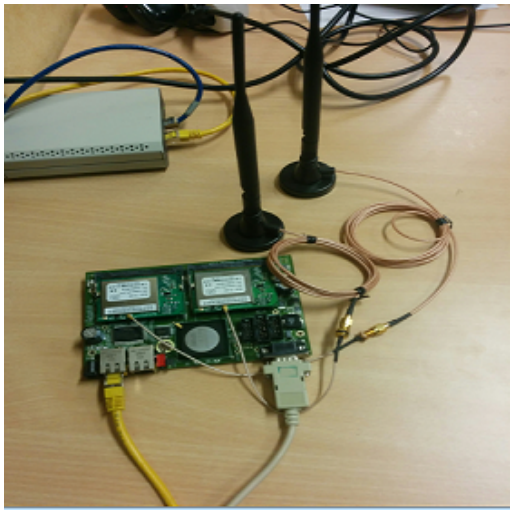


FIGURE 3.11: Avila



FIGURE 3.12: Banana Pi

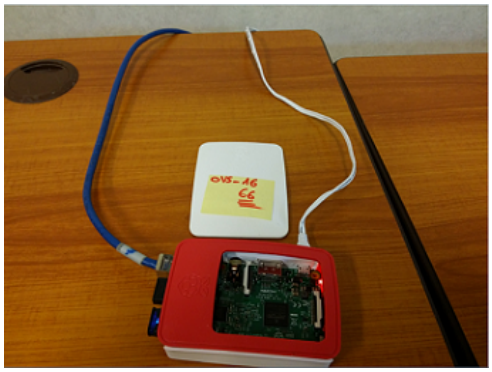


FIGURE 3.13: Raspberry Pi



FIGURE 3.14: Contrôleur

TABLE 3.1: Quelques éléments de la plateforme

Chapitre 4

Vers un service de découverte de topologie générique pour les Réseaux Multi-saut Sans-fil

Après avoir décrit la mise en place de la plateforme, ce chapitre présente le service de découverte de topologie implémenté par le contrôleur SDN. Il expose les insuffisances du service de découverte de topologie utilisé par les contrôleurs actuels pour une mise en application aux réseaux multi-sauts sans-fil, et nos propositions d'amélioration et d'extension.

4.1 Objectifs du service de découverte de topologie

Dans une architecture SDN on distingue des switch OF en charge de l'acheminement des flux et un contrôleur SDN, où est centralisé l'intelligence du réseau. Pour assurer convenablement les fonctions de contrôle sur le plan de données c'est-à-dire les switch OF, le contrôleur a besoin des informations à jour sur l'état du réseau [**Reference14**], en particulier sa topologie et c'est le rôle du service de découverte de topologie qu'il implémente. Et grâce à ce service le contrôleur possede une vision globale du réseau.

A travers ce travail nous cherchons à développer un service de découverte de topologie générique, c'est-à-dire qui peut être utilisé dans de divers autres fonctions réseau et qu'il soit compatible et opérationnel dans différents scénarios du réseau multi-sauts sans-fil. Cette optique de standardisation du service se traduit par des représentations réseau (au niveau du contrôleur) suffisamment exhaustives pour couvrir les besoins des différentes fonctions et algorithmes qui les instancient (routage, contrôle de topologie, affectation de canaux, gestion de mobilité,...).

En complément à cette généricité, le service proposé doit être en mesure de fournir (d'une façon continue et périodique, ou à la demande) des informations sur les noeuds formant la topologie du réseau (position, niveau de batterie, vitesse,...) et sur les métriques des liens les reliant (SNR, SINR, RSSI,...). Cette quête des paramètres de la topologie réseau nous donne une représentation plus complète et efficace du réseau.

Avec les performances plutôt modestes des connexions multi-sauts sans-fil, l'efficacité recherchée dans la représentation de la topologie réseau doit satisfaire des contraintes en termes d'économie (sur-débit réseau 'overHead' réduit). Cette économie est obtenue en jouant sur les périodes de mesure et de remontée des

attributs selon l'importance courante des noeuds et liens.

La représentation réseau que nous proposons est un graphe de connectivité entre noeuds du réseau où les arêtes représentent des liens sans fil (ou liaisons de données sans-fil au sens de la couche liaison de données du modèle OSI) reliant directement les deux noeuds. De ce fait, un lien sans-fil peut partager la capacité du canal radio (ou du lien physique (au sens de la couche physique du modèle OSI) multipoint sans fil) avec d'autres liens sans-fil dont les noeuds d'extrémité sont typiquement à proximité. Plusieurs attributs sont associés aux noeuds et aux liens, certains représentent des caractéristiques intrinsèques et d'autres des états ou des mesures de performance. Il est à noter qu'à un noeud du réseau, il est possible d'associer sa position, son niveau de batterie, sa vitesse ainsi que des informations relatives à son unité de calcul et sa taille mémoire. A une interface réseau sans-fil, il est possible d'associer sa technologie, le type d'antenne utilisé et ses caractéristiques (sensibilité, etc.), le canal radio sur lequel il opère, et la puissance d'émission. Enfin, à un lien sans fil, il est possible d'associer des métriques de mesure de la qualité ou performance du lien. Ces métriques sont primordiales pour les différentes fonctions réseau car elles permettent de mesurer et capturer les performances variables et peu prévisibles des liens. Elles sont donc utilisées pour choisir les routes les plus performantes, d'identifier des zones d'interférence et procéder à des réallocations de canaux ou un changement de topologie, etc.

La définition de métriques de liens sans-fil a fait l'objet d'une multitude de travaux de recherche ; En dépit du grand nombre de métriques proposées, aucune ne s'est imposée comme métrique polyvalente 'de-fait'. En revanche, il a été établi que les métriques devraient être choisies selon les exigences des applications supportées par le réseau, des propriétés du réseau et de l'environnement dans lequel il opère. En conséquence et comme évoqué ci-avant, la représentation réseau que l'on propose supporte plusieurs métriques de liens dont :

— un ensemble de métriques élémentaires que l'on retrouve dans une multitude de métriques de l'état de l'art. Il s'agit notamment : SNR, SINR, RSSI, df/dr, etc.

— certaines des métriques les plus connues de la littérature dont ETX et ETT.

4.2 Le service de découverte du contrôleur Ryu

Le service de découverte de topologie tel qu'il est défini par 'Ryu Controller' implique d'une part la découverte des noeuds (qui forme la topologie) et leurs caractéristiques (ID, nombre de ports, ...), et d'autre part, le suivi d'état de ces paramètres (Liveness, Changement de port,...). En plus de la découvertes des noeuds, et à l'aide du protocole OFDP (basé sur de LLDP), Ryu permet aussi de découvrir les liens qui relient ces noeuds [**Reference7**] [**Reference6**].
En projetant cette vision de la découverte proposée par Ryu avec celle que nous avons proposé ci-avant, nous constatons un manque, d'une part, en terme de récupération des métriques des liens (niveau de performance des liens) où Ryu n'implémente pas un service qui permet d'avoir et d'exploiter ce type d'information, et d'autre part, en terme d'informations périodiques et complémentaires sur les noeuds (position, vitesse, niveau de batterie...)

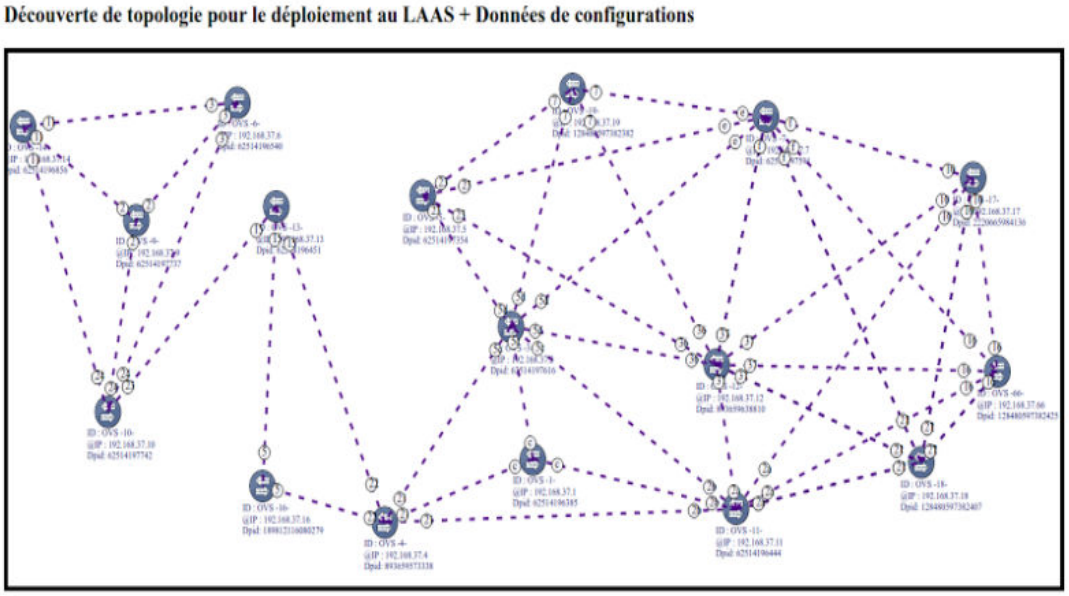


FIGURE 4.1: Visualisation de la plateforme déployée.

TABLE 4.1: Données Techniques des OVS 11 et 17.

	OVS 11	**OVS 17**
Type de carte	Avila	BananaPi R1
DatapathID	62514196444	2220665984136
Nombre de ports +(le port local)	2 + 1	1 + 1
Adresses MAC des ports	wlan0 : 00 :0E :8E :22 :F7 :0F wlan1 : 00 :0E :8E :22 :EF :DC	wlan0 : AC :A2 :13 :C0 :D4 :41
Liens	6 (dont unavec l'OVS 17)	4 (dont un avec l'OVS 11)

4.2.1 Quelques éléments sur la plateforme utilisée pour décrire le service de découverte

Nous nous reposons sur la plateforme (plus précisément, une portion (Figure 4.1)) déployée au LAAS et décrite au chapitre précédent. Nous nous placerons dans le contexte d'un transport de trafic de contrôle hors-bande ('out-band').

Dans ce qui va suivre, et pour pouvoir bien suivre les captures de trafic, nous allons focaliser l'étude de la découverte des noeuds et la découverte des liens sur les deux OVS 11 et 17 de la plateforme. Ces deux cartes qui partagent un même canal (qui ont une liaison directe) ont été choisies par rapport à leurs caractéristiques différentes (différents types de carte, différent nombre de ports/liens,...). Pour pouvoir bien suivre ce qui suit, le tableau 4.1 résume les caractéristiques de chaque carte.

4.2.2 Découverte des noeuds

Comme expliqué dans le chapitre 1 (spécification du protocole OpenFlow), c'est au noeud (OVS) d'initier la connexion vers le contrôleur (initiation d'une connexion TCP standard) car il est le premier à pouvoir joindre l'autre partie (adresse IP du contrôleur donnée aux OVS aux premières configurations). Après cela et après

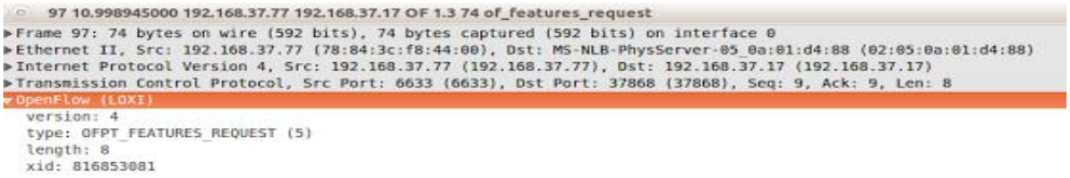


FIGURE 4.2: Requête OF_FEATRES_REQUEST (Ctrl vers OVS 17).

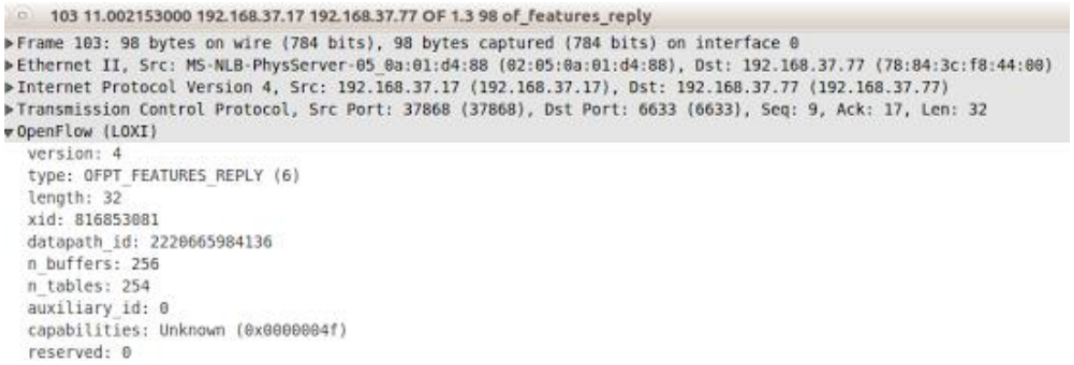


FIGURE 4.3: Réponse OFPT_FEATRES_REPLY (OVS 17 vers Ctrl).

la négociation de la version OpenFlow à adopter (avec un échange des paquets Hello), le découverte des caractéristiques des noeuds commence, elle est assurée par un échange des paquets 'OFPT_FEATURES_REQUEST' (initié par le contrôleur) et 'OFPT_FEATURES_REPLY' (réponse de l'OVS). Les Figures 4.2 et 4.3 (capture WireShark) illustre cet échange (entre le contrôleur et le OVS-17) et met en évidence le contenu des paquets.

Avec la Figure 4.3 de la capture de message OFPT_FEATURES_REPLY, les paramètres récupérés par le contrôleur à la fin de cet échange sont identifiables :

— Datapath_Id : l'identifiant unique du noeud.

— n_buffers : fixe le nombre de paquets maximal qui peut être mémorisé, lors de l'envoi d'un 'packet-in'.

— n_tables : indique le nombre de tables des flux supportées par le switch.

— Auxiliary-Id : étant à 0, elle indique que la connexion OpenFlow utilisée est la connexion principale

— Capabilities : Parmi les fonctionnalités prévues par le standard, il dresse celles qui sont supportées par le noeud.

Avec le même mécanisme de requête/réponse, le contrôleur récupère les informations relatives aux ports des OVS à l'aide du message OpenFolw 'OFPMP_PORT_DESC_STATS_REQUEST' (initié par le contrôleur) et 'OFPMP_PORT_DESC_STATS_REPLY' (réponse de l'OVS). Les captures qui suivent illustrent cet échange (Contrôleur / OVS17).

— Avec la capture du message OFPMP_PORT_DESC_STATS_REPLY de l'OVS 17 (Figure 4.5), nous arrivons à voir les (1+1) ports du noeud (ports Wlan + port local), comme indiqué dans le tableau 4.1

— Port no : numéro de port dans le noeud

— Hw addr : l'adresse Mac du port

— Name : Le nom du port choisi par le système

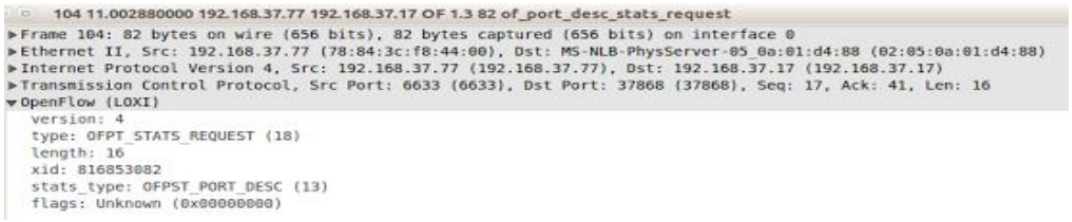


FIGURE 4.4: Requête OFPMP_PORT_DESC_STATS_REQUEST (Vers OVS17).

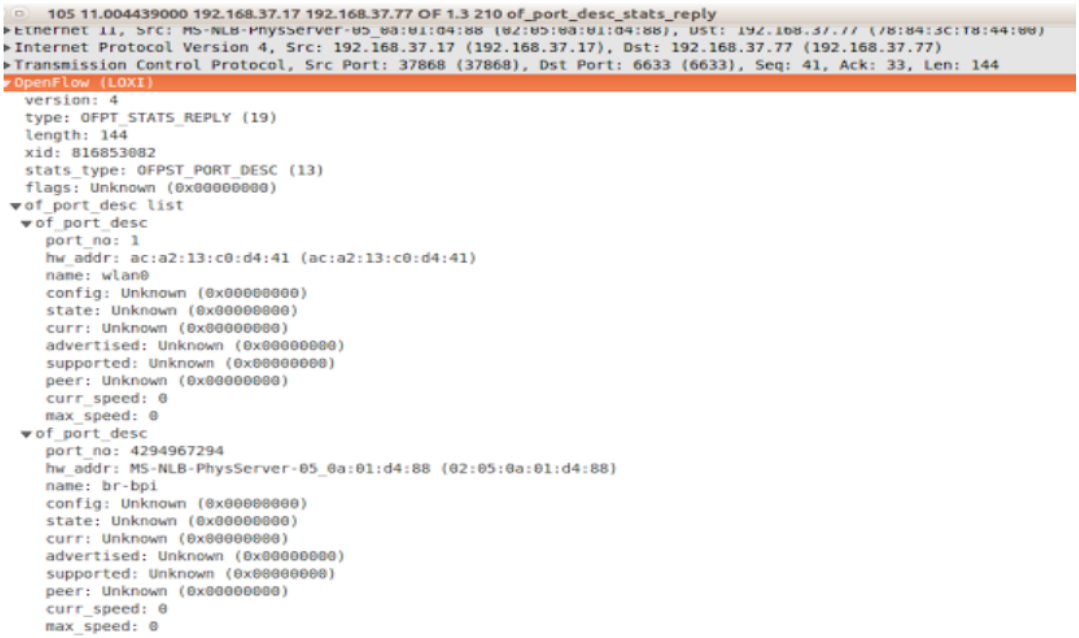


FIGURE 4.5: Réponse OFPMP_PORT_DESC_STATS_REPLY (d'OVS17)..

— Config : qui indique la configuration courante du port dont, une valeur à 0 indique une configuration active du port en régime nominal (actif avec les droits pour recevoir, relayer et remonter vers le contrôleur)

— State : indique l'état du lien physique 'Down' (lien physique non présent), 'Blocked' (bloqué par un autre protocole tel que STP (Spanning Tree Protocol)), 'Live' (pour une utilisation avec des mécanismes de recouvrement de type 'Fast Failover').

— suivi d'une liste, potentiellement nulle, de propriétés du port ('Port Description Properties' dans Openflow) décrivant diverses configurations ou états du port. (relatives à un port de type Ethernet) :

  — 'Current', 'advertised', 'supported' and 'peer' correspondent respectivement aux caractéristiques de port (features dans la terminologie Openflow) courantes, annoncées et supportées par le port et celles annoncées par le port pair. Une feature combine le mode de fonctionnement du lien (duplexité + débit), le type de lien (paire ou fibre) et les possibilités d'autonégociation, support de trames PAUSE. En l'absence d'indication, ce sont les valeurs du max_speed ou curr_speed qui sont utilisées.

  — 'curr_speed' et 'max_speed' qui indiquent respectivement le débit courant et maximum en kbps. Ces valeurs peuvent être différentes car en dépit des propriétés d'un port, le débit qui est choisi prend en compte les possibilités du port pair. Dans la capture de trafic présentée à la Figure 4.5, ces valeurs sont indéfinies. Ceci s'explique par le fait que l'interface wlan1 n'est pas une interface Ethernet.

4.2.3 Suivi de la présence et des caractéristiques des noeuds

Tout au long du fonctionnement du SDN, le contrôleur doit garder une vision 'temps réel' sur la topologie du réseau, et il doit être informé de tout changement apporté au cours de son exécution. Ce suivi est assuré dans Ryu de deux manières différentes (un suivi continu et un suivi évènementiel) :

— ***Le suivi continu*** : Avec ce type de suivi Ryu s'assure de l'existence (Liveness) des noeuds tout au long de la période du fonctionnement. Cette fonctionnalité est assurée par les messages du protocole OpenFlow 'OFPT_ECHO_REQUEST', 'OFPT_ECHO_REPLY', ils sont initiés d'un côté comme de l'autre et présente plusieurs buts d'utilisation (les comportements autres que ceux présentés dans cette section sont décrits en Annexe C). Pour prouver son existence (encore en vie) un OVS envoi un OFPT_ECHO_REQUEST chaque période de 2500 ms (peut être changé selon le besoin), ces messages seront envoyés uniquement si aucun autre message n'a été envoyé durant cette période. Les captures dans les Figures 4.6 et 4.7 illustrent cet échange.

— ***Le suivi événementiel*** : Cet autre type de suivi est rattaché à un événement de changements (rajout/suppression/modification/') des caractéristiques d'un noeud « OVS » préalablement déclarées au contrôleur, comme par exemple une interface qui tombe dans un OVS donné. Pour bien mettre en évidence ce comportement nous avons provoqué la chute (ifconfig wlan0 down) d'une interface dans l'OVS17 et capturé son comportement. La Figure 4.8 illustre cet exemple.

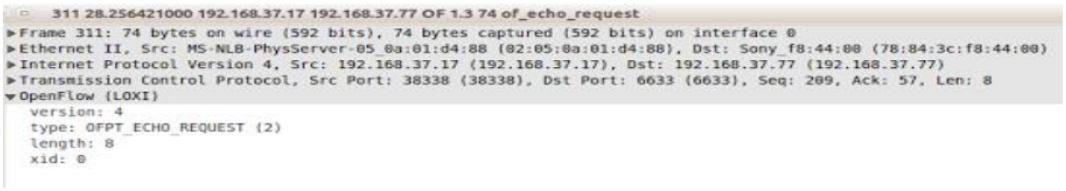


FIGURE 4.6: Paquet OFPT_ECHO_REQUEST (de l'OVS17 vers le Ctrl).

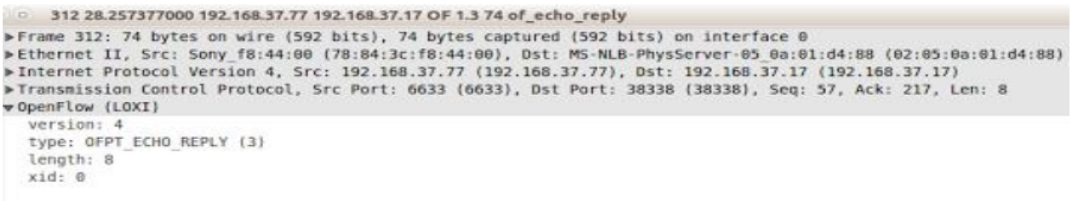


FIGURE 4.7: Paquet OFPT_ECHO_REPLY (du Ctrl vers OVS17).

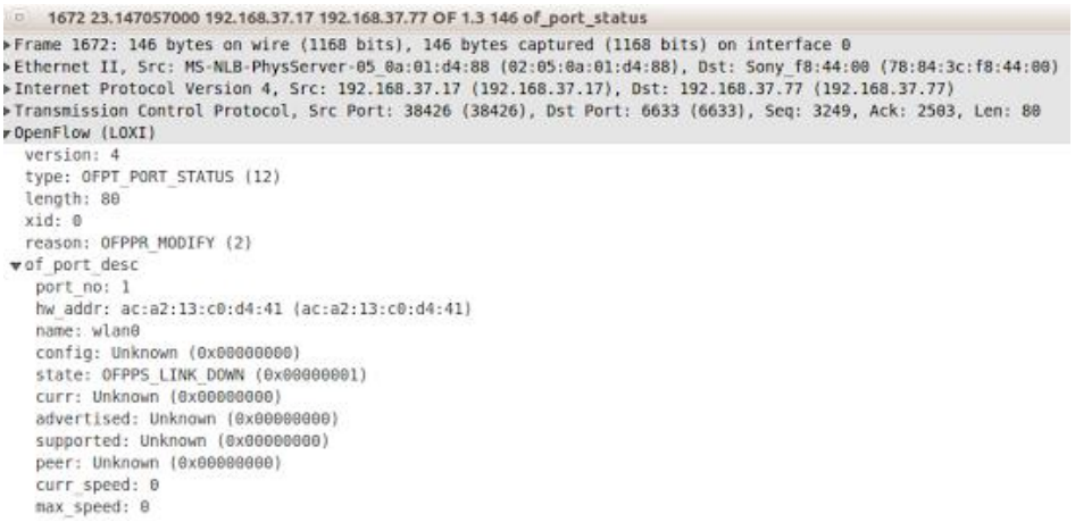


FIGURE 4.8: paquet OFPT_PORT_STATUS (de l'OVS17 vers le Ctrl).

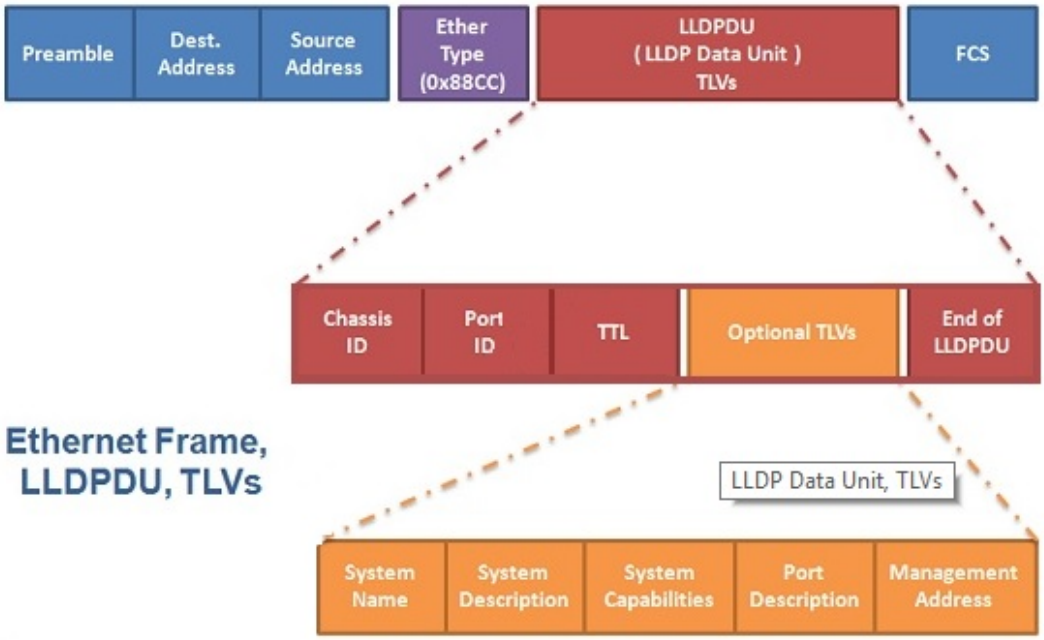


FIGURE 4.9: Trame LLDP.

4.2.4 Découverte et suivi des liens

Dans le contexte OpenFlow SDN, il n'existe pas un standard officiel qui définit la découverte et le suivi des liens, toutefois la plupart des contrôleurs SDN implémentent cette fonctionnalité via des mécanismes similaires et qui se basent sur les trames LLDP. Ce mécanisme est appelé dans la littérature OFDP [**Reference7**] (OpenFlow Discovery Protocol).Nous adopterons également cette appellation dans le reste du document. OFDP est une adaptation du protocole LLDP (Link Layer Discovery Protocol) [**Reference8**] aux spécificités d'une découverte de liens dans le contexte d'un réseau SDN.

4.2.5 LLDP

LLDP [**Reference9**] [**Reference10**] pour (Link layer discovery protocol) est défini par le standard IEEE sous le nom 802.1AB, c'est un protocole de couche liaison de données du modèle OSI (protocole de niveau 2), Il est directement véhiculé dans une trame et conçu pour fonctionner sur les réseaux de type IEEE 802 (Ethernet, '). Il s'agit d'une simple émission continue des messages (LLDP) à intervalle régulier (Ce n'est pas une communication bidirectionnelle), il peut fonctionner sous une forme centralisée où toutes les données sont regroupées dans une entité centrale (qui prend en charge la gestion de toutes les informations), ou sous une forme 'indépendante' où chaque noeud construit sa propre vision des liens qui le relient à ses voisins directs. Chacun de ces messages est composé d'une suite de structures appelées TLV (Type-Length-Value) servant à contenir les informations et d'un propre EtherType (0x88CC) qui est donc indiqué dans le champ « Type » de la trame Ethernet.

La trame LLDP est envoyée dans une trame Ethernet, et elle n'a pas d'entêtes spécifiques. La Figure 4.9 présente la PDU LLDP et ses champs TLV (Type Length Value). Comme dans toute encapsulation dans des trames Ethernet, nous avons les adresses mac de la source qui contient l'adresse de la machine émettrice et nous avons l'adresse de la destination où elle contient toujours une des valeurs suivantes

TABLE 4.2: Spécification des champs de la trame LLDP.

TLV Type	TLV Name	Description
Mandatory TLVs (obligatoires)		
000 0000	End-of-LLDPDU	Fin des TLVs
000 0001	Chassis ID	ID du noeud émetteur
000 0010	Port ID	ID du port de transmission
000 0011	Time-to-live	Durée de vie de la trame
Basic Management TLV Set (optionels)		
000 0100	Port Description	Description textuelle du port
000 0101	System Name	Nom de la machine émettrice
000 0110	System Description	Description de la machine émettrice
000 0111	System Capabilities	Fonctionnalités de la machine émettrices
000 1000	Management Address	Adresse IP de management de la machine
000 1001 - 111 1110	Reserved for future standardization	/
111 1111	Organizationally Specific TLVs	/

01 :80 :c2 :00 :00 :0e, ou 01 :80 :c2 :00 :00 :03, ou 01 :80 :c2 :00 :00 :00 (Multicast), suivi par le type de la trame, dans notre cas c'est '0x88CC'. Le tableau 4.2 récapitule les champs TLV et leurs rôles : La trame comporte des 'Basic Management TLV Set' pour les IEEE 802.1 et IEEE 802.3 Organizationally Specific TLV Set, (Il n'y a pas d'adaptation du LLDP pour la norme 802.11).

4.2.6 OFDP

Comme indiqué ci-avant, OFDP [**Reference11**] repose sur du LLDP en l'adaptant pour un fonctionnement sur un réseau SDN. Dans ce type de réseau (SDN) les switches OpenFlow (OVS) n'ont aucun mécanisme qui permet de découvrir 'toute' la topologie (dans certains cas ils ne se limitent qu'à la découverte de leurs voisins directs), il est donc de la responsabilité du contrôleur d'implémenter ce type de service, toutefois, les OVS ont des mécanismes prédéfinis qui leurs permettent de collaborer avec le contrôleur afin d'assurer un bon fonctionnement de ce service.

Les switches OpenFlow (OVS) peuvent fonctionner sous deux modes 'LLDP enabled' ou 'LLDP disabled'. Le mode LLDP enabled est un mode qui permet aux OVS de générer des packets LLDP et découvrir ses voisins directs mais il lui est interdit de forwarder ces messages par le protocole OpenFlow (impossible de transférer vers le contrôleur), ce mode ne peut donc pas être fonctionnel lors de la présence d'un contrôleur dans le réseau (en SDN). L'autre mode 'LLDP disabled' est le mode par défaut des OVS (en présence d'un contrôleur). Avec ce mode de fonctionnement les OVS ont une règle préinstallée qui leur dit d'envoyer tous les paquets LLDP vers le contrôleur (dans cette configuration le contrôleur est le seul générateur des paquets LLDP et les OVS n'ont pas le droit de les générer ni de les modifier).
Grâce au mécanisme de découverte des noeuds (réalisée par le protocole Openflow et décrit ci-avant), le contrôleur a connaissance des différents noeuds et des ports de chaque noeud. Pour assurer la découverte de tous les liens du réseau, le contrôleur transmet à chaque noeud un message Openflow 'packet-out' par port actif. Ce dernier inclut un message LLDP décrivant les caractéristiques du noeud et, via une

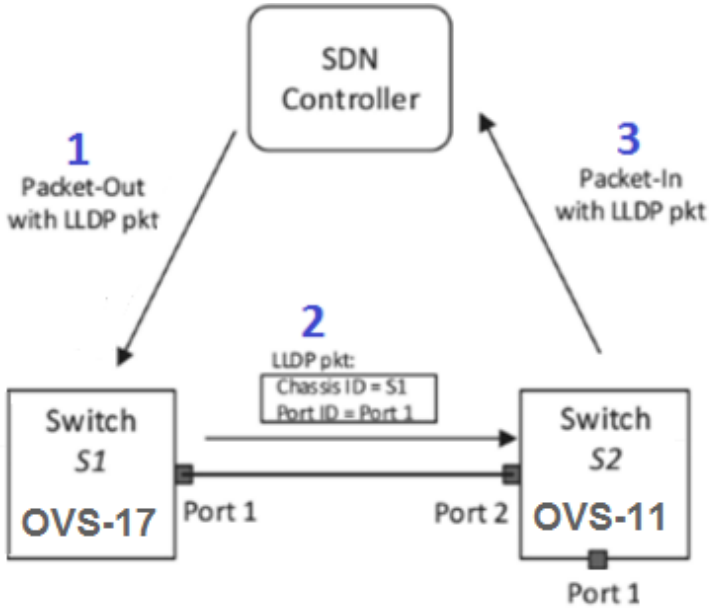


FIGURE 4.10: Exemple de la circulation de la trame LLDP entre OVS17 et OVS11.

règle Openflow véhiculée dans le 'packet-out', demande explicitement au noeud de relayer le message LLDP sur le port choisi. Un noeud qui reçoit un paquet LLDP se doit de le remonter au contrôleur en l'encapsulant dans un message Openflow 'packet-in'. Une règle Openflow doit donc être installée au niveau des noeuds pour les messages LLDP avec comme traitement de les remonter vers le Controlleur.
Pour bien comprendre le fonctionnement de l'OFDP nous allons prendre l'exemple de l'OVS 11 et 17 (présentés ci-avant). Avec la Figure 4.10 où le noeud S1 représente l'OVS17 et le noeud S2 représente l'OVS11, nous illustrons le mécanisme décrit ci-avant. (Pour simplifier la compréhension du mécanisme les deux OVS 11 et 17 ont été isolés du reste du groupe).

1. Le contrôleur génère une trame Packet-out vers l'OVS 17 dans laquelle est encapsulé la trame LLDP. Figure 31 décrit la trame. On y retrouve le message LLDP avec la description de l'OVS17 et la règle Openflow qui demande de transmettre le paquet joint sur le port numéro 1. Cf. Figure 4.11
2. l'OVS17 applique la règle et fait sortir la trame LLDP (dés-encapsulé de la trame Paquet-out et encapsulé dans le protocole de niveau 2 (Ethernet)) par le port 1. les figures 32 et 33 des captures faites dans le port 1 de l'OVS 17 illustre cette étape du mécanisme où on voit clairement la trame LLDP ' seule' sortir du port1. Cf. Figure 4.12 et 4.13
3. A la réception du message LLDP, l'OVS 11 (S2) applique la règle préinstallée dans les OVS et remonte le paquet LLDP vers le contrôleur dans un message Openflow 'packet-in'. La Figure 4.14 illustre le paquet.
4. Finalement, le contrôleur en recevant ce paquet retrouve la trame LLDP qu'il avait envoyé à l'OVS 17 via l'OVS11 'S2' et Il conclut sur l'existence d'un lien direct entre 'OVS17', port 1 et 'OVS11', port 2.

Le suivi de l'état des liens repose sur l'envoi périodique par le contrôleur de messages LLDP selon le schéma décrit ci-avant. Tant que les messages reviennent vers le contrôleur en deçà d'un temps d'aller-retour prédéfini, le lien est considéré actif. Dans le contrôleur RYU, la période d'émission des messages LLDP est fixée

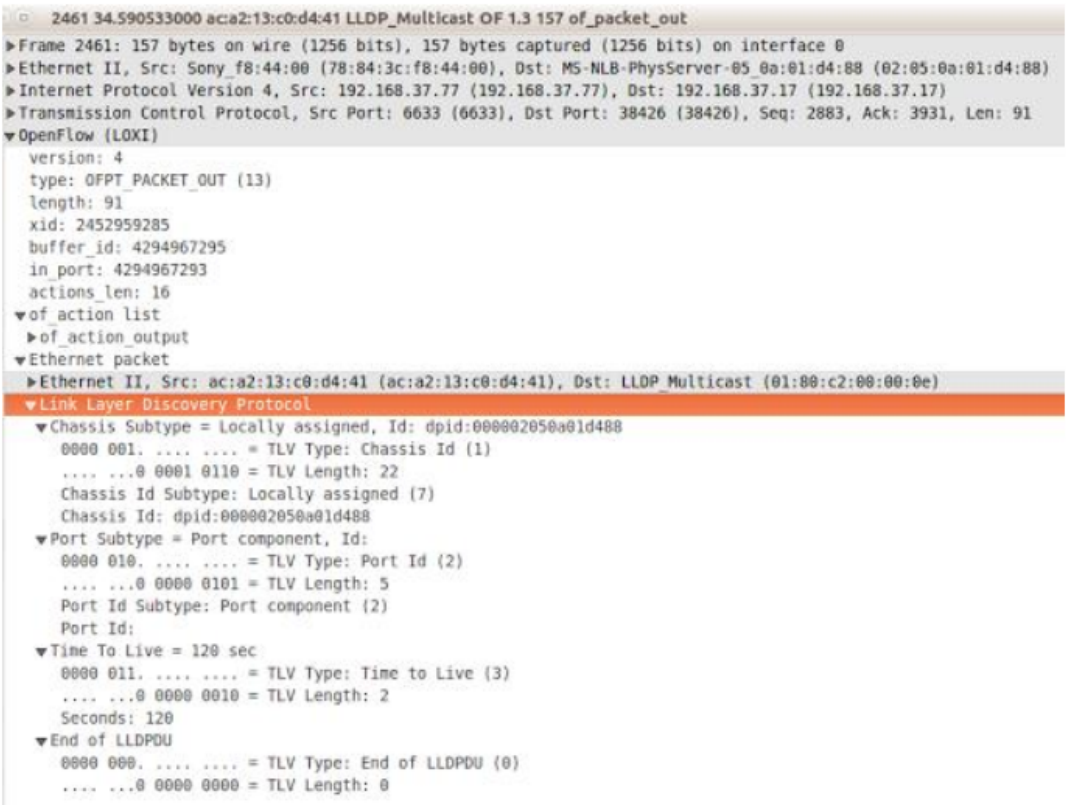


FIGURE 4.11: Paquet 'Packet_out LLDP' (du Ctrl vers OVS17).

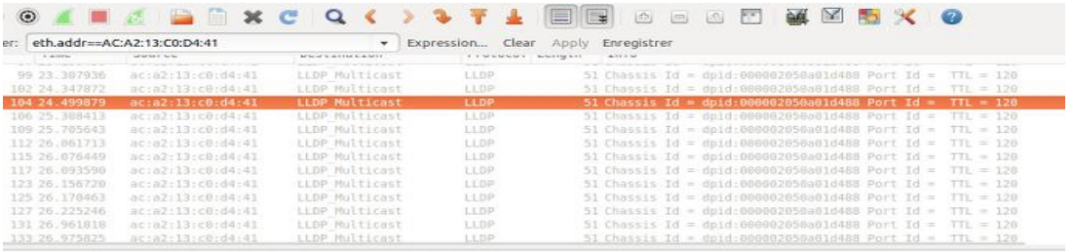


FIGURE 4.12: Capture de la trame LLDP au niveau de l'OVS17.

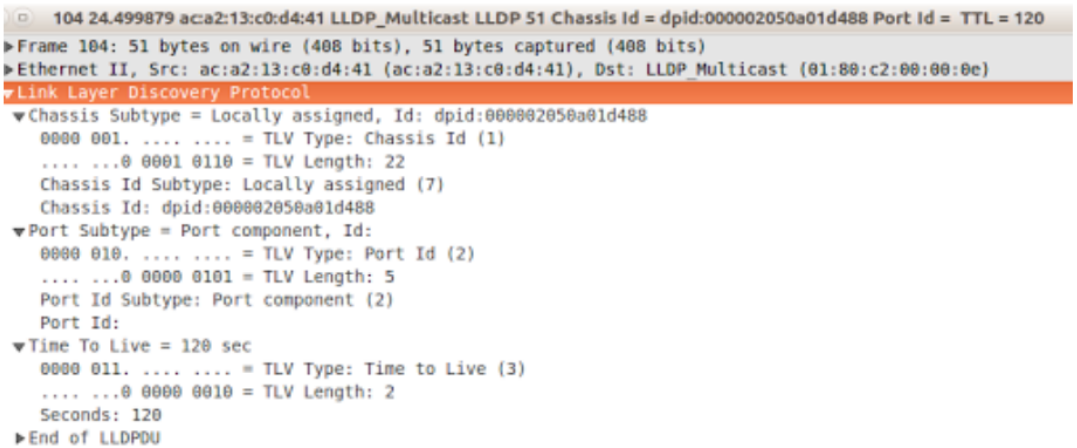


FIGURE 4.13: Paquet 'LLDP' (de l'OVS17 vers OVS11).

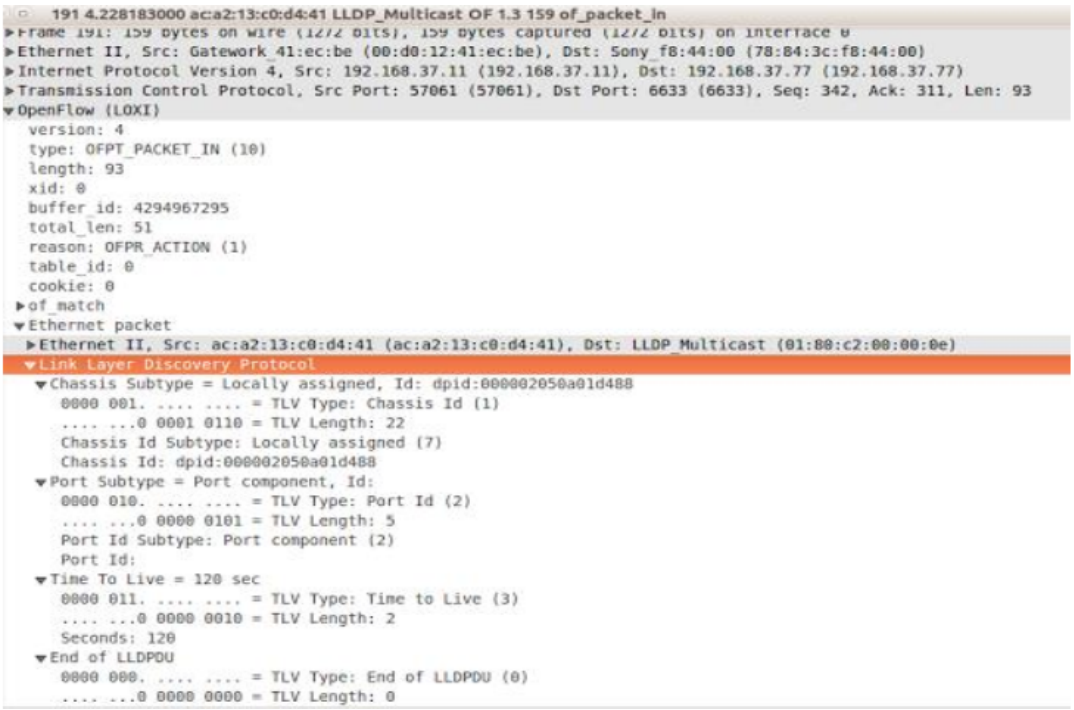


FIGURE 4.14: Paquet 'Packet_IN LLDP' (de l'OVS17 vers le Ctrl).

à 0.9 s. Par défaut, dix pertes successives de messages LLDP impliquent la perte du lien. Bien évidemment, la période et le nombre de pertes sont des paramètres ajustables. Une autre situation où le lien est considéré obsolète est lorsque le message 'packet-in' remonte depuis un nouveau noeud.

4.3 Analyse des limites du service de découverte de Ryu pour les réseaux multi-saut sans-fil

Au terme des besoins exprimés ci-avant, le contrôleur Ryu présente des limitations qui peuvent être résumées en deux points :

1. Le premier point résume le manque en termes d'informations sur les noeuds, où plusieurs paramètres importants dans un contexte de connexion multi-sauts sans-fil manquent à l'appel, parmi ces paramètres on peut citer la position, le niveau de batterie, les canaux dans lesquelles opèrent les différents ports, type d'antenne et autres. Sachant que ce type de paramètre est très dynamique et nécessite un rafraichissement périodique.
2. Le deuxième point regroupe à la fois le manque des métriques relatives aux liens, et l'incapacité quant à la gestion des connexions multipoints.
   (a) Ryu (dans sa configuration actuelle) n'est qu'en mesure de déceler la présence ou pas d'un lien ; il ne peut en aucun cas fournir des informations relatives à la qualité de ce dernier. Cette limitation est valable pour tous les contrôleurs SDN actuellement utilisés.
   (b) Les liens sans-fil dans un réseau type Ad-hoc/Mesh, ont tendance à être des liens multipoints La transmission d'un message LLDP sur un port sans-fil d'un noeud donné peut être reçu par plusieurs noeuds opérant sur le même canal. De ce fait, plusieurs messages Openflow 'packet-in'

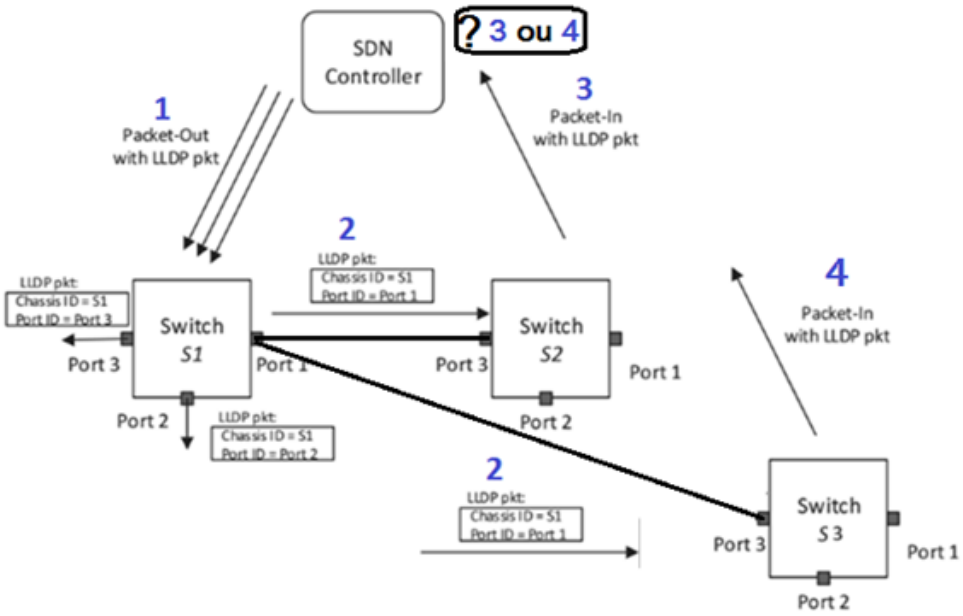


FIGURE 4.15: .Illustration du problème des connexions multipoints

liés à un même message LLDP peuvent remonter au contrôleur depuis ces différents noeuds. Chaque message reçu par le contrôleur RYU sera interprété comme un changement de lien dû à un changement de connectique (du fait qu'il ne connaît que les liaisons point-à-point). Ainsi, le service de découverte de RYU notifie une succession de changements de liens alors qu'il aurait dû découvrir la présence de plusieurs liens entre chaque couple de noeuds. la Figure 4.15 illustre la problématique.

4.4 Propositions d'extensions du service de découverte de RYU

A travers cette section nous allons proposer des solutions afin de répondre aux limitations (du contrôleur Ryu) présentées ci-dessus.

4.4.1 Prise en compte des attributs de noeuds et de ports

Le protocole Openflow dispose d'une procédure 'OFPT_FEATURES_REQUEST' et 'OFPT_FEATURES_REPLY' permettant de découvrir les caractéristiques des noeuds (présentée précédemment), il est donc pertinent de voir si on ne peut pas l'adapter pour qu'il puisse réponde à nos besoins. Cet échange qui intervient qu'une seule fois dans la vie d'une connexion contrôleur/OVS, et qui se présente sous la forme illustrée sur les Figures 4.2 et 4.3, ne peut être modifié pour répondre à nos besoins (l'ajout des attributs) pour deux raisons :

1. Fréquence d'envoi du message : Nous avons besoin de transmettre des informations en continu, et les messages (OFPT_FEATURES_REQ/REP) ne sont échangés qu'une seule fois.
2. Taille de la trame : La taille de la trame est figée, et ne peut pas contenir toutes les informations que nous voulons transmettre. (Le champ 'reserved' sur 32 bits et le masque capabilities sur 32 bits sont insuffisants pour supporter ces attributs)

```
Features Openflow d un noeud

/* Switch features. */
struct ofp_switch_features {
struct ofp_header header;
uint64_t datapath_id; /* Datapath unique ID. The lower
48-bits are for a MAC address, while
the upper 16-bits are implementer-                    defined. */
uint32_t n_buffers; /* Max packets buffered at once. */
uint8_t n_tables; /* Number of tables supported by datapath.*/
uint8_t auxiliary_id; /* Identify auxiliary connections */
uint8_t pad[2]; /* Align to 64-bits. */
/* Features. */
uint32_t capabilities; /* Bitmap of support "ofp_capabilities". */
uint32_t reserved;
};
```

```
/* Capabilities supported by the datapath. */
enum ofp_capabilities {
OFPC_FLOW_STATS = 1 << 0, /* Flow statistics. */
OFPC_TABLE_STATS = 1 << 1, /* Table statistics. */
OFPC_PORT_STATS = 1 << 2, /* Port statistics. */
OFPC_GROUP_STATS = 1 << 3, /* Group statistics. */
OFPC_IP_REASM = 1 << 5, /* Can reassemble IP fragments. */
OFPC_QUEUE_STATS = 1 << 6, /* Queue statistics. */
OFPC_PORT_BLOCKED = 1 << 8, /*Switch will block looping ports.*/
OFPC_BUNDLES = 1 << 9, /* Switch supports bundles. */
OFPC_FLOW_MONITORING = 1 << 10, / * Switch supports flow monitoring. */
};
```

L'étude des spécifications du protocole OpenFlow nous a amené à proposer deux pistes afin de répondre à ce besoin :

1. Exploitation des messages 'OFPT_EXPERIMENTER' ; ce sont des messages relatifs au protocole OpenfLow qui peuvent être initiés d'un côté comme de l'autre (OVS ou contrôleur), prévus par Openflow pour définir des extensions à Openflow (d'où le nom 'experimenter').
2. Exploitation des messages OFPT_ECHO_REQUEST ; Ces messages peuvent être initiés par les OVS à tout moment, et peuvent contenir une taille variable de données.

Similairement et dans le but de satisfaire le besoin en terme de métriques des liens, Openflow dispose d'une procédure permettant de décrire les caractéristiques d'un port. Comme décrit dans la section 3.1, elle se base sur l'échange de messages Openflow de type 'OFPT_MULTIPART_REQUEST' et 'OFPT_MULTIPART_REPLY' qui encapsule une structure 'OFPMP_PORT_DESC' qui décrit les caractéristiques du port. En l'état de la version 1.5.1 de OpenFlow, seuls les des ports Ethernet et Optiques ont été considérés. Les travaux menés sur l'utilisation du paradigme SDN sur les réseaux sans-fil militent pour l'intégration dans les prochaines versions du standard Openflow une description des ports sans-fil (et notamment de type 802.11, utilisé dans ce travail). A défaut, nous proposons notre propre description de ce type de port avec les attributs que nous avons identifiés en utilisant le 'Openflow Port Description Property' de type 'Experimenter'. La structure que nous proposons est définie ci-dessous.

```
Les types de descr de port par la version 1.5.1 de OpenFlow

/* Port stats property types.
```

```
*/
enum ofp_port_stats_prop_type {
OFPPSPT_ETHERNET = 0, /* Ethernet property. */
OFPPSPT_OPTICAL = 1, /* Optical property. */
OFPPSPT_EXPERIMENTER = 0xFFFF, /* Experimenter property. */
};
```

```
Description des ports de type Wifi

/* wifi port description property . */
struct ofp_port_desc_prop_wifi {
uint16_t type ; /* OFPPDPT_WIFI . */
uint16_t length ; /* Length in bytes of the property . */
uint8_t pad [4]; /* Align to 64 bits . */
/* Bitmaps of OFPPF_ * that describe features . All bits
zeroed if unsupported or unavailable . */
uint32_t channels_curr ; /* Current used channels */
uint32_t channels_supported ; /* Supported channels */
uint32_t phy_modes_curr ; /* Current used PHY modes */
uint32_t phy_modes_supported ; /* Supported PHY modes */
uint16_t antenna_sensitivity ; /* antenna sensitivity */
uint16_t tx_pwr_min ; /* Minimum TX power */
uint16_t tx_pwr_max ; /* Maximum TX power */
uint16_t tx_pwr_curr ; /* Current TX power */
}
```

Certains attributs d'un port Wifi peuvent changer dans le temps. C'est par exemple le cas du canal sur lequel opère l'interface ou de la puissance d'émission. Contrairement aux attributs des noeuds qui sont supposés statiques, Openflow a défini un mécanisme pour pouvoir notifier au contrôleur le changement des caractéristiques d'un port. Il s'agit du mécanisme basé sur le message Openflow de type 'OFPT_PORT_STATUS' décrit ci-avant.

4.4.2 Prise en compte de liens multipoints sans-fil dans le service de découverte de RYU

La gestion des liens multipoints était identifiée comme une limitation dans le contrôleur Ryu, de ce fait, dans cette section nous allons présenter l'algorithme que suit Ryu dans le traitement d'une telle situation (connexion multipoints) ainsi qu'un alogorithme permettant de remédier à cette limitation.

Algorithme actuellement utilisé par RYU

La gestion des liens multipoints était identifiée comme une limitation dans le contrôleur Ryu, de ce fait, dans cette section nous allons présenter l'algorithme que suit Ryu dans le traitement d'une telle situation (connexion multipoints) et proposer un algorithme pour remédier à cette limitation.

```
L1
Wait for Event PACKET_IN:
        SI LLDP
                Src = GetFromLLDPMsg
                Dst = GetFromLLDPMsg
                old_peer =GetPeerHowHaveThis (Src)(s'il existe)
                SI old_peer != None ET old_peer.Dst != Dst
                        old_link = GetLinkFromOldPeer
                        Delete Old_link From ALL_LINKS
                        Set Event Delete Link
LOOP  L1
```

Algorithme modifié

Dans le but de rendre la découverte de topologie de Ryu fonctionnelle pour les réseaux Ad-hoc deux approches ont été proposées et implémentées : La première solution est de modifier l'algorithme de tel sorte qu'un événement ancien lien, (Old_link_Event), n'est enclenché que si un lien considéré comme existant, (présent dans la liste des liens All_links de Ryu), n'est pas mis à jour au bout d'un nombre de paquet LLDP émis. Avec ce comportement nous assurons une gestion des connexions multipoints. La deuxième solution consiste à désactiver la fonctionnalité des vérifications des anciens liens, qui n'est en aucun cas intéressante dans les réseaux Ad-hoc, et de ne laisser que le TimeOut comme mécanisme de suppression des liens.

4.4.3 Prise en compte de métriques de qualité du lien

Le bon fonctionnement du service de découverte de topologie en termes de récupération des métriques des liens nécessitait d'une part la présence d'agents (processus) dans les OVS qui vont prendre en charge l'évaluation des différentes mesures de la qualité des liens ; et d'autre part, de remonter ces mesures vers contrôleur. Le fait d'avoir un processus pour la récupération des informations des métriques des liens est une thématique largement abordée dans la littérature scientifique relative au routage dans les réseaux multi-sauts sans-fil, c'est pour cela que dans notre travail nous allons nous focaliser que sur la partie de remonté des données (de OVS vers le contrôleur).

Trois options ont été étudiées, une première basée sur l'extension d'OFDP, une deuxième sur l'utilisation d'agent SNMP et enfin une troisième utilisant l'échange de messages OFPT_ECHO_REPLY. Les paragraphes suivants détaillent ces propositions et comparent cela à la bibliographie et les expérimentations menées par d'autres chercheurs.

Extension de OFDP : LLDP et agents OVS

On cherche à étendre le protocole OFDP, solution très intéressante, puisque OFDP a pour objectif de permettre au contrôleur de découvrir et suivre l'état des liens. Ainsi, avec cette extension, le contrôleur pourra découvrir, suivre l'état et la qualité des liens. Le principe général de cette solution est de profiter des paquets LLDP qui circulent d'une façon périodique dans le réseau, et d'y ajouter les métriques relatives au lien en cours de détection ou préalablement détecté. cette méthode met en avant la nécessité pour le noeud récepteur du message LLDP d'adjoindre à ce dernier un ou plusieurs TLVs transportant la (les) mesure(s) de la qualité des liens avant d'encapsuler l'ensemble dans un 'packet-in'. Même si le principe semble simple sa mise en oeuvre soulève des difficultés. En effet, avec OFDP, un 'Open vSwitch' est, par défaut, en mode 'LLDP disabled' (se référer à la section 3.3) donc incapable d'analyser et de traiter le paquet LLDP. D'un autre côté, le mode 'LLDP enabled' (qui analyse et traite les paquets LLDP) est à son tour inadapté car interdisant le relayage des messages LLDP vers un contrôleur (en plus il peut être source de messages LLDP). Ainsi, avec cette proposition, un nouveau mode de fonctionnement d'Open vSwitch' intermédiaire doit être défini. Dans ce mode : un noeud ne peut générer un message LLDP ; A la réception d'un message LLDP, le noeud doit l'analyser pour identifier le lien sans-fil concerné, le modifier pour inclure les TLVs relatifs à la mesure de la qualité de lien, avant de le relayer via la session Openflow dans un message 'packet-in'.

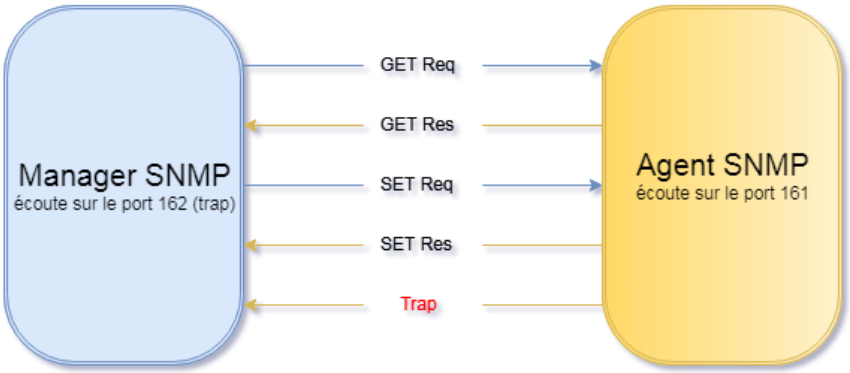


FIGURE 4.16: Interactions entre manager et agent SNMP.

SNMP

a. Présentation du protocole SNMP

SNMP **[Reference12]**, pour Simple Network Management Protocol, est un protocole de la couche 6 du modèle OSI qui permet de collecter des données sur des équipements (postes de travail, imprimantes, switch, routeurs ') sur un réseau IP pour avoir des informations sur leur état mais aussi de définir le comportement de ces équipements en modifiant quelques informations spécifiques. Le SNMP permet ainsi à un administrateur réseau de gérer son parc informatique en lui fournissant les fonctions de supervision, de diagnostic '
Une architecture SNMP est constituée d'un manager SNMP, côté administrateur, et des agents SNMP sur tous les équipements à « surveiller ».
L'agent SNMP joue le rôle de serveur et écoute sur le port UDP 161, lorsqu'il recevra une requête de la part du client, le manager, il y répondra convenablement. Une requête pouvant être « Quelle est la température de ton CPU ' »
L'agent SNMP peut également envoyer des informations au manager sans que ce dernier n'ait fait une demande, ceci par le bien des messages « traps », messages d'alerte. Si l'on continue avec l'exemple de la température du CPU, une alerte pourra être émise par l'agent lorsque cette température dépasse un certain seuil. Le manager dispose ainsi d'une fonction serveur, qui écoute sur le port UDP 162, pour traiter ces alertes.
La Figure 4.16 recapitule le fonctionnement decrit ci-dessus.

Les données à récupérer(GET) ou à modifier (SET), pour configuration, sur les équipements sont présentées au niveau de l'agent par des variables. Ces variables, identifiées par des OID (Object Identifier), sont organisées de manière hiérarchique. Les informations contenues dans ces variables sont structurées par le SMI Structure Of Management Information un sous ensemble de l'ANS.1 (Abstract Syntax Notation One) qui est un langage de description pour définir des structures de données qui peuvent être sérialisées et désérialisées de manière standardisée (multiplateforme). Il est largement utilisé dans les télécommunications et les réseaux informatiques. L'ensemble des informations au niveau de l'agent sont regroupées dans une base de données appelée MIB Management Information Base, qui se charge également faire la traduction entre les OIDs (qui sont numériques ex : 1.3.6.1.1.1.4) dans une forme plus lisible (« iso.org.dod.internet.directory.mib-2.ip » ou tout simplement « ip »)

Il est à noter qu'un OID est une paire clé-valeur.

La Figure 4.17 montre un exemple d'arbre d'OIDs

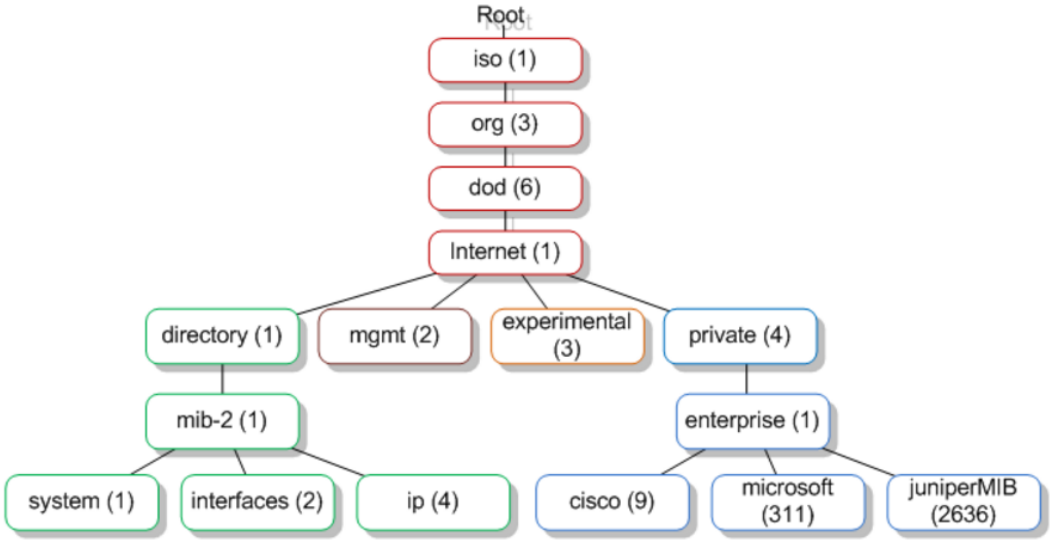


FIGURE 4.17: Exemple d'arbre OID.

b. Utilisation du SNMP sur la plateforme (Noeuds sans fil et contrôleur SDN)
Un manager SNMP tourne au niveau du contrôleur SDN et des agents SNMP au niveau des noeuds sans-fil. Les configurations concernent beaucoup plus les noeuds. Des nouveaux OIDs ont été définis pour contenir les informations telles : la puissance d'émission, le canal sur lequel opère une interface sans-fil, le RSSI, '. Ces OIDs sont greffés sur « experimental » (confère Figure 4.18) Ainsi on a :
Sur chaque feuille de l'arbre précédent on ajoute le niveau (101) qui n'a pas de signification particulière et après viennent les OIDS concrets :

— « .101.1 » représente la puissance d'émission sur l'interface
— « .101.2 » représente le canal de l'interface et ainsi de suite

Le manager du côté du contrôleur SDN peut avoir accès à ces informations à travers le protocole SNMP comme indiqué ci-avant. En effet le service de découverte de topologie maintient une base de données des liens entre les différents noeuds du plan de données. Un lien est caractérisé par deux noeuds d'extrémité et chaque noeud par son « dpid », le « port_no », le « port_name » qui représentent respectivement l'identifiant du noeud, le numéro et le nom du port à travers lequel le lien en question est établi. En utilisant ces informations le contrôleur (manager SNMP) interroge les noeuds et demande les informations adéquates pour pouvoir caractériser un lien donné.
Voici (Figure 4.19) un exemple d'informations recueillies au niveau du contrôleur caractérisant un lien sans-fil entre l'OVS 10 port 2 (wlan1) et l'OVS 14 port 1 (wlan0)

OF messages

L'on peut de nouveau exploiter les messages 'OFPT_ECHO_REQUEST/REPLY' qui en plus de véhiculer certains attributs de noeuds, transportent les mesures de la qualité des liens. Dans le but de réduire l'overHead et de récupérer les métriques des liens d'une façon 'intelligente' où la fréquence de récupération des métriques dépend de la sollicitation du noeud (un noeud exposé à des variations 'perturbations,



FIGURE 4.18: L'OID experimental.

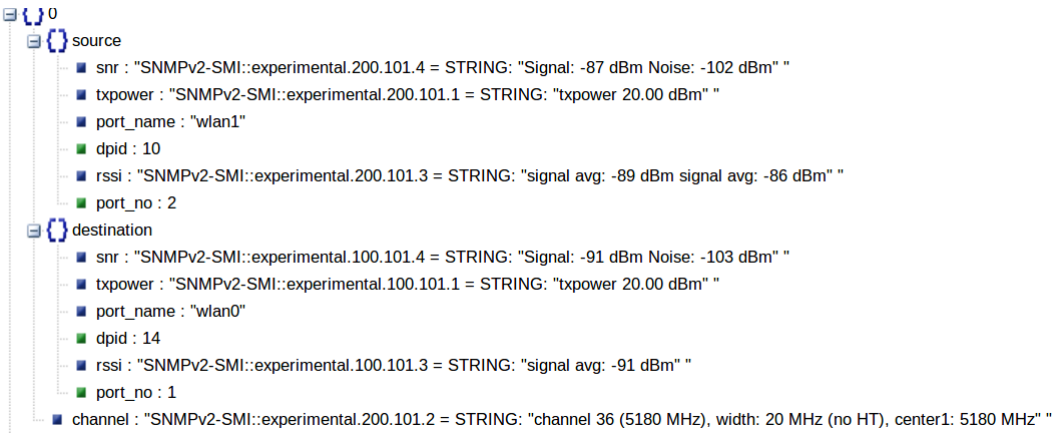


FIGURE 4.19: Metriques de lien recueillies par SNMP.

plusieurs connexions,..' sera interrogé à une fréquence plus élevée qu'un noeud isolé et stable).Plusieurs propositions ont été faites dans ce sens :

— **Récupération des métriques à la demande du contrôleur** : Pour se faire, le contrôleur envoi des ECHO_REQUEST vers l'OVS (le nombre d'envoi dépend de l'importance de l'OVS en question) et l'OVS répond à ces sollicitations par des ECHO_REPLY dans lesquelles il note ses métriques.

— **Récupération des métriques suite à un évènement** : L'information des métriques est remontée de l'OVS vers le contrôleur si et seulement si il y a eu un dépassement d'un seuil (préalablement défini) d'un des paramètres décrivant le lien. Dans ce cas les ECHO_REQUEST (contenant les métriques) sont initiés par l'OVS et contrôleur fait que acquitter le message par le ECHO_REPLY.

Il serait également possible d'utiliser les messages OFPT_EXPERIMENTER comme d'écrit en début de chapitre.

4.5 Positionnement par rapport à l'existant

4.5.1 En termes de découverte de topologie

Dans l'article SDN-WISE 'WIreless SEnsor network' [**Reference13**] la découverte de topologie est centralisée dans une entité nommée 'WISE-VISOR Topology manager', séparée du contrôleur, où les noeuds se découvrent indépendamment et envoient au gestionnaire de topologie des TD-Packet (Topology discovery packet) avec comme information le niveau de batterie, RSSI, Nbr hops vers le Ctrl, voisins directes...).
Si un noeud 'A' reçoit un TD-Packet du noeud 'B', il exécute les opérations suivantes :

1. Mettre B dans la liste des voisins directs et noter son RSSI et son niveau de batterie (Si il existe déjà mettre à jour ses caractéristiques).
2. Vérifier si le chemin par B est le meilleur pour joindre le contrôleur (en termes de sauts), si oui mettre à jour cette information.
3. Noter ces informations (du noeud lui-même 'A') dans la table du TD.
4. Envoyer ces informations du TD vers WISE-VISOR par un broadcast wireless channel.

A partir de ces informations le WISE-VISOR construit une vision globale de la topologie qui sera exploitée par le contrôleur.

4.5.2 En termes de réduction du surdébit (overhead)

Sachant que la majorité du trafic entre les OVS et le(s) contrôleur(s) est du LLDP, et dans le but de réduire ce trafic, certains travaux (exemple [**Reference11**]) proposent plutôt que d'envoyer un Packet-out par port, d'envoyer un seul Packet-out par switch et d'installer une règle dans l'OVS pour inonder 'Flood' ce message LLDP sur tous les ports. Du fait que nous sommes dans une configuration LLDP disabled, l'OVS n'a pas le droit de modifier la trame LLDP pour changer le champ TLV Port-ID (port de sortie de la trame)), il sera impossible au contrôleur de savoir par quel port la trame est revenu. L'OSV a tout de même le droit de changer les entêtes de la trame LLDP, ces travaux proposent donc de changer l'adresse MAC source de l'entête du paquet LLDP et mettre celle du port depuis laquelle la trame est sortie, et de dire au contrôleur de créer les liens en raisonnant sur le champs adresse MAC source de la trame et non pas sur les Port-ID. Grâce à ce nouveau mécanisme ils ont pu réduire l'overHead du protocole de découverte de topologie dans le réseau SDN.

4.5.3 Conclusion

Ce chapitre a proposé différentes solutions pour l'implémentation de la découverte de topologie et comparé celles ci à la bibliographie, le chapitre 5 détaille l'implémentation de l'une d'elle.

Chapitre 5

Analyse expérimentale des communications contrôleur SDN à noeud sans-fil

5.1 Préambule

Les communications entre contrôleur et noeuds sont au coeur du fonctionnement d'une architecture réseau SDN. Elles permettent au contrôleur de découvrir les noeuds et la topologie du réseau, de programmer à distance l'acheminement des noeuds en alimentant leurs tables et de récupérer l'état et les statistiques des noeuds. De ce fait, l'essentiel du trafic (que l'on appellera trafic de contrôle) échangé par ces communications doit être livré avec un certain niveau de performances, en termes de fiabilité et de délai. En effet, une notification de changement d'état (défaillance, ..) ne peut être perdue et doit être livrée rapidement au contrôleur, puis aux applications de contrôle concernées afin de lancer les actions de recouvrement appropriées.

Dans le contexte d'un réseau SDN filaire, les communications entre contrôleur et noeuds ont de très bonnes performances. En effet, souvent le trafic de contrôle est transporté en 'hors-bande' ('out-band') sur un réseau filaire dédié surdimensionné par rapport aux besoins. Dans le cas, d'un transport 'dans la bande' ('in-band'), le trafic de contrôle est traité par les noeuds de manière prioritaire par rapport au trafic de données.

Les choses sont quelque peu différentes dans le contexte de réseaux multi-sauts sans-fil. Premièrement, les communications multi-sauts sans-fil ont plutôt des performances modestes en termes de pertes, de délai et de capacité (débit). De plus, un transport 'hors-bande' du trafic de contrôle peut s'avérer impossible pour des raisons d'encombrement ou de consommation d'énergie. C'est ce constat qui motive ce travail dont l'objectif est d'évaluer expérimentalement les performances des communications entre contrôleur et noeuds pour le cas particulier du réseau multi-sauts sans fil déployé au LAAS (cf Chapitre 3) et dont on a une visualisation sur la Figure 4.1. Ces évaluations serviront ensuite d'entrées pour identifier les applications de contrôle qui peuvent être considérées pour des réseaux multi-sauts sans-fil de type SDN (avec la même envergure que celui utilisé dans nos expérimentations) et pour éventuellement identifier des recommandations permettant d'ajuster certains paramètres protocolaires.

5.2 Objectifs

Les mesures expérimentales ont été réalisées avec les objectifs suivants :

— Évaluer l'impact de la charge, de la dynamique et de l'envergure du réseau sur le volume du trafic de contrôle supporté par le réseau. L'idée est d'évaluer le sur-débit protocolaire ('overhead') occasionné. Plus prospectivement, nous souhaitons également analyser la possibilité d'avoir recours aux technologies LP-WAN (Low-Power Wide Area Networks) pour le transport à un saut du trafic de contrôle ;

— Évaluer l'impact de la charge du réseau sur la perte et le délai de livraison des messages de contrôle. Plusieurs buts sont recherchés :

  — évaluer la réactivité d'une action contrôle ou d'une notification pour analyser l'applicabilité de l'approche à certaines applications de contrôle réseau et

  — adapter certains paramètres protocolaires au contexte que l'on considère.

Ces analyses considérerons, lorsque cela est justifié, aussi bien le cas d'un transport 'Out-of-band' que le cas 'in-band' :

— Expérimentation sous contrôle Out-band : Avec cette configuration le trafic de contrôle transite via un autre réseau que celui des données (dans notre cas réseau Ethernet) Il s'agit du cas le plus favorable en termes de performances. En pratique ce ne sera pas un réseau Ethernet qui sera utilisé, mais pour les expériences que nous avons menées et qui ciblaient l'évaluation du volume de trafic de contrôle, cela ne présente pas d'inconvénient. Dans l'avenir un réseau sans fil LPWAN pourrait être utilisé.

— Expérimentation sous contrôle In-band : Avec cette configuration le trafic de contrôle transite dans le même réseau que celui des données. (le cas le plus défavorable pour les performances). Dans cette configuration, et en plus de l'évaluation du volume du trafic, nous allons aussi focaliser les tests sur la quantification des délais.

En plus de ces configurations différentes du mode de contrôle, la plateforme était sollicitée (selon les objectifs) par des générateurs de trafic installés dans des différents noeuds sans-fil (utilitaire Iperf). Au niveau du contrôleur Ryu, nous avons le service de découverte de topologie et une application Ryu simple switch (permet de donner un comportement switch L2 aux OVS) qui s'exécutent (selon les besoins des tests).

5.3 Evaluation du débit du trafic de contrôle

5.3.1 Scénario considéré

Nous évaluons en premier lieu le surdébit occasionné par le trafic de contrôle. Ce trafic de contrôle comprend majoritairement les messages OpenFlow incluants ceux relatifs à la découverte des liens (OFDP) mais aussi les messages permettant d'entretenir les connexions TCP/IP établies entre le contrôleur et les différents noeuds (Les ACKs). Nous utilisons un nombre croissant de noeuds (5, 8, 12) pour évaluer l'impact de l'envergure réseau sur le volume du trafic de contôle. A part le service de découverte de topologie aucune autre application n'est lancée au niveau du contrôleur et il n'y a pas de sollicitation du réseau de données par une quelconque appplication.

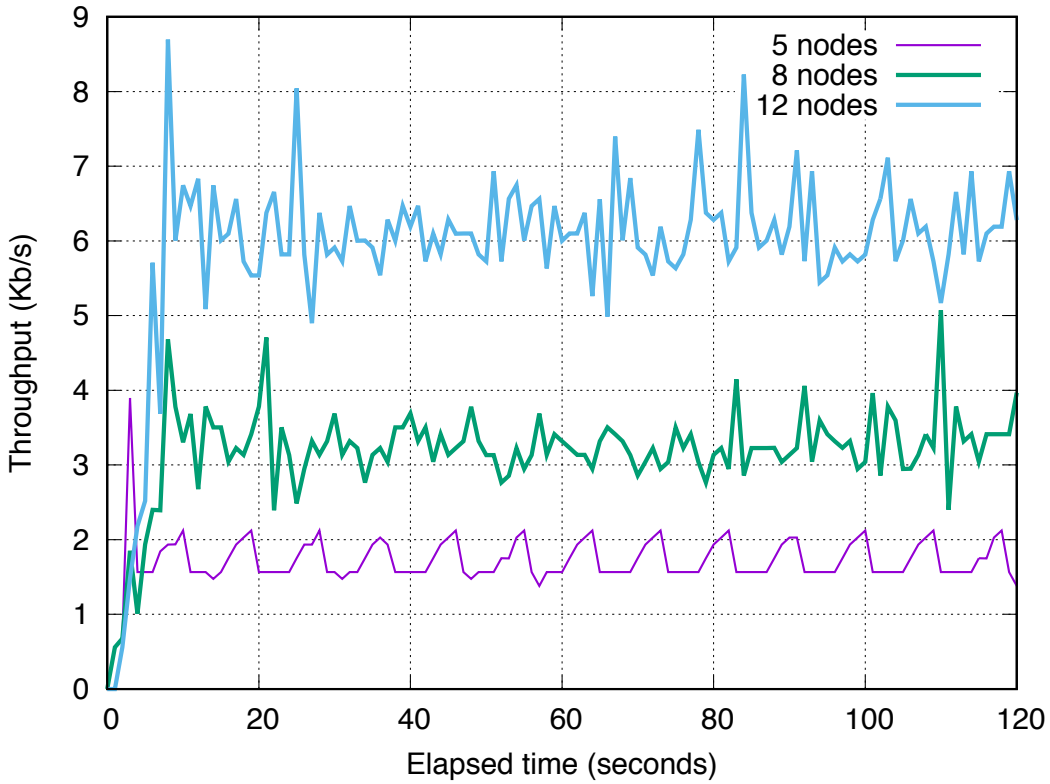


FIGURE 5.1: Débit du trafic(messages) OpenFlow.

5.3.2 Résultats expérimentaux

La figure 5.1 présente le débit du trafic de contrôle observé. L'analyse de ces résultats est présentée comme suit :

1. L'on peut observer sur ces courbes de legers pics au démarrage marquant une première période pendant laquelle intervient l'initialisation des sessions OpenFlow avec les différents noeuds du réseau, la découverte des noeuds et leurs caractéristiques et l'installation des règles OpenFlow sur les noeuds pour permettre l'acheminement des messages LLDP. La durée de cette période est sensiblement la même indépendamment des configurations considérées. Cela s'explique par le fait que la capacité du réseau de contrôle est surdimensionnée par rapport au trafic qu'il transporte.
   Ensuite s'installe un régime « permanent » pendant lequel le trafic OFDP est échangé entre le contrôleur et les noeuds. Comme décrit dans le chapitre 4, des messages OFDP sont périodiquement transmis à l'attention de chaque port actif de chaque noeud. C'est ce qui explique la relative régularité du trafic observé.
2. Le débit du trafic de contrôle augmente quand le nombre de noeuds augmente dans le réseau. Plus précisement le débit augmente quand le nombre de ports actifs et de liens augmentent dans le réseau, ceci à cause du processus de découverte des liens OFDP. En isolant quelques noeuds (exemple de la configuration à 5 noeuds) on evalue le débit du trafic de contrôle qu'ils envoient vers le contôleur et celui du trafic qu'ils reçoivent de ce dernier. En fonction du nombre de ports actifs de ces noeuds, du nombre de connexions qu'ils ont avec d'autres noeuds et des débits observés tableau 5.1 on comprend mieux l'évolution du trafic de contôle en fonction de l'envergure du réseau.

TABLE 5.1: Débits individuels.

Noeuds	Ports	Liens	Débits (Oct/s vers OVS)	Debits (vers ctrl)
OVS4	1	2	103	205
OVS6	2	2	206	209
OVS8	1	1	103	104
OVS9	1	2	104	205
OVS11	2	3	205	311

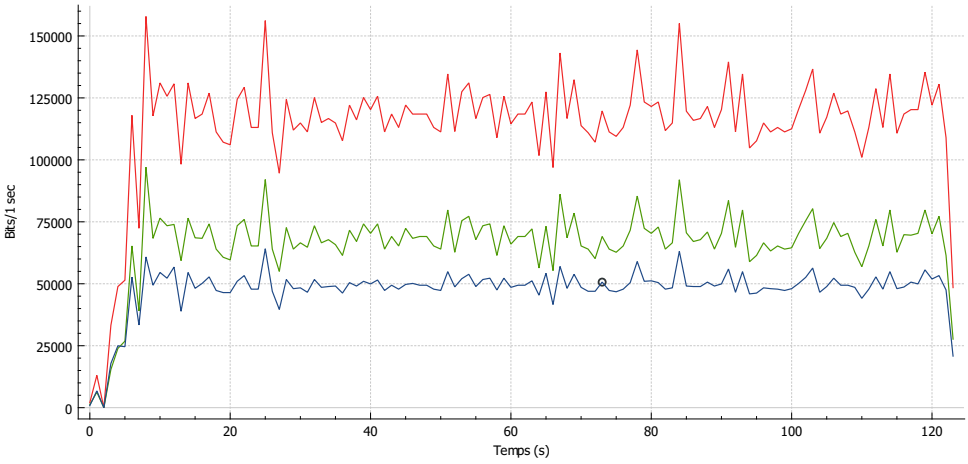


FIGURE 5.2: Le trafic de contrôle au niveau du contrôleur (séparé).

3. Sur la figure 5.2 on peut observer simultanément le débit du trafic de contrôle (configuration à 12 noeuds) en rouge, et la décomposition de ce trafic de contrôle en trafic émis (downlink) en bleu et en vert le trafic reçu (uplink) par le contrôleur. A l'aide de cette figure, on peut bien voir que le trafic de contrôle montant est supérieur au trafic descendant. Ce qui est un résultat tout à fait normal et qui est dû aux connexions multipoints des liens sans fil, où pour un seul message OFDP émis (OFDP Packet-out) nous allons avoir plusieurs messages OFDP en retour (OFDP Packet-in). Ceci est également illustré dans le tableau 5.1.

   Remarque : Les messages OpenFlow étant échangés au dessus d'une connexion TCP/IP, chaque message OFDP (packet-in ou packet-out) est suivi par un ACK ce qui fait que quand bien même les débits soient différents dans le sens montant et le sens ascendant le nombre de paquets est le même dans ces deux sens. En effet un ACK est composé de 56 octets tandis qu' un message OFDP Packet-out et un message OFDP Packet-in sont composés respectivements de 157 octets et 159 octets d'où la différence de débits constatée.

5.3.3 Conclusion

Les résultats présentés nous ont permis d'avoir des ordres de grandeur sur le sur-débit occasionné par les procédures d'initialisation des sessions Openflow et de découverte de topologie. A titre d'exemple, pour la configuration réseau à 12 noeuds, le sur-débit moyen global mesuré au niveau du contrôleur est de l'ordre de 10 koctets/s avec une répartition d'environ 6 koctets/s et 4 koctets/s respectivement dans le sens montant et descendant. A l'échelle d'un seul noeud, le débit est de quelques centaines d'octets par seconde en fonction de ses caractéristiques :

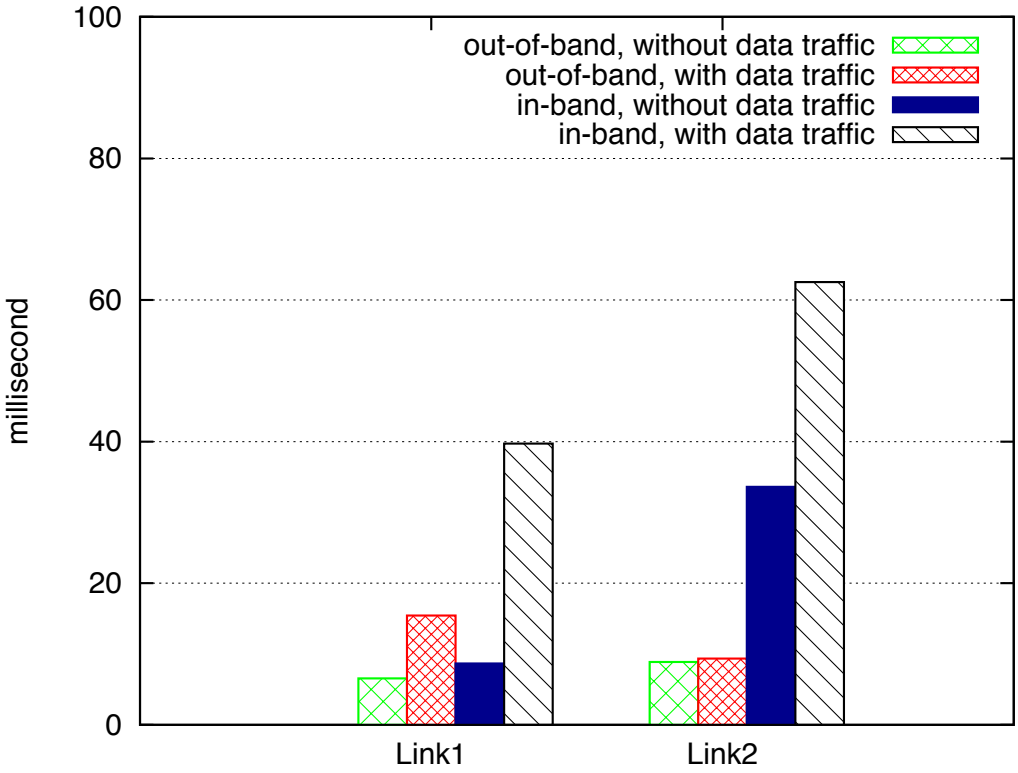


FIGURE 5.3: Délai des messages OFDP du packet-out au Packet-in pour la détection d'un lien.

nombre de ports actifs et nombre de liens sans-fil qu'il possède avec ses voisins directs. A titre d'exemple, le sur-débit moyen dans le sens montant (vers le contôleur) du noeud OVS 14 s'élève à 0.369 koctets/s.
Enfin, il est à noter que les résultats présentés peuvent être dérivés depuis le fonctionnement des procédures évoquées ci-avant. Ils nous ont néanmoins permis de valider notre protocole expérimental. Avec ces débits du trafic de contrôle relativement faibles on peut envisager la possibilité d'utiliser un réseau LP-WAN (exemple de la technologie LoRa) pour un contrôle out-of-band, réseau sans-fil où tous les noeuds seraient à portée radio mutuelle avec le contrôleur.

5.4 Evaluation de l'impact de la charge du réseau sur les messages OFDP

La procédure de découverte de liens et leur suivi génère un délai, de l'émission du packet-out OFDP par le contrôleur, l'échange de la trame LLDP entre noeuds sans fil jusqu'à la réception du packet-in OFDP correspondant. La figure 5.3 représente ce délai pour le réseau expérimental présenté sur les figures 5.4 et 5.5 du tableau 5.2.

Il est minime lorsque ces paquets sont envoyés avec un réseau externe dédié pour de contrôle (contrôle out-of-band). Il est moins de 10 ms, mais pourrait augmenter si le noeud doit gérer également le trafic de données. Le délai est plus élevé lorsque ce trafic de contrôle est véhiculé par le même réseau que celui des données applications(contrôle in-band).

Comme illustré sur la figure configurations du réseau expérimental,en considérant l'OVS 5 , il reçoit un packet-out OFDP du contrôleur, ensuite envote la trame

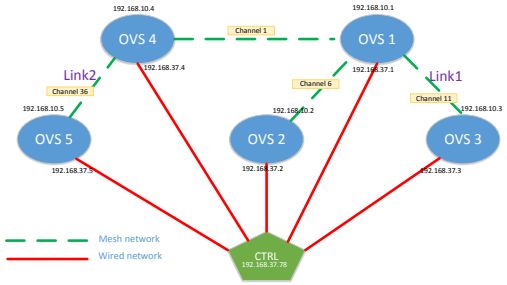


FIGURE 5.4: Contrôle out-of-band

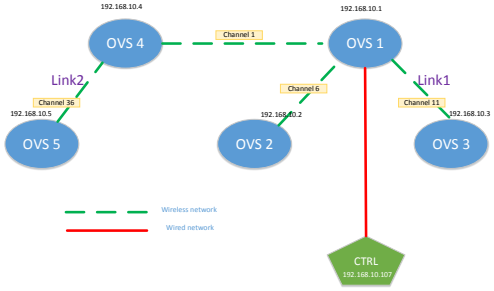


FIGURE 5.5: Contrôle in-band

TABLE 5.2: Réseau expérimental - delai OFDP

LLDP contenue dans ce paquet sur son port concerné. Le trafic LLDP passe alors par OVS 4 à travers Lien 2 et prend deux sauts pour atteindre le contrôleur en configuration in-band (un seul saut avec une configuration out-of-band). Il faut moins de 40 ms pour que le contrôleur puisse avoir les caractéristiques de Link2 dans le meilleur des cas, et environ 60 ms lorsque le trafic de données est envoyé en même temps. Avec le Lien 1 entre OVS 3 et OVS 1, les paquets LLDP ne traversent qu'un seul saut, tant avec une configuration out-of-band qu'en in-band avant d'atteindre le contrôleur, encapsulé dans un packet-in. Dans ce cas, le délai est presque identique dans les deux types de contrôle, sauf dans le cas où il y 'a présence d'autres charges sur le réseau de données.

5.4.1 Conclusion

Cette section a montré la pertinence d'une approche SDN sur réseaux sans fil qui se base sur une découverte de topologie offrant toutes les informations nécessaire à l'élaboration de stratégies de gestion de réseau complexes. Le sur-débit induit par la signalisation openflow est limité et rend de fait possible une approche de signalisation out-of band par réseau LPWAN a un saut. Face à la complexité des réseaux multi-sauts sans fil et aux besoins en terme de connectivité et en qualité de service, tant pour les scénarios d'urgence que pour des applications militaires, nous pensons que cette approche est la bonne.

Conclusion générale

Les travaux décrits dans ce rapport visent la mise en application du concept Software Defined Networking (SDN) aux réseaux multi-sauts sans-fils. Trois contributions y sont décrites. La première concerne la mise en place d'une plateforme réseau SDN sans-fil au sein du Laboratoire avec une quinzaine de noeuds embarquant le switch logiciel OpenVSwitch et qui communiquent avec un contrôleur SDN par le biais de deux types de contrôle : un contrôle out-of-band et in-band. La deuxième contribution a porté sur l'étude du service de découverte de topologie actuellement implémenté au sein du contrôleur RYU. Elle nous a permis d'analyser l'adéquation de ce service aux réseaux multi-sauts sans-fil et de proposer plusieurs extensions dont certaines ont été mises en oeuvre. Enfin la plateforme a ensuite servi pour la réalisation d'une analyse expérimentale dont l'objectif est de caractériser les performances des communications entre contrôleur et noeuds sans-fil.

Ces travaux ouvrent la voie à des travaux plus algorithmiques sur le routage optimal à qualité de service abordé dans le contexte de réseaux fixes. En effet, toutes les métriques nécessaires sont maintenant connues du contrôleur qui peut appliquer des algorithmes d'optimisation. Il a, de plus, été montré qu'une approche de signalisation out-of band est tout à fait possible avec un réseau dédié sans fil de type LPWAN dont les propriétés sont une longue portée de communication mais un débit faible. Le sur-débit openflow est suffisamment faible pour être envisagé sur un réseau de ce type. Face à la complexité des réseaux multi-sauts sans fil et aux contraintes en terme de connectivité et en qualité de service, tant pour les scénarios d'urgence que pour des applications militaires, nous pensons que cette approche est la bonne. La garanti d'une connectivité permanente pour le réseau de signalisation permet d'envisager tout type de contrôle du réseau afin de garantir les propriétés de l'application. Une technologie de communication de plus courte portée, tel que le WiFi ou d'autres techniques UWB, permet de transporter les données applicatives avec des conditions de qualité de service suffisantes.

Annexe A

Annexe

A.1 Quelques commandes utiles

Commandes sur les commutateurs openFlow (OpenVSwitch)

- **ovs-vsctl show**
  Pour afficher le contenu de la base de données du switch (sa configuration notamment :)
- **ovs-vsctl add-br br0**
  Création d'un bridge (br0 ici)
- **ovs-vsctl del-br br0**
  Suppression du bridge br0
- **ovs-vsctl add-port br0 port_1**
  Ajouter le port "port_1" au bridge br0
- **ovs-vsctl del-port br0 port_1**
  Supprimer le port "port_1" du bridge br0
- **ovs-ofctl show br0**
  Pour le monitorage des ports du bridge br0.Affichage de la configuration de br0 avec ces différents ports
- **ovs-ofctl dump-flows br0**
  Affiche les règles installées dans le switch manuellement ou par une app du contrôleur
- **ovs-appctl bridge/dump-flows br0**
  Affiche toutes les règles du switch
- **ovs-ofctl add-flow br0 ip,in_port=2,actions=1**
  Ajouter une règle. Ici la règle demande au switch d'envoyer sur son port 1 tous les flux ip qu'il reçoit sur son port 2.
- **ovs-vsctl set-controller br0 tcp :192.68.1.254 :6653**
  Configurer l'adresse IP du contôleur

A.2 Mise en marche de la plateforme

A.2.1 Cas du contrôle out-of-band

Tous les noeuds sont connectés au VLAN (192.168.37.0/20) qui sera utilisé pour le trafic de contrôle.Le réseau des données est 192.168.10.0/24 (réseau sans fil).
Sur la machine du contrôleur On lance le script (out_of_band.sh).Dans ce script, à travers "SSH" on accède à toutes les cartes et on procède aux configurations suivantes :

— **ovs-vsctl emer-reset**
Permet de réinitialiser toutes configurattions (à l'aide de ovs-vsctl) préalablement établies sur l'OVS.

— **ovs-vsctl set-controller br0 tcp :192.168.37.78 :6633**
Configure l'adresse IP du contrôleur, et également son port.

— **ovs-vsctl set-fail-mode br0 secure/standalone**
Le mode ?secure ? désactive le fonctionnement ?fallback ? qui autorise l ?OVS de fonctionner en tant qu ?un switch traditionnel (Mac-learning switch) en cas d ?échec ou de non connexion avec le contôleur.Le mode standalone active le fonctionnement ?fallback ?

— **ovs-vsctl set bridge br0 stp_enable=true**
Active le protocle STP sur les OVS.

— **ovs-vsctl set bridge br0 other-config :disable-in-band=true**
Désactive le mode in-band.

A.2.2 Cas du contrôle in-band

A l ?exception du noeud OVS-11 tous les noeuds sont toujours connectés au VLAN 192.168.37.0/20 mais dans ce cas il ne servira que pour la configuration à distance des noeuds.Le contrôle passera dans le même réseau que celui des données 192.168.10.0/24. L ?OVS-11 est connecté sur le contrôleur à travers un cable ethernet. On lance le script in_band.sh qui comporte les commandes suivantes :

— **ovs-vsctl set-controller br0 tcp :192.168.10.107 :6633**

— **ovs-vsctl set bridge br0 other-config :disable-in-band=false**
En activant le mode in-band des règles internes sont installées sur chaque OVS permettant d'assurer ce fonctionnement.

— **ovs-vsctl set-fail-mode br0 standalone**

— **ovs-vsctl set bridge br0 stp_enable=true**

A.2.3 Exemples d'applications à lancer sur le contrôleur

— Visualisation de la topologie du réseau
On lance au niveau du contrôleur (Ryu) l'application gui_topology :
PYTHONPATH=. ./bin/ryu-manager –observe-links ryu/app/-gui_topology/gui_topology.py

— Tester la connectivités des différents noeuds sur le réseaude données (192.168.10.0) :
PYTHONPATH=. ./bin/ryu-manager –observe ryu/app/simple_switch13.py
Et par exemple, avec travers les utilitaires "ping" ou "iperf" on teste la connectivé au niveau des noeuds