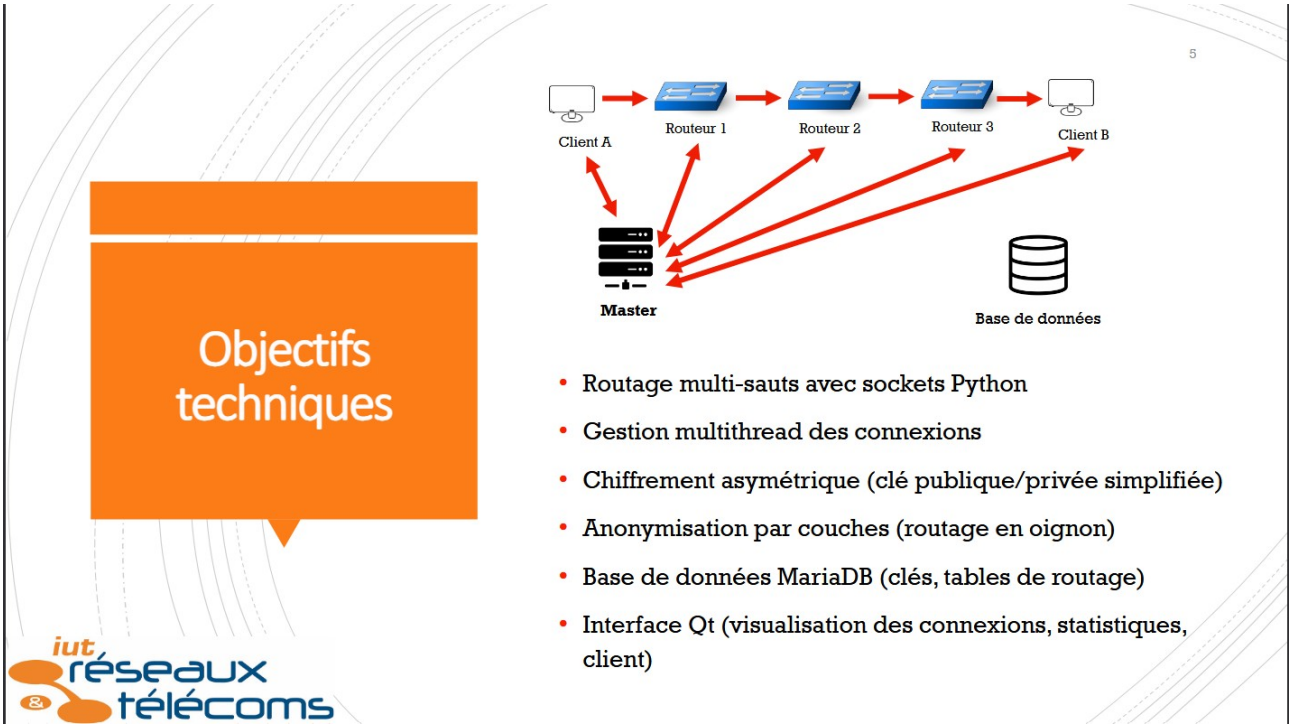
Documentation de réponse au cahier des charges

Éléments implémentés et non implémentés :



Plusieurs objectifs techniques nous étaient demandés de réaliser et ils ont tous été implémentés :

Notre projet est capable de router un message a travers différents « programme » grâce aux socket python.

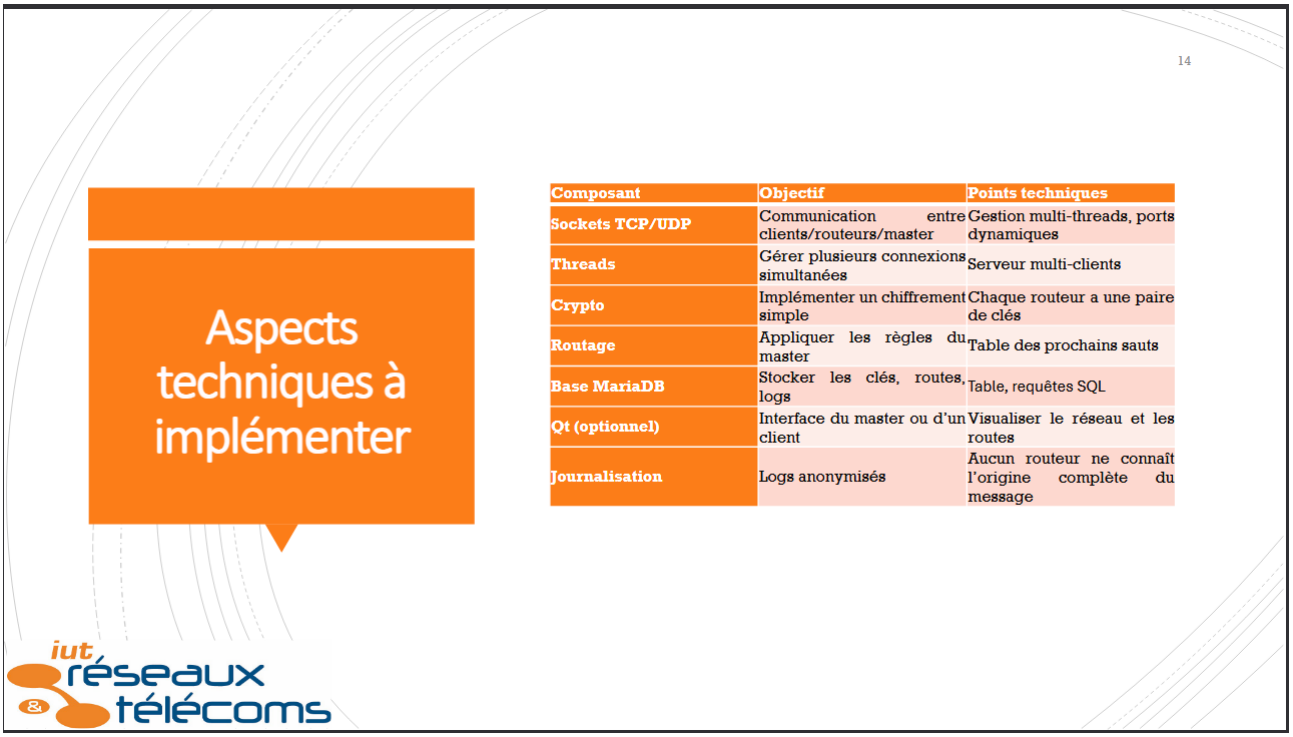
Il y a aussi une gestion multithread des connexions car dans le serveur master il y a une boucle infinie « while true » qui attend les connexions entrantes et quand un client ou un routeur se connecte il lance un thread dédié à cette connexion et peut retourner dans sa boucle d’écoute.

Le chiffrement asymétrique est aussi présent nous avons codé une version simplifiée du chiffrement RSA.

L'anonymisation par couche se fait aussi grâce à la fonction « couche_onion » qui avant d'envoyer le message, il choisit un chemin et un nombre de routeur aléatoire, il va ensuite chiffrer le message commençant par le dernier routeur puis va rechiffrer le message couche par couche en remontant dans les routeurs.

Une base de données est bien implantées c'est elle qui gère stock tout les nœuds présent sur le réseau mais aussi leur information importante comme leur adresse IP, leur port et pour les routeur leur clés.

Et enfin une interface graphique en PyQt6 est présente pour le serveur master et les clients.



Dans notre projet tout les aspects techniques Socket TCP/UDP, threads, Crypto, Routage, Base MariaDB et PyQt6 sont implémentés, il n'y a que la journalisation qui n'est pas implémenté, le serveur master sait qui est actif sur le réseaux mais il ne sait pas qui envoie des messages à qui (ce qui est en soi plus sécurisé). Mais par contre le

point technique est bien respecté aucun routeur ne connaît l'origine complète du message.

Structure du code, modules, protocole, API :

Structure du code et modules :

Notre projet comprend 4 modules principaux codés en python :

1. Le serveur master :

-il sert d'annuaire, distribue les clés des routeurs au client, et vérifie si les nœuds présents sur le réseau sont bien fonctionnel (fonction ping)

-Utilisation de la bibliothèque « mysql.connector » , « socket », « threading », « time », « sys » et « PyQt6 » et il appelle des fonctions du fichier « cryptage.py »

2. Le routeur :

-il sert à la bonne transmission des messages entre les clients mais c'est aussi grâce a lui que l'anonymisation des messages peut se faire. Sans routeur impossible pour le client de faire un message à couches.

-Utilisation de la bibliothèque « socket », « threading », « sys », « time » et il appelle des fonctions du fichier « cryptage.py »

3. Le client :

-il sert à envoyer des messages anonymisés à d'autres client, c'est lui qui crée les couches de cryptage

-Utilisation de la bibliothèque « socket », « threading », « sys », « time », « PyQt6 » et il appelle des fonction du fichier « cryptage.py »

4. Le programme de cryptage :

-il sert à fournir aux autres programmes des primitives mathématiques

-Utilisation de la bibliothèque « random » et de la fonction « isprime » de la bibliothèque « sympy »

protocole et API :

Dans notre projet nous utilisons des sockets TCP : fiable et sécurisé, il assure l'arrivée des paquets dans le bon ordre et sans casse, ce qui est très important quand on utilise des programme de chiffrage car une seule erreur de caractère empêcherais tout le décryptage.
Grace a ces socket les clients et les routeurs envoie un message contenant plusieurs information qui sont séparer grace a la méthode « split() »
le format du message est <Commande>|<Nom>|<Port>|<clé N>|clé E>
il existe trois commandes qui sont reconnues par le serveur :

- « ENRE » qui est une demande d'enregistrement

- « PING » qui est envoyé toutes les 2 secondes pour vérifier que le nœud est toujours actif

- « LISTE_CLIENT » qui envoie la liste des clients et toutes leur info enregistrées

- « LISTE_ROUTEUR » qui envoie la liste des routeurs et toutes leur info enregistrées

Une seul fonction utilise une socket UDP et c'est la fonction « adresse_ip » de tout les programmes : elle simule une connexion UDP vers google (8.8.8.8) qui force le système d'exploitation à déterminer quelle interface réseau (et donc quelle IP) serait utilisée pour sortir.

Description de l'algorithme de chiffrage (forces et faiblesses) :

L'algorithme utilisé dans notre projet est RSA (Rivest-Shamir-Adleman), un algorithme asymétrique.

Voici une description de son fonctionnement :

Chaque routeur génère une clé publique (n, e) et une clé privée (n, d) sachant que n est le produit de deux grands nombres premiers p et q

puis vient le chiffrement le message m est transformé en un nombre via le code ASCII , puis puisque le RSA ne peut chiffrer que des données plus petites que la taille de la clé, nous découpons le message en blocs.

Enfin le déchiffrement, le routeur utilise sa clé privée d pour retrouver le message grâce au modulo de n.

Forces :

Comme chaque couche de l'oignon est chiffrée avec une clé différente, le routeur 1 ne peut mathématiquement pas lire le message destiné au routeur 3 ou au client.

Aussi cette méthode de chiffrage et déchiffrage avec clé est très pratique car ces clés sont très faciles à envoyer aux différents client.

Faiblesses :

Si nous devons envoyer de long message, le RSA est très très long et la taille des paquets peut devenir extrêmement volumineux c'est pour ça que le client ne peut chiffrer qu'avec 5 routeurs.

Puisque notre version simplifiée de RSA ne comprend pas de padding ou de « bruit », un observateur pourrait très facilement trouver un paquet à cause de leur taille grâce à Wireshark et le suivre.

Rapport de projet avec gestion du projet :

Les différentes fonctionnalités énoncées dans le rapport de projet ont bien toutes été faites sauf les log anonymisés, le temps imparti pour chaque tache annoncé dans le diagramme a été un peu petit peu trop court car nous avons dû un peu étaler la programmation au début des vacances, mais sinon les délai pour la rédaction de la documentation ont été respectés.