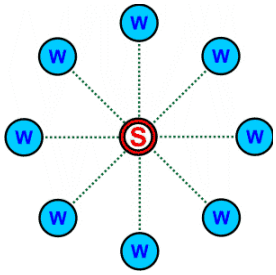
Lightweight publish-subscribe application protocol



Si richiede di progettare e implementare un protocollo applicativo di pubblicazione-iscrizione simile a quello MQTT e testarlo su una topologia composta da 8 nodi e PAN coordinator che agisce come un MQTT broker.
Si devono implementare le seguenti funzioni:

1. CONNECTION: una volta attivata la struttura, ogni nodo invia un messaggio di CONNECT al PAN coordinator, che risponde con un messaggio di CONNACK.

2. SUBSCRIBE: ogni nodo può iscriversi a tre topic [TEMPERATURA, UMIDITÀ e LUMINOSITÀ]. Per iscriversi a questi topic il node invia un messaggio di SUBSCRIBE con il propio ID, il topic a cui richiede di iscriversi e il QoS; il messaggio di SUBSCRIBE riceve una risposta di iscrizione dal PAN coordinator tramite il messaggio di SUBACK. *[Nel nostro caso abbiamo implementato anche l'iscrizione multipla a più topic, 3 nodi si iscrivono a due topic ciascuno]*

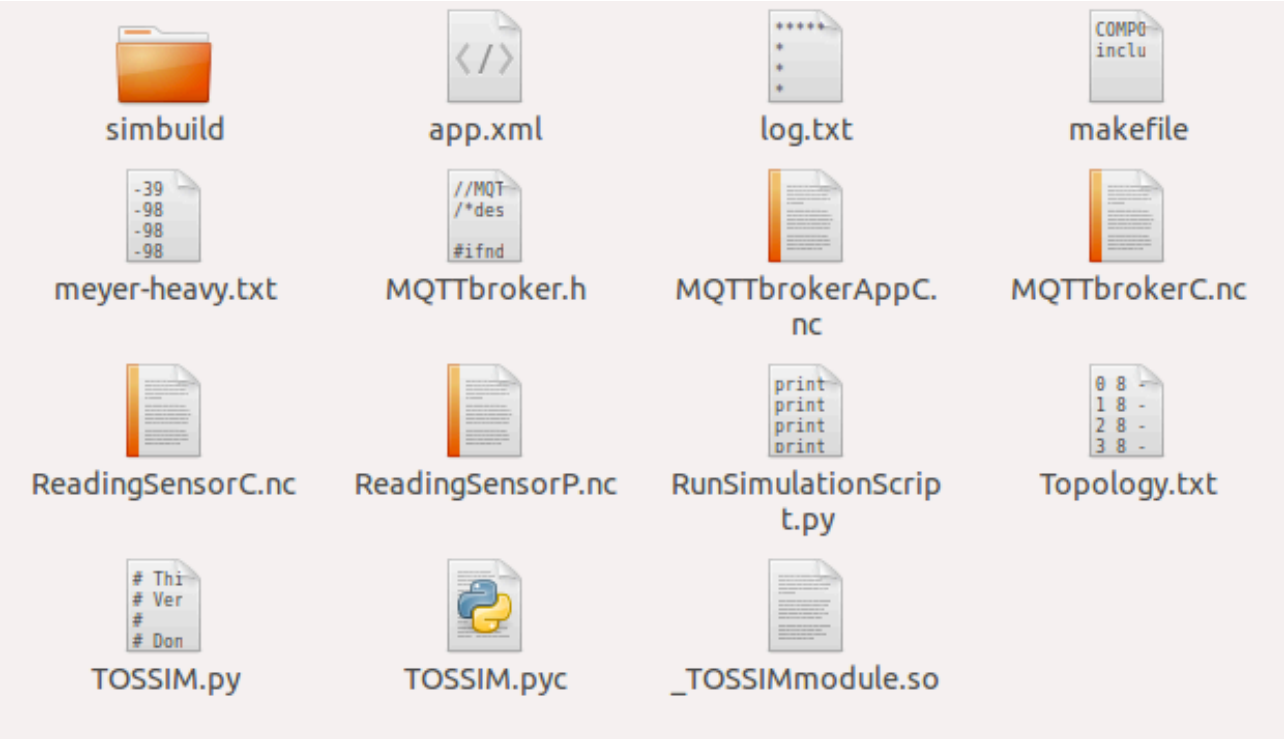
3. PUBLISH: il nodo pubblica il valore letto dal sensore al PAN coordinator che farà inoltro del messaggio ai nodi che sono iscritti al topic del messaggio. [*Nel nostro caso abbiamo implementato un solo nodo che pubblica i valori*]

4. Gestione del QoS: nel PUBLISH message abbiamo due livelli di QoS:

- 0 = il nodo trasmette il messaggio solo 1 volta.
- 1 = il nodo continua a ritrasmettere fino a che il PAN coordinator non risponde con un messaggio di PUBACK

Nella nostra simulazione abbiamo utilizzato come ambiente di simulazione TOSSIM.

Struttura del progetto



- log.txt = è il file che contiene i risultati della simulazione che sono stampati a video al fine di verificare la corretta esecuzione.
- meter-heavy.txt = contiene un elenco di valori del rumore di canale.
- MQTTbroker.h = rappresenta la “libreria” della nostra simulazione, contiene le strutture dati e le variabili della nostra simulazione (campi del pacchetto utilizzato, strutture per tenere conto delle iscrizioni ai topic, etc…)
- MQTTbrokerC.nc = contiene il codice della simulazione, la gestione degli eventi e le funzioni implementate per la gestione della simulazione.
- MQTTbrokerAppC.nc = contiene i collegamenti tra le interfacce e i componenti.
- ReadingSensorC.nc e ReadingSensorP.nc = sono dei file che gestiscono la lettura del valore dal sensore in seguito alla scadenza di un timer, sono degli esempi che abbiamo trovato all’interno della macchina virtuale.
- RunSimulationScript.py = gestisce l’esecuzione della simulazione caricando la topologia della rete, il modello del canale disturbato dal rumore e tutti gli eventi necessari alla simulazione.

Eventi implementati

AMControl.startDone / AMCControl.stopDone = attivazione di un timer di ogni nodo, al termine di esso è possibile inviare la mia richiesta di connessione al broker.

manage_conn.sendDone = se la connessione del nodo alla rete ha avuto successo (pacchetto ricevuto e confermato correttamente dall'ack), allora è possibile inviare la richiesta di iscrizione per un certo topic, altrimenti viene effettuato un altro tentativo per la connessione.

manage_sub.sendDone = se la richiesta di connessione è stata accettata dal broker allora è possibile registrare l'iscrizione del nodo al topic specificato e il relativo QoS nei registri del broker. I nodi iscritti possono ora ricevere messaggi inerenti il/i topic a cui sono registrati ed, eventualmente, pubblicare aggiornamenti.

manage_reqpub.sendDone = uno o più nodi *[nel nostro caso è stato scelto il nodo 0]* possono inviare una richiesta di pubblicazione al broker, se essa viene accettata allora il nodo può pubblicare una rilevazione riguardante il topic desiderato.
.

Read.readDone = una volta richiesta la pubblicazione, il nodo acquisisce il dato alla scadenza di un timer e costruisce un pacchetto da inviare al broker contenente tale valore che verrà, a seguito della conferma con ack, inoltrato ai nodi iscritti al topic su cui si sta pubblicando.

manage_pub.sendDone = il pacchetto contenente il dato rilevato può essere ricevuto correttamente e inoltrato ad altri nodi, oppure possono esserci problemi di vario genere; in questo secondo caso viene deciso come procedere in base al QoS del pubblicante: se pari a 0 il messaggio viene considerato perso, mentre se pari a 1 il nodo pubblicante continuerà a inviare il pacchetto al broker fino alla ricezione dell'ack di conferma.

manage_fwd.sendDone = è la conclusione della fase di inoltro per ogni nodo iscritto al topic su cui viene pubblicato l'aggiornamento. Se è ricevuto correttamente, l'inoltro è concluso, altrimenti è da ripetere.

Funzioni implementate

- task void nodeReqConn();
- task void nodeReqSub();
- task void nodeReqPub();
- task void pubToBroker();
- task void forward();

nodeReqConn() = costruisce il messaggio di richiesta di connessione e ne richiede l'ack al broker per verificare la corretta connessione.

nodeReqSub() = costruisce il messaggio di richiesta di iscrizione con il topic e il QoS desiderato e richiedo l'ack di conferma di ricezione al broker.

nodeReqPub() = costruisce il messaggio di richiesta di pubblicazione e lo invio al broker.

pubToBroker() = legge il valore dal sensore.

forward() = una volta ricevuto il messaggio il broker controlla le registrazioni al topic del messaggio ricevuto, nel caso un nodo sia iscritto invia il messaggio ad esso.