Compte rendu projet informatique :

Programme de routage avec cache

Anaïs Charpentier
Julien Wu
Romain Malinge

Résumé

Dans ce rapport, vous trouverez les différents choix, raffinages et tests que nous avons mis en place pour réaliser deux programmes capables de router des adresses IP vers des interfaces. Vous trouverez également la présentation des principaux packages et types de données que nous avons choisis d'utiliser, le tableau d'organisation de l'équipe et finalement des bilans techniques puis personnels sur le projet.

Sommaire

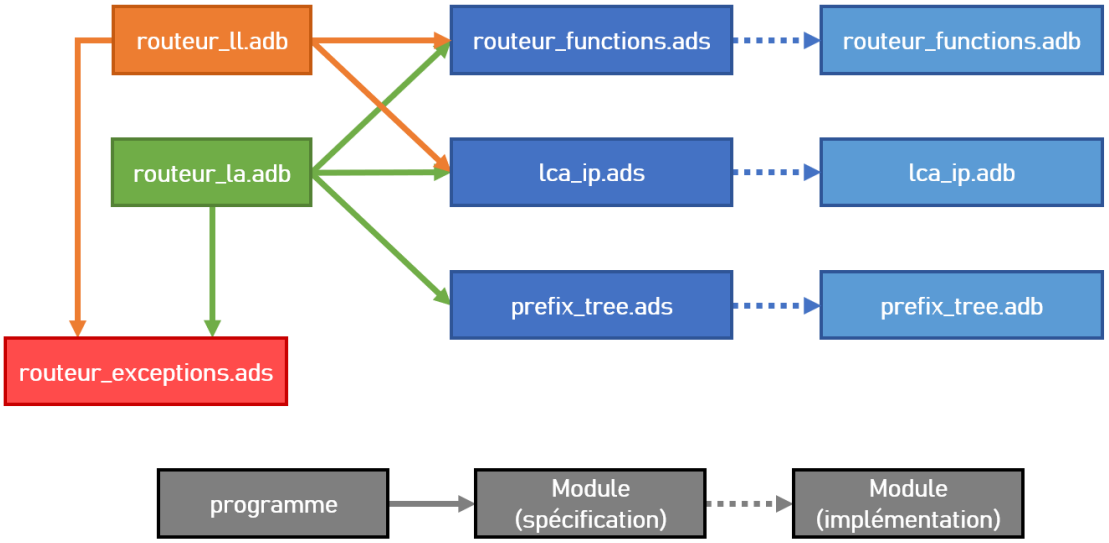
Introduction

Un routeur est un appareil permettant de créer un réseau Wi-Fi. Il doit pour cela être relié à un modem. Il envoie les informations provenant d'Internet à vos appareils personnels (ordinateurs, téléphones et tablettes). Il fonctionne en transmettant les adresses IP qu'il reçoit vers des interfaces selon une table de routage. Pour améliorer son efficacité on peut utiliser un cache qui conserve un sous ensemble des informations de la table de routage, celles qui ont des chances d'être utilisées dans le futur. L'idée étant que si une information a été utilisée, elle a de bonnes chances d'être réutilisée.

Notre projet consiste à programmer un routeur avec un cache et différentes options notamment celle de la politique de suppression des éléments du cache.

I - Architecture des programmes en modules

Pour l'architecture des programmes nous avons choisi de faire une organisation en module. De cette manière les programmes principaux routeur_ll et routeur_la peuvent utiliser les trois modules qui permettent de router les IP.



II - Les principaux choix de conception

Nos principaux choix se sont portés sur la façon d'implémenter le cache sous ces différentes formes et sur son actualisation selon les politiques.

Afin de simplifier les packages dans le cas de la Lca, nous avons décidé de toujours enregistrer le dernier élément ajouté à la Lca Cache au début de cette dernière (le cache suit l'ordre chronologique des utilisations) et de supprimer soit le dernier élément du Cache (FIFO et LRU) soit l'élément avec la plus petite fréquence (LFU). C'est-à-dire une seule fonction enregistrer et deux fonctions supprimer pour gérer 3 politiques différentes.

Par ailleurs, pour organiser et optimiser notre programme, nous avons choisi de répertorier toutes les fonctions nécessaires à l'utilisation d'une Lca (à l'exception des fonctions définies par des Pour_Chaque qui doivent être dans le programme principal) dans un module à part nommé lca_ip. Cela nous permet de réutiliser le même module pour les deux Lca Table de routage et Cache (dans le cas LL).

De plus, nous avons aussi décidé de créer un module routeur_functions regroupant la majorité des sous-programmes utiles aux programmes principaux (routeur_ll et routeur_la) afin d'améliorer la lisibilité du code.

Quant à la modélisation d'un arbre à préfixe, nous avons décidé de l'implémenter dans un module prefix_tree contenant l'ensemble des programmes utiles à la manipulation d'un arbre à préfixe.

Principale modification faite par rapport au premier livrable :

Modification de l'IP et du masque enregistrés dans le cache (avec l'ajout de la variable Grand_masque et de la fonction Trouver Grand Masque) afin de rectifier la cohérence du cache. En effet, dans le premier rendu du raffinage notre programme ne respectait pas le mode de fonctionnement détaillé dans la partie "*1.4.2 Cohérence du cache*" du sujet en ne mettant pas toujours le masque nécessaire.

III- Principaux éléments de conceptions

A - Algorithmes

À partir de la ligne de commande, le programme récupère la liste des arguments et les traite pour obtenir les options choisies (politique, taille du cache, affichage des statistiques) et la liste des noms des fichiers contenant la table de routage, les IPs à router et finalement celui où seront écrits les résultats du routage .
Ensuite, le programme va traiter le fichier paquet ligne par ligne jusqu'à la fin de ce dernier ou la rencontre de la ligne "fin".
Pour chaque ligne il y a deux possibilités :

- La ligne n'est pas une adresse ip
  Le programme affiche ce qui est demandé par la ligne (parmi des options prédéfinies sinon un message d'erreur s'affiche expliquant que le fichier n'est pas au bon format)

- La ligne est une adresse ip
  Cette dernière va alors être routée, pour cela le programme va créer puis actualiser un cache. L'adresse IP sera alors comparée à celles présentent dans le cache(en prenant en compte les masques). Puis si aucune correspondance n'a été trouvée dans le cache on la compare à celles de la table de routage. FInalement on écrit l'IP et l'interface correspondante trouvée dans le fichier résultat.

Actualisation du cache :
Actualiser le cache selon son type (LA ou LL) et l'option de politique choisie

- Cas LL :

  - Si l'adresse IP a été trouvée :
    On incrémente la fréquence de l'adresse du cache qui a permis de router l'IP puis :
    - Si la politique est LRU : on remet la cellule correspondante à l'adresse du cache à la fin de ce dernier (on la supprime puis la ré-enregistre)

  - Si l'adresse IP n'a pas été trouvée :
    On l'ajoute avec le masque et le port correspondant et une fréquence nulle puis si on a déjà ajouté un nombre d'IP supérieur à la taille max de cache :
    - Si la politique est LRU ou FIFO :

On supprime la dernière cellule de la Lca Cache

- Si la politique est LFU :

  On supprime la cellule de Cache avec la fréquence la plus basse.

- Cas LA :

  - Si l'adresse IP a été trouvée :

    On incrémente la fréquence de l'adresse du cache qui a permis de router l'IP et :

    -Si la politique est LFU :

    On met son rang à sa fréquence

    - Si la politique est LRU :

    On met son rang à rang_max_existant_dans_Cache+1

  - Si l'adresse IP n'a pas été trouvée :

    On l'ajoute avec le masque et le port correspondant, une fréquence nulle et un rang égal à rang_max_existant_dans_Cache+1 puis
    Si on a déjà ajouté un nombre d'IP supérieur à la taille de cache maximum :

    On cherche le rang le plus haut présent dans l'arbre
    On supprime la feuille où le rang est le plus bas puis on ré-enregistre l'adresse IP avec un rang nul

Finalement selon l'option "Bavard" le programme va afficher ou non les statistiques du routage qu'il vient d'effectuer.

B - Types : T_Lca_IP et T_Arbre

Pour construire les deux programmes : Suivant que le cache est représenté par une liste chaînée (LL) ou un arbre préfixe (LA) (La table de routage étant toujours stockée avec une liste chaînée (LL).) Nous allons définir deux types principaux :

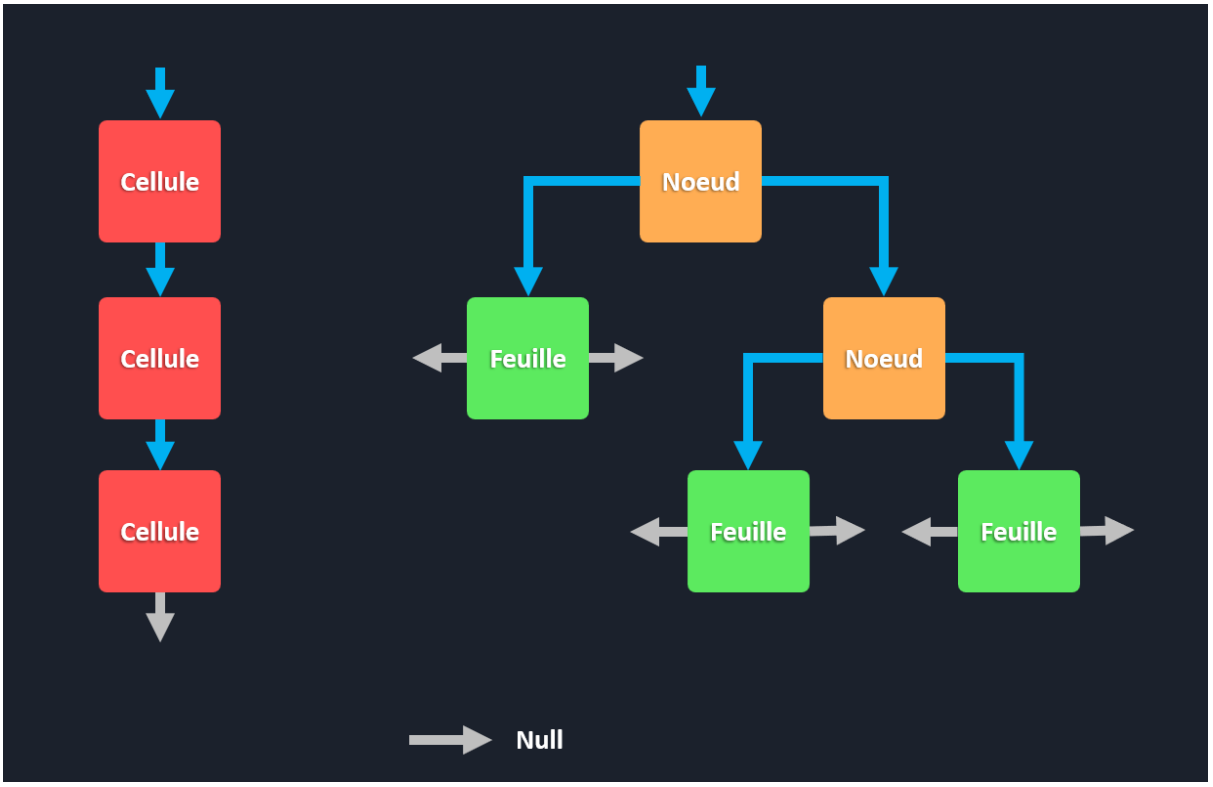
Pour la liste chaînée : **T_LCA_IP**
C'est un pointeur vers un enregistrement **T_Cellule** qui contient

Destination	**T_Adresse_IP**	
Masque	**T_Adresse_IP**	
Port	**Unbounded_String**	– Interface
Fréquence	**Integer**	– Nombre d'utilisation d'une adresse dans le cache (0 dans la table)
Suivant	**T_LCA_IP**	

Pour l'arbre préfixe : **T_Arbre**
C'est un pointeur vers un enregistrement **T_Noeud** qui contient

Destination	**T_Adresse_IP**	
Masque	**T_Adresse_IP**	
Port	**Unbounded_String**	– Interface
Feuille	**Boolean**	– Indique si la cellule est une feuille ou un noeud
Fréquence	**Integer**	– Nombre d'utilisation d'une adresse dans le cache (0 dans la table)
Rang	**Integer**	– Le rang qui indique la feuille à supprimer
Gauche	**T_Arbre**	
Droite	**T_Arbre**	

Où T_Adresse_IP est le type bien défini dans le sujet du projet.



IV - Test et robustesse des programmes

Pour la robustesse nous avons notamment créé une liste d'exceptions dans un module appelé routeur_exceptions.ads qui répondent à l'ensemble des problèmes et/ou exceptions que nous avons pu rencontrer au cours du codage du routeur. Nous gérons les exceptions dans les programmes principaux routeur_ll et routeur_la. Lorsqu'une exception est levée, le programme affiche un message d'erreur. Par exemple, si sur la ligne 11 du fichier table.txt (contenant la table de routage) il serait écrit une unique chaîne de caractère au lieu d'une adresse ip suivi de son masque et de son interface alors le programme affichera :

```
/!\ ERREUR /!\ La ligne 11 du fichier table.txt est incorrecte
```

En ce qui concerne les Tests nous avons créé une table et un paquet test pour vérifier le fonctionnement des trois politiques puis nous avons codé trois fichiers de test : Test_Functions, Test_Prefix_Tree, Test_Lca_IP pour tester respectivement les fonctions des modules routeur_functions, prefix_tree et lca_ip.

Les programmes des deux routeurs (LL et LA) sont donc dans l'ensemble robuste et répondent aux attentes du sujet.

V - Les difficultés rencontrées

Tout au long de ce projet, nous avons rencontré différentes difficultés. La première étant tout simplement la gestion du type T_Adresse_IP qui est un type défini dans le sujet du Projet et qui a pour but de nous faciliter la tâche.

```
type T_Adresse_IP is mod 2**32
```

Dans l'intention de mieux comprendre le fonctionnement de ce type, nous nous sommes référés aux exemples donnés dans le sujet. En effet, pour pouvoir programmer correctement le routeur, il faut pouvoir manipuler le type T_Adresse_IP et pour cela nous avons "rangés" la majorité des fonctions portant sur ce type dans le module routeur_functions. Ce module comporte diverses fonctions d'utilités différentes : Initialiser un T_Adresse_IP, Afficher un T_Adresse_IP, Extraire d'un fichier texte un T_Adresse_IP, Transformer une chaîne de caractère en T_Adresse_IP.

Un second problème qui s'est posé en travers de notre progression dans le projet est la conversion d'une chaîne de caractères vers le type T_Adresse_IP. Pour résoudre ce problème nous avons fini par comprendre qu'il suffisait de caster le type T_Adresse_IP sur les entiers qui représentaient l'adresse IP lus dans les fichiers. Si Entier est une variable entière et si IP est une variable T_Adresse_IP alors :

IP := T_Adresse_IP(Entier); -- IP est le T_Adresse_IP associé à la variable entière Entier

Le dernier problème que nous avons rencontré est l'implantation de la fonction pour supprimer proprement une feuille d'un arbre à préfixe. Une première approche, plutôt itérative, était de supprimer la feuille contenant la bonne IP puis de déterminer, pendant le chemin vers la bonne feuille, l'entier à partir duquel les nœuds possiblement inutiles pouvaient se situer. Cette méthode, malgré sa pertinence, n'a pas porté ses fruits. Nous nous sommes alors tournés vers une autre méthode plutôt récursive qui n'étudiait pas localement la branche où se situait la feuille supprimée mais l'arbre dans son ensemble. En effet, après avoir supprimé la bonne feuille, l'appel à un sous-programme permet de re-parcourir l'ensemble de l'arbre à la recherche des nœuds inutiles et de les supprimer. Cette fonction de suppression est importante afin de respecter le concept d'arbre préfixe.

VI - Bilan technique

Les programmes répondent à la totalité du cahier des charges. De plus, ils sont robustes et les résultats sont affichés de manière à être le plus lisible possible.

VII - Bilans personnels

- Julien WU

Le projet d'informatique m'a beaucoup plu, de par le fait que j'aime la matière, mais aussi que nous avons pu réaliser un programme entièrement, qu'il a une application concrète et qu'il répond à un enjeu important, celui d'envoyer vers les bonnes interfaces les paquets entrants. Ce projet m'a éclairé sur l'importance de l'utilisation des commentaires et de la sémantique des fonctions, ainsi que sur le défi qu'est la programmation avec une équipe. Nous nous sommes régulièrement réunis dans le but d'avancer sur ce projet et cela a porté, selon moi, ses fruits.

- Anaïs CHARPENTIER

Personnellement j'ai beaucoup apprécié ce projet, le sujet étant très intéressant et concret. De plus, le travail de groupe permet de partager les idées de manière à être plus efficace. Cela m'a permis de progresser, premièrement dans ma façon de coder mais également d'un point de vue rédactionnel. Effectivement, ceci étant mon premier projet d'informatique à plusieurs, j'ai vite compris qu'écrire un programme clair et précis est aussi primordial qu'une répartition claire des tâches. Ajouter à cela, j'ai pu approfondir et utiliser plusieurs notions déjà vu en TDs et/ou TPs, les LCA et les Arbres afin de mieux les comprendre, savoir les implanter et les utiliser.

- Romain MALINGE

Pour ma part, ce projet d'informatique a vraiment été très enrichissant sous plusieurs aspects : la programmation, la culture informatique et le travail de groupe. Concernant le travail de groupe, j'ai particulièrement apprécié le fait de devoir se répartir les tâches efficacement au sein d'une équipe pour atteindre un objectif commun. De plus, le projet m'a beaucoup appris sur les types abstraits, avec de nombreuses fonctionnalités Ada que j'ai découvert. J'ai également beaucoup aimé travailler sur l'interface utilisateur, qui est pour moi l'un des points les plus importants pour faire un bon programme.
Pour finir, je remercie tout particulièrement Armand pour son aide précieuse.

Annexes

Tableau “Qui a fait quoi”

Equipe
CD-4

	nom court	NOM	Prénom
Auteurs	JW	WU	Julien
	AC	CHARPENTIER	Anaïs
	RM	MALINGE	Romain

Quoi ?		Qui ?			
Modules	**Sous-programmes**	**Spécification**	**Codage**	**Test**	**Relecture**
routeur_ll	traiter la ligne de commande(identique pour routeur_ll et routeur_la)	AC	All	RM	All
	router les IPs	JW	All	RM	All
routeur_la	router les IPs	RM	RM	JW	All
routeur_functions	affichage des IPs	AC	AC	JW	All
	initialisation et passage au type T_Adresse_IP	JW	RM	JW	All
lca_ip	Suppression et Enregistrement	AC	AC	RM	All
	trouver des paramètres dans la lca (Trouver_Grand_Masque, La_Frequence,...)	AC	JW	AC	All
prefix_tree	Suppression et Enregistrement	JW	RM	JW	All
	Affichage et comparaison	RM	RM	AC	All

Grille d’évaluation du code

Numéro	OK (O/N)	Règle	Justification / commentaire (optionnel)
1	O	Le programme ne doit pas contenir d'erreur de syntaxe.	
2	O	Les modules doivent être documentés : expliquer l'objectif du module (en début de fichier)	
3	O	Les sous-programmes doivent être documentés : une phrase pour l'objectif, les paramètres doivent être expliqués, les conditions d'utilisation, les effets.	
4	O	Les sous-programmes doivent être séparés par deux lignes blanches.	
5	O	Les sous-programmes ne doivent pas être trop longs ni complexes : environ 10-15 lignes et pas plus de 3 structures de contrôle.	
6	O	Il ne doit pas y avoir de code redondant. Le code redondant doit être factorisé via l'utilisation de structures de contrôle, sous-programmes, etc.	
7	O	On doit utiliser les structures de contrôle adaptées (conditionnelle, répétition, exception...)	
8	O	On doit utiliser des constantes nommées plutôt que des constantes littérales.	
9	O	Les raffinages doivent être respectés dans le programme.	
10	O	Les actions complexes doivent apparaître sous forme de commentaires placés AVANT les instructions correspondantes.	
11	O	Une ligne blanche doivent séparer les principales actions complexes	
12	O	Le rôle des variables doit être explicité à leur déclaration (commentaire).	
13	O/N	Il ne doit pas y avoir de variables globales.	Pas de solution trouvée pour un argument en in out d’une fonctions récursive
14	O	Les identifiants choisis doivent être significatifs.	

RAFFINAGE DES PROGRAMMES DE ROUTAGE

Préambule :

Les fonctions Afficher, Initialiser et Enregistrer utilisées sur les variables Table et Cache(quand le cache est une lca), sont celles du package LCA vu en TD, TP et PR2 mais avec 4 éléments par cellules. Afin de faciliter la lecture des raffinages nous avons décidé de ne pas les raffiner. Ces mêmes fonctions sont raffinées quand Cache est un arbre et quand elles viennent donc du package prefix_tree.
Les types T_Lca et T_Arbre sont ceux définis dans la partie **III-B** du rapport
Pour une meilleure lisibilité les opérations du routeur sont détaillées dans 2 sous-raffinages en fonction du type du cache (Raffinage 2 : Routeur_LL et Raffinage 3 : Routeur_LA). Ils reprennent le raffinage là où les opérations complexes sont surlignées en bleu.

Raffinage 1: Routeur_L[LA]

R0: Gérer le routage d'un paquet d'IP avec un cache	
R1: Comment “Gérer le routage d'un paquet d'IP avec un cache” ?	
Paramétrer le programme	Nom_Paquet **out String**, Nom_Table **out String**, Nom_Resultat **out String**, Politique **out T_Politique**, Bavard **out Boolean**, Cache **out T_Cache**, Fin **out Bool**, Nbr_ajoute **out Integer**, Taille_Cache **out Integer**, Grand_Masque **out**, Nbr_route **out**
Gérer les fichiers	Nom_Paquet **in**, Nom_Table **in**, Nom_Resultat **in**, Paquet **out File_Type**,Table **out T_Lca**, Resultat **out File_Type**, Grand_Masque **in out**
Tant que non End_Of_File(Paquet) **ou non** Fin **Faire** Traiter une ligne du fichier Paquet	Paquet **in**, Fin **in** Paquet **in**, Table **in**, Resultat **in out**, Cache **in out**, Fin **in out**, Politique **in**, Nbr_ajoute **in out**, Taille_Cache **in**, Grand_Masque **in**
FinTQ	
Fermer les fichiers	Paquet **in**, Resultat **in**
Gérer les erreurs	
Traiter les statistiques	Bavard **in**, Politique **in**, Taille_Cache **in**, Nbr_ajoute **in**, Nbr_routées **in**, Nom_Paquet **in**, Nom_Table **in**, Nom_Resultat **in**
R2: Comment “Paramétrer le programme” ?	
Initialiser les variables	Nom_Paquet **out**, Nom_Table **out**, Politique **out**, Nom_Resultat **out**, Bavard **out**, Taille_Cache **out**, Fin **out**, Nbr_ajoute **out**, Grand_Masque **out**, Nbr_route **out**
Pour a **De** 1 **À** Argument_Count **Faire** *-- du module Ada.Command_Line*	a **out Integer**,
Traiter l’élément du a eme Argument	a **in**, Politique **in out**, Bavard **in out**, Nom_Paquet **in out**, Nom_Table **in out**, Nom_Resultat **in out**, Taille_Cache **in out**
Lever une erreur dans la ligne de commande	Taille_Cache **in**, Nom_Paquet **in**, Nom_Table **in**, Nom_Resultat **in**
FinPour	
Initialiser le cache	Cache **out**
R2: Comment “Gérer les fichiers” ?	
Open (Paquet, In_File, Nom_Paquet)	Nom_Paquet **in**, Paquet **out**
Create (Resultat, Out_File, Nom_Resultat)	Nom_Resultat **in,** Resultat **out**
Open (File_Table , In_File, Nom_Table)	Nom_Table **in,** File_Table **out File_type**
Enregistrer la table sous forme de LCA	File_Table **in**, Table **out** Grand_Masque **in out**
Close (File_Table)	File_Table **in**

R2: Comment “Traiter une ligne du fichier Paquet” ?	
Ligne_Paquet <-- Get_Line(Paquet) Trim(Ligne_Paquet, Both) **Selon** Ligne_Paquet **Dans** table => Afficher (Table) cache => Afficher le Cache fin => Fin <-- Vrai stat => Afficher les statistiques Autres => Rediriger l’IP **FinSelon**	Ligne_Paquet **out Unbonded_String**, Paquet **in** Ligne_Paquet **in out** Ligne_Paquet **in** Table **in** Cache **in** Fin **in out** Politique **in**, Taille_Cache **in**, Nbr_ajoute **in** Table **in**, Ligne_Paquet **in**, Cache **in out**, Resultat **in out**, Politique **in**, Nbr_ajoute **in out**, Taille_Cache **in**, Grand_Masque **in**
R2: Comment “Traiter les statistiques” ?	
Si Bavard **Alors** Afficher les statistiques **Sinon** **Rien** **FinSi**	Bavard **in** Politique **in**, Taille_Cache **in**, Nbr_ajoute **in**, Nbr_routées **in**, Nom_Paquet **in**, Nom_Table **in**, Nom_Resultat **in**
R2: Comment “Fermer les fichiers” ?	
Close (Paquet) Close (Resultat)	Paquet **in** Resultat **in**
R2: Comment “Gérer les erreurs” ?	
Exception CONSTRAINT_ERROR => Put (“Vous rentrer un mauvais type”) TAILLE_CACHE_ERROR => Put (“La taille du cache doit être supérieure à 0“) NOT_TXT_ERROR => Put (“Vous avez indiqué des noms de fichier sans .txt à la fin“) ROUTAGE_ERROR => Put (“L’adresse du paquet ne figure pas sur la table de routage“)	
R3: Comment “Initialiser les variables” ?	
Nom_Paquet <-- “paquet.txt” Nom_Table <-- “table.txt” Nom_Resultat <-- “resultats.txt” Taille_Cache <-- 10 Politique <-- FIFO Bavard <-- Vrai Fin <-- Faux Grand_Masque <-- 0 Nbr_ajoute <-- 0 Nbr_route <-- 0	Nom_Paquet **out** Nom_Table **out** Nom_Resultat **out** Taille_Cache **out** Politique **out** Bavard **out** Fin **out** Grand_Masque **out** Nbr_ajoute **out** Nbr_route **out**
R3: Comment “Traiter l’élément du a eme Argument” ?	
Si Argument(a)(1) = “-” **Alors** modifier le paramètre correspondant **Sinon** **Rien** **FinSi**	a **in** Taille_Cache **in out**, Politique **in out**, Bavard **in out,** Nom_Paquet **in out**, Nom_Table **in out**, Nom_Resultat **in out**

R3: Comment “Lever une erreur dans la ligne de commande” ?	

Si Taille_Cache < 1 **Alors**
  Lever TAILLE_CACHE_ERROR
SinonSi Nom_Paquet ne finit pas par “.txt” **Alors**
  Lever NOT_TXT_ERROR
SinonSi Nom_Table ne finit pas par “.txt” **Alors**
  Lever NOT_TXT_ERROR
SinonSi Nom_Resultat ne finit pas par “.txt” **Alors**
  Lever NOT_TXT_ERROR
FinSi
	Taille_Cache **in**
	
	Nom_Paquet **in**
	
	Nom_Table **in**
	
	Nom_Resultat **in**

R3: Comment “Enregistrer la table sous forme de LCA” ?	

Initialiser (Table)
Tant que non End_Of_File(File_Table) **Alors**
  Analyser une ligne de Table
 	
  Enregistrer (Table, Destination, Masque, Port, Frequence)
FinTQ
	Table **out**
	File_Table **in**
	File_Table **in**, Destination **out T_Adresse_IP**, Masque **out T_Adresse_IP**, Port **out String**
	Table **in out**, Destination **in**, Masque **in**, Port **in,** Grand_Masque **in out**

R3: Comment “Afficher les statistiques” ?	
Ecrire(“Politique : “ + POLITIQUE'Image(Politique)) **Ecrire**(“Taille maximale du cache : “ + Taille_Cache) **Ecrire**(“Nombre d’ajout au cache : “ + Nbr_Ajoute) **Ecrire**(“Nombre d’IP routées “ + Nbr_routées) **Ecrire**(“Paquet: “ + Nom_Paquet) **Ecrire**(“Table “ + Nom_Table) **Ecrire**(“Résultat “ + Nom_Resultat)	Politique **in** Taille_Cache **in** Nbr_Ajoute **in** Nbr_routées **in** Nom_Paquet **in** Nom_Table **in** Nom_Resultat **in**
R3: Comment “Rediriger l’IP” ?	

Passer l’IP au format T_adresse_IP
Nbr_route <-- Nbr_route +1
Chemin_Trouver <-- False
Comparer IP au Cache
 	
Si non(Chemin_Trouver) **Alors**
  Comparer l’IP à la lca Table
  Grand_Masque <-- Masque
  Trouver_Grand_Masque(Table, IP, Destination, Grand_Masque)
  Modifier le cache
 	
Sinon
  Actualiser le cache
FinSi
Ajouter une ligne dans Resultat avec IP et Port
	Ligne_Paquet **in,** IP **out T_adresse_IP**
	Nbr_route **in out**
	Chemin_Trouver **out Boolean**
	Cache **in**, IP **in,** Chemin_Trouver **out**, masque **out T_Adresse_IP**, Port **out String**, Politique **in**
	
	Chemin_Trouver **in**
	Table **in**, IP **in**, masque **out**, Port **out**
	masque **in**, Grand_Masque **out**
	Table **in**, IP **in**, Destination **in**, Grand_Masque **in out**
	
	IP **in,** masque **in,** Port **in**, Cache **in out**, Politique **in**, Taille_Cache **in**, Nbr_Ajoute **in**, Grand_Masque **in**
	
	Politique **in ,** Cache **in out**, IP **in**, masque **in**
	
	IP **in**, Port **in**, Resultat **in out**

R4: Comment “modifier le paramètre correspondant” ?	

Selon Argument(a)(2) **Dans**
  ‘c‘=> Taille_Cache <-- +Argument(a+1)
  ‘P’ => Politique <-- +Argument(a+1)
  ‘S’ => Bavard <-- False
  ‘s’ => Bavard <-- True
  ‘p’ => Nom_Paquet <-- +Argument(a+1)
  ‘t’ => Nom_Table <-- +Argument(a+1)
  ‘r => Nom_Resultat <-- +Argument(a+1)
  Autres => **Rien**
FinSelon
	a **in**
	Taille_Cache **in out**, a **in**
	Politique **in out**, a **in**
	Bavard **in out**
	Bavard **in out**
	Nom_Paquet **in out**, a **in**
	Nom_Table **in out**, a **in**
	Nom_Resultat **in out**, a **in**

R4: Comment "Analyser une ligne de Table" ?	
Convertir l'IP Destination venant du fichier File_Table Convertir l'IP Masque venant du fichier File_Table Get_Line (File_Table, Port) Trim (Port, Both) **Si** Masque > Grand_Masque **Alors** Grand_Masque <--Masque **Sinon** null; **FinSi**	File_Table **in**, Destination **out** File_Table **in**, Masque **out** Port **in**, Port **out** Port **in out** Masque **in**, Grand_Masque **in** Masque **in**, Grand_Masque **out**
R4: Comment "Passer l'IP au format T_adresse_IP" ?	
UN_OCTET <-- 2 ** 8 U_Str_IP <-- +(Ligne_Paquet) Nombre <-- 0 IP <-- 0 **Pour** i **De** 1 **A** Lenght(U_Str_IP) **Faire** Element <-- U_Str_IP(i) Traiter l'élément de Str_IP **FinPour** IP <-- IP * UN_OCTET + T_Adresse_IP(Nombre)	UN_OCTET **out T_Adresse_IP** Ligne_Paquet **in**, U_Str_IP **out String** Nombre **out Integer** IP **in**, U_Str_IP **in** i **in out Integer**, U_Str_IP **in** i **in**, U_Str_IP **in**, Element **out Character** Element **in**, Nombre **in out**, IP **in out**, UN_OCTET **in** IP **in out**, Nombre **in**, UN_OCTET **in**
R4: Comment "Comparer l'IP à la lca LCA" ? *-- LCA = Table ou Cache dans le programme LL*	
Si LCA = Null **Alors** **Lever** ROUTAGE_ERROR **SinonSi** (IP and LCA.all.Masque) = LCA.all.Destination Actualiser Masque et Port **Sinon** Comparer l'IP à la lca LCA.all.Suivant **FinSi**	LCA **in** LCA **in**, IP **in** LCA **in**, Masque **in out**, Port **in out** IP **in**, LCA **in**, Masque **in out**, Port **in out**
R4: Comment " Trouver_Grand_Masque (Lca ,Destination, IP, Grand_Masque) *-- LCA = Table ou Cache dans le programme LL*	
Si Est_Vide(Lca) **Alors** **Rien** **SinonSi** (Lca.all.Destination and Grand_Masque) = (IP and Grand_Masque) **Et** Grand_Masque < Lca.all.Masque **Alors** Grand_Masque <-- Lca.all.Masque; Trouver_Grand_Masque (Lca.all.Suivant, Destination, IP, Grand_Masque) **Sinon** Trouver_Grand_Masque (Lca.all.Suivant, Destination, IP, Grand_Masque) **FinSi**	Lca **in** Lca **in,** Gand_Masque **in**, IP **in** Lca **in,** Gand_Masque **out** IP **in,** Lca **in**, Destination **in**, Grand_Masque **in out** IP **in,** Lca **in**, Destination **in**, Grand_Masque **in out**
R4: Comment "Ajouter une ligne dans Resultat avec IP et Port" ?	
Ajouter l'IP dans Resultat Put (Resultat, " ", 1) Put (Resultat, Port, 1) New_Line (Resultat)	IP **in**, Resultat **in out** Resultat **in out** Port **in**, Resultat **in out** Resultat **in out**
R5: Comment "Convertir l'IP Une_IP venant du fichier File_Table" ?	
UN_OCTET <-- 2 ** 8 Get(Un_Fichier, Entier) Une_IP <-- T_Adresse_IP(Entier) **Pour** i **De** 1 **À 3 Faire** Get(Un_Fichier, Charatere) Get(Un_Fichier, Entier)	UN_OCTET **out T_Adresse_IP** Un_Fichier **in File_type**, Entier **out Integer** Une_IP **out T_Adresse_IP ,** Entier **in** i **in out Integer** Un_Fichier **in**, Charatere **out Character** Un_Fichier **in**, Entier **in out** Une_IP **in out**

Une_IP<-- Une_IP*UN_OCTET + T_Adresse_IP(Entier) **FinPour**	
R5: Comment “Traiter l’element de Str_IP” ?	
Si Element in ‘0’..’9’ **Alors** Nombre <-- Nombre * 10 + Integer’Image(Element) **Sinon Faire** IP <-- IP * UN_OCTET + T_Adresse_IP(Nombre) Nombre <-- 0 **FinSI**	Element **in** Nombre **in out**, Element **in** IP **in out**, Nombre **in**, UN_OCTET **in** Nombre **in out**
R5: Comment “Actualiser Masque et Port” ?	
Si LCA.all.Masque > Masque **Alors** Masque <-- LCA.all.Masque Port <-- LCA.all.Port **FinSi**	LCA **in**, Masque **in** Masque **in out** Port **in out**
R5: Comment “Ajouter l’IP dans Resultat” ?	
Récupérer les Composantes de IP sous forme de tableau Put (Resultat, IP_Comp(1), 1) **Pour** i **De** 2 **À** 4 **Faire** Put (Resultat, ".") Put (Resultat, IP_Comp(i), 1) **FinPour**	IP **in**, IP_Comp **out T_Ip_Comp** Resultat **in out**, IP_Comp **in** i **in out Integer** Resultat **in out** Resultat **in out**, IP_Comp **in**, i **in**
R6: Comment “Récupérer les Composantes de IP sous forme de tableau” ?	
Créer un tableau d’entier de taille 4 IP_Copie <-- IP1 **Pour** i **De** 1 **À** 4 **Faire** IP_Comp(5-i) <-- Natural (IP_Copie mod 2 ** 8) IP_Copie <-- IP_Copie / 2 ** 8 **FinPour**	IP_Comp **out** IP **in**, IP_Copie **out T_Adresse_IP** i **in out Integer** i **in**, IP_Copie **in**, IP_Comp **in out** IP_Copie **in out**

Raffinage 2: Routeur_LL

R3: Comment “Initialiser le cache” ?	
Cache <-- Null	Cache **out**
R3: Comment “Afficher le cache” ?	
Afficher (Cache)	Cache **in**
R4: Comment “Comparer IP au Cache” ?	
Si Cache = Null **Alors** **Rien** **SinonSi** (IP and Cache.all.Masque) = Cache.all.Destination **Alors** Cache.all.Frequence <-- Cache.all.Frequence + 1 Chemin_Trouver <-- Vrai Masque <-- Cache.all.Masque Port <-- Cache.all.Port **Sinon** Comparer IP au Cache.Suivant **FinSi**	Cache **in** IP **in ,** Cache **in** Cache **in out** Chemin_Trouver **out** Masque **out,** Cache **in** Port **out**, Cache **in** Cache **in,** IP **in,** masque **out**, Port **out**, Chemin_Trouver **out**,Politique **in**
R4: Comment “Modifier le cache” ?	
Enregistrer(Cache, (IP and Grand_Masque), Grand_Masque, Port , 0) Nbr_ajoute <-- Nbr_ajoute + 1 **Si** Nbr_ajoute > Taille_Cache **Alors** Supprimer une cellule du cache **FinSi** Ajouter la cellule a la fin de Cache	Politique **in**, Cache **in out,** IP **in**, masque **out**, Port **out**, Grand_Masque **in** Nbr_Ajoute **in out** Nbr_Ajoute **in,** Taille_Cache **in** Cache **in out**, IP **in**, Masque **in**, Port **in** Cache **in out**, IP **in**, Masque **in**, Port **in**
R4: Comment “Actualiser le cache” ?	
Si Politique = +”LRU” **Alors** Ranger Cache par ordre chronologique **Sinon** **Rien** **FinSi**	Politique **in** Cache **in out**, IP **in**, Masque **in**
R5: Comment "Supprimer une cellule du Cache” ?	
Si Politique = +”LFU ” Supprimer la cellule la moins utilisées **Sinon** Supprimer la première cellule de Cache **FinSi**	Politique **in** Cache **in out** Cache **in out**
R5: Comment “Ranger Cache par ordre chronologique” ?	
Mettre la cellule avec l’IP et le Masque correspondant à la fin de Cache	Cache **in out**, IP **in**, Masque **in**
R6: Comment “Supprimer la première cellule du Cache“ ? FiFO / LRU	
Supprimer_premiere(Cache)	Cache **in out**
R6: Comment “Supprimer la cellule la moins utilisées” ? LFU	
Freq_Min <-- Premiere_Frequence(Cache) Dest_suppr <-- Premiere_Destination(Cache) Mas_suppr <-- Premier_Masque(Cache) Chercher le minimum de fréquence dans Cache	Cache **in**, Freq_Min **out Integer** Cache **in**, Dest_suppr **out T_Adresse_IP** Cache **in**, Mas_suppr **out T_Adresse_IP** Cache **in out,** Freq_Min **in**

Supprimer la cellule de Cache avec la fréquence la plus basse	Cache **in out**, Freq_Min **in**
R6: Comment “Mettre la cellule avec l’IP et le Masque correspondant à la fin de Cache ” ?	
Si Cache = Null **Alors** **Rien** **SinonSi** (IP and Cache.all.Masque) = Cache.all.Destination **Alors** A_Supprimer <-- Cache Cache = Cache.all.Suivant Enregistrer(Cache , A_Supprimer.all.destination, A_Supprimer.all.masque, A_Supprimer.all.port, A_Supprimer.all.frequence) Libérer A_Supprimer **Sinon** Ranger Cache.all.Suivant par ordre chronologique **FinSi**	Cache **in** Cache **in**, IP **in** A_Supprimer **out T_LCA,** Cache **in** Cache **in out** A_Supprimer **in**, Cache **in out** Cache **in out,** IP **in**
R7: Comment “Supprimer la cellule de Cache avec la fréquence la plus basse” ?	
Si Cache = Null **Alors** Rien **SinonSi** Cache.all.Frequence < Freq_Min **Alors** A_Supprimer <-- Cache Cache = Cache.all.Suivant Libérer A_Supprimer **Sinon** Supprimer la cellule de Cache.all.Suivant avec la fréquence la plus basse **FinSi**	Cache **in** Cache **in,** Freq_Min **in** Cache **in,** A_Supprimer **Out T_LCA** Cache **in out** A_Supprimer **in** Cache **in out**, Freq_Min **in**

Raffinage 3: Routeur_LA

R3: Comment "Initialiser le cache" ?	
Cache <-- Null	Cache **out T_Arbre**
R3: Comment "Afficher l'arbre Cache" ? -- *le nom diffère de celui du R2 pour mettre un accent sur la récursivité*	
Si Cache = Null **Alors** **Rien** **Sinon** Afficher l'arbre Cache.all.Gauche Gérer l'affichage d'une feuille Afficher l'arbre Cache.all.Droite **FinSi**	Cache **in** Cache **in** Cache **in** Cache **in**
R4: Comment "Gérer l'affichage d'une Feuille" ?	
Si Cache.all.Feuille **Alors** Afficher Cache.all.Destination Afficher Cache.all.Masque Afficher Cache.all.Port **FinSi**	Cache **in** Cache **in** Cache **in** Cache **in**
R4: Comment "Comparer IP au Cache" ?	
I <-- 1 -- *Compteur* **Si** non(Cache = Null) **Alors** Comparer IP à l'arbre Cache **Sinon** **Rien** **FinSi**	I **out integer** Cache **in**, IP **in,** Chemin_Trouver **in out**, masque **out**, Port **out**, I **in out**
R4: Comment "Modifier le cache"?	
Si Nbr_ajoute >= Taille_Cache **Alors** Trouver l'IP de la destination la moins utilisée de Cache Supprimer IP_LRU de Cache **FinSi** I <-- 1 Enregistrer un élément dans Cache	Nbr_ajoute **in**, Taille_Cache **in** IP_LRU **out T_Adresse_IP** ,Cache **in** IP_LRU **in**, Cache **in out** I **out integer** IP **in,** masque **in,** Port **in**, Cache **in out**, masque **in**, Port **in**, Grand_Masque **in**
R5: Comment "Comparer IP à l'arbre Cache"? -- *le nom diffère de celui du R3 pour mettre un accent sur la récursivité*	
Si Cache.all.Feuille **Et** (IP and Cache.all.Masque) = Cache.all.Destination **Alors** Chemin_Trouver <-- Vrai masque <-- Cache.all.Masque Port <-- Cache.all.Port **SinonSi** non(Cache.all.Feuille) **Alors** Comparer IP dans la branche correspondante **Sinon** **Rien** **FinSi**	Cache **in**, IP **in** Chemin_Trouver **in out** Cache **in**, IP **in** masque **out** Cache **in**, Port **out** Cache **in** Cache **in**, IP **in,** Chemin_Trouver **in out**, masque **out**, Port **out**, I **in out**
R5: Comment "Trouver l'IP de la destination la moins utilisée de Cache" ?	
Min := Nbr_Ajoute Déterminer l'élément, de destination Destination, de plus petit rang Min dans Cache	Min out **Integer,** Nbr_Ajoute **in** Cache **in,** Min **in out,** Destination **out**
R5: Comment "Enregistrer un élément dans Cache" ?	

Si Cache = Null **Alors** Nouv_Cellule <-- new T_NOEUD ((IP and Grand_Masque), Grand_Masque, Port, Vrai, Null, Null) Cache <-- Nouv_Cellule **SinonSi** (IP and Cache.all.Masque) = Cache.all.Destination **Alors** **Rien** **SinonSi** Cache.all.Feuille **Alors** Gérer la génération d'un nouveau noeud **SinonSi** not(Cache.all.Feuille) **Et** (IP and 2**(31-l) = 0) **Alors** l <-- l +1 Enregistrer un élément dans le Cache.all.Gauche **Sinon** l <-- l +1 Enregistrer un élément dans le Cache.all.Droite **FinSi**	Cache **in** IP **in,** masque **in,** Port **in,** Nouv_Cellule **out T_NOEUD**, Grand_Masque **in** Cache **out, Nouv_Cellule in** Cache **out** Cache **in,** IP **in** Cache **in** Cache **in out** Cache **in,** IP **in** l **in out** IP **in**, Cache **in out,** masque **in,** Port **in** l **in out** IP **in**, Cache **in out,** masque **in,** Port **in**
R6: Comment "Comparer IP dans la branche correspondante" ?	
Si IP and 2**(31-l) = 0 **Alors** l <-- l +1 Comparer IP à l'arbre Cache.all.gauche **Sinon** l <-- l +1 Comparer IP à l'arbre Cache.all.droite **FinSi**	IP **in**, l **in** l **in out** Cache **in**, IP **in,** Chemin_Trouver **in out**, masque **out**, Port **out**, l **in out** l **in out** Cache **in**, IP **in,** Chemin_Trouver **in out**, masque **out**, Port **out**, l **in out**
R6: Comment "Gérer la génération d'un nouveau noeud" ?	
Si Cache.all.Destination and 2**(31-l) = 0 **Alors** Nouv_Noeud <-- new T_NOEUD (Null, Null, Null, Faux, Cache, Null) Vider Cache Cache <-- Nouv_Noeud **Sinon** Nouv_Noeud <-- new T_NOEUD (Null, Null, Null, Faux, Null, Cache) Vider Cache Cache <-- Nouv_Noeud **FinSi** Enregistrer un élément dans Cache	Cache **in** Nouv_Noeud **out T_NOEUD** Cache **in out** Cache **in out**, Nouv_Noeud **in** Nouv_Noeud **out** Cache **in out** Cache **out**, Nouv_Noeud **in**
R7 : Comment "Déterminer l'élément, de destination Destination, de plus petit rang Min dans Cache" ?	
Si Cache est vide **Alors** Rien **SinonSi** Cache.all.Feuille et Arbre.all.Rang < Min **Alors** Destination <-- Arbre.all.Destination Min <-- Arbre.all.rang **Sinon** "Déterminer l'élément, de destination Destination, de plus petit rang Min dans Cache.all.Gauche Déterminer l'élément, de destination Destination, de plus petit rang Min dans Cache.all.Droite **FinSi**	Cache **in** Cache **in**, Min **in** Destination **out** Min **in out** Cache **in**, Destination **out**, Min **in out** Cache **in**, Destination **out**, Min **in out**

Grille d’évaluation du raffinage

	Règle	I/P/A/+	Commentaires
Forme	**Respect de la syntaxe** **Ri : Comment "... une action complexe ..." ?** **des actions combinées avec des structures de contrôle** **Rj : ...**	+	
	Verbe à l'infinitif pour les actions complexes	+	
	Nom ou équivalent pour expressions complexes	+	
	Tous les Ri sont écrits contre la marge et espacés	+	
	Les flots de données sont définis	+	
	Une seule décision ou répétition par raffinage	+	
	Pas trop d'actions dans un raffinage (moins de 6)	A	
	Bonne présentation des structures de contrôle	+	
Fond	**Le vocabulaire est précis**	A	
	Le raffinage d'une action décrit complètement cette action	+	
	Le raffinage d'une action ne décrit que cette action	+	
	Les flots de données sont cohérents	+	
	Pas de structure de contrôle déguisée	+	
	Qualité des actions complexes	A	