Présentation du projet

Routeur simple / Routeur avec cache

BOULARD Arno, DE LECLUSE TREVOEDAL Alexis, EL MORHDER Yahya

Structure du projet

Le projet est séparé en 3 programmes principales:

- Routeur_main.adb : Programme principale de la version simple du routeur.

Structure du projet

Le projet est séparé en 3 programmes principales:

- Routeur_main.adb : Programme principale de la version simple du routeur.
- Routeur_LA.adb : Routeur avec cache sous forme d'un arbre préfixe.

Structure du projet

Le projet est séparé en 3 programmes principales:

- Routeur_main.adb : Programme principale de la version simple du routeur.
- Routeur_LA.adb : Routeur avec cache sous forme d'un arbre préfixe.
- Routeur_main_cache_liste.adb : Routeur avec cache sous forme d'une liste chaînée associative.

Modules

Modules de base

Permet la réutilisabilité.

- Adresse_IP : Module de la manipulation des adresses IP.
- LCA : Module de la liste chaînée pour la table de routage et aussi le cache.
- Trie : Module de l'arbre préfixe pour le cache.

Autres Modules

- Cache_trie : Encapsulation du cache sous forme trie.
- Cache_liste : Encapsulation du cache sous forme liste.
- Routeur : Encapsulation des fonctionnalités du routeur simple.
- Routeur_cache_liste : Encapsulation du routeur avec cache de structure LCA.

Conception

Raffinages successifs

- Décomposer le problème en sous-problèmes plus simples.
- Répartition des responsabilités.
- Traçabilité entre la conception et l'implémentation.

Conception

Type de données

Pour la construction du routeur, On a adopté *Unbounded_String* comme type de données pour la représentation de l'adresse IP du a la facilité de manipuler les strings.

Table de routage

La structure de la table de routage est une LCA de type clé *T_IP* et valeur *T_Route*{*T_IP*, *Interface*}.

Conception

Trie

- Exploiter directement la représentation binaire des adresses IPv4.
- Temps de recherche borné par la longueur de l'adresse (32 bits).

Conception

L'implémentation de la structure d'un arbre préfixe est la suivante:

```
1 type T_Cellule_Trie is record
2     Enfants : T_Enfants ; -- Tableau [0..1] de T_Trie
3     Est_Route : Boolean ; -- Indique si route presente
4     Interface_Sortie : Unbounded_String ;
5 end record ;
6 type T_Trie is access T_Cellule_Trie ;
```

Algorithmes

Algorithme de routage

Pour le routage, on recherche d'abord dans le cache si la destination à router est déjà contenu dans le cache. La rapidité de la recherche depend de la structure du cache employée:

- Cache LCA : recherche complete de complexité: $O(n)$.
- Cache Trie : recherche de complexité: $O(32)$.

Pour un cache de taille $+100$, l'arbre préfixe est plus efficace.

Algorithmes

Algorithme de routage

Au cas où le cache ne contient pas une route correspondante, on effectue une recherche du plus long préfixe dans la table LCA. Pour chaque route de la table :

- Application du masque de la route à l'adresse cible.
- Comparaison avec la destination de la route.
- Si correspondance : calcul du nombre de bits à 0 du masque.
- Sélection de la route avec le plus petit nombre de bits à 0 (masque le plus long).

Enfin, on ajoute la route au cache avec le plus long masque existant sur la route en effectuant un remplacement selon la politique de gestion si la taille dépasse la taille maximale.

Gestion du cache

La gestion du cache se fait selon les trois politiques qui permet de réduire les recherches coûteuses, les avantages de chaque politique varient entre eux:

- FIFO: plus simple à implémenter.
- LFU: utile sur trafic stable.
- LRU: souvent plus réaliste.

Conclusion

Les choix de conception retenus permettent d'illustrer clairement l'impact des structures de données et des politiques de cache sur les performances d'un routeur. L'architecture modulaire et la démarche de raffinage ont facilité le développement, les tests et la comparaison des différentes solutions, tout en assurant un code structuré, évolutif et conforme aux objectifs du projet.

Merci!