

Grandmont Network Monitoring

Reprise d'un projet :

Non

Partenaire :

Étudiants chargés du projet :

Étudiant 1 : Serhat TURAN
Étudiant 2 : Matt BESCHON
Étudiant 3 : Léo GAUTRON

Professeurs ou tuteurs responsables:

Pierre CLAVEL
Florent CHRÉTIEN

Rapport de projet : Network Monitor

Sommaire :

1. Partie Commune du projet

1.1 Expression du besoin

1.2 Contraintes de développement (choix langages, librairies, protocoles)

1.3 Analyse UML

1.4 Analyse et conception de la base de données

1.5 Répartition des tâches

1.6 Plannings prévisionnel et réel (Diagramme Gantt)

1.7 Recette globale

1.8 Partie Individuelle Gautron Léo(tech3)

1.9 Partie Individuelle Turhan serhat (tech1)

1.10 Partie Individuelle Beschon Matt (tech2)

1.11 conclusion du projet

1. Partie commune :

1.1 Expression du besoin :

Problématique

Ces derniers mois, plusieurs dysfonctionnements ont été constatés sur le réseau informatique du lycée et sur l'accès à Internet. Ces difficultés ont largement impacté le travail des étudiants et des élèves.

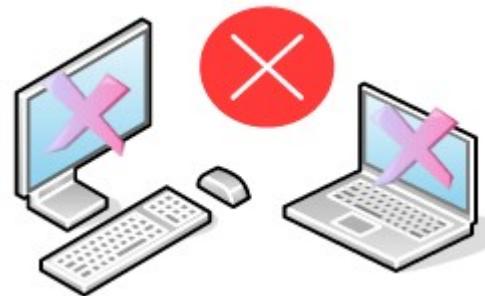
De manière répétitive, parfois quotidienne, il a fallu du temps pour :

- ✓ constater le problème,
- ✓ trouver son origine,
- ✓ y remédier.

Des heures de travail ont été perdues et un peu frustration générée, pour les enseignants et les élèves.

Parmi les problèmes constatés :

- Interruption de l'accès complet à Internet.
- Dysfonctionnement du service DNS.
- Pollution massive du réseau et baisse des performances (débit).
- Problème d'attribution des adresses IP (service DHCP).
- Dysfonctionnement du pare-feu de la section.
- Perte de l'alimentation d'un switch.



Solution technique envisagée et objectifs

L'équipe enseignante du lycée a proposé l'idée d'un système technique capable de surveiller l'état et la disponibilité du réseau informatique, de détecter rapidement un problème, puis d'avertir aussitôt les utilisateurs, et enfin de fournir des détails techniques sur le dysfonctionnement signalé.

Le projet consiste donc à réaliser le développement de cette solution de ce système de surveillance.

Le système sera installé sur le réseau de la section STS SNIR. Ce réseau est « hors-domaine », c'est-à-dire qu'il est isolé du reste du réseau de l'établissement, car il possède son propre pare-feu.

De fait, les problèmes affichés pourront rapidement être résolus localement, ou bien signalés par les correspondants locaux à l'organisme de gestion (le GIP RECIA).

Remarque : même si le système sera installé sur le réseau de la

section SNIR, il sera capable de détecter parfois des problèmes qui impactent le réseau de tout le lycée.

1.2 Les contraintes de développement :

Nous maîtrisons mieux le langage C++, ce qui nous permet de développer des solutions plus efficaces et optimisées pour notre projet. PHP, de son côté, bénéficie d'une grande communauté de développeurs, ce qui facilite l'accès à des ressources, des bibliothèques et des forums d'entraide, rendant le développement web plus rapide et moins coûteux en temps.

Le système d'exploitation Raspberry Pi OS est particulièrement bien adapté à notre projet en raison de sa faible consommation d'énergie. Cette caractéristique est cruciale pour des applications nécessitant une utilisation continue tout en maintenant des coûts énergétiques réduits.

PfSense est spécifiquement indiqué dans le cahier des charges en tant que solution de pare-feu et de routeur. Sa robustesse et sa flexibilité en font un choix idéal pour gérer la sécurité et le flux de données dans notre infrastructure réseau.

Enfin, les colonnes de signalisation sont également mentionnées dans le cahier des charges. Elles jouent un rôle essentiel dans notre projet, assurant une communication claire et visuelle des états ou des alertes, ce qui est crucial pour le bon fonctionnement et la sécurité de notre système.

Libssh est facile à utiliser et nous l'avons déjà employée dans des projets précédents, ce qui en facilite l'intégration dans notre environnement actuel. De même, libMysql est une bibliothèque que

nous avons déjà utilisée, et sa facilité d'utilisation nous permet de gérer efficacement les interactions avec la base de données.

Libsftp, quant à elle, bénéficie d'une grande communauté de développeurs. Cela garantit un accès facile à des ressources, des exemples de code et du support, rendant son utilisation tout aussi simple et efficace pour nos besoins de transfert de fichiers sécurisés.

1.3 Analyse UML

Diagramme des cas d'utilisation :

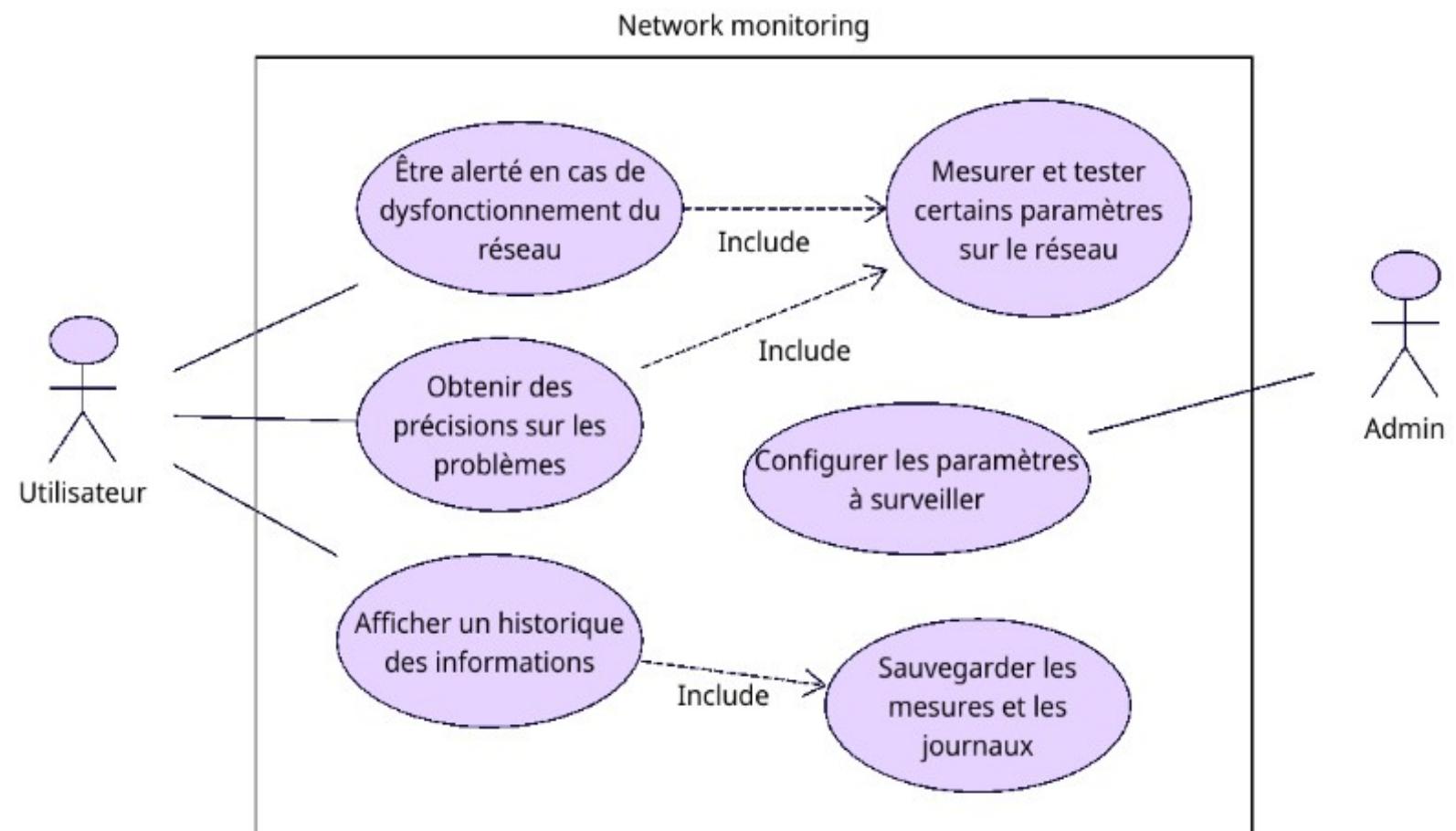
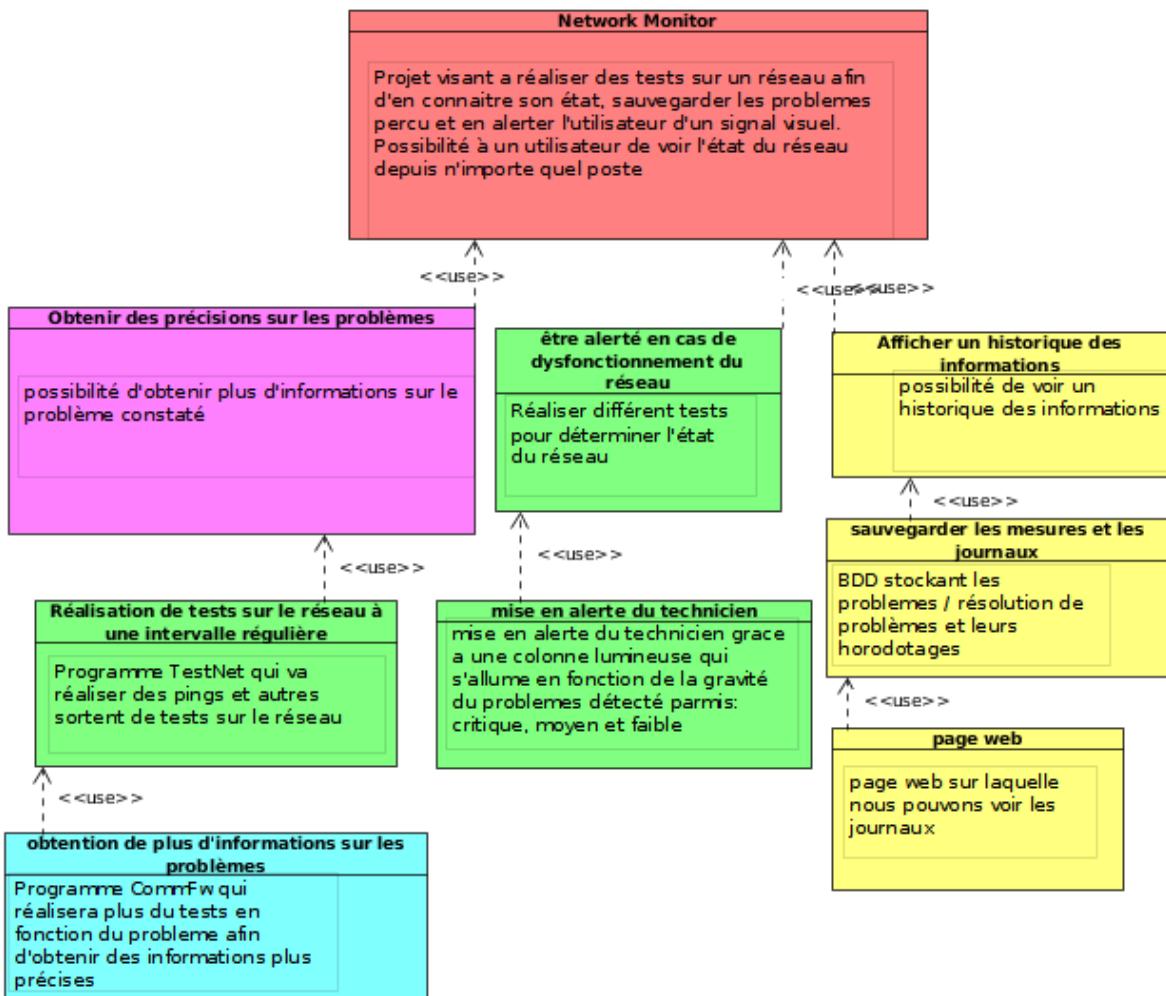


Diagramme d'exigences :



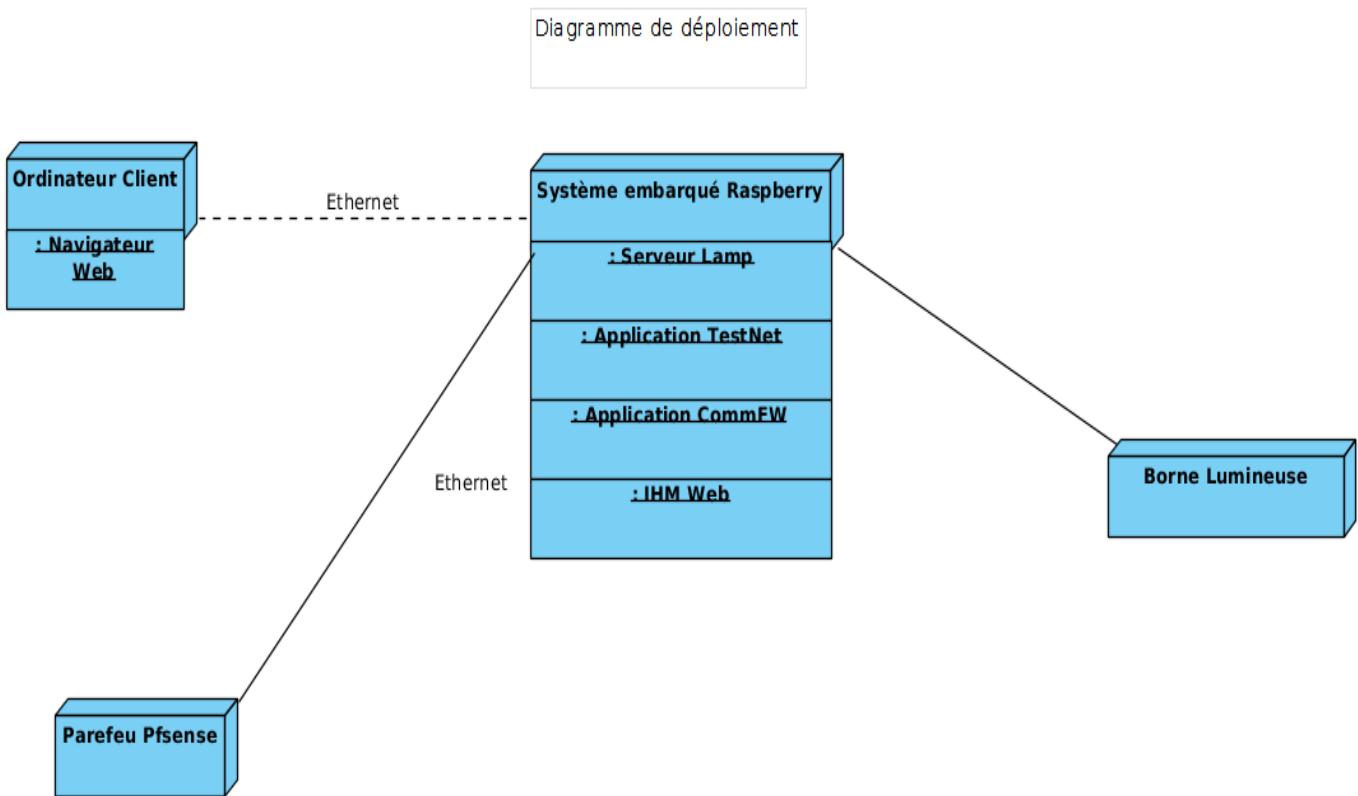
Légende

Serhat

Matt

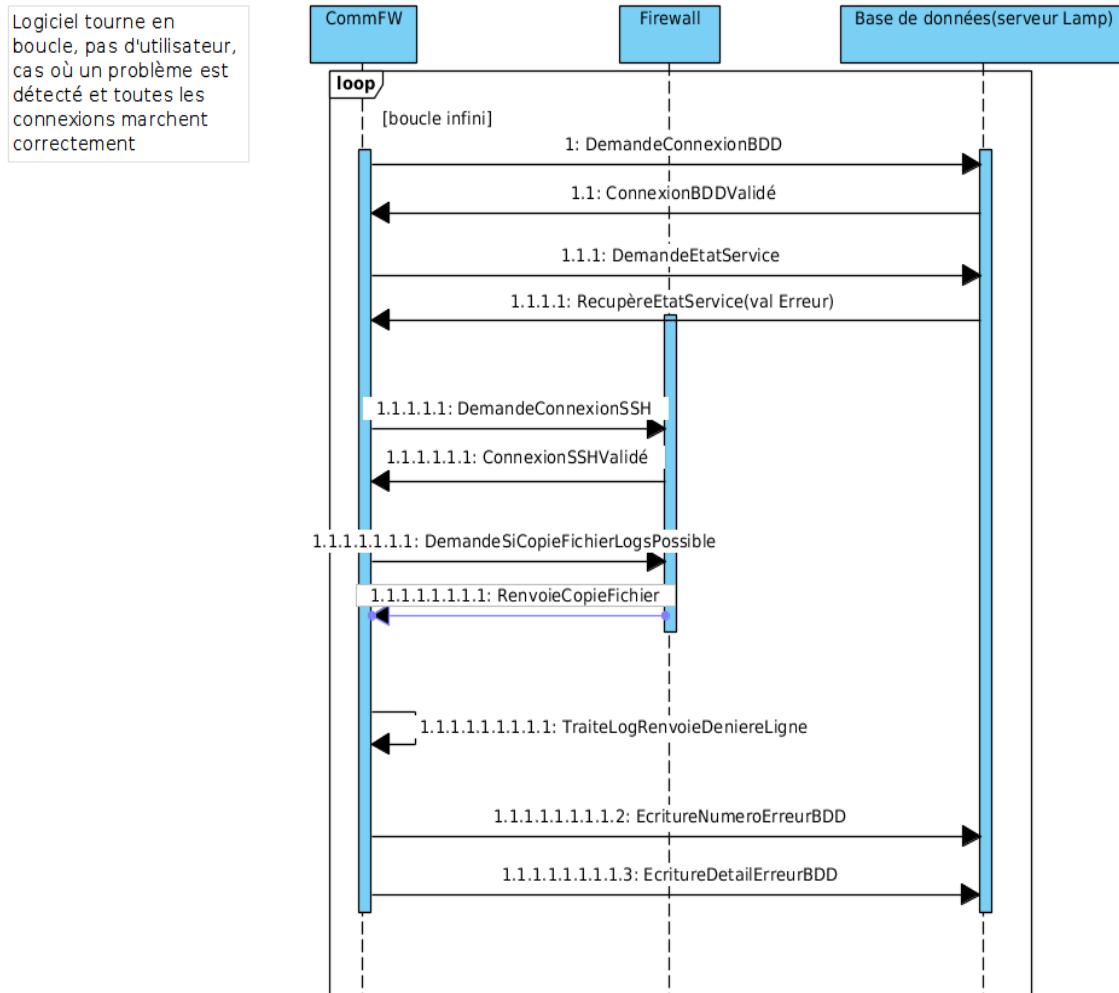
Léo

Diagramme de déploiement :

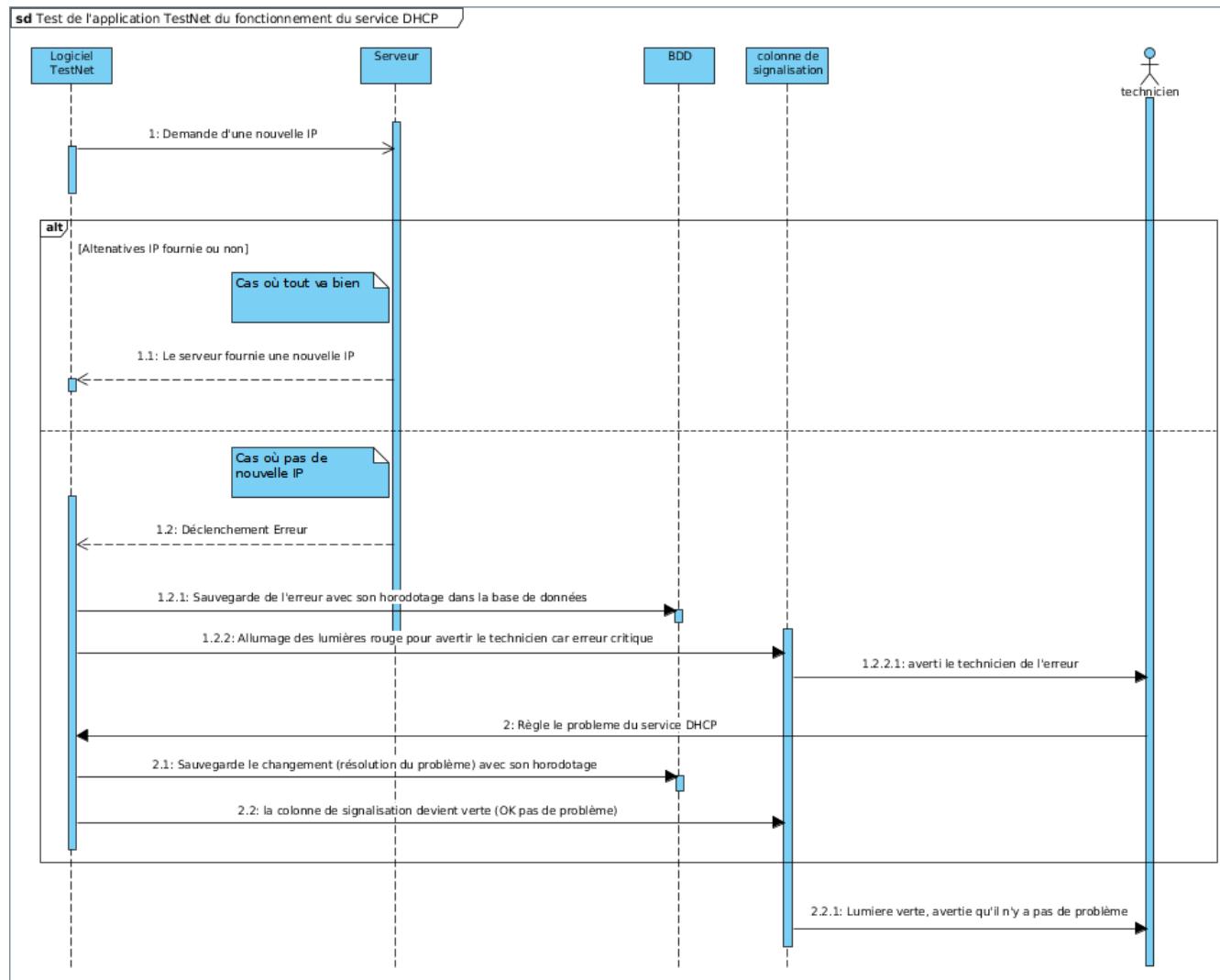


Diagrammes de séquence : Technicien 3 : Partie CommFW :

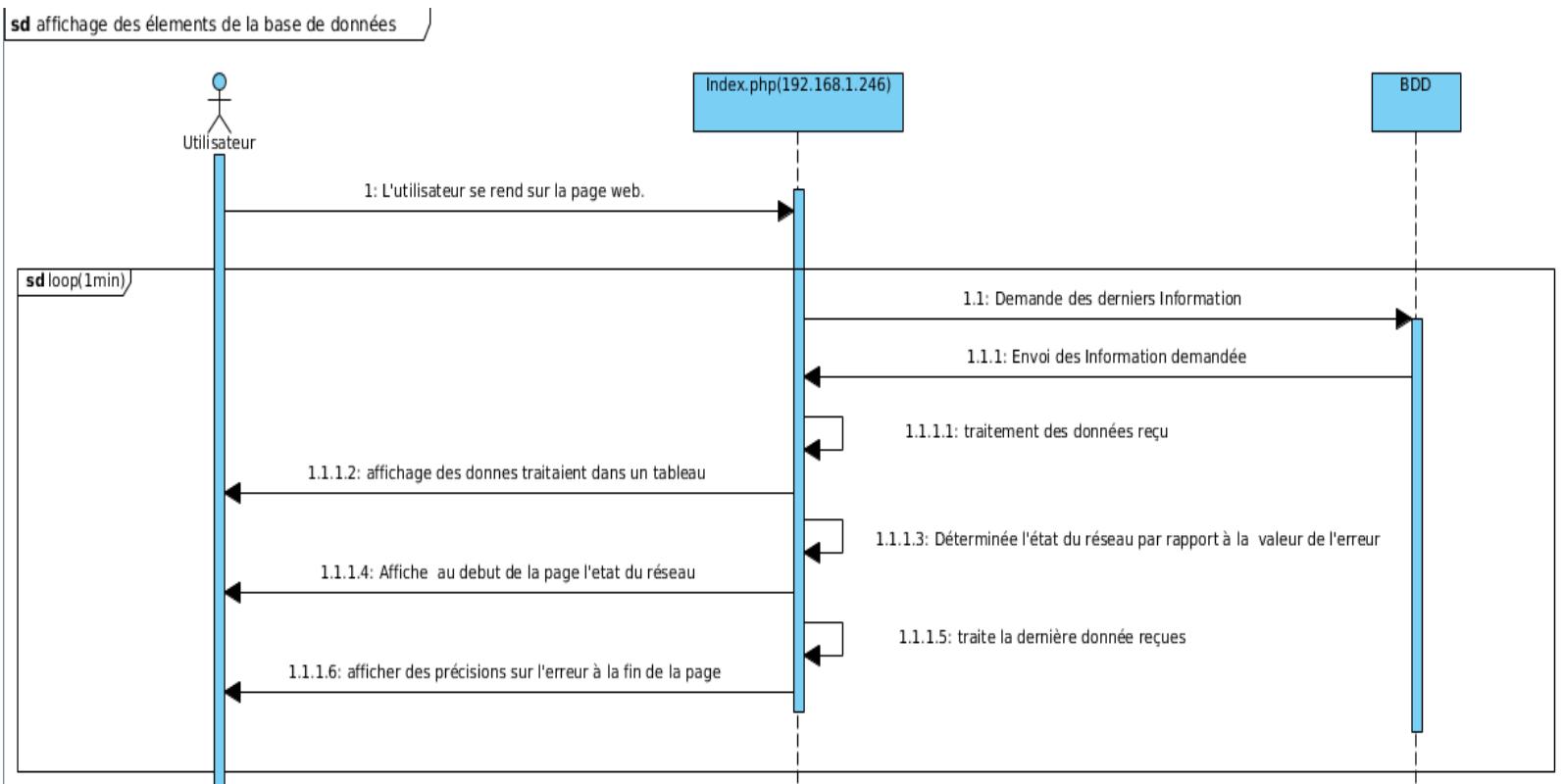
sd Diagramme de séquence : Communication entre logiciel, Pare-Feu et base de données]



Technicien 1 : Partie TestNet:



Technicien 2 : Partie Base de données :



1.4 Modèle Conceptuel des données :

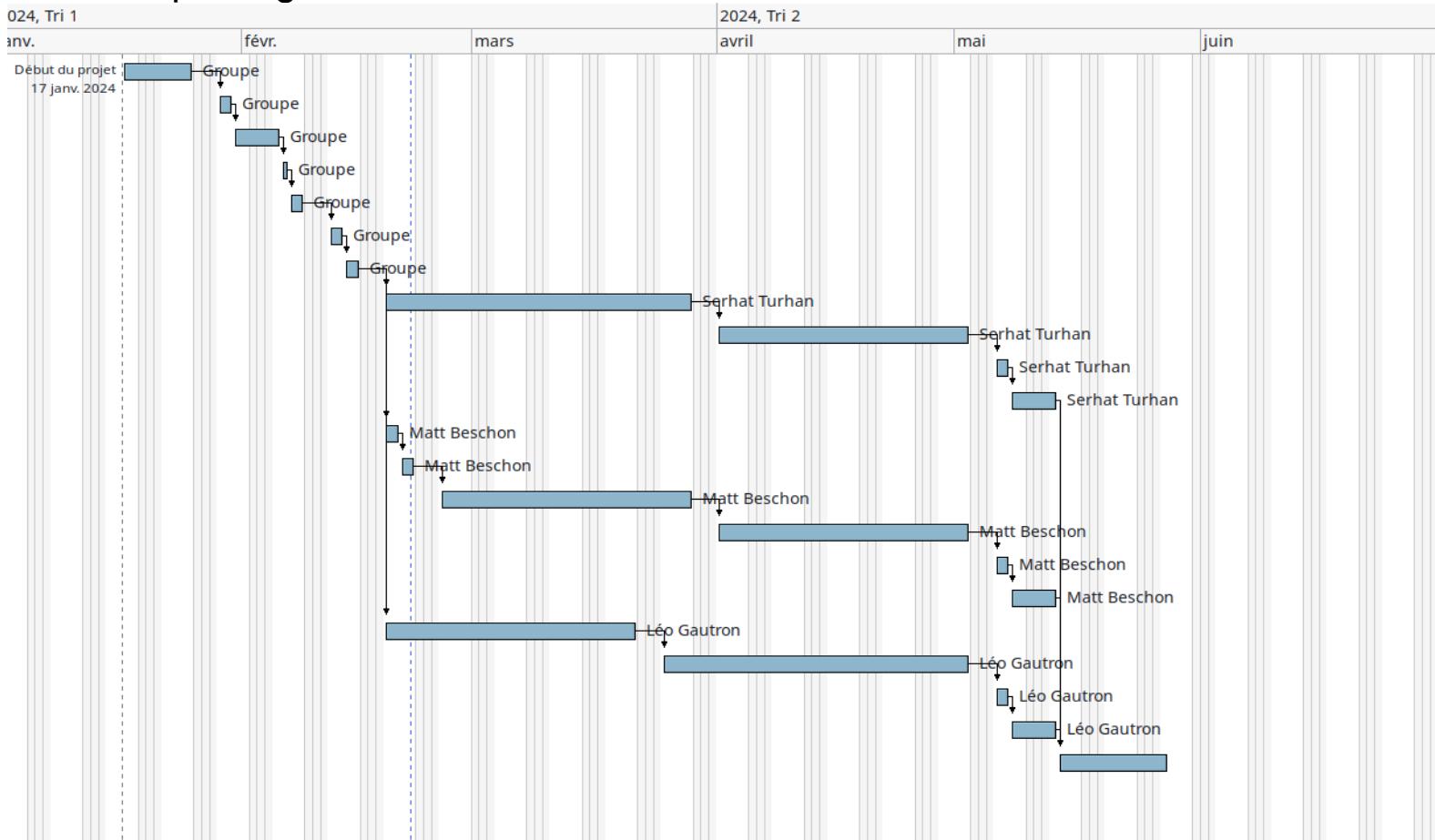
Log
Date_Hheure
Val_Erreur
Description_Erreur

Numero_Erreur
Date_Hheure
Val_Erreur
Description_Erreur

1.5 Répartition des tâches

TPÉ	Nom	Travail
1	Faire l'étude de l'application proposée sous forme d'analyse UML. (Groupe)	6j
2	Compléter et valider l'échéancier. (Groupe)	2j
3	Établir la modélisation des modèles conceptuels de données. (Groupe)	3j
4	Définir les choix de technologies utilisées pour mettre en œuvre les différents éléments du système, et plus particulièrement les protocoles d'échanges de données ainsi que l'IHM. (Groupe)	1j
5	Dans une table, définir toutes les données qui seront enregistrées ou échangées : type, précision, plage de valeur, unité... (Groupe)	2j
6	Définir les seuils d'alerte. (Groupe)	2j
7	Définir les protocoles d'échanges de données entre les applications (choix du protocole, noms de topic, valeurs). (Groupe)	2j
8	Création et mise au point de l'application TestNet. (Technicien 1)	24j
9	Mettre en œuvre le système de signalisation visuel. (Technicien 1)	20j
10	Produire la documentation qui permettra à un utilisateur d'installer, de configurer et d'utiliser l'ensemble des outils (Technicien 1)	2j
11	Rédiger les documents de recette (Technicien 1)	3j
12	Installation de l'OS de l'ordinateur embarqué. (Technicien 2)	2j
13	Installation du serveur Lamp. (Technicien 2)	2j
14	Modélisation de la base de données. (Technicien 2)	20j
15	Création et mise au point de l'application IHM. (Technicien 2)	20j
16	Produire la documentation qui permettra à un utilisateur d'installer, de configurer et d'utiliser l'ensemble des outils (Technicien 2)	2j
17	Rédiger les documents de recette (Technicien 2)	3j
18	Installation et configuration de l'OS Pare-feu. (Technicien 3)	20j
19	Création et mise au point de l'application CommFW. (Technicien 3)	24j
20	Produire la documentation qui permettra à un utilisateur d'installer, de configurer et d'utiliser l'ensemble des outils (Technicien 3)	2j
21	Rédiger les documents de recette (Technicien 3)	3j
22	Produire la documentation destinée à l'utilisateur final. (Groupe)	8j

1.6 planing Réel:



1.7 La Recette globale :

Fiche de recette gloable

<input checked="" type="checkbox"/> Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale <input type="checkbox"/> Test de faisabilité <input type="checkbox"/> Test unitaire <input type="checkbox"/> Test d'intégration			
Nom de l'étudiant : Beschon Matt			
Date : 24/05/24			
objectif(s) de la recette			
ID	Action/Méthode	Comportement / Résultat attendu	Status OK / NOK
1	Création et mise au point de l'application TestNet.	L'application effectue bien tous les tests qu'on lui demande	ok
2	Mettre en œuvre le système de signalisation visuel.	Vert:ok orange:faible rouge:moyen rouge+son:critique	ok
3	Installation de l'OS de l'ordinateur embarqué	L'ordinateur embarqué est fonctionnel et le ssh est l'I2c et bien activés	ok
4	Installation du serveur Lamp.	Le serveur Lamp est bien installé sur l'ordinateur embarqué	ok
5	Modélisation de la base de données.	La Base de données est bien modélisé (1 base de données Network Monitor 2 tables Log et Numero_error chaque table possède 3 champs Date Heure, Val Erreur et Description Erreur	ok
6	Création et mise au point de l'application IHM.	L'Ihm affiche bien l'état du réseau actuel, elle affiche bien les 10 derniers logs reçus dans la table Log et elle affiche aussi des détails sur la dernière erreur reçue	ok
7	Installation et configuration de l'OS Pare-feu.	Le pare-feu est fonctionnel, l'accès au logs est possible, la connexion ssh est activé	ok
8	Création et mise au point de l'application CommFW.	L'application récupère le numéro de l'erreur et envoie bien à la base de données, le numéro de l'erreur, la description tiré des fichiers logs.	ok

1.8Partie individuelle : Léo Gautron

Introduction :

L'année dernière, le réseau du lycée a subi des perturbations majeures, compromettant la continuité des services. Pour remédier à cette situation, nous avons entrepris un projet crucial visant à renforcer notre infrastructure réseau. Mon rôle dans ce projet a été déterminant, englobant plusieurs tâches essentielles :

- **Installation et configuration de l'OS Pare-feu** : Mise en place d'un système d'exploitation dédié à la gestion du pare-feu pour assurer une protection optimale du réseau.
- **Création et mise au point de l'application CommFW** : Développement d'une application spécialisée en C++ pour superviser et analyser les fichiers logs du réseau, permettant une remontée rapide des problèmes.
- **Compléter les caractéristiques et les performances de l'application** : Amélioration continue de l'application pour garantir sa robustesse et son efficacité.
- **Valider le fonctionnement envisagé, par des tests de faisabilité** : Réalisation de tests rigoureux pour s'assurer que l'application répond aux besoins opérationnels.
- **Développer et tester l'application** : Cycle complet de développement et de tests pour assurer la fiabilité et la performance de CommFW.
- **Produire la documentation** : Création de guides détaillés pour l'installation, la configuration et l'utilisation des outils, destinés aux utilisateurs finaux.

- **Rédiger les documents de recette** : Élaboration des documents de validation pour garantir que toutes les spécifications et exigences ont été respectées.

Ces étapes ont été cruciales pour renforcer notre infrastructure et prévenir toute perturbation future, assurant ainsi l'efficacité de notre réseau.

Solutions multiples et envisagées :

Pour ce qui est de la programmation, nous avions envisagé d'utiliser plusieurs langages de programmation tel que le C, C++ ou Python. Néanmoins notre choix s'est porté sur le langage de programmation C++ car nous étions beaucoup plus familier avec celui-ci et nous avions un bagage plus conséquent concernant les librairies et leur utilisations.

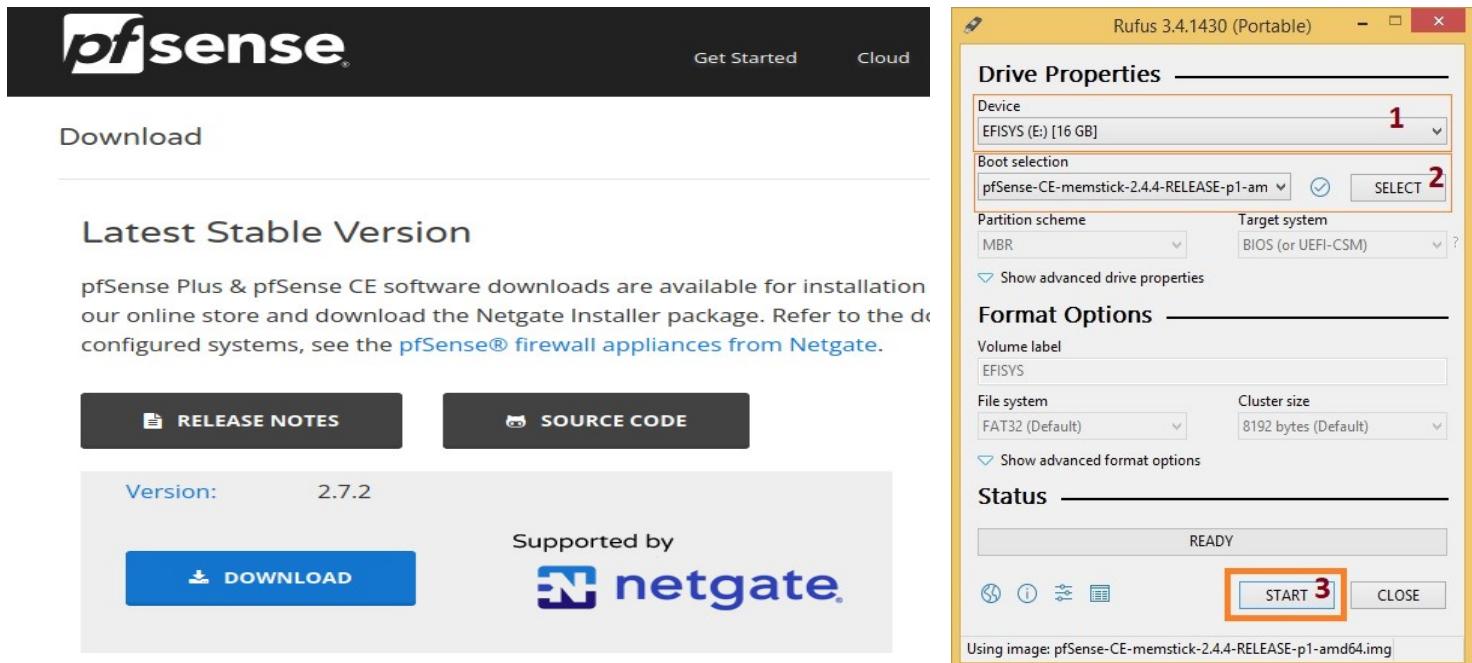
Pour ce qui est de la partie pare-feu, nous avions envisagé d'utiliser un pare-feu netgate. Finalement notre choix s'est porté sur un PC avec la distribution PfSense. L'ordinateur était disponible ce qui a fait de cette solution une option pratique et économique pour assurer la sécurité du réseau. Cela nous a évité de racheter un pare-feu netgate.

Conception , configuration, réalisation :

Partie 1 : Pfsense

Tout d'abord, avant de pouvoir utiliser Pfsense sur notre ordinateur, plusieurs paramètres sont à prendre en compte :

-Il faut tout d'abord commencer par installer l'iso Pfsense sur une clé USB et formater l'iso avec rufus.



-Par la suite, il faut brancher la clé USB avec l'iso sur le PC qui nous servira de pare-feu et appuyer sur la touche pour boot en l'occurrence F2.

-Le PC ne disposant que d'une seule carte réseau, il est nécessaire d'avoir une deuxième carte réseau afin d'avoir deux sorties (WAN et LAN)

-Le PC disposant maintenant de deux cartes réseau (sortie re0 et rl0), il faut attribuer une sortie pour le WAN et une autre pour le LAN, en l'occurrence, le LAN a pour sortie rl0 et le WAN a pour sortie re0.

-Pour savoir quel IP nous allons attribuer côté LAN. Nous avons choisi au préalable de créer notre réseau sous l'IP suivante : 192.168.1.0/24. Donc nous allons identifier le côté LAN du Pfsense sous l'adresse IP suivante : 192.168.1.254.

-Afin d'attribuer une IP au WAN, il faut pouvoir accéder au parefeu du dessus afin de configurer une adresse IP en fonction de l'adresse MAC du WAN. (MAC WAN : 20:cf:30:f0:8c:7a, IP WAN : 172.16.150.10/18).

Si tout est bien fonctionnel, nous aurons une page avec bien nos deux interfaces (re0 et rl0 avec l'adresse IP de chacun)

```
pfSense - Serial: MT700A009508661 - Netgate Device ID: 6e2a605f512194831618
*** Welcome to pfSense 2.7.2-RELEASE (amd64) on pfSense ***
WAN (wan)      -> re0      -> v4/DHCP4: 172.16.150.10/18
LAN (lan)      -> rl0      -> v4: 192.168.1.254/24

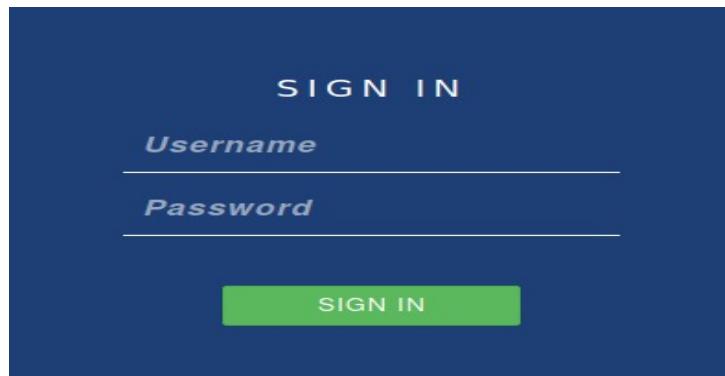
0) Logout (SSH only)          9) pfTop
1) Assign Interfaces          10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults   13) Update from console
5) Reboot system               14) Disable Secure Shell (sshd)
6) Halt system                 15) Restore recent configuration
7) Ping host                   16) Restart PHP-FPM
8) Shell

Enter an option: █
```

-Par la suite, il faut pouvoir accéder à l'interface graphique du pfsense, c'est à dire ouvrir un navigateur depuis un PC dans le local et renseigner dans l'url, l'adresse IP du parefeu local (192.168.1.254)

192.168.1.254

-Une fenêtre s'ouvre afin de rentrer l'identifiant et le mot de passe choisi au préalable lors de l'installation de pfSense en l'occurrence : (identifiant : admin / mdp : admin7557)



-Il est nécessaire d'avoir un IP fixe pour notre serveur LAMP qui a matérialisé par notre raspberry, on va donc dans la partie serveur DHCP, mettre l'adresse MAC de la raspberry en IP fixe, cette adresse

DHCP Static Mappings				
Static ARP	MAC address	IP address	Hostname	Description
	e4:5f:01:87:59:54	192.168.1.246		
				Add Static Mapping

IP doit être en dehors de la plage défini au préalable par le serveur DHCP. (MAC adresse Raspberry : e4:5f:01:87:59:54, IP Fixe Raspberry : 192.168.1.246)

-Il faut activer le service DHCP, afin de pouvoir attribuer à chaque nouvel appareil, une adresse IP, il est nécessaire d'entrer une plage

IP afin d'avoir une intervalle pour le DHCP. (Pool range: 192.168.1.10 à 192.168.1.99)

Primary Address Pool		
Subnet	192.168.1.0/24	
Subnet Range	192.168.1.1 - 192.168.1.254	
Address Pool Range	192.168.1.10	192.168.1.99
From	To	
The specified range for this pool must not be within the range configured on any other address pool for this interface.		
Additional Pools	+ Add Address Pool	
If additional pools of addresses are needed inside of this subnet outside the above range, they may be specified here.		

-Il faut ensuite, pour avoir accès à internet, renseigner le serveur DNS du pare-feu extérieur à notre réseau local dans la partie serveur DHCP (serveur DNS : 172.16.180.189).

Server Options	
WINS Servers	WINS Server 1
	WINS Server 2
DNS Servers	172.16.180.189
	DNS Server 2
	DNS Server 3
	DNS Server 4

-Afin d'avoir un accès au fichier log du pfsense nous allons devoir dans un premier temps, autoriser la connexion ssh au pfsense, c'est à dire activer le « Secure shell server », puis dans un second temps, autoriser la connexion du WAN vers le LAN via le port 22.

Secure Shell											
Secure Shell Server	<input checked="" type="checkbox"/> Enable Secure Shell										
SShd Key Only	Password or Public Key										
When set to <i>Public Key Only</i> , SSH access requires authorized keys and these keys must be configured for each user that has been granted secure shell access. If set to <i>Require Both Password and Public Key</i> , the SSH daemon requires both authorized keys and valid passwords to gain access. The default <i>Password or Public Key</i> setting allows either a valid password or a valid authorized key to login.											
Allow Agent Forwarding	<input type="checkbox"/> Enables ssh-agent forwarding support.										
SSH port	22										
Note: Leave this blank for the default of 22.											
Floating	WAN	LAN									
Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	1/936 KiB	*	*	*	LAN Address	443 80 22	*	*		Anti-Lockout Rule	⚙️

Après avoir suivi précisément toutes les étapes de la configuration, nous disposerons d'un pfSense avec un accès SSH pour exploiter les fichiers logs, d'un serveur DHCP fonctionnel pour attribuer une IP à chaque nouvel appareil, ainsi que d'une protection pour notre réseau local.

Maintenant que le pare-feu est en place, nous allons pouvoir récupérer les données du fichiers logs, et créer le code CommFW qui permettra de récupérer le numéro de l'erreur, et d'aller chercher dans les fichiers la ligne correspondant à l'erreur indiqué.

Dans un premier temps, nous allons créer un code afin de pouvoir se connecter au pare-feu.

Nous utiliserons alors les librairies en C++ « libssh/libssh.h » et « libssh/sftp.h » et les librairies classiques en C++ (string.h, cstdlib, iostream, string, vector)

Le code finit, la connexion en ssh possible, il fallait maintenant pouvoir copier les fichiers logs présents dans le pare-feu avec comme répertoire : /var/log

Il faut pouvoir récupérer 3 types de fichiers : le fichier dhcp.log, gateways.log et ntpd.log

Nous copierons les trois fichiers en local dans un répertoire nommé « log_firewall »

Voici le code permettant la copie et le transfert de ces fichiers :

```
// Répertoire avec les fichiers logs sur le pare-feu et où les copier en local
const char *remoteFiles[] = {"var/log/dhcpd.log", "var/log/ntp.log", "var/log/gateways.log"};
const char *localFiles[] = {"home/etu903/log_firewall/log_firewall_dhcp.txt", "home/etu903/log_firewall/log_firewall_ntp.txt", "home/etu903/log_firewall/log_firewall_gateways.txt"};

// Copier chaque fichier log vers son répertoire
for (int i = 0; i < 3; ++i) {
    // Ouvrir le fichier distant en lecture
    sftp_file remoteFileHandle = sftp_open(sftpSession, remoteFiles[i], O_RDONLY, 0);
    if (!remoteFileHandle) {
        cout << "Failed to open remote file: " << ssh_get_error(sshSession) << endl;
        sftp_free(sftpSession);
        ssh_disconnect(sshSession);
        ssh_free(sshSession);
        return 1;
    }

    // Ouvrir le fichier local en écriture
    FILE *file = fopen(localFiles[i], "wb");
    if (!file) {
        cout << "Failed to open local file." << endl;
        sftp_close(remoteFileHandle);
        sftp_free(sftpSession);
        ssh_disconnect(sshSession);
        ssh_free(sshSession);
        return 1;
    }

    // Transférer les données du fichier distant vers le fichier local
    char buffer[4096];
    size_t bytesRead;
    while ((bytesRead = sftp_read(remoteFileHandle, buffer, sizeof(buffer))) > 0) {
        if (fwrite(buffer, 1, bytesRead, file) != bytesRead) {
            cout << "Error writing to local file." << endl;
            fclose(file);
            sftp_close(remoteFileHandle);
            sftp_free(sftpSession);
            ssh_disconnect(sshSession);
            ssh_free(sshSession);
            return 1;
        }
    }

    sftp_session sftpSession = sftp_new(sshSession);
    if (!sftpSession) {
        cout << "Échec de la création de la session SFTP." << endl;
        ssh_disconnect(sshSession);
        ssh_free(sshSession);
        return 1;
    }

    // Initialiser la session SFTP
    rc = sftp_init(sftpSession);
    if (rc != SSH_OK) {
        cout << "Échec de l'initialisation de SFTP : " << ssh_get_error(sshSession) << endl;
        sftp_free(sftpSession);
        ssh_disconnect(sshSession);
        ssh_free(sshSession);
        return 1;
    }
}
```

Par la suite, il est nécessaire de pouvoir traiter ses fichiers, afin de pouvoir en tirer les informations essentielles.

Le traitement de ses fichiers s'effectuent de la façon suivante :

-Dans un premier temps, nous allons récupérer le numero de l'erreur

-Dans un second temps, nous allons détecter de quel types est le problème (dhcp, ntpd, gateways)

-Puis nous appelerons la fonction associé au problème, puis écrirons dans la base de données la ligne où le problème est détecté.

Nous appelerons trois fonctions permettant de suivre ce parcours :

-RecuperationValErreur : qui permet de récupérer la valeur de l'erreur

```
int commfw::RecuperationValErreur()
{
    Connexion_BDD();
    string demande_val_Erreur = "SELECT Val_Erreur FROM Numero_Erreur ORDER BY Date_Heure desc LIMIT 1;";
    const char *requeteSelectChar = demande_val_Erreur.c_str();

    // Exécuter la requête de sélection
    if (mysql_query(conn, requeteSelectChar)) {
        cout << "Erreur lors de l'exécution de la requête de sélection : " << mysql_error(conn) << endl;
        mysql_close(conn);
        return 1;
    }

    // Récupérer le résultat de la requête de sélection
    MYSQL_RES *resultat = mysql_store_result(conn);
    if (resultat) {
        MYSQL_ROW ligne = mysql_fetch_row(resultat);
        if (ligne) {
            int valeur = atoi(ligne[0]);
            cout << "Valeur récupérée : " << valeur << endl;
            return valeur;
        } else {
            cout << "Aucun résultat trouvé." << endl;
        }
        mysql_free_result(resultat);
    } else {
        cout << "Erreur lors de la récupération du résultat de la requête de sélection." << endl;
    }
}
```

détecter et inscrite dans la table Numero_Erreur de la base de données

- EcritureBDD : qui permet d'écrire la ligne correspondant à l'erreur et de renvoyer le numéro de l'erreur dans la table Log

```
int commfw::Ecriture_BDD()
{
    Connexion_BDD();
    int val_erreur=RecuperationValErreur();
    char* Description = NULL;

    switch(val_erreur) {
        case 1:
            Description = Traitement_log_ntpd();
            break;
        case 2:
            Description = "Débit inférieur à 3Mb/s";
            break;
        case 3:
            Description = "Problèmes matériels liés au switch";
            break;
        case 4:
            Description = Traitement_log_dhcp();
            break;
        case 5:
            Description = "Problèmes de résolutions de noms (DNS)";
            break;
        case 6:
            Description = Traitement_log_gateways();
            break;
        default:
            Description = "Aucun problème détecté";
            break;
    }

    string description>Description);
    string requete = "INSERT INTO Log (Val_Erreur, Description_Erreur) VALUES ('" + to_string(val_erreur) + "', '" + description + "');";

    // Exécuter la requête d'insertion
    const char *requeteChar = requete.c_str();
    if (mysql_query(conn, requeteChar)) {
        cout << "Erreur lors de l'exécution de la requête d'insertion : " << mysql_error(conn) << endl;
        mysql_close(conn);
        return 1;
    }
    cout << "Requête d'insertion exécutée avec succès." << endl;
    mysql_close(conn);

    return 0;
}
```

Traitement_log : qui permet de renvoyer la ligne correspondant à l'erreur et de la stocker dans un pointeur

```

char* commfw::Traitement_log_ntpd()
{
    string nomFichier = "/home/etu903/log_firewall/log_firewall_ntp.txt";
    ifstream fichier(nomFichier);
    string ligne;
    string derniereLigneTrouvee;

    if (!fichier.is_open()) {
        cout << "Impossible d'ouvrir le fichier." << endl;
    }

    while (getline(fichier, ligne)) {
        if (ligne.find("TIME_ERROR") != string::npos) {
            derniereLigneTrouvee = ligne;
        }
    }

    fichier.close();

    if (!derniereLigneTrouvee.empty()) {
        cout << derniereLigneTrouvee << endl;
    } else {
        cout << "Aucune ligne contenant TIME_ERROR n'a été trouvée." << endl;
    }
    char* Chaine_log_ntpd = new char[derniereLigneTrouvee.size() + 1];
    strcpy(Chaine_log_ntpd, derniereLigneTrouvee.c_str());
    return Chaine_log_ntpd;
}

```

Afin de traiter chacun des trois fichiers, nous effectuons notre analyse de fichier selon 2 critères :

-La date de l'erreur

-Les mots clés concernant les erreurs

Les mots clés sont différents selon les fichiers : -Gateways : « error »

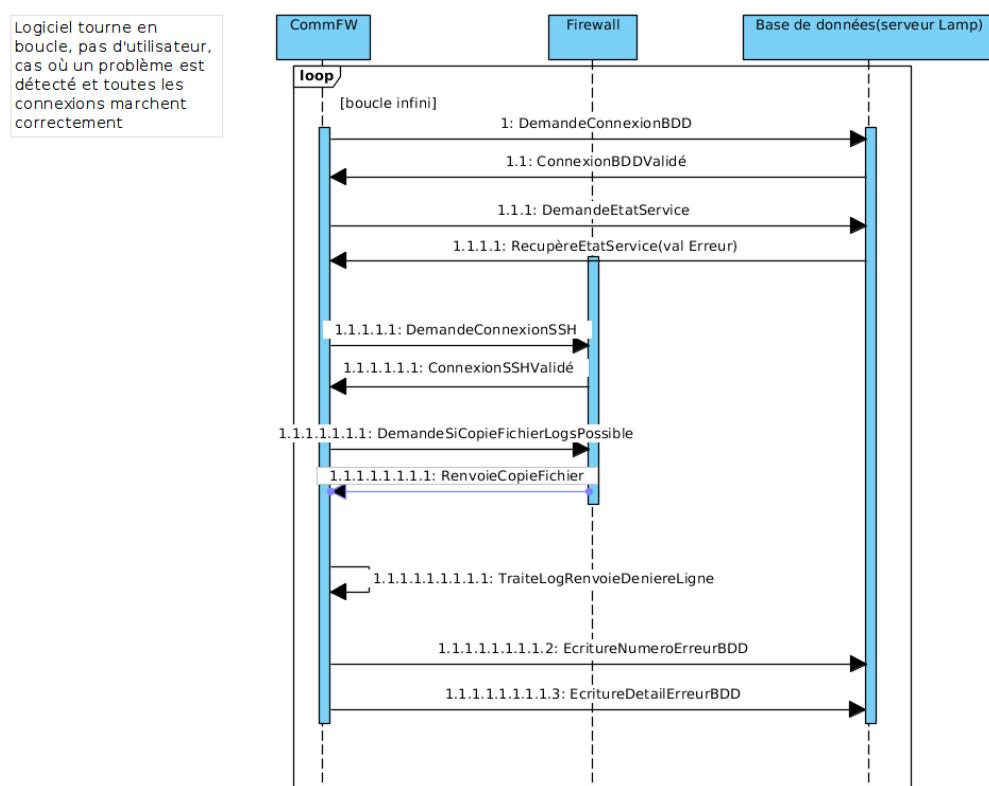
-DHCP : « Abandoning » et « no free leases »

-NTPD : « TIME_ERROR »

Puis pour finir, nous allons arranger un peu notre code afin d'avoir une seule fonction de connexion : c'est à dire la fonction « Connexion_BDD » qui permet de directement se connecter à la base de données, elle renvoie un message erreur si c'est impossible.

Pour mieux comprendre le schéma complet, je vous invite à regarder ci-dessous le diagramme de séquence qui permet la communication entre logiciel, pare-feu et base de données

sd Diagramme de séquence : Communication entre logiciel, Pare-Feu et base de données



Test Unitaire :

Fiche de test

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
 Test de faisabilité
 Test unitaire
 Test d'intégration

Nom de l'étudiant : GAUTRON Léo

Date : 27/05/2024

Objectif de la recette : Etablir le bon fonctionnement de la partie Pfsense + code du projet

Conditions de réalisation

- Besoins matériels
 - Pare-feu Pfsense
 - Ordinateur
 - Switch
 - Accès Intranet
- Besoins logiciels (nom et version)
 - Base de données
- Pré-requis
 - Base de données modélisé

Scénario

ID	Démarche / Opération	Comportement / Résultat attendu	Status OK / NOK
1	Installation et configuration de l'OS pfsense sur un ordinateur	Le pare-feu est fonctionnel, l'accès au log est possible, la connexion ssh est activé	OK
2	Création et mise au point de l'application CommFW	L'application récupère le numéro de l'erreur et envoie bien à la base de données, le numéro de l'erreur, la description tiré des fichiers logs.	OK

Fiche de test

- | |
|--|
| <input type="checkbox"/> Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
<input type="checkbox"/> Test de faisabilité
<input checked="" type="checkbox"/> Test unitaire
<input type="checkbox"/> Test d'intégration |
|--|

Nom de l'étudiant : GAUTRON Léo

Date : 27/05/2024

Objectif du test : Vérifier l'envoi de données vers la base de données

Conditions de réalisation

- Besoins matériels
 - Pare-feu
 - Raspberry
 - Ordinateur
- Besoins logiciels (nom et version)
 - Base de données
- Noms des fichiers utilisés
 - log_firewall.txt

Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	On branche le pfsense afin d'avoir un côté LAN et WAN, il faut pouvoir avoir un connexion entre PC et Base de données		Le pfsense est allumé	OK
2	Il faut ensuite ouvrir le dossier contenant le programme CommFW	/home/admin/CommFW	Le répertoire en haut est bien home/admin/CommFW et il contient bien l'executable CommFW	OK
3	Puis dans ce répertoire, il faut ouvrir un terminal et executer le programme CommFW avec la commande ./CommFW	./CommFW	Le programme se lance bien et ouvre un terminal avec plusieurs informations concernant les connexions	OK
4	Puis il faut ouvrir un navigateur	Chrome, Firefox, Opera...	Le navigateur s'ouvre	OK
5	Rentrer dans l'URL , l'ip de la base de données	192.168.1.246	L'ip nous envoie sur une interface graphique WEB avec un tableau concernant les derniers fichiers logs obtenues	OK
6	Puis observer si la dernière date de log correspond à l'heure actuelle		L'heure et la date correspondent bien à l'heure et la date actuelle	OK

Problèmes rencontrés :

Partie Pfsense :

Lors de la mise en place de Pfsense, nous avons rencontré plusieurs problèmes. Tout d'abord, le disque dur de l'ordinateur destiné à Pfsense présentait de nombreux problèmes d'installation. Il était défectueux, ce qui nous a obligés à le remplacer avant de pouvoir continuer avec l'installation du système. Ensuite, comme nous étions déjà derrière un autre pare-feu Pfsense, nous devions configurer un serveur DNS. Malheureusement, nous ne disposions pas de l'adresse IP de ce serveur, ce qui a compliqué la configuration réseau initiale.

Partie C++ :

Lors de notre développement en C++, nous avons rencontré plusieurs problèmes. Tout d'abord, certaines bibliothèques que nous souhaitions utiliser n'étaient plus à jour et ne fonctionnaient plus correctement. Cela nous a obligés à chercher des alternatives de ces bibliothèques, ce qui s'est avéré chronophage. Ensuite, l'apprentissage et l'utilisation de ces bibliothèques ont été particulièrement difficiles, ajoutant une couche supplémentaire de complexité à notre projet. Nous avons également fait face à des erreurs fuites et basiques, notamment lors de la création de l'exécutable avec les bibliothèques MySQL et libssh. Ces erreurs mineures ont souvent entraîné des retards significatifs.

En ce qui concerne la partie base de données, il a fallu structurer notre programme de manière rigoureuse pour garantir une récupération et un renvoi efficace des données. La connexion à la base de données MySQL a nécessité une attention particulière pour gérer correctement les requêtes et les réponses. L'intégration de

libssh a également été cruciale pour assurer la sécurité des échanges de données. Malgré ces défis, nous avons réussi à mettre en place un système cohérent qui permet à notre programme de récupérer les données nécessaires et de les renvoyer de manière efficace et sécurisée.

Solutions trouvées :

Pour surmonter les défis rencontrés lors de notre projet, nous avons adopté plusieurs solutions efficaces. Concernant les bibliothèques C+ + obsolètes, nous avons cherché sur Internet des bouts de code et des solutions alternatives pour les faire fonctionner correctement. Cette recherche en ligne nous a permis de trouver des exemples et des astuces pour contourner les problèmes de compatibilité.

Pour les problèmes liés à Pfsense, nous avons demandé un accès au pare-feu, spécifiquement l'adresse IP, afin de pouvoir configurer correctement le serveur DNS. Cela a résolu les complications liées à la configuration réseau initiale.

En ce qui concerne le disque dur défectueux de l'ordinateur destiné à Pfsense, nous l'avons remplacé par un nouveau disque dur, ce qui nous a permis de continuer l'installation sans interruption.

La programmation, bien que complexe et chronophage, a été facilitée par l'utilisation d'exemples de données que nous avons trouvés en ligne. Ces exemples nous ont guidés dans la structuration de notre programme, particulièrement pour la partie base de données. En suivant ces modèles, nous avons pu assurer une récupération et un renvoi efficaces des données, intégrant correctement les bibliothèques MySQL et libssh dans notre projet. De plus, la partie traitement des fichiers s'est avérée particulièrement difficile. Cependant, grâce à une recherche approfondie et à des exemples pratiques, nous avons réussi à implémenter des solutions robustes

pour le traitement des fichiers. Grâce à ces solutions, nous avons pu surmonter les obstacles et mener à bien notre développement.

Conclusion :

En conclusion, ce projet a été une expérience riche en apprentissages et en défis, me permettant de développer de nouvelles compétences en programmation et en gestion de réseau. Bien que nous ayons réussi à mettre en place une solution fonctionnelle, il existe plusieurs possibilités d'évolution et d'améliorations futures.

Tout d'abord, l'intégration d'un véritable pare-feu Netgate pourrait améliorer la sécurité et la performance de notre infrastructure réseau. Ensuite, réécrire le code en langage C permettrait de tirer parti des capacités multitâches, optimisant ainsi les performances de notre application.

Par ailleurs, il serait bénéfique de faire évoluer notre programme pour qu'il puisse renvoyer la date et l'heure, ce qui améliorerait la précision des données traitées. Un traitement plus approfondi des fichiers logs est également envisageable, permettant une analyse plus détaillée et pertinente des informations. De plus, l'extension du programme pour qu'il puisse gérer un plus grand nombre de tests et de fichiers renforcerait sa robustesse et sa flexibilité.

Une autre amélioration significative serait l'utilisation d'une clé de connexion pour accéder à la base de données et au pare-feu, éliminant ainsi la nécessité d'entrer le mot de passe à chaque fois. Cette méthode augmenterait la sécurité et la commodité d'utilisation de notre système.

Enfin, la mise en place d'un mode de fonctionnement dégradé assurerait une continuité de service en cas de défaillance partielle du système, augmentant ainsi la fiabilité globale de notre solution.

Ces perspectives d'évolution témoignent de l'ambition de notre projet et des nombreuses opportunités d'amélioration qui s'offrent à nous. En continuant à explorer ces pistes, nous pourrions développer une solution encore plus performante et adaptée aux besoins futurs.

Annexe :

1.0 Manuel d'utilisation : Installer et configurer Pfsense

Manuel de Configuration et d'Installation de Pfsense avec Rufus sur un PC

Pré-requis

1. Matériel nécessaire :

- Un PC dédié pour Pfsense.
- Une clé USB d'au moins 1 Go.
- Un autre ordinateur pour créer la clé USB bootable.

2. Logiciels nécessaires :

- Rufus (outil pour créer des clés USB bootables).
- Image ISO de Pfsense (téléchargeable sur le site officiel de Pfsense).

Étape 1 : Téléchargement des fichiers nécessaires

1. Télécharger Rufus :

- Rendez-vous sur le site officiel de Rufus
- Téléchargez la dernière version de Rufus.

2. Télécharger l'ISO de Pfsense :

- Rendez-vous sur le site officiel de Pfsense

- Sélectionnez la dernière version de Pfsense stable

Étape 2 : Crédation de la clé USB bootable avec Rufus

1. Insérer la clé USB :

- Insérez la clé USB dans l'ordinateur qui servira à créer la clé USB bootable.

2. Lancer Rufus :

- Ouvrez l'application Rufus.

3. Configuration de Rufus :

- Dans la section "Périphérique", sélectionnez votre clé USB.
- Cliquez sur "Sélection" pour choisir l'image ISO de Pfsense que vous avez téléchargée.
- Les autres paramètres devraient se configurer automatiquement

4. Démarrer le processus :

- Cliquez sur "Démarrer" pour lancer le processus de création de la clé USB bootable.
- Un avertissement apparaîtra vous informant que tous les fichiers sur la clé USB seront supprimés. Confirmez et continuez.
- Attendez que le processus se termine.

Étape 3 : Installation de Pfsense sur le PC

1. Démarrage sur la clé USB :

- Insérez la clé USB dans le PC destiné à Pfsense.
- Démarrez le PC et accédez au BIOS (généralement en appuyant sur une touche comme F2)
- Configurez le BIOS/UEFI pour démarrer à partir de la clé USB.

2. Lancer l'installation de Pfsense :

- Le PC devrait démarrer sur l'interface d'installation de Pfsense.
- Sélectionnez les options par défaut pour la plupart des étapes.
- Lorsque vous arrivez à l'étape de partitionnement, choisissez d'installer Pfsense sur le disque dur principal.

3. Configuration initiale de Pfsense :

- Après l'installation, le système redémarrera. Retirez la clé USB pour permettre au PC de démarrer sur le disque dur.
- Suivez les instructions à l'écran pour la configuration initiale, notamment la configuration du réseau (LAN et WAN).

-Entrer les adresses IP LAN et WAN (LAN : 192.168.1.254)

4. Accéder à l'interface web de Pfsense :

- Une fois la configuration réseau de base terminée, connectez un ordinateur au réseau LAN de Pfsense.
- Ouvrez un navigateur web et accédez à l'interface web de Pfsense via l'adresse IP par défaut (généralement <http://192.168.1.254>).
- Connectez-vous avec les identifiants par défaut (nom d'utilisateur : `admin`, mot de passe : `pfsense`).

5. Finaliser la configuration :

- Suivez l'assistant de configuration pour ajuster les paramètres de base, y compris le mot de passe administrateur, la configuration du pare-feu, et les paramètres DNS.
- Configurez les paramètres DNS en demandant l'adresse IP correcte si vous êtes derrière un autre pare-feu.

Étape 4 : Configuration avancée et maintenance

1. Mise à jour de Pfsense :

- Accédez à l'onglet "System" > "Update" pour vérifier et appliquer les mises à jour disponibles.

2. Configuration des règles de pare-feu :

- Utilisez l'interface web pour configurer des règles de pare-feu adaptées à vos besoins spécifiques (onglet "Firewall" > "Rules").

En suivant ce manuel, vous devriez être en mesure de configurer et d'installer Pfsense avec succès en utilisant Rufus sur un PC. Pour des configurations plus avancées, il est recommandé de consulter la documentation officielle de Pfsense et d'explorer les forums de support pour des conseils supplémentaires.

2.0 Code C++ :

2.1 CommFW.h :

```
#ifndef COMMFW_H
#define COMMFW_H
#include <iostream>
#include <libssh/libssh.h>
#include <libssh/sftp.h>
#include <fcntl.h>
#include <unistd.h>
#include <fstream>
#include <string>
#include <vector>
#include <mysql/mysql.h>
#include <sstream>
#include <string.h>
#include <cstdlib>

class commfw
{
private :
    MYSQL *conn;
```

```
public:  
    commfw();  
    int Copie_Fichier_log();  
    char* Traitement_log_dhcp();  
    char* Traitement_log_ntpd();  
    char* Traitement_log_gateways();  
    int Ecriture_BDD();  
    int RecuperationValErreur();  
    int Connexion_BDD();  
};
```

```
#endif // COMMFW_H
```

2.2 CommFW.cpp :

```
#include "commfw.h"  
using namespace std;  
  
commfw::commfw()  
{  
}  
  
}  
int commfw::Copie_Fichier_log()  
{  
    ssh_session sshSession = ssh_new(); // Initialiser la session SSH  
    if (!sshSession) {  
        cout << "Impossible de créer la session SSH" << endl;  
        return 1;  
    }  
  
    // Définir les options de la session SSH  
    ssh_options_set(sshSession, SSH_OPTIONS_HOST,  
    "172.16.150.10");  
    ssh_options_set(sshSession, SSH_OPTIONS_USER, "admin");  
  
    // Se connecter au serveur  
    int rc = ssh_connect(sshSession);  
    if (rc != SSH_OK) {
```

```
    cout << "Échec de la connexion au serveur : " <<
ssh_get_error(sshSession) << endl;
    ssh_free(sshSession);
    return 1;
}

// Authentification avec mot de passe
rc = ssh_userauth_password(sshSession, nullptr, "admin7557");
if (rc != SSH_AUTH_SUCCESS) {
    cout << "Échec de l'authentification : " <<
ssh_get_error(sshSession) << endl;
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}

// Ouvrir une session SFTP
sftp_session sftpSession = sftp_new(sshSession);
if (!sftpSession) {
    cout << "Échec de la création de la session SFTP." << endl;
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}

// Initialiser la session SFTP
rc = sftp_init(sftpSession);
if (rc != SSH_OK) {
    cout << "Échec de l'initialisation de SFTP : " <<
ssh_get_error(sshSession) << endl;
    sftp_free(sftpSession);
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}

// File paths for remote and local files
```

```

const char *remoteFiles[] = {"/var/log/dhcpd.log", "/var/log/ntp.log",
"/var/log/gateways.log"};
const char *localFiles[] =
{"/home/etu903/log_firewall/log_firewall_dhcp.txt",
"/home/etu903/log_firewall/log_firewall_ntp.txt",
"/home/etu903/log_firewall/log_firewall_gateways.txt"};

// Copy each remote file to respective local file
for (int i = 0; i < 3; ++i) {
    // Open remote file for reading
    sftp_file remoteFileHandle = sftp_open(sftpSession,
remoteFiles[i], O_RDONLY, 0);
    if (!remoteFileHandle) {
        cout << "Failed to open remote file: " <<
ssh_get_error(sshSession) << endl;
        sftp_free(sftpSession);
        ssh_disconnect(sshSession);
        ssh_free(sshSession);
        return 1;
    }

    // Open local file for writing
    FILE *file = fopen(localFiles[i], "wb");
    if (!file) {
        cout << "Failed to open local file." << endl;
        sftp_close(remoteFileHandle);
        sftp_free(sftpSession);
        ssh_disconnect(sshSession);
        ssh_free(sshSession);
        return 1;
    }

    // Transfer data from remote file to local file
    char buffer[4096];
    size_t bytesRead;
    while ((bytesRead = sftp_read(remoteFileHandle, buffer,
sizeof(buffer))) > 0) {

```

```

if (fwrite(buffer, 1, bytesRead, file) != bytesRead) {
    cout << "Error writing to local file." << endl;
    fclose(file);
    sftp_close(remoteFileHandle);
    sftp_free(sftpSession);
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}
}

// Close file handles
fclose(file);
sftp_close(remoteFileHandle);
}

// Close SFTP session
sftp_free(sftpSession);
// Disconnect SSH session
ssh_disconnect(sshSession);
// Free SSH session
ssh_free(sshSession);

cout << "Les fichiers ont été copiés avec succès !" << endl;
return 0;
}

char* commfw::Traitement_log_dhcp()
{
    string nomFichier =
"/home/etu903/log_firewall/log_firewall_dhcp.txt";
    ifstream fichier(nomFichier);
    string ligne;
    string dernièreLigneTrouvée;

    if (!fichier.is_open()) {
        cout << "Impossible d'ouvrir le fichier." << endl;

```

```

}

while (getline(fichier, ligne)) {
    if (ligne.find("Abandoning") != string::npos || ligne.find("no free
leases") != string::npos) {
        derniereLigneTrouvee = ligne;
    }
}

fichier.close();

if (!derniereLigneTrouvee.empty()) {
    cout << derniereLigneTrouvee << endl;
} else {
    cout << "Aucune ligne contenant Abandoning et no free leases
n'a été trouvée." << endl;
}
char* Chaine_log_dhcp = new char[derniereLigneTrouvee.size() +
1];
strcpy(Chaine_log_dhcp, derniereLigneTrouvee.c_str());
return Chaine_log_dhcp;
}

char* commfw::Traitement_log_ntpd()
{
    string nomFichier = "/home/etu903/log_firewall/log_firewall_ntp.txt";
    ifstream fichier(nomFichier);
    string ligne;
    string derniereLigneTrouvee;

    if (!fichier.is_open()) {
        cout << "Impossible d'ouvrir le fichier." << endl;
    }

    while (getline(fichier, ligne)) {
        if (ligne.find("TIME_ERROR") != string::npos ) {
            derniereLigneTrouvee = ligne;
        }
    }
}
```

```

        }
    }

fichier.close();

if (!derniereLigneTrouvee.empty()) {
    cout << derniereLigneTrouvee << endl;
} else {
    cout << "Aucune ligne contenant TIME_ERROR n'a été trouvée."
<< endl;
}
char* Chaine_log_ntpd = new char[derniereLigneTrouvee.size() +
1];
strcpy(Chaine_log_ntpd, derniereLigneTrouvee.c_str());
return Chaine_log_ntpd;
}

char* commfw::Traitement_log_gateways()
{
{
    string nomFichier =
"/home/etu903/log_firewall/log_firewall_gateways.txt";
    ifstream fichier(nomFichier);
    string ligne;
    string derniereLigneTrouvee;

    if (!fichier.is_open()) {
        cout << "Impossible d'ouvrir le fichier." << endl;
    }

    while (getline(fichier, ligne)) {
        if (ligne.find("error") != string::npos ) {
            derniereLigneTrouvee = ligne;
        }
    }
}

fichier.close();

```

```

if (!derniereLigneTrouvee.empty()) {
    cout << derniereLigneTrouvee << endl;
} else {
    cout << "Aucune ligne contenant TIME_ERROR n'a été
trouvée." << endl;
}
char* Chaine_log_ntpd = new char[derniereLigneTrouvee.size()
+ 1];
strcpy(Chaine_log_ntpd, derniereLigneTrouvee.c_str());
return Chaine_log_ntpd;
}
}

int commfw::Ecriture_BDD()
{
    Connexion_BDD();
    int val_erreur=RecuperationValErreur();
    char* Description = NULL;

    switch(val_erreur) {
        case 1:
            Description = Traitement_log_ntpd();
            break;
        case 2:
            Description = "Debit inférieur à 3Mb/s";
            break;
        case 3:
            Description = "Problèmes matériels liés au switch";
            break;
        case 4:
            Description = Traitement_log_dhcp();
            break;
        case 5:
            Description = "Problèmes de résolutions de noms (DNS)";
            break;
        case 6:
            Description = Traitement_log_gateways();
            break;
    }
}

```

```

    default:
        Description = "Aucun problème détecté";
    break;
    break;
}

string description>Description);
string requete = "INSERT INTO Log (Val_Erreur,
Description_Erreur) VALUES (" + to_string(val_erreur) + ", " +
description + ")";

// Exécuter la requête d'insertion
const char *requeteChar = requete.c_str();
if (mysql_query(conn, requeteChar)) {
    cout << "Erreur lors de l'exécution de la requête d'insertion : " <<
mysql_error(conn) << endl;
    mysql_close(conn);
    return 1;
}
cout << "Requête d'insertion exécutée avec succès." << endl;
mysql_close(conn);

return 0;
}
int commfw::RecuperationValErreur()
{
    Connexion_BDD();
    string demandeval_Erreur = "SELECT Val_Erreur FROM
Numero_Erreur ORDER BY Date_Heure desc LIMIT 1;";
    const char *requeteSelectChar = demandeval_Erreur.c_str();

// Exécuter la requête de sélection
if (mysql_query(conn, requeteSelectChar)) {
    cout << "Erreur lors de l'exécution de la requête de sélection : "
<< mysql_error(conn) << endl;
    mysql_close(conn);
    return 1;
}

```

```

}

// Récupérer le résultat de la requête de sélection
MYSQL_RES *resultat = mysql_store_result(conn);
if (resultat) {
    MYSQL_ROW ligne = mysql_fetch_row(resultat);
    if (ligne) {
        int valeur = atoi(ligne[0]);
        cout << "Valeur récupérée : " << valeur << endl;
        return valeur;
    } else {
        cout << "Aucun résultat trouvé." << endl;
    }
    mysql_free_result(resultat);
} else {
    cout << "Erreur lors de la récupération du résultat de la requête
de sélection." << endl;
}
}

int commfw::Connexion_BDD()
{
    // Paramètres de connexion
    const char *serveur = "192.168.1.246";
    const char *utilisateur = "admin";
    const char *motDePasse = "admin5775";
    const char *baseDeDonnees = "Network_Monitor";
    const int port = 3306; // Port MySQL par défaut

    conn = mysql_init(nullptr); // Initialisation de la structure de
    connexion

    if (!conn) {
        cout << "Impossible d'initialiser la connexion MySQL." << endl;
        return 1;
    }

    // Établir la connexion à la base de données

```

```

if (!mysql_real_connect(conn, serveur, utilisateur, motDePasse,
baseDeDonnees, port, nullptr, 0)) {
    cout << "Impossible de se connecter à la base de données : " <<
mysql_error(conn) << endl;
    mysql_close(conn);
    return 1;
}
cout << "Connexion réussie à la base de données MySQL." <<
endl;
}

```

2.3 main.cpp :

```

#include "commfw.h"
using namespace std;

int main()
{
    int apt=0;
    commfw App;
    do{
        App.Copie_Fichier_log();
        App.Ecriture_BDD();
        sleep(60);
    }
    while(apt==0);
}

```

Test lines findings :

```

#include <iostream>
#include <fstream>
#include <string>
#include <vector>

```

```

int CommFW::Traitement_log_dhcp()
{
    std::string nomFichier =
"/home/etu903/log_DHCP/log_firewall_dhcp.txt";
    std::ifstream fichier(nomFichier);
    std::string ligne;
    std::string derniereLigneTrouvee;

    if (!fichier.is_open()) {
        std::cerr << "Impossible d'ouvrir le fichier." << std::endl;
        return 1;
    }

    while (getline(fichier, ligne)) {
        if (ligne.find("DHCPDISCOVER") != std::string::npos &&
ligne.find("no free leases") != std::string::npos) {
            derniereLigneTrouvee = ligne;
        }
    }

    fichier.close();

    if (!derniereLigneTrouvee.empty()) {
        std::cout << "Dernière ligne contenant DHCPDISCOVER et no
free leases :" << std::endl;
        std::cout << derniereLigneTrouvee << std::endl;
    } else {
        std::cout << "Aucune ligne contenant DHCPDISCOVER et no
free leases n'a été trouvée." << std::endl;
    }

    return 0;
}

```

Code testant si la copie de fichier local distant est possible :

```

#include <iostream>
#include <libssh/libssh.h>

```

```
#include <libssh/sftp.h>
#include <fcntl.h> /* For O_RDONLY */
#include <unistd.h> /* For open(), creat() */
#include <sstream> /* For stringstream */
#include <vector> /* For vector */

using namespace std;

// Function to split a string into lines
vector<string> splitIntoLines(const string& input) {
    vector<string> lines;
    stringstream ss(input);
    string line;
    while (getline(ss, line, '\n')) {
        lines.push_back(line);
    }
    return lines;
}

int main() {
    // Initialize SSH session
    ssh_session sshSession = ssh_new();
    if (!sshSession) {
        cerr << "Failed to create SSH session." << endl;
        return 1;
    }

    // Set options
    ssh_options_set(sshSession, SSH_OPTIONS_HOST,
    "172.16.150.10");
    ssh_options_set(sshSession, SSH_OPTIONS_USER, "admin");

    // Connect to the server
    int rc = ssh_connect(sshSession);
    if (rc != SSH_OK) {
```

```
    cerr << "Failed to connect to server: " <<
ssh_get_error(sshSession) << endl;
    ssh_free(sshSession);
    return 1;
}

// Authenticate with password
rc = ssh_userauth_password(sshSession, nullptr, "admin7557");
if (rc != SSH_AUTH_SUCCESS) {
    cerr << "Authentication failed: " << ssh_get_error(sshSession) <<
endl;
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}

// Open SFTP session
sftp_session sftpSession = sftp_new(sshSession);
if (!sftpSession) {
    cerr << "Failed to create SFTP session." << endl;
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}

// Initialize SFTP session
rc = sftp_init(sftpSession);
if (rc != SSH_OK) {
    cerr << "SFTP initialization failed: " <<
ssh_get_error(sshSession) << endl;
    sftp_free(sftpSession);
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}
```

```
// Open remote file for reading
const char *remoteFile = "/var/log/dhcpd.log";
sftp_file remoteFileHandle = sftp_open(sftpSession, remoteFile,
O_RDONLY, 0);
if (!remoteFileHandle) {
    cerr << "Failed to open remote file: " <<
ssh_get_error(sshSession) << endl;
    sftp_free(sftpSession);
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}

// Seek to the end of the file
if (sftp_seek(remoteFileHandle, -1, SEEK_END) != SSH_OK) {
    cerr << "Failed to seek to the end of the file." << endl;
    sftp_close(remoteFileHandle);
    sftp_free(sftpSession);
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}

// Read data from the end of the file
stringstream lastLinesBuffer;
char lastChar = '\0';
int numLines = 0;
while (numLines < 40) {
    char currentChar;
    if (sftp_read(remoteFileHandle, &currentChar, 1) != 1) {
        cerr << "Error reading file." << endl;
        sftp_close(remoteFileHandle);
        sftp_free(sftpSession);
        ssh_disconnect(sshSession);
        ssh_free(sshSession);
        return 1;
```

```
}

lastLinesBuffer << currentChar;

if (lastChar == '\n') {
    numLines++;
}

lastChar = currentChar;

// Move the file pointer back by 2 bytes
if (sftp_seek(remoteFileHandle, -2, SEEK_CUR) != SSH_OK) {
    cerr << "Failed to seek back in the file." << endl;
    sftp_close(remoteFileHandle);
    sftp_free(sftpSession);
    ssh_disconnect(sshSession);
    ssh_free(sshSession);
    return 1;
}

// Break loop if we have reached the beginning of the file
if (sftp_tell(remoteFileHandle) == 0) {
    break;
}
}

// Close remote file handle
sftp_close(remoteFileHandle);

// Close SFTP session
sftp_free(sftpSession);

// Disconnect SSH session
ssh_disconnect(sshSession);

// Free SSH session
```

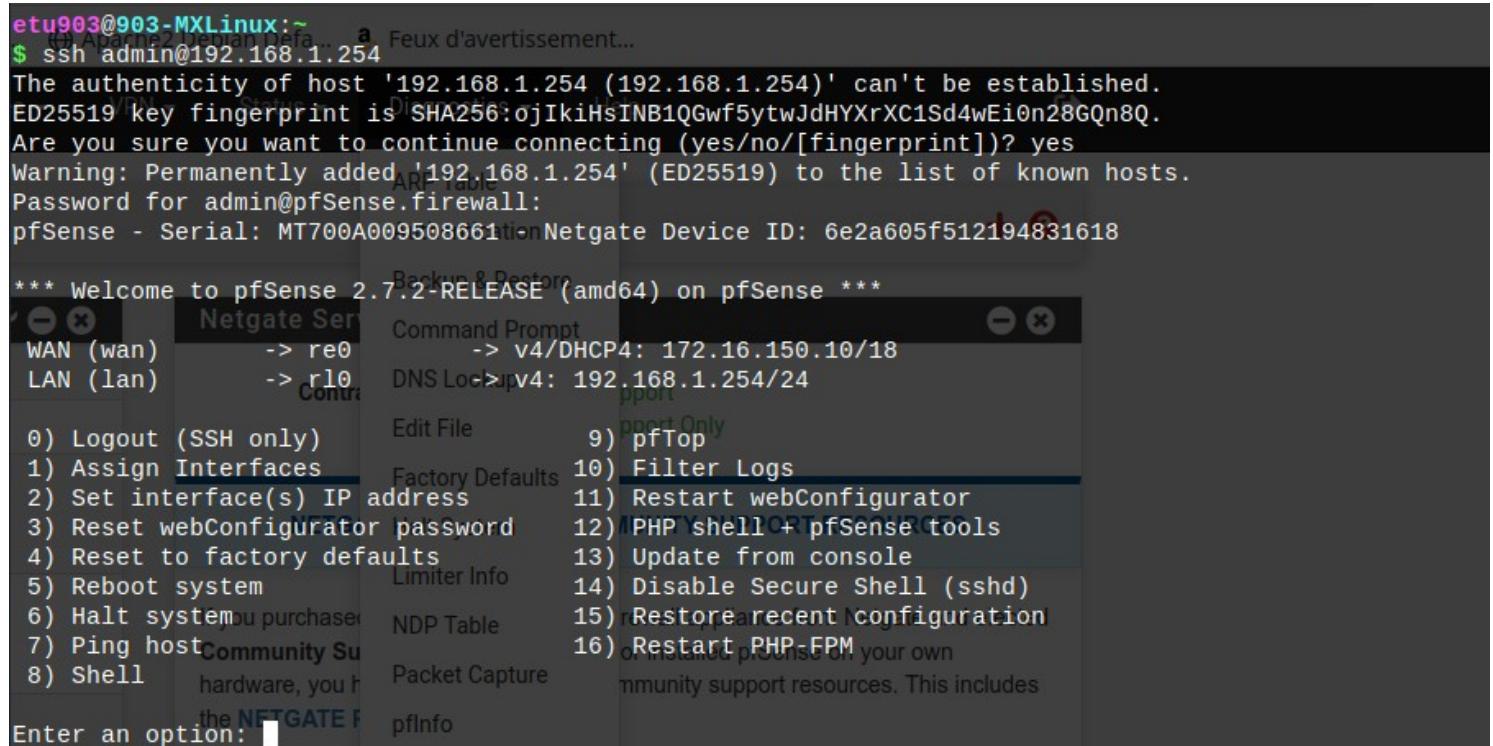
```

    ssh_free(sshSession);

    // Get the last 40 lines from the buffer
    string lastLines = lastLinesBuffer.str();
    vector<string> lines = splitIntoLines(lastLines);
    for (const string& line : lines) {
        cout << line << endl;
    }

    return 0;
}

```



```

etu903@903-MXLinux:~ $ ssh admin@192.168.1.254
The authenticity of host '192.168.1.254 (192.168.1.254)' can't be established.
ED25519 key fingerprint is SHA256:ojIkiHsINB1QGwf5ytwJdHYXrXC1Sd4wEi0n28GQn8Q.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.254' (ED25519) to the list of known hosts.
Password for admin@pfSense.firewall:
pfSense - Serial: MT700A009508661ti-nNetgate Device ID: 6e2a605f512194831618

*** Welcome to pfSense 2.7.2-RELEASE (amd64) on pfSense ***
[Netgate Server] Command Prompt [Report Only]
WAN (wan)      -> re0      -> v4/DHCP4: 172.16.150.10/18
LAN (lan)      -> r10     DNS Lockupv4: 192.168.1.254/24

0) Logout (SSH only)      Edit File      9) pfTop
1) Assign Interfaces      Factory Defaults 10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP Shell OPT pfSense tools
4) Reset to factory defaults 13) Update from console
5) Reboot system          Limiter Info 14) Disable Secure Shell (sshd)
6) Halt system            NDP Table 15) Restore recent configuration
7) Ping host              Community Su 16) Restart PHP FPM
8) Shell                  hardware, you h  Enter an option: [the NETGATE F

Connexion SSH Pfsense :

```

Code connexion BDD :

```

int commfw::Connexion_BDD()
{
    // Paramètres de connexion
    const char *serveur = "127.0.0.1";
    const char *utilisateur = "admin";
    const char *motDePasse = "admin5775";
    const char *baseDeDonnees = "Network Monitor";
    const int port = 3306; // Port MySQL par défaut

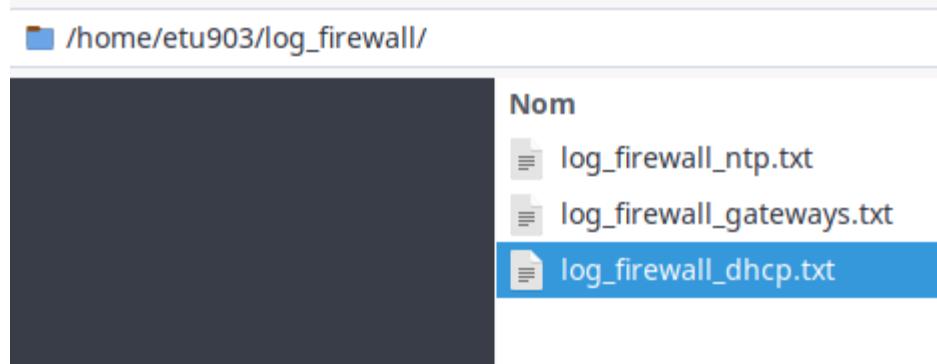
    conn = mysql_init(nullptr); // Initialisation de la structure de connexion

    if (!conn) {
        cout << "Impossible d'initialiser la connexion MySQL." << endl;
        return 1;
    }

    // Établir la connexion à la base de données
    if (!mysql_real_connect(conn, serveur, utilisateur, motDePasse, baseDeDonnees, port, nullptr, 0)) {
        cout << "Impossible de se connecter à la base de données : " << mysql_error(conn) << endl;
        mysql_close(conn);
        return 1;
    }
    cout << "Connexion réussie à la base de données MySQL." << endl;
}

```

Répertoire contenant les fichiers logs (PC):



1.9Partie individuelle : Serhat Turan

Introduction :

Dans le cadre de notre projet informatique, nous avons été confrontés à un défi crucial : la fiabilité du réseau de la section SNIR (Systèmes Numériques et Réseaux Informatiques) de notre lycée. L'année dernière, ce réseau a été fréquemment affecté par des dysfonctionnements variés, perturbant ainsi les activités éducatives et pratiques des élèves. Afin de remédier à cette situation et d'assurer une meilleure stabilité et performance du réseau, nous avons développer un système de tests automatisés.

Ce projet s'articule principalement autour de la réalisation de divers tests réseaux en C++. En tant que responsable de la programmation des tests, j'ai développé des modules spécifiques pour évaluer différents aspects critiques du réseau. Les tests incluent des vérifications DNS, DHCP, passerelle, NTP, ainsi que des évaluations de débit Internet et de fonctionnalité des switches Ethernet.

Ces tests ont pour objectif de détecter de manière proactive les problèmes potentiels, de prévenir le technicien via un signal lumineux et de faciliter la maintenance préventive. Ainsi, nous visons à améliorer significativement la stabilité du réseau et à offrir un environnement d'apprentissage optimal pour les élèves de la section SNIR. Nous envoyons également les erreurs détecté dans une base de données.

En résumé, ce projet représente une réponse technique aux besoins concrets du lycée, combinant programmation et compréhension approfondie des infrastructures réseaux pour créer un outil de surveillance et de diagnostic efficace.

Les Solutions Multiples Envisagées :

Pour répondre aux défis de fiabilité du réseau de la section SNIR du lycée, plusieurs solutions ont été envisagées avant de sélectionner la meilleure approche. Les critères de choix incluaient la facilité d'intégration, la robustesse de la solution, et la capacité à fournir des diagnostics clairs et rapides. Voici un aperçu des solutions envisagées et la justification du choix final :

1. Langage de Programmation

a. Python

- Avantages : Facilité d'apprentissage, bibliothèque riche pour les tests réseau.
- Inconvénients : Performances moins optimales pour des tests intensifs en temps réel.

b. Java

- Avantages : Portabilité, support étendu pour le développement réseau.
- Inconvénients : Nécessite une machine virtuelle, moins performant pour des systèmes embarqués.

c. C++

- Avantages : Haute performance, contrôle précis des ressources système, grande familiarité du programmeur.
- Inconvénients : Complexité de programmation plus élevée.

Justification du Choix :

Le choix du C++ s'est imposé principalement en raison de notre aisance avec ce langage et de ses performances optimales pour des tests en temps réel. Le C++ permet une gestion fine des ressources, essentielle pour les diagnostics réseau précis.

2. Plateforme Matérielle

a. Serveur Dédié

- Avantages : Puissance de calcul élevée, flexibilité.
- Inconvénients : Coût élevé, consommation énergétique importante.

b. Ordinateur Portable

- Avantages : Portabilité, facilité d'accès.
- Inconvénients : Moins adapté pour une intégration continue et permanente dans l'infrastructure réseau.

c. Raspberry Pi

- Avantages : Faible coût, faible consommation d'énergie, taille compacte, parfaite pour une solution embarquée.
- Inconvénients : Performances limitées par rapport à un serveur dédié.

Justification du Choix :

Le Raspberry Pi a été choisi en raison de sa simplicité d'intégration comme système embarqué, son coût abordable, et sa consommation énergétique réduite. Il offre suffisamment de puissance pour exécuter les tests nécessaires sans encombrer le réseau existant.

3. Système de Signalisation

a. Interface Graphique

- Avantages : Visuel clair, interactif.
- Inconvénients : Nécessite un écran, moins adapté pour une notification instantanée et visible de loin.

b. Notification par Email/SMS

- Avantages : Informations détaillées, accessibles partout.
- Inconvénients : Dépendance à un autre réseau (internet ou mobile), moins de visibilité immédiate.

c. Colonne de Signalisation à Trois Lumières avec Buzzer

- Avantages : Visibilité immédiate, simplicité de compréhension, installation fixe.
- Inconvénients : Indique seulement le niveau de gravité, sans détails spécifiques.

Justification du Choix :

La colonne de signalisation à trois lumières avec un buzzer a été choisie pour sa simplicité et son efficacité en termes de signalisation immédiate. Les niveaux de gravité sont facilement identifiables :

- Vert : Fonctionnement normal.
- Orange : Problème mineur.
- Rouge : Problème majeur.
- Rouge + Buzzer : Problème critique nécessitant une intervention immédiate.

Cette approche permet aux techniciens de réagir rapidement aux incidents sans nécessiter une surveillance constante d'un écran ou de notifications électroniques.

Conclusion :

Après avoir évalué plusieurs solutions, nous avons choisi d'utiliser le C++ pour la programmation en raison de notre expertise avec ce langage et de ses performances. Le Raspberry Pi a été sélectionné comme plateforme matérielle pour son coût, son efficacité énergétique et sa simplicité d'intégration. Enfin, la colonne de

signalisation à trois lumières avec buzzer a été retenue pour sa simplicité et son efficacité en termes de signalisation visuelle et sonore. Cette combinaison nous permet de proposer une solution intégrée, performante et adaptée aux besoins spécifiques de notre lycée.

La conception, configuration, réalisation :

Installation de la colonne lumineuse sur le raspberry pi :

Afin d'envoyer toutes nos erreurs dans la Base de données on a une fonction qui demande simplement la valeur de l'erreur et qui va ensuite gérer le message à envoyer:

```
void TestNet::ConnexionBDD(int val_erreur)
{
    MYSQL *conn;
    // Initialisez la structure de connexion
    conn = mysql_init(nullptr);
    if (!conn) {

        // retourne une erreur
    }

    // Paramètres de connexion
    const char *serveur = "127.0.0.1";
    const char *utilisateur = "admin";
    const char *motDePasse = "admin5775";
    const char *baseDeDonnees = "Network_Monitor";
    const int port = 3306; // Port MySQL par défaut

    // Établir la connexion à la base de données
    if (!mysql_real_connect(conn, serveur, utilisateur, motDePasse, baseDeDonnees, port, nullptr, 0)) {

        mysql_close(conn);
        // retourne une erreur
    }

    // Définir le problème en fonction de la valeur d'erreur
    const char* Problem;
    switch (val_erreur) { // Allumage des lumières en fonction de l'erreur
    case 0:
        Problem = "Aucun Problème";
        relay.allOff();
        relay.on(2);
        break;
    case 1:
        Problem = "problème Fonctionnement du service NTP";
        relay.on(3);
        break;
    case 2:
        Problem = "problème Qualité du réseau";
        relay.on(1);
        break;
    case 3:
        Problem = "problème Fonctionnement du switch";
        relay.on(1,4);
        sleep(3);

        break;
    case 4:
        Problem = "problème Fonctionnement du service DHCP";
    }
}
```

```

case 3:
    Problem = "problème Fonctionnement du switch";
    relay.on(1,4);
    sleep(3);

    break;
case 4:
    Problem = "problème Fonctionnement du service DHCP";
    relay.on(1,4);
    sleep(3);

    break;
case 5:
    Problem = "problème Fonctionnement de la résolution de nom";
    relay.on(1,4);
    sleep(3);

    break;
case 6:
    Problem = "problème Accessibilité de la passerelle";
    relay.on(1,4);
    sleep(3);

    break;
default:
    Problem = "Erreur inconnue";
    break;
}

char Description[200];
strcpy(Description, Problem);

string description(Description);
char requete[500]; // Augmentez la taille pour éviter le débordement de tampon
sprintf(requete, "INSERT INTO Numero_Erreur (`Val_Erreur`, `Description_Erreur`) VALUES ('%d', '%s')", val_erreur, Description);

const char *requeteChar = requete;

// Exécuter la requête
if (mysql_query(conn, requeteChar)) {

    mysql_close(conn);
    // retourne une erreur
}

// Fermer la connexion
mysql_close(conn);

```

Test NTP :

Qu'est que NTP :

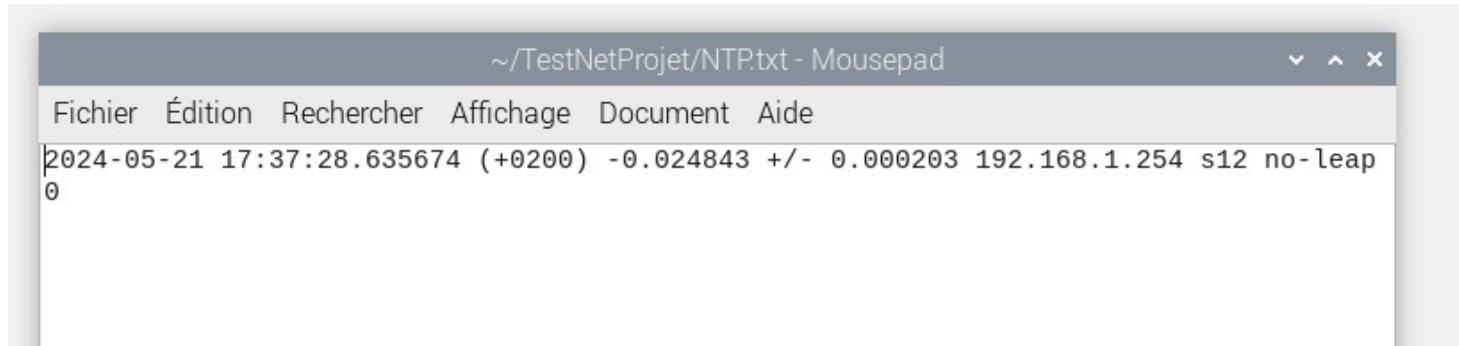
Le NTP (Network Time Protocol) est un protocole conçu pour synchroniser les horloges des systèmes informatiques sur un réseau. Il permet de coordonner précisément l'heure entre différents dispositifs, ce qui est essentiel pour la sécurité, les transactions financières, et d'autres applications critiques. NTP utilise une hiérarchie de serveurs pour diffuser l'heure exacte provenant de

sources fiables comme les horloges atomiques ou les services GPS, assurant une cohérence temporelle à travers le réseau.

Comment tester le protocole NTP ?

Pour tester le bon fonctionnement du protocole NTP du pfSense(pare-feu), on se doit de stopper le protocole sur le raspberry via la commande système : « systemctl stop ntp » puis nous devons mettre à jour le NTP du raspberry en fonction du pfSense donc via la commande : « ntpdate -v 192.168.1.254 », l'adresse IP en question est l'adresse IP du pfSense. Cette commande va automatiquement relancer le protocole et mis à jour en fonction du pfSense, on enchaîne avec la commande « echo \$? » qui va renvoyer « 1 » en cas d'erreur détectée et « 0 » si tout s'est bien passé. Nous pouvons voir les résultats reçus en se rendant dans le fichier « NTP.txt », on peut y voir lors de la réussite la mise à jour du protocole et la valeur « 0 ». Alors que lors d'un échec on peut voir des messages d'erreurs ainsi que la valeur « 1 » à la ligne suivante. Voir les différentes captures ainsi que les résultats attendus via la colonne de signalisation :

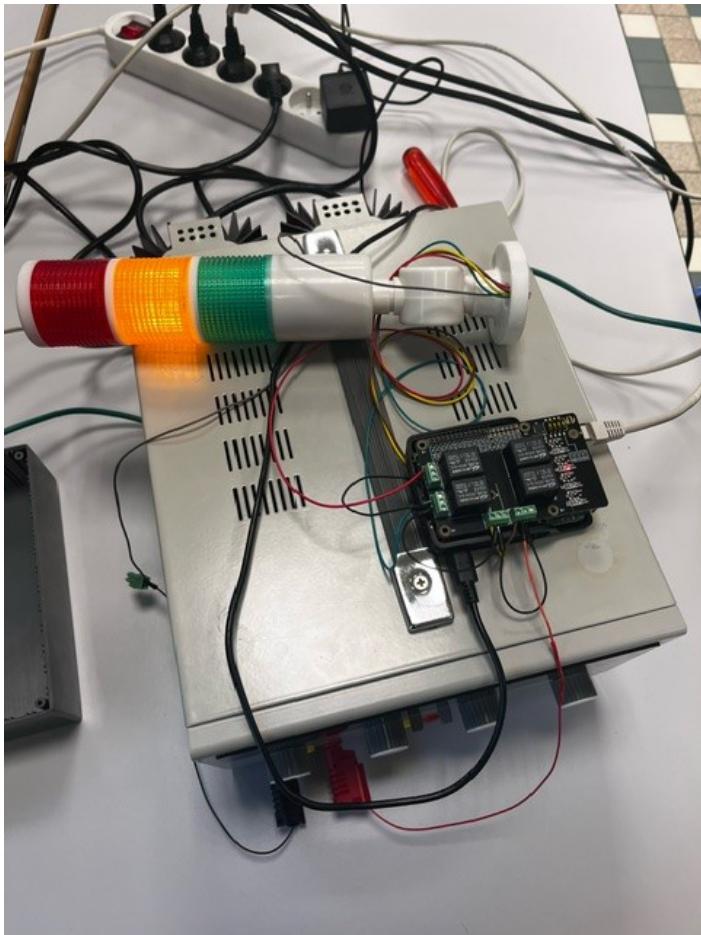
Capture du fichier « NTP.txt » lors de la réussite :



The screenshot shows a terminal window titled '~ /TestNetProjet/NTP.txt - Mousepad'. The menu bar includes 'Fichier', 'Édition', 'Rechercher', 'Affichage', 'Document', and 'Aide'. The main pane displays the following text:

```
2024-05-21 17:37:28.635674 (+0200) -0.024843 +/- 0.000203 192.168.1.254 s12 no-leap
0
```

Lampe si échec du test NTP (gravité faible donc lumière orange):



Extrait de code du test NTP :

```

}

void TestNet::NTP()
{
    char InfosTerminal[200];
    int i=0;
    int longueur;
    fstream ntp("NTP.txt");
    freopen("NTP.txt","w+",stdout); // on stocke tout ce qui sort du terminal dans un fichier

    system("sudo systemctl stop ntp"); //arret du service ntp
    system("sudo ntpdate -v 192.168.1.254 && echo $?"); //application du NTP du pfsense sur ce poste

    ntp.close();
    fstream lecture("NTP.txt");

    while(!lecture.eof()) //Pour recuperer seulement le dernier element du fichier ( 0 ou 1 car resultat de echo $?)
    {
        lecture >> InfosTerminal;

    }
    lecture.close();

    if(InfosTerminal[0] != '0') // Si apres test "echo $" on ne recoit pas 0, probleme detecte
    {
        Valeur_Erreur = 1;

        ConnexionBDD(Valeur_Erreur);
    }
    else // Si apres test "echo $" on recoit 0, pas de probleme
    {

        if(Valeur_Erreur == 1)
        {
            Valeur_Erreur = 0;
            ConnexionBDD(Valeur_Erreur);
        }
    }
}

```

Test Débit Internet :

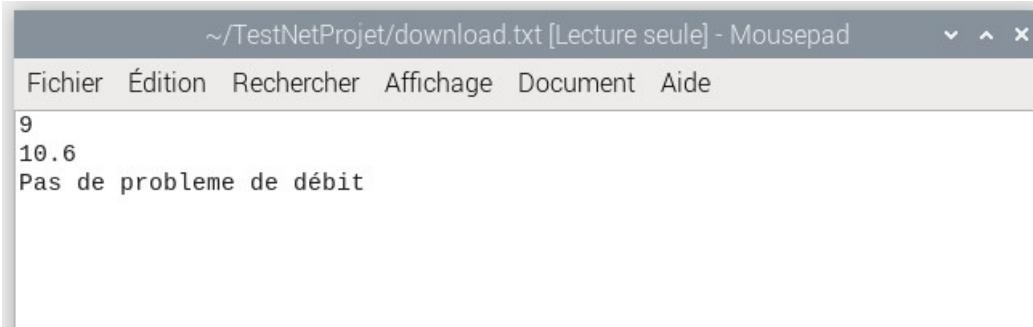
Qu'est ce que le débit Internet :

Le débit internet, ou bande passante, mesure la quantité de données transférées via une connexion internet en un temps donné, généralement en Mbps (mégabits par seconde). Il se divise en débit descendant (download), pour les données reçues, et débit montant (upload), pour les données envoyées. Ce débit influence la vitesse de téléchargement, la qualité du streaming vidéo, et la fluidité des vidéoconférences.

Pour tester le débit internet, nous avons procédé en téléchargeant un fichier (avec curl) qui permet d'installer depuis un terminal de 100 Mb via un lien(« <https://bouygues.testdebit.info/100M.iso> »), en le dirigeant vers /dev/null pour supprimer immédiatement le contenu sans le stocker sur le disque. Nous avons démarré un timer avant de lancer le téléchargement et l'avons arrêté une fois le téléchargement terminé, ce qui nous a permis de mesurer précisément le temps total d'installation. En divisant la taille du fichier (100 Mb) par le temps de téléchargement obtenu (par exemple $100\text{Mb}/15\text{s} = 6.66\text{Mb/s}$), nous avons pu calculer le débit internet. Cette méthode nous a fourni une mesure précise du débit en Mb/s, reflétant ainsi les performances réelles de notre connexion internet. Grâce à cette approche, nous avons pu évaluer efficacement le débit disponible et vérifier la qualité de notre connexion réseau.

Ainsi pour le test de débit, j'ai conclu que en étant inférieur à 3 Mb/s cela refléter un problème, car en France la loi oblige les fournisseurs internet à fournir a minima 8Mb/s.

Fichier « download.txt » lors de la réussite (débit > 3Mb/s) :

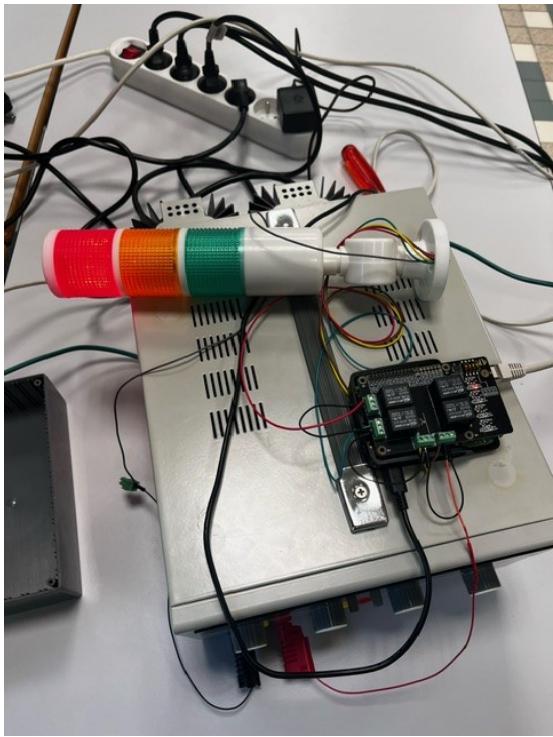


The screenshot shows a terminal window titled '~/TestNetProjet/download.txt [Lecture seule] - Mousepad'. The menu bar includes 'Fichier', 'Édition', 'Rechercher', 'Affichage', 'Document', and 'Aide'. The main pane displays the following text:

```
9
10.6
Pas de probleme de débit
```

On peut y voir en première ligne le temps d'installation et en deuxième ligne le débit en Mb/s.

Lampe si débit < 3Mb/s (gravité moyenne donc lumière rouge):



Extrait de code du test du débit :

```
j void TestNet::QualiteReseau()
{
    float temps;
    float taillefichier = 95.4;
    float debit;

    fstream download("download.txt");
    freopen("download.txt","w",stdout); // on stocke tout ce qui sort du terminal

    time_t begin = time(NULL); //on démarre un timer
    system( "curl -4 -o /dev/null https://bouygues.testdebit.info/100M.iso");
    // on télécharge un fichier de 100mb dans un fichier qui le supprimera directement

    download.close();
    temps = end - begin; //on calcule le temps d'installation

    cout << temps << endl;
    debit = taillefichier / temps; // on calcule le débit
    cout << debit << endl;

    if(debit < 3) // Si débit < a 3mb/s probleme de débit détecté
    {
        Valeur_Erreur = 2;
        ConnexionBDD(Valeur_Erreur);
    }
    else // Si débit > a 3mb/s pas de probleme
    {

        cout <<"Pas de probleme de débit"<<endl;
        if(Valeur_Erreur == 2)
        {
            Valeur_Erreur = 0;
            ConnexionBDD(Valeur_Erreur);
        }
    }
}
```

Test switch ethernet :

Le but des tests des switch ethernet consiste à vérifier si les ports des switch sont fonctionnels afin d'éviter de chercher des erreurs autre part, cela nécessite d'avoir des switch manageable donc en pouvant lui attribuer une IP, malheureusement la section SNIR du lycée n'étant pas faites avec des switch manageable, ce test devient inutile et bien trop complexe à réaliser dans la section si nous devions remplacer tous les switch.

Test DHCP :

En conclusion de notre test DHCP, nous avons suivi une procédure systématique pour vérifier le bon fonctionnement du protocole DHCP sur notre réseau. Voici les étapes détaillées que nous avons suivies :

Extraction initiale des adresses IP : Nous avons commencé par extraire toutes les adresses IP actuelles et les avons enregistrées dans un fichier. Cette étape nous a permis d'avoir une référence initiale des adresses IP attribuées par le serveur DHCP.

Suppression de l'adresse IP : Ensuite, nous avons supprimer l'adresse IP actuelle en utilisant la commande appropriée. Cette opération simule la libération de l'adresse IP par un client DHCP.

Nouvelle demande d'adresse IP : Après avoir libéré l'adresse IP, nous avons effectué une nouvelle demande d'adresse IP en utilisant la commande «dhclient». Cette commande de supprimer les IP présentes sur la machine..

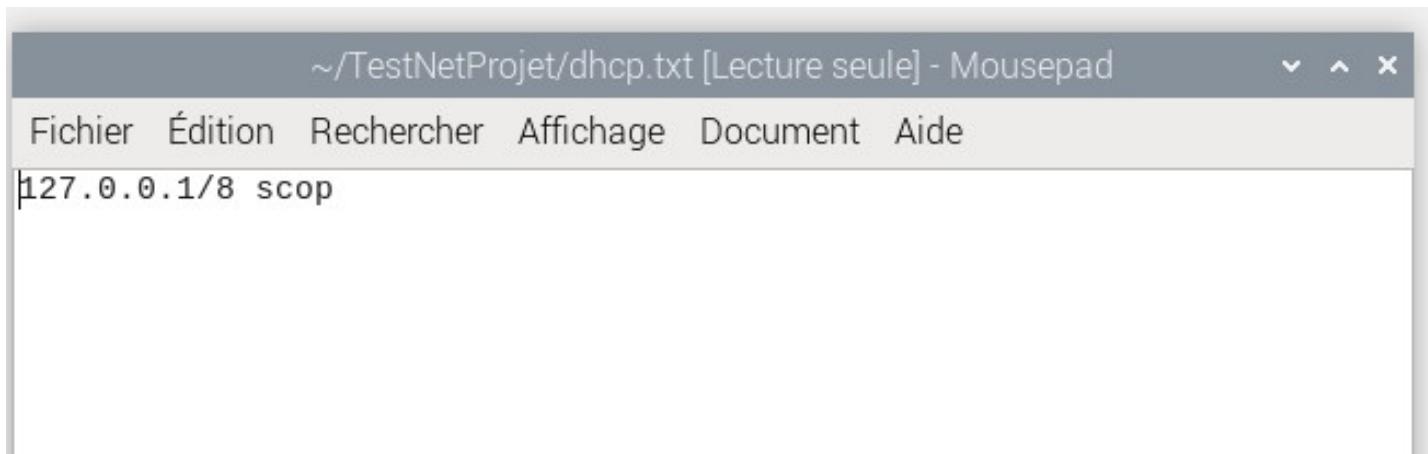
Extraction des nouvelles adresses IP : Une fois la nouvelle adresse IP attribuée, nous avons extrait de nouveau toutes les adresses IP et les avons comparées aux adresses extraites initialement. Cette comparaison nous permet de vérifier si le

serveur DHCP attribue les adresses IP de manière cohérente et conforme aux attentes.

En comparant les deux séries d'adresses IP extraites, si dans la seconde extraction on obtient pas notre IP, cela signifie que le protocole DHCP n'est pas fonctionnel. D'ailleurs, après avoir attribuer une IP fixe à l'adresse MAC de notre poste, après une nouvelle demande d'IP on obtient toujours celle que l'on a décidé (L'IP de notre raspberry ici est : 192.168.1.246).

Ainsi, ce test vérifie que le serveur DHCP fonctionne de manière fiable et efficace, garantissant une attribution des adresses IP aux périphériques sur le réseau.

Fichier « dhcp.txt » après la 1ere extraction d'ip, on peut y voir que seule le localhost est présent car les autres IP ont été supprimées :



The screenshot shows a window titled '~/TestNetProjet/dhcp.txt [Lecture seule] - Mousepad'. The menu bar includes 'Fichier', 'Édition', 'Rechercher', 'Affichage', 'Document', and 'Aide'. The main text area contains the line '127.0.0.1/8 scop'.

```
127.0.0.1/8 scop
```

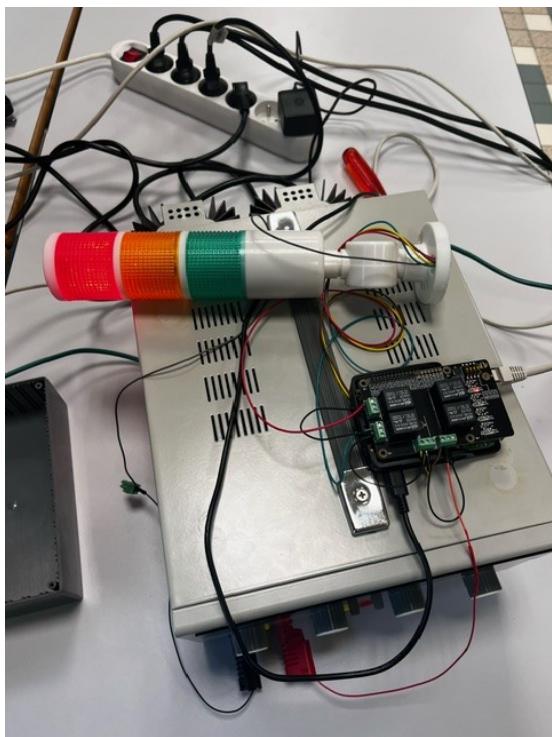
Fichier « dhcp2.txt » après la 2eme extraction d'ip, on peut y voir qu'on a retrouvé l'IP fixe que l'on avait choisie (via l'adresse MAC) après la nouvelle demande d'IP :

```
*~/TestNetProjet/dhcp2.txt [Lecture seule] - Mousepad
```

Fichier Édition Rechercher Affichage Document Aide

```
127.0.0.1/8 scop
192.168.1.246/24
Pas de probleme DHCP|
```

Lampe si échec du test DHCP(gravité critique donc lumière rouge + buzzer):



Extrait de code du test DHCP :

```
void TestNet::testDHCP()
{
    string InfosDHCP;
    string InfosDHCP2;

    fstream DHCP("dhcp.txt");
    freopen("dhcp.txt", "w+", stdout); // on stocke tout ce qui sort du terminal dans un fichier 1
    system("sudo dhclient"); // On demande une ip
    system("sudo dhclient -r"); // On supprime l'ip
    system("ip a | grep \"inet \" | cut -c 10-25"); // on récupere toutes les ip présentes
    DHCP.close();

    fstream DHCP2("dhcp2.txt");
    freopen("dhcp2.txt", "w+", stdout); // on stocke tout ce qui sort du terminal dans un fichier 2
    system("sudo dhclient"); // on demande une ip

    system("ip a | grep \"inet \" | cut -c 10-25"); // on récupere toutes les ip
    DHCP2.close();

    ifstream lecture("dhcp.txt");

    while(!lecture.eof()) // on récupere la dernière ligne du fichier 1
    {
        lecture >> InfosDHCP;
    }

    lecture.close();

    ifstream lecture2("dhcp2.txt"); // on récupere la dernière ligne du fichier 2
    while(!lecture2.eof())
    {
        lecture2 >> InfosDHCP2;
    }
    lecture2.close();

    if (strcmp(InfosDHCP.c_str(), InfosDHCP2.c_str()) == 0) // on compare les 2 lignes récupérées
    {
        Valeur_Erreur = 4;
        ConnexionBDD(Valeur_Erreur); // lignes identiques donc problème
    }
    else // lignes différentes donc nouvelle ip recuperer alors pas de problemes
    {
        cout << "Pas de probleme DHCP" << endl;
        if(Valeur_Erreur == 4)
        {
            Valeur_Erreur = 0;
            ConnexionBDD(Valeur_Erreur);
        }
    }
}

char commande[100];
```

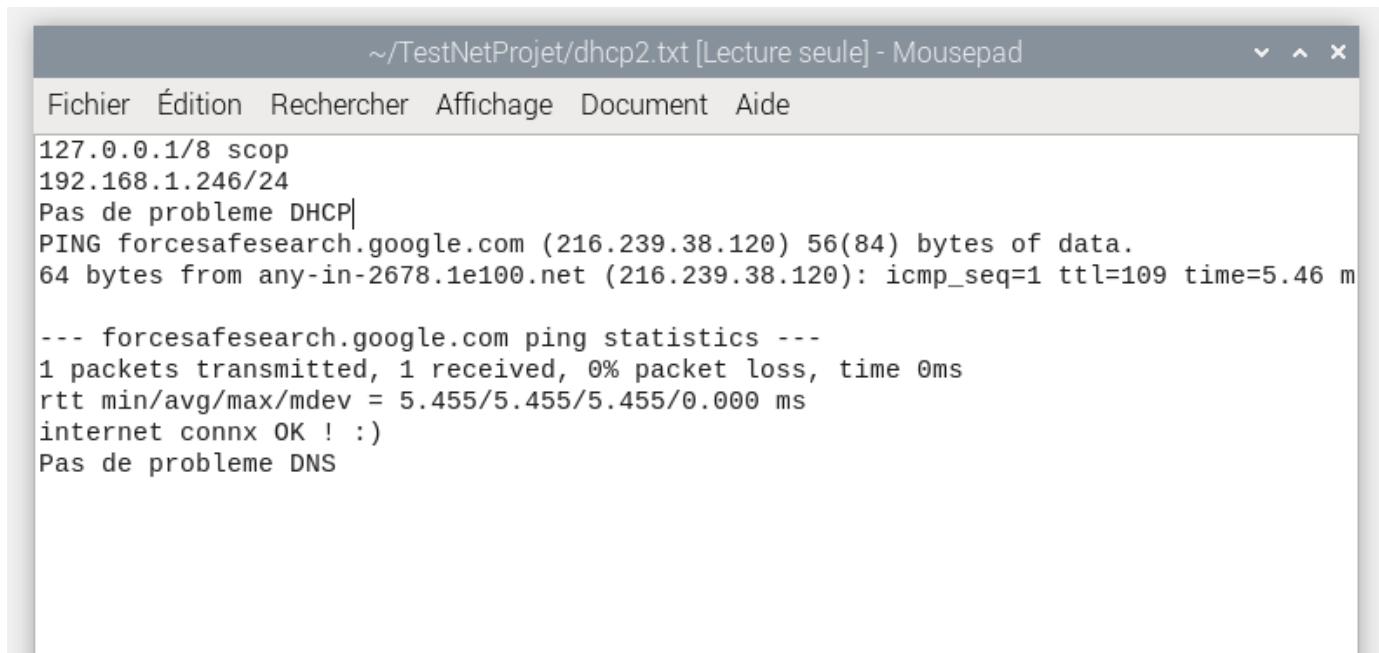
Test DNS :

Qu'est ce qu'un DNS :

Le protocole DHCP (Dynamic Host Configuration Protocol) est un protocole réseau qui automatise l'attribution des adresses IP et autres paramètres de configuration aux appareils sur un réseau.

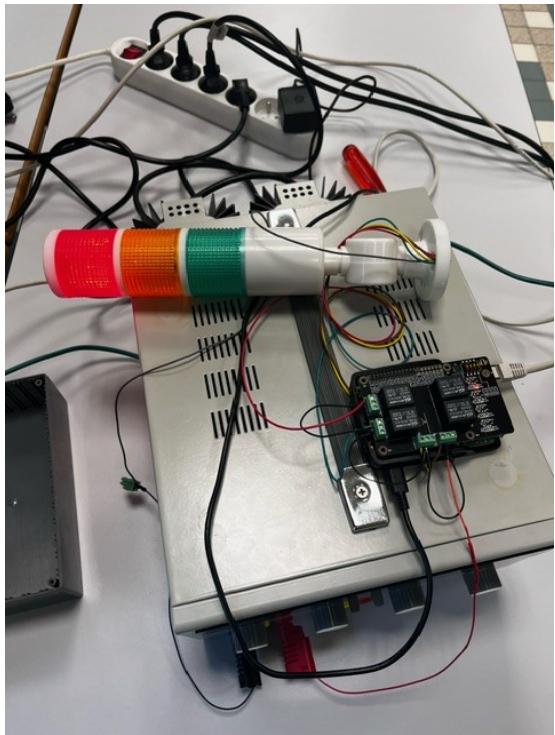
Pour réaliser ce test du DNS, nous réalisons un ping sur l'adresse www.google.com, si ce ping échoue on en conclu que le service DNS n'est pas fonctionnel et on enverra donc dans la BDD un problème et on allumera la colonne lumineuse en conséquences.

Fichier « dhcp2.txt » qui contient également le résultat du test DNS on peut y voir la réussite du ping de google :



```
~/TestNetProjet/dhcp2.txt [Lecture seule] - Mousepad
Fichier Édition Rechercher Affichage Document Aide
127.0.0.1/8 scop
192.168.1.246/24
Pas de probleme DHCP|
PING forcesafesearch.google.com (216.239.38.120) 56(84) bytes of data.
64 bytes from any-in-2678.1e100.net (216.239.38.120): icmp_seq=1 ttl=109 time=5.46 m
--- forcesafesearch.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 5.455/5.455/5.455/0.000 ms
internet connx OK ! :)
Pas de probleme DNS
```

Lampe si échec du test DNS(gravité critique donc lumière rouge + buzzer):



Extrait de code du test DNS :

```
void TestNet::pingDNS()
{
    if (system( "ping -c1 www.google.com" ) ) { // On ping une adresse dns
        printf("internet connx failed \n");
        Valeur_Erreur = 5;
        ConnexionBDD(Valeur_Erreur); // Probleme détecté donc on envoi l'erreur à la BDD
    }
    else {
        printf("internet connx OK ! :) \n");

        cout <<"Pas de probleme DNS" << endl;
        if(Valeur_Erreur == 5)
        {
            Valeur_Erreur = 0;
            ConnexionBDD(Valeur_Erreur);
        }
    }
}
```

Test Passerelle :

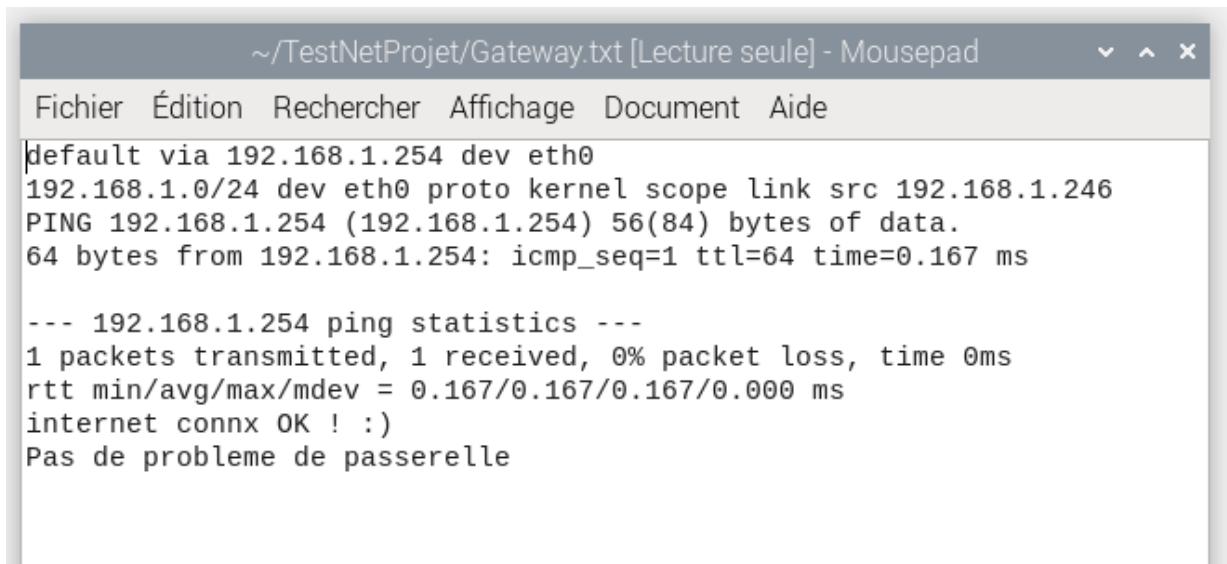
Qu'est ce que la passerelle :

Dans le monde de l'informatique, une passerelle est une composante essentielle pour connecter différents réseaux entre eux. Que ce soit pour relier un réseau local à Internet ou pour faciliter la communication entre réseaux distants, les passerelles agissent comme des ponts, permettant le flux d'informations. Elles peuvent être matérielles, sous la forme de routeurs ou de commutateurs, ou logicielles, intégrées à des serveurs. Grâce à leur capacité à traduire les données et à router le trafic, les passerelles jouent un rôle crucial dans l'interconnexion des réseaux, assurant ainsi une communication fluide et sécurisée.

Pour réaliser ce test de fonctionnement et présence de la passerelle, on effectue la commande « ip route », les résultats de celle ci vont être retrancrit dans un document « Gateway.txt ». On va ensuite récupéré

le contenu du fichier dans une chaîne de caractère et on va réaliser un filtrage sur cette chaîne jusqu'à obtenir dans notre chaîne seulement l'adresse IP de la passerelle. Grâce à cette IP on va finalement pouvoir faire ping afin de constater si celle-ci est fonctionnelle ou non. Si aucune IP n'est obtenue après la commande « ip route » ceci est considéré comme une erreur et nous agirons en conséquences donc en envoyant dans la BDD le problème et en allumant la colonne lumineuse de couleur ainsi que le déclenchement du buzzer (gravité de problème critique). Si le ping échoue cela constitue le même problème et les mêmes conséquences.

Fichier « Gateway.txt » dans lequel on peut retrouver la 1ère ligne du résultat de la commande « ip route » dans laquelle on peut retrouver l'IP de la passerelle (route par défaut), puis le résultat du ping de cette adresse.



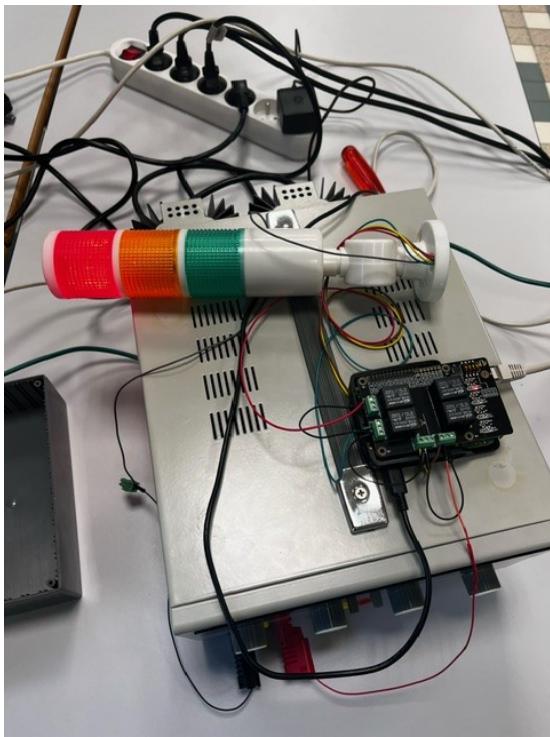
```
~/TestNetProjet/Gateway.txt [Lecture seule] - Mousepad
```

Fichier Édition Rechercher Affichage Document Aide

```
default via 192.168.1.254 dev eth0
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.246
PING 192.168.1.254 (192.168.1.254) 56(84) bytes of data.
64 bytes from 192.168.1.254: icmp_seq=1 ttl=64 time=0.167 ms

--- 192.168.1.254 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.167/0.167/0.167/0.000 ms
internet connx OK ! :)
Pas de problème de passerelle
```

Lampe si échec du test de la passerelle (gravité critique donc lumière rouge + buzzer):



```
void TestNet::FichierGateway()
{
    char InfosTerminal[100];
    int i=0;

    fstream outfile("Gateway.txt"); //on cree et ouvre un fichier nommé gateway.txt et on le nomme outfile

    freopen("Gateway.txt","w",stdout); //Tout ce qui sortira du terminal sera retransmis dans le fichier gateway.txt
    system("ip route"); //On envoie la commande ip route dans le fichier gateway.txt
    // verifier fichier présent !!!
    outfile.getline(InfosTerminal, 100, '\n'); //on recupere la premiere ligne du fichier
    outfile.close();

    string Passerelle (InfosTerminal);
    Passerelle.erase(0,12); //on efface les 12 premiers caracteres de la ligne ("default via ")
    while(i<Passerelle.length())
    {
        if(Passerelle[i] == ' ')
        {
            Passerelle[i]= '\0';
            break;// on sort des le premier espace trouvé
        } //on supprime tout ce qu il y a a partir du premier espace trouvé
        i++;
    }

    ippasserelle= Passerelle;
}
```

Extrait de code du test passerelle :

```

void TestNet::pingpasserelle()
{
    char commande[100];
    sprintf(commande,"ping -c1 %s",ippasserelle.c_str()); // On insère l'ip recuperer da

    if (system( commande ) ) { // On ping l'ip
        printf("internet connx failed \n");
        Valeur_Erreur = 6;
        ConnexionBDD(Valeur_Erreur); // Probleme détecté donc on envoi l'erreur à la BDD
    }
    else {
        printf("internet connx OK ! :) \n");

        cout <<"Pas de probleme de passerelle"<<endl;
        if(Valeur_Erreur == 6)
        {
            Valeur_Erreur = 0;
            ConnexionBDD(Valeur_Erreur);

// On insère l'ip recuperer dans la fonction FichierGateway dans la commande de ping
    }
}

```

```

int main()
{
    int BoucleInfini = 1;
    TestNet test;
    Relay relay;

    relay.allOff();
    relay.on(2);

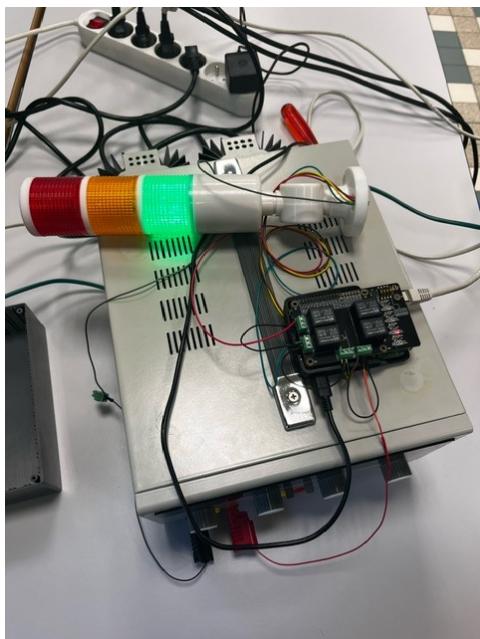
    do{
        test.testDHCP();
        sleep(15);
        test.pingDNS();
        sleep(10);
        test.FichierGateway();
        sleep(5);
        test.pingpasserelle();
        //switch
        sleep(5);
        test.QualiteReseau();
        sleep(5);
        test.NTP();
        sleep(5);

        sleep(10);
    }while(BoucleInfini == 1);
}

```

Afin que le programme indéfiniment sans avoir à le toucher on met une boucle infinie dans le fichier main.cpp afin que ca ne s'arrête jamais (à part à la main depuis le terminal) :

Lorsqu'il n'y a aucun problème la colonne lumineuse est en mode « ok »(lumière verte car aucun problème) :



Améliorations possibles :

L'une des principales amélioration serait de changer les switch de la section SNIR du lycée pour y mettre des switch manageables et ainsi pouvoir réaliser les tests de switchs qui n'ont pas pu être réalisé avant.

Yes=Managed Switches



No=Unmanaged Switches



Mode d'emploi :

Ce programme consiste à réaliser des tests et sauvegarder les résultats en temps réel dans des documents qui se mettent à jour lors de changements, ainsi qu'en envoyer dans la base de données les différents problèmes et à allumer la colonne de signalisation en fonction de la gravité des problèmes constatés (faible, moyen et critique).

Afin d'utiliser ce programme il faut, avoir l'executable dans un document ainsi les documents qui se créeront seront dans le dossier de l'executable, cela facilitera l'accès au document si l'on souhaite y accéder.

La ligne a entrer dans le terminal afin de créer l'executable est :
g++ main.cpp TestNet.cpp -o TestNet -lssh -lmysqlclient

Une fois l'executable présent, nous pouvons l'exécuter avec la commande « ./NomDeLExecutable » (ici ./TestNet), cela exécutera le programme et celui ci tournera à l'infini jusqu'à ce qu'on y mette fin de sois même.

On peut donc mettre fin au programme en exécutant la combinaison de touches « ctrl + C » dans le terminal du programme.

Une fois le programme exécuter, on peut se rendre dans les fichiers (emplacement de l'executable) et constater le contenu des fichiers textes.

Test faisabilité:

Fiche de test

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
 Test de faisabilité
 Test unitaire
 Test d'intégration

Nom de l'étudiant : Turhan Serhat

Date : 13/02/2024

Objectif du test : Vérifier le fonctionnement du service DHCP par un ping

Conditions de réalisation

- Besoins matériels
Pc
switch
parefeu
- Besoins logiciels (nom et version)
...
• Noms des fichiers utilisés
Gateway.txt

Scénario				
ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Réaliser un ping sur une IP d'un ordinateur sur le réseau via le code c++ avec la fonction system()		Ping répondant donc DHCP fonctionnel	OK
2				
3				

```
void TestNet::pingDNS()
{
    if (system( "ping -c1 www.google.com" ) ) {
        printf("internet connx failed \n");
        pingdns = 1;
    }
    else {
        printf("internet connx OK ! : \n");
        pingdns = 0;
    }
}
```

Test Unitaire :

Fiche de test

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
- Test de faisabilité
- Test unitaire
- Test d'intégration

Nom de l'étudiant : Turhan Serhat

Date : 13/02/2024

Objectif du test : tester la qualité du réseau et le débit

Conditions de réalisation

- Besoins matériels

Pc

serveur dhcp/parefeu

paquet avec taille prédefinie

- Besoins logiciels (nom et version)

...

- Noms des fichiers utilisés

...

Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Sur PC Exécuter l'application <u>TestNet</u> en tapant <u>./TestNet</u> dans le terminal		Le débit s'affiche dans le fichier " <u>download.txt</u> "	OK
2				OK
3				

Annexes (captures d'écran et schémas)

...

```
void TestNet::QualiteReseau()
{
    char InfosDownload[100];
    char InfosUpload[100];

    ofstream download("download.txt");
    ofstream("download.txt", std::ofstream::out | std::ofstream::trunc);
    freopen("download.txt","w",stdout);
    system( "curl -4 -o /dev/null https://bouygues.testdebit.info/100M.iso");
    //freopen("download.txt","w",stdout);
    download.close();
```

1.10 Partie individuelle : Matt Beschon

Dans le cadre de mon projet, je suis chargé de plusieurs tâches essentielles pour assurer le bon fonctionnement d'un système embarqué et de son application associée. La première étape consiste à installer le système d'exploitation sur l'ordinateur embarqué, ce qui constitue la base sur laquelle tout le reste du projet se développe. Ensuite, je mets en place un serveur LAMP (Linux, Apache, MySQL, PHP) pour garantir un environnement serveur stable et performant.

Une fois l'infrastructure en place, je passe à la modélisation de la base de données, élément central pour le stockage et la gestion des données de l'application. Parallèlement, je crée et mets au point l'interface homme-machine (IHM), en veillant à ce que ses caractéristiques et performances répondent aux besoins spécifiés.

Pour assurer la qualité et la fiabilité de l'application, je mène des tests de faisabilité pour valider le fonctionnement envisagé. Ces tests permettent de détecter et de corriger d'éventuels problèmes avant la phase de développement final. Par la suite, je développe et teste l'application en continu pour garantir une performance optimale.

Enfin, je produis une documentation complète pour guider les utilisateurs dans l'installation, la configuration et l'utilisation des outils. De plus, je rédige les documents de recette pour formaliser les critères d'acceptation du projet, assurant ainsi que toutes les exigences initiales sont satisfaites.

Solutions envisagées et justifications

Dans le cadre de ce projet, plusieurs solutions ont été envisagées pour répondre aux besoins spécifiques du système embarqué et de l'application associée. Voici pourquoi l'utilisation de PHP et d'un Raspberry Pi avec un serveur LAMP a été retenue :

Utilisation de PHP

1. **Facilité d'intégration** : PHP est un langage de script côté serveur largement utilisé pour le développement web. Il s'intègre facilement avec Apache, MySQL et d'autres composants du serveur LAMP, facilitant le développement et la gestion de l'application.
2. **Large communauté et support** : PHP bénéficie d'une vaste communauté de développeurs, offrant un large éventail de ressources, de bibliothèques et de support technique.
3. **Flexibilité et performance** : PHP est suffisamment flexible pour s'adapter aux différents besoins du projet et offre de bonnes performances pour les applications web légères à moyennes, ce qui est souvent le cas pour les systèmes embarqués.

Utilisation d'un Raspberry Pi

1. **Coût abordable** : Le Raspberry Pi est une solution matérielle très économique, permettant de réduire les coûts globaux du projet tout en offrant des capacités suffisantes pour héberger un serveur LAMP.
2. **Faible consommation d'énergie** : Contrairement à des serveurs traditionnels, le Raspberry Pi consomme très peu d'énergie, ce qui est idéal pour des applications embarquées qui doivent souvent fonctionner en continu.
3. **Compact et portable** : La petite taille du Raspberry Pi en fait un choix idéal pour des projets embarqués où l'espace est limité.
4. **Communauté et documentation** : Le Raspberry Pi dispose d'une vaste communauté et d'une documentation abondante, ce qui facilite le développement et le dépannage.

Cahier des charges et serveur LAMP

Le cahier des charges précise l'utilisation d'un serveur LAMP (Linux, Apache, MySQL, PHP) pour plusieurs raisons :

1. **Compatibilité et stabilité** : Le stack LAMP est reconnu pour sa stabilité et sa compatibilité entre ses différents composants, offrant une plateforme fiable pour le développement d'applications web.
2. **Open Source** : Les composants du stack LAMP sont tous open source, ce qui permet de réduire les coûts logiciels tout en bénéficiant de mises à jour et de support communautaire.
3. **Performance adéquate** : Pour des applications de taille moyenne, le stack LAMP offre des performances suffisantes, surtout lorsqu'il est hébergé sur un Raspberry Pi pour des systèmes embarqués.

En résumé, l'utilisation de PHP et d'un Raspberry Pi avec un serveur LAMP répond de manière optimale aux exigences du projet en termes de coût, de performance, de flexibilité et de support communautaire.

Autres solutions envisagées

Solutions matérielles

Intel NUC

- **Avantages** : Haute performance, support complet pour les systèmes d'exploitation de bureau, robustesse et fiabilité.
- **Inconvénients** : Coût plus élevé, consommation d'énergie plus élevée, taille légèrement plus grande.

MEAN Stack (MongoDB, Express.js, Angular, Node.js)

- **Avantages** : Moderne et adapté aux applications single-page (SPA), non-bloquant grâce à Node.js, architecture full JavaScript.
- **Inconvénients** : Courbe d'apprentissage plus élevée, performance variable pour certaines applications embarquées.

Django (Python framework) avec PostgreSQL

- **Avantages** : Haute performance, framework robuste et sécurisé, support pour applications complexes.
- **Inconvénients** : Peut être surdimensionné pour des applications simples, nécessite des compétences en Python.

La conception, configuration, réalisation (en fonction du travail demandé)

I) Installation de Raspberry Pi

Pour installer Raspberry Pi OS avec bureau sur une carte SD et activer SSH, j'ai suivi ces étapes :

1. Téléchargement de Raspberry Pi Imager :

- J'ai téléchargé le logiciel Raspberry Pi Imager depuis le site officiel de Raspberry Pi.
- Après le téléchargement, j'ai installé le logiciel sur mon ordinateur.

2. Préparation de la carte SD :

- J'ai inséré une carte microSD dans le lecteur de carte SD de mon ordinateur.
- En utilisant Raspberry Pi Imager, j'ai sélectionné "Raspberry Pi OS (32-bit)" comme système d'exploitation.
- J'ai choisi ma carte SD comme périphérique de stockage.

- Après avoir sélectionné les options appropriées, j'ai lancé le processus d'écriture, sachant que cela effacerait toutes les



données sur la carte SD.

3. Installation sur Raspberry Pi :

- Une fois l'écriture terminée, j'ai retiré la carte SD de l'ordinateur et l'ai insérée dans mon Raspberry Pi.
- J'ai connecté le Raspberry Pi à un écran et à un clavier, puis l'ai alimenté en le branchant.
- J'ai suivi les instructions à l'écran pour effectuer la configuration initiale, choisissant le pays, la langue et me connectant au réseau Wi-Fi.

4. Activation de SSH :

- Après avoir accédé au bureau de Raspberry Pi OS, j'ai ouvert le menu en cliquant sur l'icône de la framboise en haut à gauche de l'écran.

- Dans le menu, j'ai sélectionné "Preferences" puis "Raspberry Pi Configuration".
- Dans l'onglet "Interfaces", j'ai activé SSH en cochant l'option "Enabled" et j'ai sauvégarde les paramètres en cliquant sur "OK".

5. Accès via SSH :

- J'ai noté l'adresse IP fixe attribuée à mon Raspberry Pi, configurée par Leo pour une identification facile sur le réseau.
- En utilisant cette adresse IP, j'ai pu accéder à mon Raspberry Pi via SSH depuis un autre ordinateur sur le même réseau.

II) Installation du serveur LAMP.

Étapes de l'Installation

1. Installation d'Apache

Pour installer Apache, j'ai utilisé la commande suivante :

```
sudo apt install apache2
```

2. Modification des Droits d'Accès

Pour faciliter la modification des fichiers dans le répertoire par défaut d'Apache, j'ai modifié les droits d'accès avec la commande

```
chmod -R 777 /var/www/html/
```

3. Installation de PHP et PHP-FPM

Pour installer PHP et PHP-FPM, j'ai utilisé la commande suivante

```
sudo apt install php php-fpm
```

4. Activation de PHP-FPM dans Apache

Pour permettre à Apache d'utiliser PHP-FPM, j'ai exécuté les commandes suivantes :

```
sudo a2enmod proxy_fcgi setenvif  
sudo a2enconf php8.2-fpm
```

5. Remplacement du Fichier Index

J'ai remplacé le fichier index.html par un fichier nommé index.php dans le répertoire /var/www/html/ avec le contenu suivant :

```
<?php phpinfo(); ?>
```

Cela permet de vérifier que PHP fonctionne correctement en affichant les informations de configuration de PHP.

6. Installation de MariaDB

Pour installer MariaDB et l'extension PHP pour MySQL, j'ai utilisé la commande suivante :

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

7. Configuration de MariaDB pour un Accès Distant

J'ai modifié le fichier de configuration de MariaDB pour permettre l'accès distant :

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

Dans ce fichier, j'ai modifié la ligne contenant bind-address pour qu'elle ressemble à ceci :

```
bind-address = 0.0.0.0
```

8. Installation de phpMyAdmin

Pour installer phpMyAdmin, j'ai utilisé la commande suivante :

```
sudo apt install phpmyadmin
```

9. Vérification et Configuration Finale

Pour vérifier que tout fonctionne correctement, j'ai accédé à l'URL http://<adresse_ip_serveur> et confirmé que la page PHPInfo s'affiche correctement. J'ai également vérifié l'accès à phpMyAdmin via http://<adresse_ip_serveur>/phpmyadmin.

Conclusion

L'installation du serveur LAMP a été effectuée avec succès en suivant les étapes décrites ci-dessus. Le serveur est maintenant configuré et prêt à héberger des applications web. Cependant, il est recommandé d'améliorer la sécurité en restreignant les permissions des fichiers et en configurant des mesures de sécurité supplémentaires pour MariaDB et phpMyAdmin.

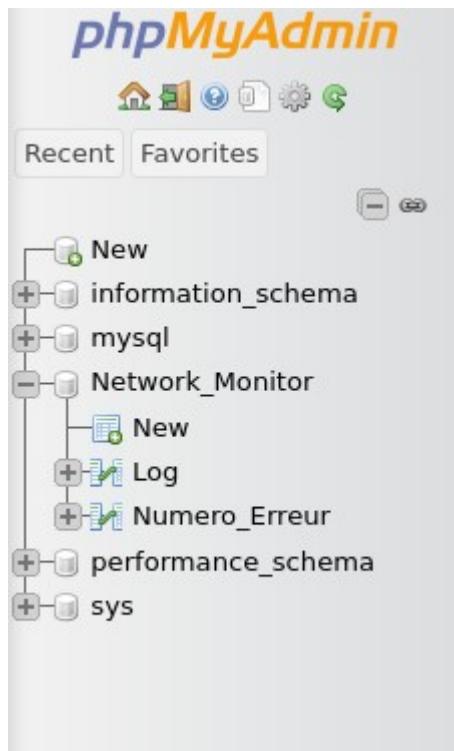
III) Modelisation base de données

Étape 1: Connexion à phpMyAdmin

1. J'ouvrirais mon navigateur Web et je naviguerais vers l'URL où phpMyAdmin est installé.
2. Je me connecterais en utilisant mes identifiants MySQL.

Étape 2: Création de la base de données

- 1.Dans le panneau de gauche, je cliquerais sur "Nouvelle base de données".
- 2.Je saisirais le nom de ma base de données, par exemple "Network_monitor", et je sélectionnerais l'encodage et le classement appropriés.
- 3.Je cliquerais sur "Créer" pour créer la base de données.



Étape 3: Sélection de la base de données

- 1.Une fois la base de données créée, je sélectionnerais "Network_monitor" dans le panneau de gauche pour m'assurer que je travaille dans la bonne base de données.

Étape 4: Création de la table "numero_erreur" en utilisant l'interface graphique

- 1.Une fois connecté à phpMyAdmin et après avoir sélectionné la base de données "Network_monitor", je cliquerais sur l'onglet "Structure".
- 2.Ensuite, je cliquerais sur le bouton "Ajouter une colonne" pour ajouter chaque colonne nécessaire à ma table.
- 3.Pour chaque colonne, je remplirais les détails suivants :
 - **Nom** : Je donnerais un nom à la colonne, par exemple "id", "Date_Heure", "val_erreur", "description_erreur".
 - **Type** : Je sélectionnerais le type de données approprié pour chaque colonne. Par exemple, "INT" pour l'ID, "DATETIME" pour la date et l'heure, "INT" pour la valeur d'erreur et "TEXT" pour la description de l'erreur.
 - **Longueur/Valeur** : Si nécessaire, je spécifierais la longueur ou la valeur maximale pour les types de données.
 - **Attributs** : J'appliquerais les attributs appropriés, comme "AUTO_INCREMENT" pour l'ID.
 - **Défaut** : Pour la colonne "Date_Heure", je définirais la valeur par défaut sur "CURRENT_TIMESTAMP".
 - **Null** : Pour les colonnes qui peuvent être nulles, je cocherais la case "Nul".

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Date_Heure	datetime			Yes	current_timestamp()			Change Drop More
2	Val_Erreur	int(11)			No	None			Change Drop More
3	Description_Erreur	text	utf8mb4_general_ci		No	None			Change Drop More

Étape 5: Répéter le processus pour la table "Log"

1.Je répéterais les étapes précédentes pour créer la table "Log" en utilisant les mêmes procédures, en ajoutant les colonnes nécessaires avec leurs types de données appropriés.

Étape 7: Terminé !

Maintenant, j'ai créé ma base de données "Network_monitor" avec les tables "numero_erreur" et "Log" en utilisant l'interface graphique de phpMyAdmin,La base de données est maintenant opérationnelle.

IV)creation IHM php

- **Actualisation automatique** : Pour maintenir les informations à jour, j'ai configuré la page pour se rafraîchir automatiquement toutes les 40 secondes en utilisant la balise meta. Cela garantit que je vois toujours les dernières données sans avoir à actualiser manuellement la page.
- **Personnalisation CSS** : J'ai appliqué une feuille de style CSS à ma page web pour personnaliser son apparence et son style, offrant ainsi une expérience visuelle cohérente et attrayante pour les utilisateurs.

```
<!DOCTYPE html>
<html>
<head>
    <title>Network Monitor</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
    <meta http-equiv="refresh" content="40">
</head>
<body>
<?php
```

- **Paramètres de connexion** : J'ai stocké les informations de connexion à la base de données, telles que le serveur, le nom d'utilisateur, le mot de passe et le nom de la base de données,

dans des variables. Cette approche rend la gestion des informations de connexion plus flexible et facilite les mises à jour éventuelles.

- **Connexion à la base de données** : J'ai établi une connexion à la base de données en utilisant les paramètres de connexion que j'ai définis précédemment. Ensuite, je vérifie si la connexion a réussi à l'aide d'une instruction conditionnelle.

```
//Parametres de Connexion
$serveur = "localhost";
$utilisateur = "admin";
$motdepasse = "admin5775";
$basededonnees = "Network_Monitor";
$connexion = new mysqli($serveur, $utilisateur, $motdepasse, $basededonnees);

if ($connexion->connect_error) {
    die("La connexion a échoué : " . $connexion->connect_error);
}
```

- **Définition des variables pour la détection de l'état du réseau** : Si la connexion à la base de données est établie avec succès, je définis une variable `sql_last_error` pour contenir la requête SQL qui récupère la dernière erreur enregistrée. Ensuite, j'exécute cette requête et je stocke le résultat dans une variable `last_error_row` en utilisant la fonction `fetch_assoc()`. Cette fonction récupère les données sous forme de tableau associatif. À partir de `last_error_row`, j'extrais et assigne les valeurs à des variables spécifiques : `last_error`, `error_description`, et `error_datetime`. Ces variables me permettent de déterminer l'état actuel du réseau. Si `last_error` a une valeur de 0, j'affiche que tout est OK. Sinon, j'affiche un message en fonction du code d'erreur pour informer les utilisateurs sur l'état du réseau.

```

$sql_last_error = "SELECT Val_Erreur, Description_Erreur, Date_Heure FROM `Log` ORDER BY `Date_Heure` DESC LIMIT 1"; //Requete qui Definit la Derniere Erreur
$result_last_error = $connexion->query($sql_last_error);
$last_error = $result_last_error->fetch();
$error_description = $last_error["Description_Erreur"];
$error_datetime = $last_error["Date_Heure"];

```

- **Affichage de l'état du réseau :** J'évalue l'état actuel du réseau en fonction de la dernière erreur enregistrée. Si aucune erreur n'est survenue (la valeur est 0), j'affiche un message indiquant que tout est OK. Sinon, j'attribue un message en fonction du code d'erreur pour informer les utilisateurs sur l'état actuel du réseau.

```

if ($last_error == 0)
{
    echo "<h1>État : Ok</h1>";
}

elseif ($last_error >= 1 && $last_error <= 6) // Permet de definir l'etat du reseau
{
    $error_messages = [
        1 => "ok (NTP)",
        2 => "Moyen (Qualité du Réseau)",
        3 => "Critique (Switch)",
        4 => "Critique (DHCP)",
        5 => "Critique (DNS)",
        6 => "Critique (Passerelle)"
    ];
    echo "<h1>État : ".$error_messages[$last_error]."</h1>";
}

else {
    echo "<h1>État : Inconnu</h1>";
}

```

État : Critique (Passerelle)

- **Récupération des derniers logs :** Si la connexion à la base de données est établie avec succès, j'exécute une requête SQL

pour récupérer les 10 derniers logs de la table "log" dans la base de données "network_monitor".

- **Affichage des logs** : Si des logs sont récupérés avec succès de la base de données, je les affiche dans un tableau pour que les utilisateurs puissent les consulter. Cela fournit une traçabilité des événements récents et permet une analyse ultérieure si nécessaire.

```
$sql_logs = "SELECT * FROM `Log` ORDER BY `Date_Heure` DESC LIMIT 10 "; //Requete SQL qui recupere les 10 dernieres données de la base de données
$result_logs = $connexion->query($sql_logs);

if ($result_logs->num_rows > 0)
{
    echo "<table>";
    echo "<tr><th>Date & Heure</th><th>Valeur Erreur</th><th>Description</th></tr>";
    while ($row = $result_logs->fetch_assoc()) {
        echo "<tr><td>".$row["Date_Heure"]."</td><td>".$row["Val_Erreur"]."</td><td>".$row["Description_Erreur"]."</td></tr>";
    }
    echo "</table>";
```

Date & Heure	Valeur Erreur	Description
2024-04-11 09:39:52	6	Apr 3 11:47:42 pfSense dpinger[33157]: WAN_DHCP 172.16.180.189: sendto error: 65
2024-04-11 09:38:58	5	Problèmes de résolutions de noms (DNS)
2024-04-11 09:33:41	4	Apr 11 09:33:28 pfSense dhcpd[57175]: DHCPDISCOVER from 00:0c:29:fa:53:35 via rl0: network 192.168.1.0/24: no free leases
2024-04-11 09:25:32	3	Problèmes matériels liés au switch
2024-04-11 09:25:21	2	Débit inférieur à 3Mb/s
2024-04-11 09:25:09	1	Apr 11 09:18:02 pfSense ntpd[89112]: kernel reports TIME_ERROR: 0x2007: PPS Frequency Sync wanted but no PPS; PPS Time Sync wanted but no PPS signal
2024-04-11 09:24:45	0	Aucun problème détecté

- **Affichage des détails de l'erreur** : Si la dernière erreur enregistrée est différente de 0, j'affiche des détails supplémentaires sur cette erreur pour aider les utilisateurs à comprendre la nature du problème. Cela peut inclure des informations telles que la description de l'erreur, la date et l'heure de son occurrence, etc.

```
if ($last_error != 0) //Details sur la derniere Erreur
{
    echo "<div class='error-details'>";
    echo "<h2>Détails de la dernière erreur :</h2>";
    echo "<p>Date & Heure : " . $error_datetime . "</p>";
    echo "<p>Valeur Erreur : " . $last_error . "</p>";
    echo "<p>Description : " . $error_description . "</p>";
    echo "</div>";
}
```

Détails de la dernière erreur :

Date & Heure : 2024-04-11 09:39:52

Valeur Erreur : 6

Description : Apr 3 11:47:42 pfSense dpinger[33157]: WAN_DHCP 172.16.180.189: sendto error: 65

- Déconnexion sécurisée : Une fois toutes les opérations terminées, je m'assure de me déconnecter de la base de données pour des raisons de sécurité et pour libérer les ressources.

```
$connexion->close();
?>
</body>
</html>
```

Le cahier de recette

Fiche de test unitaire

- | |
|--|
| <input type="checkbox"/> Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale |
| <input type="checkbox"/> Test de faisabilité |
| <input checked="" type="checkbox"/> Test unitaire |
| <input type="checkbox"/> Test d'intégration |

Nom de l'étudiant : Beschon matt

Date : 04/06/24

Objectif du test : Vérifier le bon fonctionnement de la page web ainsi que la base de données

Conditions de réalisation

- Besoins matériels
Raspberry Pi
- Besoins logiciels (nom et version)
Base de données
- Noms des fichiers utilisés
- ...

Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Brancher un câble RJ45 au raspberry puis le mettre sous tension grâce au câble d'alimentation. Depuis un ordinateur qui a un accès à l'intranet aller sur un navigateur web et dans la zone URL taper : 192.168.1.246	192.168.1.246	Le navigateur web affiche bien les informations suivantes : -l'état du réseau actuelle -les 10 derniers logs reçus sur le tableau log -Des détails sur le dernier log reçu	OK
2	Brancher un câble RJ45 au raspberry puis le mettre sous tension grâce au câble d'alimentation. Depuis un ordinateur qui a un accès à l'intranet aller sur un navigateur web et dans la zone URL taper : 192.168.1.246/ Phpmyadmin	192.168.1.246/phpmyadmin	Le Navigateur web affiche bien une page de connexion	OK

Fiche de Recette globale

Fiche de recette gloable

- | |
|---|
| <input checked="" type="checkbox"/> Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale |
| <input type="checkbox"/> Test de faisabilité |
| <input type="checkbox"/> Test unitaire |
| <input type="checkbox"/> Test d'intégration |

Nom de l'étudiant : Beschon Matt

Date : 24/05/24

objectif(s) de la recette

- Etablir le bon fonctionnement de ma partie du projet
- Besoins matériels
Raspberry Pi
Ordinateur avec acces a l'intranet
- Besoins logiciels (nom et version)
Base de données
IHM(PHP)
- Pré-requis
Analyser les tâches demandées

ID	Action/Méthode	Comportement / Résultat attendu	Status OK / NOK
1	Installation de l'OS de l'ordinateur embarqué	L'ordinateur embarqué est fonctionnel et le ssh est l'I2c et bien activés	ok
2	Installation du serveur Lamp.	Le serveur Lamp est bien installé sur l'ordinateur embarqué	ok
3	Modélisation de la base de données.	La Base de données est bien modélisé (1 base de données Network Monitor 2 tables Log et Numero_erreur chaque table possède 3 champs Date Heure, Val Erreur et Description Erreur	ok
4	Création et mise au point de l'application IHM.	L'Ihm affiche bien l'état du réseau actuel, elle affiche bien les 10 derniers logs reçus dans la table Log et elle affiche aussi des détails sur la dernière erreur reçue	ok

Les problèmes rencontrés

Lors de la collaboration sur le projet, nous avons rencontré une difficulté nécessitant le téléchargement de bibliothèques supplémentaires telles que `libmysql`. Pour remédier à cette situation, nous avons dû exécuter la commande suivante : `apt install mysql-server`. De même, nous avons également été confronté à un problème similaire avec la bibliothèque `libssh`. Pour résoudre cette problématique, j'ai dû saisir la commande : `apt install libssh`.

Conclusion personnelle

Conclusion Personnelle

Ce projet m'a permis de renforcer mes compétences en systèmes embarqués et en développement d'applications web. J'ai été responsable de l'installation et de la configuration du système d'exploitation sur le Raspberry Pi, ainsi que de la mise en place d'un serveur LAMP, assurant ainsi un environnement stable et performant pour le développement de l'application.

La modélisation de la base de données et la création de l'interface homme-machine ont été des étapes cruciales, me permettant de concevoir des solutions répondant aux besoins du projet. Les tests de faisabilité et le développement continu de l'application ont garanti sa qualité et sa fiabilité.

Rédiger une documentation complète et des documents de recette a facilité l'utilisation et l'adoption du produit final, tout en assurant que toutes les exigences initiales étaient respectées.

Le choix du Raspberry Pi et de PHP s'est avéré judicieux en termes de coût, de performance, et de support communautaire.

Malgré quelques défis techniques, notamment la gestion des dépendances logicielles, j'ai pu les surmonter et en tirer des leçons précieuses.

En conclusion, ce projet a été une expérience enrichissante qui a consolidé ma capacité à gérer des projets complexes et à développer des solutions robustes et efficaces. Je suis satisfait des résultats et confiant dans la qualité du travail accompli.

1.11Conclusion du Projet :

En conclusion, le développement d'un système de surveillance du réseau informatique avec une colonne de signalisation et une IHM web est une entreprise complexe mais essentielle pour garantir la stabilité et la sécurité des infrastructures informatiques. Ce projet nécessite une planification minutieuse, une conception robuste, un développement soigné et des tests rigoureux pour assurer son efficacité et sa fiabilité.

En suivant les étapes décrites dans le plan, vous pourrez créer un système qui répond aux besoins de surveillance en temps réel, d'alerte rapide en cas de problèmes et de visualisation claire des données de réseau. De plus, la mise en place d'une documentation détaillée et d'un support technique solide assurera une utilisation fluide et une maintenance efficace à long terme.

En fin de compte, ce projet contribuera à renforcer la sécurité et la performance du réseau informatique, offrant aux utilisateurs une tranquillité d'esprit et une capacité accrue à réagir rapidement aux incidents potentiels.