Un Protocole de Routage : Dynamic Source Routing (DSR) Protocol

Projet de Développement Formel de Système
3A-SIL – 2025 - 2026

Dans le cadre de ce projet, nous proposons de modéliser un protocole de routage : Dynamic Source Routing (DSR) Protocol.

1 Description Informelle

L'objectif de ce projet est de produire, en utilisant le raffinement, un modèle décrivant la gestion d'un protocole de routage. Le modèle résultant de ce développement permettra de supporter et de traiter différentes fonctionnalités de ce protocole.

Le protocole DSR (Fig. 1) est un protocole de routage simple et efficace conçu spécifiquement pour être utilisé dans des réseaux *ad hoc* sans fil multi-sauts composés de nœuds mobiles. Il permet au réseau de s'auto-organiser et de s'auto-configurer complètement, sans nécessiter d'infrastructure ou d'administration de réseau existante. Dans les techniques de routage à la source, un expéditeur détermine la séquence complète des nœuds par lesquels il transmet le paquet de données. L'expéditeur liste explicitement cette route dans l'en-tête du paquet, en identifiant chaque « saut » de transmission par l'adresse du prochain nœud auquel le paquet de données est transmis sur son chemin vers le nœud de destination. L'expéditeur transmet ensuite le paquet sur son interface réseau sans fil au premier saut identifié dans la route source. Lorsqu'un hôte reçoit un paquet, si cet hôte n'est pas la destination du paquet, il transmet simplement le paquet au prochain saut identifié dans la route source dans l'en-tête du paquet. Une fois que le paquet atteint sa destination, il est livré à l'hôte.

Le protocole présenté ici est explicitement conçu pour être utilisé dans l'environnement sans fil d'un réseau *ad hoc*. Il n'y a pas d'annonces périodiques de routeurs dans le protocole. Au lieu de cela, lorsqu'un nœud a besoin d'une route vers un autre nœud, il en détermine une dynamiquement en se basant sur une table de routage locale ou sur une route mise en cache et sur les résultats d'un protocole de découverte de route [1].

DSR se compose de deux mécanismes : *route discovery* and *route maintenance*.

1.1 Route Discovery

Chaque fois qu'une source doit communiquer avec une destination et qu'elle n'a pas de route dans sa table de routage, elle diffuse un message de demande de route (*route request*, RREQ) pour trouver une route. Chaque voisin reçoit le RREQ et, s'il n'a pas déjà traité la même demande auparavant, ajoute sa propre adresse à la liste d'adresses du RREQ, puis rediffuse le paquet. Ce processus se poursuit jusqu'à ce que le compteur de sauts maximum soit dépassé (auquel cas la RREQ est *rejetée*), ou jusqu'à ce que la destination soit atteinte. Dans ce dernier cas, la destination reçoit la RREQ, ajoute son adresse et génère un paquet de réponse de route (*route response*, RREP) vers la source en utilisant l'inverse de la route accumulée.

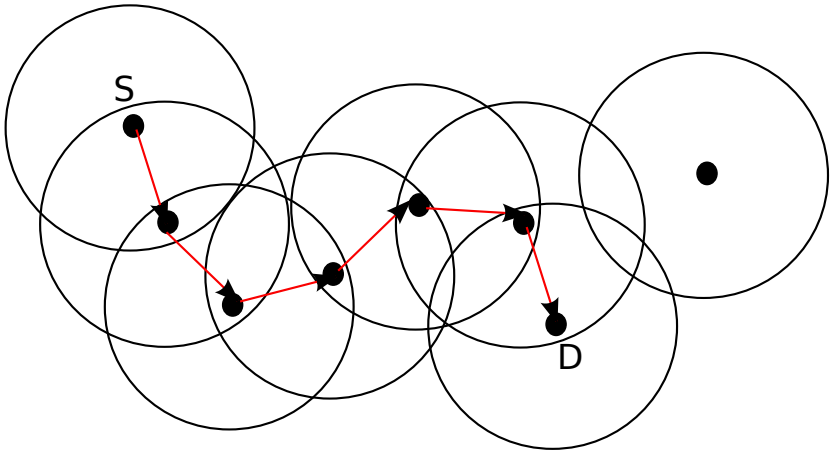

FIGURE 1 – DSR adhoc network

1.2 Route Maintenance

La maintenance des routes est utilisée pour gérer les routes précédemment découvertes (cache, expiration, commutation). Chaque nœud le long de la route, lorsqu'il transmet le paquet au prochain saut, est responsable de la détection du lien connecté suivant. Lorsque le mécanisme de retransmission et d'accusé de réception détecte que le lien est rompu, le nœud détecteur renvoie un paquet d'erreur de route (*route error packet*, RERRP) à la source du paquet. Le nœud source recherche alors dans son cache de route s'il existe une route alternative vers la destination de ce paquet. S'il en existe une, le nœud modifie la route source dans l'en-tête du paquet et l'envoie en utilisant cette nouvelle route. Lorsqu'un paquet d'erreur de route (RERRP) est reçu ou entendu, le lien en erreur est supprimé du cache de route local, et toutes les routes qui contiennent ce saut doivent être tronquées à ce point [1]. La source peut alors tenter d'utiliser toute autre route vers la destination qui se trouve déjà dans son cache de route, ou peut invoquer à nouveau la découverte de route pour trouver une nouvelle route.

2 Exigences et Hypothèses

Le protocole doit fonctionner dans un environnement où l'état des liaisons peut changer à tout moment. Si l'environnement change suffisamment rapidement, les liens signalés comme étant hors service peuvent en fait être en service et vice versa. Par conséquent, la table de routage locale peut avoir peu de rapport avec la topologie réelle du réseau. Pour résoudre ce problème, nous nous concentrons sur le cas limite, et le plus important, du comportement de l'algorithme : lorsque l'environnement est suffisamment « calme ». Dans ce cas, nous nous attendons à ce que la table de routage locale se "stabilise" finalement aux états de la topologie globale réelle. Selon la théorie de base des graphes, tout graphe peut être décomposé en une collection de composantes fortement connexes.

Nous distinguons les exigences de sûreté (préfixées par **SAF**) et les exigences fonctionnelles (préfixées par **FUN**).

2.1 Exigences de sûreté

Les exigences de sûreté suivantes sont définies.

— **SAF-R1 :** Les paquets de données doivent être transmis avec succès du nœud source au nœud de destination dans un réseau ad hoc dynamique.

2.2 Exigences fonctionnelles

Les exigences fonctionnelles suivantes sont définies.

— **FUN-R1.** Si l'environnement est inactif pendant un temps suffisamment long, la communication se stabilise et chaque nœud a une vue correcte des liens entre tous les nœuds de son sous-réseau connecté.

— **FUN-R2.** Le protocole de découverte de route doit découvrir une nouvelle route à partir du réseau connecté où l'état des liens peut changer à tout moment.

2.3 Hypothèses sur l'environnement dans lequel évoluent les rovers.

— **HYP-H1 :** Il existe un nombre fini de nœuds ou d'hôtes.

— **HYP-H2.** Il existe des liens dirigés, à sens unique, entre certaines paires de nœuds distincts. Les liens peuvent apparaître et disparaître à tout moment.

— **HYP-H3.** Les nœuds communiquent par diffusion, le nœud (x) envoyant un message à un autre nœud (y) lorsqu'ils sont directement connectés.

— **HYP-H4.** Lorsqu'une liaison tombe en panne, tout message envoyé sur cette liaison et non encore reçu est perdu. Ceci reflète le fait que la communication est asynchrone. Il y a un délai entre la transmission et la réception du message, et les messages peuvent être perdus pendant cet intervalle de temps.

— **HYP-H5.** Les hôtes ne se déplacent pas continuellement à une vitesse telle qu'il faille inonder (= lorsqu'un un paquet arrive à un nœud, il est envoyé à toutes les liaisons sortantes sauf celle sur laquelle il est arrivé) chaque paquet.

3 Objectifs du Projet

Nous nous proposons de concevoir un modèle Event-B associé à la gestion du protocole de routage DSR. Ce modèle devra être construit graduellement avec la prise en compte pas-à-pas des différentes exigences.

Le raffinement sera mis en œuvre pour définir la séquence de modèles, allant du modèle abstrait jusqu'au modèle concret. Le dernier modèle de cette séquence devra prendre en compte la totalité des exigences. Il s'agira alors du modèle terminal.

4 Une modélisation possible

Une stratégie de raffinement possible est décrite ci-dessous. Les différentes exigences pouvant être prises en compte par chaque modèle.

Modèle initial. Spécifier le protocole de communication de base pour l'envoi, la réception, la perte de paquets de données et les changements de topologie du réseau.

Premier raffinement. Introduction de l'architecture *store and forward* pour les paquets de données passant du nœud source au nœud de destination.

Deuxième raffinement. Introduction de la table de routage locale.

Troisième raffinement. Introduction du protocole de découverte de route pour découvrir une nouvelle route.

Quatrième raffinement. Fournit des informations plus détaillées sur le protocole de découverte de route.

Cinquième raffinement. Introduction de numéros de séquence pour le suivi des informations des paquets de demande de nouveaux itinéraires.

5 Remarques

— Attention, la stratégie de raffinement de la section précédente est proposée par les auteurs du sujet de projet. Elle n'est pas unique, et n'est peut-être pas adaptée à **votre** modélisation. Il n'est pas indispensable de la suivre. Vous pouvez définir votre propre stratégie, et surtout y réfléchir longuement pour ne pas à avoir à redévelopper vos modèles à chaque fois que vous aurez identifié un problème.
— Il est important de positionner les différentes exigences prises en compte à chaque niveau de raffinement.
— Les exigences précédentes peuvent être complétées si vous jugez qu'elles ne sont pas suffisamment précises.
— Des exigences de sûreté additionnelles peuvent également être introduites partout où cela est jugé nécessaire.

6 Travail demandé.

Il est demandé de réaliser un développement prenant en compte les exigences précédentes.

Si vous définissez votre propre stratégie de raffinement, il est demandé de l'expliciter comme cela a été fait dans la section 4.

Pour chacun des modèles produit, il est demandé de suivre les étapes suivantes.

1. Définir le modèle associé à chaque niveau de modélisation.
2. Dériver un système états-transitions correspondant à ce modèle.
3. Utiliser la plate-forme Rodin comme support de modélisation
4. Compléter le modèle par les propriétés de sûreté pertinentes sous forme d'invariant
5. Valider le modèle en utilisant l'animation à l'aide du model checker ProB
6. Vérifier l'absence de blocage sur ce modèle en utilisant ProB.
7. Utiliser la logique LTL ou CTL pour décrire des propriétés pertinentes
8. Modifier les préférences de ProB pour permettre l'utilisation de 5 éléments dans les ensembles définis puis animer le même modèle avec ces nouvelles préférences modifiées.

7 Exigences pour la réalisation du projet

1. Le projet est réalisé en **groupes de 2 étudiants**. Chaque membre devra préciser **dans le rapport final quelle aura été sa contribution** au projet.
2. Le langage de modélisation utilisé est le langage Event-B sur la plate-forme Rodin
3. L'ensemble des concepts du cours devront être mis en œuvre, en particulier :
   — Conception en utilisant la méthode des raffinages
   — Justification des choix des types de données manipulés
   — Conception de composants : **Context** et **Machine**.
   — Utilisation de ProB pour valider/animer les modèles
   — Utilisation des prouveurs de la plate-forme Rodin pour prouver les différentes obligations de preuve des modèles produits.

8 Livrables

Les **documents à rendre** sont :

- une archive contenant les modèles **Event-B**
- un **rapport** (rapport.pdf) contenant au moins :
  - un résumé qui décrit l'objectif et le contenu du rapport (10 lignes max),
  - une introduction qui présente le problème traité et le plan du document
  - l'architecture du développement
  - la justification des principaux choix réalisés
  - la présentation des principaux raffinement et types de données
  - la tableau de preuves
  - la démarche adoptée pour animer les modèles avec ProB
  - les difficultés rencontrées et les solutions adoptées en justifiant vos choix (en particulier quand vous avez envisagé plusieurs solutions)
  - un bilan technique donnant un état d'avancement du projet et les perspectives d'amélioration/évolution
  - un bilan *personnel* et *individuel* : intérêt, temps passé, temps passé à la conception, temps passé à la modélisation, la preuve et la validation, temps passé à la mise au point, temps passé sur le rapport, enseignements tirés de ce projet, etc.

Références

[1] David B. Johnson and David A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*, pages 153–181. Springer US, Boston, MA, 1996.