OS embarqués : projet

Thibaut EHLINGER et Benjamin HERB
Université de Strasbourg

Mardi 13 décembre 2016

Dans ce rapport, nous allons vous présenter notre travail concernant le TP2 d'IoT. Pour rappel, il fallait dans un premier temps modifier certains paramètres (PanID, canal radio) et compiler une version de `Contiki OS` compatible avec les nœuds de la plate-forme FIT/IoT-lab. Ensuite il fallait flasher les six nœuds qui nous étaient attribués avec le binaire obtenu. Puis, nous avons dû mettre en œuvre RPL, Foren6 et Stack RIME. Nous allons vous présenter les différentes étapes de ce TP.

1 Contiki OS

Contiki OS est un OS open source spécialisé dans les microcontrôleurs à basse consommation d'énergie reliés à un réseau. Dans un premier temps, nous avons modifié le code de Contiki OS. Pour modifier le canal il fallait modifier `RF2_XX_CHANNEL` dans le fichier `platform/openlab/radiorf2xx.c`. Nous lui avons affecté la valeur **17**. Pour le PanID, nous avons fait un `define IEEE802154_CONF_PANID` et nous avons utilisé l'ID `0x0007`.

Pour compiler Contiki OS, il fallait utiliser la commande `make` avec la bonne cible :

```
$ make target=iotlab-m3
```

Une fois le binaire généré, nous l'avons téléversé sur nos contrôleurs avec

```
$ node-cli -i 56476 -l strasbourg,m3,2+30-34 -up tutorial_m3.elf
```

2 RPL

Une fois nos contrôleurs dotés d'un OS, nous avons mis en œuvre RPL, un protocole de routage IPV6 prévu pour les WSN (*Wireless sensor network*). RPL construit un DODAG (*Destination Oriented Directed Acyclic Graph*) orienté vers le BR (*Border router*) de notre réseau.

Pour information, notre sous-réseau a pour adresse IPv6 `2001:660:4701:f0a6/64`. Notre *border-router* est le nœud numéro **`m3-2`**. Pour déployer notre réseau nous avons dû :

1. lancer `tunslip6` sur le serveur SSH ;

```
$ sudo tunslip6.py 2001:660:4701:f0a6::1/64 -L -a m3-1 -p 20000
slip connected to ``172.16.10.1:20000''
```

2. affecter au nœud `m3-2` son rôle de *border-router* ;

```
$ node-cli --update ~/border-router.iotlab-m3 -l strasbourg,m3,2
```

3. récupérer l'adresse du *border-router*. Il s'agissait de `2001:660:4701:f0a6::a685`
4. déployer un serveur HTTP sur tous les autres nœuds du réseau :

```
$ node-cli --update ~/http-server.iotlab-m3 -l strasbourg,m3,30-34
```

Ci-dessous le texte lu sur l'interface web du BR :

```
Neighbors
fe80::a786
fe80::b286
fe80::a885
fe80::b287
fe80::b086
Routes
2001:660:4701:f0a6::a786/128 (via fe80::a786) 16711327s
2001:660:4701:f0a6::b286/128 (via fe80::b286) 16711398s
2001:660:4701:f0a6::a885/128 (via fe80::a885) 16711398s
2001:660:4701:f0a6::b287/128 (via fe80::b287) 16711397s
2001:660:4701:f0a6::b086/128 (via fe80::b086) 16711396s
```

Et le texte lu sur l'un des autres nœuds :

```
 Neighbors

fe80::a685 PREFERRED
fe80::b286
fe80::b287
fe80::b086
fe80::a885

Default Route

fe80::a685

Routes
```

3 Foren6

Dans cette section, nous allons déployer Foren6 sur l'un de nœuds de notre réseau dans le but de visualiser la construction du graphe RPL. Foren6 est un outil d'analyse non-intrusif de réseaux 6LowPan . Pour visualiser les données, nous avons téléchargé et compilé les