

**REPUBLIQUE DU SENEGAL**



**Un peuple - un but - une foi**

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR DE LA RECHERCHE**

**DIRECTION GENERALE DE L'ENSEIGNEMENT SUPERIEUR**

**DIRECTION DE L'ENSEIGNEMENT SUPERIEUR PRIVE**

**ECOLE SUPERIEUR DE TECHNOLOGIE ET DE MANAGEMENT**



**MEMOIRE DE FIN DE CYCLE**

**Pour l'obtention de la licence en TELEINFORMATIQUE**

**Option : Télécommunication et Réseaux**

**INTITULE**

**MISE EN PLACE D'UNE PLATEFORME SDN ET VXLAN  
POUR LA MEDECINE**

**Présenté et soutenu par :**

**Yacine Mbaye**

**Sous la direction de :**

**Dr Keba Gueye**

**Enseignant à ESTM**

**Année Académique : 2023-2024**

**REPUBLIQUE DU SENEGAL**



**Un peuple - un but - une foi**

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR DE LA RECHERCHE**

**DIRECTION GENERALE DE L'ENSEIGNEMENT SUPERIEUR**

**DIRECTION DE L'ENSEIGNEMENT SUPERIEUR PRIVE**

**ECOLE SUPERIEUR DE TECHNOLOGIE ET DE MANAGEMENT**



**MEMOIRE DE FIN DE CYCLE**

**Pour l'obtention de la licence en TELEINFORMATIQUE**

**Option : Télécommunication et Réseaux**

**INTITULE**

**MISE EN PLACE D'UNE PLATEFORME SDN ET VXLAN  
POUR LA MEDECINE**

**Présenté et soutenu par :**

**Yacine Mbaye**

**Sous la direction de :**

**Dr Keba Gueye**

**Enseignant à ESTM**

**Année Académique : 2023-2024**

## **Dédicaces**

### **Ce travail est dédié :**

-À mes parents, dont l'amour et le soutien inconditionnels m'ont permis d'arriver jusqu'ici. Que Dieu me les garde en bonne santé ;

- À mes maîtres coraniques, sources de sagesse et d'inspiration, que Dieu leur accorde longue vie et prospérité ;

-À mes frères et sœurs, qui ont toujours été à mes côtés, dans les moments de joie comme dans ceux de difficultés ;

-À mes amis, pour leur présence et leur amitié sincère, qui m'ont aidé à garder espoir tout au long de ce parcours.

## Remerciements

À l'issue de ce travail, je tiens à exprimer ma sincère reconnaissance et mes remerciements les plus chaleureux à mon encadrant, **Monsieur le Professeur Dr Kéba Gueye**, pour m'avoir proposé ce sujet et pour ses précieux conseils, sa direction éclairée et son soutien constant tout au long de cette aventure. Je lui suis particulièrement reconnaissant pour sa patience et sa compréhension, qui ont permis d'achever ce travail exigeant. Je vous prie d'accepter l'expression de mon admiration la plus profonde et de mon respect sincère.

Je tiens également à exprimer ma profonde reconnaissance à **Monsieur Chérif Diouf**, dont l'aide précieuse et sa disponibilité constante ont été d'une grande importance tout au long de la réalisation de ce mémoire. Sa générosité et son esprit d'entraide font de lui une personne exceptionnelle, toujours prête à soutenir ceux qui en ont besoin. Un grand merci pour son accompagnement bienveillant.

Je tiens également à remercier chaleureusement **Monsieur Mamadou Diallo et Monsieur François Mbengue** de la Sonatel pour leur aide précieuse et leur disponibilité tout au long de ce projet. Leur soutien et leurs conseils m'ont été d'une grande aide et ont grandement facilité la réalisation de mon travail.

Je souhaite également remercier chaleureusement **Zakaria Gueye** pour les précieux conseils qu'il m'a apportés tout au long de ce travail. Son soutien et ses suggestions ont grandement contribué à l'avancement de mon mémoire.

Je tiens également à exprimer ma sincère gratitude à **Monsieur Dioh et Monsieur Fam** du Ministère de l'Éducation, qui m'ont apporté une aide précieuse tout au long de ce projet. Leur soutien constant et leur enthousiasme m'ont énormément motivé.. Un grand merci à eux pour leur disponibilité et leur encouragement.

Je tiens également à exprimer ma gratitude à **Monsieur Nouha Coulibaly** pour m'avoir toujours encouragé et soutenu. Ses encouragements m'ont motivé à persévérer et à donner le meilleur de moi-même tout au long de ce travail.

Je tiens également à remercier chaleureusement **tous mes amis**, qui ont toujours été présents pour moi, m'apportant leur soutien moral et leur encouragement tout au long de ce parcours. Leur amitié et leur présence ont été d'une grande importance, et je leur suis

profondément reconnaissant pour leur soutien . Un grand merci à eux pour avoir été là, à chaque étape.

Je remercie également **tous mes camarades de classe** avec qui j'ai partagé de très bons moments d'échange et d'entraide. Leur soutien et nos échanges ont été d'une grande richesse et ont contribué à rendre cette expérience encore plus agréable et enrichissante.

Je tiens également à exprimer mes remerciements à l'ensemble du corps professoral et administratif de l'**ESTM** pour leur accompagnement et leur soutien tout au long de mon parcours. Leur disponibilité et leur professionnalisme ont grandement facilité mon travail et mon apprentissage.

Enfin, je tiens à exprimer toute ma gratitude à ma famille, notamment à **Ma Mère, Mon Père, Mon Grand Frère et Mes deux Petites Sœurs**, pour leur amour, leur soutien et leur encouragement constant. Un merci particulier à **Ma Mère**, qui m'a énormément aidé dans mes études. Sa patience, sa bienveillance et son soutien m'ont permis de surmonter de nombreuses difficultés et de rester concentré sur mes objectifs. Leur présence, leur compréhension et leurs sacrifices m'ont été d'une grande aide et m'ont donné la force de continuer. Un grand merci à eux pour tout ce qu'ils ont fait et continuent de faire pour moi.

## **Avant-Propos**

L'ESTM (École Supérieure de Technologie et de Management) est un établissement privé d'enseignement supérieur. Fondée en 2001, elle délivre des diplômes homologués par l'État, reconnus par le CAMES et les entreprises. L'ESTM cultive la passion, l'excellence, le savoir pratique et le professionnalisme. Elle propose des filières telles que la Télécommunication, le Génie Logiciel et Administration des Réseaux, la Banque-Assurance, le Génie Électrique et Énergies Renouvelables, le Transport Logistique, ainsi que le Marketing et la Communication.

Le présent mémoire s'inscrit dans le cadre de la licence en Téléinformatique Réseaux et Télécommunications et constitue une étape importante pour l'obtention du diplôme. Il explore la mise en place d'une plateforme innovante utilisant les technologies SDN et VXLAN dans le domaine médical.

Ce document constitue notre première recherche. À ce titre, nous remercions les membres du jury pour avoir accepté d'évaluer ce travail et sollicitons leur indulgence lors de cette évaluation.

# Sommaire

Dédicaces .....	I
Remerciements .....	II
Avant-Propos.....	IV
Glossaire.....	VII
Liste des figures .....	IX
Résumé.....	XI
Abstract .....	XII
Introduction générale.....	1
CHAPITRE 1 : Présentation générale .....	4
1.1    Présentation du lieu de stage.....	4
1.2    Présentation du sujet .....	5
1.2.1 Contexte .....	5
1.2.2 Problématique.....	6
1.2.3 Objectifs.....	6
CHAPITRE2 : Généralités sur SDN et IoT .....	8
2.1. Introduction aux réseaux SDN.....	8
2.1.1. Définition.....	8
2.1.2. Avantages et Fonctionnement du SDN .....	8
2.1.2.1. Avantage .....	9
2.1.2.2. Fonctionnement .....	9
2.1.3 Différence entre un réseau SDN et un réseau traditionnel.....	11
2.2 Internet des objets (IOT) .....	12
2.2.1. Définition.....	12
2.2.2. Avantages et Fonctionnement de l'IOT.....	13
2.2.2.1. Avantage de l'iot dans le domaine de la medecine.....	13
2.2.2.2. Fonctionnement .....	14
2.1.3. Intégration de l'IoT dans les Réseaux SDN .....	15
2.3. Conclusion .....	16
Chapitre 3 : Étude du protocole Vxlan et des protocole IoT.....	18
3.1. Etude du protocole Vxlan .....	18
3.1.1 Définition.....	18
3.1.2 Avantage et Fonctionnement de Vxlan .....	19
3.1.2.1 Avantage .....	19
3.1.2.2 Fonctionnement .....	19

3.1.3 Différence entre VxLan et VLAN.....	20
3.2. Etude des protocole IOT.....	21
3.2.1. Étude de divers protocoles IoT.....	21
3.2.2 Choix du protocole MQTT .....	23
3.3. Conclusion .....	28
Chapitre4 : Architecture et Mise en Œuvre de la Solution.....	30
4.1 Schéma d'architecture de l'infrastructure .....	30
4.2 Outils et Plateformes Utilisés .....	31
4.2.1 Présentation des outils et plateformes.....	31
4.2.1.1 Le microcontrôleur esp32.....	31
4.2.1.2 Plateforme Node.js.....	32
4.2.1.3 Open vSwitch.....	33
4.2.1.4 Opendaylight .....	33
4.3 Déploiement de la Solution .....	36
4.3.1 Déploiement et configuration des capteurs IoT sur ESP32 .....	36
4.3.2 Mise en place de la plateforme de visualisation et la base de donnée .....	42
4.3.2.1 Mise en place de la plateforme .....	42
4.3.2.2 Mise en place de la base de données.....	56
4.3.3 Configuration du tunnel VXLAN entre les machines virtuelles .....	58
4.3.4 Gestion du Réseau SDN via OpenDaylight .....	60
4.4. Conclusion .....	63
Chapitre5 : Tests de la solution .....	65
5.1 Tests de Transmission des Données Capteurs vers le Serveur MQTT.....	65
5.2 Visualisation sur Plateforme et Stockage en Base de Données .....	65
5.3 Tests de la connectivité et de la sécurité du tunnel VXLAN et la visualisation de la plateforme sur la machine client a distance .....	68
5.4 Vérification des règles de gestion sur SDN.....	71
5.5. Conclusion .....	72
Conclusion Generale .....	73
Bibliographie.....	74



## **Glossaire**

**ESTM** : École Supérieure de Technologie et de Management.

**SDN** : Software-Defined Networking.

**VXLAN** : Virtual Extensible LAN.

**IOT** : Internet of Things.

**IP** : Internet Protocol.

**VM** : Virtual Machine.

**UDP** : User Datagram Protocol.

**VNI** : VXLAN Network Identifier.

**VTEP** : VXLAN Tunnel Endpoint.

**KVM** : Kernel-based Virtual Machine.

**VLAN** : Virtual Local Area Network.

**HTTP** : HyperText Transfer Protocol.

**WWW** : World Wide Web.

**TCP** : Transmission Control Protocol.

**MQTT** : Message Queuing Telemetry Transport.

**IBM** : International Business Machines.

**CLOUD** : Cloud Computing.

**QOS** : Quality of Service.

**OAuth** : Open Authorization.

**TLS1** : Transport Layer Security 1.

**PUB/SUB** : Publish/Subscribe.

**ESP32** : ESP32 (Microcontrôleur avec Wi-Fi et Bluetooth intégrés).

**XML** : eXtensible Markup Language.

**JSON** : JavaScript Object Notation.

**SOC** : System on Chip.

**E/S** : Entrée/Sortie.

**API** : Application Programming Interface.

**OVS** : Open vSwitch.

**OSI** : Open Systems Interconnection.

**ONF** : Open Networking Foundation.

**OSGI** : Open Service Gateway Initiative.

**VCC** : Voltage Common Collector.

**GND** : Ground (Masse).

**NPM** : Node Package Manager.

## Liste des figures

Figure 2.1 : Architecture détaillée d'un réseaux SDN [3].....	10
Figure 2.2 : Comparaison entre un réseau traditionnel et un réseau SDN [3].....	12
Figure 2.3 : Vue d'ensemble de l'internet des objets [4]. ....	13
Figure2.4 : Exemple d'architecture de télésurveillance de patient .....	14
Figure 2.5 : Architecture de l'internet des objets [5]. ....	15
Figure 3.1 : Technologie de tunneling VXLAN [8].....	18
Figure 3.2 : Fonctionnement de VXLAN.....	20
Figure 4.1 : Architecture générale du système .....	30
Figure 4.2 : esp32 [5]. ....	32
Figure 4.3 : Plateforme opendaylight .....	36
Figure 4.4: Intégration de l'ESP32 et du capteur.....	37
Figure 4.5 : Fichier de configuration du capteur et de l'ESP32.....	39
Figure 4.6 : Installation du serveur Node.js et NPM.....	43
Figure 4.7 : Vérification des versions de Node.js et de NPM.....	43
Figure 4.8 : Initialisation d'un nouveau projet Node.js.....	44
Figure 4.9 Installation de la bibliothèque MQTT.....	44
Figure 4.10 : Installation de la bibliothèque Express .....	44
Figure 4.11 : Installation de la bibliothèque Socket.IO.....	45
Figure 4.12 : Code de configuration du serveur dans le fichier app.js.....	48
Figure 4.13 : Code de configuration pour l'authentification dans le fichier login.html.....	52
Figure 4.14 : Fichier login.css pour la mise en forme de la page d'authentification .....	53
Figure 4.15 : Structure de la page d'accueil dans le fichier index.html.....	54
Figure 4.16 : Fichier style.css pour la mise en forme de la page .....	56
Figure 4.17 : Installation de MySQL-Server.....	56
Figure 4.18 : Création de la Base de Donnée .....	56
Figure 4.19 : Création de la table temperature dans la Base de Données .....	57
Figure 4.20 : Création de la table utilisateurs pour l'authentification sur la plateforme.....	57
Figure 4.21 : Création de deux utilisateurs pour l'authentification.....	57
Figure 4.22 : Installation de OpenVSwitch sur la première machine .....	58
Figure 4.23 : Configuration de openvswitch sur la première machine .....	58
Figure 4.24 : Visualisation du pont OpenvSwitch créé sur la première machine .....	59
Figure 4.25 : Installation de OpenVSwitch sur la deuxième machine .....	59
Figure 4.26 : Configuration de openvswitch sur la deuxième machine .....	59

Figure 4.27 : Visualisation du pont OpenvSwitch créé sur la deuxième machine.....	60
Figure 4.28 : Visualisation de la topologie des ponts configurés dans OpenDaylight.....	60
Figure 4.29 : Affichage des ports de tunnels VXLAN sur le premier dans la plateforme Opendaylight .....	61
Figure 4.30 : Affichage des ports de tunnels VXLAN sur le deuxième pont dans la plateforme Opendaylight .....	61
Figure 4.31 : Configuration des règles sur le premier pont.....	62
Figure 4.32 : Configuration des règles sur le deuxième pont.....	62
Figure 4.33 : Visualisation des règles configurées sur le premier pont .....	62
Figure 4.34 : Visualisation des règles configurées sur le deuxième pont .....	62
Figure 5.1 : Test de connexion de l'ESP32 au réseau Wi-Fi et au broker MQTT avec transmission des données du capteur DHT11 .....	65
Figure 5.2 : Lancement du serveur Node.js et connexion au broker MQTT pour recevoir les données.....	66
Figure 5.3 : Authentification à la plateforme avec l'utilisateur admin .....	66
Figure 5.4 : Visualisation en temps réel des données du capteur via l'interface web de la plateforme.....	67
Figure 5.5 : Enregistrement des données de température dans la base de données MySQL pour conservation et analyse.....	67
Figure 5.6 : Test de connectivité entre les machines virtuelles via le tunnel VXLAN .....	68
Figure 5.7 : Capture du trafic réseau avec Wireshark pour vérifier le passage des paquets à travers le tunnel VXLAN .....	69
Figure 5.8 : Accès à distance à la plateforme via l'authentification de l'utilisateur yacine.....	69
Figure 5.9 : Test de l'accès à la plateforme via le tunnel VXLAN entre la machine client et la machine serveur.....	70
Figure 5.10 : Visualisation à distance des variations de température en temps réel via le tunnel VXLAN .....	71
Figure 5.11 : Blocage de la communication entre le client et le serveur après application des règles via le contrôleur SDN .....	71
Figure 5.12 : Impossibilité d'accès à la plateforme de visualisation à distance depuis la machine cliente après application des règles SDN.....	72

## Résumé

La mise en place d'une plateforme intégrant les technologies SDN et VXLAN dans le domaine médical vise à transformer la gestion des données de santé en assurant une collecte et une transmission sécurisées des informations provenant de capteurs IoT variés, tels que les dispositifs Raspberry Pi et ESP32. Cette plateforme offre une gestion centralisée et dynamique du trafic réseau, garantissant une infrastructure évolutive et fiable pour acheminer les données vers un système de visualisation. Les professionnels de santé peuvent ainsi accéder en temps réel aux informations cruciales, comme la température des patients, via une interface intuitive. Les tunnels VXLAN sécurisés garantissent la confidentialité des données médicales, tandis que le contrôleur SDN permet une gestion fine des interconnexions et des politiques de sécurité adaptées aux exigences réseau.

Ce mémoire explore les défis techniques et les meilleures pratiques pour intégrer ces technologies avancées dans le secteur médical. L'objectif est d'améliorer la qualité des soins tout en répondant aux exigences croissantes en matière de sécurité et de gestion des données dans les environnements de santé modernes.

### Mots-clés

- Réseaux SDN
- VXLAN
- IoT médical
- Sécurité des données
- Plateforme de visualisation

## **Abstract**

The implementation of a platform integrating SDN and VXLAN technologies in the medical field aims to transform health data management by ensuring secure collection and transmission of information from various IoT sensors, such as Raspberry Pi devices and ESP32. This platform provides centralized and dynamic management of network traffic, ensuring a scalable and reliable infrastructure for routing data to a visualization system. Healthcare professionals can therefore access crucial information, such as patient temperatures, in real time via an intuitive interface. Secure VXLAN tunnels guarantee the confidentiality of medical data, while the SDN controller allows fine-grained management of interconnections and security policies adapted to network requirements.

This dissertation explores the technical challenges and best practices for integrating these advanced technologies into the medical sector. The goal is to improve the quality of care while meeting the increasing demands for security and data management in modern healthcare environments.

### **Keywords**

- SDN Networks
- VXLAN
- Medical IoT
- Data Security
- Visualization Platform

# Introduction générale

L'évolution rapide des technologies de l'information a révolutionné de nombreux secteurs, y compris celui de la santé, en introduisant des solutions innovantes pour relever les défis croissants de gestion et de sécurisation des données médicales. Dans cette perspective, l'intégration des technologies SDN (Software-Defined Networking) et VXLAN (Virtual Extensible LAN) représente une avancée majeure, permettant une gestion centralisée et flexible des réseaux pour assurer une transmission sécurisée des données de santé. En combinant ces technologies avec des dispositifs IoT (Internet des objets) comme les ESP32, ce projet vise à créer une plateforme robuste pour la collecte, le transport et la visualisation des données médicales, garantissant leur confidentialité et leur accessibilité en temps réel pour les professionnels de santé.

Ce mémoire examine en détail les différentes étapes de la mise en place de cette plateforme SDN et VXLAN dans un contexte médical. Il est structuré en cinq chapitres principaux, chacun apportant une compréhension progressive des technologies et des méthodes employées :

- **Présentation générale** : Ce premier chapitre présente le cadre de réalisation du projet, y compris le lieu de stage, le contexte de l'étude, la problématique rencontrée, et les objectifs spécifiques visés.
- **Généralités sur SDN et IoT** : Ce chapitre explore les concepts fondamentaux du SDN, en mettant en lumière ses avantages et son fonctionnement, ainsi que ses différences par rapport aux réseaux traditionnels. Il présente également l'IoT en expliquant son rôle essentiel dans la collecte de données médicales en temps réel, son fonctionnement et son intégration dans les réseaux SDN.
- **Étude du protocole VXLAN et des protocoles IoT** : Ici, nous analysons le protocole VXLAN, en mettant en évidence ses avantages, son fonctionnement et sa différence par rapport aux réseaux VLAN traditionnels, ainsi que l'étude des protocoles IoT pertinents pour notre projet. Ce chapitre justifie également le choix du protocole MQTT pour la communication IoT.
- **Architecture et mise en œuvre de la solution** : Ce chapitre se concentre sur le déploiement pratique de la solution. Il détaille l'architecture de l'infrastructure, les outils et plateformes utilisés, ainsi que les étapes de configuration, incluant le déploiement des

capteurs IoT, la mise en place de la plateforme de visualisation, la mise en place de la base de donnée, la configuration du tunnel VXLAN et la gestion des switches virtuels via SDN.

- **Tests de la solution** : Enfin, le dernier chapitre présente les tests réalisés pour valider la solution. Les tests incluent la transmission des données capteurs, la connectivité et la sécurité du tunnel VXLAN, la visualisation de la plateforme en local et à distance ainsi que la vérification des règles de gestion réseau via le contrôleur SDN.





# **Présentation Générale**

# CHAPITRE 1 : Présentation générale

Dans cette partie, nous examinerons le cadre dans lequel le projet a été réalisé, le contexte et en détail la problématique qui nous a poussés à mettre en place ce projet enfin définir les objectifs fixés.

## 1.1 Présentation du lieu de stage

 **Historique** : L'école supérieure de technologie et de management (ESTM) de Dakar est une école supérieure privée d'enseignement supérieur, universitaire et professionnel. Elle a été créée en 2001 par des universitaires et des professionnels des secteurs des technologies, de l'information, de la communication et du management. Les enseignements dispensés s'inspirent des normes exigées par le CAMES (Centre Africain et Malgache pour l'enseignement supérieur) et donc superposable à ceux dispensés dans les meilleures écoles tant sur le continent africain que sur le reste du monde. L'ESTM est implantée à Dakar, capitale politique du Sénégal et située sur l'avenue Bourguiba prolongée x front de Terre et à l'avenue Cheikh Anta DIOP en face hôpital Fann. Elle dispose d'un département Sciences et Technologies, un département management et communication et un département Génie électrique et énergie renouvelable. Des formations ouvertes et à distance (FOAD) y sont également dispensées. A l'intérieur de ses quatre départements, l'ESTM forme des ingénieurs et des techniciens supérieurs très appréciés dans les différentes villes africaines. Ces formations se font en cours du jour, comme en en cours du soir, aussi bien en formation initiale qu'en formation continue pour le compte des entreprises, sociétés et particuliers.

 **Options et Filières** : Les formations à l'ESTM assurent toujours à l'étudiant quel que soit le parcours qu'il a choisi un développement personnel. Les enseignements se déroulent en cours du jour et en cours du soir selon le choix et concernent les filières suivantes :

Licence Téléinformatique

- Option Télécommunications et Réseaux
- Option Génie Logiciel et Administration Réseaux

Master en Téléinformatique

- Option Télécommunications et Réseaux
- Option Génie Logiciel et Administration Réseaux

#### Licence en de Gestion

- Option finance comptabilité
- Option Gestion de projet
- Option Transport et logistique

#### Master en Gestion

#### Licences Professionnelles

- Marketing - Communication,
- Banque - Assurance,
- Management des services & réseaux de communication,
- Génie logiciel & applications mobiles,
- Gestion des ressources humaines

## **1.2 Présentation du sujet**

Dans cette section, nous aborderons le contexte de la problématique ainsi que les objectifs visés.

### **1.2.1 Contexte**

Le domaine médical est en pleine transformation numérique, avec une demande croissante pour des systèmes de surveillance à distance permettant un suivi en temps réel de l'état de santé des patients. Dans ce contexte, l'intégration des technologies de réseaux avancés, telles que les réseaux définis par logiciel (SDN) et les réseaux superposés virtuels (VXLAN), ouvre de nouvelles perspectives pour la gestion sécurisée des données de santé. Grâce aux dispositifs IoT, comme les capteurs connectés et les microcontrôleurs (Raspberry Pi, ESP32), il devient possible de collecter des données essentielles, telles que la température ou la fréquence cardiaque, et de les acheminer vers une plateforme centralisée, accessible par les professionnels de santé, même à distance.

### **1.2.2 Problématique**

Comment mettre en œuvre une infrastructure réseau alliant SDN et VXLAN pour assurer la collecte, la transmission sécurisée et la visualisation des données médicales provenant de capteurs IoT, afin de faciliter la gestion et le suivi à distance des patients tout en garantissant la confidentialité et la sécurité des informations de santé ?

### **1.2.3 Objectifs**

L'objectif de ce projet est de concevoir et de déployer une plateforme SDN et VXLAN pour le domaine médical. Cette plateforme permettra de connecter et de superviser des capteurs IoT, de visualiser les données médicales en temps réel, et de sécuriser les échanges entre les différents acteurs. Plus précisément, il s'agit de :

- Connecter un capteur de température à un ESP32 pour la collecte de données, et envoyer ces données vers un serveur MQTT distant.
- Mettre en place une plateforme Node.js pour la visualisation et le stockage des données dans une base de données.
- Configurer un tunnel VXLAN entre deux machines virtuelles pour garantir une communication sécurisée.
- Intégrer un contrôleur SDN pour gérer dynamiquement les interconnexions et les configurations réseau, en assurant une gestion centralisée et une sécurité optimale.

Ce projet a pour ambition de proposer une solution technologique répondant aux besoins actuels des professionnels de santé pour un suivi des patients plus performant, sécurisé et accessible à distance.

# **Introduction aux Réseaux Définis par Logiciel (SDN) et à l'Internet des Objets (IoT)**

## **CHAPITRE2 : Généralités sur SDN et IoT**

Dans ce chapitre, nous aborderons tout d'abord les principes fondamentaux du SDN, notamment sa définition, ses avantages, son fonctionnement et la différence du réseau SDN et traditionnel. Ensuite, nous explorerons l'Internet des Objets (IoT), en définissant ce concept et en examinant ses avantages sur le côté de la médecine ainsi que son fonctionnement. Enfin, nous nous focaliserons sur l'intégration de l'IoT dans les réseaux SDN, en analysant comment ces deux technologies interagissent et se complètent mutuellement pour créer des environnements réseau intelligents et évolutifs

### **2.1. Introduction aux réseaux SDN**

#### **2.1.1. Définition**

SDN signifie littéralement Software-Defined Networking, c'est-à-dire le réseau défini par l'application. Cette approche innovante de la conception des réseaux se distingue par sa flexibilité, sa gestion simplifiée, son coût réduit et son adaptabilité, ce qui le rend parfait pour les applications modernes qui nécessitent beaucoup de bande passante et qui changent souvent.

Cette architecture sépare les fonctions de contrôle (décider où les données doivent aller) des fonctions de redirection (transporter les données). Cela permet de programmer directement le contrôle du réseau et de simplifier l'infrastructure pour les applications et les services réseau. De plus le SDN facilite l'évolution du réseau en permettant l'ajout ou la suppression d'appareils sans besoin de modifications matérielles majeures, tout en assurant une application uniforme des politiques de sécurité grâce à son contrôle centralisé [1].

#### **2.1.2. Avantages et Fonctionnement du SDN**

Dans cette section, nous examinerons les raisons pour lesquelles le SDN est si utile et comment il fonctionne. Nous discuterons de la façon dont il facilite la gestion des réseaux en permettant leur contrôle par des logiciels, ce qui les rend plus flexibles et adaptables.

### 2.1.2.1. Avantage

- Réseaux programmables : Avec le SDN, il est facile de changer les règles du réseau en modifiant simplement une politique centrale, plutôt que de devoir ajuster de multiples configurations sur différents équipements réseau. Cela simplifie le processus et permet de développer des fonctions réseau plus avancées en utilisant un contrôleur centralisé.
- Flexibilité : Le SDN offre une grande flexibilité pour gérer le réseau. On peut facilement rediriger le trafic, examiner des flux spécifiques, tester de nouvelles stratégies ou détecter des flux imprévus.
- Politique unifiée : Grâce à son contrôleur centralisé, le SDN assure une politique réseau unifiée et à jour. Cela élimine le risque d'erreurs humaines telles que des règles incohérentes entre différents équipements réseau. L'administrateur peut simplement spécifier une nouvelle règle et le contrôleur s'assurera que cette règle est appliquée de manière cohérente sur tous les dispositifs pertinents.
- Routage : Le SDN permet une gestion centralisée des informations de routage, en déléguant cette fonction au contrôleur via une interface dédiée.
- **Coût-efficacité** : Le SDN offre une solution rentable en automatisant la surveillance et les mises à jour des appareils, réduisant ainsi les coûts de maintenance et le besoin de personnel sur site [2].

En résumé, le SDN offre une gestion simplifiée, flexible et centralisée du réseau, tout en facilitant le déploiement de nouvelles fonctionnalités avancées et en assurant une politique réseau uniforme et cohérente.

### 2.1.2.2. Fonctionnement

Le réseau défini par logiciel (SDN) fonctionne en séparant le plan de contrôle du plan de données. Le plan de contrôle, qui prend des décisions sur la manière de router le trafic, est déplacé vers un logiciel, tandis que le plan de données, qui transmet réellement le trafic, reste

dans le matériel. Cela permet aux administrateurs de gérer l'ensemble du réseau via une seule interface [2].

La figure 2.1 offre un aperçu du fonctionnement des réseaux définis par logiciel.

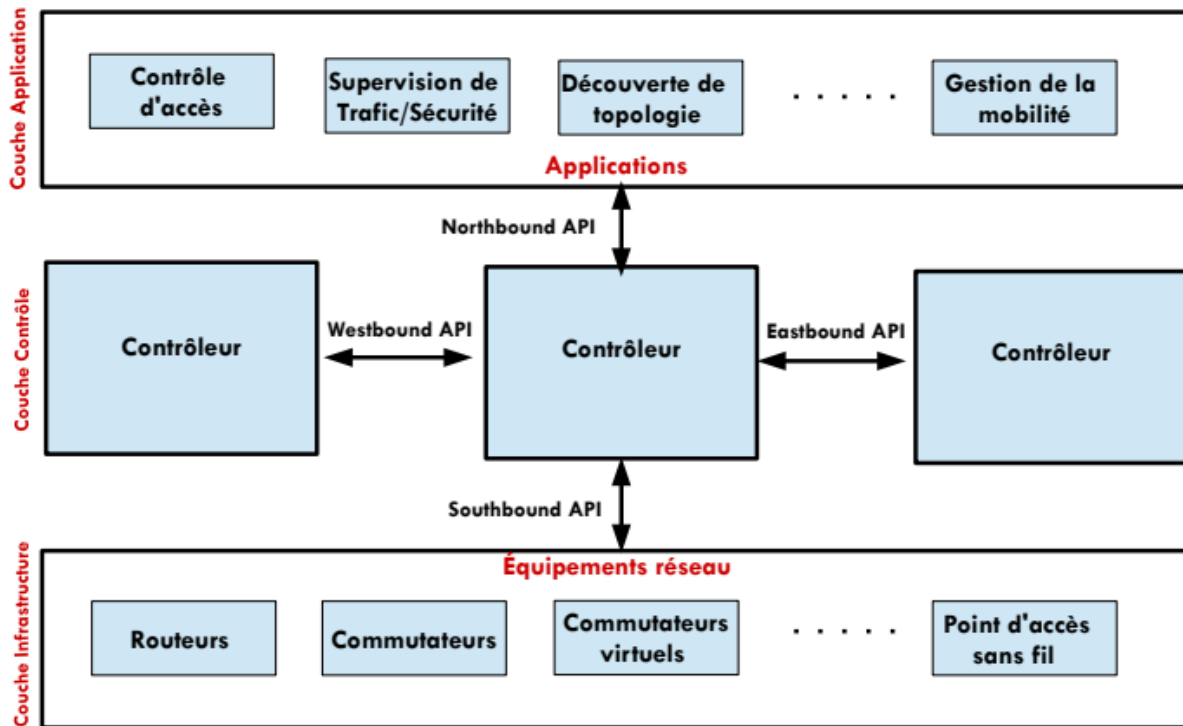


Figure 2.1 : Architecture détaillée d'un réseaux SDN [3].

### Explication de l'architecture

Une architecture SDN typique comprend trois parties, qui peuvent être situées dans différents emplacements physiques :

- **Applications :** Dans le contexte du SDN, les applications font référence aux logiciels ou aux programmes qui interagissent avec le réseau pour différentes raisons. Ces applications communiquent avec le contrôleur SDN pour demander des ressources réseau ou pour obtenir des informations sur l'état du réseau dans son ensemble. Par exemple, une application de surveillance du réseau peut demander des données sur la bande passante utilisée par différents appareils connectés au réseau, ou une application de sécurité peut demander des informations sur les activités suspectes détectées.
- **Contrôleurs :** Les contrôleurs SDN sont les cerveaux du réseau. Ils sont chargés de prendre des décisions sur la manière de router le trafic en fonction des informations



qu'ils reçoivent des applications. Les contrôleurs utilisent ces informations pour créer des politiques de routage et de commutation, décider des chemins optimaux pour les paquets de données et mettre à jour dynamiquement ces décisions en fonction des changements dans le réseau.

- **Périphériques réseau :** Les périphériques réseau dans un environnement SDN sont généralement des dispositifs matériels physiques traditionnels (commutateurs ou des routeurs) ou des instances de logiciels de commutation virtuels dans des environnements virtualisés. Ils sont responsables de la transmission réelle des données à travers le réseau. Ces périphériques reçoivent des instructions du contrôleur sur l'endroit où déplacer les données en fonction des politiques définies. Par exemple, si le contrôleur détermine qu'un paquet de données doit être envoyé d'un appareil à un autre, il envoie cette instruction aux périphériques réseaux appropriés pour qu'ils acheminent le paquet selon les spécifications du contrôleur [3].

### **2.1.3 Différence entre un réseau SDN et un réseau traditionnel**

Traditionnellement, un réseau informatique est composé d'équipements réseau interconnectés tels que des switchs et des routeurs. Dans chaque équipement réseau, il y a deux parties principales : une pour transmettre les données (la couche de transmission) et une pour gérer le fonctionnement de l'équipement (le plan de contrôle). Dans ce modèle, les équipements réseau sont fermés, ce qui empêche l'ajout facile de nouvelles applications ou fonctionnalités.

De plus, avec cette approche traditionnelle de gestion des réseaux, chaque équipement doit être configuré indépendamment. Ce qui est non seulement lourd et compliqué, mais aussi risqué car cela peut entraîner des configurations non uniformes. D'où le SDN pour permettre la centralisation des configurations.

Le SDN a pour idée de simplifier la gestion des réseaux et de quitter l'architecture distribuée, pour revenir à un système de gestion centraliser du réseau afin de gérer plus facilement les réseaux. La figure 2.1 donne un aperçu sur la différence entre l'architecture traditionnelle et le SDN [3].

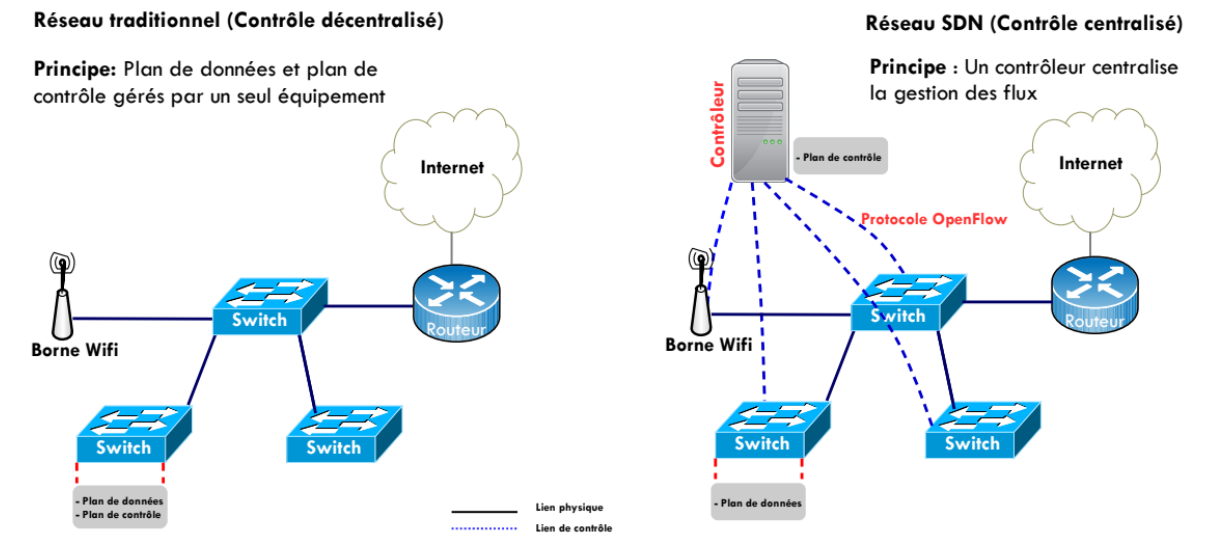


Figure 2.2 : Comparaison entre un réseau traditionnel et un réseau SDN [3].

## 2.2 Internet des objets (IOT)

### 2.2.1. Définition

L'Internet des Objets (IoT) désigne l'ensemble des objets capables de se connecter à un réseau Internet. Initialement, cette connectivité s'appuyait principalement sur le Wi-Fi, mais les avancées technologiques, notamment l'émergence de la 5G, ont révolutionné le domaine. Aujourd'hui, l'IoT concerne surtout des objets dotés de capteurs, de logiciels et d'autres technologies qui leur permettent d'échanger des données entre eux.

La 5G, avec ses vitesses de transmission élevées et sa faible latence, ouvre de nouvelles possibilités pour l'IoT en permettant de traiter de vastes ensembles de données de manière rapide et fiable, pratiquement partout. Cela facilite le développement de solutions avancées dans divers domaines, tels que la santé, les villes intelligentes, l'agriculture, et bien d'autres, en rendant l'IoT plus accessible et efficace.[4]

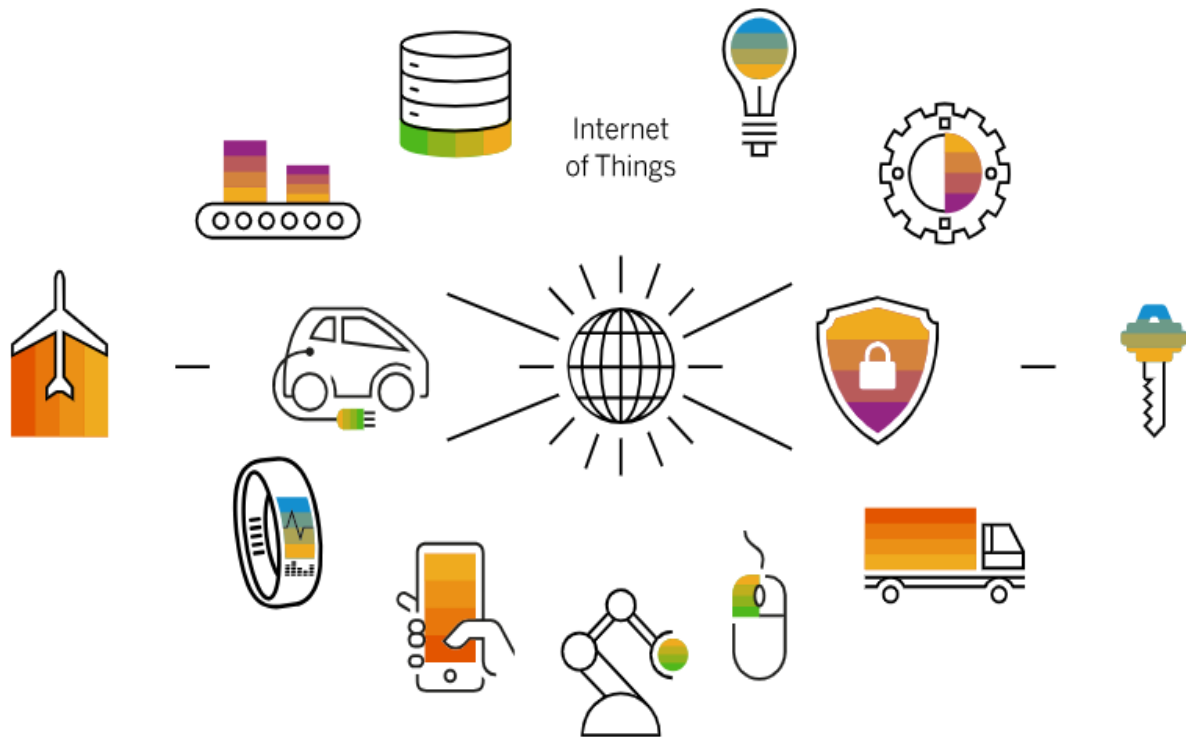


Figure 2.3 : Vue d'ensemble de l'internet des objets [4].

## 2.2.2. Avantages et Fonctionnement de l'IOT

L'Internet des Objets (IoT) offre une multitude d'avantages et fonctionne en intégrant des appareils physiques à Internet. Cette section explorera ces avantages sur le côté de la médecine ainsi que le fonctionnement de l'IoT.

### 2.2.2.1. Avantage de l'iot dans le domaine de la médecine

Le domaine de la médecine, et plus largement celui de la santé, connaît une évolution constante grâce à l'intégration des technologies de l'information et de la communication. L'avènement de l'Internet des objets (IoT) ouvre la voie à des avancées significatives. Les applications IoT permettent la surveillance à distance des informations de santé et de la condition physique des patients, le déclenchement d'alarmes en cas de conditions critiques, et dans certains cas, le contrôle à distance de certains traitements ou paramètres médicaux.

Cette solution permet une intervention rapide des professionnels de santé en cas de conditions critiques, améliore la qualité des soins grâce à un suivi continu des signes vitaux, réduit la charge de travail du personnel médical en automatisant la collecte et l'analyse des données, et optimise les ressources hospitalières en se concentrant sur les patients nécessitant une attention immédiate. Pour les patients, elle offre un confort et une sécurité accrus en leur permettant de rester chez eux tout en étant surveillés en permanence.

De cette manière, plusieurs avantages peuvent être mis en avant. Du point de vue du patient, cela permet une réduction du temps et des risques. Le patient peut rester chez lui et se rétablir tout en bénéficiant d'une supervision et d'une assistance en temps réel. Pour le centre hospitalier, cela se traduit par une réduction considérable des coûts grâce à la diminution des services nécessaires. De plus, les hôpitaux pourront traiter un plus grand nombre de patients sans nécessiter davantage de lits et de personnel [6].

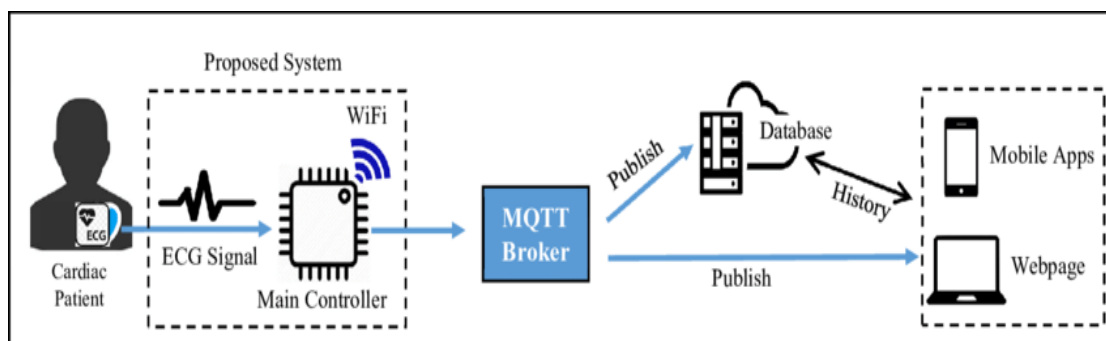


Figure2.4 : Exemple d'architecture de télésurveillance de patient

#### 2.2.2.2. Fonctionnement

Un système IoT classique fonctionne par le biais de la collecte et de l'échange de données en temps réel. Un système IoT se compose de trois éléments :

La figure 2.3 présente une vue d'ensemble du fonctionnement de l'Internet des objets.

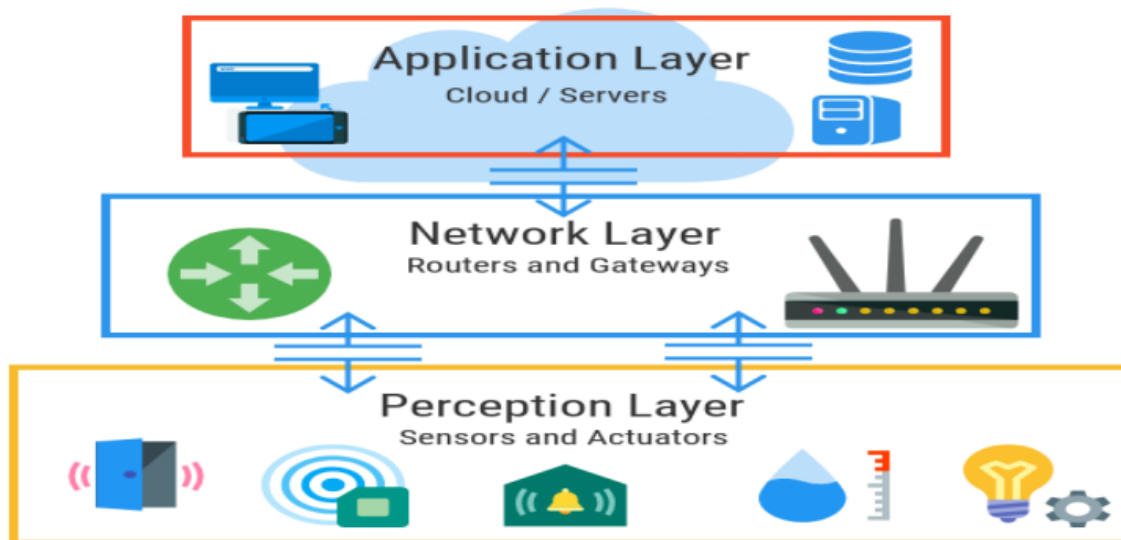


Figure 2.5 : Architecture de l'internet des objets [5].

### Explication de l'architecture

**Couche perception** La couche de perception est également connue sous le nom de couche « Capteurs » dans l'IoT. Le but de cette couche est d'acquérir les données de l'environnement à l'aide de capteurs et d'actionneurs. Cette couche détecte, collecte et traite les informations, puis les transmet à la couche réseau.

**Couche réseau** La couche réseau de l'IoT sert à la fonction de routage et de transmission des données vers différents hubs et appareils IoT sur Internet. À cette couche, les plates-formes de cloud computing, les passerelles Internet, les dispositifs de commutation et de routage ...etc.

**Couche application** La couche application est considérée comme une couche supérieure de l'architecture IoT conventionnelle. Cette couche fournit des services personnalisés en fonction des besoins des utilisateurs. La responsabilité principale de cette couche est de relier l'écart majeur entre les utilisateurs et les applications [5].

### **2.1.3. Intégration de l'IoT dans les Réseaux SDN**

Avec l'essor rapide des objets connectés, la gestion de ces dispositifs aux caractéristiques diverses pose un défi significatif pour les administrateurs de réseaux. Les réseaux traditionnels sont souvent rigides et peinent à s'adapter à la variété croissante d'appareils connectés, ainsi

qu'à la volumétrie de données et à la mobilité associées à l'IoT. Pour surmonter ces limitations, les chercheurs se sont tournés vers le Software-Defined Networking (SDN), une technologie qui révolutionne la gestion des réseaux en introduisant une séparation entre le plan de contrôle et le plan de données.

Contrairement aux réseaux conventionnels où le contrôle et les données sont intégrés, SDN permet une gestion centralisée et programmable du réseau, offrant une flexibilité et une adaptabilité accrues. Cette approche est particulièrement bénéfique pour l'IoT, car elle permet de mieux gérer la diversité des appareils connectés et de leur trafic.

Les architectures SDN pour l'IoT représentent ainsi une avancée majeure en facilitant la gestion de la sécurité. En centralisant les politiques de sécurité et en permettant une gestion dynamique des flux de données, SDN simplifie la sécurisation des objets connectés tout en améliorant leur efficacité opérationnelle [3].

## **2.3. Conclusion**

Ce chapitre a permis de mettre en lumière les concepts fondamentaux du SDN et de l'IoT, en exposant leurs principes, avantages et fonctionnement. L'intégration de l'IoT dans les réseaux SDN se révèle être une solution innovante pour relever les défis liés à la gestion des objets connectés dans des environnements complexes. Cette synergie, en combinant flexibilité, centralisation et adaptabilité, ouvre la voie à des réseaux intelligents et efficaces, notamment dans le domaine médical, où elle offre des perspectives prometteuses pour des soins connectés et personnalisés.

# **ETUDE DU PROTOCOLE VXLAN ET DES PROTOCOLE IOT**

## Chapitre 3 : Étude du protocole Vxlan et des protocole IoT

Dans ce chapitre, nous commencerons par aborder les principes fondamentaux du protocole VXLAN, notamment sa définition, ses objectifs, ses avantages et son fonctionnement, ainsi que ses différences par rapport au réseau VLAN. Ensuite, nous explorerons les principaux protocoles IoT et justifierons le choix du protocole MQTT.

### 3.1. Etude du protocole Vxlan

#### 3.1.1 Définition

Le Virtual extensible Local-Area Network(VXLAN) est une norme technologique de virtualisation des réseaux. Elle permet de partager un même réseau physique entre plusieurs organisations différentes, ou locataires, sans qu'aucun ne soit en mesure de voir le trafic réseau des autres. D'un point de vue technique, un VXLAN permet de segmenter un réseau physique en un maximum de 16 millions de réseaux virtuels ou logiques.

Comme le montre la figure 1-1, VXLAN est essentiellement une technologie de tunneling. Elle relie des appareils sur un réseau en encapsulant les paquets de données et en les envoyant à travers le tunnel. Les serveurs sont connectés à différents points du réseau VXLAN dans le centre de données, ce qui fonctionne comme un commutateur virtuel de niveau 2 [7].

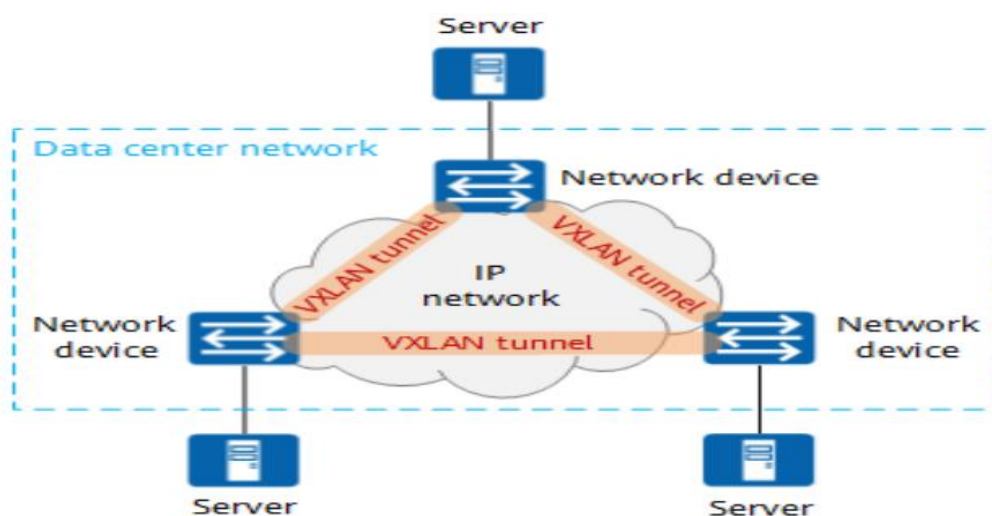


Figure 3.1 : Technologie de tunneling VXLAN [8].



### **3.1.2 Avantage et Fonctionnement de Vxlan**

VXLAN est une technologie de réseau qui permet de créer des tunnels pour transporter des données sur un réseau IP. Cette section examinera les avantages de VXLAN et expliquera comment il fonctionne.

#### **3.1.2.1 Avantage**

Le VXLAN est une composante cruciale pour répondre aux besoins spécifiques des centres de données modernes.

Premièrement, le VXLAN permet la migration dynamique des machines virtuelles (VM), facilitant ainsi la gestion des ressources et la maintenance sans interruption du service. Grâce à l'encapsulation du trafic dans des paquets IP standard, le VXLAN garantit la continuité de la connectivité réseau des VM, indépendamment de leur localisation physique dans le centre de données. Cela est essentiel pour assurer une disponibilité et une efficacité opérationnelle optimales dans un environnement médical où la réactivité et la fiabilité des systèmes sont critiques.

Deuxièmement, avec l'expansion du centre de données et l'augmentation du nombre de locataires, tels que différents services médicaux ou applications cliniques, il devient impératif d'assurer une isolation sécurisée des flux de données. Le VXLAN adresse cette nécessité en permettant l'isolement virtuel des réseaux logiques au sein du même infrastructure physique. Chaque locataire peut ainsi bénéficier d'un réseau dédié et sécurisé, ce qui est essentiel pour respecter les exigences de confidentialité et de sécurité des données médicales sensibles.

#### **3.1.2.2 Fonctionnement**

Le protocole de tunnelisation VXLAN encapsule les trames Ethernet de niveau 2 dans des paquets UDP de niveau 4, ce qui permet de créer des sous-réseaux virtualisés de niveau 2 qui s'étendent sur des réseaux physiques de niveau 3. Chaque sous-réseau segmenté est identifié par un identifiant de réseau VXLAN (VNI) unique.

L'entité responsable de l'encapsulation et de la décapsulation des paquets s'appelle le point de terminaison de tunnel VXLAN (VTEP). Un VTEP peut être un équipement réseau indépendant, tel qu'un routeur ou un commutateur matériel, ou un commutateur virtuel déployé sur un serveur.

Les VTEP encapsulent des trames Ethernet dans des paquets VXLAN, qui sont ensuite envoyés au VTEP de destination sur un réseau IP ou tout autre réseau de niveau 3. Ils sont alors décapsulés et transférés au serveur de destination.

Pour prendre en charge les équipements qui ne peuvent pas fonctionner comme des VTEP de manière autonome, tels que les serveurs bare metal, des VTEP matériels tels que certains commutateurs et routeurs Juniper peuvent encapsuler et décapsuler les paquets de données. De plus, les VTEP peuvent résider sur des hôtes hyperviseurs, comme des machines virtuelles basées sur un noyau (KVM), afin de prendre directement en charge les charges de travail virtualisées. Ce type de VTEP est connu sous le nom de VTEP logiciel [8].

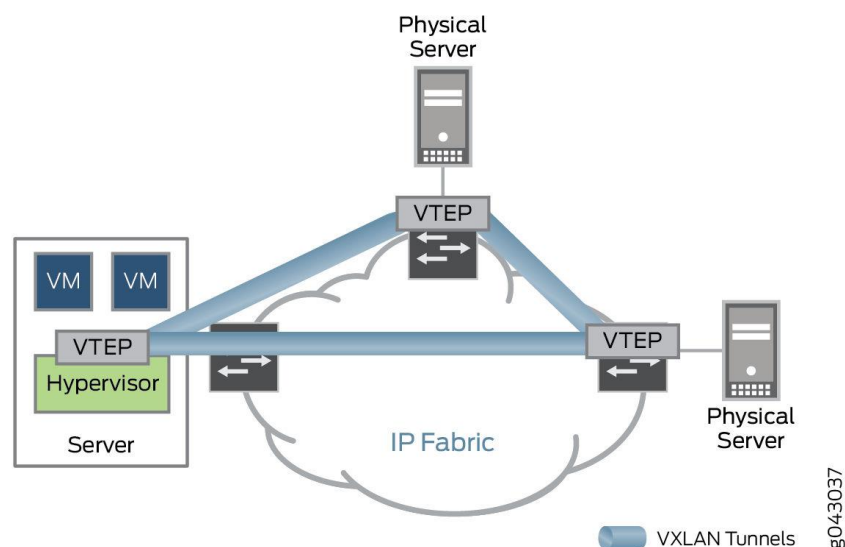


Figure 3.2 : Fonctionnement de VXLAN

### 3.1.3 Différence entre VxLan et VLAN

Le VLAN est une technologie qui isole les réseaux traditionnels en utilisant des identifiants de VLAN. Cependant, il présente des limitations. Il ne peut supporter qu'environ 4 000 VLANs, ce qui est insuffisant pour isoler efficacement les locataires dans de grands centres de données. De plus, chaque VLAN est un petit réseau virtuel fixe de couche 2, ce qui ne permet pas la migration dynamique à grande échelle des machines virtuelles (VM).

En revanche, VXLAN surmonte ces limitations. VXLAN utilise un champ d'identification appelé VNI (Virtual Network Identifier) de 24 bits, ce qui permet d'identifier

jusqu'à 16 millions de locataires. Cela représente une nette augmentation par rapport aux 4 000 locataires possibles avec les VLANs. VXLAN crée des tunnels virtuels à travers le réseau IP sous-jacent du centre de données, transformant ainsi le réseau en un grand commutateur virtuel de couche 2. Cela répond aux besoins de migration dynamique à grande échelle des VM, facilitant ainsi la flexibilité opérationnelle et la gestion des ressources dans un environnement de centre de données moderne.

Bien que VXLAN soit basé sur le concept de VLAN, il offre des fonctionnalités étendues qui le rendent plus adapté aux grands centres de données, en particulier pour la virtualisation et la gestion efficace des réseaux dans des environnements dynamiques.

## **3.2. Etude des protocole IOT**

Les protocoles de communication sont comme des règles que les appareils suivent pour se parler. Ces règles définissent comment les messages doivent être structurés (syntaxe), ce qu'ils signifient (sémantique), et comment s'assurer que tout se passe au bon moment (synchronisation). On peut les comparer aux langues humaines : chaque protocole a ses propres règles pour que les appareils puissent se comprendre.

Ces protocoles sont essentiels car ils assurent que les appareils IoT peuvent échanger des informations de manière fiable et efficace, tout comme les langues humaines nous permettent de communiquer entre nous [5].

Dans le domaine de l'IoT, il existe plusieurs protocoles populaires, nous examinerons quelques-uns de ces protocoles populaires dans la section suivante.

### **3.2.1. Étude de divers protocoles IoT**

**Hypertext Transfer Protocol :** Le protocole de transfert hypertexte (Hypertext Transfer Protocol : HTTP) est le protocole de communication derrière le World Wide Web (WWW). Il est basé sur une architecture client-serveur et fonctionne à la manière des demandes et des réponses. HTTP utilise TCP (protocole de contrôle de transmission) pour fournir des connexions fiables. HTTP est un protocole sans état, car le client et le serveur ne maintiennent pas de connexion pendant la communication.



**Websocket 27 :** WebSocket est un protocole de communication conçu pour les navigateurs Web et les serveurs Web, mais contrairement à HTTP, WebSocket fournit une communication en duplex intégral sur une seule connexion TCP. WebSocket est avec état, car le client et le serveur maintiennent une connexion pendant la communication. Le WebSocket permet une plus grande interaction entre un navigateur et un serveur Web, permet un transfert de données en temps réel et des flux de messages. À ce jour, WebSocket est implémenté dans tous les principaux navigateurs Web, par exemple Firefox 6, Safari 6, Google Chrome 14, Opera 12.10 et Internet Explorer



**MQ Telemetry transport (MQTT) :** MQ Telemetry transport (MQTT) est un protocole de communication machine à machine légère conçue pour les appareils IoT par IBM.



MQTT est basé sur un modèle publisher-subscriber, où publisher publie des données sur un serveur (également appelé broker), et le subscriber s'abonne au serveur et reçoit des données du serveur. Le broker MQTT est responsable de la distribution des messages et peut-être quelque part dans le Cloud[5].

### 3.2.2 Choix du protocole MQTT

Dans le domaine de l'Internet des Objets (IoT), le choix d'un protocole de communication fiable et adapté est essentiel pour garantir l'efficacité, la sécurité et la scalabilité des échanges entre appareils connectés. Le **protocole MQTT (Message Queuing Telemetry Transport)** se distingue par ses nombreux avantages, en particulier pour les systèmes nécessitant une communication légère, flexible et optimisée pour les réseaux à faible bande passante. Il répond aux contraintes des dispositifs IoT, souvent limités en ressources, et s'impose comme un standard dans la transmission de données pour de multiples applications.

Le choix du protocole MQTT repose sur trois éléments clés : son importance, car il permet des échanges rapides tout en économisant la bande passante et l'énergie, ce qui est parfait pour les appareils aux ressources limitées ; son principe, basé sur un modèle de publication et d'abonnement qui simplifie les échanges sans communication directe entre les appareils ; sa fiabilité, grâce à des niveaux de qualité de service (QoS) qui garantissent la livraison des messages, même en cas de perturbations du réseau. ; et son fonctionnement, utilisant un agent centralisé appelé courtier qui reçoit et transmet les messages, assurant ainsi une communication fiable même en cas de connexion instable [9].

#### Importance

Le protocole MQTT est devenu une norme pour la transmission de données IoT, car il offre les avantages suivants [9] :

**Léger et efficace** : L'implémentation de MQTT sur l'appareil IoT nécessite des ressources minimales, de sorte qu'il peut même être utilisé sur de petits microcontrôleurs. Par exemple, un message de contrôle MQTT minimal peut ne contenir que deux octets de données. Les en-têtes des messages MQTT sont également de petite taille afin que vous puissiez optimiser la bande passante du réseau.

**Évolutif** : L'implémentation de MQTT nécessite une quantité minimale de code qui consomme très peu d'énergie en fonctionnement. Le protocole possède également des fonctionnalités intégrées pour prendre en charge la communication avec un grand nombre d'appareils IoT cela signifie que ce protocole est conçu pour pouvoir gérer facilement un grand nombre d'appareils

sans trop compliquer ou ralentir le système. Vous pouvez donc implémenter le protocole MQTT pour vous connecter à des millions de ces appareils.

**Fiable** : De nombreux appareils IoT se connectent sur des réseaux cellulaires peu fiables, avec une faible bande passante et une forte latence. MQTT possède des fonctionnalités intégrées qui réduisent le temps que le dispositif IoT prend pour se reconnecter au cloud. Il définit également trois niveaux de qualité de service différents pour garantir la fiabilité des cas d'utilisation IoT : au plus une fois (0), au moins une fois (1) et exactement une fois (2).

Voici comment MQTT assure cette fiabilité :

### **Reconnexion rapide**

Si la connexion est interrompue, MQTT permet à l'appareil de **se** reconnecter rapidement au serveur (cloud). Cela signifie que même si le réseau tombe en panne ou devient lent, l'appareil IoT pourra se reconnecter facilement et continuer à envoyer ses données sans trop de retard.

### **Niveaux de qualité de service (QoS)**

MQTT propose trois niveaux de qualité de service (QoS) pour garantir que les messages sont envoyés de manière fiable. Cela permet de choisir à quel point on veut être sûr que le message est bien reçu.

#### **Au plus une fois (QoS 0) :**

- Le message est envoyé une seule fois sans confirmation. C'est rapide, mais il y a un risque que le message se perde si la connexion est mauvaise.
- Utilisé pour les situations où perdre un message n'est pas grave.

#### **Au moins une fois (QoS 1) :**

- Le message est envoyé et confirmé au moins une fois. L'appareil s'assure que le message est bien reçu, même s'il doit le renvoyer plusieurs fois.
- C'est un bon compromis entre fiabilité et performance.

#### **Exactement une fois (QoS 2) :**

- Le message est envoyé une seule fois et confirmé de manière à ce qu'il ne soit ni perdu, ni dupliqué. C'est le niveau le plus fiable, mais il est un peu plus lent à cause des multiples étapes de confirmation.
- Utilisé pour des situations où chaque message est critique (comme les données médicales).

**Sécurisé :** MQTT permet aux développeurs de chiffrer facilement les messages et d'authentifier les appareils et les utilisateurs à l'aide de protocoles d'authentification modernes, tels que OAuth, TLS1.3, Customer Managed Certificates, etc.

Bonne prise en charge

Plusieurs langages comme Python disposent d'un support étendu pour implémenter le protocole MQTT. Les développeurs peuvent donc l'implémenter rapidement avec un codage minimal dans tout type d'application.

## Principe

MQTT (Message Queuing Telemetry Transport) fonctionne différemment des modèles de communication réseau traditionnels où un client demande des données à un serveur, et le serveur répond directement au client. Dans MQTT, on utilise un modèle de publication/abonnement (pub/sub), qui permet de découpler l'expéditeur du message (appelé éditeur) du récepteur (appelé abonné) en utilisant un agent de messagerie (appelé courtier) [9].

**Voici les principaux éléments du modèle de MQTT :**

- **Éditeur :** C'est l'appareil ou l'application qui envoie des messages sur un sujet particulier. Par exemple, un capteur qui envoie des données de température à un serveur MQTT.
- **Abonné :** C'est l'appareil ou l'application qui écoute les messages envoyés sur un sujet spécifique. Par exemple, un tableau de bord qui affiche les données de température reçoit les messages publiés par le capteur.

- **Courtier (broker)** : Le courtier est le composant central qui gère la communication entre les éditeurs et les abonnés. Il reçoit tous les messages des éditeurs et les transmet aux abonnés appropriés en fonction des sujets auxquels ils sont abonnés.

Contrairement à la communication client-serveur traditionnelle, où le client et le serveur doivent se connaître (via des adresses IP, des ports, etc.), MQTT introduit plusieurs types de découplages entre l'éditeur et l'abonné, ce qui facilite la gestion de la communication dans les systèmes distribués et l'Internet des objets (IoT).

- **Découplage spatial** : L'éditeur et l'abonné n'ont pas besoin de connaître l'emplacement l'un de l'autre. Ils ne communiquent pas directement, mais via le courtier de messages. Par exemple, un capteur IoT qui publie des données n'a pas besoin de connaître l'adresse IP du dispositif qui les recevra.
- **Découplage temporel** : L'éditeur et l'abonné n'ont pas besoin d'être connectés en même temps. Cela signifie qu'un capteur peut publier des messages même si l'abonné n'est pas connecté à ce moment-là. Le courtier peut stocker les messages et les livrer lorsque l'abonné se reconnecte.
- **Découplage de la synchronisation** : L'éditeur et l'abonné peuvent envoyer ou recevoir des messages de manière asynchrone. Cela signifie que l'éditeur n'a pas besoin d'attendre que l'abonné soit prêt pour recevoir les messages, et l'abonné peut récupérer les messages quand il est prêt, sans interrompre le processus de publication.

## **Composant**

MQTT implémente le modèle de publication/abonnement en définissant les clients et les agents comme ci-dessous [9].

- **Client MQTT** : Un client MQTT est un appareil ou une application qui utilise le protocole MQTT pour communiquer via un réseau. Cela peut être un serveur, un ordinateur, un microcontrôleur (comme l'ESP32), ou même une application mobile. Si le client envoie des messages, il agit comme un éditeur, et s'il reçoit des messages, il agit comme un récepteur. Fondamentalement, tout appareil qui communique à l'aide de MQTT sur un réseau peut être appelé un appareil client MQTT.



- **Agent MQTT** : L'agent MQTT est le système dorsal qui coordonne les messages entre les différents clients. Les responsabilités de l'agent comprennent la réception et le filtrage des messages, l'identification des clients abonnés à chaque message et l'envoi des messages à ces derniers. Il est également responsable d'autres tâches telles que :
  - Autoriser et authentifier les clients MQTT
  - Transmettre des messages à d'autres systèmes pour une analyse plus approfondie
  - Traiter les messages et les sessions clients manqués
- **Connexion MQTT** : Les clients et les agents commencent à communiquer en utilisant une connexion MQTT. Les clients initient la connexion en envoyant un message *CONNECT* à l'agent MQTT. L'agent confirme qu'une connexion a été établie en répondant par un message *CONNACK*. Le client MQTT et l'agent ont tous deux besoin d'une pile TCP/IP pour communiquer. Les clients ne se connectent jamais entre eux, uniquement avec l'agent.

## **Fonctionnement**

Le fonctionnement de MQTT est brièvement présenté ci-dessous [9].

1. Un client MQTT établit une connexion avec l'agent MQTT.
2. Une fois connecté, le client peut soit publier des messages, soit s'abonner à des messages spécifiques, soit faire les deux.
3. Lorsque l'agent MQTT reçoit un message, il le transmet aux abonnés qui sont intéressés.

Examinons les détails pour mieux comprendre.

### **Rubrique MQTT**

Le terme « rubrique » (topic) fait référence aux mots-clés que l'agent MQTT utilise pour filtrer les messages destinés aux clients MQTT. Les rubriques sont organisées de manière hiérarchique, à l'instar d'un répertoire de fichiers ou de dossiers. Par exemple, considérons un système de maison intelligente fonctionnant dans une maison à plusieurs niveaux qui possède différents appareils intelligents à chaque étage. Dans ce cas, l'agent MQTT peut organiser les rubriques comme suit :

*notremaison/rez-de-chaussée/salon/lumière*

*notremaison/premierétage/cuisine/température*

### **Publication MQTT**

Les clients MQTT publient des messages qui contiennent la rubrique et les données au format octet. Le client détermine le format des données, comme les données textuelles, les données binaires, les fichiers XML ou JSON. Par exemple, une lampe dans le système de maison intelligente peut publier un message *allumer* pour la rubrique *salon/lumière*.

### **Abonnement MQTT**

Les clients MQTT envoient un message *SUBSCRIBE* (S'ABONNER) à l'agent MQTT, pour recevoir des messages sur les rubriques qui les intéressent. Ce message contient un identifiant unique et une liste d'abonnements. Par exemple, l'application de maison intelligente sur votre téléphone veut afficher le nombre de lumières allumées dans votre maison. Il s'abonnera à la rubrique *lumière* et augmentera le compteur pour tous les messages *allumer*.

## **3.3. Conclusion**

Ce chapitre a permis d'explorer les protocoles VXLAN et MQTT, deux technologies essentielles pour la mise en place d'une plateforme de surveillance médicale à distance. VXLAN offre une solution de virtualisation du réseau sécurisée et flexible, tandis que MQTT permet une communication efficace et fiable entre les dispositifs IoT. Ces deux protocoles, en combinaison, répondent aux exigences de sécurité, de performance et de fiabilité nécessaires dans le domaine médical connecté.

## **CONCEPTION ET DEPLOIEMENT DE LA SOLUTION**

## Chapitre4 : Architecture et Mise en Œuvre de la Solution

Ce chapitre présente l'architecture de la solution ainsi que les étapes de déploiement et de configuration.

### 4.1 Schéma d'architecture de l'infrastructure

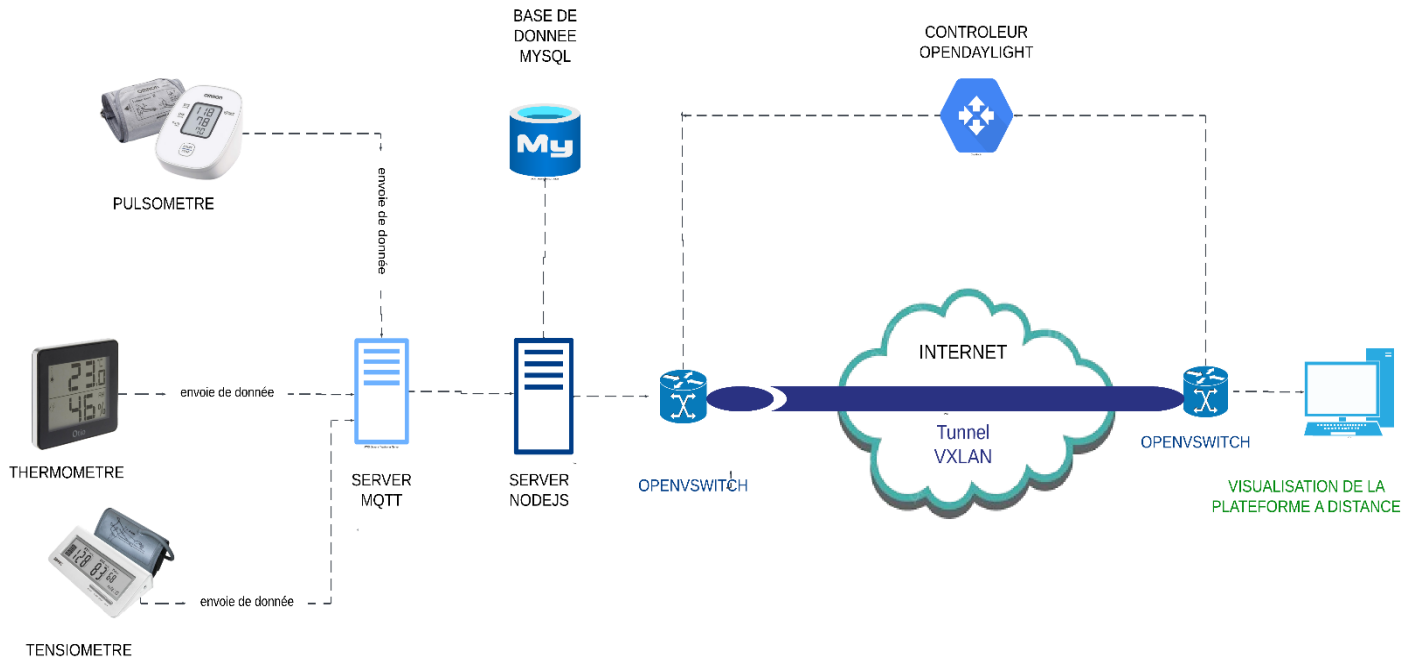


Figure 4.1 : Architecture générale du système

**L'idée de ce projet** est de développer une plateforme de télésurveillance médicale permettant de collecter, transmettre et visualiser les données de santé des patients en temps réel et en toute sécurité.

Trois dispositifs connectés à un **ESP32** via Wi-Fi sont utilisés pour collecter les données des patients : un pulsomètre (pour mesurer la fréquence cardiaque), un tensiomètre (pour mesurer la tension artérielle) et un thermomètre (pour mesurer la température corporelle).

Ces capteurs envoient leurs données au serveur MQTT qui transmet ces informations à un serveur **Node.js**. Ce serveur est responsable du traitement des données et de leur mise à disposition via une interface web accessible à distance.

Une base de données MySQL est utilisée pour stocker les informations médicales de manière structurée et sécurisée, permettant un suivi à long terme et une analyse détaillée.

Afin de permettre une transmission sécurisée des données à distance, un **Open vSwitch** est utilisé pour encapsuler les données dans un tunnel **VXLAN**. Ce mécanisme garantit que les données peuvent être consultées et visualisées par une machine distante en toute sécurité.

Le réseau défini par logiciel (**SDN**) connecte les deux **Open vSwitch** et permet d'appliquer des règles spécifiques pour gérer et optimiser le trafic réseau, assurant ainsi une qualité de service (QoS) adaptée.

Ce système offre au médecin la possibilité de visualiser les données vitales des patients (fréquence cardiaque, tension, température, ect...) en temps réel via une interface web sécurisée, améliorant ainsi la prise en charge des patients à distance.

## **4.2 Outils et Plateformes Utilisés**

Décrivons les outils et plateformes choisis pour la mise en œuvre, Ensuite nous expliquerons aussi leur rôle ou fonction dans le projet.

### **4.2.1 Présentation des outils et plateformes**

#### **4.2.1.1 Le microcontrôleur esp32**

L'ESP32 développé par la société Espressif, est une carte de développement à faible coût dédiée à l'internet des objets (IoT) et les applications embarquées. C'est un (SoC) system on a chip doté de communications sans fil Wifi et Bluetooth. Nos capteurs et nos actionneurs seront reliés à l'ESP32 dans notre système [5].

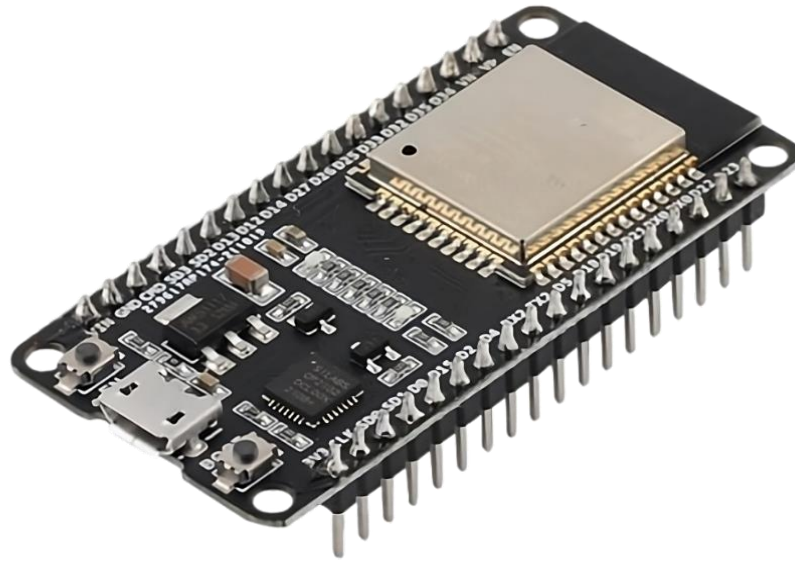


Figure 4.2 : esp32 [5].

Caractéristiques :

- Alimentation : 5 Vcc via micro-USB 3,3 Vcc via broches Vin
- Microprocesseur : Tensilica LX6 Dual-Core
- Fréquence : 240 MHz • Mémoire SRAM, Flash: 512 kB, 4 Mb
- E/S disponibles : 15 E/S digitales dont 10 compatibles PWM 2 x sorties analogiques (DAC) 15 x entrées analogiques (ADC)
- Interfaces : I2C, SPI, 2 x UART
- Interface Wifi 802.11 b/g/n 2,4 GHz • Bluetooth : Classique / BLE
- Antenne intégrée

#### 4.2.1.2 Plateforme Node.js

Node.js est un environnement d'exécution JavaScript côté serveur, basé sur le moteur V8 de Google Chrome. Il est conçu pour créer des applications réseau performantes grâce à son modèle d'E/S non bloquant, ce qui le rend idéal pour les opérations intensives en données, comme celles des applications IoT.

Dans notre projet, Node.js sert de plateforme pour développer une API qui collecte et transmet les données des capteurs connectés à l'ESP32. Sa capacité à gérer plusieurs connexions

simultanément permet une communication efficace entre les dispositifs IoT et le serveur. De plus, l'écosystème riche de bibliothèques, comme Express.js, facilite le développement de l'API et la gestion des requêtes HTTP, rendant la visualisation des données accessible et réactive pour les utilisateurs.

#### **4.2.1.3 Open vSwitch**

Open vSwitch (OVS) est un commutateur Ethernet logiciel open source conçu pour les environnements de machines virtuelles. Multicouche et distribué, il permet d'effectuer des fonctions de routage de niveau 2 et 3 du modèle OSI, tout en prenant en charge divers protocoles, dont OpenFlow, ce qui le rend compatible avec les réseaux SDN. Dans ce projet, elle nous a permis de faire communiquer des machines virtuelles Linux entre elles.

#### **4.2.1.4 Opendaylight**

OpenDaylight, développé par l'ONF avec le soutien de partenaires majeurs tels que Cisco et RedHat, est un contrôleur SDN (Software Defined Networking) qui permet de gérer et contrôler les équipements réseau via des protocoles comme OpenFlow. Pour ce projet, la version Beryllium-SR4 d'OpenDaylight a été utilisée, bien que des versions plus récentes soient disponibles. OpenDaylight offre une API REST pour l'interaction avec des applications externes, ce qui facilite le développement de solutions SDN.

Les modules d'OpenDaylight sont organisés en sous-projets, tels que openflowplugin pour OpenFlow et dlux pour l'interface graphique. Cette architecture modulaire, basée sur OSGi, permet de mettre à jour chaque composant individuellement si nécessaire, assurant ainsi flexibilité et simplification de la maintenance. Par ailleurs, l'installation de fonctionnalités spécifiques, comme le module dlux, est nécessaire pour visualiser et contrôler l'architecture réseau [3].

## INSTALLATION DE OPENDAYLIGHT

- ✚ Après avoir installé Java et le fichier d'archive de Karaf (version oxygen) requis pour OpenDaylight depuis leur site officiel, nous extrayons les deux fichiers téléchargés (Java et Karaf) dans le dossier de téléchargement.

```
root@yasmina-sdn-opendaylight:/home/yasmina2/Downloads# ls
jre-8u431-linux-x64.tar.gz  karaf-0.8.4.tar.gz
root@yasmina-sdn-opendaylight:/home/yasmina2/Downloads# tar -xvzf karaf-0.8.4.tar.gz
karaf-0.8.4/system/
karaf-0.8.4/system/org/
karaf-0.8.4/system/org/apache/
karaf-0.8.4/system/org/apache/karaf/
karaf-0.8.4/system/org/apache/karaf/features/
karaf-0.8.4/system/org/apache/karaf/features/framework/
```

```
root@yasmina-sdn-opendaylight:/home/yasmina2/Downloads# ls
jre-8u431-linux-x64.tar.gz  karaf-0.8.4  karaf-0.8.4.tar.gz
root@yasmina-sdn-opendaylight:/home/yasmina2/Downloads# tar -zxvf jre-8u431-linux-x64.tar.gz
jre1.8.0_431/COPYRIGHT
jre1.8.0_431/LICENSE
jre1.8.0_431/README
jre1.8.0_431/THIRDPARTYLICENSEREADME-JAVAFX.txt
jre1.8.0_431/THIRDPARTYLICENSEREADME.txt
jre1.8.0_431/Welcome.html
jre1.8.0_431/bin/jjs
```

```
root@yasmina-sdn-opendaylight:/home/yasmina2/Downloads# ls
jre1.8.0_431  jre-8u431-linux-x64.tar.gz  karaf-0.8.4  karaf-0.8.4.tar.gz
```

- ✚ Nous recherchons le chemin exact d'installation de Java

```
root@yasmina-sdn-opendaylight:/home/yasmina2/Downloads# whereis java
java: /usr/share/java
```

- ✚ Ouvrons le fichier .bashrc et ajoutez les lignes suivantes pour définir le chemin de Java et mettre à jour le PATH

```
GNU nano 6.2 .bashrc *
export JAVA_HOME=/usr/share/java/jre1.8.0_431
export PATH=$PATH:$JAVA_HOME/bin
```

- ✚ Déplacez le dossier extrait de Java vers /usr/java/home pour un accès global

```
root@yasmina-sdn-opendaylight:/home/yasmina2/Downloads# mv jre1.8.0_431 /usr/share/java/
root@yasmina-sdn-opendaylight:/home/yasmina2/Downloads# cd /usr/share/java/
root@yasmina-sdn-opendaylight:/usr/share/java# ls
jre1.8.0_431  libintl-0.21.jar  libintl.jar
root@yasmina-sdn-opendaylight:/usr/share/java#
```



- ✚ Pour appliquer les changements de variables d'environnement rechargeons les Modifications de .bashrc

```
root@yasmina-sdn-openshift:~# source .bashrc
root@yasmina-sdn-openshift:~# echo $JAVA_HOME
/usr/share/java/jre1.8.0_431
```

- 🚩 Lançons Karaf en exécutant la commande suivante dans le répertoire d'installation d'OpenDaylight

[illegible]

- ✚ Installer les d'épendances pour étendre les fonctionnalités de la plateforme SDN

```
opendaylight-user@root>feature:install odl-dlux-core
opendaylight-user@root>feature:install odl-mdsal-apidocs
opendaylight-user@root>feature:install odl-restconf odl-mdsal-all
opendaylight-user@root>feature:install odl-l2switch-all
opendaylight-user@root>feature:install odl-l2switch-switch-ui
opendaylight-user@root>feature:install odl-restconf-all
opendaylight-user@root>feature:install odl-openflowplugin-flow-services-rest
opendaylight-user@root>feature:install odl-openflowplugin-app-topology-manager
opendaylight-user@root>feature:install odl-dluxapps-nodes
```

🚦 Accéder à l'interface web dans un navigateur a cette adresse

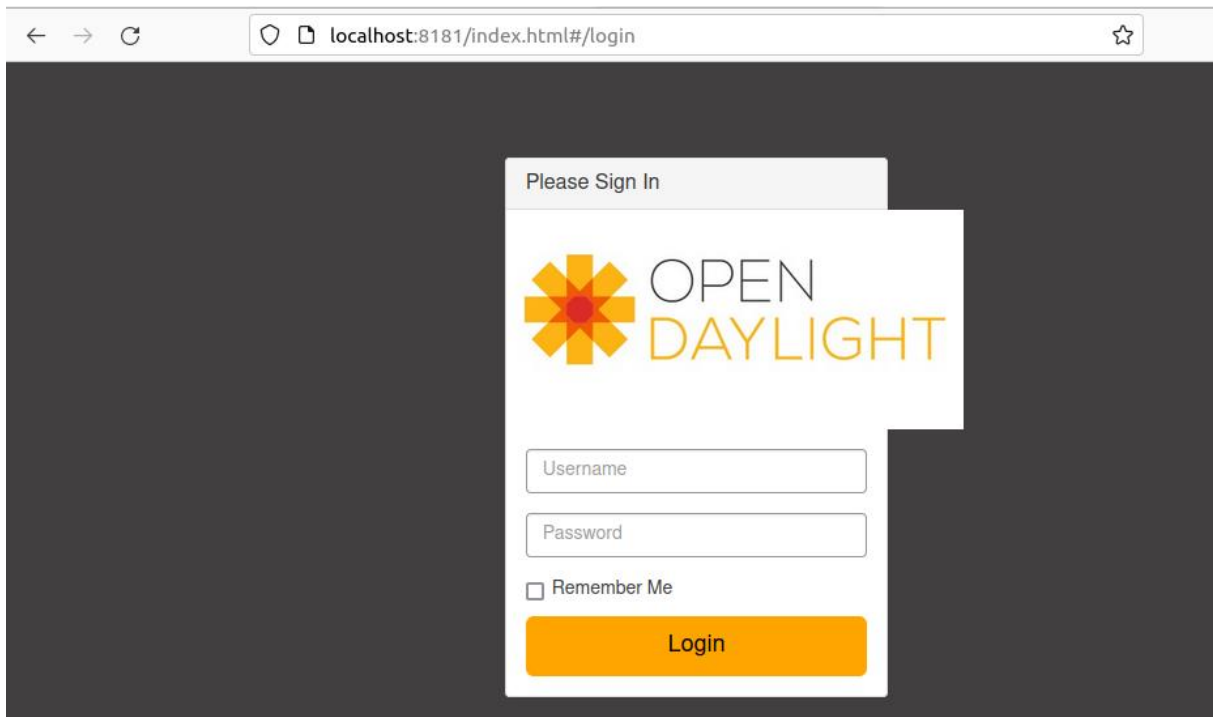


Figure 4.3 : Plateforme opendaylight

## 4.3 Déploiement de la Solution

Cette section détaille les étapes nécessaires à la configuration et à l'implémentation des différents composants de notre projet.

### 4.3.1 Déploiement et configuration des capteurs IoT sur ESP32

🚦 Tout d'abord je mets en place mon ESP32 sur le breadbord et le relie à mon capteur dht11 comme suite :

Connecter la broche VCC (+) du capteur DHT11 à la broche V5 de la carte ESP32.

Connectez la broche GND (-) du capteur DHT11 à la broche GND de la carte ESP32.

Connectez la broche de sortie de données du capteur DHT11 à la broche G26 de la carte ESP32.

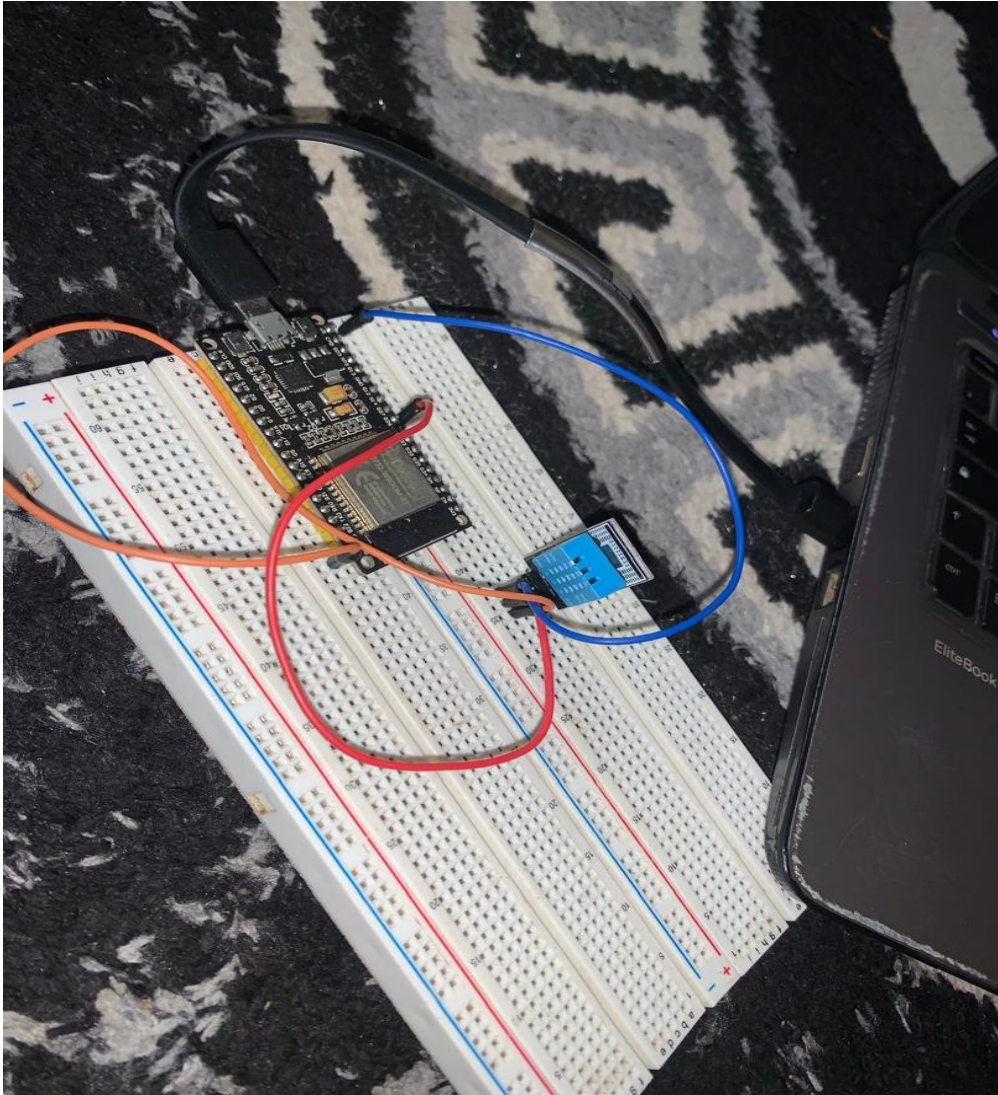


Figure 4.4: Intégration de l'ESP32 et du capteur

- ✚ L'Arduino IDE me permet d'écrire un code qui est destiné à être utilisé avec un ESP32 pour lire des données d'un capteur DHT11 (qui mesure la température) et envoyer ces données à un serveur MQTT via une connexion Wi-Fi

```

#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

// Définition du type de capteur de température et humidité
#define DHTTYPE DHT11 // DHT 11
const int DHTPin = 26; // Pin GPIO connecté au capteur DHT

// Configuration WiFi et MQTT
const char* ssid = "ESTM-ETAGEL";
const char* password = "Estm2017";
const char* mqtt_server = "server.blactachnos.com";
const char* Mqttusername = "invites"; // Nom d'utilisateur MQTT
const char* Mqttpassword = "invites"; // Mot de passe MQTT

WiFiClient espClient;
PubSubClient client(espClient);

DHT dht(DHTPin, DHTTYPE);

void setup_wifi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}

void reconnect() {
  while (!client.connected()) {
    if (client.connect("ESP8266Client", Mqttusername, Mqttpassword)) {
      Serial.println("Connected to MQTT broker");
    } else {
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(9600);
  dht.begin();
  setup_wifi();
  client.setServer(mqtt_server, 1883);
}

```

```

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    float t = dht.readTemperature();

    if (isnan(t)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    char temperatureTemp[7];
    dtostrf(t, 6, 2, temperatureTemp);

    client.publish("capteur/temperature", temperatureTemp);

    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.println(" *C");

    delay(3000); // Attendre 3 secondes entre les mesures
}

```

---

Figure 4.5 : Fichier de configuration du capteur et de l'ESP32

### **Explication du code :**

- ✚ Inclure des bibliothèques wifi.h (pour la connexion Wi-Fi de l'esp32) ; PubSubClient.h (Pour la communication MQTT) et DHT.h (pour interagir avec le capteur dht11)

Ensuite on définit le type de capteur utilisée et le pin auquel il s'est branché

Par la suite on défini le wi-fi auquel l'esp32 doit se connecter et le mot de passe

On configure le MQTT en définissant le nom de domaine de son server, le username et le mot de mot de passe

Et enfin on crée un client pour gérer la connexion Wi-fi, un autre pour le MQTT et crée un objet pour interagir avec le capteur

```

#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

// Définition du type de capteur de température et humidité
#define DHTTYPE DHT11 // DHT 11
const int DHTPin = 26; // Pin GPIO connecté au capteur DHT

// Configuration WiFi et MQTT
const char* ssid = "ESTM-ETAGEL";
const char* password = "Estm2017";
const char* mqtt_server = "server.blactachnos.com";
const char* Mqttusername = "invites"; // Nom d'utilisateur MQTT
const char* Mqttpassword = "invites"; // Mot de passe MQTT

WiFiClient espClient;
PubSubClient client(espClient);

DHT dht(DHTPin, DHTTYPE);

```

- ✚ Dans void setup\_wifi, le code permet de démarrer la connexion Wi-Fi avec les coordonnées qu'on a défini tout à l'heure et tant que l'esp32 n'est pas connecté la boucle continue à s'exécuter toute les 5s
- ✚ Dans void reconnect, le code permet de se connecter au server MQTT avec les identifiants qu'on a insérer tout à l'heure

```

void setup_wifi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {

```

```

        delay(500);
    }
}

void reconnect() {
    while (!client.connected()) {
        if (client.connect("ESP8266Client", Mqttusername, Mqtppassword)) {
            Serial.println("Connected to MQTT broker");
        } else {
            delay(5000);
        }
    }
}
}

```

✚ Dans void setup(), le code permet de configurer le capteur, d'appeler la fonction Wi-Fi qu'on a créé tout à l'heure et de se connecter au mqtt\_server au port 1883

✚ Dans void loop(), il vérifie si le client MQTT s'est connecter sinon il tente de se reconnecter

Ensuite on lit la valeur de la température et le stocker dans la variable t et si on a une erreur on affiche se message

Ensuite on publie sur le sujet capteur/temperature la valeur qu'on a convertie

Enfin on affiche la valeur de la température sur le moniteur série et attendre 3s avant de republier

```

void setup() {
    Serial.begin(9600);
    dht.begin();
    setup_wifi();
    client.setServer(mqtt_server, 1883);
}

```

```

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  float t = dht.readTemperature();

  if (isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  char temperatureTemp[7];
  dtostrf(t, 6, 2, temperatureTemp);

  client.publish("capteur/temperature", temperatureTemp);

  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C");

  delay(3000); // Attendre 3 secondes entre les mesures
}

```

---

## 4.3.2 Mise en place de la plateforme de visualisation et la base de donnée

### 4.3.2.1 Mise en place de la plateforme

🔧 D'abord on installe les paquets nodejs et npm



```

root@yasmina-vxlan-plateforme:~# sudo apt policy -y nodejs
nodejs:
  Installed: 12.22.9~dfsg-1ubuntu3.6
  Candidate: 12.22.9~dfsg-1ubuntu3.6
  Version table:
*** 12.22.9~dfsg-1ubuntu3.6 500
    500 http://nl.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages
    500 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages
    100 /var/lib/dpkg/status
 12.22.9~dfsg-1ubuntu3 500
    500 http://nl.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages
root@yasmina-vxlan-plateforme:~# sudo apt policy npm
npm:
  Installed: 8.5.1~ds-1
  Candidate: 8.5.1~ds-1
  Version table:
*** 8.5.1~ds-1 500
    500 http://nl.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages
    500 http://nl.archive.ubuntu.com/ubuntu jammy/universe i386 Packages
    100 /var/lib/dpkg/status

```

Figure 4.6 : Installation du serveur Node.js et NPM

- Après avoir installé nodejs et npm, on peut voir leurs version en tapant cette commande sur le terminal de vscode

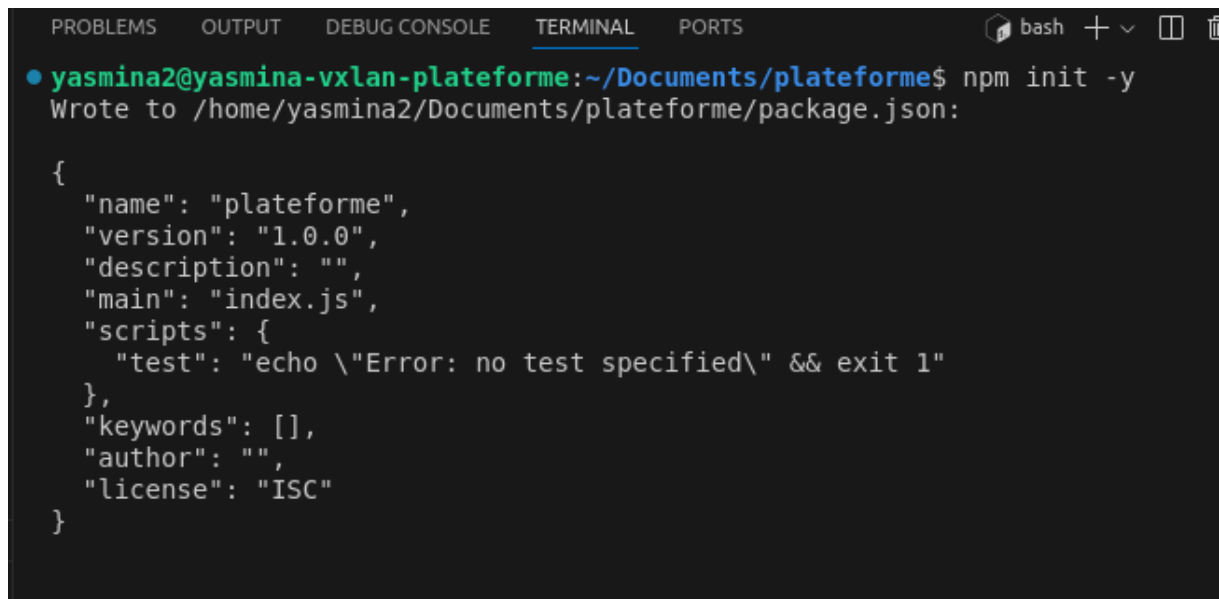
```

● yasmina2@yasmina-vxlan-plateforme:~/Documents/plateforme$ node -v
v22.11.0
● yasmina2@yasmina-vxlan-plateforme:~/Documents/plateforme$ npm -v
10.9.0
○ yasmina2@yasmina-vxlan-plateforme:~/Documents/plateforme$

```

Figure 4.7 : Vérification des versions de Node.js et de NPM

- Puis on initialise un nouveau projet nodejs sur le dossier plateforme crée dans le dossier Documents

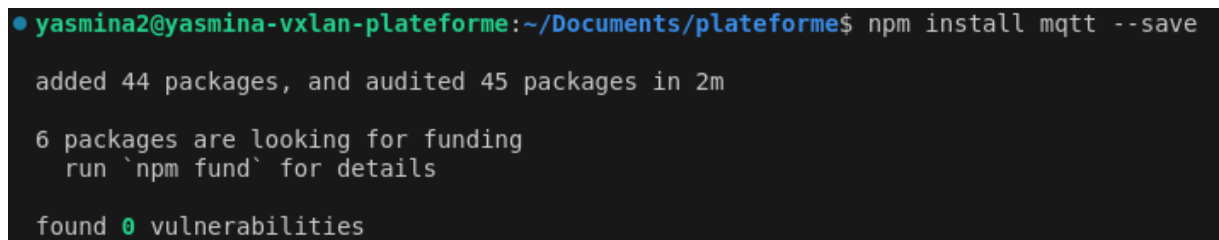


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● yasmina2@yasmina-vxlan-plateforme:~/Documents/plateforme$ npm init -y
Wrote to /home/yasmina2/Documents/plateforme/package.json:

{
  "name": "plateforme",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Figure 4.8 : Initialisation d'un nouveau projet Node.js

✚ Ensuite on Install le bibliothèque MQTT pour nodejs



```
● yasmina2@yasmina-vxlan-plateforme:~/Documents/plateforme$ npm install mqtt --save
added 44 packages, and audited 45 packages in 2m

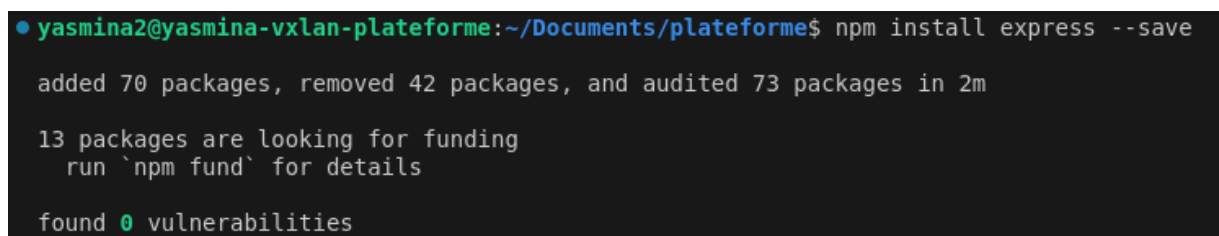
6 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Figure 4.9 Installation de la bibliothèque MQTT

✚ Ensuite on install express

✚ Express est un Framework pour Nodejs qui simplifie la création d'application, server et d'API



```
● yasmina2@yasmina-vxlan-plateforme:~/Documents/plateforme$ npm install express --save
added 70 packages, removed 42 packages, and audited 73 packages in 2m

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Figure 4.10 : Installation de la bibliothèque Express

✚ Ensuite on Install socket.io qui permet la communication bidirectionnelle entre le server et le client

```
● yasmina2@yasmina-vxlan-plateforme:~/Documents/plateforme$ npm install socket.io --save
added 19 packages, changed 1 package, and audited 64 packages in 1m

6 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Figure 4.11 : Installation de la bibliothèque Socket.IO

### **Code Serveur (app.js)**

Après avoir installer les bibliothèques nécessaire, on crée un fichier app.js et y insérer ce code qui nous permet de mettre en place une application web utilisant Express pour servir les fichiers statiques (comme index.html, style.css, script.js), Socket.io pour la communication en temps réel et MQTT pour recevoir les données des capteur

```

1 // Importation des bibliothèques
2 const mqtt = require('mqtt');
3 const express = require('express');
4 const http = require('http');
5 const socketIo = require('socket.io');
6 const path = require('path');
7 const mysql = require('mysql');
8 const bodyParser = require('body-parser'); // Pour parser les requêtes POST
9 const session = require('express-session'); // Pour gérer les sessions
10
11 const app = express();
12 const server = http.createServer(app);
13 const io = socketIo(server);
14
15 // Configuration du broker MQTT
16 const mqttBrokerUrl = 'mqtt://server.blactachnos.com'; // URL de votre broker MQTT
17 const mqttUsername = 'invites'; // Nom d'utilisateur MQTT
18 const mqttPassword = 'invites'; // Mot de passe MQTT
19
20 // Configuration de la connexion à MySQL
21 const db = mysql.createConnection({
22   host: 'localhost',
23   user: 'yasmina',
24   password: 'passer',
25   database: 'capteur'
26 });
27
28 // Établir la connexion MySQL
29 db.connect((err) => {
30   if (err) {
31     console.error('Erreur de connexion à MySQL :', err);

```

```

32   console.log('Connecté à MySQL');
33   });
34
35   // Connexion au broker MQTT
36   const mqttClient = mqtt.connect(mqttBrokerUrl, {
37     username: mqttUsername,
38     password: mqttPassword
39   });
40
41   mqttClient.on('connect', () => {
42     console.log('Connecté au broker MQTT');
43     mqttClient.subscribe('capteur/temperature');
44   });
45
46   mqttClient.on('message', (topic, message) => {
47     const temperature = parseFloat(message.toString());
48     console.log(`Message reçu : Température = ${temperature} °C`);
49
50     // Insérer la température dans la base de données
51     const query = 'INSERT INTO temperature (temperature) VALUES (?)';
52     db.query(query, [temperature], (err, result) => {
53       if (err) {
54         console.error('Erreur d\'insertion dans la base de données :', err);
55         return;
56       }
57       console.log('Température insérée dans la base de données');
58     });
59
60     // Envoyer les données au client via Socket.IO
61     io.emit('capteur/temperature', message.toString());
62   });

```

```

63 // Route pour gérer la connexion
64 app.post('/login', (req, res) => {
65     const { nom_utilisateur, mot_de_passe } = req.body;
66
67     const query = 'SELECT * FROM utilisateurs WHERE nom_utilisateur = ? AND mot_de_passe = MD5(?)';
68     db.query(query, [nom_utilisateur, mot_de_passe], (err, results) => {
69         if (err) {
70             console.error('Erreur lors de la requête SQL :', err);
71             return res.status(500).send('Erreur serveur');
72         }
73
74         if (results.length > 0) {
75             req.session.authenticated = true;
76             req.session.user = results[0];
77             return res.redirect('/dashboard');
78         } else {
79             return res.send('Nom d\'utilisateur ou mot de passe incorrect');
80         }
81     });
82 });
83
84 // Configuration de Socket.IO
85 io.on('connection', (socket) => {
86     console.log('Nouvelle connexion Socket.IO');
87 });
88
89 // Démarrer le serveur
90 const PORT = process.env.PORT || 3000;
91 server.listen(PORT, () => {
92     console.log(`Serveur démarré sur le port ${PORT}`);
93 });

```

Figure 4.12 : Code de configuration du serveur dans le fichier app.js

### Explication du code

- ✚ Tout d'abord on initialise les modules MQTT, express, http, socketIO, mysql et path (path est un outil essentiel en Node.js pour manipuler les chemins de fichiers de manière fiable et multiplateforme.)

```

JS app.js > ...
1 // Importation des bibliothèques
2 const mqtt = require('mqtt');
3 const express = require('express');
4 const http = require('http');
5 const socketIo = require('socket.io');
6 const path = require('path');
7 const mysql = require('mysql');

```

- ✚ Ensuite on initialise l'application express, express est utilisée pour créer un serveur http utilisant express et attacher socketIO au serveur http

```
9   const app = express();
10  const server = http.createServer(app);
11  const io = socketIo(server);
```

- ✚ Ensuite on donne les informations pour se connecter au serveur MQTT

```
15  // Configuration du broker MQTT
16  const mqttBrokerUrl = 'mqtt://server.blactachnos.com'; // URL de votre broker MQTT
17  const mqttUsername = 'invites'; // Nom d'utilisateur MQTT
18  const mqttPassword = 'invites'; // Mot de passe MQTT
```

- ✚ On donne aussi les informations pour se connecter à la base de données et poser la condition Si la connexion à la base de données est réussie, afficher "Connexion réussie", sinon afficher "Erreur de connexion".

```
18  // Configuration de la connexion à MySQL
19  const db = mysql.createConnection({
20    host: 'localhost',
21    user: 'yasmina',
22    password: 'passer',
23    database: 'capteur'
24  });
25
26  // Établir la connexion MySQL
27  db.connect((err) => {
28    if (err) {
29      console.error('Erreur de connexion à MySQL :', err);
30      return;
31    }
32    console.log('Connecté à MySQL');
33  });
```

- ✚ Cette partie permet de se connecter au broker MQTT avec ces identifiants

```
35 // Connexion au broker MQTT
36 const mqttClient = mqtt.connect(mqttBrokerUrl, {
37   username: mqttUsername,
38   password: mqttPassword
39 });
```

- ✚ Si la connexion du broker est réussie on affiche ce message dans la console et de s'abonner au sujet capteur/température pour recevoir les données

```
41 mqttClient.on('connect', () => {
42   console.log('Connecté au broker MQTT');
43   mqttClient.subscribe('capteur/température');
44 });
```

- ✚ Ce code écoute les messages MQTT pour recevoir la température, la convertit en nombre et l'affiche. Ensuite, il insère cette température dans une base de données MySQL et envoie la même donnée aux clients via **Socket.IO** pour une mise à jour en temps réel.

```
46 mqttClient.on('message', (topic, message) => {
47   const temperature = parseFloat(message.toString());
48   console.log(`Message reçu : Température = ${temperature} °C`);
49
50   // Insérer la température dans la base de données
51   const query = 'INSERT INTO temperature (temperature) VALUES (?)';
52   db.query(query, [temperature], (err, result) => {
53     if (err) {
54       console.error('Erreur d\'insertion dans la base de données :', err);
55       return;
56     }
57     console.log('Température insérée dans la base de données');
58   });
59
60   // Envoyer les données au client via Socket.IO
61   io.emit('capteur/température', message.toString());
62 });
```



- ✚ Pour cette ligne de code Express est utilisé pour servir des fichiers statiques depuis le répertoire courant c'est-à-dire tous les fichiers qui se trouve au même répertoire que app.js peuvent être accédés via le navigateur

```
64 // Servir les fichiers statiques
65 app.use(express.static(path.join(__dirname, '')));
```

- ✚ Là on a défini une route via Express qui permet que lorsqu'on accède à l'URL / il m'amène la page index.html

```
68 app.get('/', (req, res) => {
69   res.sendFile(path.join(__dirname, 'index.html'));
70 });
```

- ✚ Socket.io écoute les connexions entrantes par exemple si un client se connecte se message s'affiche dans la console

```
72 // Configuration de Socket.IO
73 io.on('connection', (socket) => {
74   console.log('Nouvelle connexion Socket.IO');
75 });
```

- ✚ Le serveur est démarré et prêt à recevoir des requêtes.

```
77 // Démarrer le serveur
78 const PORT = process.env.PORT || 3000;
79 server.listen(PORT, () => {
80   console.log(`Serveur démarré sur le port ${PORT}`);
81 });
```

### Code l'authentification (login.html)

✚ Pour l'authentification, on crée un formulaire

```
<body>
  <h1>Connexion</h1>
  <form action="/login" method="POST">
    <label for="nom_utilisateur">Nom d'utilisateur :</label>
    <input type="text" id="nom_utilisateur" name="nom_utilisateur" required>
    <br>
    <label for="mot_de_passe">Mot de passe :</label>
    <input type="password" id="mot_de_passe" name="mot_de_passe" required>
    <br>
    <button type="submit">Se connecter</button>
  </form>
</body>
```

Figure 4.13 : Code de configuration pour l'authentification dans le fichier login.html

### Code pour embellir la page de l'authentification (login.css)

✚ On crée sur le meme répertoire un dossier login.css pour embellir la page de l'authentification

```
<style>
  body {
    font-family: Arial, sans-serif;
    background-color: #f0f4f8;
    margin: 0;
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
    align-items: center;
    height: 100vh;
    padding-top: 50px;
  }

  h1 {
    color: #2c3e50;
    text-align: center;
    margin-bottom: 20px;
  }

  form {
    background-color: #fff;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 300px;
    max-width: 100%;
  }
```

```

input[type="text"], input[type="password"] {
  width: 100%;
  padding: 10px;
  margin-bottom: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
  font-size: 16px;
  background-color: #f9f9f9;
}

input[type="text"]:focus, input[type="password"]:focus {
  border-color: #3498db;
  outline: none;
}

button {
  width: 100%;
  padding: 10px;
  background-color: #3498db;
  color: white;
  border: none;
  border-radius: 5px;
  font-size: 16px;
  cursor: pointer;
  transition: background-color 0.3s;
}

```

Figure 4.14 : Fichier login.css pour la mise en forme de la page d'authentification

### Code Client (index.html)

- ✚ Pour le client on crée sur le même répertoire qu'avec app.js un fichier index.html qui permet aux utilisateurs de voir les données de température des capteurs en temps réel dans leur navigateur.

```

<> index.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Visualisation des Données</title>
7    <link rel="stylesheet" href="./style.css">
8  </head>
9  <body>
10   <h1>Données de Capteur</h1>
11   <div id="temperature"></div>
12
13   <script src="/socket.io/socket.io.js"></script>
14   <script>
15     const socket = io();
16
17     // Écouter les messages de température
18     socket.on('capteur/temperature', (message) => {
19       document.getElementById('temperature').innerText = `Température: ${message} °C`;
20     });
21
22   </script>
23 </body>
24 </html>

```

Figure 4.15 : Structure de la page d'accueil dans le fichier index.html

### Explication du code

✚ Cette partie crée un Titre et un div pour afficher les données de la température

```

10   <h1>Données de Capteur</h1>
11   <div id="temperature"></div>
12

```

✚ Ensuite on crée un script pour charger la bibliothèque client de Socket.io Cela permet à la page web d'établir une connexion en temps réel avec le serveur via Socket.IO. et d'initialiser la connexion à Socket.io .

```

12   <script src="/socket.io/socket.io.js"></script>
13   <script>
14     const socket = io();
15
16

```

- ✚ Et enfin on écoute les messages envoyer sur le server et le texte à l'intérieur de l'élément div avec l'ID temperature est mis à jour sur le contenu de la page avec les données reçu

```
17 // Écouter les messages de température
18 socket.on('capteur/temperature', (message) => {
19   document.getElementById('temperature').innerText = `Température: ${message} °C`;
20 });
```

### Code pour embellir la page html (style.css)

- ✚ On crée sur le même répertoire un fichier style.css pour appliquer des styles à la page web pour améliorer son apparence.

```
# style.css > #temperature
2  body {
3    font-family: Arial, sans-serif;
4    background-color: #f4f4f9;
5    margin: 0;
6    padding: 0;
7    display: flex;
8    flex-direction: column;
9    align-items: center;
10   justify-content: center;
11   height: 100vh;
12 }
13
14  h1 {
15    color: #333;
16    margin-bottom: 20px;
17  }
18
19  #temperature{
20    background-color: #fff;
21    border: 2px solid #ddd;
22    border-radius: 8px;
23    padding: 20px;
24    margin: 10px 0;
25    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
26    width: 300px;
27    text-align: center;
28    font-size: 1.2em;
29  }
```

```

31     #temperature {
32     | border-left: 8px solid ■ #ff7f50;
33     | }
34
35

```

Figure 4.16 : Fichier style.css pour la mise en forme de la page

### 4.3.2.2 Mise en place de la base de données

✚ Installation de mysql-server

```

root@yasmina-base-de-donnee:~# apt install mysql-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are
libflashrom1 libftdi1-2 libllvm13

```

Figure 4.17 : Installation de MySQL-Server

✚ On crée la base de donnée capteur, on crée aussi un utilisateur avec son mot de passe et lui allouer des privilèges

```

mysql> CREATE DATABASE capteur;
Query OK, 1 row affected (0,03 sec)

mysql> CREATE USER 'yasmina' IDENTIFIED BY 'passer';
Query OK, 0 rows affected (0,06 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,01 sec)

```

Figure 4.18 : Création de la Base de Donnée

```

mysql> GRANT ALL PRIVILEGES ON capteur.* TO 'yasmina'@'%';
Query OK, 0 rows affected (0,05 sec)

```

```

mysql> ALTER USER 'yasmina'@ '%' IDENTIFIED WITH mysql_native_password BY 'passer';
Query OK, 0 rows affected (0,01 sec)

```

- ✚ Puis on entre dans la base de données et y crée une table nommée température et 2 colonnes « température » pour la valeur de la température et une autre pour avoir la période qu'on l'a lu

```
mysql> USE capteur;
Database changed
mysql> CREATE TABLE temperature ( id INT AUTO_INCREMENT PRIMARY KEY,
temperature FLOAT, timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP );
Query OK, 0 rows affected (0,16 sec)
```

Figure 4.19 : Création de la table temperature dans la Base de Données

- ✚ Nous créons aussi une table utilisateurs avec 2 colonnes nom et mot de passe afin de permettre aux médecins de s'authentifier avant d'accéder à la plateforme. Cela garantit un contrôle d'accès sécurisé et personnalisé pour chaque utilisateur.

```
mysql> CREATE TABLE utilisateurs (
-> id INT AUTO_INCREMENT PRIMARY KEY,
-> nom_utilisateur VARCHAR(255) NOT NULL UNIQUE,
-> mot_de_passe VARCHAR(255) NOT NULL
-> );
Query OK, 0 rows affected (0,07 sec)
```

Figure 4.20 : Création de la table utilisateurs pour l'authentification sur la plateforme

- ✚ Nous ajoutons deux utilisateurs dans la table créée : yacine et admin. Ces comptes permettront de tester le mécanisme d'authentification et de valider l'accès à la plateforme.

```
mysql> INSERT INTO utilisateurs (nom_utilisateur, mot_de_passe)
-> VALUES ('admin', MD5('password123')); -- Utilisez un hachage pour plus de sécurité
Query OK, 1 row affected (0,01 sec)
```

```
mysql> INSERT INTO utilisateurs (nom_utilisateur, mot_de_passe)
-> VALUES ('yacine', MD5('yass123'));
Query OK, 1 row affected (0,01 sec)

mysql> select * from utilisateurs;
+----+-----+-----+
| id | nom_utilisateur | mot_de_passe |
+----+-----+-----+
| 1 | admin          | 482c811da5d5b4bc6d497ffa98491e38 |
| 2 | yacine         | b8f9377ecdf26b202dd896e07495d521 |
+----+-----+-----+
2 rows in set (0,01 sec)
```

Figure 4.21 : Création de deux utilisateurs pour l'authentification

### 4.3.3 Configuration du tunnel VXLAN entre les machines virtuelles

#### Au niveau de la 1ere machine

- ✚ D'abord j'ai installé openvswitch-switch

```
root@yasmina-vxlan-plateforme:~# sudo apt install openvswitch-switch
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openvswitch-switch is already the newest version (2.17.9-0ubuntu0.22.04.1).
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2 libllvm13
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
```

Figure 4.22 : Installation de OpenVSwitch sur la première machine

- ✚ Ensuite j'ai créé un pont OVS nommé switch1 à l'aide d'Open vSwitch.
- ✚ Puis uploader switch4 au niveau du contrôleur opendaylight qui a pour adresse 192.168.32.129
- ✚ Ensuite j'ai créé un tunnel vxlan en ajoutant un port au niveau du pont de type vxlan qui a pour interface nommé v1 et l'adresse du VTEP distant
- ✚ Après cela j'ai ajouté une adresse ip sur le pont pour la communication entre les 2 tunnels créés

```
root@yasmina-vxlan-plateforme:~# ovs-vsctl add-br switch1
root@yasmina-vxlan-plateforme:~# sudo ovs-vsctl set-controller switch1 tcp:192.168.32.129:6633
root@yasmina-vxlan-plateforme:~# ovs-vsctl add-port switch1 v1 -- set interface v1 type=vxlan option:remote_ip=192.168.32.133 ofport_request=10
root@yasmina-vxlan-plateforme:~# sudo ip addr add 10.10.10.1/24 dev switch1
root@yasmina-vxlan-plateforme:~# ifconfig switch1 up
root@yasmina-vxlan-plateforme:~# ifconfig
```

Figure 4.23 : Configuration de openvswitch sur la première machine

- ✚ En mettant ifconfig on verra bien le pont et le VxLan configurés



```

switch1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::1ccf:c1ff:fe14:ab44 prefixlen 64 scopeid 0x20<link>
    ether 1e:cf:c1:14:ab:44 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12 bytes 1641 (1.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vxlan_sys_4789: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 65000
    inet6 fe80::60f9:d3ff:fec1:a98 prefixlen 64 scopeid 0x20<link>
    ether 76:86:67:db:75:a2 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12 bytes 1473 (1.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 4.24 : Visualisation du pont OpenvSwitch créé sur la première machine

### Au niveau de la 2eme machine

✚ D'abord j'ai installé openvswitch-switch

```

root@yasmina-client:~# sudo apt install openvswitch-switch
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

```

Figure 4.25 : Installation de OpenVSwitch sur la deuxième machine

- ✚ Ensuite j'ai créé un pont OVS nommé switch2 à l'aide d'Open vSwitch).
- ✚ Puis uploader switch2 au niveau du contrôleur qui a pour adresse 192.168.32.129
- ✚ Ensuite j'ai créé un tunnel vxlan en ajoutant un port au niveau du pont de type vxlan qui a pour interface nommé v2 et l'adresse du VTEP distant
- ✚ Après cela j'ai ajouté une adresse ip sur le pont pour la communication entre les 2 tunnels créés

```

root@yasmina-client:~# ovs-vsctl del-br switch2
root@yasmina-client:~# ovs-vsctl add-br switch2
root@yasmina-client:~# sudo ovs-vsctl set-controller switch2 tcp:192.168.32.129:6633
root@yasmina-client:~# ovs-vsctl add-port switch2 v2 -- set interface v2 type=vxlan option:remote_ip=192.168.32.129
ofport_request=10
root@yasmina-client:~# sudo ip addr add 10.10.10.2/24 dev switch2
root@yasmina-client:~# ifconfig switch2 up
root@yasmina-client:~# ifconfig

```

Figure 4.26 : Configuration de openvswitch sur la deuxième machine

✚ En mettant ifconfig on verra bien le pont et le VxLan configurés

```

switch2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.2 netmask 255.255.255.0 broadcast 0.0.0.0
    inet6 fe80::c73:aff:fe35:1f42 prefixlen 64 scopeid 0x20<link>
    ether 0e:73:0a:35:1f:42 txqueuelen 1000 (Ethernet)
    RX packets 4 bytes 432 (432.0 B)
    RX errors 0 dropped 4 overruns 0 frame 0
    TX packets 41 bytes 5460 (5.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vxlan_sys_4789: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 65000
    inet6 fe80::18b5:7fff:feda:ec5a prefixlen 64 scopeid 0x20<link>
    ether 86:43:e6:33:4a:01 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 17 bytes 1666 (1.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 4.27 : Visualisation du pont OpenvSwitch créé sur la deuxième machine

### 4.3.4 Gestion du Réseau SDN via OpenDaylight

Visualisation de la Topologie des Ponts Créés sur OpenDaylight

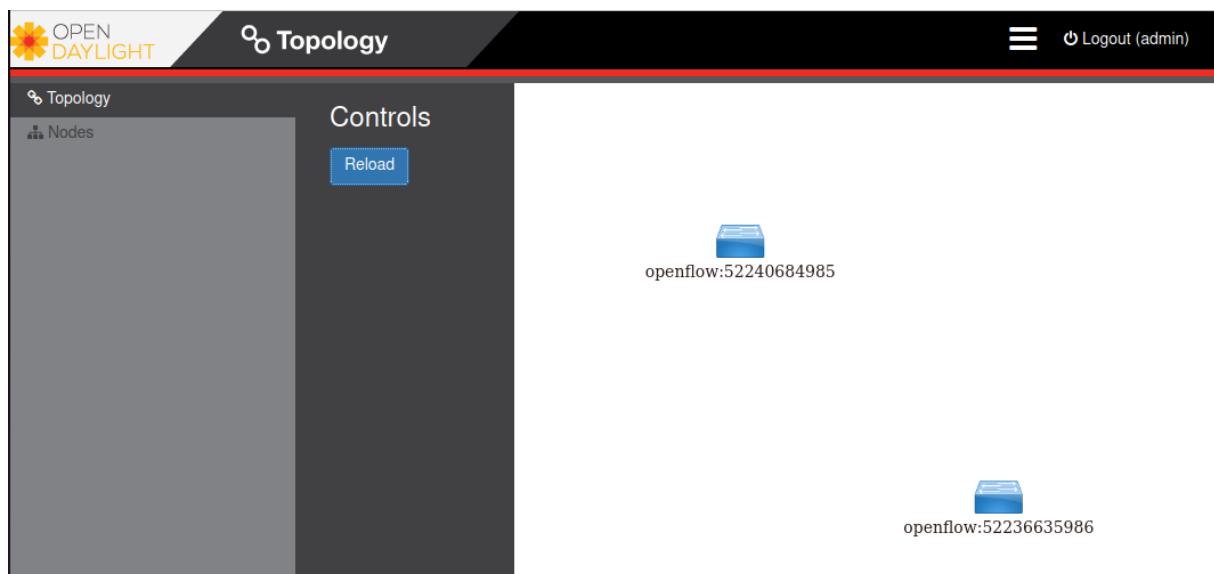


Figure 4.28 : Visualisation de la topologie des ponts configurés dans OpenDaylight

- ✚ Affichage des Ports de Tunnels VXLAN sur les Ponts Créés dans la plateforme.opendaylight
- ✚ **Pour switch1**

Node Connector Id	Name	Port Number	Mac Address
openflow:33877646420804:LOCAL	switch1	4294967294	1e:cf:c1:14:ab:44
openflow:33877646420804:10	v1	10	ca:9a:9d:90:09:8e

Figure 4.29 : Affichage des ports de tunnels VXLAN sur le premier dans la plateforme Opendaylight

#### Pour switch2

Node Connector Id	Name	Port Number	Mac Address
openflow:15887255281474:10	v2	10	fe:fd:f6:6b:f9:fc
openflow:15887255281474:LOCAL	switch2	4294967294	0e:73:0a:35:1f:42

Figure 4.30 : Affichage des ports de tunnels VXLAN sur le deuxième pont dans la plateforme Opendaylight

#### Configuration des Flux et Règles sur les Dispositifs ou Ponts Créés :

Pour empêcher la communication entre le client (switch2) et le serveur (switch1) via OpenFlow, on ajoute des règles de filtrage sur chaque switch.

#### Règle sur switch1 (serveur) :

Bloque le trafic du client (10.10.10.2) vers le serveur (10.10.10.1).

- **ovs-ofctl** est un utilitaire en ligne de commande utilisé pour gérer les flux OpenFlow dans Open vSwitch (OVS). Il permet de manipuler, d'afficher et de supprimer les règles de flux qui déterminent comment le trafic réseau est acheminé par un switch Open vSwitch.
- **in\_port=ens33** : Applique la règle aux paquets arrivant sur l'interface ens33 de switch1.
- **nw\_src=10.10.10.2** : La source est 10.10.10.2 (client).

- **nw\_dst=10.10.10.1** : La destination est 10.10.10.1 (serveur).
- **actions=drop** : Les paquets sont bloqués.

```
root@yasmina-vxlan-plateforme:~# sudo ovs-ofctl add-flow switch1 "in_port=ens33,dl_type=0x0800,nw_src=10.10.10.2,nw_dst=10.10.10.1,actions=drop"
```

Figure 4.31 : Configuration des règles sur le premier pont

#### 🚦 Règle sur switch2 (client) :

Bloque le trafic provenant du serveur (10.10.10.1) vers le client (10.10.10.2).

- **in\_port=ens33** : Applique la règle aux paquets arrivant sur l'interface ens33 de switch2.
- **dl\_type=0x0800** : Filtre les paquets IPv4.
- **nw\_src=10.10.10.1** : La source du paquet est 10.10.10.1 (serveur).
- **nw\_dst=10.10.10.2** : La destination est 10.10.10.2 (client).
- **actions=drop** : Les paquets correspondants sont bloqués.

```
root@yasmina-client:~# sudo ovs-ofctl add-flow switch2 "in_port=ens33,dl_type=0x0800,nw_src=10.10.10.1,nw_dst=10.10.10.2,actions=drop"
```

Figure 4.32 : Configuration des règles sur le deuxième pont

🚦 Après avoir ajouté ces règles, nous pouvons vérifier qu'elles ont bien été appliquées en utilisant la commande suivante sur chaque switch :

#### Sur switch1 (serveur) :

```
root@yasmina-vxlan-plateforme:~# sudo ovs-ofctl dump-flows switch1
cookie=0x0, duration=15.953s, table=0, n_packets=0, n_bytes=0, ip,in_port=ens33,nw_src=10.10.10.2,nw_dst=10.10.10.1 actions=drop
```

Figure 4.33 : Visualisation des règles configurées sur le premier pont

#### Sur switch2 (client) :

```
root@yasmina-client:~# sudo ovs-ofctl dump-flows switch2
cookie=0x0, duration=18.560s, table=0, n_packets=0, n_bytes=0, ip,in_port=ens33,nw_src=10.10.10.1,nw_dst=10.10.10.2 actions=drop
```

Figure 4.34 : Visualisation des règles configurées sur le deuxième pont

## **4.4. Conclusion**

Ce chapitre a présenté l'architecture et la mise en œuvre de la solution de télésurveillance médicale. Nous avons intégré divers composants matériels et logiciels pour collecter, transmettre et visualiser les données de santé des patients en temps réel, tout en assurant leur sécurité. La combinaison de l'ESP32, de Node.js, de MQTT, d'Open vSwitch et d'OpenDaylight permet de répondre aux besoins de la télémédecine, offrant ainsi une solution fiable et sécurisée pour le suivi à distance des patients.

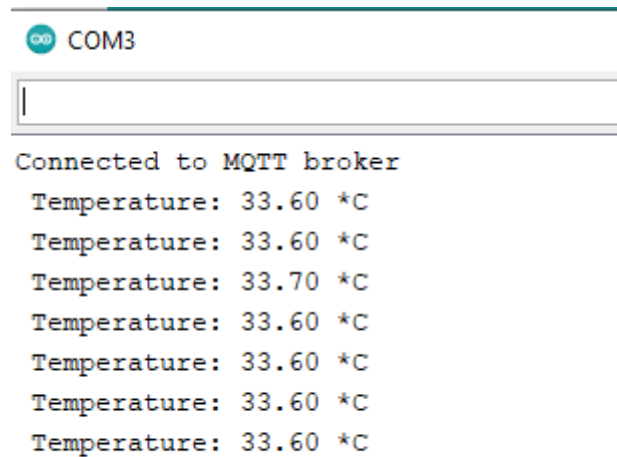
## **VALIDATION ET TESTS DE LA SOLUTION**

## Chapitre5 : Tests de la solution

Le dernier chapitre présente les tests réalisés pour valider la solution. Les tests incluent la transmission des données capteurs, la connectivité et la sécurité du tunnel VXLAN, ainsi que la vérification des règles de gestion réseau via le contrôleur SDN.

### 5.1 Tests de Transmission des Données Capteurs vers le Serveur MQTT

- Après avoir compilé et téléversé le code sur l'ESP32, les tests ont confirmé que l'appareil se connecte correctement au réseau Wi-Fi et au broker MQTT. Les données de température et d'humidité mesurées par le capteur DHT11 sont publiées en temps réel sur le topic MQTT configuré. Les résultats montrent que la transmission des données est fluide et stable.



```
COM3
Connected to MQTT broker
Temperature: 33.60 *C
Temperature: 33.60 *C
Temperature: 33.70 *C
Temperature: 33.60 *C
Temperature: 33.60 *C
Temperature: 33.60 *C
Temperature: 33.60 *C
```

Figure 5.1 : Test de connexion de l'ESP32 au réseau Wi-Fi et au broker MQTT avec transmission des données du capteur DHT11

### 5.2 Visualisation sur Plateforme et Stockage en Base de Données

- Une fois le code terminé, nous avons lancé le serveur Node.js à l'aide de la commande appropriée (node app.js). Le serveur s'est connecté avec succès au broker MQTT et a commencé à recevoir les données publiées sur le topic configuré.

```
o yasmina2@yasmina-base-de-donnee:~/Documents/plateforme$ node app.js
Serveur web Node.js en cours d'exécution sur le port 3000
Connecté à la base de données MySQL
Connecté au broker MQTT
Message reçu sur le sujet capteur/temperature: 33.60
Température insérée dans la base de données
Message reçu sur le sujet capteur/temperature: 33.60
Température insérée dans la base de données
Message reçu sur le sujet capteur/temperature: 33.60
Température insérée dans la base de données
```

Figure 5.2 : Lancement du serveur Node.js et connexion au broker MQTT pour recevoir les données

- ✚ En accédant à l'interface de la plateforme via un navigateur web (en entrant l'adresse IP du serveur suivie du port par défaut), le médecin doit s'authentifier en utilisant les informations stockées dans la base de données. Pour vérifier le bon fonctionnement du système d'authentification, je me suis authentifié avec l'utilisateur admin

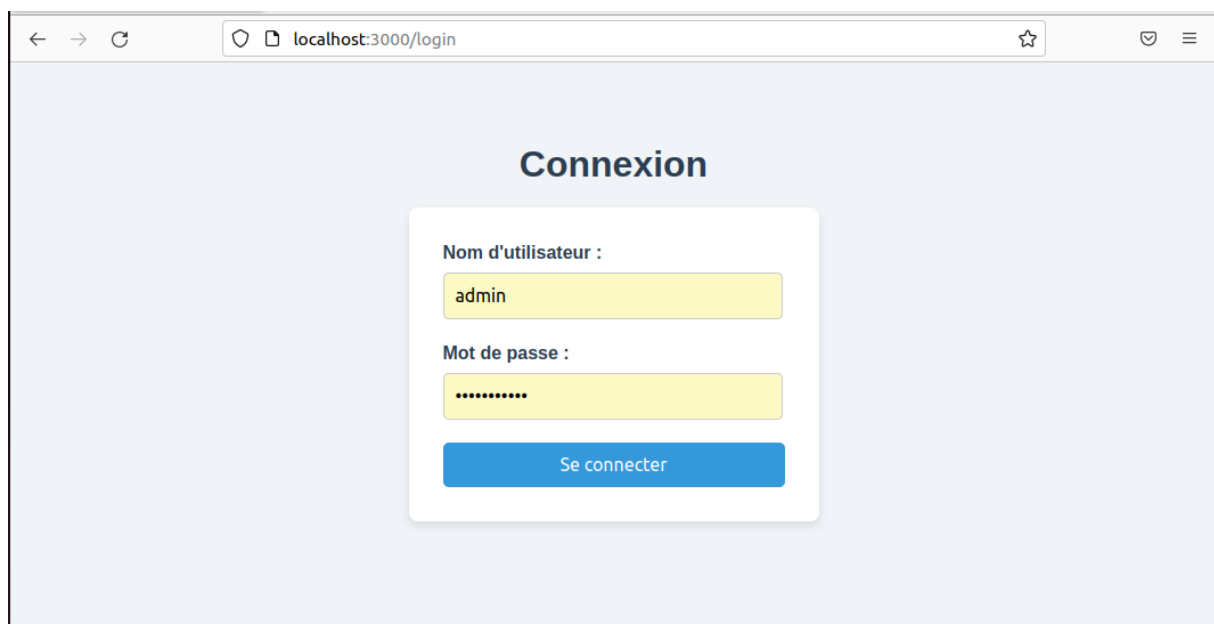


Figure 5.3 : Authentification à la plateforme avec l'utilisateur admin

- ✚ Après l'authentification, nous avons pu visualiser les données du capteur en temps réel. La page affiche les valeurs de température mesurées par le capteur, et celles-ci sont actualisées dynamiquement.



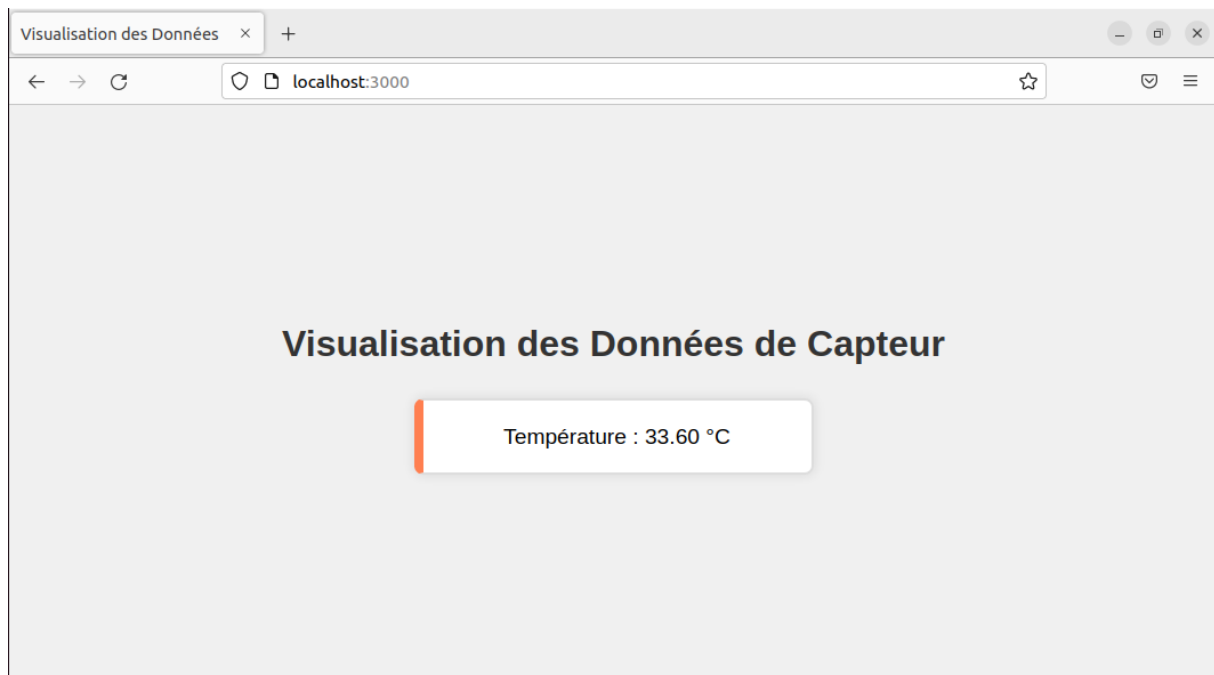


Figure 5.4 : Visualisation en temps réel des données du capteur via l'interface web de la plateforme

✚ Pour la partie stockage, nous avons également vérifié que les données étaient correctement enregistrées dans la base de données MySQL. Chaque nouvelle mesure de température reçue était insérée dans une table dédiée, permettant une conservation des valeurs pour une analyse ultérieure.

```
mysql> select * from temperature;
```

id	temperature	timestamp
490	33.6	2024-11-09 17:23:20
491	33.6	2024-11-09 17:23:23
492	33.6	2024-11-09 17:23:26
493	33.6	2024-11-09 17:23:29
494	33.6	2024-11-09 17:23:32
495	33.7	2024-11-09 17:23:35
496	33.6	2024-11-09 17:23:38
497	33.6	2024-11-09 17:23:41
498	33.6	2024-11-09 17:23:44
499	33.7	2024-11-09 17:23:47
500	33.7	2024-11-09 17:23:50

11 rows in set (0,00 sec)

Figure 5.5 : Enregistrement des données de température dans la base de données MySQL pour conservation et analyse

### 5.3 Tests de la connectivité et de la sécurité du tunnel VXLAN et la visualisation de la plateforme sur la machine client a distance

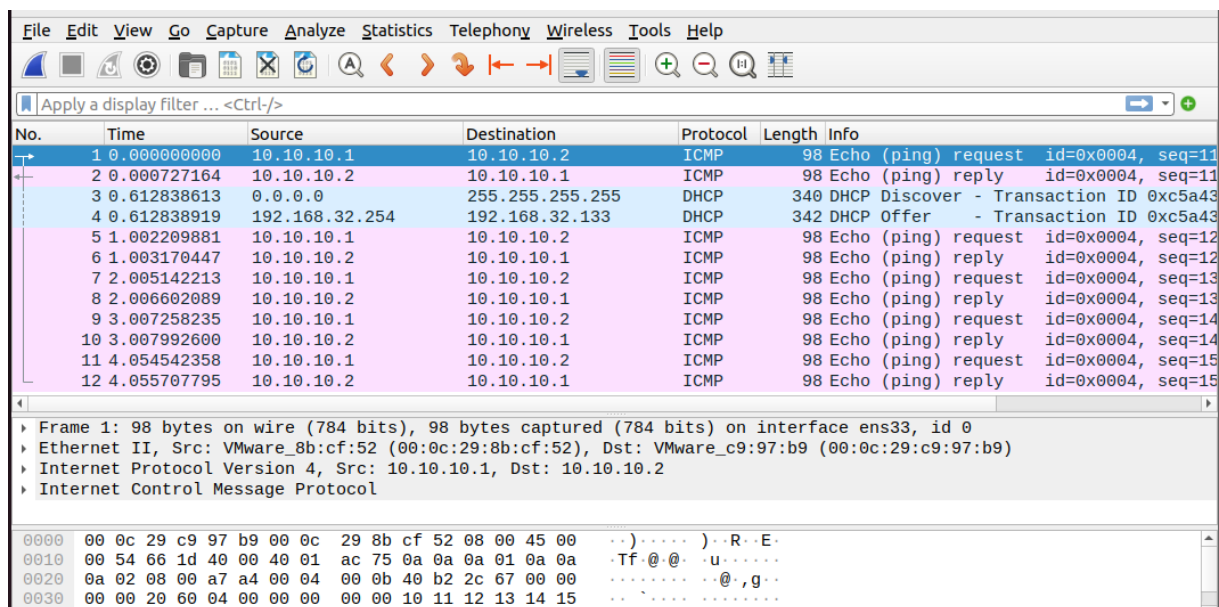
- ✚ Les tests de connectivité avec le tunnel VXLAN ont montré que les deux machines virtuelles (client et serveur) communiquent correctement.

```
root@yasmina-client:~# ping 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=1.40 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=0.905 ms
64 bytes from 10.10.10.1: icmp_seq=3 ttl=64 time=1.04 ms
64 bytes from 10.10.10.1: icmp_seq=4 ttl=64 time=1.23 ms

root@yasmina-vxlan-plateforme:~# ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=4.45 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.868 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=0.827 ms
64 bytes from 10.10.10.2: icmp_seq=4 ttl=64 time=1.18 ms
```

Figure 5.6 : Test de connectivité entre les machines virtuelles via le tunnel VXLAN

- ✚ En utilisant Wireshark, nous avons capturé et observé le trafic réseau, confirmant que les paquets traversent bien le tunnel VXLAN établi.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.10.1	10.10.10.2	ICMP	98	Echo (ping) request id=0x0004, seq=11
2	0.000727164	10.10.10.2	10.10.10.1	ICMP	98	Echo (ping) reply id=0x0004, seq=11
3	0.612838613	0.0.0.0	255.255.255.255	DHCP	340	DHCP Discover - Transaction ID 0xc5a43
4	0.612838919	192.168.32.254	192.168.32.133	DHCP	342	DHCP Offer - Transaction ID 0xc5a43
5	1.002209881	10.10.10.1	10.10.10.2	ICMP	98	Echo (ping) request id=0x0004, seq=12
6	1.003170447	10.10.10.2	10.10.10.1	ICMP	98	Echo (ping) reply id=0x0004, seq=12
7	2.005142213	10.10.10.1	10.10.10.2	ICMP	98	Echo (ping) request id=0x0004, seq=13
8	2.006602089	10.10.10.2	10.10.10.1	ICMP	98	Echo (ping) reply id=0x0004, seq=13
9	3.007258235	10.10.10.1	10.10.10.2	ICMP	98	Echo (ping) request id=0x0004, seq=14
10	3.007992600	10.10.10.2	10.10.10.1	ICMP	98	Echo (ping) reply id=0x0004, seq=14
11	4.054542358	10.10.10.1	10.10.10.2	ICMP	98	Echo (ping) request id=0x0004, seq=15
12	4.055707795	10.10.10.2	10.10.10.1	ICMP	98	Echo (ping) reply id=0x0004, seq=15

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface ens33, id 0  
Ethernet II, Src: VMware\_8b:cf:52 (00:0c:29:8b:cf:52), Dst: VMware\_c9:97:b9 (00:0c:29:c9:97:b9)  
Internet Protocol Version 4, Src: 10.10.10.1, Dst: 10.10.10.2  
Internet Control Message Protocol

0000 00 0c 29 c9 97 b9 00 0c 29 8b cf 52 08 00 45 00 ...R..E.  
0010 00 54 66 1d 40 00 40 01 ac 75 0a 0a 0a 01 0a 0a ..Tf.@@.u.....  
0020 0a 02 08 00 a7 a4 00 04 00 0b 40 b2 2c 67 00 00 .....@.,g..  
0030 00 00 20 60 04 00 00 00 00 00 10 11 12 13 14 15 ..'.....

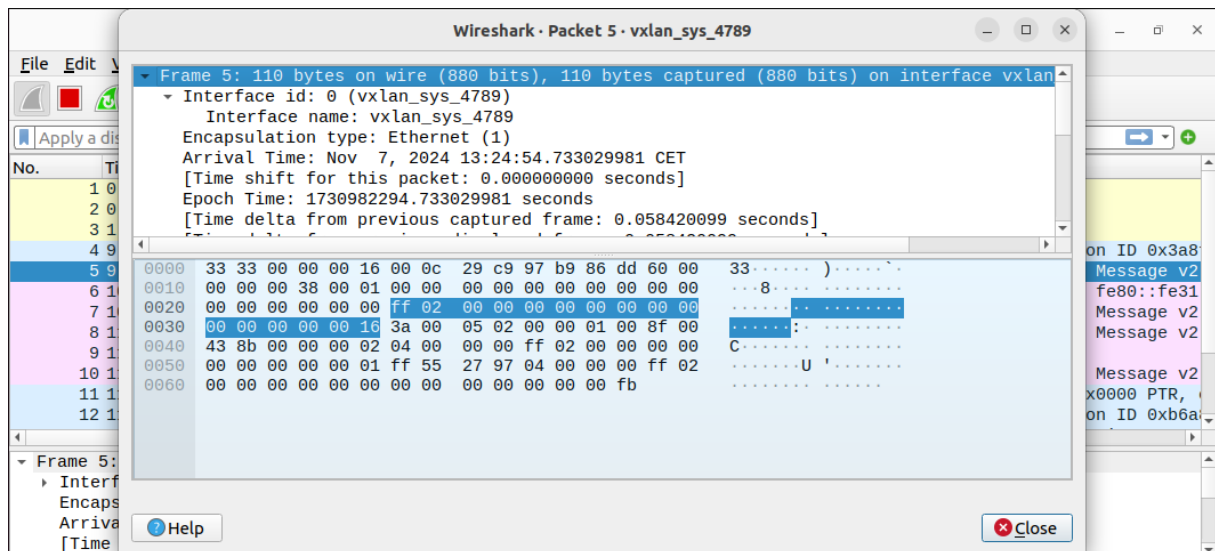


Figure 5.7 : Capture du trafic réseau avec Wireshark pour vérifier le passage des paquets à travers le tunnel VXLAN

- Ensuite, nous avons testé l'accès à la plateforme déployée sur la machine serveur (adresse IP : 10.10.10.1) depuis la machine cliente (adresse IP : 10.10.10.2). En entrant l'adresse IP du serveur suivie du port dans le navigateur du client, nous avons pu visualiser l'interface de la plateforme en temps réel. Cela confirme que le tunnel VXLAN permet une communication sécurisée et transparente entre les deux machines. Cependant, avant de pouvoir visualiser les données, le médecin doit d'abord s'authentifier avec l'utilisateur yacine.



Figure 5.8 : Accès à distance à la plateforme via l'authentification de l'utilisateur yacine

## Visualisation des données après authentification

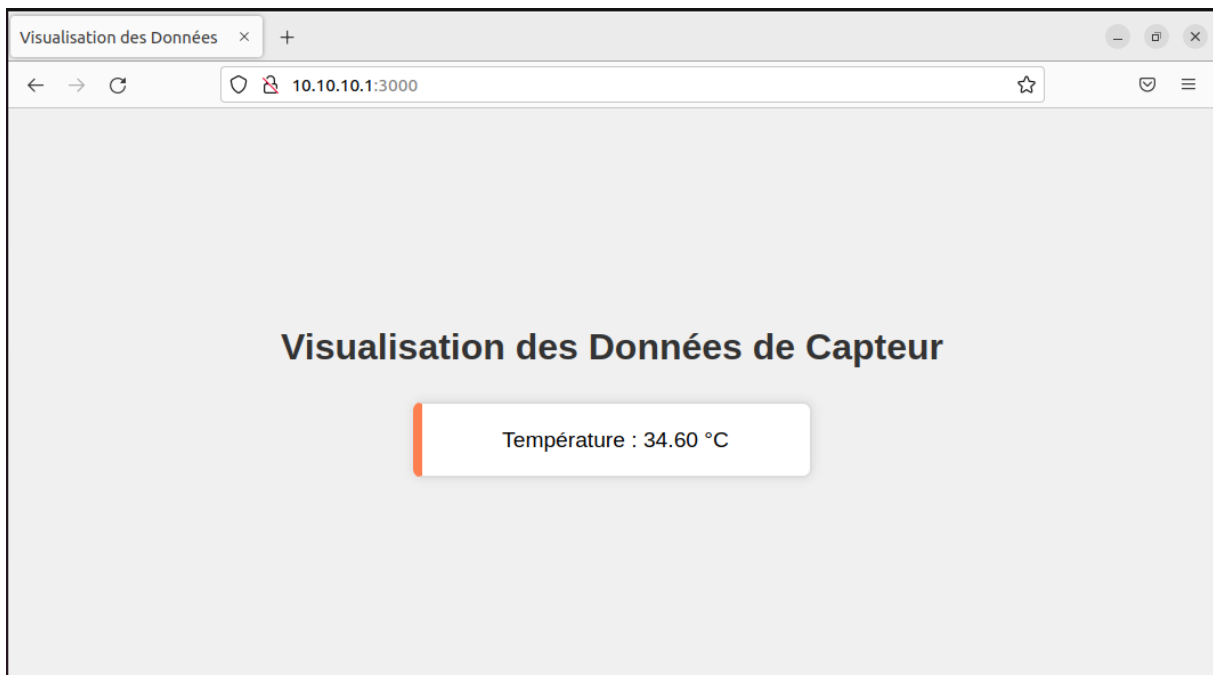



Figure 5.9 : Test de l'accès à la plateforme via le tunnel VXLAN entre la machine client et la machine serveur

 Grâce au tunnel VXLAN configuré entre les deux machines, il est possible de visualiser à distance les changements de température mesurés par le capteur. Lorsqu'une nouvelle valeur de température est envoyée par l'ESP32 et reçue par le serveur MQTT, elle est immédiatement affichée sur la plateforme Node.js. La machine cliente, connectée via le tunnel VXLAN, peut accéder à cette plateforme en temps réel et observer les variations de température, comme si elle était sur le même réseau local.

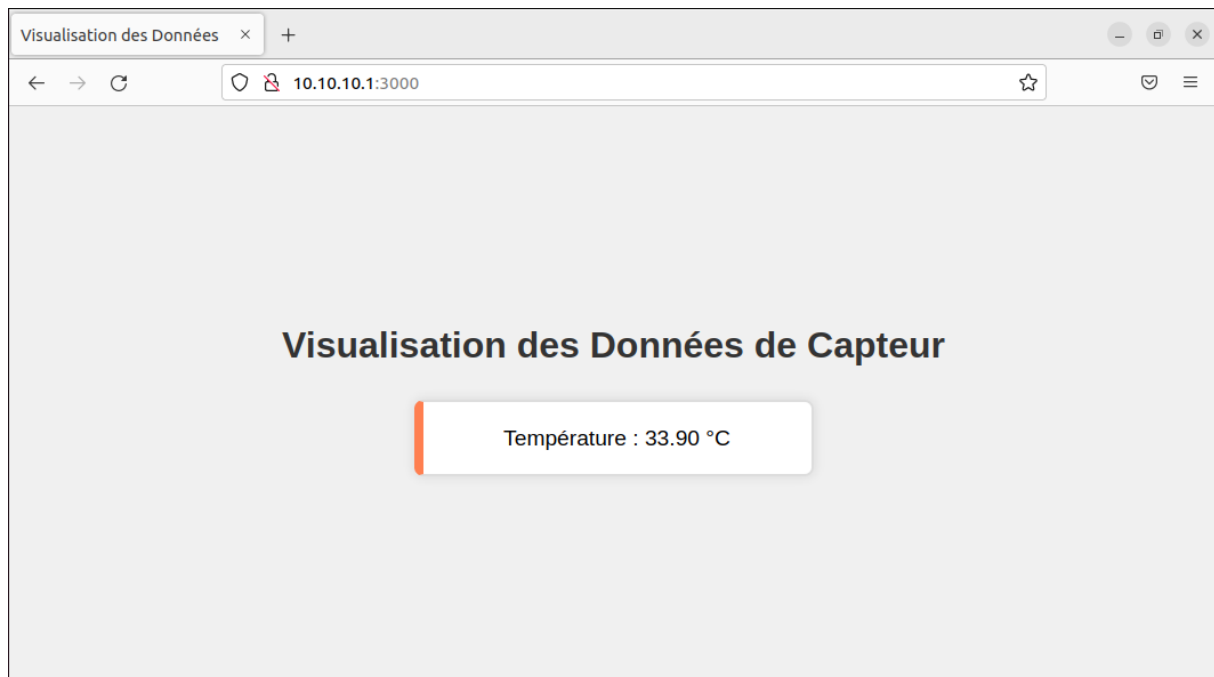


Figure 5.10 : Visualisation à distance des variations de température en temps réel via le tunnel VXLAN

## 5.4 Vérification des règles de gestion sur SDN

✚ Après application des règles via le contrôleur SDN, la communication entre le client et le serveur est bloquée.

```
root@yasmina-vxlan-plateforme:~# ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
From 10.10.10.1 icmp_seq=10 Destination Host Unreachable
From 10.10.10.1 icmp_seq=11 Destination Host Unreachable
From 10.10.10.1 icmp_seq=12 Destination Host Unreachable
From 10.10.10.1 icmp_seq=13 Destination Host Unreachable
From 10.10.10.1 icmp_seq=14 Destination Host Unreachable
From 10.10.10.1 icmp_seq=15 Destination Host Unreachable
^C
```

```
root@yasmina-client:~# ping 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
From 10.10.10.2 icmp_seq=1 Destination Host Unreachable
From 10.10.10.2 icmp_seq=2 Destination Host Unreachable
From 10.10.10.2 icmp_seq=3 Destination Host Unreachable
From 10.10.10.2 icmp_seq=4 Destination Host Unreachable
From 10.10.10.2 icmp_seq=5 Destination Host Unreachable
From 10.10.10.2 icmp_seq=6 Destination Host Unreachable
```

Figure 5.11 : Blocage de la communication entre le client et le serveur après application des règles via le contrôleur SDN

Et la on voit que la machine cliente ne pourra plus accéder à la plateforme de visualisation à distance.

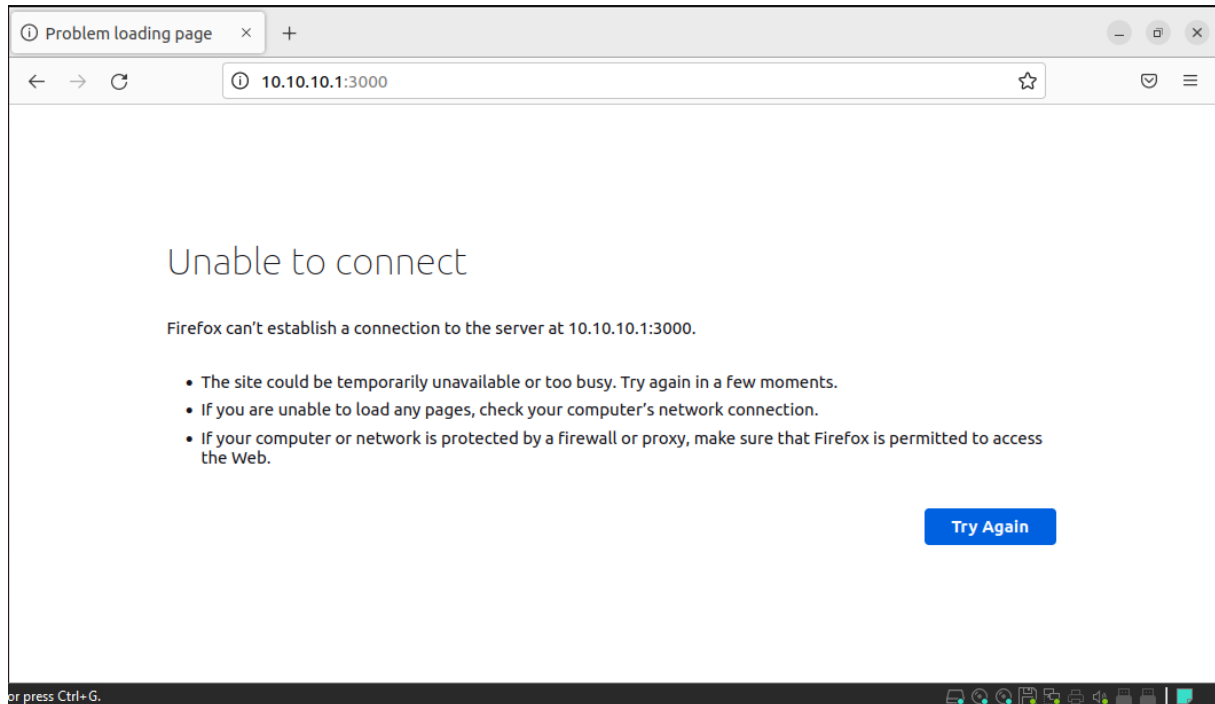


Figure 5.12 : Impossibilité d'accès à la plateforme de visualisation à distance depuis la machine cliente après application des règles SDN

## 5.5. Conclusion

Les tests réalisés ont permis de valider l'efficacité de la solution de télésurveillance médicale. La transmission des données capteurs via le serveur MQTT a été confirmée comme fluide et stable. La plateforme de visualisation a bien fonctionné, permettant une mise à jour en temps réel des données et leur stockage sécurisé dans la base de données MySQL. En ce qui concerne la connectivité, le tunnel VXLAN a prouvé son efficacité en permettant une communication sécurisée entre les machines client et serveur, et ce, même à distance. Enfin, les règles de gestion du réseau via le contrôleur SDN ont bien été appliquées, garantissant un contrôle optimal du trafic réseau.

## **Conclusion Generale**

En conclusion, ce mémoire explore la mise en place d'une plateforme innovante combinant les technologies SDN et VXLAN dans le domaine médical, afin d'offrir une solution performante pour la gestion et la visualisation des données de santé. Grâce à l'intégration de capteurs IoT, de protocoles de réseau avancés et d'une architecture de gestion centralisée, cette solution vise à optimiser la communication entre les différents acteurs médicaux, qu'ils soient localisés sur le même site ou à distance.

En offrant une infrastructure évolutive, sécurisée et centralisée, ce projet permet non seulement de renforcer la qualité des soins en facilitant l'accès aux données médicales essentielles, mais également de répondre aux exigences strictes de confidentialité et de gestion du trafic dans un environnement critique.

Les défis techniques et les meilleures pratiques abordées dans ce travail constituent une base solide pour l'adoption de technologies avancées dans le secteur de la santé, contribuant ainsi à l'évolution des soins vers une approche plus connectée et réactive.

## Bibliographie

- [1] [https://hepia.infolibre.ch/Virtualisation-Reseaux/Le\\_SDN\\_pour\\_Les\\_Nuls\\_Jerome\\_Durand\\_JRES\\_2015.pdf](https://hepia.infolibre.ch/Virtualisation-Reseaux/Le_SDN_pour_Les_Nuls_Jerome_Durand_JRES_2015.pdf) (consulté le 01/07/2024 à 13H34)
- [2] Mr Ndiaye Wade, "Etude de système Edge computing basée sur les réseau SDN", Mémoire de master 2 Université Cheikh Anta Diop De Dakar, Faculté des Science et Technique, Département Informatique, 2019.
- [3] <https://www.theses.fr/2019REIMS011.pdf> (consulté le 24/06/2024 a 15h17)
- [4] <https://www.sap.com/france/products/artificial-intelligence/what-is-iot.html> (consulté le 02/07/2024 a 16h12)
- [5] Mr Arona Seye, « Etude et mise en place d'une Plateforme intégrant du Cloud Computing », Mémoire licence ESTM, Télécommunications et Réseaux, Département Téléinformatique, 2022.
- [6] Mr Kéba Gueye « Etude de la convergence des reseaux LTE-A/IoT et ses applications en e-sante, e-formation et e-agriculture » Thèse Doctorat Ecole Supérieur Politechnique, Télécommunication, Département Téléinformatique, 2020.
- [7] <https://www.juniper.net/fr/fr/research-topics/what-is-vxlan.html#:~:text=D'un%20point%20de%20vue,'un%20en%20Dt%C3%AAte%20VXLAN> (consulté le 12/07/2024 / a 15H20)
- [9] <https://aws.amazon.com/fr/what-is/mqtt/#:~:text=L'agent%20MQTT%20est%20le,des%20messages%20%C3%A0%20ces%20derniers.> (consulté le 31/10/2024 a 15H45)



