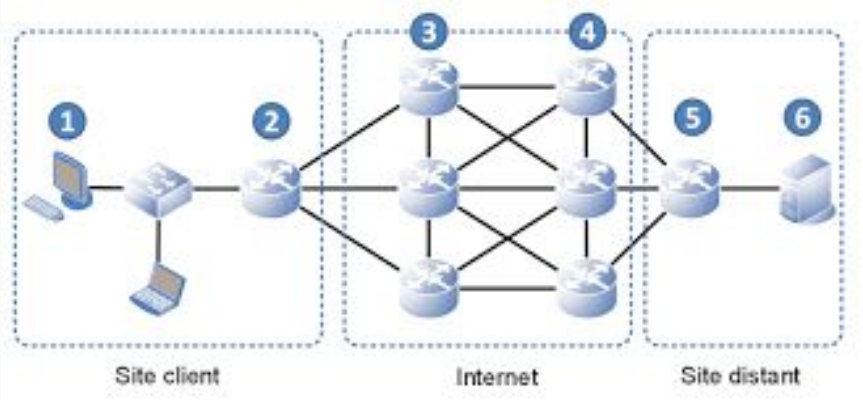
Présentation PIM Projet 3

Chloé, Fantin, Hamza

Visualisation du sujet : Routage

Le routage consiste en la navigation de paquet à travers un réseau de routeurs. Le but est de réussir à envoyer chaque paquet d'un point A à un point B. Pour ça, nous possédons une table de routage qui contient l'ensemble des routes possibles (départ, arrivée, étapes). Pour chaque trajet, on cherche le chemin correspondant.



Détail du fonctionnement

Les différentes étapes :

Comparaison : On compare l'adresse de destination du paquet avec chaque couple (adresse destination, masque) de la table.

Critère de sélection : Une ligne convient si les bits de l'adresse du paquet sont égaux à ceux de la destination de la table pour tous les bits fixés à 1 dans le masque.

Règle du masque le plus long : Si plusieurs lignes correspondent au paquet, c'est impérativement celle qui possède le **masque le plus long** qui est sélectionnée.

Ajout du Cache :

Fonctionnement : Le routeur interroge d'abord le cache. S'il y a un "défaut de cache" (information non trouvée), il consulte la table complète, puis insère la nouvelle route dans le cache.

Gestion de l'espace : Comme le cache a une taille limitée, des politiques d'éviction sont prévues pour supprimer des données anciennes ou peu utilisées : **FIFO** (premier entré, premier sorti), **LRU** (moins récemment utilisé) ou **LFU** (moins fréquemment utilisé).

Implantation technique :

Liste chaînée (L) : Une structure simple où les routes sont stockées les unes après les autres.

Arbre préfixe ou Trie (A) : Une structure plus performante où chaque nœud représente un bit (0 ou 1) de l'adresse IP.

Détail de l'architecture : routage

Module Principal : routage (simple, listes ou arbre)

1. Module Principal : `Routage`

Rôle : Initialiser les structures, traiter les fichiers d'entrée et libérer la mémoire.

Fonctions principales :

Configuration : Appelle `Gerer_commandes` pour configurer les paramètres (taille du cache, politique `-p`, fichiers `-t`, `-q`, `-r`).

Chargement : Charge la table de routage en mémoire via `Table_routage`.

Exécution : Ouvre les flux de données et lance la procédure `Traiter_les_paquets`.

Nettoyage : Assure la destruction des listes chaînées (`Detruire`) pour éviter les fuites mémoire.

routage simple :

utilise uniquement une LCA, n'a pas de cache, n'est pas le plus performant

routage cache liste :

ajout d'un cache basé sur le module Listes, politique de remplacement (fifo,lru,lfu).

routage cache arbre :

le cache devient un arbre, c'est le plus performant

Détail de l'architecture : Fonctions_globales

Module Principal : Fonctions_globales

2. Module de Logique Métier : `Fonctions_globales` (`.ads` / `.adb`)

Rôle : Faire le pont entre les fichiers textes et les types de données Ada (TAD).

Fonctions principales :

`Id_ad_IP` : Convertit une adresse IP au format `String` (ex: "192.168.1.1") en un type modulaire `T_Adresse_IP` (32 bits).

`association_ad_des` : La fonction la plus critique. Elle vérifie si l'adresse est dans le cache ; si non, elle parcourt la table de routage pour trouver l'interface correspondante.

`Identifier_commande` : Analyse les lignes du fichier de paquets pour détecter les commandes de débogage (`table`, `cache`, `stat`).

`Gerer_commandes` : Parse les arguments de la ligne de commande (gestion du mode verbeux, statistiques, etc.).

routage simple :

traiter les paquets consulte d'abord le cache

routage cache liste/arbre :

ajout des paramètres cache, politique, cache_taille. apparition de ajout_cache et élimination

routage cache arbre :

le cache devient un arbre, c'est le plus performant

Détail de l'architecture : Listes (propre à cache listes)

Module Principal : Listes

3. Module de Structure de Données : `LISTES` (`.ads` / `.adb`)

Rôle : Fournir une implémentation robuste de liste chaînée.

Fonctions principales :

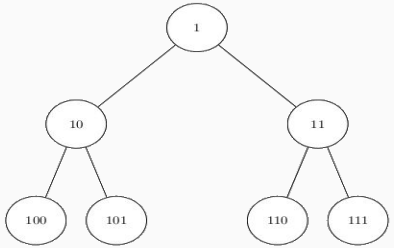
`association_liste` : Implémente l'algorithme **LPM (Longest Prefix Match)**. Elle compare l'IP avec le masque de chaque route et sélectionne celle ayant la correspondance la plus longue.

`Elimination` : Gère la libération d'espace dans le cache. Selon la politique choisie, elle appelle `Elimination_FIFO`, `Elimination_LRU` ou `Elimination_LFU`.

`Ajout_routeur` : Insère une nouvelle route ou une nouvelle entrée de cache dans la liste.

Détail de l'architecture :
Cache (propre à arbre et listes)

Module Principal : Cache



4. Module de Gestion Spécifique : Cache (`.ads` / `.adb`)

Rôle : Optimiser l'accès aux routes les plus fréquentes ou récentes.

Fonctions principales :

`Rechercher_Dans_Cache` : Tente une correspondance exacte pour gagner du temps par rapport à un parcours de table.

`Deplacer_En_Tete` (LRU) : Réorganise la liste à chaque accès pour placer l'élément utilisé au début.

`Incrementer_Frequence` (LFU) : Met à jour le compteur d'utilisation d'une route.

routage simple :

pas de cache

routage cache liste :

cache en liste

routage cache arbre :

cache en arbre

Détail de l'architecture : Exceptions

Module Principal : Exceptions

5. Modules de Robustesse : `Exceptions`

Le projet utilise plusieurs fichiers d'exceptions (`routeur_exceptions.ads`, `cache_exceptions.ads`, `sda_exceptions.ads`) pour traiter les erreurs proprement :

`Fichier_Inconnu_Error` : Si `table.txt` ou `paquets.txt` manque.

`Taille_Cache_Error` : Si la taille du cache est incohérente.

`Commande_Inconnu_Error` : Si une commande invalide est lue dans le fichier de paquets.

Pour les routages avec cache, les exceptions sont bien évidemment adapté pour les erreurs lié aux différents caches.