IFT585 - Télématique

Travail pratique #1 – Hiver 2104

Date 27/01/2104

Enseignant

Bessam Abdulrazak

Statut Version

En vigueur 1.0.0

Contexte

Le protocole à fenêtre d'anticipation est un protocole basé sur les points suivants :

1. Envoi de plusieurs trames avant réception d'un acquittement.
2. Le nombre de trames autorisées à être envoyées est indiqué par la fenêtre de l'émetteur.
3. Le nombre de trames acceptables par le destinataire est indiqué par la fenêtre du récepteur.

Énoncé

Dans ce TP on s'intéresse au cas où la taille des fenêtres d'anticipation est variable selon le choix de l'utilisateur.

À partir du canevas proposé par Tanenbaum (Protocole 5 et 6), réaliser un programme illustrant le fonctionnement d'un protocole bidirectionnel à tampons multiples et reprises multiples (sélectives et globales.

Votre programme devra être constitué de trois « thread » qui représenteront :

(1) la station émettrice,
(2) la station réceptrice, et
(3) le réseau.

Votre programme devra permettre la copie d'un fichier entre (1) et (2), en utilisant (3) pour simuler le réseau. Les deux threads (1) et (2) devront être utilisés de la même façon (les paramètres et les commandes étant les mêmes).

Votre programme prendra les 4 paramètres suivants :

1. La taille du tampon utilisé de chaque côté du « thread » simulant le réseau (tampon d'envoi).
2. Le délai de temporisation (« time-out ») de l'émetteur (premier « thread »).
3. Le fichier à copier
4. L'emplacement de destination pour la copie du fichier.

La station émettrice

Ce « thread » lit le fichier d'entrée octet par octet, crée une trame réseau de « N » octets (vous pouvez décider du format de la trame) contenant le numéro de la trame, ainsi que le code détecteur/correcteur, et ensuite place la trame de « N » octets dans un tampon d'envoi. Le tampon d'envoi doit être un vecteur (« array ») de taille correspondante au premier paramètre entré dans le programme multiplié par « N » octets (si 20 a été entré et N=5, le tampon est de taille 100 octets pouvant contenir 20 trames de 5 octets). Ce vecteur agit comme une file circulaire (facile à construire avec un modulo %). Lorsqu'il reste de l'espace non utilisé dans le tampon (où que de l'espace se libère), on remplit ces espaces avec de nouvelles informations à envoyer. Dès que le réseau (second « thread ») est disponible pour l'envoi, on envoie une trame sur le réseau (sauf si aucune trame n'a à être envoyée). Toutes les trames qu'on a envoyées restent dans le tampon tant que le « ACK » n'a pas été reçu pour cette trame. Si un « NAK » est reçu pour une trame ou bien que le délai de transmission (« time-out ») est atteint, on envoie de nouveau la trame. Lorsqu'un « ACK » est reçu, mais que le deuxième octet de la trame contenant le « ACK » n'est pas identique au « ACK » du premier octet, ne pas prendre en compte cette trame (ne pas la considérer comme un « NAK »).

La station réceptrice

Ce « thread » attend qu'une trame soit disponible sur le réseau (3) et la lit lorsque disponible. Lorsqu'une trame est reçue, le « thread » de destination va s'assurer que la trame est correcte (dans le cas de code détecteur). Si ce n'est pas le cas, il jette la trame et envoie un « NAK » à la source. Si elle est bonne, il envoie un « ACK ». La gestion des « NAK » et des « ACK » dépend du programme (rejet global ou rejet sélectif) utilisé. Lorsqu'un octet est reçu, le programme l'écrit dans le fichier de destination.

Le réseau

Ce « thread » travaille avec deux variables de « N » octets qui correspondent aux variables « envoie source » et « réception destination » ainsi que deux variables booléennes qui correspondent aux variables « prêt à émettre source » ainsi que la variable « donnée reçue destination ». Lorsque la station émettrice veut émettre, elle regarde si le réseau est prêt à émettre avec la variable booléenne « prêt à émettre source » et utilise une fonction qui place la donnée à émettre dans la variable « envoie source » et met la variable « prêt à émettre source » à faux. Lorsque la variable « prêt à émettre source » est à faux et que la variable « donnée reçue destination » est également à faux, le « thread » (3) va prendre la trame dans la variable « envoie source », va la placer dans la variable « réception destination », et va mettre

les variables « prêt à émettre source » et « donnée reçue destination » à vrai. Lorsque la station réceptrice voudra aller chercher une trame, elle regardera la variable « donnée reçue destination » et si elle est à vrai, elle ira récupérer la trame à l'aide d'une fonction de récupération de la donnée. Cette fonction va retourner le contenu de la variable « réception destination » et va mettre la variable « donnée reçue destination » à faux. Les envois de « ACK » se font de la même manière que l'envoie des octets, mais avec 4 nouvelles variables: « envoie destination », « réception source », « prêt à émettre destination » et « donnée reçue source ». Vous pouvez décider de la structure du « ACK » et du « NAK » dans le l'entête de la trame. Il est important de simuler la latence entre l'émission par (1) d'une trame et sa réception par (2).

Des erreurs de transferts devront pouvoir être introduites dans (3), soit directement à l'exécution du programme, soit avec un fichier d'erreurs donné comme cinquième paramètre. Si les erreurs sont données à l'aide d'un fichier, il est nécessaire de les afficher à l'écran avec les informations d'émission et de réception.

Annexe 1: Consignes supplémentaires :

1. L'usager devra pouvoir introduire des ***erreurs durant le transfert*** du fichier. Vous devrez identifier et prendre en compte les différents types d'erreurs qui peuvent arriver pendant la transmission des paquets.
    - Nous parlons ici de protocoles bidirectionnels. Les threads 1 et 3 sont identiques (émetteur et récepteur au même temps). Le réseau peut induire des erreurs dans les deux sens, c-à-d. lorsque l'émetteur envoie au récepteur et lorsque le récepteur répond à l'émetteur.
    - Nous avons un seul format de trame (et des champs spécifiques) pour envoyer les données et signaliser le ACK et le NAK
2. Vous devrez utiliser un ***fichier de commandes*** pour configurer les paramètres et les commandes. L'utilisateur pourra donc choisir l'emplacement du fichier d'entrée, l'emplacement du fichier de sortie, introduire les différents paramètres (***taille des fenêtres***, code ***correcteur***, rejet ***global ou sélectif*** et ***erreurs***) et simulant le réseau.
    - Le protocole correcteur doit être implémenté en utilisant le code Hamming sur l'ensemble de la trame.
3. Votre programme devra afficher à l'écran les données envoyées et reçues par l'émetteur ainsi que par le récepteur. L'usager doit pouvoir suivre facilement le transfert du fichier.
    - L'affichage peut se faire en mode console ou en utilisant un interface graphique.
4. Vous n'avez pas besoin de gérer l'envoi multiple de fichiers dans ce TP.

Annexe 2 : Livrables

Les livrables suivants doivent être remis via Moodle :

- Un zip programme comprenant une application exécutable sous Windows, le code source complet (fichier .rar, .zip) ;
- Un rapport collectif de travail : Le rapport devra contenir un imprimé décrivant le contenu du zip programme, la procédure d'exécution, une méthode pas à pas pour compiler et exécuter les programmes, un guide d'usager expliquant le fonctionnement de chacun des programmes, une légère analyse expliquant comment vous avez effectué la conception de ces programmes ainsi qu'une indication concernant ce qui fonctionne bien, ce qui fonctionne moins bien et ce qui ne fonctionne pas dans ces programmes (La rédaction de votre travail ne doit pas excéder 8 pages, plus une page de garde précisant le nom et le matricule des auteurs).
- Rapport individuel d'activité : Un rapport personnel d'une page maximum, décrivant la participation du reste de l'équipe, et non pas de son propre travail.

Gabarit de correction

Les points suivants sont octroyés par l'accomplissement des fonctionnalités requises.

Aspect	Points
Rapport explicatif détaillé sur le fonctionnement du programme	20
Rapport individuel d'activité	5
Liste des erreurs possibles	5
Implémentation d'un algorithme de Rejet global	20
Implémentation d'un algorithme de Rejet sélectif	20
Afficher à l'écran les données	10
Implémentation d'un algorithme de Hamming	20
Total	100

Date de remise

Vendredi 28 février 8 :00 PM (À rendre au plus tard le lundi **3 mars 2014 8 :00 AM**).

Annexe 3

Compilation et exécution des programmes

Sachez que vous pouvez développer votre programme dans le langage de programmation C++, C# ou Java. Après la remise, le chargé de correction peut prendre contact avec vous pour fixer un rendez-vous. Lors de ce rendez-vous, le chargé de correction vous fournira le code source que vous avez fourni par soumission automatique et vous devrez présenter la compilation et l'exécution de votre programme.

Soumission des travaux

Il est de responsabilité des étudiants de débuter le travail le plus tôt possible et de pouvoir le soumettre électroniquement avant l'heure d'échéance pour la soumission du travail. L'incapacité de trouver un poste de travail ou de se connecter à distance quelques minutes avant l'heure d'échéance ne sont pas des raisons valables pour justifier un retard.

Tant que la date limite n'est pas atteinte, vous pouvez soumettre autant de fois que vous le désirez : seule la dernière soumission est conservée.

Pénalité pour retard

Les travaux pratiques remis en retard sont sujets à une pénalité. Les travaux seront remis électroniquement via le portail du cours sur Moodle. La note sera réduite de 33% pour chaque tranche de 24h de retard. En conséquence, la note attribuée après 2 jours de retard est zéro. Si votre travail n'est pas terminé à temps, vous devrez notifier le professeur par courrier électronique.

Plagiat

Un document ou un programme dont une partie est tirée d'un livre, d'une publication scientifique ou même d'un site Internet, doit être référencé adéquatement. Lors de la correction du TP une attention spéciale sera portée au plagiat, défini dans le Règlement des études comme « le fait, dans une activité pédagogique évaluée, de faire passer indûment pour siens des passages ou des idées tirés de l'œuvre d'autrui » (voir également le plan de cours).

Il est fortement recommander d'implémenter par vous-mêmes les algorithmes, plutôt que de recopier une implémentation existante de ces algorithmes : l'objectif de ce TP réside dans la compréhension des différents algorithmes à implémenter, et non pas dans le programme résultant.

Directives particulières

La correction des travaux pratiques et des rapports est entre autres basée sur le fait que chacune des réponses soit :

⇒ claire, c'est-à-dire lisible et compréhensible pour le correcteur ;

⇒ précise, c'est-à-dire exacte ou sans erreur ;

⇒ complète, c'est-à-dire que toutes les étapes de résolution du problème sont présentes ;

⇒ concise, c'est-à-dire que la méthode de résolution est la plus courte possible.

La correction des programmes prend en compte la qualité de code et celle de la documentation. Il est fortement recommandé de respecter les normes départementales de programmation. Pour mémoire, ces normes sont disponibles à cette adresse :

http://www.dmi.usherb.ca/~fraikin/cours/Normes/normes-de-programmation.pdf

Le correcteur ou la correctrice peut soustraire jusqu'à 5% de chaque évaluation pour la qualité du français. Des consignes supplémentaires ou des modifications pourront être communiquées au cours du trimestre.

Documentation

[Tanenbaun2011]
TANENBAUM, Andrew S. ;
WETHERALL, David ;
Réseaux, 5e édition ;
Pearson Éducation France, 2011, 970 pages,
ISBN 2-7440-7521-3.

[Kurose2012]
James F. KUROSE, Keith W. ROSS ;
Computer Networking : A Top-Down Approach. 6e édition ;
Addison-Wesley 2012; 864 pages,
ISBN-10: 0132856204
ISBN-13: 9780132856201