Master 1 TIIL 2016/2017

Algorithmes Distribués - Projet

Routage par inondation

E. Fabiani

1 Introduction

L'objectif de ce projet est de mettre en œuvre et de comparer différentes versions améliorées par réduction des messages transmis. Tous les algorithmes mis en œuvre doivent être conformes au modèle synchrone. Pour rappel, le fonctionnement de base du routage par inondation consiste, lorsqu'un routeur reçoit un message, à le retransmettre à tous ses voisins.

Dans le système on aura deux composants de nature différente :

– des "terminaux" qui sont des équipements relié à un seul routeur, qui sont potentiellement des émetteur et récepteur de message, mais qui ne relaient pas les messages reçus
– des "routeurs"qui sont connectés à d'autres routeurs et éventuellement à un terminal, qui ont pour objectif de relayer les messages reçus via un routage par inondation. Un "routeur" ne peut être ni la source ni la destination finale d'un message.

2 Programme de départ, fichier d'aide et contraintes de fonctionnement

Un fichier d'aide (projetAide.occ) est fourni par mail et contient le code de la procédure "terminal", les types de données de départ, un guide pour l'écriture du programme principal, ainsi que la procédure d'affichage des évènements. De plus il faut en partie réutiliser la topologie créée pour le TP sur le routage RIP.

2.1 Contraintes de fonctionnement

Pour simplifier, il n'y a pas de files d'attente dans les routeur. On supposera donc qu'il ne peut y avoir qu'un seul message en cours d'acheminement par inondation à un instant donné. Si un routeur reçoit plusieurs message différents de *null* lors d'un tour, ces messages sont obligatoirement identiques, et vous devez faire en sorte que ce soit le cas dans vos simulations, via les commandes de création de message.

La mise en œuvre respectera le modèle synchrone, ce qui veut dire qu'à chaque tour un nœud (terminal ou routeur) doit envoyer un message à tous ses voisins et recevoir un message de tous ses voisins (qui est peut être un paquet "null" ou un paquet de données).

2.2 Topologie et programme principal

Le réseau de processus est bidirectionnel : tout composant (routeur ou terminal) est connecté a un voisin par 2 canaux.

Votre réseau doit être composé de :

– 7 routeurs numérotés de 0 à 6
– 3 terminaux numérotés de 0 à 2

Pour la topologie de connexion des routeurs entre eux, vous pouvez reprendre celle créée pour le TP sur le routage RIP, en adaptant les paramètre de lancement des processus routeur.

En ce qui concerne les 3 terminaux, chacun d'eux est connecté directement à un seul routeur, selon une disposition que vous devez choisir (le terminal 0 n'est pas obligatoirement connecté au routeur 0). En repartant de la topologie du TP RIP, cela nécessite donc de rajouter un canal d'entrée et de sortie aux routeurs qui sont connectés

à un terminal. Par conséquent le nombre maximal de canaux d'entrées/sorties est égal à 4 (3 routeurs et 1 terminal) et la constante maxVois est fixée à 4 dans le fichier d'aide.

Le fichier d'aide fourni un exemple de lancement et de connexion des routeurs et des terminaux.

2.3 Modélisation des paquets de données

Un message de type PAQUET est constitué des champs suivants :

- l'identifiant du terminal source (*source*) qui sert aussi à définir les paquets "null" (valeur à -1)
- l'identifiant du terminal destination (*dest*)
- le numéro du paquet (*num*), géré par l'émetteur

Comme l'important est de pouvoir tester l'acheminement des paquets, il n'est pas nécessaire que les paquets de données contiennent des données.

2.4 Commande de création de paquets de données

Pour tester le système, il est nécessaire de prévoir l'émission de paquets. Pour cela, on doit utiliser un tableau, donné en paramètre à chaque terminal, qui indique à quel moment (tour de boucle synchrone) le paquet créé doit être émis, et quel est le destinataire.

Une commande de création de paquet est définie par le type CREATION, constitué des champs suivants :

- l'identifiant du nœud destination (*dest*)
- entier indiquant le moment *time* (numéro de tour) où le paquet doit être émis

Comme il est prévu qu'il y ait plusieurs émissions en provenance d'un même terminal pendant la durée de la simulation, chaque terminal dispose d'un tableau de commande de création en paramètre (de taille 2).

2.5 Comportement de la procédure Terminal

Le programme de la procédure "terminal" est fourni et ne doit pas être modifié. Dans de programme, à chaque tour de boucle synchrone, le terminal effectue les actions suivantes :

- Émission/Réception
- Affichage du paquet reçu si non null (évènement *Recept*)
- Affichage du paquet émis si non null (évènement *Envoi*)
- Si le paquet reçu est à destination du terminal : affichage du paquet (évènement *Arrive*)
- Si le terminal doit envoyer un paquet au tour suivant : création du paquet et affichage du paquet (évènement *Creation*)

Vous pouvez en grande partie vous inspirer de ce programme pour l'écriture du programme routeur (notamment pour la gestion de l'affichage des évènements)

2.6 Gestion de l'affichage

Le système est doté d'un multiplexeur d'affichage, vers lequel tous les composants enverront des évènements à afficher (création, envoi, réception, arrivée) ainsi que l'identification du composant (nature et identifiant), les caractéristiques du paquet concerné, le numéro du cycle correspondant, et le numéro de port concerné.

Le type AFFPAQUET contient les champs suivants :

- *comp* : un entier qui indique la nature du composant ayant envoyé le message, en utilisant les valeurs prédéfinies (*TERMINAL, ROUTEUR*)
- *id* : l'identifiant du composant
- *evt* : un entier qui indique la nature de l'évènement associé au paquet, en utilisant les valeurs prédéfinies (*Creation, Envoi, Recept, Arrive*)
- *time* : le cycle courant du routeur
- *port* : le numéro de port associé à l'évènement
- *paquet* : le paquet à afficher

3 Objectif 1 : Inondation “de base” (5 points)

3.1 Mise en œuvre (4,5 points)

Écrivez le programme de la procédure *routeur* pour l’inondation “de base”. Le comportement doit être : si lors d’un tour le routeur reçoit un message différent de *null*, il doit le renvoyer à tous ses voisins lors du tour suivant. Le nombre de tour est infini. Vous devez également mettre en place l’affichage des évènements (de type *Envoi* et *Recept*)

3.2 Simulation et question (0,5 point)

En supposant que l’émetteur initial d’un message ne l’envoie qu’une seule fois et que le destinataire final d’un message ne le renvoie pas, cette version de “base” vous semble t’elle viable ? Justifiez votre réponse par l’observation d’une simulation.

4 Objectif 2 : Mémorisation des messages déjà reçus (4 points)

Pour améliorer la version précédente, un routeur ne va pas rediffuser un message si il a déjà été reçu (lors d’un tour précédent).

4.1 Mise en œuvre (3 points)

Pour effectuer cela, il faut mémoriser le fait d’avoir déjà reçu un message. Pour simplifier, on se base sur le fait qu’il ne peut y avoir que 2 créations de paquets par un terminal pendant la durée de la simulation, et que les numéros de ces paquets sont donc 0 et 1. Il est donc nécessaire de disposer d’un tableau bidimensionnel de booléen, qui, en fonction de l’identifiant du terminal émetteur et du numéro de paquet, permet de savoir si le paquet a déjà été reçu lors d’un tour précédent.

Mettez en œuvre cette modification, en vérifiant, après la mise à jour des paquets à émettre, si le paquet avait déjà été reçu : si oui, il faudra envoyer le message *null* à la place. Si non, il faut mémoriser l’information.

4.2 Simulation et questions (1 point)

Observez par une simulation que le fonctionnement est bien amélioré. Mettez en œuvre 2 simulations différentes avec pour chacune 1 seule création de paquet, pour un terminal émetteur différent.

1. Au bout de combien de cycles le message n’est plus en circulation dans le réseau pour vos 2 exemples ?
2. Ce nombre est-il identique pour les 2 exemples ?
3. Y a-t-il une relation entre ce nombre et la topologie du réseau ?

5 Objectif 3 : Comptage du nombre d’exemplaires de message envoyés (3 points)

Afin de mesurer les performances de l’algorithme en cours et des suivant, on souhaite, pour un message, mesurer le nombre de fois qu’il a été envoyé. Cette fonction va être mise en œuvre dans la procédure d’affichage.

5.1 Mise en œuvre (2,5 points)

On se base sur le même principe que pour la partie précédente. A chaque message possible (2 par terminal) la procédure d’affichage associe un compteur. A chaque évènement d’envoi reçu, la procédure incrémente le compteur correspondant, et affiche sa valeur dans l’affichage de l’évènement “Envoi”.

5.2 Simulation et question (0,5 points)

Reprenez vos 2 simulations précédentes, et mesurez le nombre de paquets émis pour un message dans les 2 cas de figure. Ce nombre est-il identique pour les 2 exemples ?

6 Objectif 4 : Non relayage du message aux voisins “sources” (3 points)

Pour réduire le nombre de transmissions de paquet “inutiles”, on va faire en sorte qu’un routeur n’envoie pas un message vers les voisins qui le lui ont envoyé en premier.

6.1 Mise en œuvre (2 points)

Concrètement, en se basant sur le programme précédent, le comportement à mettre en œuvre est : lors d’un tour, si le routeur reçoit un (ou plusieurs exemplaires) message pour la première fois, il doit renvoyer ce message, au tour suivant, à tous ses voisins sauf celui (ou ceux) qui lui ont envoyé ce message lors du tour courant.

6.2 Simulation et questions (1 point)

1. Compte tenu de ce nouveau programme, si un relais reçoit un message au tour T, est-il possible qu’il reçoive ce même message au tour T+1 ? Est-il possible qu’il reçoive ce même message au tour T+2 ? Justifiez vos réponses, en vous aidant d’un petit schéma si nécessaire.
2. Pour les 2 simulations précédentes : mesurez le nombre de paquets émis pour un message avec ce nouvel algorithme, ce nombre est-il identique pour les 2 exemples ? Ce nombre est-il réduit par rapport la simulation du programme de la partie précédente ?

7 Objectif 5 : Non relayage du message aux voisins “frères” (3 points)

Un routeur voisin “frère” est un voisin qui, pour un terminal émetteur identifié, reçoit le message pour la première fois au même moment (c’est à dire dans le même tour). Il est donc inutile de lui envoyer le message. Notez bien que le fait d’être frère (ou non) dépend du terminal émetteur du message.

7.1 Question préalable (0,5 points)

Si un routeur reçoit pour la première fois un message provenant d’un terminal au tour T, à quel tour recevra-t-il le même message provenant de son (ou ses) routeur(s) frère(s) ?

7.2 Mise en œuvre (2 points)

Pour savoir si un voisin est “frère”, il le faut le détecter lors du premier envoi par un terminal émetteur, en détectant la caractéristique mise en évidence dans la réponse à la question précédente. Par la suite, cette information doit être utilisée pour ne pas relayer un message vers un voisin “frère”, pour le même terminal émetteur .

7.3 Simulation et questions (0,5 points)

1. Effectuez 2 simulations avec chacune 2 créations de paquet provenant du même terminal émetteur (séparées par un nombre de tours suffisamment élevé pour éviter la présence des 2 message au même moment dans le réseau). Mesurez le nombres de paquets émis pour la 1ère et la 2ème création et vérifiez qu’il est inférieur lorsque les routeurs connaissent leurs frères.

2. Compte tenu de ce nouveau programme, pour un terminal émetteur donné et en supposant qu'un premier relayage de message ait permis aux routeurs de connaitre leurs voisins "frères" : lors des envois de message par le même terminal, y aura-t-il plus de paquets transmis que dans le cas de l'utilisation d'un arbre de recouvrement créé par un BFS ? Justifiez votre réponse, en vous aidant d'un petit schéma si nécessaire.

8 Travaux à remettre, modalités et barème

Modalités de remise Votre travail de mise en œuvre doit être achevé avant le mardi 18 janvier à 18h, et remis selon les modalités suivantes :

- Travail en binôme ou seul au choix, mais tout binôme devra être annoncé par mail avant le vendredi 9 décembre, après quoi les binômes ne seront plus acceptés.
- Mail de remise a envoyer à fabiani@univ-brest.fr avant le mardi 18 janvier à 18h (1 point en moins par jour de retard) contenant :
  - rapport **en format pdf**
  - listings complets des mises en œuvre pour les étapes terminées (compilées et exécutables), soit 5 programmes occam en tout au plus pour les objectifs 1 à 5
  - traces d'exécution complètes des expérimentations demandées pour les objectifs terminés, soit 11 traces : 1 trace (limitée) pour l'objectif 1 et 2 traces pour chacun des objectifs 2, 3, 4, 5
- Un acquittement au mail sera renvoyé sous 48h.

Sauvegarde des traces : n'oubliez pas que vous pouvez facilement sauvegarder les traces de vos exécutions dans un fichier texte par redirection. Par exemple : `./projet > trace1.txt`

Contenu du Rapport Le rapport doit contenir :

- Une section sur votre topologie choisie au TP5 : code de la procédure principale et schéma de la topologie.
- Une section pour chaque objectif achevé. Chaque section doit contenir :
  - Vos commentaires sur la partie : choix de mise en œuvre éventuels et si nécessaire : précisions par rapport à l'énoncé, problèmes rencontrés, cas d'erreur non résolus, améliorations possibles, ...
  - Le listing du code développé pour l'objectif, pour la procédure *routeur* (il n'est pas nécessaire de donner le programme principal), reproduit de façon lisible (respect des indentations).
  - Les exemples de trace d'exécution partiellement reproduites, avec un commentaire en expliquant la signification et montrant l'adéquation avec le comportement souhaité
  - Les réponses aux questions

Il est inutile de perdre du temps à recopier l'énoncé dans le rapport.

Barème de notation Le projet est noté sur 18 points (car 2 points viennent de la note des TD rendus). La notation prends en compte la qualité du programme, mais aussi dans une moindre mesure les commentaires (dans le code ou le rapport) et la clarté du code. Si la forme du rapport laisse à désirer (phrases incompréhensibles, fautes de français nombreuses, structure non respectée, ...) jusqu'à 2 points pourront être soustraits à la note obtenue. Tout oubli (code, trace, ...) dans le rapport ou les listings entrainera un 0 pour la partie manquante. Si de nombreuses similitudes sont détectées dans des travaux de groupes différents, les éléments concernés ne seront pas pris en compte dans la notation, sans préavis, pour tous les groupes concernés. Si le nombre d'étapes concernées est significatif, la procédure disciplinaire usuelle sera engagée.

Le barème pour les différents objectifs est indiqué sur les parties correspondantes.

Seules les objectifs pour lesquels le programme compile sans erreur seront notées.