Projet Python :

26/04/2024

Stratégies de gestion de flux à l'entrée d'un un réseau de communication

IN407 : Concepts avancés de programmation

MENDES Rafael
ASSIS Hugo

Introduction :

Le projet étudie un réseau de communication à commutation de paquets, où les paquets provenant de diverses sources peuvent dépasser la capacité de transmission du lien, entraînant un stockage temporaire dans un buffer. Si ce buffer atteint sa capacité maximale, les paquets excédentaires sont perdus.
L'arrivée des paquets suit un processus de Poisson.
L'objectif principal est de développer une application pour comparer différentes stratégies de gestion de flux à l'entrée du réseau. Le projet se divise en deux parties distinctes.

PARTIE 1 : Structure abstraites de données

Dans cette première phase, nous envisageons le système comprenant les sources, la file d'attente et le lien de transmission. L'objectif est de développer une implémentation orientée objet de ce système.
En ce qui concerne le contexte, nous utilisons des paquets ainsi que leur taille. Les méthodes implémentées permettent de supprimer les paquets vides (ayant une taille de 0), de vider la file en fonction d'un nombre maximal représentant la vitesse du lien (en utilisant la méthode "transmit_packets"), de surveiller le niveau de remplissage, et enfin de calculer le taux de perte. À chaque ajout de paquet, nous vérifions s'il y a suffisamment de place dans la file. Si c'est le cas, le paquet est ajouté, sinon il est considéré comme perdu. Nous tenons compte des succès et des pertes à l'aide d'un dictionnaire.

PARTIE 2 : Structure abstraites de données

Dans cette seconde phase du projet, nous instaurons des buffers individuels pour chaque source S_i, dotés d'une capacité C_i. Ces buffers servent à stocker les paquets émis par chaque source avant leur acheminement vers la file d'attente B. Les paquets de chaque source S_i parviennent à leur buffer B_i selon un processus de Poisson caractérisé par le paramètre λ_i. L'objectif principal est d'évaluer et de comparer les performances de différentes stratégies de traitement des paquets au sein des diverses files d'attente.

Méthodologie :

Pour ce faire, on crée plusieurs classes, comme la classe Source, la classe Buffer, la classe Paquet et enfin la classe Graph.

- La classe Buffer offre des opérations de type FIFO (First In, First Out), ce qui signifie que les éléments sont retirés dans le même ordre où ils ont été ajoutés. Elle est idéale pour la gestion d'une file d'attente. Elle utilise une liste pour stocker les éléments, conserve le suivi du dernier élément retiré et peut être configurée avec une taille maximale.
- La classe Source représente une source de paquets capable de générer une liste de paquets selon la loi de Poisson, déterminée par un paramètre λ (lambda).
- La classe Packet représente de manière rudimentaire un paquet de données. Cette classe simple contient deux attributs : la taille du paquet et éventuellement un identifiant, permettant par exemple de distinguer la source du paquet.
- La classe "Graph" est la plus complète et la plus complexe de toutes. Il s'agit d'un widget personnalisé qui regroupe plusieurs éléments. Elle comprend une barre de progression permettant de visualiser le taux de remplissage, des étiquettes pour afficher des informations, des curseurs pour modifier les variables, ainsi que les variables Tkinter associées. Cette classe utilise la classe "Buffer" pour le stockage des données et pour obtenir les informations nécessaires.

Problèmes rencontrés :

Les défauts des classes Buffer et Source sont les suivants :
- Chaque Buffer est associé à une seule Source.
- Un Buffer inclut la Source alors qu'ils devraient être séparés.
- On crée un Buffer pour chaque Source ainsi qu'un Buffer principal.
- Il devrait être possible de connecter les Sources au Buffer principal.

Solutions Trouvées :

Nous avons envisagé de fusionner la partie graphique des classes Buffer et Source tout en les distinguant. Nous pourrions explorer l'héritage de classe, où la partie graphique serait la classe de base, tandis que les fonctions et méthodes seraient des sous-classes héritant de cette partie graphique. La classe de base ne serait plus nommée Buffer, mais autrement, laissant ainsi la possibilité de créer les classes Buffer et Client. Il ne nous restait plus qu'à développer les stratégies d'évacuation, où le buffer principal sélectionne les clients à évacuer en fonction de certains critères.

Conclusion :

Ce projet nous a permis d'explorer les réseaux de communication à commutation de paquets. À travers notre application, nous avons testé différentes façons de gérer le flux de données, ce qui nous a donné un aperçu des performances de chaque approche. Malgré quelques défis rencontrés, comme la conception des structures de données, nous avons trouvé des solutions et exploré de nouvelles idées pour améliorer notre application. En fin de compte, cette expérience nous a aidés à mieux comprendre les réseaux de communication et à développer nos compétences en programmation.