LINGI1341

RÉSEAUX INFORMATIQUES

TRTP: TRUNCATED RELIABLE TRANSPORT PROTOCOL

Rapport 1 - Implémentation

Auteurs: Thomas des Touches - 38571500 Pierre Lamotte - 65441500



Table des matières

1	Introduction	2
2	Architecture	2
3	Timestamp	3
4	PTYPE_NACK	3
5	Timeout	3
6	Partie critique	3
7	Performances	3
8	Stratégies de tests	4
9	Interopérabilité	4

1 Introduction

Afin de répondre à la demande de l'entreprise **Crousti-Croc**, il nous a été demandé de créer un programme composé d'un serveur et d'un client. Ces deux entités communiquant à distance au moyen du protocole UDP, nous avons dû générer un programme permettant d'assurer la bonne réception des paquets envoyés d'un hôte à l'autre.

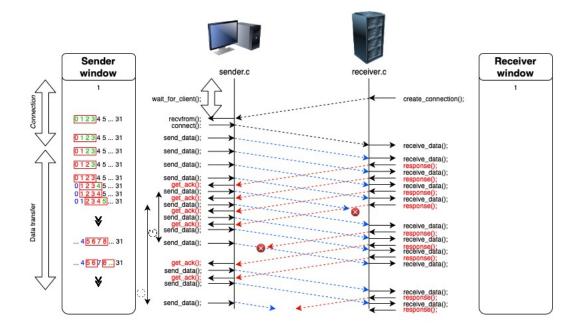
Ce rapport accompagne le code à fournir et permet de clarifier et visualiser certains aspects du projet.

Le code n'étant pas opérationnel lors de cette première soumission, le but principal de ce rapport est d'apporter les éléments principaux de notre réflexion.

2 Architecture

Afin de créer un programme efficace, nous avons d'abord étudié l'architecture de ce dernier. Cette étude nous a permis de mieux percevoir les actions à réaliser par le programme, tant au niveau client que serveur.

L'évolution de la fenêtre d'émission, l'accusation de réception des paquets ou encore l'évolution de l'horloge (voir section 5) sont affichés sur la figure suivante afin de mieux concevoir les actions du programme au cours du temps.



La figure précédente étant incomplète, plusieurs éléments ne sont pas définis (e.g. : la fenêtre du receveur, la déconnexion,...). Ces éléments seront précisés dans le second rapport.

3 Timestamp

Nous avons choisi d'utiliser le timestamp afin de transmettre des informations sur le réseau. Ce champ contient l'heure à laquelle le paquet a été envoyé afin de définir la durée de l'horloge (voir section 5).

4 PTYPE_NACK

La réception d'un non acquittement par le serveur lui idique que le paquet correspondant au numéro de séquence fourni a été tronqué sur le trajet. Suite à cela, le serveur renvoie immédiatement (si sa fanêtre d'émission le lui permet) le paquet de données correspondant au non acquittement.

5 Timeout

Afin de déterminer et d'initialiser la valeur du *timeout*, nous avons déterminé les bornes supérieures et inférieures de celui-ci.

La borne supérieure étant définie par la latence du réseau à laquelle s'ajoutent le temps de traitement du packet reçu et une marge de sécurité.

La borne inférieure est définie par le RTT du dernier packet envoyé. Une marge de sécurité est également appliquée, fixée arbitrairement à 15% du RTT. Ces deux valeurs nous donnent alors :

MAX_TIMEOUT: 2 * 2000 + 100 = 4100 ms**MIN_TIMEOUT**: LAST_RTT¹ * 1.15 ms

En partant de ce postulat, nous avons décidé d'optimiser l'horloge en la faisant décroître et croître comme suit :

Décroissance : NEW_TIMEOUT = (OLD_TIMEOUT + LAST_RTT) / 2

Croissance: NEW_TIMEOUT = 2 * OLD_TIMEOUT

Ces croissance et décroissance étant respectivement bornées par MIN_TIMEOUT et MAX_TIMEOUT.

6 Partie critique

Nous n'avons malheureusement pas eu le temps de définir la partie critique du programme.

7 Performances

Nous n'avons malheureusement pas été capables d'effectuer les test de performances par manque de temps.

^{1.} Temps passé entre l'émission du dernier packet et de son accusé de réception

8 Stratégies de tests

Notre binôme a décidé d'effectuer des tests sur une majeure partie des fonctions utilisées telles que celles permettant l'encodage ou le décodage des paquets, la récupération de l'adresse IPv6, la connexion, etc. Les test sont menés de sorte à vérifier que les fonctions soient robustes. Ces test nous permirent également de corriger les erreurs de notre code. Certaines fonctions restent cependant à tester.

9 Interopérabilité

Le principal objectif à atteindre afin d'assurer l'opérabilité étant le traitement du timestamp, nous pensons devoir gérer tous les types d'encodage de cette partie afin d'assurer une interopérabilité optimale. Nous y travaillerons pour la seconde soumission.