Université de Caen Normandie
U.F.R des sciences
L2 – informatique

Réseaux 1
Couche liaison de données

Introduction

Nous allons implémenter, dans ce TP, des solutions pour communiquer via la couche 1 et 2 du modèle **OSI** en développant un simulateur d'un réseau de communication. La couche 2 vise à échanger des trames de données en s'appuyant sur la couche 1. On passe d'un signal brut (i.e. la séquence de ± sur le premier TP) à des messages structurés, dont on peut reconnaître le début et la fin, l'émetteur et le destinataire. La couche 2 couvre aussi le transfert correct de ces trames de données au niveau local (sans intermédiaires identifiés): vérifier que les messages ont bien été reçus, corriger les erreurs, partager les canaux de communication etc.

Sur ce TP, nous allons mettre en pratique la structuration des trames pour couvrir les besoins de la couche2.

Nous aurons à notre disposition le contenu du niveau 1 du modèle OSI, qui reprend les travaux du TP1. On peut échanger des bits de donnée via une transformation en signal avec relativement peu d'erreurs.

Simulateur d'un réseau de communication

Afin de bien effectuer le lien avec le premier TP, on va commencer par mettre en place un environnement de simulation qui distingue les différents éléments d'un réseau. Cet environnement de simulation est disponible sur Ecampus.

- Étudiez l'algorithme «Simulation». Cet algorithme simule le transfert complet d'un message en passant par la couche 2 puis la couche 1 du modèle **OSI**. Cet algorithme contient trois parties : **émission**, qui simule le travail de l'émetteur; **transmission**, qui simule les effets causés par la transmission (ex: **bruit aléatoire sur le canal**); **réception**, qui simule le travail du récepteur.
- «Émission» prend en entrée en entrée un message (message représenté par «o» et «i» où«o» est représenté par «0» et «i» est représenté par un «1»); il effectue les opérations relatives à la couche 2: transformer le message en une séquence de bits de sorte à ceque le récepteur puisse, à partir de cette même séquence de bits récupérer le message initial.Puis, il effectue les opérations relatives à la couche 1: transformer la séquence de bits en un signal, de sorte à ce que le récepteur puisse, à partir de cette même séquence de signaux récupérer le message initial. Cette seconde fonction correspond au codage **NRZ**,rédigé pendant le TP1
- «Transmission» simule les effets causés par la transmission: le fait qu'il y ait déjà des signaux qui résident sur le canal, les erreurs possibles.
- «Réception» simule le travail du récepteur. Il effectue les opérations inverses de l'émetteur: transformer signaux en une séquence de bits (couche 1); transformer une séquence de bits en un message (couche 2)

Architecture du simulateur Réseaux (modèle OSI)

1. Étendre couche2, pour que la fonction ajoute la bordure «01111110» au début du message
2. Améliorez couche2R pour supprimer une partie inutile de la trame.
3. Testez la transmission sur le message «oooiii»
4. Étendre couche2, pour que la fonction ajoute au début de la trame la taille du message à envoyer, en binaire sur 8 bits
5. Étendre couche2R, pour que la fonction retrouve le message initial
6. Testez la transmission sur le message «oooiii»
7. Étudier la quantité de signaux échangés par rapport à la taille du message initial
8. Intégrer les fonction de codage du TP1 qui permettent de réaliser le codage NRZI et éventuellement le codage Manchester
9. Nous supposons vous envoyez des impulsions (ex: signal WiFi) au lieu de « + et - », quelles parties auriez-vous à ré-écrire? Qu'est-ce que cela implique pour les couches supérieures?

Adressage des machines

Il arrive fréquemment que plusieurs systèmes écoutent le même canal de communication, ex: ondes écoutées par plusieurs systèmes, répétiteurs qui retransmettent tous les signaux à toutes les machines connectées.Le niveau 2 du modèle **OSI** est en charge de déterminer qui envoie un message et pour qui, en local (connexion directe entre l'émetteur et le récepteur, les connexions qui demandent de passer par des intermédiaires identifiés sont gérées par la couche 3.

- Écrire un algorithme «AddIdentification» qui ajoute l'adresse de l'émetteur et du récepteuren binaire, sur quatre bits par adresse.
- A partir de maintenant, l'émetteur est la machine 1 etle récepteur est la machine 3•Écrire un algorithme côté récepteur «WhoIsTheSender» qui indique le nom de l'émetteur.
- Afficher cette information quand un message est reçu
- Écrire un algorithme «IsAddressedToMe» qui vérifie si un message reçu est bien adresséau récepteur. Si ce n'est pas le cas, retourner «message ignoré»

- Envoyer un message vers la machine 2. Constater que le message est bien envoyé par l'émetteur et ignoré par le récepteur.
- Envoyer le même message sur la machine 3, constater qu'il est bien intégré par le récepteur.

Détection et Correction d’erreurs

Les séquences de bits provenant de la couche 1 sont parfois erronées. Afin de transmettre des données fiables, la couche 2 doit détecter, voire même réparer ces erreurs.

1. Écrire un code «SignalSwapError» qui prend une séquence de signaux et un entier n et qui altère le n-ième signal de la séquence
2. Dans Simulation, appliquer «SignalSwapError» entre le signal émis et le signal reçu, sur un signal de la séquence de signaux qui code le message (ex: le 30ème). Ne pas essayer cela sur un bit de l'entête (les erreurs sur l'entête sont détectées autrement, notamment parce que l'émetteur ne reçoit jamais l'**ACK**)
3. Ré-exécutez le code pour transférer «1111011100001011010», constater qu'une erreur s'est glissée ! Appliquez l'algorithme «parity_bit» pour détecter l'erreur. Si une erreur est détectée, retournez «erreur de transfert» au lieu du code à transférer. Notez qu'il faut appliquer l'algorithme à l'envoi et à la réception et rajoutez un bit à la trame
4. Rajoutez une autre erreur dans le signal avec «SignalSwapError». Constater que «parity_bit» échoue à trouver l'erreur.
5. Rédigez l'algorithme «double_parity» pour détecter les erreurs. Comparer avec «parity_bit»
6. Rédigez l'algorithme «CRC» pour détecter les erreurs. Comparer avec le « double_parity »
7. Enlevez le second SignalSwapError.
8. Rédigez l’algorithme de correction de Hamming pour réparer les erreurs
9. Comparer les différentes solutions en termes de coût d'envoi et d'efficacité à détecter les erreurs