iut réseaux & télécoms

Conception d'une architecture distribuée avec routage en oignon

Projet d'implémentation d'un système anonyme client-serveur

R3.09 | Programmation évènementielle

SAÉ 3.02 | Développer des applications communicantes

Frédéric DROUHIN - BUT 2 - 2025-2026

Créer des outils et applications informatiques pour les R&T

- AC23.01 | Automatiser l'administration système avec des scripts
- AC23.02 | Développer une application à partir d'un cahier des charges donné, pour le Web ou les périphériques mobiles
- AC23.03 | Utiliser un protocole réseau pour programmer une application client/serveur
- AC23.04 | Installer, administrer un système de gestion de données
- AC23.05 | Accéder à un ensemble de données depuis une application et/ou un site web

- CE3.01 | en étant à l'écoute des besoins du client
- CE3.02 | en documentant le travail réalisé
- CE3.03 | en utilisant les outils numériques à bon escient
- CE3.04 | en choisissant les outils de développement adaptés
- CE3.05 | en intégrant les problématiques de sécurité

Introduction au projet

- **Objectif principal :**
  Concevoir et implémenter un système de communication anonyme entre plusieurs clients à l'aide de routeurs virtuels.
- **Principe clé :**
  Anonymat via le **routage en oignon** (chaînes de routeurs qui ne connaissent que leur voisin direct).
- **Pourquoi ?**
  Garantir l'anonymat des communications sur un réseau distribué.

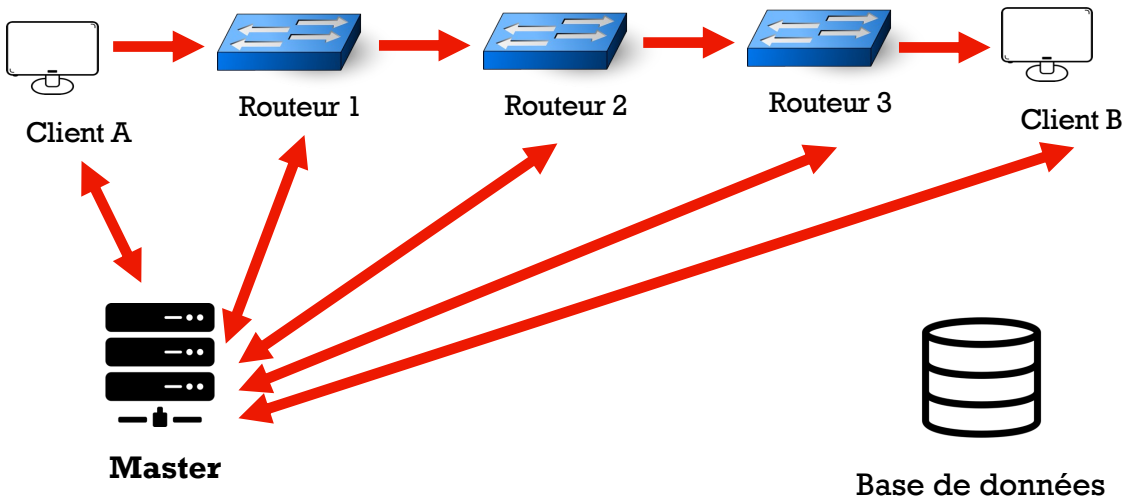
Objectifs techniques et Intérêt pédagogique

- Implémenter un système client-serveur multi-routeurs utilisant les sockets Python sur au minimum 2 machines.
- Gérer la communication simultanée par threads.
- Concevoir un chiffrement asymétrique simplifié (clé publique/privée) et un système de couches de chiffrement.
- Développer une interface Qt pour visualiser les connexions et la topologie et le client
- Stocker les informations de routage et de clé dans une base de données MariaDB.

- **Objectif général :** concevoir une architecture distribuée composée de :
  - Plusieurs **routeurs virtuels** interconnectés ;
  - Un **serveur maître** (superviseur, générateur de règles et de routes) ;
  - Plusieurs **clients** souhaitant échanger des messages **de manière anonyme**, via plusieurs routeurs.

- **Principe** : chaque message traverse une **chaîne de routeurs**, où chaque routeur ne connaît **que son voisin précédant et suivant**. Les données sont **chiffrées par couches successives**, comme dans Tor (The Onion Router).

Objectifs techniques



- Routage multi-sauts avec sockets Python
- Gestion multithread des connexions
- Chiffrement asymétrique (clé publique/privée simplifiée)
- Anonymisation par couches (routage en oignon)
- Base de données MariaDB (clés, tables de routage)
- Interface Qt (visualisation des connexions, statistiques, client)

iut réseaux & télécoms

Durée et gestion de projet

- FA
  - Durée :
    - 15,75h encadrés de R3.09
    - 8,75h encadrés de SAE
    - 18,5h d'autonomie
  - Groupe (au choix)
    - 1 étudiant : notation sur 30
    - 2, 3, 4 étudiants : notation sur 20
- FI
  - Durée
    - 15,75h encadrés de R3.09
    - 10,5h encadrés de SAE
    - 39,5h d'autonomie
  - Groupe (aux choix)
    - 1 étudiant : notation sur 25
    - 2 étudiants : notation sur 20

Date de rendu :

Gestion de projet : 20 novembre à 23:59

Projet final : 31 décembre à 23:59

Gestion de projet

- Chaque étudiant ou groupe devra planifier et suivre le déroulement de son projet :
  - Définition des tâches et répartition claire des rôles
  - Utilisation d'un outil de suivi (Trello, GitHub Projects, etc.)
  - Rédaction d'un planning prévisionnel
  - Mise à jour du dépôt Git à chaque étape
  - Production d'un journal de bord (commit messages explicites à extraire et à rendre dans le rapport de planning)

Rendu d'un planning : 20 novembre à 23:59

Architecture logique

- **Master** : définit la topologie et distribue les clés de chiffrement à chaque routeur.
- **Clients** : construisent les messages "en oignon" (plusieurs couches de chiffrement).
- **Routeurs** : déchiffrent **une seule couche**, découvrent l'adresse du **prochain saut**, et transmettent le reste du message.
- **Base de données** : stocke les tables de routage et les clés de chiffrement.

Destination	Next Hop	Interface	RuleID
clientA	routeurB	Eth0	1
ClientB	routeurC	Eth1	2

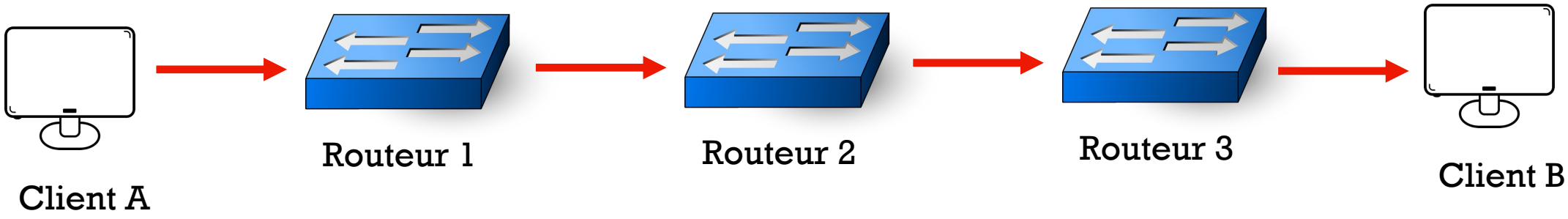
Tableau 1 exemple de table de routage

Architecture logique

- Master :
  - Le **master** génère une liste de routeurs virtuels, leurs adresses et une clé (symétrique ou RSA).
  - Il distribue les **clés publiques** des routeurs aux clients.
- Routeur :
  - Chaque routeur enregistre sa clé privée localement.
  - Le routeur reçoit un message, déchiffre et envoi au suivant
- Client :
  - Le client choisit un **chemin aléatoire** (ex. 3 routeurs).
  - Il chiffre le message **en plusieurs couches** :
  - Couche 3 : chiffrée avec la clé du Routeur 3 (contient message final + destinataire).
  - Couche 2 : chiffrée avec la clé du Routeur 2 (contient adresse du Routeur 3 + couche 3).
  - Couche 1 : chiffrée avec la clé du Routeur 1 (contient adresse du Routeur 2 + couche 2).

→ Il obtient un "oignon" qu'il envoie au premier routeur.

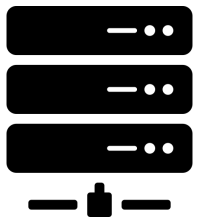
Un exemple de scénario minimal



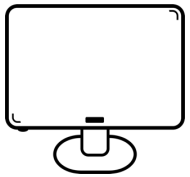
Initialisation

Routeur

Envoi sa clé public



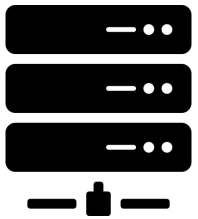
Master



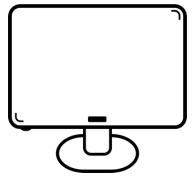
Le client A veut envoyer un message à B

Client

Demande les clés
Publics des routeurs
Nombre de routeurs



Master



Client A

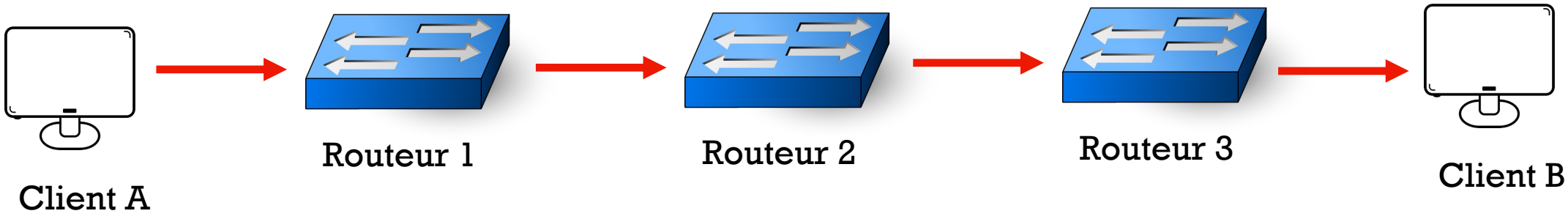
Le client calcule :

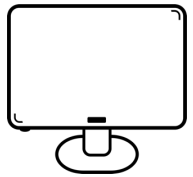
- C3 = Enc_{K3}(Dest(Client B) ET M)
- C2 = Enc_{K2}(Dest(R3) ET C3
- C1 = Enc_{K1}(Dest(R2) ET C2

Le client envoie C1 à R1

RX = routeur X
M = message
CX = message X
KX = clé du routeur X
Enc_{KX} = chiffrement avec clé X

iut réseaux & télécoms





Client A

Le client calcule :

- C3 = Enc_{K3}(Dest(Client B) ET M)
- C2 = Enc_{K2}(Dest(R3) ET C3
- C1 = Enc_{K1}(Dest(R2) ET C2

Le client envoie C1 à R1

Routeur 1

Déchiffre le message C1

- Adresse R2
- Message C2

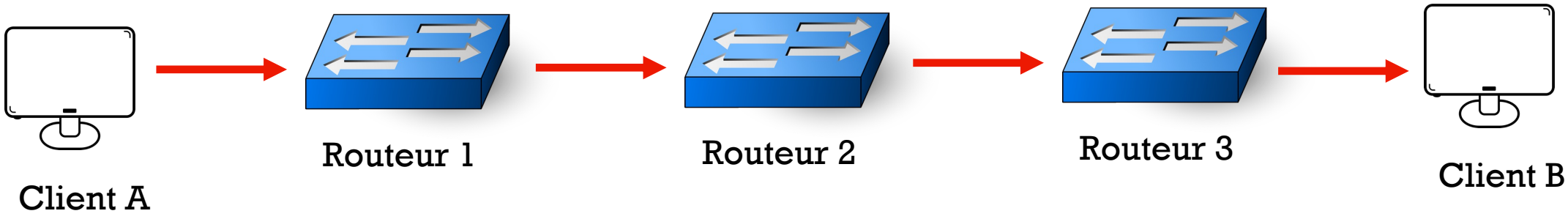
R1 envoie C2 à R2

Routeur 2

Déchiffre le message C2

- Adresse R3
- Message C3

R2 envoie C3 à R3



Routeur 2

Déchiffre le message C2

- Adresse R3
- Message C3

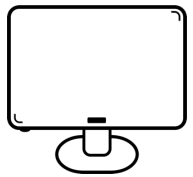
R2 envoie C3 à R3

Routeur 3

Déchiffre le message C3

- Adresse Client B
- Message (M)

R3 envoie M à Client B



Client B

Client B reçoit le message

iut réseaux & télécoms

Aspects techniques à implémenter

Composant	Objectif	Points techniques
Sockets TCP/UDP	Communication entre clients/routeurs/master	Gestion multi-threads, ports dynamiques
Threads	Gérer plusieurs connexions simultanées	Serveur multi-clients
Crypto	Implémenter un chiffrement simple	Chaque routeur a une paire de clés
Routage	Appliquer les règles du master	Table des prochains sauts
Base MariaDB	Stocker les clés, routes, logs	Table, requêtes SQL
Qt (optionnel)	Interface du master ou d'un client	Visualiser le réseau et les routes
Journalisation	Logs anonymisés	Aucun routeur ne connaît l'origine complète du message

Contraintes de développement

- Langage : Python
- Bibliothèques :
  - Socket, threading, MariaDB, PyQt5/PyQt6
  - à l'exception de tout autre librairie (par exemple pas de librairie de cryptographie)
- Développement intégralement manuel (aucun code généré automatiquement)
- Structure de code claire et commentée
- Lien GitHub obligatoire pour le dépôt final
- L'environnement de test montrant l'usage de plusieurs machines ou VM.
- Le client et le master doivent être des interfaces graphiques :
  - Client : connexion et envoi de message (non bloquant)
  - Master : visualisation, logs, distribution des clés (éventuellement démarrage des routeurs)

Livrables

- Code source complet au travers d'un lien git
- Documentation de réponse au cahier des charges
  - Eléments implémentés et non implémentés
  - Structure du code, modules, protocole, API
  - Description de l'algorithme de chiffrage (forces et faiblesses)
  - Rapport de projet avec gestion du projet
- Documentation d'installation et d'utilisation
  - Etapes pour installer et lancer le projet
  - Guide d'utilisation des interfaces et commandes
- Vidéo de démonstration (5–10 minutes)
  - L'environnement de test (machines, VM)
  - Le lancement du système complet
  - Le fonctionnement des clients, routeurs et du master
  - Une démonstration de l'anonymisation des flux (couches de chiffrement) avec par exemple une trame échangée montrant l'anonymisation
  - Un commentaire oral ou sous-titré expliquant les choix techniques
- Travail de groupe : fiche individuelle expliquant son rôle (introduction) dans le projet en décrivant de manière détaillée les éléments qu'il a travaillé (corps) et une conclusion sur son apprentissage en analyse sa compétence au travail sur du référentiel de compétences
- En fonction du planning une présentation de certaines parties de code peut être demandé à un ou plusieurs étudiants.

Evaluation

Critère	Points max
Fonctionnalités techniques (routage, chiffrement, anonymisation)	8
Documentation (installation, utilisateur, programmeur)	4
Gestion de projet (organisation, Git, suivi, journal de bord)	3
Vidéo de démonstration	2
Présentation, rigueur et respect des consignes	3

Critère	Insuffisant	Satisfaisant	Très bon
Routage multi-sauts	Non fonctionnel ou un seul saut	Chaînage basique entre routeurs	Routage complet et fiable à plusieurs sauts
Chiffrement asymétrique	Chiffrement absent ou incorrect	Chiffrement fonctionnel simple	Couches multiples correctement gérées
Anonymisation des flux	Les routeurs connaissent les origines	Anonymisation partielle	Anonymisation complète respectée
Documentation	Incomplète ou non structurée	Claire mais partielle	Complète, claire et illustrée
Gestion de projet	Peu de suivi / commits rares	Suivi régulier avec quelques lacunes	Organisation rigoureuse et continue
Vidéo de démonstration	Absente ou peu claire	Présente mais incomplète	Démonstration claire et commentée

iut réseaux & télécoms

Evaluation

- Je lis votre documentation de gestion de projet
- Je lis votre documentation de réponse et les fiches individuelles
- Je télécharge votre code depuis
  - Git partage avec fdrouhin-uha
- Je monte 3 VM - 2 linux et 1 Windows
- Je lis votre documentation d'installation
  - J'installe les outils indiqués (dont BDD)
  - J'installe le Master, 3 routeurs, 2 clients
- Je lis votre document d'utilisation
  - Je démarre le master, les routeurs et les clients avec les commandes fournies
  - J'envoi un message puis le client A et un message depuis le client B
- Je regarde le code et je regarde la documentation
- Je note selon la grille précédente
- Je prends l'évaluation des étudiants et j'ajuste la note
- Je convoque un ou plusieurs étudiants pour description du code