Utilisation de Contiki-NG sur des OpenMote B

Estelle Jezequel et Aurélien Aubry,
Département Sciences du Numérique.

Juillet 2019

1 Contiki

Contiki est un OS pour l'IoT implantant la pile 6TiSCH. Ce projet est disponible sur github à l'adresse : `https://github.com/contiki-ng/contiki-ng`

Le dossier `contiki-ng` contient les répertoires suivants :

- `/os` : code de l'OS et de la pile protocolaire
- `/arch`
- `/examples` : exemples de projets dont 6TiSCH
- `/tools`
- `/tests`

Récupérer les sources de Contiki-ng avec les commandes :

```
git clone https://github.com/contiki-ng/contiki-ng
git submodule update --init
```

2 Openmote B

Nous allons utiliser les Openmote B, une plateforme qui utilise le SoC (System on Chip) TI CC2538, qui présente un ARM Cortex-M3 de fréquence 16/32MHz et 32koctets de RAM + 256/512 koctets de FLASH.

Vous trouverez des informations sur l'utilisation de cette plateforme ici :
`https://github.com/contiki-ng/contiki-ng/wiki/Platform-openmote-cc2538`.

3 Mise en place du réseau avec les OpenMote B

3.1 Installation du programme sur les cartes

Relier l'OpenMote à l'ordinateur et trouver le port sur lequel il est branché avec la commande :

```
ls /dev/ttyU*
```

Chaque carte semble utiliser deux ports (par exemple : */dev/ttyUI0* et */dev/ttyUI1*). Seul le nom du second port nous intéresse.

Dans le dossier du projet (par exemple `contiki-ng/examples/6tisch/simple-node`), flasher la carte en utilisant la commande :

```
make TARGET=openmote BOARD=openmote-cc2538 PORT=/dev/ttyUSB1 node.upload
```

Le nom du programme (ici "node") se trouve dans le `Makefile` associé à chaque projet.

3.2 Ouverture d'un shell

Pour interagir avec un mote, on peut ouvrir un shell avec la commande (dans le répertoire du projet) :

```
make TARGET=openmote BOARD=openmote-cc2538 PORT=/dev/ttyUSB1 login
```

3.3 Déploiement d'un sniffer

Un sniffer permet d'écouter les messages transmis sur un canal spécifique.

Pour flasher le code du sniffer sur un OpenMote il faut se rendre dans le dossier `examples/sensniff` et taper la commande :

```
make TARGET=openmote BOARD=openmote-cc2538 PORT=/dev/ttyUSB1 sensniff.upload
```

Il faut ensuite aller dans le dossier `tools/sensniff` et lancer le script `sensniff.py`, en indiquant le port sur lequel est branché le sniffer :

```
python sensniff.py -d /dev/ttyUSB1
```

Une fois le script lancé, on peut changer le canal sur lequel écoute le sniffer en saisissant le numéro de canal voulu (compris entre 11 et 26).

Pour tous les tests du TP, le Channel Hopping est désactivé et on utilise le canal 20 pour que tous les messages puissent être entendus par le sniffer.

Pour pouvoir observer les paquets reçus, il faut lancer Wireshark et configurer une nouvelle interface. Pour cela, il faut aller dans `Capture` → `Options` → `Gérer les interfaces` → `Nouvelle Interface` (onglet Pipes) → saisir `/tmp/sensniff`.

Enfin, dans les préférences de Wireshark, cocher `TI CC24xx FCS format` dans `Protocoles` → `IEEE 802.15.4`.

3.4 Reflasher un OpenMote B

Si un programme Contiki est déjà sur la carte, on ne peut pas la reflasher directement sur le port USB. Dans ce cas, on a le message suivant :

```
Connecting to target...
ERROR: Timeout waiting for ACK/NACK after 'Synch (0x55 0x55)'
```

Il faut alors utiliser un JTAG et l'application Uniflash[1]. Si besoin, l'enseignant pourra vous fournir un ordinateur et une carte de développement qui vous permettra de flasher la carte.

Pour effacer le programme sur la carte, il faut dans Uniflash :

- sélectionner le port USB et la carte (pour l'OpenMote B, il s'agit de cc2538SF53)
- dans l'onglet "Settings & Utilities", sélectionner "Erase Flash" avec l'option "Entire Flash"

On peut ensuite reprogrammer, par le port USB, la carte avec le programme voulu.

4 Test RPL

4.1 Configuration d'un mote racine

Comme pour OpenWSN, le mote racine doit être branché à l'ordinateur.
Pour créer un réseau TSCH, il faut utiliser la commande suivante dans le shell :

```
tsch-set-coordinator 1
```

On peut alors récupérer les informations sur ce réseau avec la commande :

```
tsch-status
```

Il faut ensuite configurer le mote en tant que racine du réseau :

```
rpl-set-root 1
```

On peut aussi obtenir des informations sur les différents paramètres RPL du noeud :

```
rpl-status
```

La commande `help` permet de récupérer la liste des commandes disponibles (afficher les voisins, les routes...).

[1] http://www.ti.com/tool/UNIFLASH

4.2 Visualisation du réseau dans le shell

Le shell affiche régulièrement des messages listant les voisins ou les liens avec les voisins de la racine :

```
    [INFO: RPL ] nbr: fe80::212:4b00:1935:5644  277,   143 =>   420 --  2  ba
  (last tx 0 min ago)
    [INFO: RPL ] nbr: fe80::212:4b00:1935:558f  290,   191 =>   481 --  1   a
  (last tx 1 min ago)
    [INFO: RPL ] nbr: end of list
    [INFO: RPL ] links: 3 routing links in total (Periodic)
    [INFO: RPL ] links: fd00::212:4b00:1935:51fd  (DODAG root) (lifetime: infinite)
    [INFO: RPL ] links: fd00::212:4b00:1935:5644  to fd00::212:4b00:1935:51fd
(lifetime: 1620 seconds)
    [INFO: RPL ] links: fd00::212:4b00:1935:558f  to fd00::212:4b00:1935:51fd
(lifetime: 1680 seconds)
```

4.3 Ping d'un OpenMote B

Pour pinger un noeud du réseau, on peut entrer la commande suivante dans le shell de la racine :

```
ping fd00::212:4b00:1935:558f
```

Au début, deux motes sont directement reliés à la racine (pas de saut) :

```
Routing links (4 in total):
-- fd00::212:4b00:1935:51fd  (DODAG root) (lifetime: infinite)
-- fd00::212:4b00:1935:5648  to fd00::212:4b00:1935:51fd (lifetime: 1620 seconds)
-- fd00::212:4b00:1935:558f  to fd00::212:4b00:1935:51fd (lifetime: 1800 seconds)
-- fd00::212:4b00:1935:5644  to fd00::212:4b00:1935:5648 (lifetime: 1680 seconds)
```

En éloignant un des motes, on observe que la table de routage (commande `routes`) se met à jour :

```
Routing links (4 in total):
-- fd00::212:4b00:1935:51fd  (DODAG root) (lifetime: infinite)
-- fd00::212:4b00:1935:5648  to fd00::212:4b00:1935:51fd (lifetime: 1620 seconds)
-- fd00::212:4b00:1935:558f  to fd00::212:4b00:1935:5648 (lifetime: 1800 seconds)
-- fd00::212:4b00:1935:5644  to fd00::212:4b00:1935:5648 (lifetime: 1680 seconds)
```

En pingant le mote qu'on vient d'éloigner, on voit alors que le message fait un saut :

```
ping fd00::212:4b00:1935:558f
...
Pinging fd00::212:4b00:1935:558f
...
[INFO: RPL ] SRH Hop fd00::212:4b00:1935:5648
...
[INFO: TSCH ] send packet to 0012.4b00.1935.5648 with seqno 249, queue 1 1, len
21 49
...
Received ping reply from fd00::212:4b00:1935:558f, len 4, ttl 63, delay 250 ms
```