MQTT :

-Protocole

-Historique : créé en 1999 par Andy-Stanford Clark employé d’IBM et Arlen Nipper (Arcom (Eurotec))

-Principe : Protocole de messagerie légère (moins consommateur en énergie) beaucoup utilisé en IoT

qui facilite la transmission des données de télémétrie pour les clients réseau dont les ressources sont limitées. Il suit un modèle de communication de type publication/abonnement via un broker

exo1 :

Sur un unique PC : demonstration d’un échange d’information

=> serveur local 127.0.0.1

=> run Mosquitto : launchctl load ~/Library/LaunchAgents/homebrew.mxcl.mosquitto.plist

=> lancement du sucriber : mosquitto\_sub -h 127.0.0.1 -t topic

=> Publisher client : mosquitto\_pub -h 127.0.0.1 -t topic -m "Hello world!”

exo2 : Entre 2 pc + recuperation détourner d’informations

Sur un unique PC : demonstration d’un échange d’information

=> brocket (sur mon pc)

=> run Mosquitto : launchctl load ~/Library/LaunchAgents/homebrew.mxcl.mosquitto.plist

=> lancement du sucriber : mosquitto\_sub -h [mon ip] -t topic

=> Publisher client : mosquitto\_pub -h 127.0.0.1 -t topic -m "Hello world!”

Un PC extérieur peut récupérer toutes les informations. Meme avec un username et password les informations sont envoyées en clair sur le réseau (a verifier)

exo3 : restriction des accès aux informations

On peut utiliser la fonctionnalité “password” prévu dans le logiciel

=>creation du fichier contenant les user : mosquitto\_passwd -c passwordfile aurelien

=> ajout de l’user “test” mosquitto\_passwd -b passwordfile test aaaa

=>creation d’un fichier de config (mosconf.conf) contenant :

“allow\_anonymous false

password\_file /Users/lactoriaus/CA/passwordfile”

=> lancement du broker avec les config :

/usr/local/sbin/mosquitto -c /Users/lactoriaus/CA/mosconfig.conf

exo4 : sniff avec wireshark

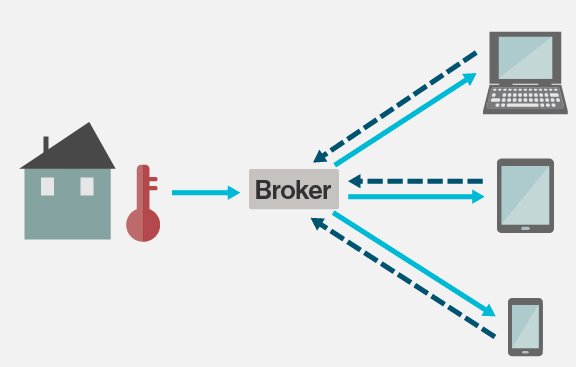
Security de exo1 ?

/usr/local/sbin/mosquitto -c /Users/lactoriaus/CA/mosconfig.conf -p 8883

sudo launchctl unload -w /System/Library/LaunchDaemons/ssh.plist

Le MQTT est un protocole simple utilisé largement dans l’IoT. Il demande très peu de resource (fonctinne très bien sur arduino). Il a cependant un gros problème se sécurité : tous les données sont en clair sur le réseau. Il est possible d’utilisé un système d’authentification (username et password), cependant toutes les informations sont envoyées en clair sur le réseau y compris le username et password. Il est facile de récupérer les informations sur wireshark (faire test pour avoir screen avec username/pwd et trame de données)

Pour sécuriser les données on peut les cryptés en utilisant le protocole SSL. On crypte les données entre le subscriber et le broker. On peut également crypter entre le publisher et broker mais dans l’IoT cela couterait beaucoup d’argent pour le developper et une arduino n’est surement pas assez puissante.

Le SSL n’empêche pas un pirate de se mettre entre la communication entre le subscriber et le broker et usurpé son identité ! on pourrait utiliser un VNP pour encore plus sécurisé mais ça coute encore plus cher.

C’est le défi des années à venir : sécuriser à bas cout !

Broker : le courtier en français sert d’intermédiaire entre un publisher (capteur) et un subscriber (interface)

Publisher : device qui va envoyé des informations au broker, il est souvent un capteur dans l’IoT

Subscriber : device qui va se connecter au broker pour récupérer des informations venant du publisher.