école supérieure de génie informatique

23/03/2025

Projet ABR_Routage

Rapport de projet

NANDILLON Maxence

3ESGI, NANTES

TABLE OF CONTENTS

1. INTRODUCTION

1.1 CONTEXTE DU PROJET

Le projet **Optimisation du routage des paquets (ARB_routage)** a été initié dans le cadre du cours d'**Algorithmique Avancée avec Python** à l'ESGI. Ce projet vise à modéliser une architecture réseau en utilisant des **arbres binaires** pour structurer une table de routage. L'objectif est d'optimiser le routage des paquets en trouvant le chemin le plus efficace entre les routeurs, en utilisant des algorithmes de parcours d'arbres tels que le **parcours en profondeur (DFS)** et le **parcours en largeur (BFS)**.

1.2 OBJECTIFS DU PROJET

Les objectifs du projet sont les suivants :

- **Représenter un réseau sous forme d'arbre binaire** : Chaque nœud de l'arbre représente un routeur, et les branches représentent les connexions entre les routeurs.
- **Implémenter un algorithme de routage** : Trouver le chemin optimal vers une destination en utilisant des parcours d'arbres.
- **Simuler l'envoi de paquets** : Simuler l'envoi de paquets en suivant le chemin optimal calculé.
- **Extension avec un arbre binaire de recherche (ABR)** : Optimiser la recherche des routes et leur coût en utilisant un ABR.

1.3 ENVIRONNEMENT DE TRAVAIL

Ce projet a été réalisé en solo dans le cadre de mes études à l'ESGI en troisième année, option **Systèmes, Réseaux et Cloud (SRC)**. Le projet s'inscrit dans le cursus de préparation au diplôme de **Chef de Projet Logiciel et Réseaux**, ce qui implique une attention particulière à la gestion de projet et à la structuration du code.

2. GESTION DE PROJET

2.1 REPARTITION DES TACHES

Étant donné que le projet a été réalisé en solo, la répartition des tâches a été gérée de manière autonome. J'ai utilisé **Miro** pour organiser les tâches, créer des diagrammes de flux, des UML, et des documents d'utilisation et de références. Cela m'a permis de visualiser clairement les étapes du projet et de suivre l'avancement.

2.2 OUTILS DE GESTION DE PROJET

- **Git et GitHub :** Utilisé pour le versioning du code et GitHub pour le stockage et la collaboration. J'ai également mis en place des Hooks, des GitHub Actions pour vérifier la qualité du code
- **Miro** : Utilisé pour la gestion des tâches, la création de diagrammes de flux, d'UML, et de documents d'utilisation. J'ai également utilisé Miro pour créer une **Story Map** incluant la présentation générale, les informations sur le projet, et les points d'attention (Lien du Miro).
- **GitHub** : Pour le versioning du code et le stockage du projet.
- **Copilote** : Pour ajouter des descriptions et des commentaires clairs dans le code, afin de le rendre compréhensible par tous.

Exemple de ce que vous pouvez retrouver dans le Miro : **Lien**

ALGORITHMIQUE AVANCE AVEC PYTHON

Présentation Générale

Ce mini-projet a été conçus pour nous permettre de comprendre et d'implémenter des structures d'arbres binaires en Python. Ils impliquent des concepts d'algorithmie avancée, notamment la manipulation des arbres, l'optimisation de parcours et l'application à des contextes pratiques comme la compilation et les réseaux. Une extension est ajoutée pour inclure la notion d'arbre binaire de recherche (ABR), permettant d'optimiser certaines opérations de recherche et d'organisation des données

Consignes Générales

- Chaque projet doit être réalisé en binôme ou individuellement.
- L'utilisation de bibliothèques externes est interdite : tout doit être codé à la main.
- Une bonne structuration du code est attendue : des fonctions claires et bien commentées.
- Un rapport expliquant la logique adoptée doit accompagner le code source.

Modalités d'Évaluation

- Date limite : 23/03/2025 à 23h59 sur MyGes

Instructions : inscrivez le contexte du cahier des charges

Exigences

L'utilisation de bibliothèques externes est interdite : tout doit être codé à la main.	Une bonne structuration du code est attendue : des fonctions claires et bien commentées.
Un rapport expliquant la logique adoptée doit accompagner le code source.	Modéliser une architecture réseau
Date limite : 23/03/2025 à 23h59 sur MyGes	Optimiser le routage des paquets en utilisant des arbres binaires pour structurer une table de routage.

Instructions : inscrivez les exigences.

Points d'attentions

Compliquer d'afficher le chemin le plus cours en fonction du temps d'exécution de la méthode de recherche la plus rapide, sans la lib time	Utilisation de la méthode BFS pour l'affichage du chemin de la racine vers le nœud "réseau" rechercher
Ajout d'un fichier cache pour stocker les recherches et optimiser la recherche de l'utilisateur	Utilisation de deux paths pour la recherche avec les trois méthodes et pour le chemin de la racine jusqu'au nœud recherché
l'arbre binaire représente le réseau informatique d'une entreprise.	On considère qu'il n'y a aucune redondance, et que le ficher cache a été réaliser afin d'apprendre à extraire des données, les formater et de les réutiliser.

Instructions : inscrivez les points d'actions à prendre en compte.

Tâches

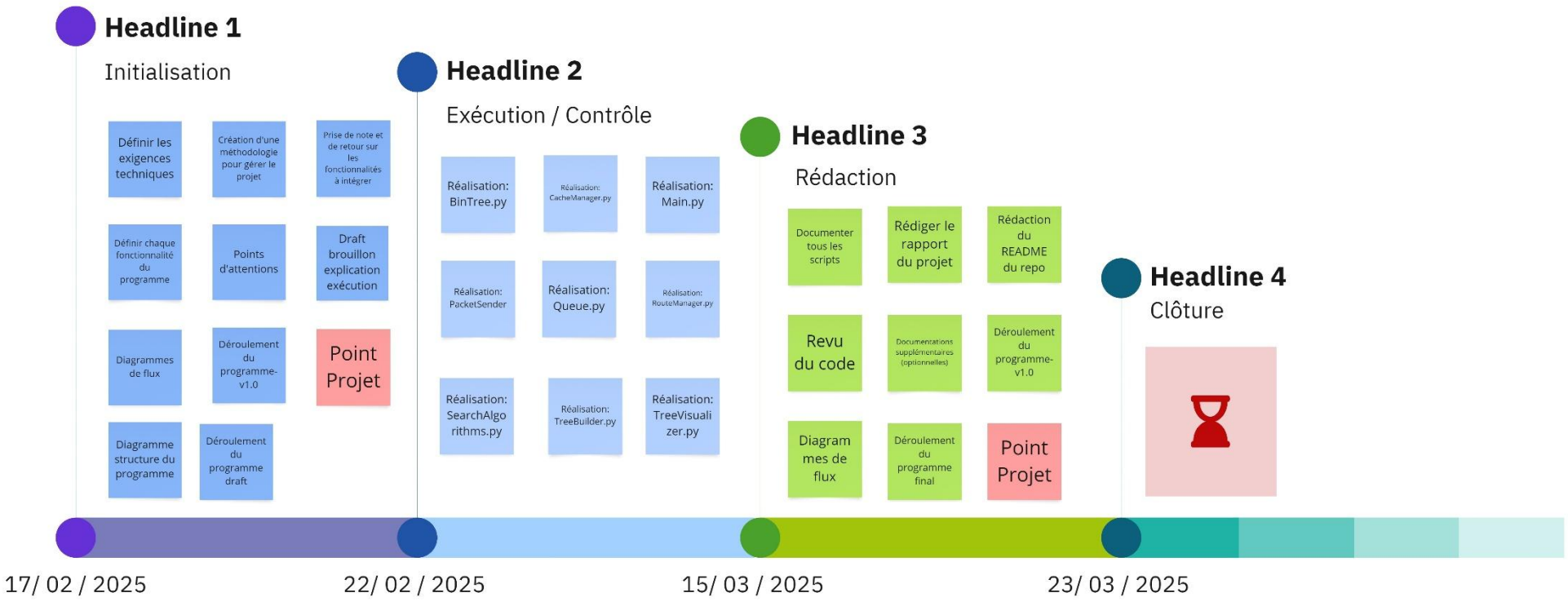
Instructions

Étape 1 : Ajouter les modules à créer sur les autocollants jaunes.
Étape 2 : Ajouter le sujet de chaque module sur les autocollants gris.
Étape 3 : Ajouter les fonctionnalités de chaque module sur les autocollants verts.
Étape 4 : Ajouter les fonctionnalités supplémentaires sur les autocollants cyans.
Étape 5 : Ajouter les Ressources documentaires à créer (descriptions/documentations).

Définir une hiérarchisation des taches : les taches prioritaires en haut de la liste.
Tâche fini : ajouter le symbole pour valider la tâche.
Bloquante : changer la couleur des autocollants en rouge si vous rencontrez un point bloquant.

Release Legend

Off-boarding plan for Maxence



Explication du programme

Ce diagramme représente la structure du programme et les interactions entre les différents fichiers

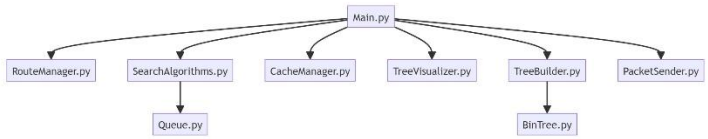
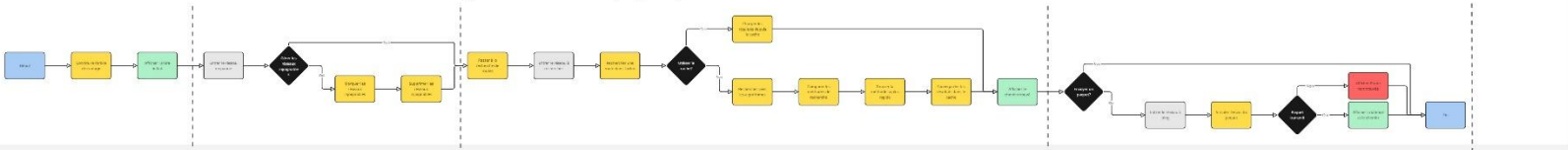
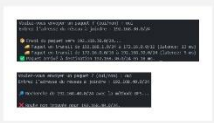


Diagramme de flux qui illustre le déroulement du programme, depuis la construction de l'arbre jusqu'à l'envoi de paquets.
Le diagramme montre les étapes principales et les interactions entre les différents modules.





3. DEVELOPPEMENT TECHNIQUE

3.1 ARCHITECTURE DU SYSTEME

Le système est basé sur une architecture modulaire, où chaque fonctionnalité est gérée par un script Python distinct.

Les principales composantes du système sont :

- **Arbre binaire** : Représente la structure du réseau, où chaque nœud est un routeur.
- **File d'attente (Queue)** : Utilisée pour gérer les paquets en attente d'envoi et pour explorer les nœuds lors du parcours en largeur (BFS).
- **Algorithme de routage** : Implémente les parcours en profondeur (DFS) et en largeur (BFS) pour trouver le chemin optimal.

Pour plus de détails, veuillez consulter le dépôt du projet, où vous trouverez une documentation complète, de l'installation à l'utilisation du programme. **https://github.com/NANDILLONMaxence/ARB_Routage**

3.2 FONCTIONNALITÉS IMPLÉMENTÉES

- **Représentation du réseau** : Chaque nœud de l'arbre représente un routeur, et les branches représentent les connexions entre les routeurs.
- **Algorithme de routage** : Utilisation de DFS et BFS pour trouver le chemin optimal vers une destination.
- **Simulation d'envoi de paquets** : Simulation de l'envoi de paquets en suivant le chemin optimal calculé.
- **Gestion des réseaux inaccessibles** : Gestion des cas où un ou plusieurs réseaux sont injoignables.
- **Extension avec ABR** : Utilisation d'un arbre binaire de recherche pour optimiser la recherche des routes et leur coût.
- **Stockage des requêtes utilisateur :** Stockage en cache des requêtes utilisateur dans un fichier .txt afin d'optimiser les futures requêtes redondantes.

3.3 OUTILS ET TECHNOLOGIES UTILISÉS

- **Python** : Langage de programmation principal utilisé pour implémenter les algorithmes et les structures de données.
- **Mermaid** : Pour la création de diagrammes de flux et d'UML.
- **Mise en « Cache »** : Utilisation d'une méthode de cache pour formater les informations de recherche dans un fichier texte.
- **Copilote** : Pour ajouter des descriptions et des commentaires dans le code.

4. DIFFICULTÉS RENCONTRÉES

4.1 PROBLÈMES TECHNIQUES

- **Gestion des parcours d'arbres** : La mise en place des algorithmes de parcours en profondeur (DFS) et en largeur (BFS) a été complexe, notamment en raison de la nécessité de gérer les nœuds et les branches de manière distincte.
- **Optimisation du routage** : Déterminer le chemin optimal avec le coût le plus faible, sans utiliser la bibliothèque **time**. Cela m'aurait permis de mesurer le temps de recherche de chaque méthode afin de sélectionner la plus rapide, plutôt que de choisir celle ayant parcouru le moins de nœuds pour atteindre le réseau cible. Cela nécessité des recherches approfondies et des tests multiples.
- **Gestion des routes obstruées avec alternative** : La gestion des cas où une route est obstruée et la proposition d'une alternative ont posé des problèmes de logique et de mise en œuvre.

4.2 SOLUTIONS APPORTÉES

- Pour les parcours d'arbres, j'ai utilisé des structures de données comme les **files d'attente (Queue)** pour gérer les nœuds à explorer lors du parcours en largeur (BFS).
- Pour l'optimisation du routage, j'ai implémenté des algorithmes de comparaison de chemins pour trouver celui avec le coût le plus faible (en termes de nœuds parcourus).
- Pour la gestion des routes obstruées, j'ai ajouté une logique de détection des routes inaccessibles. Néanmoins, j'ai retiré la proposition d'alternatives qui est difficilement applicable sur les arbres binaires. On admet que notre réseau informatique ne dispose pas de redondance entre chaque réseau.

5. CONCLUSION

5.1 BILAN DU PROJET

Le projet **Optimisation du routage des paquets (ARB_routage)** a été une expérience enrichissante qui m'a permis de mettre en pratique les connaissances acquises en algorithmique avancée. J'ai réussi à implémenter une architecture réseau en utilisant des arbres binaires et à optimiser le routage des paquets en trouvant le chemin le plus efficace.

5.2 PERSPECTIVES D'AMELIORATION

- **Automatisation des tests** : Implémenter des tests automatisés pour vérifier l'efficacité des algorithmes de routage.
- **Extension avec d'autres algorithmes** : Explorer d'autres algorithmes de routage pour améliorer les performances.

6. ANNEXES

6.1 GUIDE D'UTILISATION

Un guide d'utilisation détaillé est disponible sur le dépôt GitHub du projet : **https://github.com/NANDILLONMaxence/ARB_Routage**

6.2 LISTE DES OUTILS ET SERVICES UTILISES

- **Python** : Langage de programmation principal.
- **Miro** : Pour la gestion des tâches et la création de diagrammes.
- **Mermaid** : Pour les diagrammes de flux et d'UML.
- **Copilote** : Pour les commentaires et descriptions dans le code.