

# Rapport du projet

## Un protocole de routage : Dynamic Source Routing (DSR) Protocol

Par : **Guohao Dai, Youssef Choukrani, Hamza Ennaoui, Youssef Chiguer**

---

## **Contenu :**

1. Objectif et contenu
2. Introduction
3. Architecture de développement et raffinement
4. Tableau de preuves
5. Démarches d'animation des modèles avec ProB
6. Difficultés et solutions
7. Bilan technique
8. Bilan personnel
9. Conclusion

---

## Objectifs et contenu :

L'objectif du projet est de modéliser le protocole Dynamic Source Routing (DSR).

Notre groupe a essayé de donner une modélisation de ce protocole à l'aide du langage Event-B sous rodin. En s'appuyant sur son mode de fonctionnement et des exigences et des hypothèses additionnelles.

Ce rapport a pour but d'explicitier nos étapes de modélisation et son architecture. En outre, le rapport essaie de justifier nos choix de développement, et les solutions que nous avons choisi pour régler les problèmes rencontrés.

Au sens large, ce rapport donne une vue générale sur la modélisation et le déroulement du développement pour le groupe et pour chacun.

---

## Introduction

Le protocole en question est appelé Dynamic Source Routing protocol. Il sert à établir de la communication entre plusieurs nœuds mobiles. En effet, le protocole est capable de se mettre à jour automatiquement sans rétablir la table de routage d'une manière périodique. De telle façon que si on a besoin d'une route pour transmettre un message, un algorithme de découverte se lance et met à jour les routes dans la table de routage.

Ainsi, ce projet a pour objectif de modéliser à l'aide de Event-B ce protocole et abstraire son fonctionnement en suivant plusieurs étapes de développement.

En effet, la chaîne de modélisation doit traduire le mode de fonctionnement, l'envoi et la réception et installer une sorte de topologie réseau ainsi que le protocole de découverte.

Essentiellement, le rapport comprendra l'architecture de développement et de raffinement qui va expliquer les étapes de la modélisation et le raffinement, ainsi qu'expliquer certains choix fait dans le travail.

Ensuite, on montrera le tableau de preuve à partir des statistiques de preuves données par le logiciel Rodin.

Puis, on exprimera notre démarche de visualisation et animation par ProB. Par la suite, on va développer les difficultés rencontrées pendant le développement et les solutions prises.

Finalement, on donnera deux bilans, un sera technique sur l'avancement du projet et l'autre personnel sur le temps passé sur chaque étape et le travail fait par chaque membre du groupe.

---

## Architecture de développement et raffinement

Nous avons commencé le développement des modèles par une phase de conception pour d'abord bien comprendre les exigences et trouver une idée pour les mettre en œuvre. Dans un premier temps, nous avons défini un contexte qui introduit des ensembles et des constantes représentant l'environnement réseau. Puis nous avons implémenté le modèle abstrait et les étapes de raffinement:

*Modèle Initial - Spécification du protocole de communication de base pour envoyer, recevoir et perdre des messages.*

*Premier raffinement - Introduction de la logique 'forward' et 'store' pour permettre de transmettre des messages du nœud source au nœud de destination.*

*Deuxième raffinement - Introduction de la table de routage et du protocole de découverte sous la forme d'un événement 'discover'.*

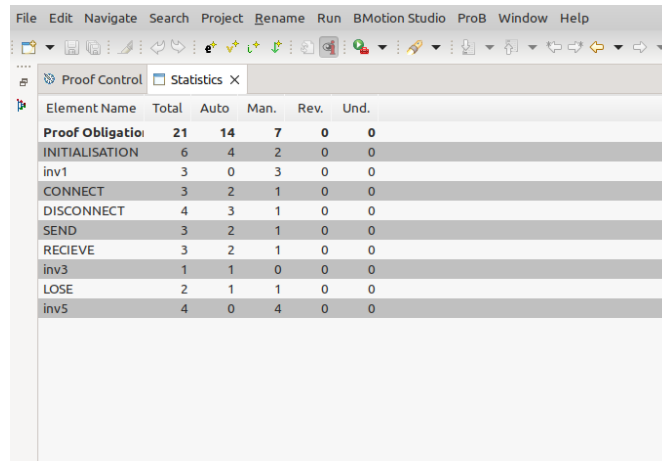
Nous avons choisi d'introduire la table de routage et le protocole de découverte dans le même raffinement pour éviter d'avoir à utiliser un 'gluing invariable' et parce que cela avait du sens dans le contexte de notre implémentation du protocole de découverte.

Nous avons intégré le protocole de découverte afin d'ajouter pour chaque node1 la route vers la destination du message depuis l'une de ses connexions. Cela signifie que si node2 est une connexion de node1 et que la table de routage entre node2 et la destination du message n'est pas vide, nous ajoutons node2 comme prochaine étape de la route comme si nous ajoutions la route existante entre node2 et la destination au route entre node1 et la destination. Cette solution simule le même comportement du protocole sans avoir à mettre en œuvre l'envoi de paquets pour découvrir de nouvelles routes.

Nous nous sommes assurés que les événements ne sont déclenchés que lorsque cela est nécessaire en définissant des gardes qui ne laissent que les événements possibles. Pour cela, nous avons défini un ensemble de variables qui aident à définir collectivement chaque état du système.

# Tableau de preuves

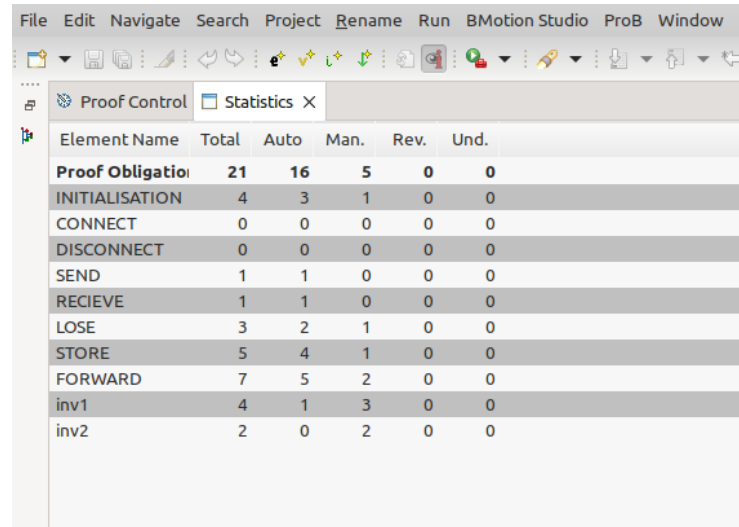
Modèle initial :



The screenshot shows the BMotion Studio interface with the 'Statistics' tab selected. The table displays the following data:

Element Name	Total	Auto	Man.	Rev.	Und.
<b>Proof Obligation</b>	<b>21</b>	<b>14</b>	<b>7</b>	<b>0</b>	<b>0</b>
INITIALISATION	6	4	2	0	0
inv1	3	0	3	0	0
CONNECT	3	2	1	0	0
DISCONNECT	4	3	1	0	0
SEND	3	2	1	0	0
RECIEVE	3	2	1	0	0
inv3	1	1	0	0	0
LOSE	2	1	1	0	0
inv5	4	0	4	0	0

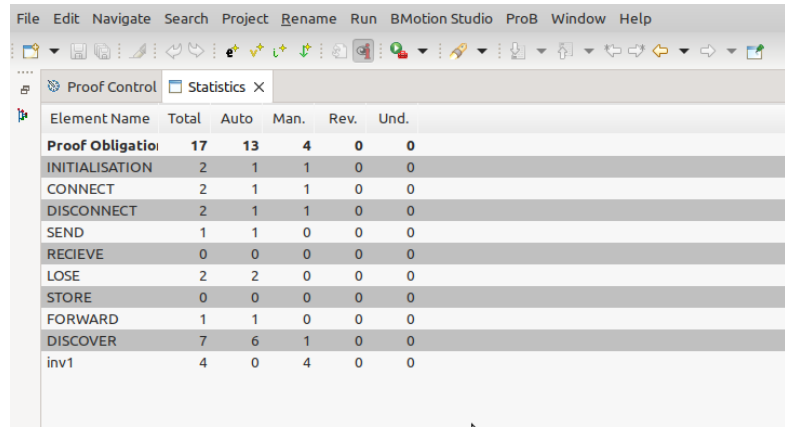
Modèle 1 :



The screenshot shows the BMotion Studio interface with the 'Statistics' tab selected. The table displays the following data:

Element Name	Total	Auto	Man.	Rev.	Und.
<b>Proof Obligation</b>	<b>21</b>	<b>16</b>	<b>5</b>	<b>0</b>	<b>0</b>
INITIALISATION	4	3	1	0	0
CONNECT	0	0	0	0	0
DISCONNECT	0	0	0	0	0
SEND	1	1	0	0	0
RECIEVE	1	1	0	0	0
LOSE	3	2	1	0	0
STORE	5	4	1	0	0
FORWARD	7	5	2	0	0
inv1	4	1	3	0	0
inv2	2	0	2	0	0

Modèle 2 :



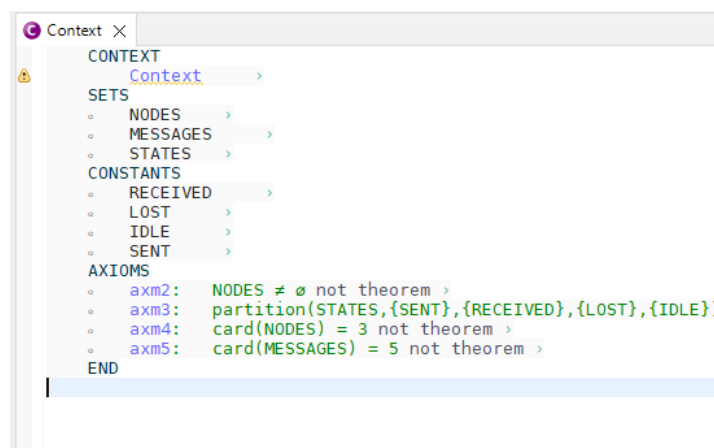
Element Name	Total	Auto	Man.	Rev.	Und.
<b>Proof Obligation</b>	<b>17</b>	<b>13</b>	<b>4</b>	<b>0</b>	<b>0</b>
INITIALISATION	2	1	1	0	0
CONNECT	2	1	1	0	0
DISCONNECT	2	1	1	0	0
SEND	1	1	0	0	0
RECIEVE	0	0	0	0	0
LOSE	2	2	0	0	0
STORE	0	0	0	0	0
FORWARD	1	1	0	0	0
DISCOVER	7	6	1	0	0
inv1	4	0	4	0	0

## Démarches d'animation des modèles avec ProB

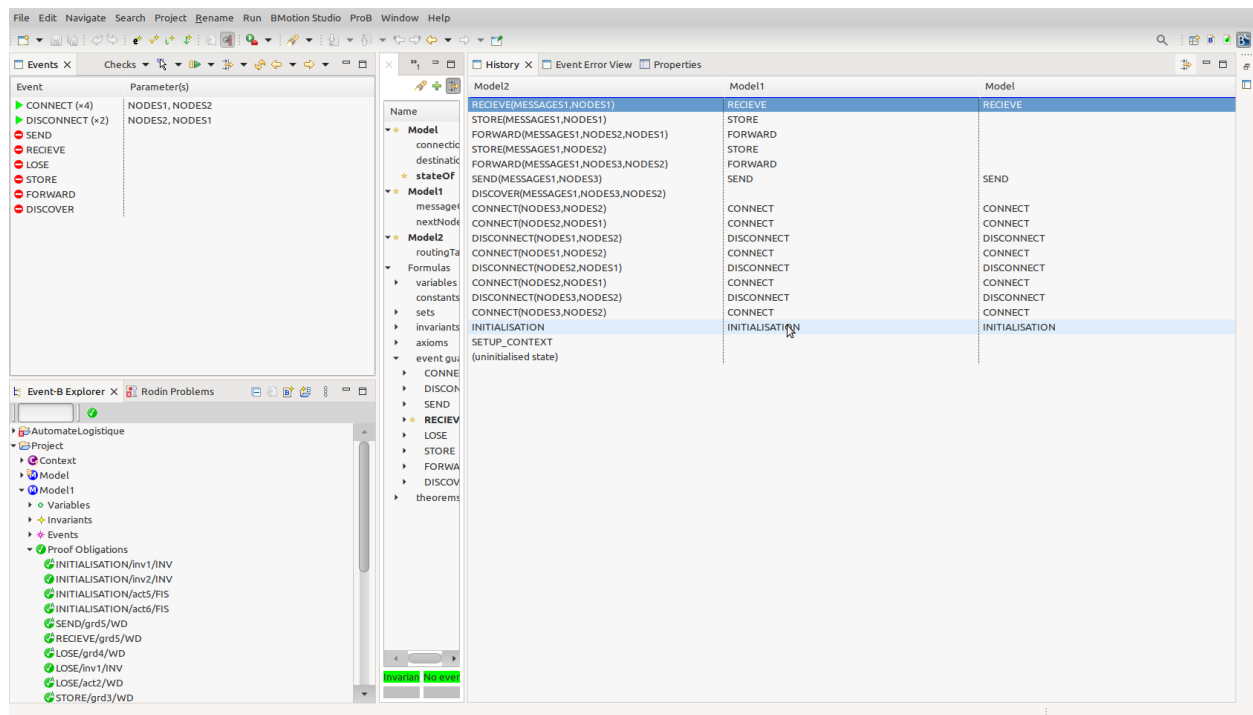
On a suivi plusieurs cas possibles d'exécution en ProB qui nous a amené vers plusieurs chemins possibles de l'arbre d'exécutions.

Les captures d'écrans de l'outil ProB dans la page suivante sont prises pour des exécutions du modèle final. On a notamment joué aussi sur le contexte pour spécifier le cardinal des ensembles NODES et MESSAGES.

Par exemple :



```
CONTEXT
Context
SETS
  NODES
  MESSAGES
  STATES
CONSTANTS
  RECEIVED
  LOST
  IDLE
  SENT
AXIOMS
  axm2: NODES ≠ ∅ not theorem
  axm3: partition(STATES,{SENT},{RECEIVED},{LOST},{IDLE})
  axm4: card(NODES) = 3 not theorem
  axm5: card(MESSAGES) = 5 not theorem
END
```





---

## Difficultés et solutions

1. Nous avons d'abord pensé à une mise en œuvre similaire au protocole RIP (*Routing Information Protocol*) dans le processus de mise à jour et de maintenance de la table de routage. L'objectif de découvrir les nœuds joignables dans le réseau est atteint en échangeant périodiquement les informations de la table de routage pour chaque nœud. Cela signifie que dans notre système, un événement doit être appelé de manière récursive pour mettre à jour périodiquement les informations de la table de routage entre les différents nœuds. Mais cela n'est pas facile à mettre en œuvre dans Event-B.

### **Solution:**

Nous nous engageons à ce que la connectivité de chaque nœud au nœud de destination d'un message soit un facteur permettant de déterminer si un nœud est ajouté ou non à la table de routage (*DISCOVER*).

Supposons qu'il existe trois nœuds (*node1*, *node2* et ***destinationOf(message)***).

Lorsque

- le *node1* et le *node2* sont connectés
- et ***routingTable(node2, destinationOf(message))*** n'est pas vide

Alors nous ajoutons ***routingTable(node1, destinationOf(message))={node2}*** ajouté à la route entre le *node1* et la ***destinationOf(message)***.

2. Une autre difficulté était comment modéliser l'ensemble des événements {SEND, FORWARD, STORE, RECEIVE}. En gros, on doit définir ce que modélise chaque événement.

### **Solution :**

On s'est mis d'accord que Send est le concept de l'envoi puis le forward envoie le message et le supprime du nœud envoyeur et le met dans la fonction ***nextNodeOf*** du nœud suivant dans la table de routage. Cette fonction a pour utilité de pouvoir trouver ce message dans l'événement STORE. Une boucle de FORWARD et

---

STORE et puis on finit par RECEIVE qui traduit la réception finale du message dans le nœud destination dans STORE.

3. Pendant la définition de quelques variables comme **messageOf** et **routingTable**, on avait rencontré des obligations de preuve EQL dans le prouveur. Puisqu'on voulait éviter l'utilisation d'un 'gluing invariable' nous avons recherché une autre solution.

Pour messageOf, on a essayé de l'introduire dans le modèle initial mais on s'est rendu compte que cette variable là doit être modifiée après l'introduction de STORE et FORWARD. Pareil pour routingTable qui est introduit dans le deuxième raffinement mais lors du troisième raffinement on pouvait plus la modifier pendant le découverte de route.

**Solution :**

Comme solution à ces problèmes là, on n'avait introduit messageOf qu'au premier raffinement et pour routingTable, on a annulé le troisième raffinement. Ainsi l'introduction de la table de routage et la découverte de routes sont toutes les deux dans le deuxième raffinement qui sera plus simple pour notre démarche de modélisation.

## Bilan technique

Conformément à notre solution, nous avons réalisé les 3 niveaux de raffinement qui nous permettent de répondre à l'ensemble des exigences fonctionnelles et de sûreté présentes dans le sujet. En corrélation avec cela, nous avons réussi à valider toutes les obligations de preuve pour tous les niveaux de raffinement présentés précédemment.

## Bilan personnel

Ce projet nous a permis de mettre à l'épreuve notre capacité de conception grâce au haut niveau d'abstraction nécessaire pour pouvoir imaginer une solution convenable qui répond à l'ensemble des exigences fonctionnelles et des exigences de sûreté du sujet.

Il était alors nécessaire de bien comprendre le problème pour trouver une bonne solution: dans ce sens, nous avons consacré plus de 5 heures à imaginer notre conception avant de

---

pouvoir la mettre à l'application. Cette étape de modélisation a été particulièrement "challenging" car souvent elle mettait en évidence des trous qui étaient présents dans notre raisonnement et auxquels on adaptait à chaque fois notre solution. Cela ne se traduit pas que par le temps passé dans cette étape (~20h), mais aussi par l'amélioration de nos capacités à manipuler la plateforme Rodin et affiner nos connaissances en Event-B. La preuve et la validation de notre modèle nous a particulièrement aidé à pousser au plus loin notre raisonnement puisqu'elles nous permettaient d'avoir une idée précise sur le degré d'exactitude de nos modifications au fur et à mesure que le projet avançait.

## Conclusion

Plusieurs protocoles de routage existent dans la nature qui permettent chacun, suivant sa conception, de résoudre un problème particulier. Dans le cadre de ce projet, la particularité du protocole DSR (*Dynamic Source Routing*) réside dans sa capacité à permettre au réseau de s'auto-organiser et de s'auto-configurer sans nécessiter d'infrastructure ou d'administration de réseau.

La complexité d'implantation de telle solution se propage aussi à sa modélisation qui était le but ultime de notre effort: en commençant par la conception et jusqu'à la modélisation et les obligations de preuve, en passant par nos raffinements, nous avons appris à être humble devant nos idées pour accepter de les améliorer grâce aux nombreux "brainstorming" effectués et aux conseils de notre professeur.

Tout cela nous a permis de clôturer en beauté notre apprentissage du *Développement Formel des Systèmes Complexes* et a ouvert notre appétit à apprendre encore de cet univers et faire le pont avec notre vie professionnelle.