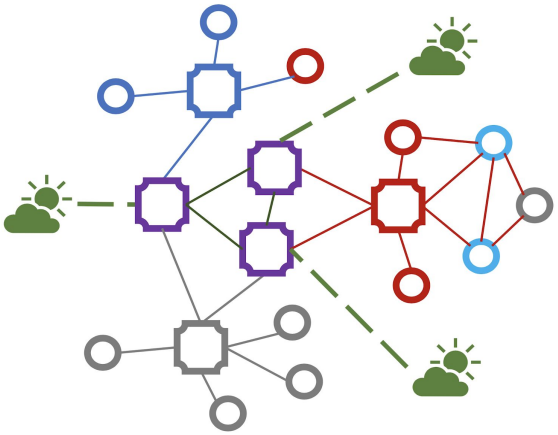
Argomenti:

- Wireless Network (protocolli MAC CSMA/CA - MACA e mobilità nelle reti)
- Mobilità nelle Ad-Hoc Network (Protocolli di routing DSR, AODV, DYMO)
- Wireless Sensor Network (Duty Cycle ed efficienza)
- Protocolli delle WSN (Protocolli MAC, S-MAC, B-MAC, X-MAC, protocolli per il sensing distribuito e per il routing, clustering nelle WSN con DMAC)
- Embedded Systems
- Segnali (Introduzione, Fourier, Conversione analogica-digitale)
- IEEE 802.15.4 e Zigbee
- Bluetooth
- MQTT
- Indoor localization

1) **Interoperability: Slides-01_Intro**



L'implementazione di una soluzione IOT non è solitamente un grosso problema, possiamo partire dal livello fisico fino ad arrivare all'implementazione dell'application layer, questo tipo di implementazione è quello che viene chiamato "vertical silos" e questo comporta un problema, la soluzione infatti funzionerà solamente da sola, solamente con i nostri dispositivi e dei cambiamenti richiedono un intervento umano. Inoltre abbiamo un vendor lock-in perchè non possiamo utilizzare componenti di altri produttori.
La soluzione a questo problema è introdurre degli standard che possono essere ad esempio il Wi-Fi o il Bluetooth o ad esempio Zigbee.

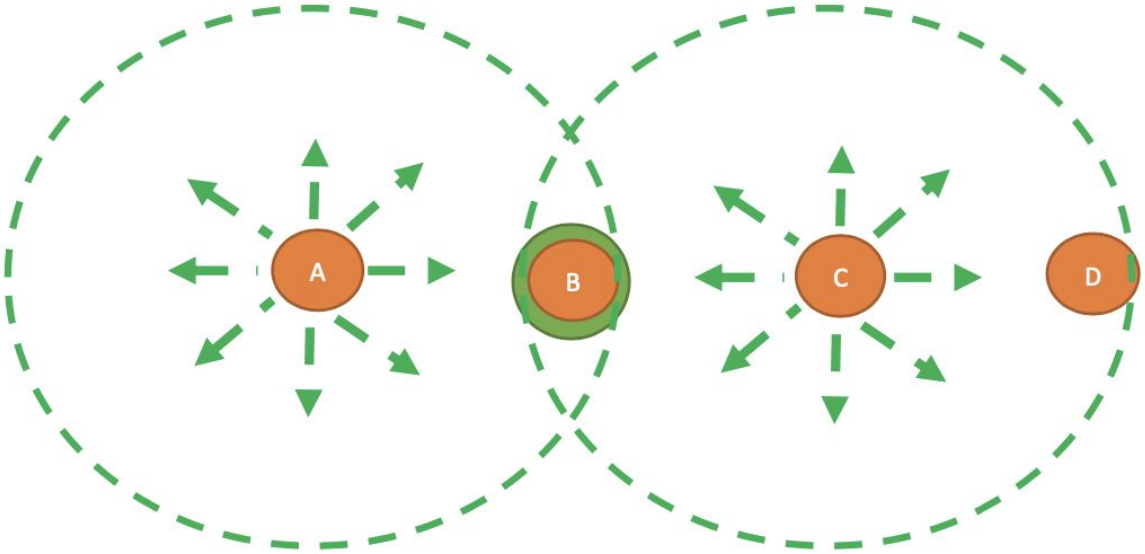
Questi standard solitamente comportano degli accordi tra produttori differenti e sono solitamente motivati da una riduzione del costo di sviluppo di una certa tecnologia, solitamente si decide di sviluppare uno standard quando la tecnologia è ormai matura e la fonte di guadagno è ormai un'altra e non vogliamo più perdere tempo e implementare cose già implementate. Questo problema di interoperabilità era inizialmente limitato alle comunicazioni wireless ma ultimamente è diventato un problema che si presenta anche a livello applicazione e infatti sono stati sviluppati vari protocolli di livello applicazione per l'IOT come ad esempio Zigbee o MQTT.
Se però si creano troppi standard allora l'interoperabilità si sposta da dispositivi di vendor differenti a standard differenti, per gestire questi standard che non sono compatibili tra loro è stata introdotto un gateway a livello applicazione che non solo fa una traduzioni dei protocolli più a basso livello ma va anche a fare un mapping dei differenti comportamenti dei protocolli a livello applicazione.
Quando ci troviamo in questa situazione abbiamo varie possibilità:

- Possiamo avere nella rete solamente dispositivi di un certo produttore che usano tutti lo stesso protocollo e il gateway.
- Potremmo avere dispositivi di vendor differenti ma con lo stesso protocollo e il gateway.
- Potremmo avere dispositivi di vari vendor con vari protocolli e un gateway che serve per l'integrazione.
- Possiamo avere dispositivi di vari vendor con vari protocolli e vari gateway distribuiti nella rete che vengono utilizzati per l'integrazione, questo è in particolare il caso che abbiamo nella immagine.

Perchè ho più di un gateway?

In questo modo evitiamo che un singolo gateway abbia le informazioni di tutte le traduzioni.

2) Hidden Terminal: Slides-02_wireless network



Il problema dell'Hidden Terminal è dovuto all'utilizzo di CSMA come protocollo MAC e i due nodi non sono in range, quindi non sentono l'uno che l'altro sta a sua volta cercando di comunicare, il problema principale è dovuto al fatto che abbiamo l'interferenza nel destinatario e non al mittente.

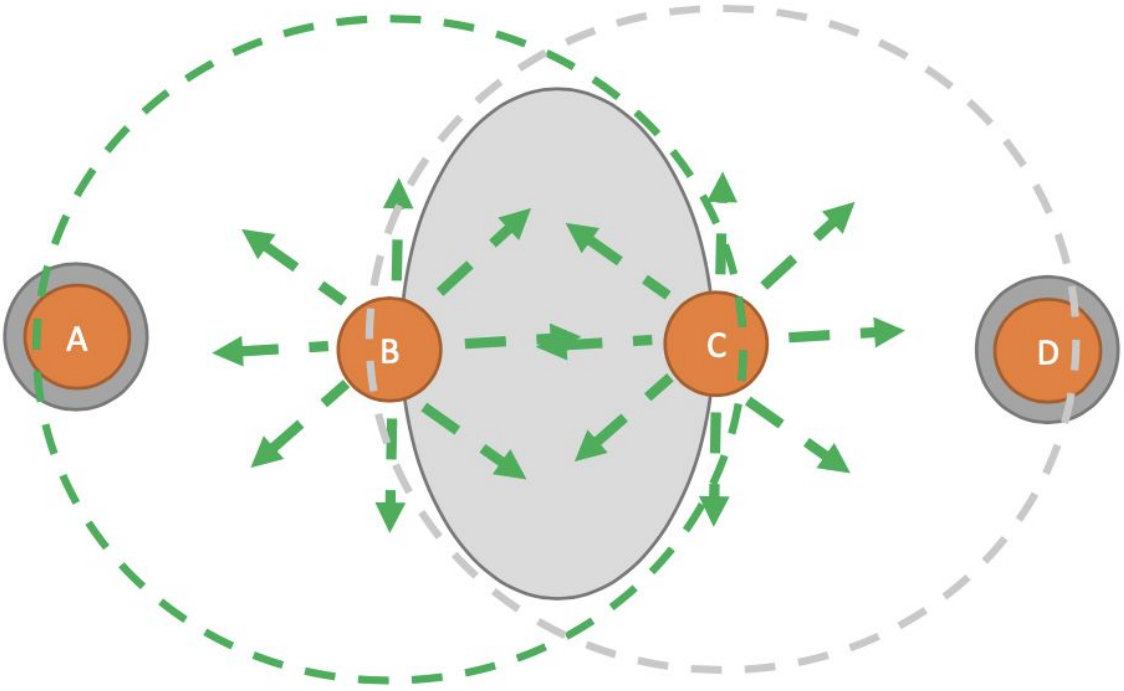
In CSMA/CD verifichiamo l'interferenza al mittente e non nel receiver, nelle wired network è ok ma non va bene nelle reti wireless perchè con il cavo comunque la trasmissione la senti mentre se hai una rete wireless non si sente che sta avvenendo la comunicazione se non è nel range.

Con il CSMA infatti il nodo che vuole trasmettere deve controllare il mezzo, se lo trova libero può inviare il pacchetto, se è occupato attende. Se poi il mittente identifica una collisione allora ferma l'invio del pacchetto e attende un certo numero di contention slots che si basa su un backoff esponenziale.
Uno dei problemi del CSMA/CD è che il controllo del mezzo viene fatto solamente dal mittente che vuole inviare il pacchetto, quindi si presentano problemi come ad esempio il problema del terminale nascosto.
In questo caso abbiamo il nodo A che vuole comunicare con B e il nodo C che vuole comunicare con B.
Quello che succede è che A controlla il mezzo e nel suo "radio range" vede che nessuno sta trasmettendo, quindi invia il pacchetto. Allo stesso tempo però C fa la stessa identica cosa e non trova il mezzo occupato perché il nodo A è fuori dalla portata del suo range quindi quello che succede è che si verificherà una collisione nel nodo B.
Quindi in pratica C non riesce a vedere la A e la C è nascosta rispetto alla comunicazione tra A e B, ecco perché è chiamato hidden terminal problem.

- Se prendiamo un mac protocol tradizionale come CSMA/CD perchè non funziona? Perchè la trasmissione avviene da un mittente e poi però ci accorgiamo della collisione nel destinatario
- MACA RTS CTS
- Con RTS e CTS possono esserci fallimenti? Se gli RTS di due nodi avvengono allo stesso momento allora ci sta una collisione
- In quali standard che abbiamo visto si utilizza RTS-CTS

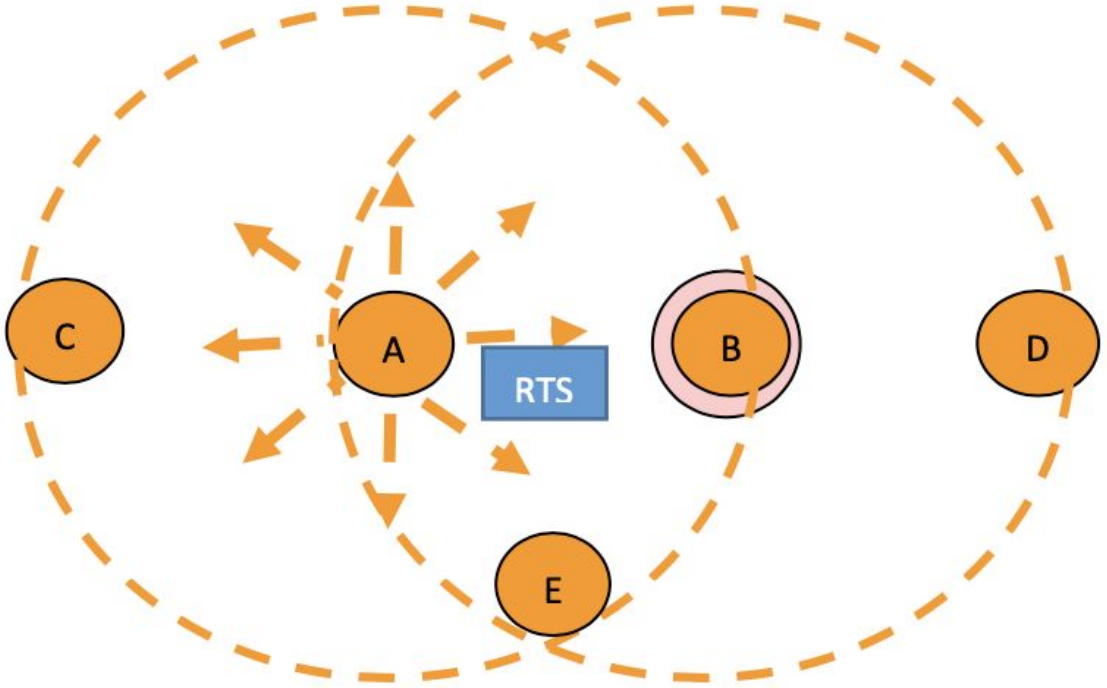
802.11

3) Exposed Terminal: Slides-02_wireless network



Il problema dell'Exposed Terminal è dovuto all'utilizzo di CSMA come protocollo MAC.
Con il CSMA infatti il nodo che vuole trasmettere deve controllare il mezzo, se lo trova libero può inviare il pacchetto, se è occupato attende. Se poi il mittente identifica una collisione allora ferma l'invio del pacchetto e attende un certo numero di contention slots che si basa su un backoff esponenziale.
Uno dei problemi del CSMA/CD è che il controllo del mezzo viene fatto solamente dal mittente che vuole inviare il pacchetto, quindi si presentano problemi come ad esempio il problema dell'exposed terminal.
Nello specifico qua abbiamo il nodo B che sta trasmettendo al nodo A e abbiamo il nodo C che vorrebbe trasmettere al nodo D.
C controlla il mezzo e vede che è occupato perchè B che sta trasmettendo è all'interno del suo range e quindi si blocca e attende, in realtà se C volesse trasmettere a D potrebbe farlo perchè il mezzo sarebbe libero e quindi le due trasmissioni potrebbero avvenire in parallelo ma in realtà la C attende comunque a causa di questo problema.
Quindi in questo caso la C è esposta rispetto alla comunicazione che avviene tra B ed A.

4) RTC/CTS: Slides-02_wireless network



In questo caso utilizziamo un protocollo che è chiamato MACA.
Questo risolve in parte i problemi di CSMA.
Il protocollo funziona in questo modo:

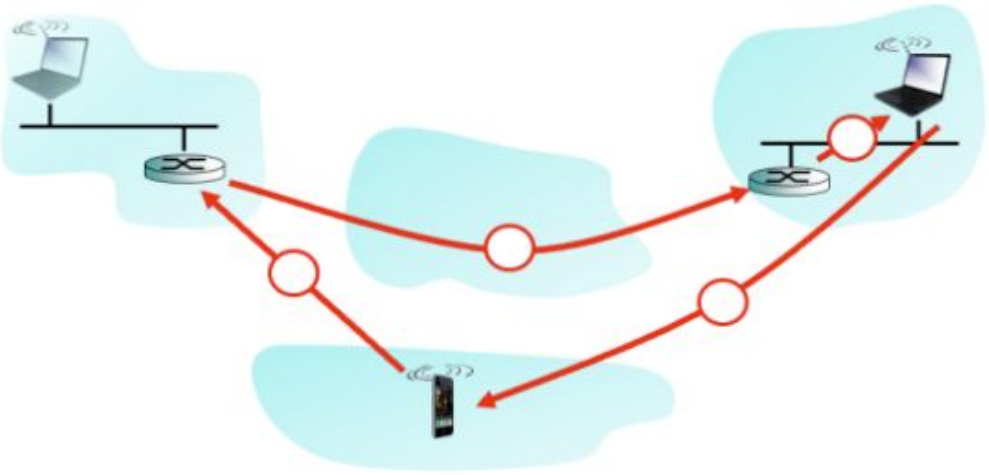
- A vuole inviare un pacchetto a B
- Per prima cosa A invia un RTS (Request to Send) al nodo B per chiedere il permesso di inviare il pacchetto. All'interno dell'RTS scriviamo la quantità di dati che vogliamo inviare con il pacchetto.
- Se B può ricevere invia ad A un CTS ovvero un Clear To Send che contiene i dati come ad esempio la lunghezza del pacchetto copiati dall'RTS.
- Ora che ha ricevuto il CTS, il nodo A invia il pacchetto a B

Per quanto riguarda gli altri nodi della rete:

- C sente solamente l'RTS e non il CTS quindi sarà libero di comunicare.
- D riceve il CTS quindi dovrebbe rimanere in silenzio e non comunicare con B
- E riceve sia il CTS che l'RTS quindi non ha problemi e rimane in silenzio

Il problema si verifica se ad esempio C e B inviano l'RTS simultaneamente ad A, in questo caso abbiamo una collisione e si utilizza un **backoff esponenziale per inviare di nuovo l'RTS ed evitare la collisione. Se ho la collisione non viene generato un CTS.**
Per risolvere il problema della collisione esiste un altro protocollo che è chiamato Macaw e che prevede il controllo del mezzo prima dell'invio dell'RTS.

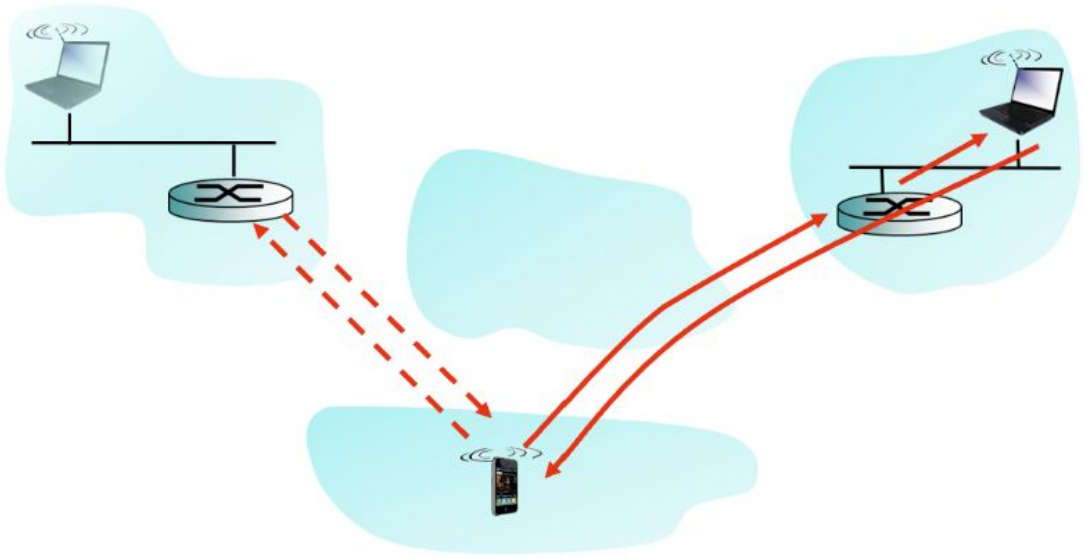
5) Mobile Networks: slides-03A_managing mobility in cellular networks



Siamo nel caso della comunicazione in una rete mobile e in particolare nell'Indirect Routing. Abbiamo un home network che è la cella a cui si è connesso inizialmente il dispositivo. Poi abbiamo all'interno dell'home network un home agent che si occupa di gestire l'ingresso e l'uscita dei dispositivi dalla rete. Al dispositivo che entra nella rete viene associato un permanent address.
Quando il dispositivo si sposta in un'altra rete allora entra in una foreign network in cui c'è un foreign agent e gli viene assegnato un care of address.

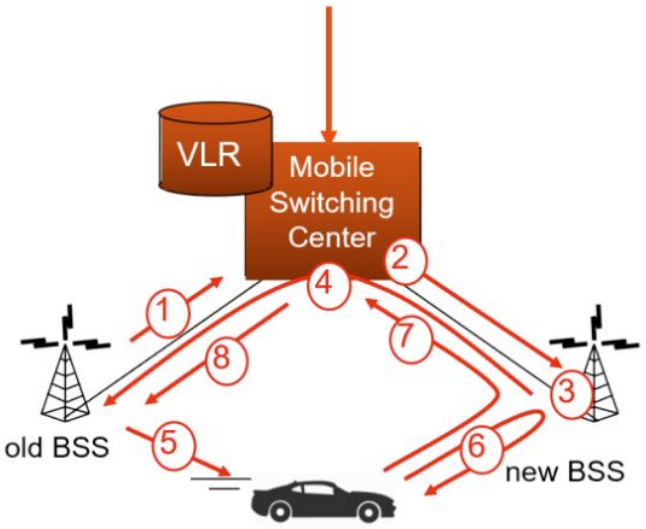
Quando qualcuno vuole contattare questo dispositivo, si conosce il suo permanent address quindi si va nel suo home network, l'home agent si accorgerà che il dispositivo si trova ora in un'altra rete e quindi inoltrerà il pacchetto al care of address. Il dispositivo ora potrà rispondere alla richiesta.
Possibili problemi di efficienza a causa del triangle routing, il dispositivo mittente e il dispositivo destinatario potrebbero già essere nella stessa rete e in questo caso facciamo un giro più lungo per comunicare.

6) Mobile Networks: slides-03A_managing mobility in cellular networks



Siamo in una rete in una cellular network con i dispositivi che possono muoversi da un home network ad un foreign network. In questo caso abbiamo un direct routing nel senso che il mittente del pacchetto che vuole inviarlo al dispositivo che si è spostato comunica prima con l'home network e quindi con l'home agent che gli risponde comunicando l'indirizzo del foreign network in cui si trova ora il nuovo dispositivo. Preso questo indirizzo ora il mittente può contattare direttamente il destinatario senza passare dall'home network.
Il problema in questo caso si pone se il dispositivo che vogliamo contattare cambia la sua foreign network dopo che il mittente ha ottenuto il foreign address.

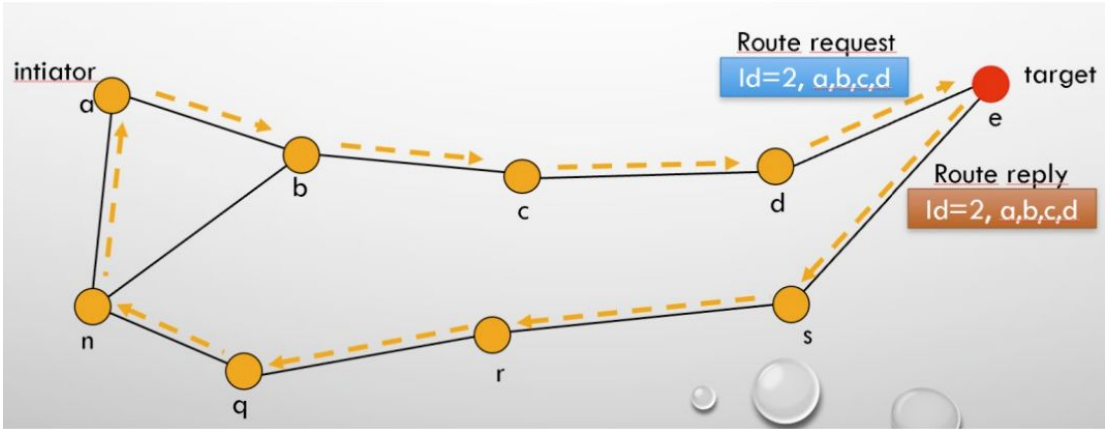
7) Mobile networks: slides-03A_managing mobility in cellular networks



L'handoff è l'operazione che avviene quando trasferiamo una chiamata o uno scambio di dati cellulari da una cella di una cellular network ad un'altra cella o da un canale all'altro. Assumiamo inizialmente che old BSS e new BSS condividono lo stesso Mobile Switching Center e che deve avvenire il rerouting. Il rerouting avviene sia perché il segnale tra la base station e il dispositivo mobile potrebbe essersi deteriorato sia perchè la cella potrebbe essere diventata sovraccarica.

- La old BSS informa il MSC che deve essere eseguito l'handoff e comunica anche la new BS verso cui il dispositivo mobile si muoverà.
- Il MSC contatta la nuova BSS e comunica che sta per avvenire un handoff in modo che la nuova BSS sia preparata.
- La nuova BSS attiva un canale radio da far utilizzare dal dispositivo mobile che deve svolgere l'handoff.
- Il nuovo BSS segnala al MSC (che inoltra al vecchio BSS) che è pronto per l'handoff e quindi che il dispositivo mobile può eseguire l'handoff. Vengono comunicate tutte le informazioni necessarie per eseguire l'handoff del dispositivo.
- Il dispositivo mobile viene informato dell'handoff, fino a questo momento ne era all'oscuro.
- Il dispositivo mobile e il nuovo BSS si scambiano dei messaggi per attivare il nuovo canale.
- Il dispositivo mobile comunica che l'handoff è avvenuto correttamente al nuovo BSS, questo inoltra il messaggio al MSC e lui poi lo inoltra al vecchio BSS che rilascia le risorse che erano state allocate per quel dispositivo mobile.

8) DSR - Slides-03B_Managing mobility in Ad hoc



Con Ad hoc network intendiamo delle reti di host mobili connessi tramite wireless, i nodi sono autonomi, indipendenti, alimentati a batteria e mobili. I pacchetti sono scambiati tramite radio, non abbiamo una infrastruttura fissata per la rete, si tratta di un sistema puramente distribuito in cui non abbiamo un coordinatore centralizzato. Questo tipo di reti vengono utilizzate per la comunicazione in posti dove non è facile arrivare o ad esempio in situazioni di emergenza.
La trasmissione tra i nodi è limitata dalla distanza e quindi si utilizza anche una trasmissione Multi Hop.
Un problema che si verifica in questo tipo di reti riguarda l'accesso al canale wireless condiviso tra tutti i nodi, abbiamo bisogno di un protocollo Media Access Control.

Per il routing all'interno di queste reti abbiamo varie possibili soluzioni, qua viene utilizzato DSR che è un protocollo che, dato un grafo di nodi ci permette di trovare il nodo destinatario che stiamo cercando e anche di gestire lo spostamento dei nodi all'interno della rete.
Il protocollo è decentralizzato e ogni nodo nella rete aiuta la ricerca dei percorsi inoltrando i pacchetti che riceve. In particolare abbiamo due fasi del protocollo, la fase di Route Discovery e la fase di Route Manteinance. La prima la eseguiamo quando vogliamo inviare un pacchetto ma non sappiamo come raggiungere il destinatario, la seconda quando invece avevamo una conoscenza di come raggiungerlo ma poi la topologia è cambiata.

Per quanto riguarda la fase di route discovery con DSR funziona così:

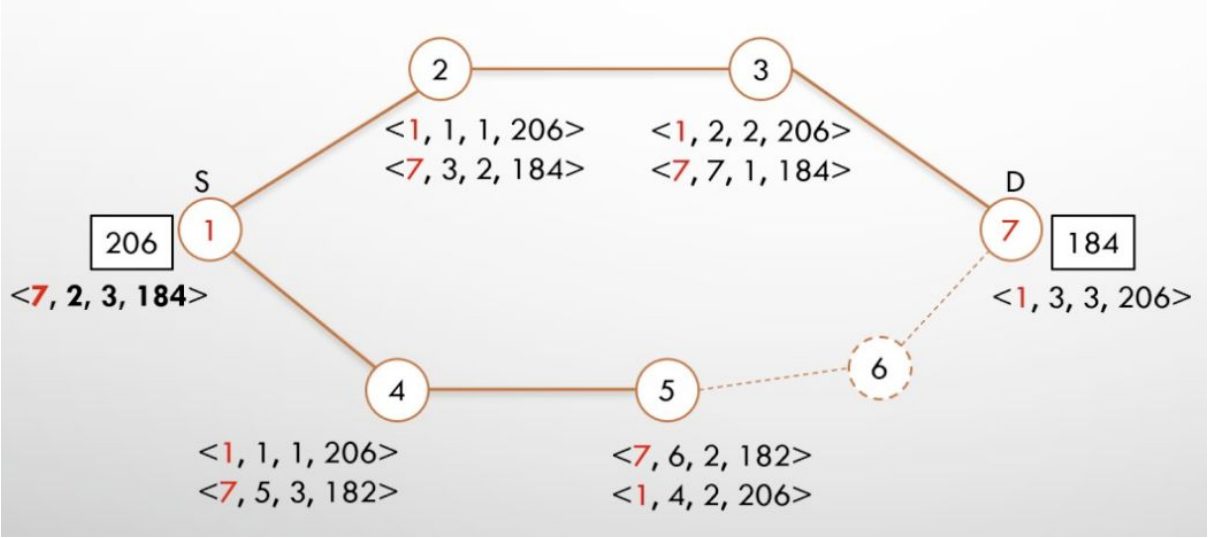
- Il nodo che vuole mandare un messaggio, in questo caso il nodo A invia una route request verso il destinatario E. Il messaggio viene inoltrato ai vicini di A ovvero a B e N.
- B e N a loro volta inoltrano il messaggio ai loro vicini, ad esempio N lo inoltra anche a B (che lo elimina perchè l'ID è uguale) e poi anche ad A che lo elimina perchè A sta già nella lista della route request.
- Quando un nodo riceve la RREQ si aggiunge nella lista della route request e poi inoltra il messaggio al nodo successivo.
- Quando la RREQ arriva al nodo target ci fermiamo, il nodo target può ricevere la RREQ da vari percorsi ma sceglie quella migliore basandosi su latenza e numero di salti.
- Ora il destinatario genera una RREP che contiene la lista della RREQ e che dovrebbe essere inviata al nodo mittente. Per l'invio si utilizza un percorso che era già nella cache del target oppure si svolge nuovamente una route request per trovare il mittente aiutandosi anche con

i dati della RREQ (solo se gli archi sono percorsi bidirezionali). I pacchetti RREP che devono essere inviati ma per cui non conosciamo un percorso per inviarli al target rimangono nel nodo target in un buffer per un certo tempo fino a quando non si trova un percorso.

I nodi intermedi possono rispondere ma se c'è un loop non rispondono per evitarlo.

- Con poco traffico e poca mobilità:
    - Sia DSR che AODV funzionano decentemente e hanno poco overhead dovuto al routing.
- Con tanta mobilità e tanto traffico:
    - AODV ha un overhead più alto per il routing dovuto ai control packets, le route diventano congestionate e devono essere riscoperte.
    - I messaggi di hello di AODV creano congestione.
    - DSR ha il problema di avere più routes e quindi potrebbe essere complicato fare una scelta e prendere la route che preferiamo utilizzare.
    - Se in DSR poi utilizziamo un caching aggressivo e reazioni molto veloci ai cambiamenti allora possiamo avere dei percorsi poco instabili.
    - DSR diventa quindi più instabile in queste condizioni

9) AODV: Slides-03B_managing mobility in adhoc

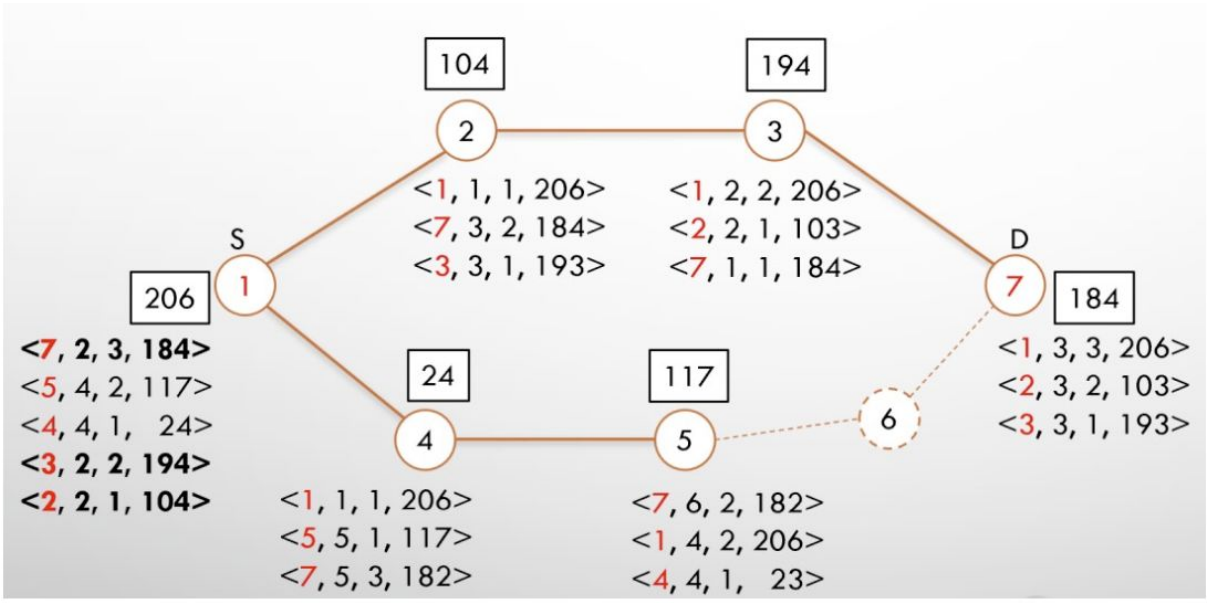


Consideriamo il caso di un altro possibile algoritmo di routing che possiamo utilizzare al posto di DSR nelle ad hoc network. Si tratta di un algoritmo decentralizzato in cui i collegamenti sono bidirezionali. Ogni nodo mantiene una struttura dati per la comunicazione unicast, una per la comunicazione multicast e poi un sequence number che viene aumentato quando cambia la topologia della rete.

Nello specifico nell'immagine vediamo la procedura di route discovery di AODV:

- Il nodo mittente 1 ha un suo sequence number che è 206 e vuole trovare un percorso per arrivare al nodo 7. Per trovare il percorso utilizza un pacchetto RREQ in cui vengono indicati: IP del mittente e del destinatario, Sequence number del mittente e del destinatario e numero di hop. Questo pacchetto viene inoltrato in broadcast ai vicini del nodo mittente. Quando viene inoltrato il pacchetto il sequence number di 1 viene aumentato.
- I nodi 2 e 4 che ricevono la RREQ aggiungono nella routing table una route per arrivare a 1 e quindi aggiungono <1,1,1,206> dove il primo 1 è la destinazione il secondo il prossimo hop e il terzo è il numero di hop per la destinazione.
- Quando un nodo intermedio riceve la RREQ, se conosce già un percorso per la destinazione può rispondere con una RREP indicando il percorso, questo succede solamente se il suo sequence number per la destinazione è maggiore o uguale del sequence number che si trova nel RREQ.
- Il nodo mittente riceve questo RREP e lo memorizza. l'algoritmo continua e anche la destinazione riceve il RREQ e risponde con un RREP che è aggiornato con il suo nuovo sequence number.
- Il nodo mittente riceve questo RREP e lo sostituisce con quello che aveva in precedenza.

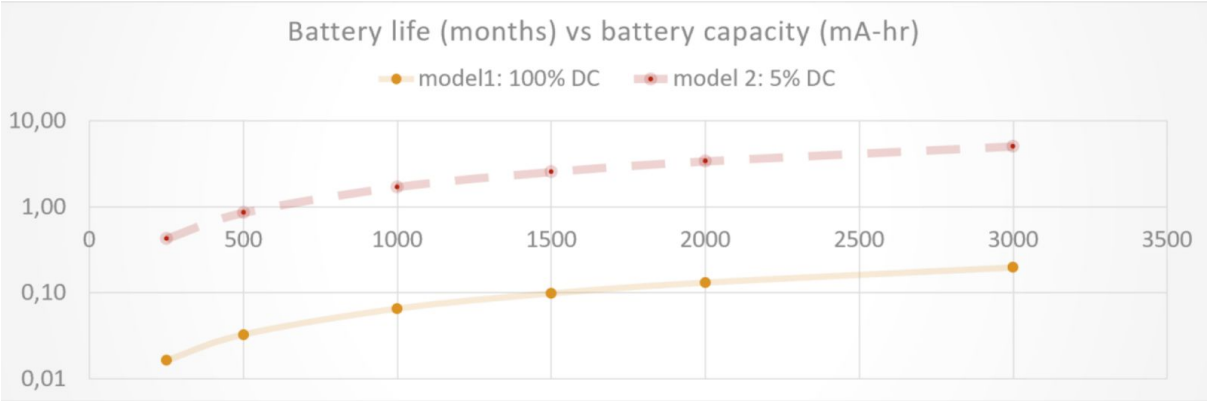
10) DYMO: Slides-03B_managing mobility in adhoc



DYMO è un algoritmo per il routing che mette insieme le caratteristiche di AODV (i protocolli di route discovery e di route maintenance sono simili) con le caratteristiche di DSR (il fatto di portarsi dietro tutte le informazioni quando mandiamo un pacchetto da un nodo all'altro).

Supponiamo di avere sempre il nodo 1 che vuole inviare un pacchetto al nodo 7, il primo nodo che viene contattato è il 2. Quando contatto il nodo 2 gli mando un RREQ in cui inserisco il mio sequence number, il sequence number del destinatario, il mio IP e quello del destinatario e poi un hop count. Quando l'RREQ arriva al nodo 2 lui memorizza nella sua RT i dati del nodo 1, poi l'RREQ viene inoltrato al nodo 3 ma vengono anche aggiunte delle informazioni aggiuntive riguardanti il nodo 2, quando il nodo 3 riceve la RREQ da 2 non solo aggiorna la RT con le nuove informazioni di 1 ma aggiunge anche delle nuove informazioni riguardanti il nodo 2.
Quando un nodo intermedio che riceve la RREQ conosce il percorso per arrivare al nodo target allora manda indietro la risposta, ad esempio il nodo 5 manda indietro un percorso per 7 e quando questo RREP passa per il nodo 4 viene anche memorizzato li il percorso per 7 così come quello per il nodo 5.
Quando questo RREP arriva al nodo 1 lui memorizza tutto, poi per esempio in questo caso quando il nodo 7 che è il target riceve il RREQ allora manda indietro il nuovo percorso e questo ha un sequence number maggiore di quello che è stato ricevuto in precedenza da 1, quindi abbiamo un aggiornamento della routing table.

11) Duty cycle: Slides-04A_WSN Design aspects



Con duty cycle intendiamo la frazione di tempo che un dispositivo passa in stato attivo rispetto al tempo totale che consideriamo.
L'idea del duty cycle è che dato che i sensori fanno una vita ripetitiva (misurazione, invio del dato, memorizzazione...) si è pensato di alternare un periodo in cui si lavora con un periodo in cui il sensore è attivo.

Nello specifico in questo grafico qua sotto vengono confrontati due differenti duty cycle. La curva più bassa rappresenta il dispositivo con duty cycle del 100% mentre quello più in alto ha un duty cycle del 5%. Nel grafico viene utilizzata la curva logaritmica per mostrare in che modo il consumo di batteria dipende dalla scelta del duty cycle.

Possiamo vedere che, anche con un aumento della capacità della batteria, abbiamo comunque che il dispositivo con duty cycle del 100% ha una vita di qualche giorno mentre il dispositivo con duty cycle del 5% avendo dei consumi ridotti può rimanere acceso e funzionante per vari mesi. Quindi in pratica anche se aumentiamo la capacità della batteria (a parità di aumento) il dispositivo con il duty cycle al 100% ha un incremento della sua durata di pochissimo tempo se confrontato con l'incremento che ha invece il dispositivo con il duty cycle al 5%.

Quindi in pratica questo grafico ci indica che se vogliamo un dispositivo che sia efficiente abbiamo la necessità di avere un duty cycle piccolo in modo da far sopravvivere la batteria per mesi.

Come lo riduciamo il duty cycle?

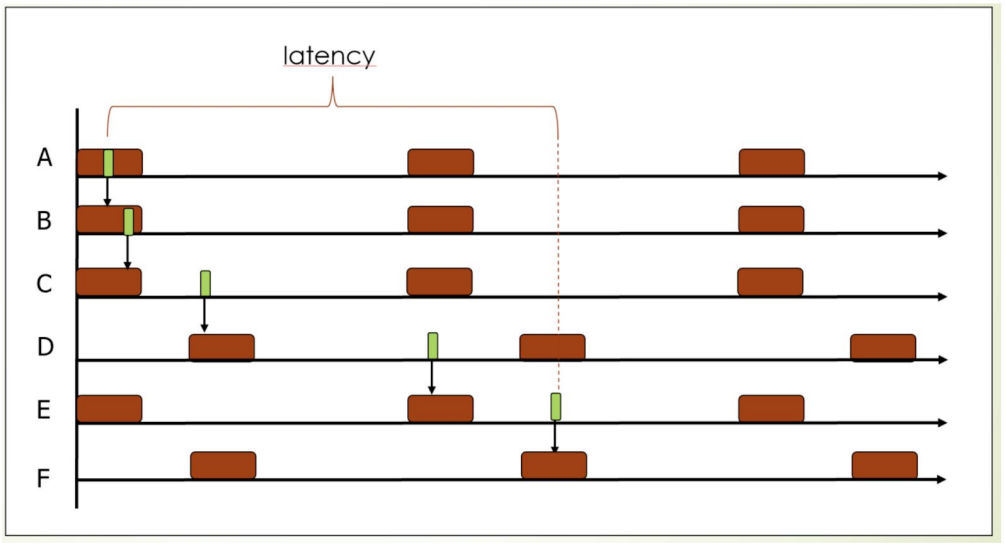
Per avere una maggiore efficienza energetica la soluzione è ridurre il duty cycle

Possibile soluzione per ridurre il duty cycle: possiamo aggregare i nostri dati e poi inviamo i dati tutti insieme e non uno alla volta. Anche se teniamo fissa la velocità del processore e della radio quello che fa la differenza è il modo in cui scriviamo il codice.

È importante notare che quello che la trasmissione consuma molto, l'obiettivo quindi è tenere spenta la radio il più a lungo possibile in modo da ridurre al minimo i consumi.
Lo stesso discorso vale anche per il processore, solamente che per spegnere il processore prendiamo una decisione localmente nel dispositivo perchè sappiamo come si comporta il dispositivo e cosa fa

mentre lo spegnimento della radio dipende anche dagli altri dispositivi nella rete perchè se spegnamo la radio vuol dire escludere il sensore dalla rete e la rete rischia di essere distrutta. Anche l’accensione e lo spegnimento della radio comunque è un qualcosa che consuma energia.

12) S-MAC slides-04C WSN - 16.9_DaDirectedDiffusion.pdf



Nelle Wireless Sensor network abbiamo una serie di sensori che sono wireless, autonomi e creano una rete senza avere la necessità di una comunicazione con un dispositivo centralizzato. Nella rete avremo uno o più nodi sink che interagiscono con il mondo esterno e vari sensori. I sensori possono produrre degli stream di dati e questi possono essere processati localmente e poi inoltrati al sink.

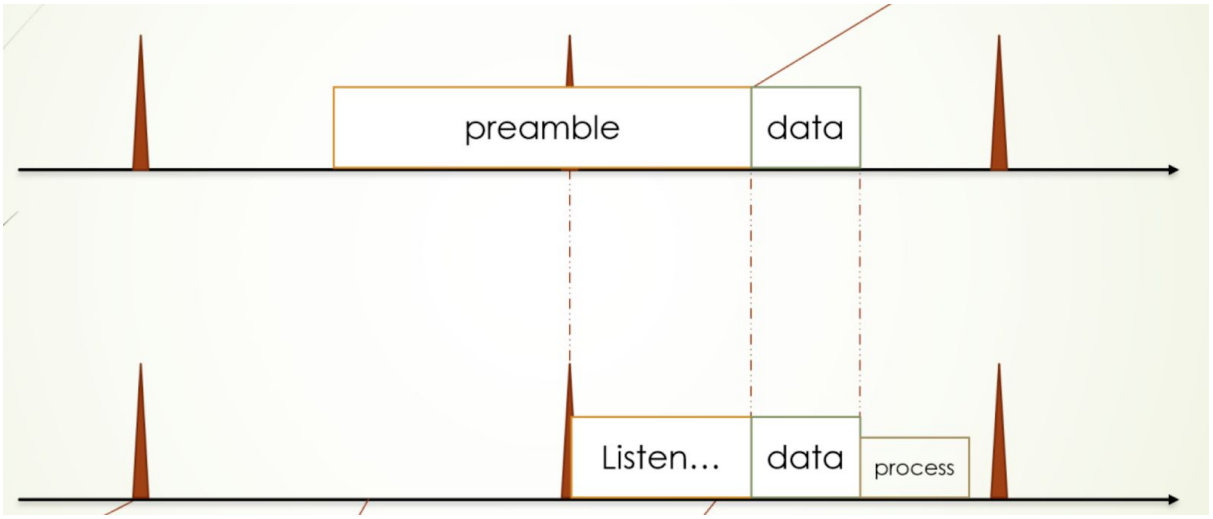
S-MAC è uno dei protocolli che possiamo utilizzare in una sensor network per l'accesso al mezzo e anche per gestire la sincronizzazione dei vari nodi in modo tale che nodi che sono vicini tra loro sincronizzano il loro listen period utilizzando un pacchetto chiamato SYNC.

L'idea è che quando un nodo entra nella rete parla con i suoi vicini e vengono scambiati i listen period in modo che i due nodi vengano attivati tutti nello stesso momento. Ad esempio abbiamo che A, B e C nella foto sotto sono sincronizzati e quindi possono scambiarsi dei dati nel momento in cui sono in ascolto. Un sensore riceve pacchetti dai vicini solamente durante il suo listen period.

Quello che vediamo nell'immagine in particolare è uno dei problemi che possiamo avere quando il nodo A vuole inviare un pacchetto al nodo F e in particolare in questo caso abbiamo che i due nodi possono comunicare solamente tramite Multi Hop.
Possiamo vedere che il pacchetto passa da A a B e poi da B a C nello stesso listen period poi abbiamo una prima attesa del nodo C che è in listen mode in un momento diverso rispetto al nodo D, quindi C va in listen mode appositamente per inviare il pacchetto a D. Ogni volta che abbiamo un invio dei pacchetti dobbiamo anche controllare che il mezzo sia libero per poter inviare il pacchetto all'altro nodo. Poi allo stesso modo il nodo D deve attendere il lister period di E e la stessa cosa la fa poi E con il nodo F. In questo modo abbiamo avuto una latenza che è cresciuta a causa dell'utilizzo del multi hop e della non sincronizzazione dei listen period. In generale si spera che la rete converga verso una sincronizzazione ma questo non avviene sempre, in alcuni casi potrebbe essere impossibile per un sensore avere un listen period compatibile con i vicini.

- Come si risolve il problema della latenza in S-MAC? E’ possibile che tutti i nodi riescano ad avere tutti la stessa sinronizzazione? Problemi con la sincronizzazione del clock perchè ci stanno i due gruppi che non convergono alla stessa organizzazione della rete e inoltre ci sono problemi con il clock perchè comunque dopo che si sono sincronizzati succede che questa sincronizzazione del clock non viene mantenuta (il clock dipende anche dalla temperatura ad esempio)

13) BMAC slides-04C WSN - 16.9_DaDirectedDiffusion.pdf



B-Mac è un altro protocollo di controllo dell'accesso al mezzo nelle sensor networks che però non sfrutta la sincronizzazione diretta tra i vari sensori presenti all'interno della rete. Con B-Mac infatti il nodo che vuole inviare dei dati ad un altro invia prima un preambolo, gli altri nodi che gli sono vicini effettuano ogni tanto un preamble sample per cercare di capire se c'è un preambolo da leggere. L'idea quindi è di spendere di più per la trasmissione dei dati che per la ricezione, il sampling al contrario sarà veloce e poco costoso. In questo caso abbiamo il nodo nella prima riga che invia il preambolo e poi il nodo nella seconda riga che esegue un sampling, prima sente il preambolo, poi riceve i dati e poi li processa.

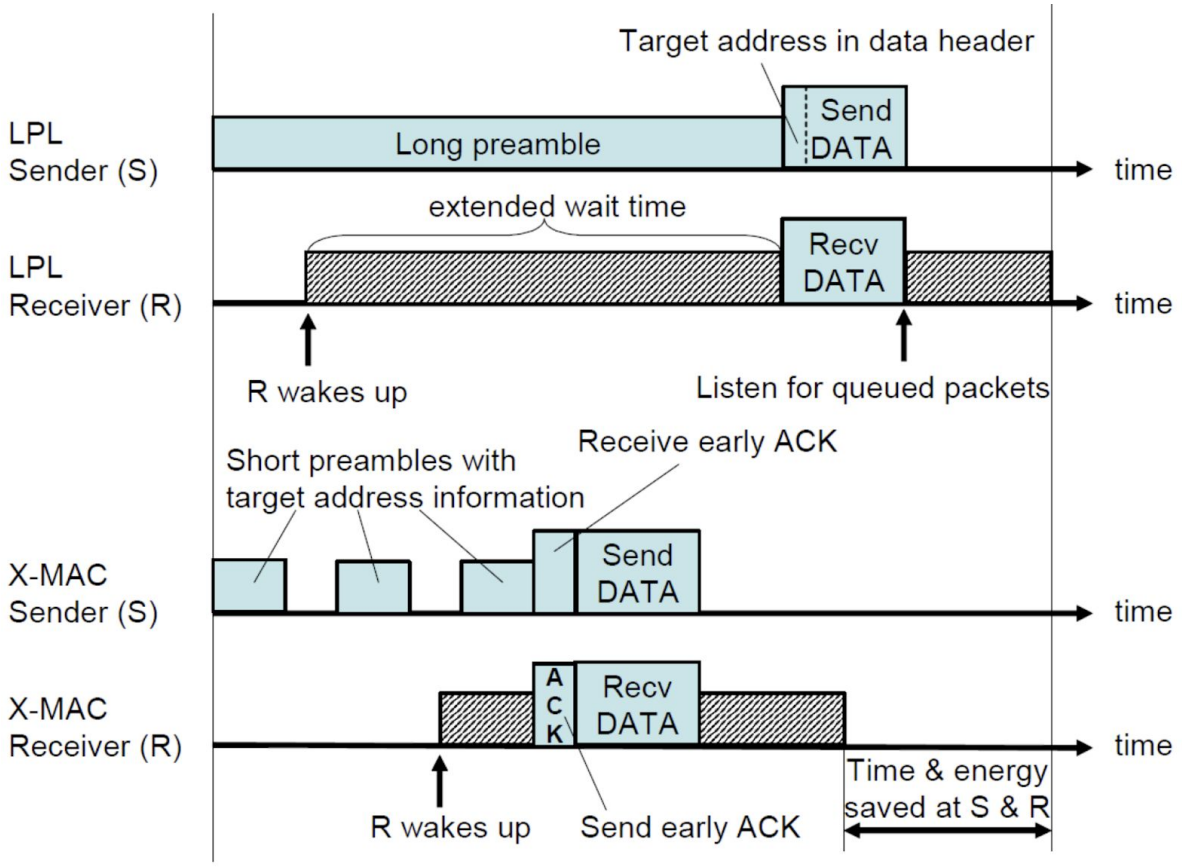
I pro di questo metodo è che non è un sistema in cui la rete si organizza e si sincronizza come ad esempio avviene in S-Mac ed è un protocollo più semplice da configurare e trasparente ai livelli superiori. Ci sono dei casi però in cui utilizzare questa forma di accesso al mezzo potrebbe non essere fattibile e potrebbe quindi diventare più costosa dell'usare una forma di sincronizzazione.

Bmac rispetto a S-mac è un problema se ho multi hop perchè ogni volta ascolti tutti il preambolo.

Domanda: C'è da considerare un trade off nella scelta della lunghezza del preambolo?
Si, si deve considerare un tradeoff per le performances dei dispositivi, se scelgo un preambolo troppo lungo quando il dispositivo che ascolta lo legge deve leggere troppo a lungo, se scelgo un preambolo troppo corto invece fa il sampling di continuo e quindi attiva e disattiva il sensore di continuo e quindi consuma troppo.
La lunghezza minima del preambolo è un parametro che dipende dalla distanza tra due sampling del sensore.

Perchè hanno deciso di far lavorare più il mittente dei destinatari?
Perchè questa soluzione è pensata per il multi hop e in questo modo se faccio 10 hop lavora più il dispositivo che invia mentre gli altri 10 ascoltano e basta.

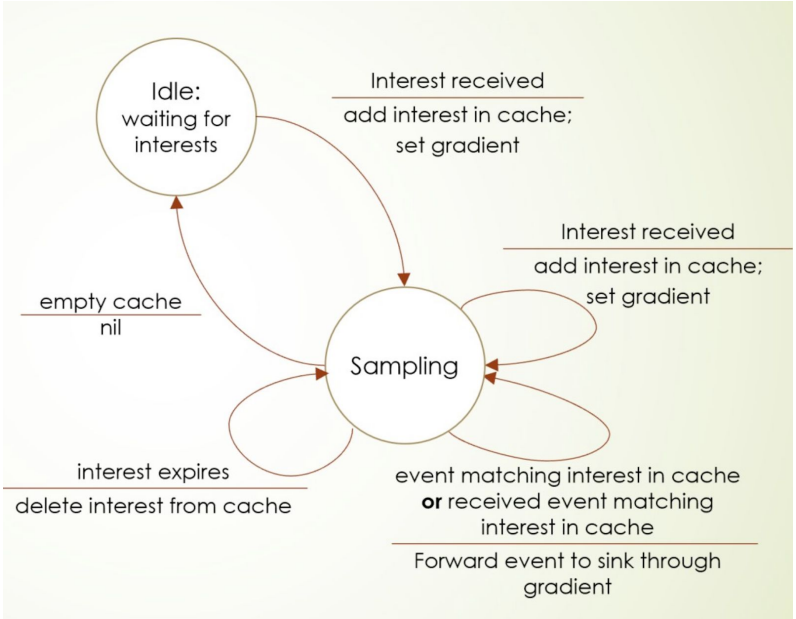
14) X-MAC/B-MAC: slides-04C WSN - 16.9_DaDirectedDiffusion.pdf



Altro algoritmo per l'accesso al mezzo nelle WSN, si tratta di una alternativa a B-Mac. Il problema che abbiamo con B-Mac è che il preambolo che viene inviato dal mittente può essere molto lungo e quindi il ricevente si trova a dover ascoltare il preambolo per un tempo molto lungo sprecando energia e tenendo la radio accesa, in B-Mac poi l'indirizzo del target si trova nel pacchetto con i dati che vengono inviati e non nel preambolo.

Con X-MAC invece il mittente invia non un singolo preambolo lungo ma tanti preamboli brevi al cui interno è inserito anche l'informazione riguardo all'indirizzo del target. Il target che riceve il preambolo vede se l'indirizzo che è nel preambolo contiene il suo indirizzo e in tal caso risponde al mittente con un ACK che quindi blocca l'invio del preambolo e avvia l'invio dei dati destinati a quel destinatario.

15) Directed Diffusion: slides-04C WSN - 16.9_DaDirectedDiffusion.pdf



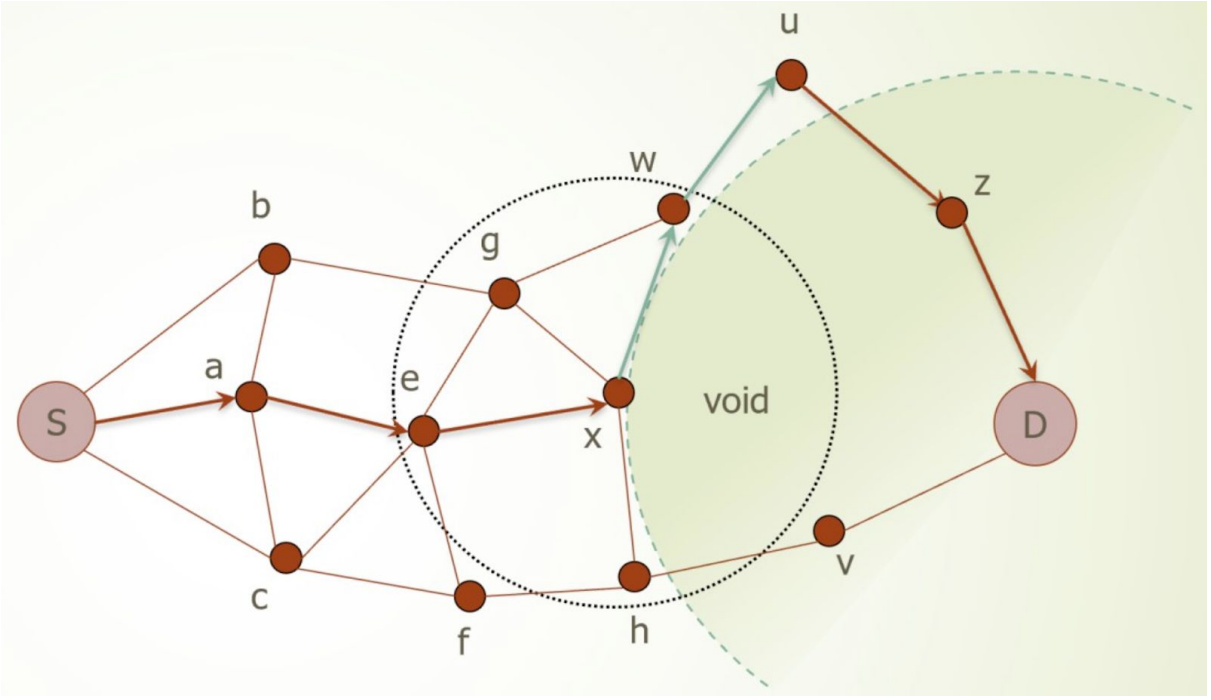
Il modello per la comunicazione nelle WSN non segue sempre l'idea della comunicazione con sender/receiver che solitamente abbiamo in internet. Ci sono varie applicazioni in cui questo modello non può funzionare, ad esempio possiamo pensare ad una applicazione in cui un nodo vuole sapere delle informazioni da un altro nodo che si trova in un certo punto della rete dove sta avvenendo un evento interessante (la temperatura è maggiore di un certo threshold). Il richiedente non deve necessariamente conoscere quale è il nodo in quella parte della rete e quindi non deve conoscere il suo IP perchè non vuole contattare nello specifico quel nodo. Al contrario invece il richiedente cerca i dati da tutti i nodi della rete che possono fornirmi quel tipo di dati, allo stesso modo i dati collezionati potranno essere inviati non solo ad uno specifico nodo ma a tutti i nodi della rete, quindi quello di cui abbiamo bisogno è un meccanismo per richiedere alla rete un dato e anche per inviare alla rete una certa risposta.

Quindi qua siamo in un contesto data centric routing e il directed diffusion è un metodo che viene utilizzato in uno scenario di distributed sensing per il routing delle query e per il routing delle risposte alle query.
I sensori sono programmati per rispondere a delle query. Il metodo funziona in questo modo e lo possiamo vedere con una macchina a stati:

- Il nodo sink che è interessato ad una certa informazione deve inviare un interest in broadcast verso gli altri nodi della rete che a loro volta lo inoltreranno quando lo riceveranno.
- Quando il nodo che è in alto riceve l'interest aggiunge l'interest all'interno della sua cache in modo tale da mantenere per un certo periodo di tempo in memoria questa informazione. Inoltre memorizza anche un gradiente che viene utilizzato per capire chi è stato il nodo che ha inviato l'interest. Questo gradiente lo utilizziamo poi quando dobbiamo mandare indietro la risposta al nodo che ha inviato l'interest.

- Una volta che ho ricevuto almeno un interest il nodo inizia il sampling e inizia a controllare se i dati che registra matchano con le richieste dell'interest. Se trova un match allora deve inviare l'informazione al nodo sink e per capire dove inviarla utilizza il gradiente che aveva calcolato in precedenza. Se invece non matcha l'interest continua. Se scade l'interest lo elimina dalla cache e se nella cache non rimane nulla torna nello stato IDLE in attesa che arrivi un nuovo interest.

16) GPSR slides-04C WSN - 16.9_DaDirectedDiffusion.pdf



4 assunzioni:

- I nodi sanno la loro posizione
- I nodi conoscono la posizione dei vicini
- I nodi si trovano in uno spazio 2
- Il nodo source conosce la posizione del nodo destinatario e nel pacchetto che spediamo sono contenute le coordinate del destinatario.

Questo è un protocollo stateless, non abbiamo una memorizzazione delle routing table, non avendo le routing table va da S a D. Ci sono pochi pacchetti per il controllo e i nodi mantengono localmente le loro informazioni, non abbiamo neanche di avere una cache molto grande nei nodi o delle routing table.

Si tratta di un altro protocollo di Routing che cerca di risolvere le limitazioni di Direct Diffusion.
GPSR funziona in due modalità, Greedy Forwarding e Perimeter Forwarding.
Supponiamo di essere nel nodo S e di voler inviare un messaggio al nodo D, quello che succede è che il nodo S che deve inviare il pacchetto considera i suoi vicini e sceglie quello che è più vicino a D e gli invia il pacchetto, in questo caso utilizziamo la Greedy Mode, se però nessuno dei suoi vicini è più vicino di S al nodo destinazione allora dobbiamo passare alla Perimeter Mode.
Ad esempio arrivati al nodo X vediamo che nessuno dei vicini di X si trova più vicino a D rispetto a X quindi passiamo in perimeter mode e utilizzando la regola della mano destra (o sinistra basta che si usa sempre quella) si inoltra il pacchetto ad uno dei vicini di X, in questo caso W che farà lo stesso ragionamento di X.

Quando siamo in perimeter mode quello che succede è che attraversiamo la faccia che contiene la nostra destinazione.

Il ritorno dalla perimeter mode alla Greedy Mode avviene quando il nodo a cui è connesso quello che stiamo considerando è più vicino alla destinazione rispetto al nodo in cui è iniziata la perimeter mode, ad esempio in questo caso in U passiamo di nuovo in greedy mode perchè il nodo Z è più vicino a D rispetto al nodo X.

Se il nodo che vorrei raggiungere non è raggiungibile allora me ne accorgo nel nodo X che è quello in cui ho iniziato la perimeter mode.

Per capire se va in loop usa le facce.

Ogni volta che entra in una nuova faccia salviamo nel pacchetto il punto di intersezione

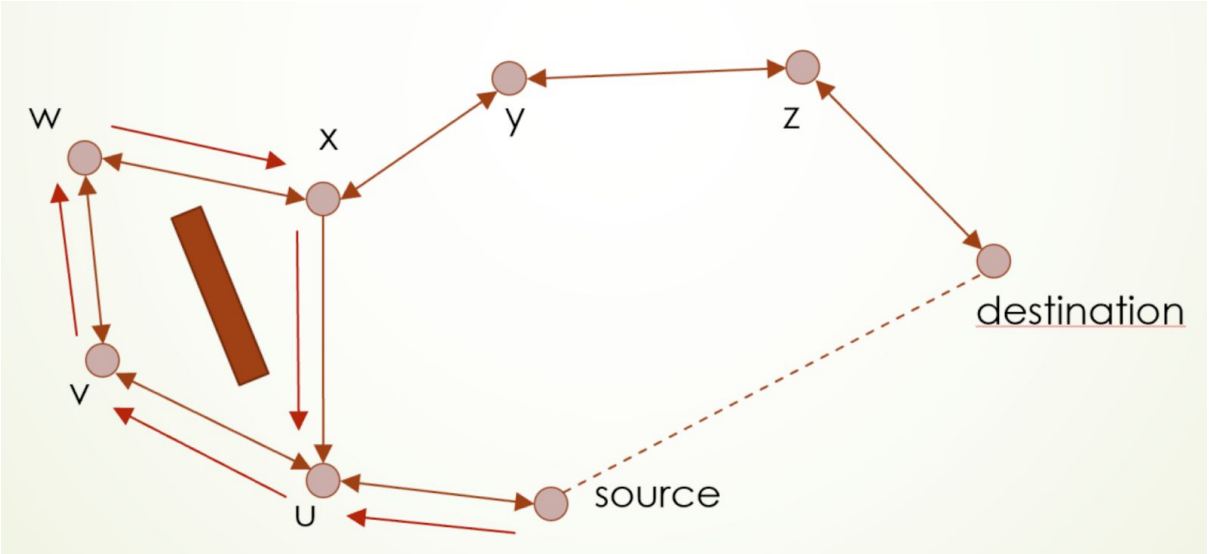
È un sistema stateless a differenza dei protocolli classici di routing.

Problemi:

- Il grafo potrebbe non essere un grafo planare ovvero alcuni archi potrebbero sovrapporsi. Possiamo fare la planarizzazione usando Relative Neighborhood o Gabriel Graph.

Quando un grafo è planare allora contiene delle facce, quelle interne sono quelle delimitate dai nodi, poi abbiamo una faccia esterna che è quella esterna ai nodi.
Nel caso in cui siamo in perimeter mode: abbiamo una retta che passa da X e da D e quando ci spostiamo da X a D passiamo sulla faccia e poi quando arriviamo al punto di intersezione della faccia con la retta allora passiamo in un'altra faccia, in questo momento memorizziamo il punto di intersezione e il l'arco corrente per fare in modo che possiamo sempre capire quale è il nodo in cui abbiamo iniziato la perimeter mode.
Quando

17) GPSR slides-04C WSN - 16.9_DaDirectedDiffusion.pdf



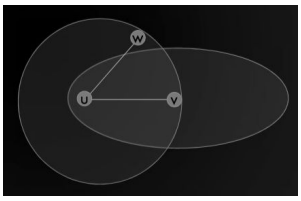
Quando utilizziamo GPSR possiamo trovarci ad avere dei problemi quando il grafo che stiamo considerando non è planare ovvero ci sono degli archi che si sovrappongono. In questi casi si devono utilizzare degli algoritmi di planarization come il Relative Neighborhood Graph e il Gabriel Graph. Quando utilizziamo questi algoritmi quello che succede è che alcuni archi vengono rimossi in modo da non avere delle sovrapposizioni.

Supponiamo di avere due nodi U e V, generalmente quello che si fa è tracciare due cerchi, uno centrato nel nodo U con raggio U-V e uno in V con lo stesso raggio. Se usiamo il Relative Neighborhood controlliamo se nell'intersezione dei due cerchi abbiamo nodi connessi con U e V oppure no, se ad esempio abbiamo un nodo tale per cui vale la formula della distanza d(u,v) <= max(d(u,w), d(w,v)) allora l'arco rimane come è. Se invece usiamo il Gabriel Graph vale la stessa idea ma la formula diventa d(u,v) <= d(u,w)^2 + d(w,v)^2 e qua in pratica vediamo se nel cerchio con diametro U-V abbiamo nodi oppure no.

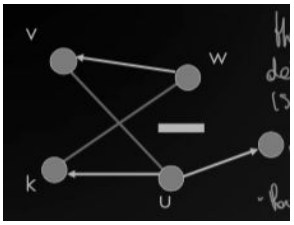
Quello che succede in alcuni casi è che in alcuni casi potremmo avere un fallimento della planarizzazione. Ad esempio in questo caso abbiamo che X non vede la V per un ostacolo mentre la U vede la V quindi quello che succede è che U rimuove il collegamento verso X mentre invece X lo lascia e si crea quindi un arco unidirezionale.

Quindi in pratica quando arriva ad U un pacchetto da inoltrare verso la destinazione, l'unica strada consiste nel passare da V e poi da W e da X. Quando poi il pacchetto arriva a X però ci troviamo, in base alla regola della mano destra, a dover inoltrare il pacchetto al nodo U e quindi si è creato un loop.
Come risolviamo il problema? Si risolve con la mutual witnesses che aggiunge un ulteriore controllo prima di rimuovere l'arco. Consideriamo infatti l'arco U-X, se non esiste un collegamento tra V e X allora l'arco U-X rimane così come è mentre se esiste anche l'arco V-X allora viene rimosso. Anche la Mutual Witness in alcuni casi però può creare dei problemi e potrebbe portarci ad avere comunque dei grafi planari.

Nelle regole per l’eliminazione dell’arco abbiamo considerato sempre un cerchio unitario ma questa è una assunzione e non è detto che si possa avere un cerchio unitario attorno al nodo.

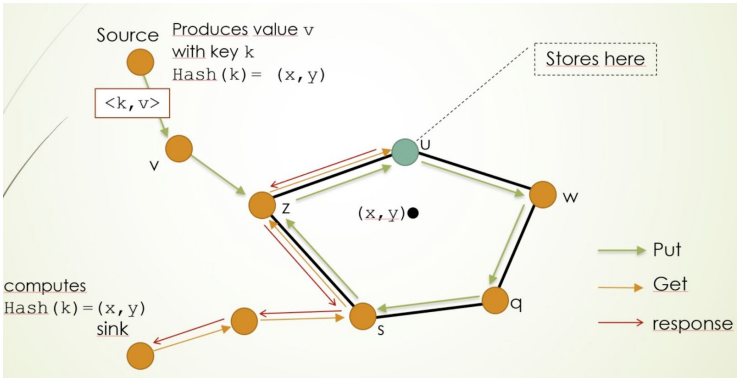


Ci sono situazioni in cui anche se usiamo la mutual witness non riusciamo ad eliminare la sovrapposizione dei nodi, qua in pratica succede che U-V si sovrappone con W-K e si dovrebbe rimuovere uno dei due, U vede che è connesso con V, V vede che è connesso con W ma W non è connesso con U e quindi non si rimuove l’arco, lo stesso discorso vale anche per l’altro arco.



Soluzione definitiva: usare CLDP con i probe, mandiamo in giro questi messaggi scrivendo dove è passato il nodo e se vedo di nuovo lo stesso messaggio vuol dire che c’è un loop e posso rimuovere il nodo in cui sono passato più volte. Questo potrebbe portare ad un cambio della topologia della rete e anche ad una disconnessione inoltre abbiamo anche problemi per quanto riguarda il fatto che inviamo dei messaggi in più nella rete per eseguire la planarizzazione. In teoria dovrebbe funzionare ma nella pratica è molto complesso e alcuni link potrebbero non funzionare più dopo la planarizzazione.

18) DCS-GHT: slides-04C WSN - 16.9_DaDirectedDiffusion.pdf



Abbiamo tre possibilità per memorizzare i dati all'interno della rete:

- Un solo nodo che memorizza i dati, storage centralizzato e i nodi inoltrano i loro dati a questo nodo
- I dati vengono memorizzati all'interno dei sensori stessi che producono il dato ma ogni volta poi dobbiamo fare una ricerca per capire dove sono i dati.
- Data Centric storage che usano la geographic hash table.

Il data Centric Storage è stata la prima proposta per organizzare le reti come data centric storage ed è un protocollo di routing che cerca di risolvere i problemi dei protocolli di routing precedenti e in particolare riesce a funzionare anche quando il sink non è disponibile.
In DCS abbiamo due opzioni, Put e Get.

- Abbiamo il nodo fonte produce una coppia chiave valore k,v e calcola l'hash di questa coppia andando a calcolare il punto x,y all'interno della rete. Una volta fatto questo si utilizza GPSR per andare a trovare il nodo più vicino al punto x,y e si memorizza in questo nodo, che in questo caso è il nodo U la coppia <k,v>. Questa è l'operazione PUT, la chiave che utilizziamo è importante perché ci permette di memorizzare gli eventi relativi a quella chiave in uno specifico nodo.
  Quando facciamo la PUT c'è una cosa che possiamo considerare, noi vorremmo avere delle proprietà con questo DCS, vorremmo avere un sistema scalabile, tollerante ai fallimenti, consistente e soprattutto vorremmo avere un sistema persistente nel senso che se cerco una certa chiave, i risultati associati dovranno essere disponibili nella rete indipendentemente dal fallimento dei nodi. Quindi quello che succede è che invece di memorizzare i dati solamente nel nodo U che è il più vicino al punto x,y andiamo a memorizzare i dati in tutti i punti che sono attorno al punto x,y in modo che se U fallisce ce ne sarà un altro in cui il dato è comunque memorizzato. I nodi differenti da U sono detti nodi replica.
- Quando poi il sink vuole fare la GET del dato identificato con la chiave K calcola l'hash di questa chiave K e trova il punto x,y. Allo stesso modo utilizzerà GPSR per andare a trovare il nodo U e gli invierà una richiesta per la coppia <k,v>. Il nodo U poi risponde al sink inviando la coppia <k,v>.

Per garantire la consistenza e anche il mantenimento del perimetro che si viene a creare attorno al punto x,y viene utilizzato un sistema di refresh del perimetro. Il nodo U che è l'home node per

questo perimetro invia al punto x,y il valore della coppia <k,v> e un pacchetto di refresh a intervalli regolari per fare in modo che i nodi sul perimetro abbiamo sempre la versione più aggiornata e per fare in modo di capire se ci sono stati cambiamenti nella topologia della rete, ad esempio potrebbe esserci un nuovo nodo più vicino a x,y che quindi diventa il nodo home oppure il nodo home corrente potrebbe essersi spostato e quindi bisogna trovarne uno nuovo. Notare che è importante che il nodo U invii il pacchetto di refresh al punto x,y perchè lui potrebbe essersi spostato

Questo meccanismo è sempre migliore di Direct Diffusion?

In questo caso dato che stiamo usando le hash table non sappiamo dove saranno mandati i dati e quindi potrebbe capitare che alcuni punti della rete potrebbero essere pieni di dati e quindi i nodi potrebbero essere pieni anche di richieste, in questo caso specifico potrebbe essere un problema.

In direct diffusion abbiamo un modello in cui non memorizziamo i dati all'interno dei nodi ma inviamo tutto quanto al sink.

Ci sono delle situazioni in cui potrebbe essere meglio Directed Diffusion perchè abbiamo che i dati nel caso di DCS vengono prodotti in un nodo, poi vengono spostati nel nodo più vicino al punto x,y e poi se dobbiamo recuperarli dobbiamo spostarci di nuovo all'interno della rete, questo in directed diffusion non succede perchè abbiamo solamente uno spostamento di dati all'interno della rete dal nodo che ha prodotto l'informazione al sink.

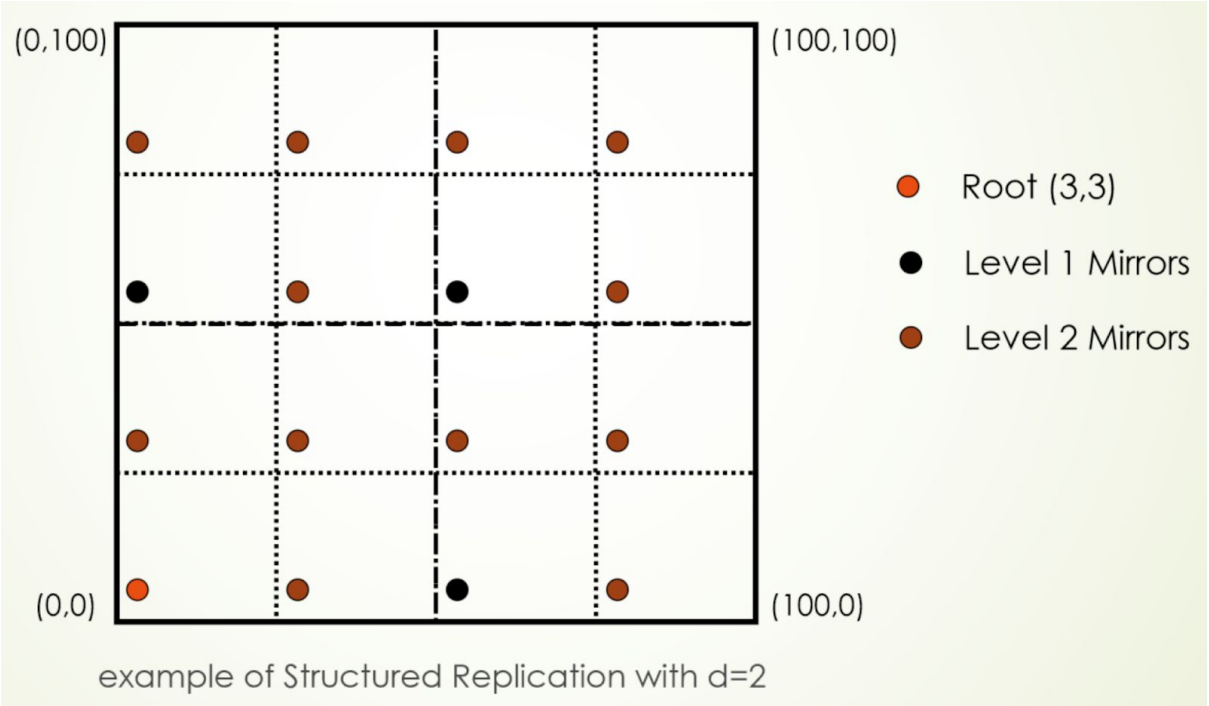
Una situazione in cui DCS potrebbe essere meglio di Directed Diffusion è quella in cui non necessitiamo dei dati live ma solamente una volta ogni tanto, quindi non ci serve che i nodi che producono l'informazione poi la inoltrino immediatamente ma possono memorizzarla e aggregarla e poi inviarla quando è necessario.

Anche in directed diffusion ci sono dei punti della rete che sono più carichi di altri, più o meno possono essere ovunque, sia vicino al sink ma anche nei punti della rete in cui magari vengono prodotti più dati.

In DCS abbiamo dei problemi con il load balancing, quale è il caso in cui il problema è più drammatico? In alcuni casi possiamo usare il mirroring e la replication ma non sempre questa soluzione può essere una soluzione. Quale è il caso più estremo?

- Il perimetro che troviamo attorno ad un certo punto x,y può essere molto grande e quindi la replicazione potrebbe essere molto costosa e quindi stiamo replicando i nostri dati in tantissimi nodi andando anche a inviare molti dati all'interno della rete per svolgere la replicazione.
- Il caso peggiore è quando il punto che selezioniamo per la memorizzazione dei dati è all'esterno del confine della rete perchè in quel caso il perimetro corrisponde all'intero perimetro.

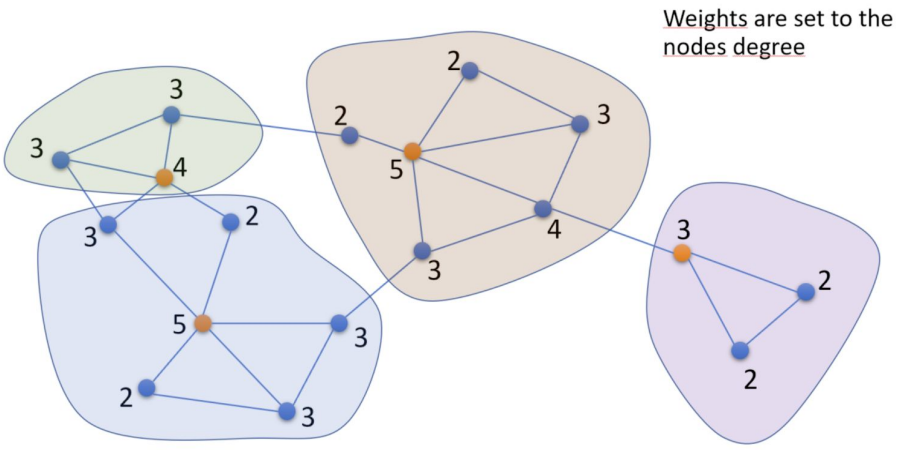
19) DCS-GHT: slides-04C WSN - 16.9_DaDirectedDiffusion.pdf



Il problema che si pone quando utilizziamo DCS-GHT è un problema di sovraccarico dell'home node. Supponiamo di avere un certo home node che nell'immagine è il nodo arancione in basso a sinistra. Se noi dobbiamo memorizzare la coppia k,v e troviamo che questo è il nodo più vicino al punto x,y salviamo tutto qua. Il problema si verifica se abbiamo tante coppie k,v da memorizzare e se vanno a finire tutte nell'home node.
Per risolvere un problema possiamo usare la structured replication, quello che facciamo è dividere l'area in vari quadrati e in ognuno di questi quadrati prendiamo un nodo mirror dell'home node. In questo modo riusciamo a suddividere il carico tra i vari nodi dei vari quadrati e non memorizziamo tutto in un unico home node.
Possiamo quindi fare una suddivisione ricorsiva in modo da ridurre il carico di dati memorizzati nell'home node. Quando si deve fare una GET, il sink che non conosce questa suddivisione invierà la query all'home node e poi l'home node inoltrerà la query al nodo mirror di primo livello che a sua volta potrebbe inoltrare al mirror di secondo livello.
I dati che vengono prodotti in un certo nodo vengono memorizzati in quel nodo e poi nei suoi nodi mirror, la quantità dei nodi mirror in cui memorizziamo dipende dalla profondità dell'albero che si viene a creare (in pratica da quante divisioni facciamo).

Per capirla bene c'è un paper, lo chiede.

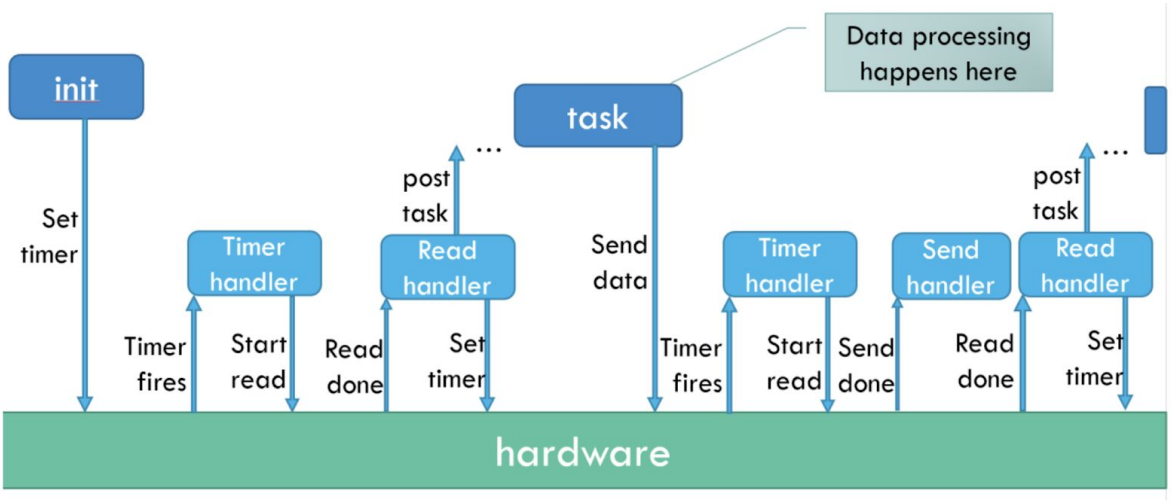
20) DMAC: slides-0D WSN clustering



DMAC è uno degli algoritmi di clustering che possiamo utilizzare all'interno della Wireless Sensor Network per la clusterizzazione.
Ciascuno dei nodi ha un ID, uno status e un peso, in particolare lo stato può essere CH (Cluster Head), ON (Ordinary Node) oppure UN (Undecided). Il peso invece dipende da vari fattori come ad esempio la quantità di batteria del nodo e il grado del nodo.

L'algoritmo di clustering funziona in questo modo, abbiamo i vari nodi che si scambiano un messaggio di hello (che viene inviato in broadcast) in cui è contenuto lo status del nodo, il suo ID e il suo peso. Quando il nodo riceve questo hello packet dai suoi vicini decide il suo status in base ad una decision rule. Ad esempio se tutti i pesi che riceve sono maggiori rispetto al suo allora sarà un ON, se invece il suo peso è il massimo sarà un CH.
Quando un nodo diventa un Cluster Head tutti i vicini vengono informati.
Un'altra regola importante del protocollo consiste nel non mettere vicini due CH, i CH devono essere sempre e comunque separati da un altro nodo.
La cosa positiva di questo protocollo è che vengono inviati O(n) messaggi e che riesce anche ad adattarsi ai cambiamenti della rete perché quando si aggiunge un nuovo nodo questo manda in broadcast l'hello packet in modo che i vicini sappiano della sua esistenza e poi gli viene assegnato uno status, può capitare che questo nodo diventi il nuovo cluster head e quindi quello che succede è che ci dovranno essere dei cambiamenti all'interno della rete perchè due cluster head non possono essere vicini tra loro. In generale la riconfigurazione viene effettuata in tempi fissati e può cambiare sia i ruoli sia i cluster. Non vogliamo che un cambiamento di questo genere modifichi tutta la rete perchè inserirebbe ulteriore overhead, ci sono però delle situazioni in cui a seconda di come sono posizionati i nodi avremo una chain reaction se il nuovo nodo che entra nella rete diventa subito CH, un esempio è il caso in cui abbiamo un alternarsi di CH e di ON. Tipicamente la probabilità di avere una chain reaction aumenta a mano a mano che aggiungiamo più nodi all'interno della rete, di norma i cambiamenti sono limitati ai nodi che sono vicini a quello che si aggiunge nella rete.

21) Embedded programming slides-04E_Embedding programming-Arduino-TinyOS



L'implementazione del Duty Cycle che abbiamo in Tiny OS è differente da quella in cui abbiamo con Arduino e vengono utilizzati dei timer. In Arduino infatti abbiamo solamente un loop e all'interno mettiamo il codice che deve essere eseguito, ogni X tempo, quando scatta un timer, quel codice viene eseguito in modo sequenziale fino alla fine, poi attendiamo di nuovo e lo eseguiamo ancora e così via.
In Arduino abbiamo un event loop mentre con TinyOS abbiamo una event based programming e i task.
In TinyOS non c'è il concetto di Event loop, abbiamo una gestione asincrona degli eventi, con TinyOS abbiamo i seguenti concetti:

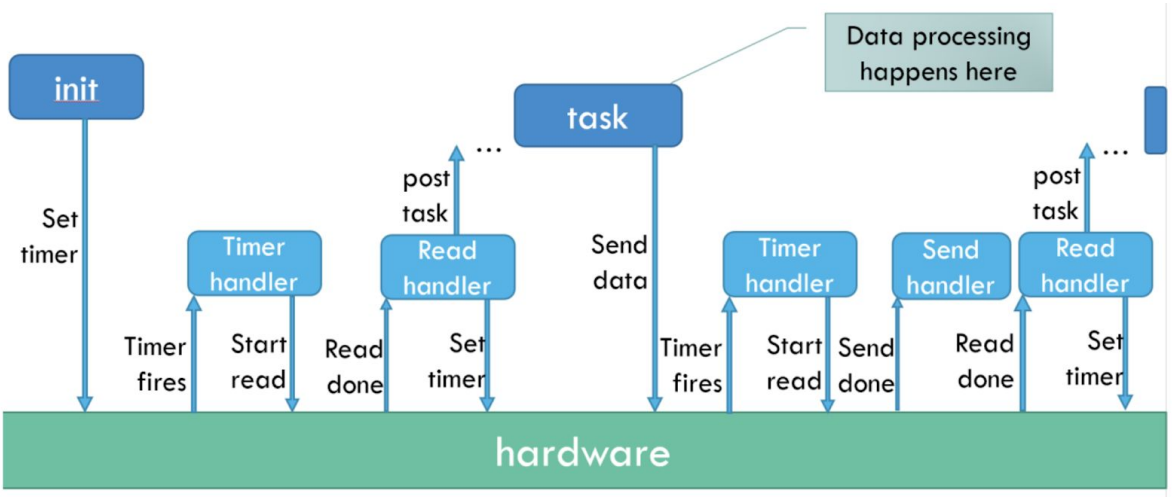
- Comandi: sono le funzioni offerte per programmare e attivare l'hardware.
- Eventi: sono una astrazione delle interruzioni, sono una sorta di upcall. Quando avviene un evento e riceviamo l'upcall il programmatore definisce un handler per quell'evento. Dobbiamo avere un handler per ogni evento.
- Tasks: sono i task che vogliamo eseguire in modo sequenziale, quando un task inizia poi viene portato a termine. Generalmente sono l'effetto di un evento, quando terminiamo di eseguire il task il processore si blocca. Il task può essere interrotto solamente da una interruzione che deve essere gestita dall'event handler e a sua volta dovrà completare tutto il lavoro (non riceve altre interruzioni e il task dell'interruzione deve essere semplice).

In particolare nell'immagine vediamo il funzionamento dell'esecuzione di TinyOS, per prima cosa abbiamo una funzione init che viene eseguita e che è scritta dal programmatore, questa setta i timer che andremo poi ad utilizzare, quando scatta il primo timer lo gestiamo e in questo caso accediamo all'hardware leggendo i dati che ci interessano, la fine della lettura dall'hardware genera un evento che viene gestito e quello che leggo viene utilizzato in un task che eseguo. Il task permette di eseguire anche un codice più complesso, ad esempio una comunicazione, quindi può richiedere del tempo. Quando finisco il task invio i dati risultanti e poi ricomincia quello che avevamo fatto in precedenza, quindi attendiamo di nuovo che il scatti il timer per fare di nuovo la stessa cosa.

Quando scatta un timer quindi abbiamo un handler che ci permette di gestirlo, l'interruzione deve essere gestita subito quando arriva e l'event handler deve essere più veloce possibile, se invece

abbiamo bisogno di più lavoro da svolgere allora possiamo postare un task nella coda. Con questo approccio un task non attende mai.

Risposta data oggi:



All'inizio vengono settati dei timer, quando questi eventi scattano allora viene generato un evento, alla generazione dell'evento viene eseguito un handler che viene implementato dal programmatore. L'evento oltre a gestirlo con un semplice handler possiamo anche gestirlo con un task, il task viene utilizzato per processare dati più complessi ma anche per sequenzializzare le operazioni. I task sono atomici tra di loro e sono gestiti tramite una coda.

Una caratteristica è avere le operazioni split phase, che significa? Qua ci sono degli esempi di operazioni split phase?

Con questo schema in questo caso non memorizziamo lo stato del thread, quando l'handler termina termina, non abbiamo da memorizzare niente, quando parte parte senza partire da uno stato pre esistente. Il motivo per cui lo schema è intricato dipende da questo fatto.

Domanda:

Con la split Phase noi abbiamo che ogni event handler non mantiene il suo stato, quindi nasce quando scatta il timer e poi muore quando viene dato all'hardware un comando? Ogni volta che l'hardware solleva un evento poi abbiamo che l'evento viene gestito ma chi lo gestisce non ha niente a che vedere con chi ha gestito l'evento precedente perchè non ho uno stato???

Differenza task vs event handler: difficoltà del codice da eseguire? Atomicità?

- Un comando attiva l'hardware, ad esempio per leggere da un sensore
- Un evento ci indica che una certa operazione è terminata

Arduino:

- Abbiamo un singolo thread che segue il loop, se devo fare una operazione di I/O attendo che venga completata e nessun altro thread viene eseguito nel frattempo.
- Se vogliamo eseguire un comando più tardi allora dobbiamo usare un comando esplicito.

Split Phase:

È un modo per implementare una comunicazione asincrona tra componenti, è usato per eseguire operazioni che hanno una lunga durata o che sono bloccanti, dividiamo l'operazione in due fasi:

- Invochiamo il comando, ad esempio una lettura
- Riceviamo il risultato dell'operazione tramite un evento

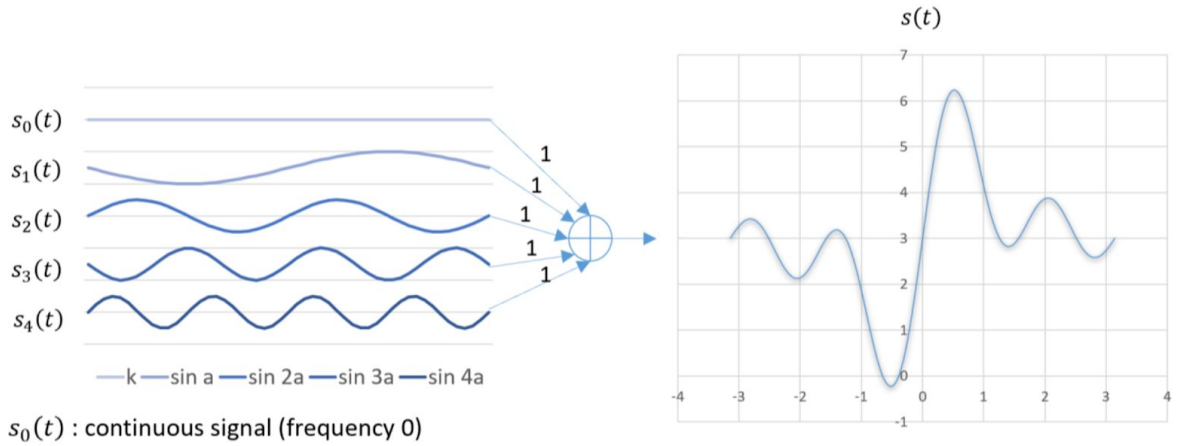
Tramite questo meccanismo il sistema non viene bloccato.
Gli eventi sono asincroni, e hanno una maggiore priorità rispetto ai task, un task può essere fermato per gestire un evento ma non può essere fermato per gestire un altro task.

Domanda:

- Cosa fa nell'init?

21) Fourier series: slides-06B_Signal theory - Fourier series

$$\frac{1}{2}a_0 + \sum_{n=1}^{\infty}(a_n \cos nt + b_n \sin nt)$$



$s_0(t)$: continuous signal (frequency 0)
$s_1(t)$: fundamental (frequency $f_0 = 1/T$)
$s_2(t)$: first harmonic (frequency $f_1 = 2/T = 2f_0$)
$s_3(t)$: second harmonic (frequency $f_2 = 3/T = 3f_0$)
$s_4(t)$: third harmonic (frequency $f_3 = 4/T = 4f_0$)

Nell'immagine abbiamo vari segnali che vengono sommati per ottenere un unico segnale. notare che i segnali sulla sinistra sono tutti ottenuti moltiplicando il primo per una costante. La cosa che possiamo notare quindi è che dato il set di segnali iniziali si può passare ad un unico segnale finale che otteniamo come somma degli altri, è possibile anche fare l'inverso e passare dal segnale somma ai singoli segnali?

Questo è quello che si cerca di fare con le Serie di Fourier che permettono di scomporre un segnale come somma di un numero infinito di funzioni continue che oscillano a frequenze differenti (scomponiamo in una serie di frequenze che producono quel segnale).
Il set di funzioni continue che vengono utilizzate sono chiamate base della decomposizione e devono essere ortogonali. In particolare la formula sopra è la formula del polinomio trigonometrico che è la formula della serie di Fourier consideriamo il segnale continuo s(t) periodico in un intervallo. All'interno della definizione abbiamo il primo termine che è costante, mentre poi abbiamo an e bn che rappresentano l'ampiezza dell'armonica.
Ognuna delle funzioni che andiamo a generare oscilla in modo differente. Questo polinomio è una combinazione lineare finita di funzioni seno e coseno con n>0.

Quali sono le condizioni per utilizzare Fourier?

Se abbiamo un segnale che non è periodico e ha supporto limitato cosa usiamo?

Come abbiamo definito la trasformata di Fourier?

Una funzione che passa dal dominio del tempo al dominio delle frequenze.
Il segnale che sta a destra viene diviso in frequenze, già nel grafico si passa dal tempo alle

frequenze. La trasformata di Fourier è un passaggio dal dominio del tempo alle frequenze ma l'abbiamo definita utilizzando l'integrale e non una sommatoria, abbiamo usato l'integrale perchè abbiamo usato la trasformata continua di Fourier

Le frequenze sono tutte un multiplo della frequenza di base….

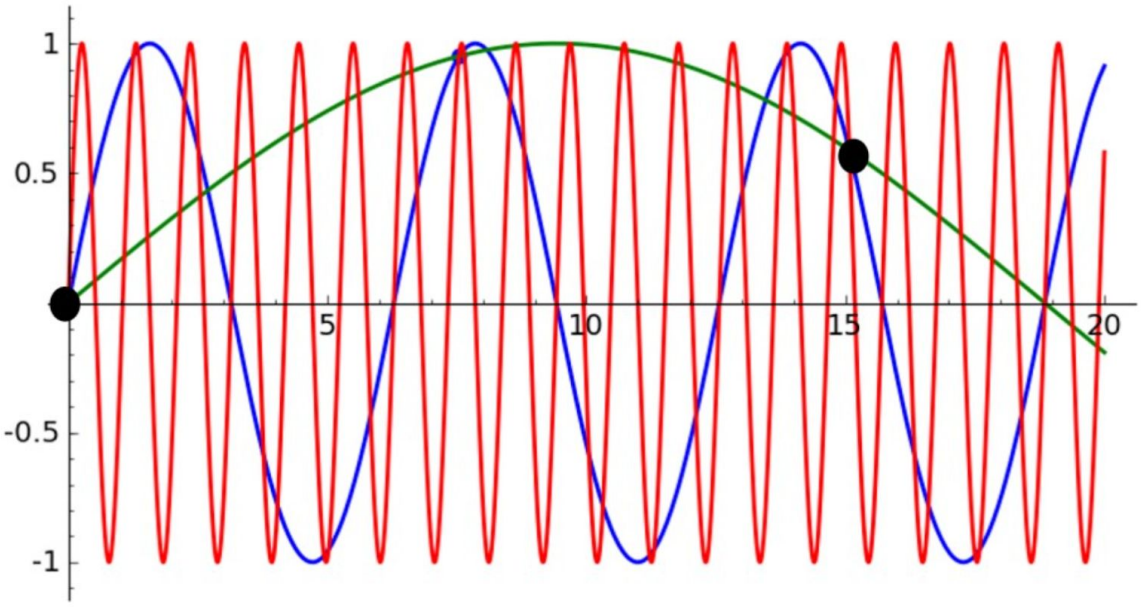
Quando due dispositivi vogliono comunicare utilizzano un canale che è di tipo digitale, all'interno di questo canale, se dobbiamo spedire un segnale analogico, c'è inizialmente un modem che fa la traduzione da segnale analogico a segnale digitale poi dalla parte opposta c'è anche un altro modem che fa la traduzione da segnale digitale ad analogico in modo da ottenere l'informazione che aveva il source.

Un segnale nel tempo viene visto come una funzione che può essere una combinazione di sinusoidi (seni e coseni), la prima parte della formula è costante, poi abbiamo seno e coseno che rappresentano le armoniche, an e bn che rappresentano le ampiezze delle armoniche. Al centro invece vediamo che è possibile con una combinazione lineare di seni e coseni ottenere il segnale oppure anche il viceversa ovvero il segnale lo decomponiamo utilizzando una combinazione lineare di seni e coseni.
Le armoniche a sinistra sono fatte così, il primo è un segnale a frequenza 0, il secondo ha la frequenza fondamentale e poi le altre sono tutti multipli di questa frequenza fondamentale. Per la decomposizione del segnale di sinistra in armoniche abbiamo dei requisiti per il segnale, il segnale deve essere periodico (ma poi il segnale possiamo anche estenderlo e possiamo renderlo periodico se è un segnale limitato in modo che rientra nella casistica dei segnali periodici e possiamo usare Fourier) e deve anche essere continuo a tratti.

Il risultato del calcolo della serie di Fourier con un segnale che è limitato nel tempo?
La serie di Fourier ha calcolato in realtà quella su tutto il periodo, quindi restituisce il segnale come se fosse periodico.

Se invece vogliamo la serie di Fourier di un segnale che è limitato nel tempo cosa si fa?
Non possiamo usare più la serie di Fourier ma usiamo la Trasformata di Fourier, che è sempre un passaggio da un dominio del tempo al dominio delle frequenze ma la trasformata di Fourier l'abbiamo definita non con una sommatoria ma con un integrale e l'abbiamo usato perchè abbiamo usato la trasformata continua di Fourier perchè era un segnale di energia che è un segnale limitato. Perchè siamo passati in una situazione in cui le frequenze non sono discretizzate ma sono continue. Nel caso dell'immagine le frequenze sono un multiplo della frequenza di base, qua viene bene perchè il segnale è periodico, se però non è periodico lo possiamo vedere come un segnale che ha un periodo infinito e in questo caso la frequenza di base tende a 0 e quindi tutti i suoi multipli sono tutte le possibili frequenze nel dominio dei reali ed è per questo che si passa all'integrale.

22) sampling and aliasing: slides-06B_Signal theory - Fourier series



Il problema dell'aliasing ce l'abbiamo quando lavoriamo con un segnale e facciamo il sampling andando a prendere solamente alcuni valori di questo segnale. Noi avremo una certa frequenza per il sampling e quindi andiamo a considerare il valore del segnale a tempi ben definiti, in questo caso ad esempio prendiamo i due punti neri.

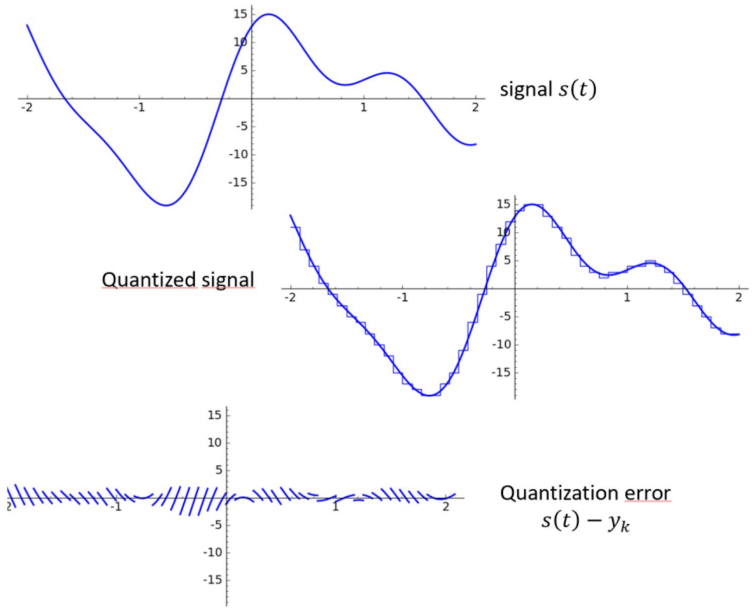
Il problema è che quando facciamo il sampling del segnale si ottengono una serie di punti, questa serie di punti però se poi la andiamo ad interpolare crea una serie infinita di armoniche che possono generare il segnale e quindi non ricostruiamo necessariamente il segnale originale perchè ne abbiamo tanti possibili.

Per ricostruire il segnale originale si deve fare il sampling ad almeno una certa frequenza che equivale al doppio della frequenza massima dello spettro della frequenza del segnale. Quindi se scomponiamo un segnale e vediamo che al massimo queste frequenze erano 10 allora dobbiamo fare il sampling a 20, in questo modo poi possiamo ricostruire il segnale originale.

DOMANDE:

- Cosa intende con spettro
- Cosa intende con banda
- Cosa intende con spettro nullo alle frequenze sopra fm?

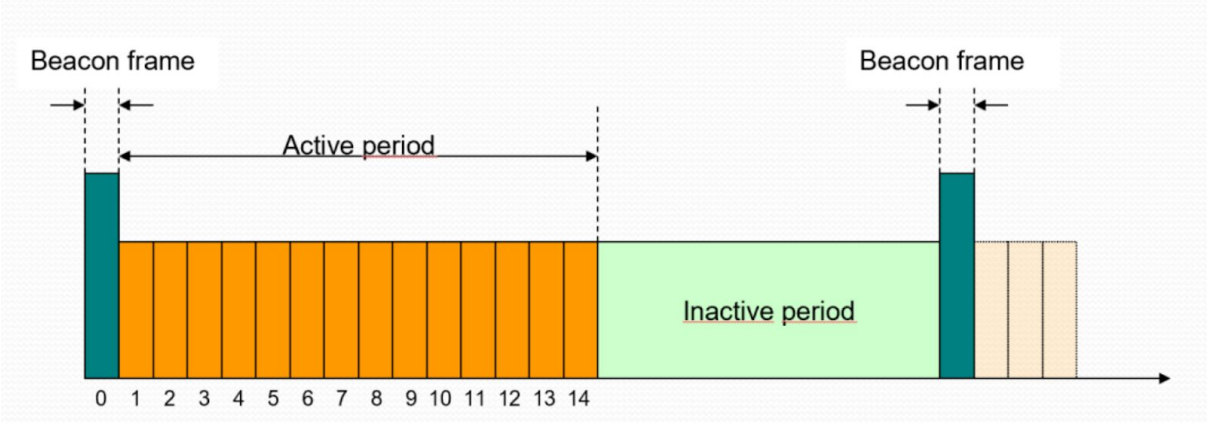
23) quantization: slides-06D_Signal theory - sampling and quantization - 16.9



Consideriamo un segnale s(t) e supponiamo di aver svolto il sampling, quello che otteniamo è una lista di valori (solitamente sono dei floating point) che sono presi dal segnale s(t) ad intervalli regolari. Ora che abbiamo questa lista dobbiamo trasformare i numeri presenti all'interno in dei numeri che possono essere utilizzati da una macchina, quindi vorremmo passare a numeri che possono assumere solamente un certo set di valori finiti.
Questa trasformazione della lista dei valori iniziali in una lista di valori finiti viene chiamata quantizzazione.
La quantizzazione consiste nell'approssimazione del valore del sample che abbiamo preso dal segnale s(t) con un intero che sarà contenuto all'interno del range [0, 2^R - 1] dove con R indichiamo la quantità di bit che vogliamo utilizzare per la nostra rappresentazione.

In pratica quello che facciamo è limitare il numero di valori che possono essere assunti dai nostri sample.
Chiaramente questa trasformazione che eseguiamo con la quantizzazione introdurrà un errore e questo sarà tanto più piccolo quanto saranno maggiori i valori che usiamo nella quantizzazione.
La differenza tra il segnale analogico reale e il valore digitale quantizzato viene chiamata errore di quantizzazione. Se vogliamo diminuire questo errore possiamo aumentare la R ovvero possiamo usare più valori per rappresentare i vari sample.
Gli errori di quantizzazione sono di vario tipo, abbiamo un overload quando abbiamo che non riusciamo a rappresentare il sample con la quantità di bit R che abbiamo scelto, quindi in pratica abbiamo che per qualche t s(t) > 2^R-1. Un altro tipo di errore è il granular noise in cui invece abbiamo che tutti i sample presenti all'interno di un certo intervallo vengono rappresentati con un singolo valore approssimato y_k.

24) IEEE 802.15.4: slides-05A_IEEE802.15.4&ZigBee 2015

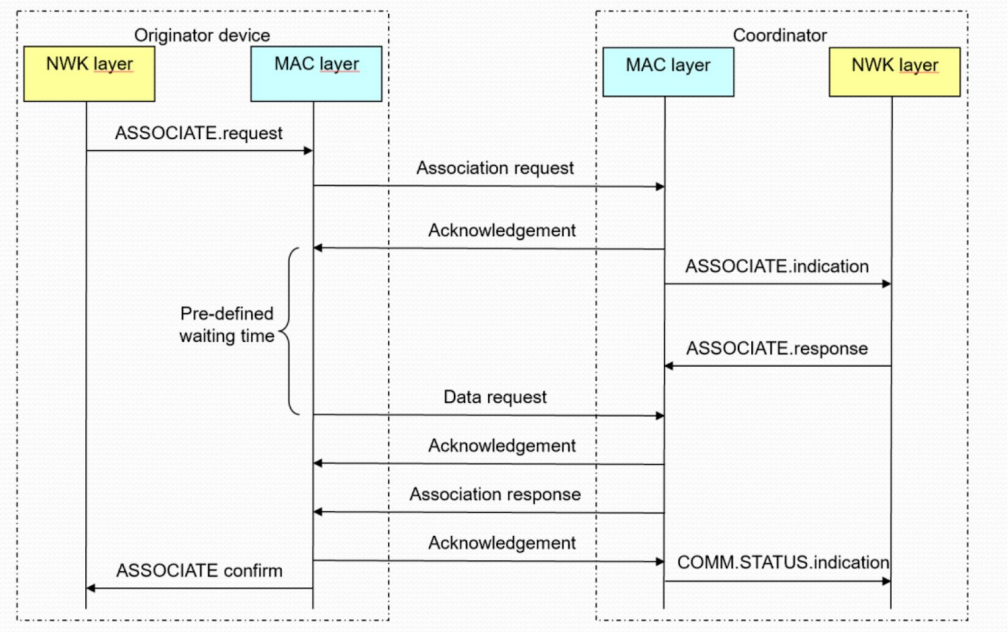


Lo standard IEEE 802.15.4 è stato concepito per regolamentare il livello fisico e il livello MAC (Media Access Control) di reti in aree personali (ovvero a corto raggio, tipicamente inferiore a 30 m) che lavorano con basse velocità di trasferimento dati.
Siamo a livello MAC, in particolare stiamo considerando l'accesso al canale condiviso, ci sono due possibili soluzioni, una consiste nell'utilizzo di una struttura dati chiamata superframe e una invece non prevede l'utilizzo del superframe.
In particolare quella nell'immagine è la struttura con il superframe che viene tipicamente utilizzata con la topologia a stella o con la topologia peer to peer ad albero, all'interno di queste topologie andiamo a distinguere due tipi di dispositivi, i Full function devices e i reduced function devices. I primi implementano tutte le funzioni del livello MAC e possono essere coordinatori all'interno delle reti mentre i secondi implementano solamente una parte del livello MAC.
In questo caso è il coordinatore della rete che fa lo sforzo maggiore per garantire la comunicazione dei vari dispositivi mentre i dispositivi cercano di preservare il più possibile l'energia.
Quando utilizziamo il superframe abbiamo una divisione in due periodi:

- Active Period: all'inizio di questo periodo il coordinatore invia un beacon frame che contiene le informazioni che descrivono la rete, ad esempio in questo modo facciamo sapere in quale canale stiamo lavorando e la lunghezza dei due periodi. Il beacon serve anche a sincronizzare i dispositivi e a identificare la PAN.
- Inactive Period: durante questo periodo non ci sono comunicazioni quindi sia il coordinatore che gli altri nodi possono spegnere la radio, alla fine di questo periodo viene spedito un altro beacon frame.

L'active period viene diviso in slots (al massimo sono 16 slots), se un dispositivo vuole comunicare deve attendere l'inizio dello slot. L'active period è diviso in due parti la prima parte è la CAP in cui l'accesso dei vari dispositivi allo slot viene deciso tramite CSMA/CA mentre la seconda parte è il CFP che è destinato alle applicazioni che hanno una particolare richiesta in termini di bandwidth e quindi viene assegnato ad uno specifico dispositivo. Il CFP è diviso in al più 7 GTS e ognuno è assegnato ad un dispositivo.
Il CAP è indispensabile perchè quando un dispositivo deve accedere alla rete non è conosciuto ancora dal coordinatore e quindi non può comunicare nel CFP.

25) IEEE 802.15.4: slides-05A_IEEE802.15.4&ZigBee 2015



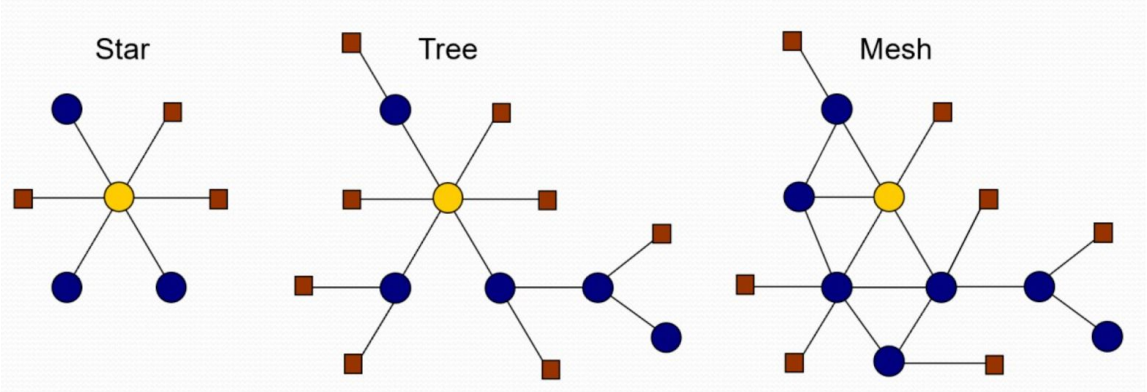
Lo standard IEEE 802.15.4 è stato concepito per regolamentare il livello fisico e il livello MAC (Media Access Control) delle personal area network (ovvero a corto raggio, tipicamente inferiore a 30 m) che lavorano con basse velocità di trasferimento dati.
Consideriamo quello che succede a livello MAC quando un nuovo nodo vuole entrare all'interno del PAN, vediamo come funziona nel dettaglio l'associate protocol:

- Nel dispositivo che si vuole collegare al coordinatore abbiamo che al livello di rete viene inoltrata una Associate.Request al Mac Layer, il Mac Layer del dispositivo trasmette una association request al Mac Layer del coordinatore che invia subito un ACK.
- La richiesta di associazione deve essere controllata dal coordinatore e questo avviene quando dal Mac Layer del coordinatore viene inviata una Associate.indicator all'upper layer.
- Tramite l'Associate.indicator il coordinatore viene effettivamente a sapere di questa richiesta, il coordinatore può sia rifiutarla sia accettarla. Se l'accetta deve selezionare un indirizzo di 16 bit per il dispositivo che deve connettersi. Dopo l'associazione l'end device dovrà poi sempre utilizzare questo indirizzo al posto del suo indirizzo originale. Per inviare questa risposta il network layer del coordinatore invia una Associate.response.
- Questa network response serve per inviare una risposta al dispositivo che ha chiesto di associarsi. Questa viene trasmessa durante una trasmissione indiretta.
- La trasmissione non può essere diretta perché il dispositivo non ha un indirizzo di rete. Quindi quello che succede è che il dispositivo una volta che riceve l'ACK saprà che la decisione del coordinatore richiederà un certo periodo di tempo, quindi il dispositivo attenderà un certo periodo di tempo, quando il tempo finisce il coordinatore avrà già inviato la Associate.Response. Il dispositivo allora può inviare una Data Request al coordinatore, il coordinatore invia un ACK e poi l'association

response message che contiene il nuovo indirizzo di rete del dispositivo. Alla fine il dispositivo invia un ACK al coordinatore. Gli ACK sono necessari per fare in modo che i due MAC layer sappiano che il protocollo sta andando avanti.

- A questo punto il Mac Layer del dispositivo conosce il nuovo indirizzo di rete ma il network layer del dispositivo non lo sa ancora così come non lo sa il network layer del coordinatore. Questa ultima comunicazione viene effettuata tramite un Associate.confirm che viene inviata al network layer dal Mac Layer mentre nel coordinatore non abbiamo più primitive da utilizzare e quindi si utilizza il communication status per inviare al network layer del coordinatore l'informazione della nuova associazione.

26) Zigbee: slides-05A_IEEE802.15.4&ZigBee 2015



Zigbee è uno standard di comunicazione costruito sopra allo standard 802.15.4. Zigbee specifica il livello rete e il livello applicazione, in particolare all'interno del livello applicazione ci sono degli altri componenti che vengono implementati a seconda delle necessità di chi sta producendo il dispositivo. A livello rete possiamo distinguere tre differenti dispositivi per la rete in Zigbee:

- End-Device: sono sia i Reduced Function Device RFD sia i Fully Function Device FFD, nell'immagine sopra sono quelli in rosso.
- Router: un dispositivo FFD che fa anche il routing, sono quelli blu
- Coordinatore: è quello giallo è un FFD che gestisce tutta la rete.

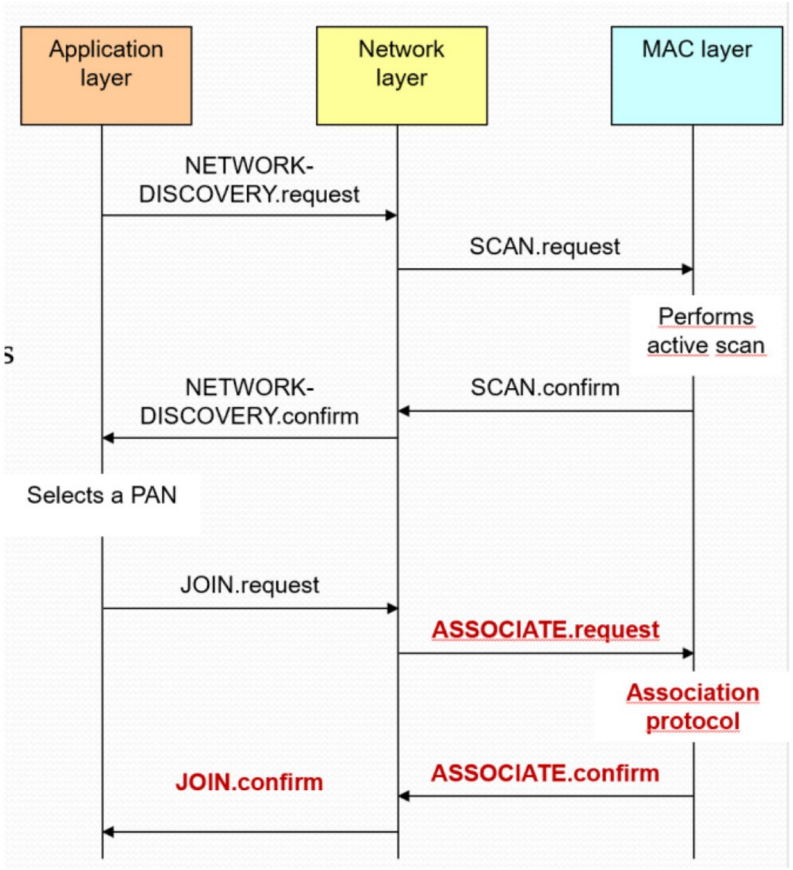
Distinguiamo anche tre topologie differenti:

- A stella: è il mapping naturale dello standard 802.15.4, questa topologia è pensata per l'utilizzo del superframe. La topologia a stella consiste in un coordinatore che è connesso con tutti gli end device. Lo scambio di dati tra i vari end device avviene tramite un coordinatore. Lo svantaggio di questa topologia è che la comunicazione dipende troppo dal coordinatore che quindi potrebbe diventare una bottleneck, la cosa positiva è che è semplice e che i pacchetti al massimo fanno due salti per arrivare a destinazione.
- Ad albero: in questo caso possiamo. se vogliamo. utilizzare una struttura superframe. Abbiamo un nodo centrale che fa da coordinatore, poi abbiamo gli end devices che devono essere connessi al coordinatore oppure ad un router. Il router mi serve per estendere la copertura della rete, gli end device sono sempre figli di un router o del coordinatore, non sono mai collegati direttamente. Solamente router e coordinatore possono avere figli. Lo svantaggio è che se un padre viene disattivato allora i figli risulteranno disconnessi dal resto della rete e anche il fatto che se due nodi sono geograficamente vicini tra loro allora non potranno comunicare direttamente.
- Mesh: la comunicazione funziona senza la struttura superframe. Abbiamo un coordinatore, vari router e gli end devices. In questo caso abbiamo che la comunicazione tra i vari dispositivi avviene tramite multi hop, se un path fallisce ne viene trovato un altro per raggiungere la destinazione, i dispositivi sono più connessi tra loro in modo da poter usare meno potenza. Rispetto alla topologia a stella

abbiamo un maggiore overhead e usa anche un protocollo più complesso per la comunicazione rispetto alla topologia a stella.

In particolare possiamo anche dire che il network layer di Zigbee è importante perchè offre dei servizi che consentono una gestione dinamica della rete e questo è importante in Zigbee perché abbiamo tanti dispositivi all'interno della rete. A livello di rete Zigbee gestisce la trasmissione dei pacchetti, l'inizializzazione della rete, il routing e la gestione di connessione e disconnessione dei dispositivi.

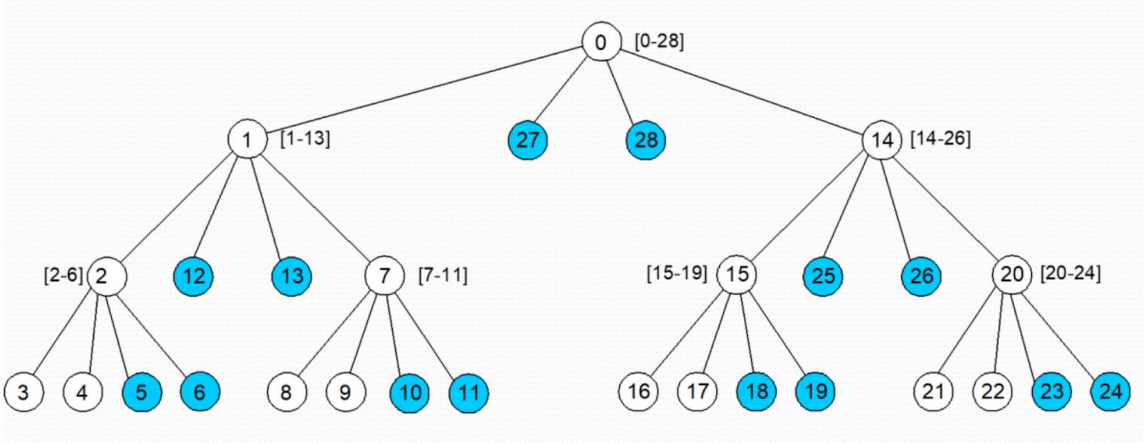
27) Zigbee slides-05A_IEEE802.15.4&ZigBee 2015



Per entrare all'interno della rete Zigbee ci sono due possibilità, l'end device o il router possono comunicare con il coordinatore per richiedere l'accesso alla rete oppure il coordinatore può direttamente contattare un dispositivo per richiedere di eseguire la join all'interno del PAN.
In particolare quello nell'immagine è la join through association ovvero la join del PAN che viene avviata da dispositivo o dal router che vuole accedere alla rete. Nell'immagine vediamo quello che succede nel dispositivo che si vuole unire alla rete, poi dall'altra parte a rispondere c'è il coordinatore della rete. La comunicazione tra i due dispositivi avviene tramite il MAC layer e quindi viene utilizzata la association procedure del mac layer.
Qui succede che l'application layer del dispositivo che avvia la connessione avvia la procedura di Join inviando al network layer una richiesta in cui si chiede di fare una scansione per capire se ci sono PAN nelle vicinanze a cui connettersi. Questa richiesta viene inoltrata al MAC Layer del dispositivo che esegue la scansione delle PAN. Questo active scan restituisce la lista delle PAN che sono state trovate che vengono comunicate al network layer tramite SCAN.confirm e poi questa conferma viene inoltrata all'application layer che deve selezionare una delle PAN che sono state trovate dal MAC Layer. Quando abbiamo scelto la PAN a cui collegarsi l'application layer manda la richiesta di join al network layer che poi inoltra la richiesta al MAC Layer. In particolare la richiesta join.request è quella che viene svolta per selezionare un nodo padre nella rete zigbee tra quelli che abbiamo nelle vicinanze. Nel caso in cui stiamo utilizzando la rete con topologia a stella il nodo padre è lo stesso coordinatore e quindi noi ci aggiungiamo alla rete come end device, se invece abbiamo una topologia ad albero il padre sarà un router o il coordinatore e noi saremo un end device o un router.

Ora da qui in poi viene utilizzato l’association process del MAC Layer che mi restituirà alla fine della procedura una conferma dell’associazione che poi viene inoltrata al network layer e all'application layer. Qua con questa procedura il dispositivo riceve un indirizzo di 16 bit che verrà utilizzato per le successive comunicazioni con la rete.

28) Zigbee slides-05A_IEEE802.15.4&ZigBee 2015



Con il protocollo di Zigbee nel momento in cui un nodo viene aggiunto all'interno di una rete, viene sempre e comunque assegnato ad un padre, questo padre nel caso in cui abbiamo una topologia a stella è il coordinatore della rete, se invece siamo in una topologia ad albero possiamo avere che il padre è un router oppure anche il coordinatore.
In entrambi i casi quindi si crea un rapporto padre-figlio tra i nodi della rete. In particolare nell'immagine abbiamo che i nodi bianchi sono i router mentre i blu sono end devices e il nodo 0 è il coordinatore. Qui in questo caso dobbiamo settare una serie di parametri per far funzionare la rete, ad esempio in questo caso decidiamo di avere un numero massimo di router figli pari a 2, un numero massimo di end devices figli pari a 2 e la profondità dell'albero pari a 3. In generale quando arriva un end devices si cerca sempre di metterlo il più alto possibile all'interno di questo albero perché riusciamo a limitare il più possibile il numero di hop necessari. Il coordinatore quando inizia ha ha disposizione 29 indirizzi, uno è il suo, i rimanenti 28 vengono assegnati agli altri 28 nodi che possono entrare nella rete (è un massimo). I primi 13 indirizzi sono per il sottoalbero a sinistra, i 13 successivi per quello a destra. Il coordinatore poi memorizza due indirizzi per gli ultimi due end devices che può avere come figli. Una problematica che si pone in questo caso riguarda il fatto che se abbiamo riempito tutto il sotto albero a sinistra e non abbiamo un altro router da mettere a destra, non potremo far entrare gli end devices all'interno della rete. Questa è una limitazione dovuta al fatto che la configurazione è molto rigida, sono state anche pensate delle soluzioni per renderla più dinamica ma c'è da dire che questa è più semplice ed efficiente da utilizzare.

29) Zigbee slides-05A_IEEE802.15.4&ZigBee 2015

Src Addr (64 bits)	Src EP	Cluster ID	Dest Addr (16/64 bits)	Addr/Grp	Dest EP
0x3232…	5	0x0006	0x1234…	A	12
0x3232…	6	0x0006	0x796F…	A	240
0x3232…	5	0x0006	0x9999	G	_
0x3232…	5	0x0006	0x5678…	A	44

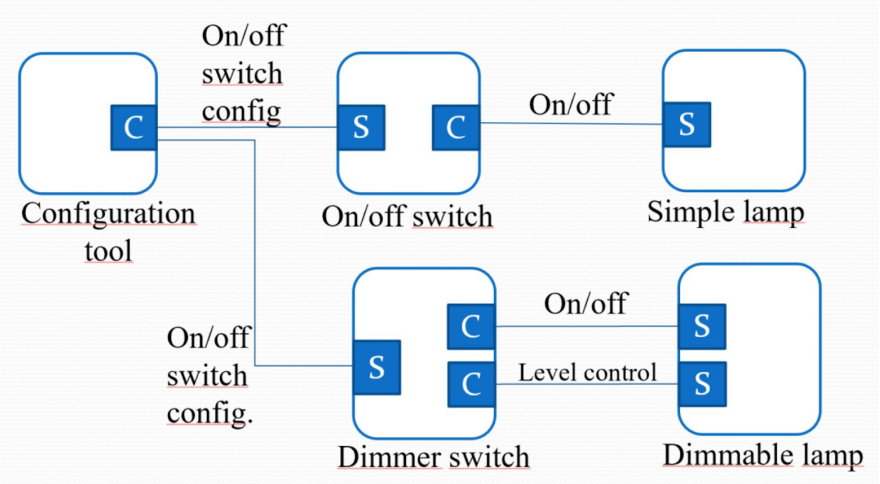
Solitamente un messaggio viene normalmente spedito a destinazione usando il direct addressing ovvero basandosi sulla coppia <endpoint di destinazione, indirizzo di destinazione>, in molti casi però questa cosa non è possibile e quindi deve essere utilizzato un indirect addressing che possa sfruttare le binding tables.
Quella dell'immagine è una binding table che ci permette di fare questo indirect addressing, la binding table viene memorizzata nel coordinatore e nei router e viene aggiornata solamente su esplicita richiesta dello ZDO (Zigbee Device Object) del coordinatore o di un router. All'interno della binding table vengono memorizzate le seguenti informazioni:

- Abbiamo l'indirizzo di destinazione e l'indirizzo mittente del pacchetto che stiamo cercando di inviare. Notare che qua non memorizziamo l'ip del dispositivo ma memorizziamo il MAC Address perchè in questo modo limitiamo le modifiche che devono essere effettuate a questa binding table. Per mantenere l'associazione tra Mac Address e Network address poi viene utilizzata una Address Map, in questo modo questa è l'unica tabella che modifichiamo mentre invece la binding table rimane sempre valida anche quando un nodo si connette o si disconnette. La address map invece viene aggiornata quando un nodo entra o esce dalla rete.
- Abbiamo l'endpoint del mittente da cui parte il pacchetto
- Abbiamo il cluster ID del mittente
- Possiamo scegliere se la destinazione è solamente un singolo dispositivo o se è un gruppo
- Abbiamo l'endpoint del destinatario.

Possiamo notare dalla binding table che qua noi abbiamo sempre lo stesso indirizzo mittente e in particolare:

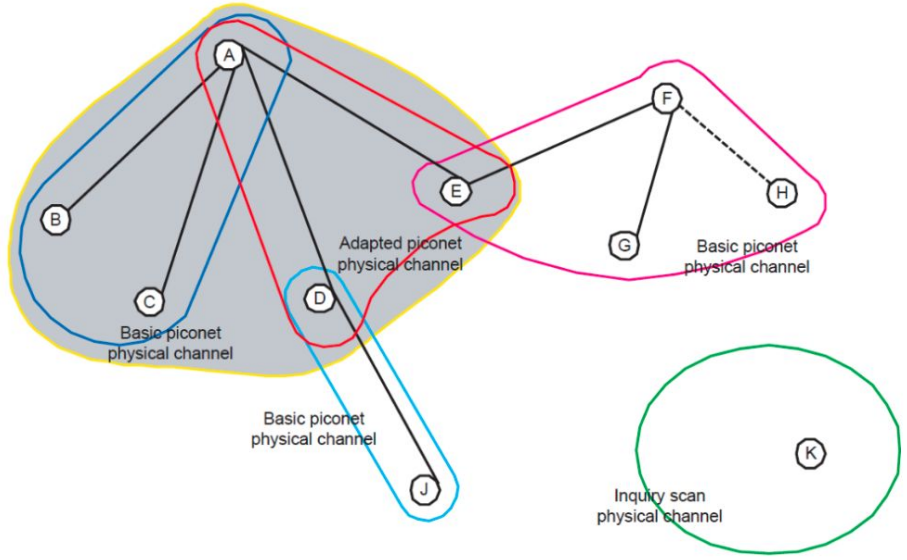
- Il primo e il terzo condividono lo stesso indirizzo mittente, l'endpoint di destinazione e il cluster ma il primo ha come destinatario un singolo indirizzo mentre il terzo ha come destinatario un gruppo.
- Il primo con il secondo condividono l'indirizzo e il cluster ma provengono da due endpoint differenti e quindi abbiamo due destinazioni differenti.
- Il primo e il quarto hanno tutti uguale ma poi sono diversi come destinatario e come endpoint nel destinatario.

30) Zigbee slides-05A_IEEE802.15.4&ZigBee 2015



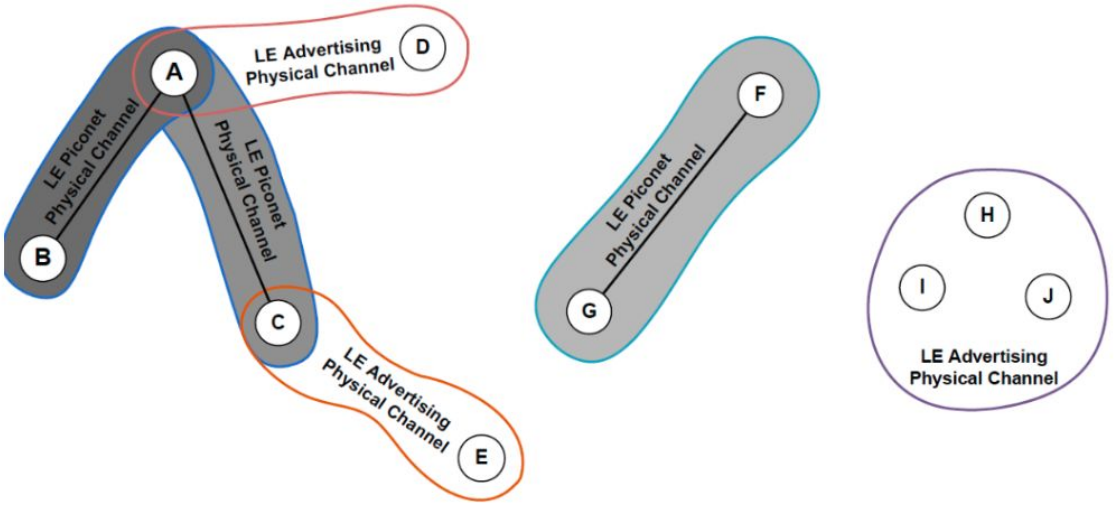
All'interno di una rete Zigbee possiamo creare dei cluster che sono delle collezioni di attributi e di comandi che definiscono l'interfaccia di una specifica funzionalità.
Il dispositivo che memorizza gli attributi è il server del cluster mentre quello che manipola gli attributi è il client.
I comandi in particolare sono messaggi con un formato specificato dalla Zigbee Cluster Library e possono essere attributi di lettura o scrittura o possono essere utilizzati ad esempio per generare un report.
Quando dobbiamo configurare dei dispositivi all'interno della rete Zigbee si utilizza un configuration tool che deve potersi collegare ai vari dispositivi come se fosse un client per applicare le varie impostazioni ai dispositivi.
Nell'immagine in particolare c'è la configurazione di un set di APO, abbiamo infatti un On/Off switch che deve essere in grado di gestire una lampada e poi abbiamo un dimmer switch che deve gestire una dimmable lamp. Il configuration tool che viene utilizzato da chi crea la rete deve lavorare come un client settando la configurazione dello switch in modo tale che quando vuole accendere o spegnere la lampada andrà a prendere il dispositivo corrispondente e farà la richiesta di accensione/spegnimento. Per il dimmer switch vale la stessa cosa. Anche le lampade vanno configurate. Questa fase di configurazione si fa ad esempio con un tablet andandosi a connettere alla rete Zigbee e poi si accede tramite un'app alle informazioni della rete. Prima di fare questa configurazione ogni dispositivo sa quali sono le sue capacità ma non sa come integrarsi con il resto, ad esempio la lampada sa di essere una lampada ma non è connessa con l'accensione, è la stessa cosa che avviene quando abbiamo delle cuffie BT e le vogliamo collegare al telefono, i due dispositivi sanno cosa possono fare ma non sono ancora connessi. In questa fase di configurazione il configuratore assegna una serie di informazioni ai vari dispositivi, ad esempio anche informazioni riguardanti la batteria, il tipo di batteria, se siamo in una situazione di emergenza o no, oppure possiamo assegnare un attributo temperatura che a seconda del dispositivo che stiamo utilizzando sarà utile da considerare. Questi attributi vengono assegnati in base al dispositivo che stiamo utilizzando, ad esempio un sistema di allarme potrebbe non essere interessato alla temperatura, un forno si.

31) Bluetooth: slides-05B_Bluetooth - 16.9



Con Bluetooth Basic Rate ed Enhanced Data Rate i dispositivi che vogliono comunicare utilizzando Bluetooth devono connettersi prima di iniziare la comunicazione e poi rimangono connessi anche quando non c'è passaggio di dati. Quindi in questo caso abbiamo un protocollo che è connection oriented. Lo standard opera a 2.4GHz e abbiamo un rate da 1 MBPS fino a 3 nel caso dell'Enhanced Data Rate.
Quella nell'immagine è l'organizzazione che è presente all'interno di una rete in cui i dispositivi sono connessi tramite Bluetooth. Nel caso del BR e dell'EDR i dispositivi che si trovano nella stessa area sono uniti in piconet. Queste piconet sono reti molto piccole che possono contenere al loro interno fino a 8 dispositivi. All'interno della piconet non tutti i nodi hanno lo stesso ruolo, in particolare ci sono due ruoli possibili il master e lo slave.
Il master è quello che si occupa della connessione dei vari dispositivi e poi permette anche il passaggio di dati da uno slave all'altro.
Gli slave sono tutti gli altri dispositivi che entrano all'interno della piconet.
Per ognuna delle piconet abbiamo un singolo master e al più 7 slave, il master può fare parte di più piconet, ad esempio qua la A è master di due piconet, allo stesso modo uno slave può fare parte di più piconet, ad esempio qua abbiamo la D che è slave rispetto ad A ma è master per il nodo J.
In generale le varie piconet possono essere connesse tra loro e questo lo vediamo con quella in blu che viene connessa con la piconet in rossa utilizzando il nodo A che fa da gateway, queste piconet che sono unite sono chiamate scatterness e Bluetooth non prevede un funzionamento particolare per questo tipo di reti quindi se ne abbiamo bisogno dobbiamo implementare ad esempio il routing da soli.
Possiamo anche notare che abbiamo un dispositivo che è K che sta all'interno di una piconet da solo, in questo caso il dispositivo non sta in una piconet insieme ad altri dispositivi semplicemente perchè sta cercando una piconet a cui connettersi.
Qua abbiamo un limite di 7 dispositivi attivi e comunicanti all'interno della rete (oltre al master) in realtà all'interno della rete possono esserci anche altri dispositivi che però devono essere in parked mode.

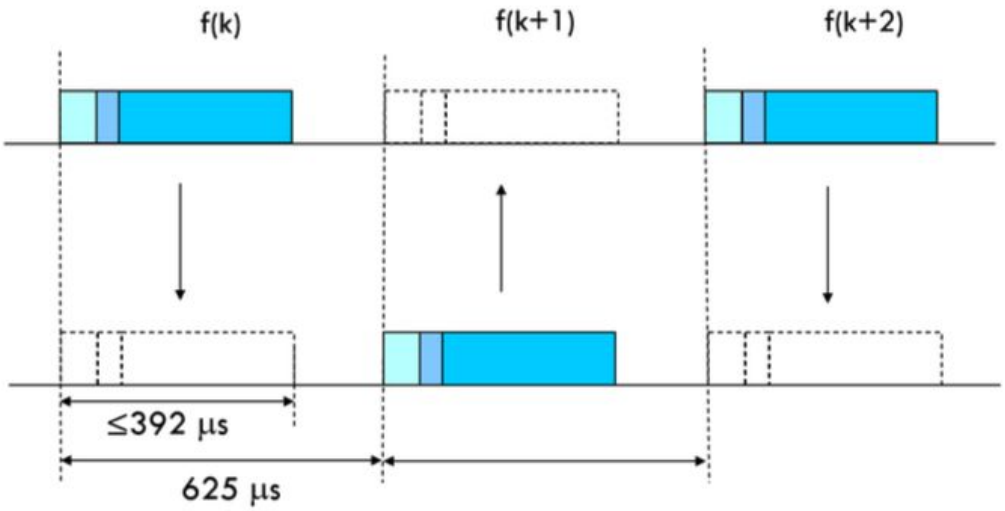
32) Bluetooth: slides-05B_Bluetooth - 16.9



Nel caso del Bluetooth Low Energy abbiamo la possibilità di far comunicare i nodi sia dopo aver formato una piconet sia senza aver formato la piconet. La piconet in questo caso è opzionale, in generale i nodi sono fuori dalla piconet e si ascoltano a vicenda a due a due, alcuni nodi saranno degli advertiser e altri saranno invece degli scanner.
Nell'immagine ad esempio abbiamo una piconet classica in cui abbiamo i nodi G e F, poi abbiamo la piconet con A, B e C. Nelle piconet che sono formate abbiamo comunque e sempre un master e dei nodi che invece sono degli slaves.
Come detto un nodo che deve comunicare non deve necessariamente essere all'interno di una piconet, quindi ad esempio abbiamo il nodo D o il nodo E che sono in grado di comunicare con il nodo A e il nodo C direttamente, in questo caso potremmo avere per esempio D che fa da advertiser e A da scanner e quindi succede che A riceverà l'internal state di D, la stessa cosa succede anche con C ed E con C che fa da scanner ed E che fa da advertiser.
In questa area abbiamo anche alcuni nodi che non fanno parte di una piconet e che comunque possono ascoltarsi tra loro, è il caso dei nodi I, H e J.

Nel caso del BLE non abbiamo un limite per quel che riguarda il numero di stazioni che sono presenti all'interno di una piconet, l'unico limite in questo caso riguarda le capacità del dispositivo e non il numero di dispositivi presenti all'interno della rete.

33) Bluetooth: slides-05B_Bluetooth - 16.9

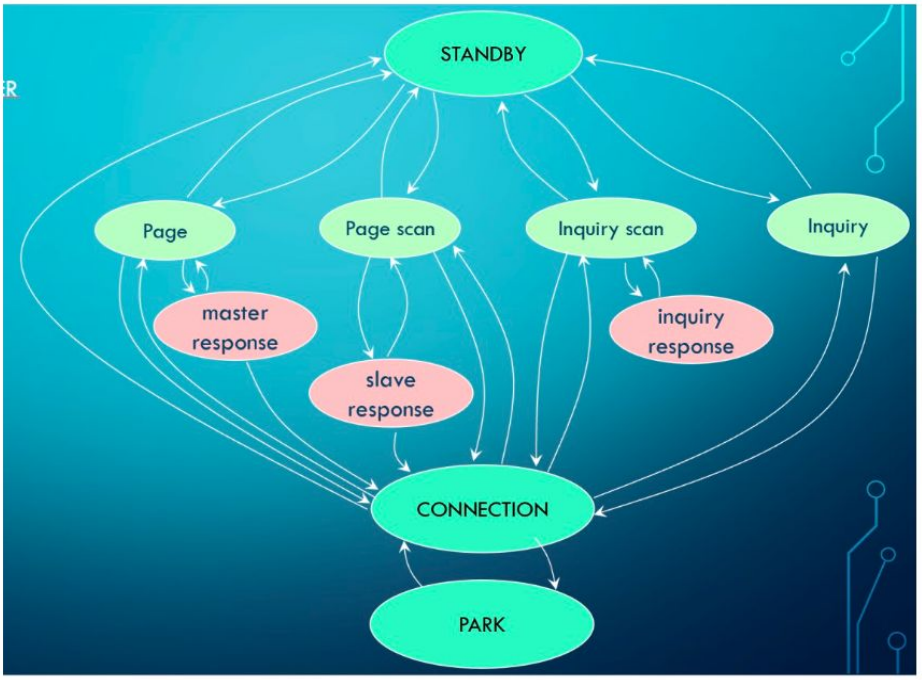


Stiamo considerando Bluetooth e in particolare la comunicazione all'interno di una piconet tra un dispositivo master e i vari slave.
La comunicazione dei dispositivi Bluetooth avviene tramite un canale fisico, ne abbiamo 79 di questi canali. Quando i dispositivi comunicano si viene a formare una sequenza di salti casuali presi tra le 79 frequenze in cui viene divisa la banda, la sequenza di salti è unica nell'ambito di una piconet ed è determinata dal master. Il canale è diviso in time slot di 625ms, ogni slot è utilizzabile in entrambi i versi della comunicazione per trasmettere un pacchetto, ogni slot corrisponde ad una particolare frequenza nella sequenza di salto. Vengono svolti 1600 hop per secondo.

Se viene utilizzata la tecnica TDD (Time Division Duplex) TDMA master e slave trasmettono in modo alternato, prima il master negli slot dispari e poi lo slave negli slot pari. Il pacchetto che trasmettiamo non può avere una lunghezza maggiore di 5 time slots, la frequenza di salto è costante per tutta la durata di un pacchetto.

Il caso più semplice è quello in cui abbiamo solamente un master e uno slave perchè la comunicazione avviene solamente tra questi due dispositivi, se invece abbiamo più di un singolo slave allora abbiamo che il master contatta uno slave, e poi nello slot successivo risponderà lo slave che era stato interpellato.

34) Bluetooth: slides-05B_Bluetooth - 16.9



Quando utilizziamo il Bluetooth BR/EDR abbiamo vari possibili stati in cui ci possiamo trovare:

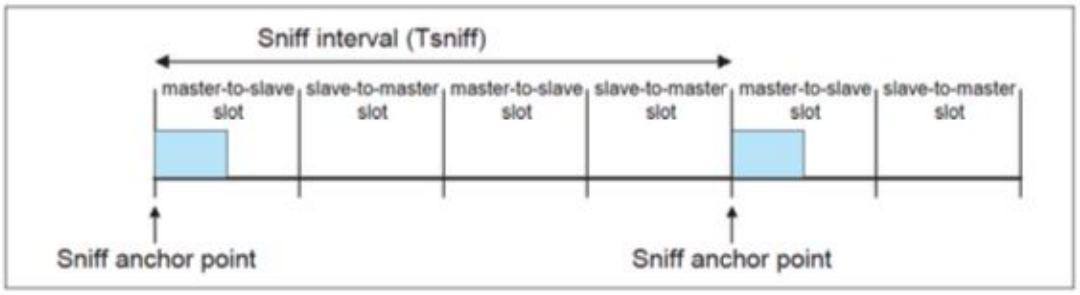
- Standby: il nodo che vuole entrare all'interno di una piconet ma che attualmente non è ancora connesso si trova in uno stato iniziale di standby da cui cerca di uscire connettendosi ad una rete.
- Connection: è lo stato in cui il dispositivo fa parte della piconet, è connesso e quindi può comunicare.
- Park: in questo stato il dispositivo non partecipa attivamente alla piconet ma comunque rimane sincronizzato e continua a ricevere i messaggi che vengono inviati in broadcast all'interno della rete. Per tornare di nuovo in stato connection deve chiedere al master.

Per muoversi da uno stato all'altro poi ci sono alcuni stati intermedi in cui il nodo può passare:

- Inquiry: per fare in modo di creare una connessione un dispositivo deve essere in grado di capire se ci sono dei dispositivi nei paraggi, questo è l'obiettivo della procedura di inquiry. Questo processo viene avviato da un dispositivo che vuole conoscere i suoi vicini o vuole creare una connessione. Tipicamente un dispositivo entra in inquiry mode in modo periodico. Importante notare che la procedura di Inquiry permette ai due dispositivi di conoscere e di scambiarsi alcune informazioni che saranno poi utili per la connessione, in particolare l'indirizzo.

- Inquiry scan: un dispositivo che vuole rendersi visibile agli altri dispositivi entra ad intervalli regolari nello stato di inquiry scan. Per trovarsi due dispositivi devono essere uno in Inquiry e uno in Inquiry Scan.
- Page: un dispositivo entra in modalità page ad intervalli regolari. questo processo consiste nella formazione della connessione tra due dispositivi Bluetooth, è importante che questa procedura sia fatta dopo l'inquiry perché dobbiamo conoscere l'indirizzo del dispositivo a cui vogliamo connetterci.
- Page Scan: il dispositivo a cui vogliamo connetterci e di cui abbiamo l'indirizzo deve essere in stato page scan, alla fine della procedura di connessione avremo che questo dispositivo sarà lo slave mentre quello che ha avviato la connessione sarà il master.

35) Bluetooth: slides-05B_Bluetooth - 16.9

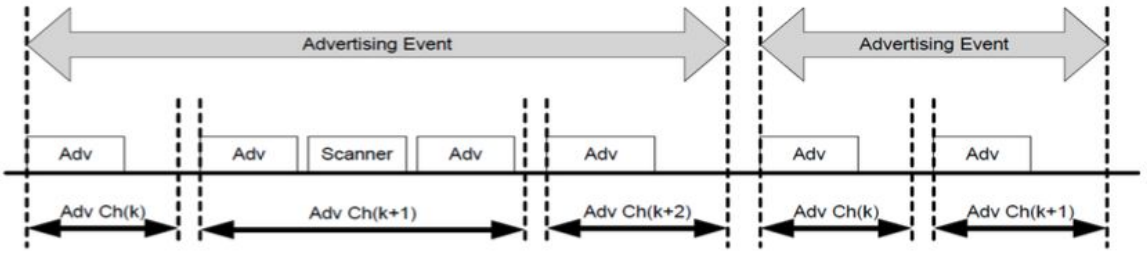


Una volta che un dispositivo Bluetooth è sincronizzato allora può entrare in uno stato di risparmio energetico che viene chiamato Sniff Mode, questo permette il dispositivo di ascoltare la piconet ad un rate inferiore.
Il periodo di tempo che passa tra due sniff può essere configurato in base all'applicazione che stiamo utilizzando e in base al dispositivo.

- Anchor point: lo slot in cui lo slave ascolterà la trasmissione del suo master.
- Tsniff: è il numero di slots che troviamo tra due anchor points

La sniff mode influenza solamente l'ACL logical transport, questo vuol dire che quando viene attivata questa modalità la trasmissione sull'ACL logical transport viene ridotta. Nella sniff mode il dispositivo quindi può "assentarsi" dalla piconet per un certo tempo, il duty cycle quindi verrà definito in base al tempo in cui il dispositivo sarà presente nella rete e sarà attivo rispetto al tempo in cui sarà assente. Un possibile utilizzo dello sniff mode è tra un computer e una tastiera bluetooth, in questo caso l'utente potrebbe non scrivere qualcosa per un certo periodo di tempo e quindi sarà possibile passare in sniff mode.
Quando l'utente poi cliccherà un tasto la connessione viene riportata nello stato attivo iniziale.
La sniff mode può tornare utile anche quando il dispositivo deve entrare in una scatterness, per un certo periodo infatti sarà assente da una certa piconet e poi entrerà nell'altra.

36) Bluetooth: slides-05B_Bluetooth - 16.9



Ci troviamo nel caso del Bluetooth Low Energy e più nello specifico nel caso in cui i vari dispositivi cercano di comunicare senza però essere realmente connessi uno con l'altro. Con BLE infatti la comunicazione può avvenire tramite la trasmissione di un evento che viene chiamato advertisement.
Ogni canale fisico nel caso di LE è diviso in unità di tempo ovvero in eventi, questi eventi sono Connection Events e Advertising events come nel caso della immagine.

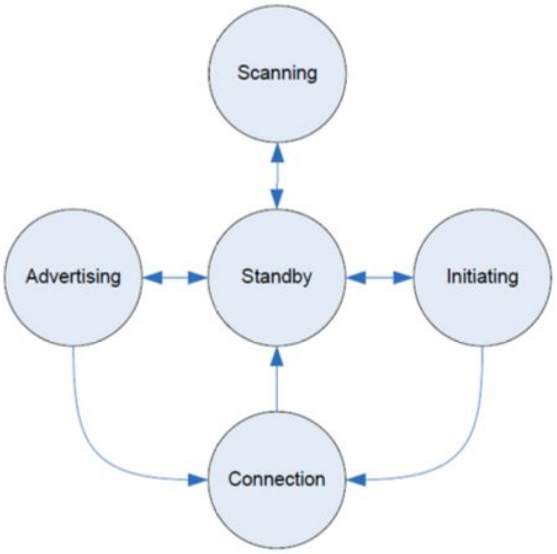
BLE utilizza 40 canali, di questi 3 sono Advertising channel che possono essere utilizzati per l'invio di dati o anche per capire se ci sono altri dispositivi nelle vicinanze. Gli altri 37 sono invece dei Data Channels.

Con BLE abbiamo uno schema TDMA (Time division multiple Access) e uno schema FDMA (Frequency division multiple access) in cui il dispositivo salta in 40 canali.
Nel caso TDMA il dispositivo trasmette un pacchetto ad un tempo determinato e poi lo scanner se vuole risponde dopo un certo intervallo.

Nello specifico nell'immagine vediamo come funziona l'evento di advertising ovvero la comunicazione tra un dispositivo BLE e gli altri quando non abbiamo connessione diretta. Gli advertisement sono inviati in frequency hopping e sono ricevuti da altri dispositivi che operano nelle vicinanze, nello specifico sappiamo che gli advertisement trasmettono pacchetti in 3 advertisement channels, qua infatti viene usato il canale k, k+1 e k+2 e poi iniziamo un secondo ciclo di advertising event. Nel primo ciclo di advertising event in particolare abbiamo che il dispositivo invia un primo advertisement nel primo canale, quando fa l'invio nel secondo canale può succedere che lo scanner risponde chiedendo che vengano inviate ulteriori informazioni, l'advertiser poi risponderà di nuovo inviando i dati successivi. Vengono svolti vari cicli di advertising event e possiamo notare anche che l'advertiser può smettere quando vuole anche se l'evento è in corso.

Questo quindi è il caso connectionless ma possiamo anche creare una piconet con BLE, in questo caso viene utilizzato un connection event. Nel caso del connectionless abbiamo che il dispositivo che invia l'advertisement è un broadcaster che invia in broadcast queste informazioni e poi c'è uno scanner che invece riceve e legge le informazioni. Il broadcaster potrebbe essere per esempio un sensore di temperatura mentre lo scanner potrebbe essere uno schermo che visualizza i dati.

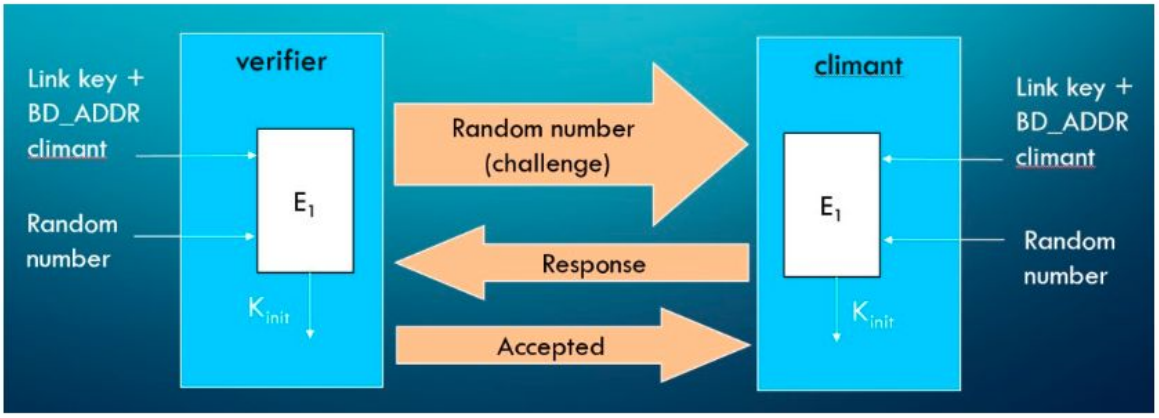
37) Bluetooth: slides-05B_Bluetooth - 16.9



La state machine di BLE è molto semplice:

- Standby: il dispositivo in questo caso non trasmette e non riceve nessun pacchetto.
- Advertising: quando il dispositivo è in standby può passare allo stato di Advertising, in questo modo trasmettiamo un advertising packet e poi attendiamo una risposta che può arrivare da un altro dispositivo. Un pacchetto advertising può essere utilizzato per due motivazioni, possiamo indicare ad un altro dispositivo (sia se sappiamo chi è il destinatario sia se non lo sappiamo) che il nostro dispositivo è pronto ad accettare richieste di connessioni in alternativa possiamo inviare informazioni a tutti i nodi che sono nelle vicinanze. Un pacchetto di questo genere è piccolo e quindi può contenere una quantità limitata di informazioni, per superare questa limitazione, è supportata una risposta che ci permette di inviare un secondo pacchetto che contiene dati addizionali che possono essere richiesti da un client utilizzando una scan request senza stabilire una connessione permanente con il dispositivo. In un advertising packet sono contenuti un certo numero di campi che tipicamente includono il nome del dispositivo e l'elenco dei servizi supportati dal dispositivo.
- Scanning: in questo stato il dispositivo attende un advertiser nell'advertising channel e si muove poi nello standby state per selezionare un advertiser in modo da iniziare la connessione.
- Initiating: il dispositivo che si trova in questo stato è chiamato initiator e possiamo arrivare in questo stato dallo standby. In questo stato il dispositivo attende un pacchetto da uno specifico dispositivo BLE e risponde anche a questi pacchetti per iniziare la connessione, quando inizia la connessione ci muoviamo nello stato connection.
- Connection: un dispositivo nello stato Connection può assumere due ruoli differenti, può essere il master o uno slave, i dispositivi che sono nello stato connection comunicano in base al tempo che viene definito dal master.

38) Bluetooth: slides-05B_Bluetooth - 16.9



Per quanto riguarda la sicurezza in Bluetooth abbiamo due obiettivi, vogliamo proteggerci contro attacchi man in the middle e vogliamo evitare che qualcuno possa leggere i messaggi che spediamo utilizzando Bluetooth. A seconda del tipo di applicazione che stiamo cercando di sviluppare avremo differenti access model in bluetooth.

Quella dell'immagine è una procedura di autenticazione del dispositivo che vuole connettersi.

Nella notazione BT il dispositivo che si connette ad un network esistente viene chiamato climant, dall'altro lato abbiamo il verifier. Il ruolo del climant è dimostrare che conosce il segreto che gli permette di connettersi alla rete, questo segreto è la link key. Il verifier è tipicamente il master della piconet e il suo ruolo è quello di verificare che il climant conosca davvero il segreto.
In pratica il climant che vuole entrare nella rete richiede l'autenticazione e si usa un meccanismo di challenge/response per verificare la reciproca conoscenza della stessa link key.
Segue uno standard response mechanism e funziona in questo modo:

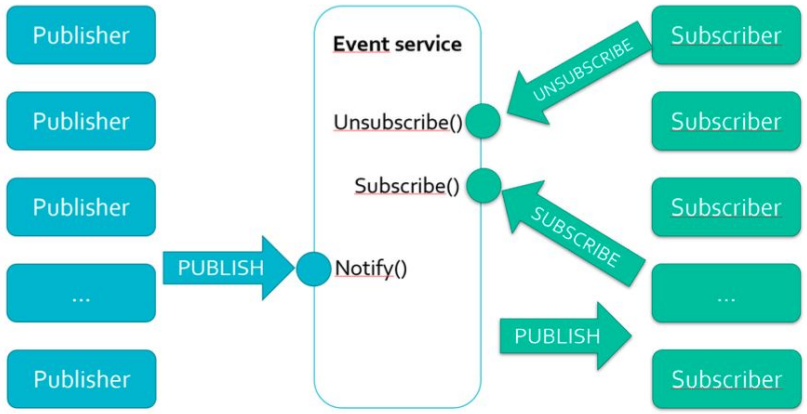
- Il climant chiede l'autenticazione e quindi il verifier inizia la procedura di comunicazione con il climant. Durante questa prima connessione il verifier va anche a conosce il BR_Addr del climant, il BR_Addr è un indirizzo unico per ogni dispositivo e pubblico di 48 bit.
- Quando il climant richiede la connessione il verifier produce un numero random e lo invia al climant.
- Il climant produce una risposta che viene mandata al verifier, la risposta è prodotta partendo da questo random number, dal BR_Addr del climant e dalla link key. La link key è una chiave privata di 128 bit. La link key viene generata partendo dal codice che inserisce l'utente che vuole connettere il dispositivo Bluetooth.
- Il verifier fa la stessa cosa perchè conosce il random number e conosce la link key e il MAC address del climant (perchè il climant aveva richiesto la connessione). Quindi

il verifier controlla che la risposta che arriva dal climant sia corretta e in quel caso accetta la connessione. Da questo punto in poi il climant è nella connessione.

Un side effect di questa comunicazione è che verifier e climant devono essere d'accordo su una chiave K init che viene utilizzata da questo punto in poi per criptare tutta la comunicazione.

Per quanto riguarda l'encryption possiamo utilizzare una semi-permanent key o una temporary key. La semi permanent key può essere utilizzata solamente all'interno di pacchetti unicast mentre la temporary key possiamo utilizzarla anche per pacchetti broadcast.

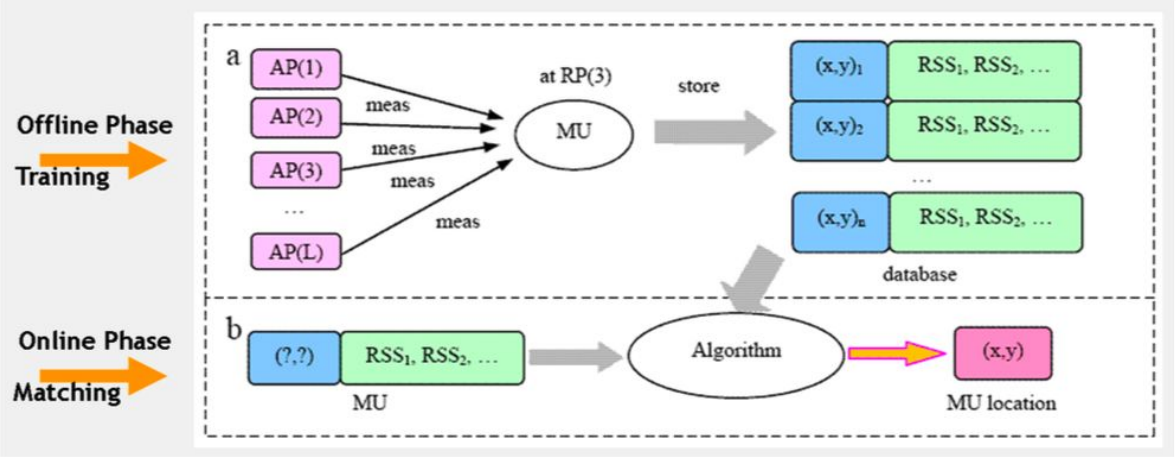
39) Publish/subscribe & MQTT



MQTT è un protocollo che è stato pensato appositamente per i dispositivi IoT per fare in modo di rendere internet più adatto loro evitando quindi di utilizzare HTTP che non è pensato per questo tipo di dispositivi. I dispositivi IoT hanno vari requisiti a diversi livelli, a livello di rete vogliono scalabilità e ridondanza, abbiamo tanti dispositivi di questo genere nella rete perchè non sono utilizzati direttamente dato che sono indipendenti ed autonomi, dovrebbero anche scalare in spazi dove non sono presenti dei router creando quindi delle reti mesh. Vorremmo anche essere in grado di configurare il livello di sicurezza di questi dispositivi e IP o TCP non me lo permettono.

MQTT è stato pensato come un protocollo di trasporto per mandare dati da un sensore ad un computer, funziona come pub sub protocol, possiamo implementarlo con poco codice e richiede una bandwidth molto bassa. MQTT prevede che nel protocollo siano presenti dispositivi con ruoli differenti, l'idea è di avere un client molto semplice lasciando tutta la parte complicata al server (ad un livello molto basso il protocollo è client server ma ad alto livello è pub-sub).
Il meccanismo di publisher-subscriber possiamo svilupparlo in questo modo, abbiamo tre tipi di dispositivi nella rete, abbiamo un broker, dei publisher e dei subscriber. Il broker è indipendente da tutto il resto, tipicamente viene installato su un computer e accetta le operazioni che vengono richieste dai publisher e dai subscriber.
In particolare i publisher possono inviare un messaggio al broker chiedendo di pubblicare un certo evento, quando si riceve un messaggio di questo genere l'event service esegue il metodo notify che avvisa i subscriber che avevano fatto richiesta di ricevere quel tipo di dato.
I subscriber invece hanno a disposizione due tipi di richieste che possono essere fatte al broker, possono chiedere di fare il subscribe di un certo evento o possono chiedere di fare l'unsubscribe. Tramite il primo ci "abboniamo" agli eventi di un certo tipo specificando un topic o un tipo di messaggio o il contenuto del messaggio a cui sono interessato, con la seconda funzione invece andiamo a chiedere che venga dimenticata la mia precedente richiesta di subscribe.
Notare che la comunicazione tra publisher e subscriber passa tutta attraverso il broker quindi i publisher ed i subscriber non hanno bisogno di conoscersi a vicenda, basta solamente conoscere il broker. La comunicazione publisher-subscriber non deve essere necessariamente sincrona.
I messaggi di Subscribe e di Unsubscribe comportano anche un ACK da parte del broker ed è obbligatorio perchè dobbiamo essere sicuri di aver fatto una di queste due operazioni

40) indoor localization – fingerprinting



Stiamo considerando il caso della Scene Analysis in cui non utilizziamo le distanze per capire la posizione dei dispositivi ma misuriamo l'RSS del target rispetto agli anchor, questi anchor vengono posizionati su una griglia all'interno di un certo ambiente.
Quello che succede è che si creano una serie di misurazioni iniziali per ogni anchor presente all'interno del nostro ambiente.
Per ogni posizione abbiamo 7 misurazioni. Quello che utilizziamo è un approccio sullo stile di machine learning o comunque di match tra varie informazioni.
La prima fase è quella sopra nell'immagine è chiamata Offline Phase e consiste nel training del nostro sistema, dobbiamo andare a prendere una serie di dati relativi alla potenza del segnale e creiamo per ogni anchor un vettore di fingerprint, all'interno del vettore abbiamo la posizione del dispositivo con associata la potenza del segnale.
Quando abbiamo concluso la fase di training possiamo utilizzare il nostro sistema per localizzare il dispositivo, in questo caso quello che facciamo è prendere un nuovo punto di cui non sappiamo le coordinate che sono il nostro obiettivo, di questo punto però conosciamo un insieme di dati della potenza del segnale (ovvero RSS Received Signal Strenght). L'idea è di fare un confronto tra questo insieme di RSS e gli altri RSS che abbiamo per tutti gli anchor e di trovare quello che è più simile in modo da capire le coordinate del punto che stiamo prendendo in considerazione.

Per fare l'online phase e quindi il matching e trovare la posizione (x,y) ci sono vari metodi, possiamo usare KNN (K Nearest Neighbors) ovvero prendiamo i nostri RSS e calcoliamo la distanza rispetto agli altri utilizzando mean square e poi la posizione del target viene stimata considerando la media delle coordinate dei migliori K dispositivi (i K che sono più vicini al set di RSS del dispositivo che stiamo localizzando).
L'alternativa più recente consiste nell'utilizzo di una Neural Network per cercare di capire il punto (x,y), quindi abbiamo il nostro dataset di fingerprint dove l'output è la posizione e dando in input il set di RSS otteniamo in output la posizione (x,y), l'accuracy di questo sistema dipende tutta da come abbiamo fatto il training e anche da quanti dati di training avevamo a disposizione.

Quali sono i possibili problemi di questa soluzione?

- Abbiamo un costo iniziale per la configurazione perchè dobbiamo costruire il nostro dataset di training con tutti i dati.
- Abbiamo dei costi per quel che riguarda il mantenimento del nostro dataset in modo che contenga informazioni aggiornate. Quindi abbiamo un costo di mantenimento.

NOTE SUI SEGNALI:

- Se abbiamo un segnale che è completamente limitato nella banda da f0 allora possiamo raccogliere tutte le informazioni che sono in quel segnale facendo il sampling ad un tempo discreto a patto che il sampling rate sia maggiore di 2f0.
- Il problema è che qua abbiamo la richiesta che il nostro segnale sia perfettamente limitato nella banda e questo non è possibile perchè non esistono segnali di questo genere perchè vorrebbe dire che il segnale si estende in modo infinito nel tempo.
- Quindi non potremo essere perfetti ma possiamo essere abbastanza bravi nel sampling del nostro segnale
- Il sampling è il processo tramite il cui un segnale continuo viene trasformato in un segnale discreto. In pratica la versione discreta di questo segnale è il segnale continuo in cui andiamo a prendere il valore del segnale ad intervalli regolari T.
- Dato che noi andiamo a non considerare il contenuto del segnale tra gli intervalli di sampling perdiamo informazione e quindi succede che si verifica una ambiguità e partendo dal segnale sampled non riusciamo a ritrovare quello di partenza. Dati vari segnali che passano per quei sample non avremo modo di distinguere quello originale da quelli finti.