Fantin Puéo--Moulet
Chloé Migliasso
Hamza Hamman

Projet de Programmation Impérative 3

	nom court	NOM	Prénom			
équipier1	HH	Hamman	Hamza			
équipier2	Fanfan	Puéo–Moulet	Fantin			
équipier3	C	MIGLIASSO	Chloe			

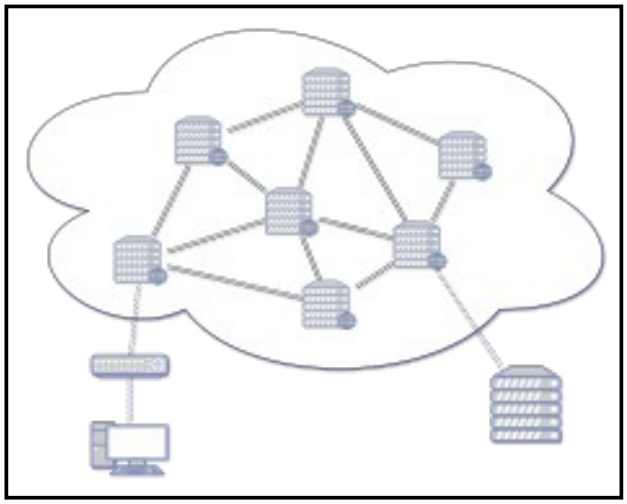
Quoi ?		**Qui ?**				
Modules	**Sous-programmes**	**Spécification**	**Raffinage**	**Codage**	**Test**	**Relecture**
Fonctions_globales	Afficher_Cle_Integer	Fanfan / C	C	C		
	Afficher_Donnee_Case	Fanfan / C	C	C		
	Id_ad_IP	Fanfan / C	HH	HH		
	Table_routage	Fanfan / C	C / Fanfan	Fanfan		
	Gerer_commandes	Fanfan / C	C / HH	C		
	Ouvrir	C	C / HH	C		
	Traiter_les_paquets	C	C / HH	C		
	Association_ad_des	Fanfan / C	C / HH	C		
	Ecrire	C	C / HH	C		
	Identifier_commande	Fanfan / C	C / HH	C		
	Ecrire_Ad_IP	C	C / HH	C		
	Afficher_Ad_IP	C	C / HH	C		
Routage_simple	Routage_simple		C / HH	C		
Routeur Exceptions	Routeur Exceptions		C	C		
LCA	–	–	–	–	–	–
pr03.gpr	–	–	–	C		

Résumé :

Ce rapport à pour objectif de retracer notre travail en présentant la manière dont nous l'avons réalisé, c'est-à-dire les choix que nous avons fait en fonction de notre compréhension du sujet et de ses contraintes. Il a aussi pour but de nous permettre de prendre du recul sur le travail en lui même ainsi que notre gestion qui passe notamment par notre organisation et notre répartition. Nous espérons exposer ici notre travail de manière clair et compréhensible.

Intro :

Dans un réseau informatique, le routeur est chargé d'aiguiller les paquets de données vers leur destination. Pour chaque paquet reçu, il doit consulter une table de routage et déterminer l'interface de sortie adéquate en fonction de l'adresse IP de destination. Cette opération repose sur une correspondance de préfixes : si plusieurs routes correspondent, celle possédant le masque le plus long est privilégiée. Pour optimiser un routeur, on peu lui rajouter un cache, une sorte de mini table de routage contenant un nombre limité de route. Généralement, ceux sont les plus fréquente ou les plus récente.



Exemple :

la table de routage contient tout les différents chemin possible entre les différents routeur. Chaque chemin est composé d'un départ et d'une arrivée précise séparé (ou pas) par différentes étapes (routeur intermédiaires).

Principaux choix réalisés :

Création de 'Liste' pour éviter de manipuler les clefs en utilisant des SDA.

Principaux algorithmes réalisés :

1. Modules de Structures de Données (SDA)

Ce sont les fondations qui stockent les routes.

- **LCA.ads/adb (Tous les routeurs)** :
  - *Rôle* : Gère la table de routage principale (le fichier table.txt).
  - *Fonctions* : Initialiser, Enregistrer, La_Valeur (pour trouver l'interface).
- **LISTES.ads/adb (Routeur Cache Liste uniquement)** :
  - *Rôle* : Gère le cache sous forme linéaire.
  - *Fonctions* : Ajout_cache, Elimination (FIFO, LRU, LFU).
- **Cache_Arbre.ads/adb (Routeur Cache Arbre uniquement)** :
  - *Rôle* : Gère le cache sous forme de structure hiérarchique bit à bit.
  - *Fonctions* : Rechercher_Arbre, Insérer_Noeud.

2. Module Fonctions_globales

C'est le moteur qui contient la logique métier.

- **Gerer_commandes (Tous, mais rôle différent)** :
  - *Simple* : Récupère juste les noms de fichiers.
  - *Caches* : Récupère aussi -c (taille) et -p (politique).
- **Table_routage (Tous)** :
  - *Rôle* : Lit le fichier texte et remplit la LCA.
- **association_ad_des (Le cœur de l'algorithme)** :
  - *Simple* : Recherche directe dans la LCA (Masque le plus long).
  - *Caches* : **Logique Hiérarchique** :
    1. Appel recherche dans Cache.
    2. Si échec : Recherche dans LCA + Mise à jour du Cache.
- **Traiter_les_paquets (Tous)** :
  - *Rôle* : Boucle principale qui lit paquets.txt et écrit les résultats.
  - *Différence* : Dans les versions Cache, elle fait circuler l'objet Cache à chaque appel.

3. Programmes Principaux (.adb)

L'ordre d'appel des fonctions dans ton code :

A. routage_simple.adb

1. Gerer_commandes
2. Initialiser(Tab_routage) (LCA)
3. Table_routage (Chargement)
4. Traiter_les_paquets (Recherche directe LCA)
5. Detruire(Tab_routage)

B. routage.adb (Cache Liste)

1. Gerer_commandes (avec -c et -p)
2. Initialiser(Tab_routage) **ET** Initialiser(Cache) (Liste)
3. Table_routage
4. Traiter_les_paquets (Logique : Cache -> Table -> Maj Cache)
5. Detruire (des deux structures)

C. Routeur_LA.adb (Cache Arbre)

1. Gerer_commandes
2. Initialiser(Tab_routage) **ET** Initialiser(Cache) (Arbre)
3. Table_routage
4. Traiter_les_paquets (Recherche optimisée dans l'arbre)
5. **Obtenir_Statistiques** (Spécifique à cette version pour mesurer le Taux de défaut).
6. Detruire

4. Modules des Exceptions

- **Routeur_exceptions.ads (Tous)** : Fichier_Inconnu_Error, Adresse_IP_Introuvable_Error.
- **SDA_Exceptions.ads (Caches uniquement)** : Taille_Cache_Error.
- **Cache_Exceptions.ads (Arbre uniquement)** : Exceptions liées à la structure de l'arbre.

Difficultés rencontrées et les solutions adoptée :

Organisation des différents modules

Nous ne nous rendions pas forcément compte de la taille et du temps nécessaire à la réflexion et au code de certaines fonction, menant ainsi à l'attente de certaines parties pour en coder d'autres ou du moins les tester.

Organisation de l’équipe :

Hamza et Fantin on travaillé sur des parties de raffinage et de code précise sur les deux version du projet (simple et cache). Chloé à elle aussi travaillé sur des partie précise mais elle c’est aussi penchée sur une grande partie des fonctions dans le but de maintenir une cohérence du projet et un certain lien entre les différents modules, procédures et fonctions.

Bilan Technique :

- Implémentation de nouveaux modules Listes et Arbre génériques adapté à notre situation
- Généricités sur modules et sur fonctions
- Fonctions récursives et itératives

Bilan personnel (Fantin) :

Ce projet m’a permis de mieux comprendre le fonctionnement types, que ce soit leurs caractéristiques, les actions possible en fonction de ces derniers etc. Il m’a aussi mieux fait comprendre les liens entre les nombreux fichier qui composent un tel projet. Ces deux points sont les deux qui me posait le plus problème en programmation impérative. J’ai passé sur ce projet plusieurs heure pour faire les raffinages qui m’ont été attribué. Quelques petites heures pour comprendre celui fait par mes camarades ainsi que les modifications et solutions trouvées par Chloé sur mes propre raffinages (un grand merci à elle). De la même façon j’ai passé plusieurs heure à coder les fonctions, procédures etc qui m’ont été attribué (une certaine partie de ces heures à coder simplement et le reste à corriger, vérifier, et encore une fois comprendre les corrections et les conseils de Chloé). Pour le rapport je dirais quelques petites heures (2h?).

<u>Bilan personnel (Chloé)</u> :

Intérêt : C'était un projet très intéressant, j'aurai adoré avoir une semaine de plus, j'avais des idées pour faire des choses encore plus génériques, mais je n'ai pas eu le temps.

Temps passé : ~40h (+/- 10h)

Temps passé à la conception : ~12h

Temps passé à l'implémentation : ~8h

Temps passé à la mise au point : ~20h

Temps passé sur le rapport : ~10 min

Enseignement tiré de ce projet :

- La complexité des modules génériques
- Réflexion structurés pour mettre en rapport les différents modules
- Fonctionnement approfondis des modules génériques
- Travail en groupe

Commentaire négatif :

- Ce projet était peu claire à la lecture (ça m'a pris environ 5 lectures totales pour comprendre, puis j'ai fait les raffinages, puis j'ai réaliser que j'avais mal compris quelque chose).
- J'ai également peut-être exagéré sur les heures de travail au totale.
- J'aurai du être plus efficace.
- Besoin de systématiquement réécrire le raffinage prend un temps considérable pour quelque chose qui à la fin n'est pas extrement lisible. Je pense qu'on devrait trouver un moyen plus efficace, parce que bouger sans cesse entre les différentes actions complexes est prône à faire des erreurs au bout d'un moment.

Notes pour le future :

- Meilleure communication dans l'équipe
- Répartition dans différents modules plus équitable (dans ce projet il y a vraiment beaucoup qui dépend de fonctions_globales.adb)
- Augmenter efficacité du travail (meilleure raffinage pour éviter le besoin de passé du temps sur la mise au point).

<u>Bilan personnel (Hamzha)</u> :

-Ce projet m'a permis d'approfondir l'organisation du travail et la structuration des tâches pour une intégration cohérente dans l'ensemble du projet. J'ai consolidé mes compétences sur les pointeurs,

les modules et la généricité en Ada, et découvert comment un arbre binaire préfixe peut représenter efficacement des adresses IP pour des recherches rapides et optimiser les performances du cache. J'ai rencontré des difficultés lors de l'implémentation des fonctions de gestion du cache arbre, notamment pour garantir la cohérence des données et implémenter correctement les politiques de suppression FIFO, LRU et LFU. Les tests, particulièrement pour valider le comportement de l'arbre avec différentes tailles de masques et politiques, ont également exigé un temps considérable pour assurer la robustesse du système.

Temps passé (~35h)
Temps passé à la conception : ~9h
Temps passé à l’implémentation : ~12h
Temps passé à la mise au point : ~14h
Temps passé sur le rapport : ~15 min

Ce projet m'a permis d'approfondir l'organisation du travail et la structuration des tâches pour une intégration cohérente dans l'ensemble du projet. J'ai consolidé mes compétences sur les pointeurs, les modules et la généricité en Ada, et découvert comment un arbre binaire préfixe peut représenter efficacement des adresses IP pour des recherches rapides et optimiser les performances du cache. J'ai rencontré des difficultés lors de l'implémentation des fonctions de gestion du cache arbre, notamment pour garantir la cohérence des données et implémenter correctement les politiques d'éviction FIFO, LRU et LFU. Les tests, particulièrement pour valider le comportement de l'arbre avec différentes tailles de masques et politiques, ont également exigé un temps considérable pour assurer la robustesse du système.