IFT3325 Téléinformatique

Devoir 2 - Conception et implantation d'un protocole de liaison fiable simplifié

Automne 2025

- Ce devoir est à rendre au plus tard **le 27 novembre (sur studium), format un dossier .zip**
- Ce devoir est à faire en équipe de deux.
- La politique du retard et de plagiat est énoncée dans le plan du cours.

Objectifs

Ce devoir a pour but de vous faire comprendre et expérimenter les principes de la couche liaison de données, notamment :
- la formation des trames ;
- la détection et la gestion des erreurs (CRC, retransmission) ;
- la maîtrise du flux et la synchronisation entre émetteur et récepteur ;
- l'encodage binaire et le bit-stuffing.

Vous concevrez, implanterez et testerez un protocole de liaison fiable simplifié fonctionnant au-dessus d'un canal non fiable que vous émulez vous-même. If faut expliquer les choix concrets faits dans le code (par ex. durée de temporisation, taille de fenêtre). Les réponses purement théoriques ou génériques ne seront pas acceptées.

Contexte général

Deux entités communiquent à travers un canal pouvant :
- introduire des erreurs de bits,
- perdre des trames ou des acquittements,
- et ajouter un délai aléatoire avant la réception.

Votre protocole doit assurer la livraison correcte et ordonnée des trames malgré ces perturbations. Vous pouvez mettre en œuvre un protocole fiable inspiré des principes de HDLC, appliqué à un canal simulé susceptible d'introduire des erreurs.
Il s'inspire des protocoles HDLC, mais vous êtes libres d'adapter vos choix.

1. Conception du protocole (2%)

La conception repose sur plusieurs choix structurants détaillés ci-dessous. Chaque choix doit être accompagné d'une justification chiffrée ou expérimentale.

Définissez :
1. Format de la trame : champs (flag, en-tête, données, CRC), longueur et justification.

2. Mécanisme de contrôle : choix entre Go-Back-N / Selective Repeat (justifier votre choix), taille de fenêtre, durée du timeout.
3. Détection d'erreurs : CRC-16 ou CRC-32, justifiez votre choix.
4. Méthode d'encapsulation : ajout des flags 01111110 et bit-stuffing.
Incluez un schéma clair du format de trame.

2. Émulation du canal (2%)

Le canal est un module séparé simulant un lien non fiable.

Paramètres configurables :
probErreur = 0.05
probPerte = 0.10
delaiMax   = 200   # en millisecondes

Chaque trame ou ACK passe par le canal ; il peut être altéré, perdu ou retardé.

Remarques :
- Le canal introduit un délai aléatoire mais préserve l'ordre des trames.
- Le délai simule la latence locale du lien.
- Le timeout doit être supérieur au délai maximal pour éviter les retransmissions inutiles.
- Vous réaliserez un test spécifique où le délai devient supérieur au timeout.

3. Implantation du protocole (6%)

Langage au choix : Java, Python ou C.
Le protocole doit afficher les principaux événements : envoi, réception, ACK, timeout, retransmission.

Joindre une capture de l'écran du terminal montrant ces événements avec horodatage. Mentionner la version du langage et du système utilisé. Toute soumission sans sortie d'exécution réelle (logs, captures) sera considérée incomplète.

Message à transmettre :
- Fichier texte message.txt d'environ 5 à 10 Ko (ASCII).
- Même fichier pour tous les scénarios.
- Segmentation en trames de 100 octets maximum (hors en-tête et CRC).
- Le récepteur recompose et compare le message reçu au fichier original.

Sortie finale attendue :
Transmission terminée.
Frames envoyées : XX
Frames retransmises : YY
ACK reçus : ZZ
Durée totale : TT s

4. Tests expérimentaux (3%)

Scénarios :
1. Canal parfait (probErreur=0, probPerte=0)
2. Canal bruité (probErreur≈0.05, probPerte≈0.1)
3. Canal instable (probErreur≈0.10, probPerte≈0.15, delaiMax=300 ms)

4. Influence du délai sur les temporisations :
Fixez timeout = 200 ms et variez delaiMax :
- 50 ms < timeout → aucune retransmission,
- 180 ms ≈ timeout → quelques faux timeouts,
- 300 ms > timeout → nombreuses retransmissions.

Pour chaque scénario, les étudiants doivent reporter :
– le nombre total de trames envoyées, reçues, retransmises, et confirmations
– la durée totale mesurée.
Ajouter une capture d'écran du terminal par scénario.
Une brève interprétation (3–5 lignes) doit accompagner chaque tableau de résultats.

5. Bit-stuffing (2%)

Avant d'ajouter les flags 01111110, appliquez le bit-stuffing HDLC :
après cinq 1 consécutifs, insérez un 0.
Le CRC est calculé sur en-tête + données (avant stuffing), puis ajouté ;
le stuffing est ensuite appliqué sur en-tête + données + CRC.
Le récepteur :
1. détecte les flags ;
2. extrait la zone utile ;
3. retire les 0 insérés après cinq 1 ;
4. vérifie le CRC.

Incluez des tests montrant : insertion correcte du 0, restauration exacte après dé-stuffing, et détection d'une corruption du bit stuffed.

Test de validation du bit-stuffing

Flux d'entrée :
01111110111111011111110111110

Résultat attendu après stuffing :
011111010111110011111001111010111110 0

Le dé-stuffing doit restituer exactement le flux d'entrée.

À inclure dans le rapport : le flux avant stuffing, après stuffing et après dé-stuffing, ainsi que la vérification que CRC(original) = CRC(destuffed). Les flux binaires avant/après stuffing et dé-stuffing doivent provenir du programme exécuté (capture d'écran).

7. Discussion et analyse (3%)

Expliquez :

1. Expliquez comment votre implémentation réagit réellement quand un ACK est perdu (donnez un extrait de log; capture d'écran).
2. Quelle différence observez-vous entre votre code et le comportement théorique attendu?
3. Si vous deviez optimiser la fiabilité, que changeriez-vous ?

8. Livrables

Archive .zip contenant :
archive.zip

```
│
├─── code/
│    ├─── canal.py
│    ├─── protocole.py
│    └─── stuffing.py
│
├─── message.txt (fichier de test utilisé)
├─── rapport.pdf (3 pages max; schéma de la trame, description et résultats, tests de bit-
                  stuffing, discussion sans oublier les captures d'écran)
└─── README.txt (instructions et paramètres)
```

Chaque module doit être autonome et clairement commenté. Les projets remis sous forme d'un seul fichier seront pénalisés.

Authenticité et présentation orale (2%)

Chaque équipe indique ses noms et matricules, fait une démonstration (5–10 min) et répond à 1–2 questions du TA.

Conseils

- Commencez simple : canal parfait → bruité → instable.
- Affichez clairement les événements et statistiques.
- Testez avec 50–100 trames pour observer des erreurs.
- Vérifiez stuffing/destuffing avec le test fourni avant toute simulation.