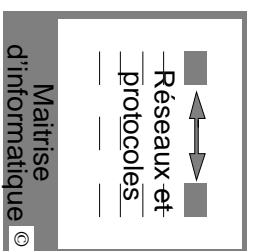


Conception de protocoles



1

But

L'objectif de ce TD est de développer un protocole réalisant un transfert de données fiable et performant entre deux équipements informatiques. Nous verrons que ce problème n'est pas trivial dans la mesure où le support de transmission produit des erreurs, occasionne des pertes, possède des caractéristiques contraignantes (débit, délai, etc.). Enfin, les équipements interconnectés possèdent leurs propres limitations (capacité mémoire, vitesse de traitement, etc.). Ce TD montre comment l'apparition progressive de problèmes simples amène à la création de protocoles de type HDLC.

1 Introduction

L'échange de données entre équipements informatiques distants est possible au niveau Physique et permet la transmission de suites de symboles caractérisée par des propriétés (débit, délai, taux d'erreurs, etc.). Cependant, ces techniques ne sont pas suffisantes pour assurer un transfert correct de la totalité des données. D'une part, les applications peuvent nécessiter un taux d'erreurs inférieur à celui proposé intrinsèquement par le circuit de données. D'autre part, le contrôle de la liaison de données nécessite de la part des équipements informatiques l'échange d'informations permettant : de donner à la liaison de données une configuration adaptée, d'établir et de rompre cette liaison de données, etc. Il apparaît donc indispensable de développer un protocole assurant un transfert efficace et correct des données et un échange des informations nécessaires à la coopération des deux équipements distants.

2 Structures de données et procédures prédéfinies

Les protocoles de communication que nous étudierons utilisent des déclarations de procédures communes et les structures de données suivantes : Message, Typetrame, Trame et Ctl.

Dans l'architecture de base pour l'interconnexion des systèmes ouverts, le niveau 2 (Liaison de Données) reçoit les informations à transmettre (messages) en provenance de l'utilisateur ou du niveau supérieur à l'aide de la procédure `recmess()`. Ces informations après ajout de l'information de contrôle d'erreur (donnée par `calcul()`) sont encapsulées dans une structure appelée trame (unité de données de

1

niveau 2). Cette trame est envoyée à l'entité distante en utilisant la procédure `em_trame()` qui émet la trame ainsi constituée vers le niveau inférieur. L'entité de niveau 2 distante reçoit cette trame du niveau inférieur en utilisant la procédure `rec_trame()`, vérifie la zone de contrôle (`ctrl_f()`) et remet le message reçu au niveau supérieur (`rem_mess()`).

Afin de simplifier la programmation du protocole élémentaire N°1, nous programmons à l'aide d'une boucle infinie. L'entité réceptrice est dédiée à la gestion du canal de transmission et attend en permanence l'arrivée d'un événement. Notons qu'une situation plus réaliste (utilisée pour le protocole N°2) consisterait à gérer la réaction aux différents événements par l'intermédiaire d'un mécanisme d'interruption. Les événements pouvant générer un traitement sont : la réception d'un message du niveau supérieur (événement RECMESS), la réception d'une trame du niveau inférieur (événement RECTRAME) ou alors le déclenchement d'un temporisateur (événement TIMER). La procédure `arrtimer()` permet d'arrêter un temporisateur et la procédure `desarrtimer()` permet de le désarrêter.

3 Un protocole élémentaire (N°1)

Prenons comme exemple initial un protocole très simple : transmission unidirectionnelle, support de transmission sans erreurs et absence de contrôle de flux. Pour programmer un tel protocole (peu réaliste), il faut écrire les procédures `emetteur()` et `recepteur()` du niveau 2.

Question 1 Comment proposez-vous de délimiter les trames ? Quel mécanisme proposez-vous de mettre en place pour autoriser la transmission transparente des données ? Donnez le format des trames utilisées par ce protocole.

Question 2 Donnez un exemple de programmation du protocole N°1. (La procédure "wait()" permet d'attendre un événement.)

4 Le protocole "envoyer et attendre" (N°2)

Nous devons prendre en compte le problème du contrôle de flux : l'émetteur doit transmettre à une vitesse inférieure à la capacité de réception du récepteur. Ceci afin d'adapter la vitesse d'émission à la vitesse de traitement du récepteur et d'éviter l'engorgement de la mémoire de ce dernier.

Question 3 Donnez une solution simple au problème de contrôle de flux permettant d'assurer l'émetteur au récepteur. Donnez le nouveau format de la trame.

On distingue maintenant deux types de trames : les trames d'information et les trames d'acquiescement. Revenons à la situation plus réaliste où le support de transmission provoque des erreurs.

Question 4 Comment détectez-vous une trame erronée ? Que faites-vous de la trame erronée ? Comment proposez-vous d'effectuer la reprise sur erreur dans ce cas ? Faites-vous une différence entre une trame erronée (la zone de contrôle est incorrecte) et une trame perdue ? Comment peut-on perdre une trame ? Traitez-vous ce dernier cas ?

Question 5 On décide de superposer un mécanisme de surveillance par temporisateur au mécanisme de détection déjà mis en place. Discutez de l'implémentation de ce temporisateur (chez l'émetteur ou chez le récepteur). Comment dimensionner la durée de ce temporisateur. Que se passe-t-il si celui-ci est trop court, trop long ?

2

Question 6 Notre choix s'est orienté vers un mécanisme de reprise par temporisation et par acquittement positif (trame ACK) et négatif (trame NACK) consécutif à toute trame d'information reçue. Lors d'une perte de trame d'acquiescement ou de rejet, peut-on observer, au niveau du récepteur, des pertes ou des duplications de trames de données ?

Les caractéristiques du protocole N°2 sont :

- une transmission unidirectionnelle,
- un contrôle de flux sur une seule trame,
- une détection d'erreurs à l'aide de la zone de contrôle,
- une détection des pertes à l'aide d'un temporisateur,
- seules les trames de données peuvent être perdues,
- une correction par acquiescement et/ou rémission.

Question 7 Proposez un format de la trame pour le protocole N°2. Donnez un exemple de programmation de ce protocole. Proposez d'abord un automate pour l'émetteur et un automate pour le récepteur. Soit la variable event mise à jour par le contrôleur d'interruption et qui permet de savoir quel événement a généré l'interruption.

La classe des protocoles de type "envoyer et attendre" regroupe toutes les fonctions classiques d'une procédure de transmission. La caractéristique essentielle de cette classe de protocoles consiste à ne transmettre une nouvelle trame de données que lorsque la précédente a été acquittée. Une copie de la trame en cours de transmission est conservée en mémoire jusqu'à l'obtention de l'accusé de réception positif. La re-transmission a lieu à la réception d'un acquiescement négatif ou à la suite d'un dépassement d'un délai de garde depuis l'émission de la trame. Dans ce dernier cas, appelé "reprise par temporisateur", on supposera que la trame est perdue (bien qu'elle ait pu être seulement très retardée !).

La limitation de ce protocole provient de l'impossibilité d'associer les acquiescements aux trames (cf question 7?). Le protocole dit du "bit-alterné" corrige ce dernier point (numérotation modulo 2).

5 Le mécanisme d'anticipation (N°3)

Un protocole de type "envoyer et attendre" employé sur un support de transmission dont le temps de propagation est très important est inexploitable. Dans l'hypothèse d'une transmission par satellite, le délai de propagation aller est égal à 270 ms. Nous choisissons une vitesse de transmission de 1Mb/s et une taille de trame de 1000 bits. Nous supposons les temps de traitement négligeables.

Question 8 Indiquez dans ce cas le déroulement temporel des opérations ainsi que le pourcentage de temps pendant lequel l'émetteur est improductif.

Question 9 Indiquez comment on peut améliorer l'utilisation du support de transmission en modifiant le protocole. Donnez, pour l'exemple précédent, la paramétrisation du protocole qui conduit à ne jamais bloquer l'émetteur.

Cette technique est appelée mécanisme d'anticipation par "fenêtre coulissante". La fenêtre de l'émetteur indique à tout instant l'ensemble des trames qui ont été émises, sans être encore acquittées.

La fenêtre du récepteur indique à tout instant l'ensemble des trames en attente de réception.

Question 10 Si l'émetteur peut transmettre W trames sans attendre d'acquiescement, que se passera-t-il si la trame i ($0 \leq i < W$) est reçue erronée ? Proposez plusieurs mécanismes permettant au récepteur de faire savoir à l'émetteur l'apparition de cette erreur.

Question 11 Comme précédemment, la détection d'erreur s'effectue aussi par l'intermédiaire d'un mécanisme de temporisation. Combien de temporisateurs et combien de temps nécessaires au minimum ce protocole ?

Question 12 La numérotation des trames se faisant modulo N , connaisant la largeur maximum de la fenêtre (W), indiquez combien de trames de données peuvent être transmises par l'émetteur sans attendre d'acquiescement et sans possibilité de confusion.

Question 13 Dans le cas d'une transmission bidirectionnelle, comment peut-on utiliser rationnellement le flot de données inverse pour acquiescer les trames d'un flot de données ?

Ce protocole se rapproche des protocoles de haut niveau (type HDLC) et permet la réalisation des fonctions comme la transmission bidirectionnelle, le contrôle des erreurs de transmission et l'émission par anticipation (fenêtre).

6 Généralités

Il est possible de corriger automatiquement les erreurs de transmission moyennant une redondance importante donc une taille du champ de contrôle non négligeable par rapport à la taille des données à transmettre. Dans ce cas, il n'est pas nécessaire d'effectuer une rémission de la trame erronée.

Question 14 Indiquer dans quelle(s) circonstance(s) un tel protocole serait préférable à un protocole avec reprise par rémission ?