IUT de Montpellier, Dép. Informatique
M???? - Réseau (2ème année) M. Molnar

TD nº 6 - Etude du routage RPL dans un domaine IdO

Les objectifs de ce TD sont :
— survoler certaines propriètés des IdOs ;
— étudier 6LowPAN ;
— étudier le protocole RPL ;
— introduire le système Contiki ;
— faire des simulations avec Cooja.

Conseil : Lisez attentivement les parties introductives. Pour enrichir vos connaissances,vous pouvez aussi suivre les liens ...

1 Internet des Objets

L'Internet des Objets (IdO) est un domaine qui se compose d'objets communiquant entre eux via l'Internet.

Il existe de nombreux types d'objets connectés : des voitures, des sondes, des caméras, des portes etc. Les besoins des communications peuvent impliquer des réseaux basés sur
— un protocole bas débit, lent, par exemple LoRa pour collecter des données (la couverture de la France est donnée)
— des solutions habituelles IP (version 6 ici).

Les objets, acteurs, capteurs sont des machines de petites tailles, en général autonomes, ce qui implique certaines contraintes :
— interface radio (sans fil)
— capacité énergétique limitée
— peu de puissance de calcul
— environnement extérieur qui peut être soumis au changement (par exemple, à l'issue des mobilités)

Pour lire d'avantage sur les IdOs :
`https://www.openscience.fr/IMG/pdf/iste_ido18v2n1_1.pdf`
`https://www.cisco.com/c/dam/global/en_ca/solutions/executive/assets/pdf/internet-of-thi`
`pdf`

2 6LowPAN

IPv6 propose un nombre d'adresses beaucoup plus important, codées sur 128 bits (2^{128} adresses), permettant ainsi de pallier à la limitation du nombre d'adresses permises par *IPv4*. C'est également un protocole plus sécurisé et adéquat de part sa simplification d'écriture (voir RFC 5952) permettant aussi une plus grande flexibilité. Les en-têtes des paquets *IPv6* ont une taille fixe de 40 octets.

L'IdO profite de cette capacité d'adressage qui permet à chaque objet d'avoir sa propre adresse sur le réseau. Cependant ce protocole n'étant pas conçu pour les objets à faibles ressources de part la taille des entêtes utilisées dans les communications, il faut en utiliser un plus optimisé. Il s'agit de IPv6 Low power Wireless Personal Area Networks (6LoWPAN).

2.1 IPv6 Low power Wireless Personal Area Networks

Les capacités des appareils étant faibles, la transmission et l'encapsulation des données sont des éléments déterminants. Ainsi, 6LoWPAN qui est un mécanisme d'encapsulation des paquets lors de la

transmission de données dans des réseaux de faibles capacités sera utilisé dans les systèmes à ressources limitées (voir *RFC 4944*).

Un réseau 6LoWPAN sans fil propose une adaptation du protocole IPv6. Au niveau II, les équipements suivent le standard IEEE 802.15.4. Ils ont une portée limitée, un faible débit, peu de mémoire et un faible coÃżt. Une couche d'adaptation entre les couches MAC et réseau réduit la surcharge de l'en-tête IPv6 et réalise une compression. Elle propose des en-têtes supplémentaires permettant de fragmenter un paquet IPv6 si ce dernier ne peut pas être contenu dans une trame MAC.

Trois types de noeuds existent dans 6LoWPAN : le routeur de bordure (BR), le routeur et l'hÃt'te. La topoligue de base est un graphe acyclique orienté (Directed Acyclic Graph - DAG). Un Graphe Orienté Acyclique (DAG) est un graphe orienté ne possédant aucun cycle (orienté). Dans un réseau 6LowPAN, plusieurs graphes acycliques orientés destination (Destination Oriented Directed Acyclic Graph - DODAG) peuvent exister et forment le DAG. La caractéristique supplémentaire apportée par ce type de graphe est que chaque arc du graphe est dirigé vers une seule et même destination qui est un BR. Le routeur de bordure est la racine (la destination) d'un DODAG. Il connecte les routes à l'Internet.

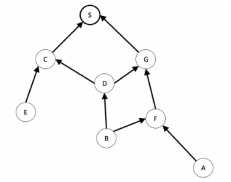

FIGURE 1 – DODAG - Graphe acyclique orienté vers une destination

Les routeurs sont des équipements intermédiaires sur les routes, et les hÃt'tes sont les équipements feuilles du réseau. Le réseau est alors un réseau sans fil, multi-saut.

2.2 Le protocole de routage standardisé RPL

Le trafic typique dans le réseau est le ramassage des données incast (multipoint-à-point, des noeuds vers le BR), mais il permet la diffusion (du BR aux noeuds), ainsi que le point-à-point entre les noeuds.

Ce protocole de routage construit un DODAG pour chaque application (et naturellement vers un BR). Les routeurs et le BR sont autorisés à router des paquets. Les paquets fragmentés sont reconstitués à chaque saut, ce qui rend le protocole plus robuste face aux pertes de fragments.

Nous illustrons brièvement la construction des routes, le routage et la maintenance / correction des routes.

2.2.1 Construction d'un DODAG et routage

L'arbre est construit couche par couche (rang par rang). La racine RPL (un BR) a un rang égal à 1, et elle envoye un message (DIO, voir sur la figure) pour mobiliser ses voisins. Les noeuds dans sa zone de couverture (1 et 4) reçoivent et traitent ce DIO. Après avoir reçu les DIO, chaque noeud essaye de se connecte au DODAG en choisissant un parent préféré en utilisant une fonction d'objective (par exemple en minimisant le nombre de sauts jusqu'à R). Les noeuds 1 et 4 vont avoir un rang 2. Leur parent est trivialement R. Les noeuds 1 et 4 émettent leur propre DIO vers leur voisinage (les noeuds 2, 3 pour le noeud 1 et les noeuds 5, 6 pour le noeud 4) qui procèdent de la même manière. Quand la construction est terminée, les noeuds peuvent communiquer avec la racine R.

Dans l'autres sens, les routes descendantes peuvent être construites grâce au message Destination Advertisement Object (DAO). Les messages DAO contiennent la destination. Pour simplifier, regardons rapidement les routes descendantes en mode "storing'. Les messages DAO sont envoyés au parent préféré qui installe à la réception du DAO la route descendante correspondante. Le DAO se propage jusqu'à la racine RPL à travers les parents préférés afin d'établir un chemin descendant complet de la racine jusqu'à la destination.

Plus largement, RPL permet de sélectionner un ensemble de parents (dont le parent préféré) tel que le rang des parents doit être inférieur au rang du noeud. Le noeud installera, en plus de la route par

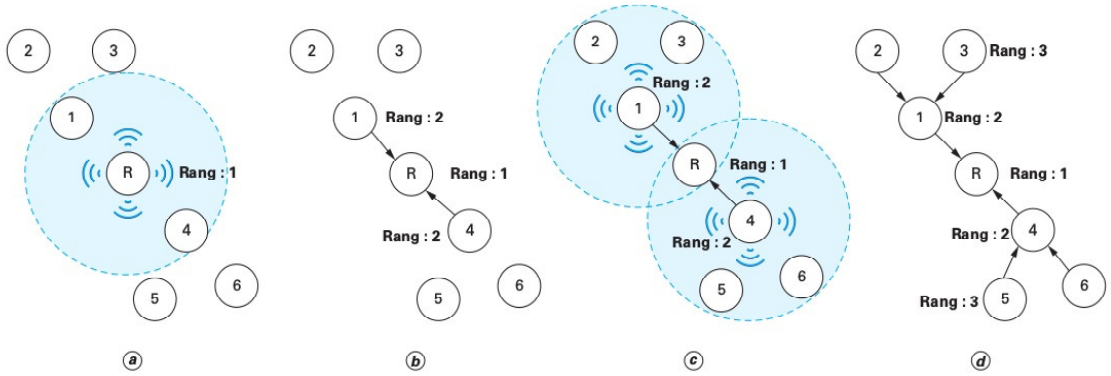

FIGURE 2 – Construction d'un DODAG

défaut vers son parent préféré, des routes vers les autres parents. Notons que pour réparer les éventuelles pannes du réseau, RPL dispose de deux mécanismes : la réparation globale (pour coconstruire le DODAG) et la réparation locale (pour refermer localement un lien manquant).

Une fois le DOOAG est construit, il est possible d'utiliser les routes sur l'arbre (dans les deux sens). Pour trouver une route entre deux noeuds, on peut toujours remonter jusqu'à la racine BR puis redescendre à la destination.

3 Contiki

Pour faire fonctionner les objets limités en ressources et la communication entre les objets, il faut un système relativement petit qui utilise peu de ressources. Pour des applications qui nécessitent une latence limitée et une robustesse, 6LowPAN et le routage RPL sont proposés. Les domaines dans l'Internet des Objets peuvent utiliser plusieurs plateformes / systèmes : TinyOS , Contiki, RIOT etc.

Contiki est un système d'exploitation "open source" conçu spécialement pour les petits objets communicants. C'est un système d'exploitation écrit en C, utilisant peu de ressource et d'énergie. Il consomme 2 kilooctets de mémoire vive et 40 kilooctets de stockage. Il permet de connecter des objets peu compliqués et peu chers, qui possèdent peu d'énergie. C'est également une boite à outils pour construire des systèmes sans fil complexes. De plus, *Contiki* propose un environnement de simulation (*Cooja*) permettant de tester des systèmes avant de passer à la réalisation.

Un lien utile :

`http://anrg.usc.edu/contiki/index.php/Contiki_tutorials`

3.1 Instant Contiki

Pour travailler efficacement avec*Contiki* et *Cooja*, il existe un environnement de développement conçu par les créateurs de *Contiki* appelé *Instant Contiki* fourni sous la forme d'un disque dur virtuel contenant tous les outils dont nous avons besoin pour utiliser *Contiki* et *Cooja*. *Instant Contiki* est fondé sur la version 14 de *Ubuntu*, l'avantage de cette version est qu'elle s'exécute de façon fluide dans une machine virtuelle. Dans notre système, la machine virtuelle utilisée pour l'ouvrir est *VMware*.

4 Cooja

Cooja est un des logiciels disponibles dans l'environnement de développement *Instant Contiki*. Il permet la simulation de différents réseaux avec la création de divers objets comme des capteurs ou des Border Routers. Nous l'utiliserons ici avec le protocole IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL). Dans la simulation, on peut utiliser les éléments intégrés dans *Cooja* (appelés des **motes**) qui font appel à Contiki.

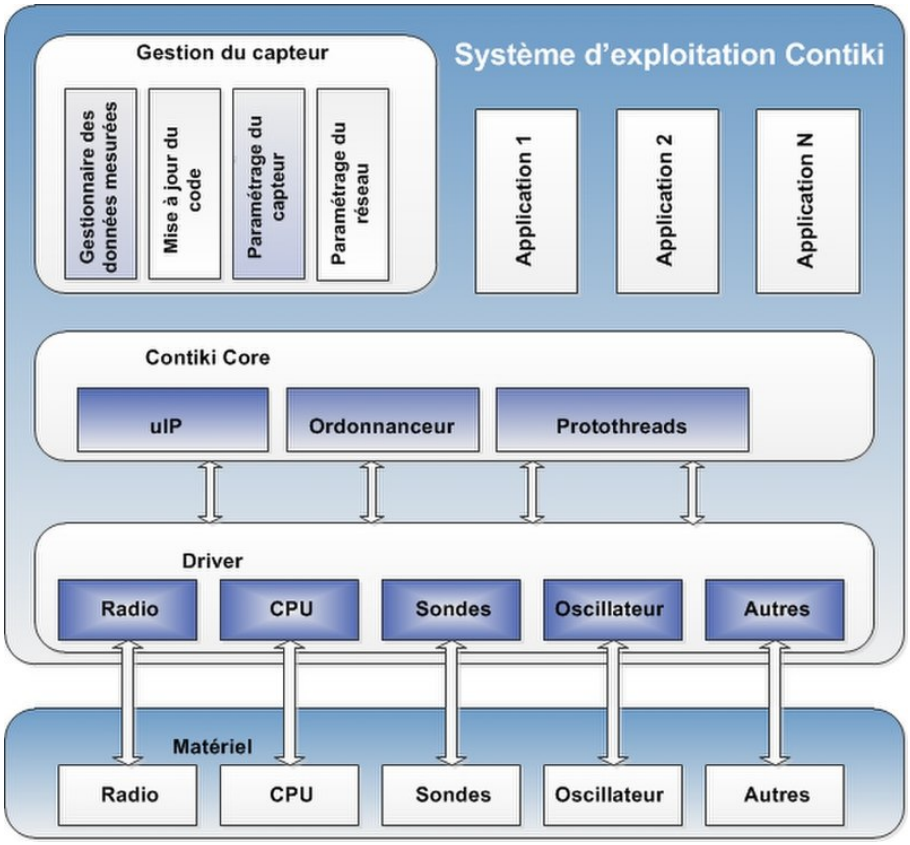

FIGURE 3 – Schéma du système d'exploitation *Contiki*

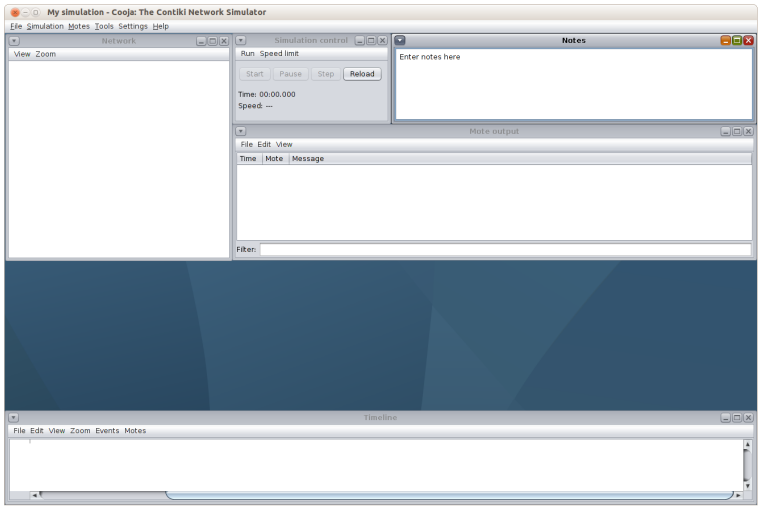

FIGURE 4 – Application *Cooja* au démarrage

Dans la simulation d'un réseau IdO, nous pouvons utiliser les éléments intégrés dans *Cooja* (appelés des **motes**) qui font appel à Contiki. Pour développer des nouvelles solutions, par exemple pour compléter / modifier le routage, on peut utiliser les fichiers supplémentaires (par exemple de RPL) écrits en C. Afin de modifier le comportement ou des parties du système, il suffit de modifier les fichiers sources.

Un lien :

`http://anrg.usc.edu/contiki/index.php/Cooja_Simulator`

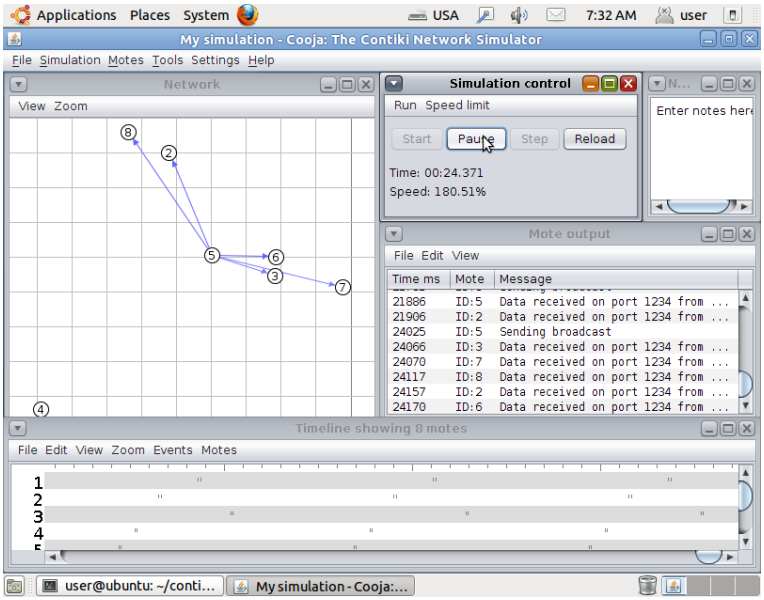

FIGURE 5 – Exemple de simulation de réseau de capteurs sans fils avec Cooja

5 Exercices

Exercice 1.
Lancer *Instant Contiki* depuis le dossier *InstantContiki3.0* en lançant
`Instant_Contiki_Ubuntu_12.04_32-bit.vmx`

IMPORTANT. Pour se connecter à Instant Contiki :
username : **user** password : **user**
(ou encore Guest mais avec mot de passe user)

Exercice 2.
Avant tout, vérifier la langue (le clavier) utilisée. Si vous avez un clavier AZERTY mais la machine virtuelle affiche la langue anglaise (ou svédoise) comme langue utilisée (à droite en haut dans un petit recangle), il faut la changer en deux étapes. 1) Cliquez sur le rectangle puis choisissez "input parameter setting", puis ajouter la langue franccaise si elle est absente. 2) Toujurs à partir de ce rectangle, changez la langue pour avoir Fr.

Exercice 3.
On peut rapidement tester, si *Contiki* fonctionne. Pour cela, il y a une petite application disponible `hello-word`.

Une fois que le bureau du système apparait (c'est comme dans ubuntu), ouvrir un terminal, puis :

```
$ cd contiki
$ cd examples/hello-world
$ make TARGET=native        # attendre la compilation
$ ./hello-world.native
```

Exercice 4.
Démarrer l'environnement d'émulation *Cooja* pour créer des simulations. Une fois que le bureau du système apparait (c'est comme dans ubuntu), ouvrir un terminal.

Dans le terminal, aller dans le répertoire cooja :
`$ cd contiki/tools/cooja`
Lancer *Cooja* par la commande :
`$ ant run`

Exercice 5.
Une fois que *Cooja* est actif, on peut créer et manipuler des simulations. Ici, on va créer une première simulation.

A) créer une *nouvelle simulation* à partir du menu FILE. Utiliser l'espace "Unit disk". Le simulateur ouvre différentes fenêtres dont :

"Network" : fenêtre représentant les différents motes et leur positionnement

"Simulation Control" : fenêtre permettant de contrÃt'ler le déroulement de la simulation

"Mote output" : fenêtre permettant de récupérer les logs des motes.

L'objectif de notre premier programme est l'activation de l'exemple hello-word.

Pour compiler ce premier exemple et faire fonctionner dans le simulateur, il faut créer un mote dans l'environnement "Network" que nous utilisons.

B) Pour la première simulation, il faut crér des types de motes. Pour cela, il faut choisir "Create new mote type" depios le menu "Mote" et selectionner un type accessible. Ajouter un mote à l'environnement de *Cooja* en suivant le menu *Motes/Add motes*. Choisissez le type *Sky mote* (un appareil bien connu). Dans la fenêtre de dialogue, indiquer le chemin vers le fichier *hello-world.c* pour "Contiki process".

C) Compiler le programme (bouton *Compile*). Plusieurs fichiers C sont compilés pour le logiciel qui sera utilisé sur le mote.

D) Créer un mote dans la simulation (bouton "Create"). Ici, on ajoute un seul mote qui sera affiché dans la fenêtre "Network".

E)Une fois le mote est ajouté à la fenêtre "'Network"', on peut commencer la simulation (fenêtre "Simulation Control"bouton "Start"). Le résultat du fonctionnement du mote (sa sorite) apparait dans la fenêtre "Mote Output".

Exercice 6.

La deuxième simulation illustre la collecte de données avec RPL.

Dans cette exercice, le protocole RPL est utilisé entre des "capteurs" et un routeur de bordure. La transmission utilise UDP. Le BR reçoit les paquets UDP des capteurs périodiquement. Pour cela, le DODAG RPL (qui est multi-hops) est utilisé.

Une présentation de l'exemple se trouve :

`http://anrg.usc.edu/contiki/index.php/RPL_UDP`

Dans cet exemple, RPL est implémenté et les paquets sont transmis un utilisant des sockets UDP entre un serveur (BR RPL) et des clients ("capteurs"). Les clients UDP envoient des paquets périodiquement au serveur, le serveur affiche les paquets reçus.

Les sources sont disponibles dans le répertoire `examples` de contiki (attention, X.X est la version de contiki utilisée) :

`~/contiki-X.X/examples/ipv6/rpl-udp/udp-server.c`

`~/contiki-X.X/examples/ipv6/rpl-udp/udp-client.c`

`~/contiki-X.X/core/net/tcpip.c`

`~/contiki-X.X/core/net/tcpip.h`

Le serveur UDP

Il a trois tâches :

1. Initialiser le DODAG
2. Créer un serveur socket UDP
3. Attendre les paquets des clients, et les afficher sur stdout.

Etudiez le code source pour ces trois fonctionnalités dans `udp-server.c`

Le client UDP

Il a deux tâches :

1. Créer une connexion socket UDP ; 2. Envoyer des paquets en utilisant le socket UDP.

Etudiez le code source pour ces fonctionnalités dans `udp-clien.c`

Pour réaliser la simulation, on propose la démarche suivante.

A) Pour utiliser la *simulation existante*, à partir du menu `FILE`, choisir `Open simulation->Browse` puis sélectionnez `~/contiki-X.X/examples/ipv6/rpl-udp/rpl-udp.csc`.

Si il y a une erreur de compilation, forcz une compilation supplémentaire :

```
$ cd contiki/examples/ipv6/rpl-udp
$ make
```

B) Pour aider l'observation des événements, la simulation peut être configurée. Quelques conseils suivent.

Dans la fenêtre "Simulation Control", on peut regler la vitesse de la simulation : " Speed limit", choisissez une petite valeur.

Dans la fenêtre "Network", dans " View", selectionner également "Radio traffic" qui permet de visualiser les éventuelles émissions / réceptions.

Les messages peuvent être observés dans la fenêtre "Motes Output".

Lancez la simulation : dans la fenêtre "Simulation Control" choisissez "Start". On peut faire des pauses ("Pause") et des redémarrages (" Start").

On peut également filtrer les sorties des objets avec l'ID d'un "objet" `node_id`.

On peut sauvegarder la sortie dans un fichier à partir de la fonction "File"

C) Observez le fonctionnement de réseau.

C.1. A l'aide des messages observés et les communications radio avec une vitesse limitée, dessinez le DODAG. Indiquez les rangs des objets.

C.2. Etudiez le comportement du RPL en cas e changements topologiques. Supprimez 2-3 objets (mais pas le BR) au hasard. Etudiez la reconstruction du DODAG et la nouvelle topologie.