Protocolli per reti ad hoc e di sensori

# Introduzione corso

## Prerequisiti

Nozioni di reti di calcolatori da laurea triennale, nozioni da reti wireless e mobili consigliate ma non obbligatori.

## Esame

Scritto su tutto il programma con domande aperte

## Orario lezioni

* Martedì 14.45 – 16.15, Aula 302
* Venerdì 14.45 – 16.15, Aula B2

### Materiale corso

* Slide Ariel
* Non ci sono libri
* Risorse varie…

## Argomenti corso

### Caratteristiche challenged networks

reti caratterizzate da sfide di progettazione, diverse dalla reti internet che usano TCP, UDP.

**Composte da dispositivi:**

* Piccoli e mobili
* Eterogenei
* Limited MTU (dati nel body e header dei pacchetti)
* Con poche risorse: bandwidth, memoria, batteria…

**La topologia delle reti cambia frequentemente:**

* Le batterie si scaricano  per risparmiare batteria i dispositivi vanno in standby
* Gli ostacoli o le interferenze possono bloccare la comunicazione wireless
* Nodi sparsi/mobili  potrebbe non esserci una sequenza di nodi intermedi che permette di individuare una rotta tra sorgente e destinazione

### Reti single-hop

Un apparato utilizza una tecnologia wireless per comunicare con una stazione base (antenna). La stazione media tutte le comunicazioni, due apparati non comunicano direttamente tra di loro, ma con la stazione base

### Reti multi-hop(MANETs)

Non c’è un infrastruttura fissa, i nodi comunicano direttamente tra di loro. Se i nodi sono troppo lontani utilizzano i nodi intermedi come router.

C’è un assunzione fondamentale: i nodi sono mobili, ma poco mobili. I protocolli di instradamento individuano una sequenza di hop tra sorgente e destinazione, ma per tutto questo tempo, e nel tempo di invio dei dati da sorgente a destinazione la topologia non cambia.

Questo non vale nelle challenged network.

### Tipi di challenged network

* WSNs
* Reti urbane
* VANETs
* IoT
* Industry 4.0
* DTNs

## WSNs – GoodFood

Usata nei vigneti.

WSN con sensori distribuiti ad alta densità sul territorio. I sensori realizzano una rete wireless. Non tutti i sensori sono uguali, ci sono micro-server che comunicano con una rete fissa.

**Funzionalità:**

* Sincronizzazione  per comunicare ed etichettare i dati in modo temporale
* Sicurezza  il canale wireless è broadcast
* Localizzazione  localizzare i sensori per dare un senso alla misurazioni senza usare un GPS
* Comunicazione data-centric invece che address-centric  si comunica con sensori caratterizzati da una certa posizione o che ha raccolto certi tipi di dato. Non si comunica per indirizzo perché dall’indirizzo non si ricava la posizione
* Scalabilità a
* Self-organization  I sensori si organizzano da soli quando la topologia cambia per avere tolleranza ai fallimenti
* Eterogeneità dei nodi

# Introduzione corso 2

## Urban / participatory sensing

Costituite dagli smartphone degli utenti.

### MetroSense projects

Collezione di progetti di urban sensing:

* Usare il microfono e il GPS del telefono per produrre commenti in una certa posizione
* Rilevazione agenti chimici
* Percorsi per ciclisti e sciatori

## Vehicular ad-hoc NETwork (VANETs)

Reti costituita da dispositivi wireless a bordo di veicoli e di apparati a lato strada.

### Traffic information system

Utilizzare le informazioni raccolte dai veicoli:

* per migliorare il calcolo delle rotte dei navigatori
* Condividere percorso taxi con più utenti
* Ridurre inquinamento
* Semafori smart
* Gestione incidenti
* Informazioni su bar / ristoranti / monumenti / attrazioni
* Tracciare veicoli rubati
* Gestire flotte (es. pescherecci)

### UAV

Unmanned ground/aerial vehicle, droni utilizzati per varie applicazioni:

* Ricerca persone disperse
* Trasporto defibrillatore
* Consegna pacchi

## Delay tolerant Networks

Reti con nodi molto mobili e sparsi, molto probabile che siano partizionate.

Sono delay tolerant perché dal momento in cui si rileva un dato, a quando viene consegnato può passare molto tempo.

### Deep-space networks

Ci sono due tipi di satelliti: satelliti in orbita bassa, satelliti in orbita geostazionaria. Quelli in orbita bassa hanno una velocità di orbita diversa da quella della rotazione terrestre, quindi i satelliti non puntano sempre nello stesso punto e anche la distanza dagli altri satelliti cambia. Quindi anche queste costellazioni di satelliti sono considerate reti challenged

### Underwater networks

Sensori trasportati dall’acqua, o sensori ancorati ma che cambiano la loro profondità nel tempo. Usati per monitorare correnti oceaniche o inquinamento acqua.

### DAKNET

La zona è così rurale che non c’è nessun ritorno economico a tirare i cavi per creare la rete internet. Ci sono dei villaggi e dei pullman o delle macchine che si muovono da un villaggio all’altro.

Nei villaggi ci sono dei kiosk su cui si possono lasciare messaggi, quando il pullman passa per un villaggio si sincronizza con il kiosk, rilascia i messaggi per quel villaggio, e carica i messaggi memorizzati nel kiosk.

### Reti opportunistiche

Le reti opportunistiche sono costituite solo da apparati wireless portati dagli utenti.

Wireless = WiFi / Bluetooth

Lo scambio di dati avviene quando c’è un’opportunità.

Idea futura per il 5G: offloading  se due dispositivi sono nella stessa cella, al posto che comunicare con la stazione base i due dispositivi comunicano direttamente tra di loro. L’antenna è usata solo per coordinare.

## Internet of Things

Interazione di “smart things” tra loro e con utenti.

# Slide 2

## Comunicazione sincrona

Normalmente la comunicazione è sincrona, il client può assumere che il server sia presente durante la comunicazione. C’è un rendezvous: cioè un appuntamento. Il server esegue il suo compito assumendo che il client rimanga presente in comunicazione fino alla risposta del server.

Randezvous stretto: dopo avere inviato i dati, il client rimane bloccato in attesa della risposta.

## Comunicazione asincrona

In reti challenged la propagazione del messaggio può richiedere tempo, perché non c’è sempre un cammino tra sorgente e destinazione. I nodi intermedi memorizzano il messaggio fino a quando non si presenta la possibilità di inoltrare il messaggio. Quindi è impossibile instaurare una comunicazione sincrona. La sorgente non fa affidamento sul fatto che la destinazione sia raggiungibile, e non aspetta la risposta perché potrebbe richiedere molto tempo.

La rete ha:

* Memoria
* Intelligenza  sa che i messaggi vanno instradati al momento opportuno, è completamente diverso da quello che fa un router normale. Il router usando la tabella di routing se scopre che non c’è un cammino per la destinazione scarta il pacchetto

Soluzione:

* Middleware per i messaggi
* Gestore di code virtuale e distribuite

## Content centric networking (CCN) / Interest centric networking (ICN)

In questo paradigma ciò che interessa al client è raggiungere un server qualunque in cui ci sia il contenuto cercato, non uno specifico server.

Il middleware aggiunge intelligenza alla rete, gestisce l’instradamento togliendo il carico al client di capire dove sono i dati.

La sorgente fa broadcast per un nome di contenuto, cerca di raggiungere tutti i dispositivi di rete raggiungibili.

**Esempio rete LAN multihost (con più router):**

In una LAN il client invia in broadcast una query a tutti i router della rete. In HTTP la query contiene l’indirizzo del server e il nome del contenuto, nel CCN la query ha solo il nome di contenuto. Un problema è trovare uno schema di naming senza ambiguità che permette al middleware di trovare il contenuto richiesto. L’interest arriva a tutti i router che hanno un middleware, i quali cercano la copia del contenuto migliore (sul server più scarico/vicino). Le tabelle di instradamento memorizzano dove sono allocati i contenuti.

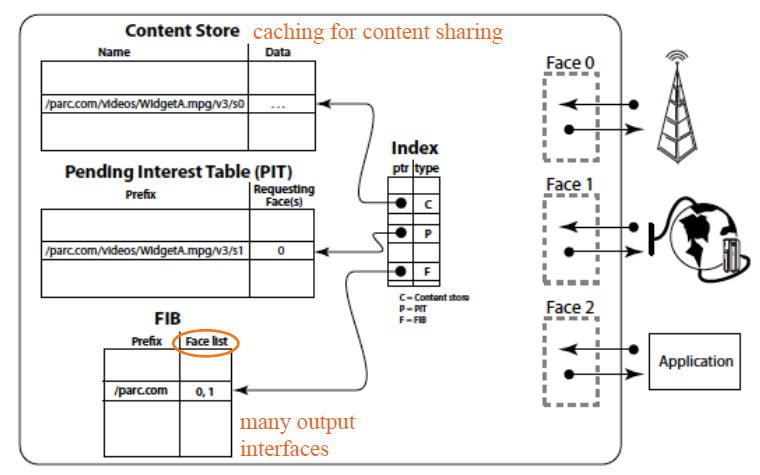
Il nome del contenuto può essere impreciso, e indicare genericamente un tipo di contenuto (keyword), il router con la sua routing table, capisce dove potrebbero essere i contenuti richiesti. L’interest ritorna un solo contenuto, trovata la direzione l’interest è reinoltrato solo in quella direzione, anche se altre destinazioni hanno un interest con una parte in comune.

**Architettura:**

* Rete tradizionale
* Middleware (CCN)
* Applicazioni

Un nodo che è nel mezzo della CCN vede arrivare degli interessi, capisce che può reinstradare gli interessi verso un nodo che probabilmente contiene il contenuto. Il nodo ricorda l’interest che ha ricevuto. Quando riceve una risposta (anche molto più tardi perché la rete è delay tolerant), la inoltra alla sorgente. Nel frattempo, il nodo può ricevere un altro interest che cerca lo stesso contenuto. In questo caso non reinoltra la richiesta perché sta ancora aspettando la risposta precedente. Memorizza comunque chi gli ha fatto questa richiesta. Quando il nodo riceve la risposta la reinoltra a tutti i nodi che avevano fatto la stessa richiesta.

## Schema CCN



### Interfacce

* Face0: interfaccia del nodo che può comunicare con un antenna
* Face1: interfaccia che parla con la rete wired classica
* Face2: interfaccia che parla con un applicazione  ci sono nodi che forniscono servizi e contemporaneamente sono dei router

### Strutture dati

**Content store**:

è una cache. Il nodo può mettere contenuti a disposizione della rete e rispondere a interessi. Inoltre, memorizza anche i contenuti ricevuti perché c’è alta probabilità che questi contenuti vengano richiesti di nuovo. In questo modo può rispondere agli interest anche se non è il creatore originale del contenuto. Il content store non è first in - first out, la memoria è limitata, alcune contenuti sono scartati secondo politiche di caching come least recently used o least frequently used.

**Pending interest table (PIT):**

Se l’interest non è memorizzato nel contest store, il nodo va a guardare nel PIT che contiene i contenuti reinoltrati nel passato. Se il contenuto è stato già reinoltrato, il nodo sta aspettando una risposta, allora alla prossima interest per lo stesso contenuto non inoltra di nuovo il contenuto.

**Forwarding information based (FIB):**

Se il contenuto non è né nel content store né nella PIT, allora devo guardare la FIB, analogo di una tabella di routing per CCN. C’è una tabella con due colonne. Si fa il matching del contenuto usando i prefissi nella colonna di sinistra, si usa la colonna di destra per reinoltrare l’interest. Per un prefisso ci possono essere più interfacce che fanno match, ma in questo paradigma la politica di reinoltro è unicast. Se ricevo due interest verso la stessa interfaccia posso fare round robin per scegliere invece un’altra interfaccia. Fare il round robin permette di avere più possibilità di ricevere una risposta perché nelle reti challenged non è detto che ci sia un cammino verso la destinazione. E inoltre si bilancia il carico facendo lavorare tutti i nodi.

## Publish-subscribe

Definizione: disaccoppiamento delle entità comunicanti, in tempo, spazio e sincronizzazione.

### Entità

* **Publisher:** qualcuno che produce contenuto e li marca per indicare cosa contiene. Il publisher manda il contenuto al broker. Il publisher non sa quanti subscriber leggono il loro contenuto.
* **Subscriber:** è il consumatore di contenuto. Il subscriber dice al broker che lui è interessato a contenuti targati in un certo modo. Aspetta poi passivamente che il broker gli porti i contenuti interessati.
* **Broker:** riceve le notifiche di pubblicazione dai publisher, riceve le notifiche di subscribe/ unsubscribe dai subscriber. Quando c’è un contenuto interessante per i subscriber inoltra il contenuto. Il paradigma non definisce che tipo di entità è il broker, dipende dalla rete.

Il broker può essere un entità distribuita, anche l’intera rete.

## Topic-based P/S

Paradigma basato su argomenti. Il publisher etichetta i contenuti generati con il nome di un canale, i subscriber possono sottoscriversi a questi canali che sono predefiniti. I canali possono poi essere organizzati in una tassonomia, una struttura ad albero in cui i nodi ai livelli più bassi sono più specializzati rispetto ai padri.

Esempio tassonomia: Animale  mammifero  canidi

Nel caso ci sia una tassonomia il subscriber quando sottoscrive ad un canale, sottoscrive anche a tutti i nodi discendenti.

### Codice sottoscrittore

Subscriber sub = new heatingSubscriber();

EventService.subscribe(sub, **“heating”**);

public class Heating implements Subscriber {

public void notify (Object o) {

if(o.temperature < 19°C)

heating\_ON();

}

}

Codice sottoscrittore interessato a tutti i dati che riguardano la temperatura. Il metodo notify() risveglia il sottoscrittore, l’oggetto Object o contiene un campo temperatura

### Scelta della tassonomia

Prima dell’installazione del sistema bisogna già decidere i topic dei dati, la struttura dei dati, e la tassonomia.

**Tassonomia a granularità grossolana:**

Se la granularità è molto grossolana il sottoscrittore riceverà anche contenuti che non interessano, questo è un problema nelle challenged network perché solitamente sono reti wireless broadcast, e si va a intasare il canale con dati inutili. Inoltre, gli apparati sono spesso a batterie. Il costo della computazione è basso, ma è alto il costo della comunicazione, ed aumenta in base alla distanza

**Tassonomia a granularità fine:**

Ci sono tanti livelli nella tassonomia, ciò permettere ai sottoscrittori di agganciarsi anche ad un livello profondo. In questo caso c’è il rischio di dover costringere un subscriber a sottoscriversi a tanti canali alternativi. Questo può portare di nuovo ad avere dati inutili.

## Content-based P/S

Sono definite delle caratteristiche che vengono analizzate dai broker della rete, che in questo modo decidono se inoltrare o meno i dati. Il sottoscrittore non si iscrive ad un canale ma a contenuti con determinate proprietà (etichette, metadati XML).

### Codice sottoscrittore

String criteria = (temperature < 19°C)

Subscriber sub = new heatingSubscriber();

EventService.subscribe(sub, **criteria**);

public class Heating implements Subscriber {

public void notify (Object o) {

heating\_ON();

}

}

* Si definisce un criterio, il broker invia il dato al subscriber solo se la temperatura è minore di 19 gradi.
* Si crea il subscriber
* Il subscriber si iscrive ai contenuti che soddisfano il criterio
* Si sveglia con notify, il dato è già stato processato dal broker

### Vantaggi/svantaggi

Non c’è da definire una tassonomia a priori, che se dovesse essere cambiata andrebbe riscritta per tutti i sensori che spesso sono difficili da raggiungere. Vanno definite delle proprietà ma è più facile che definire una tassonomia.

## Type based P/S

Le informazione sono caratterizzate da una certa struttura.

In un linguaggio object oriented sono classi. Il publisher genera un istanza di una classe, il subscriber si iscrive per ricevere oggetti appartenenti a specifiche classi

### Codice sottoscrittore

public class Sensor { public float temperature; }

public class User { public float minT, maxT; }

public class Heating implements Subscriber**<Sensor>** {

public void notify (Sensor o) {

if(o.temperature < 19°C)

heating\_ON();

}

}

* La classe utente è un partecipante al sistema, potrebbe controllare la temperatura dal telefono
* la classe Heating controlla gli attuatori del riscaldamento (i subscriber)
* Il publisher è un sensore di temperatura o il device utente
* Il subscriber riceve sia la temperatura che le preferenze dell’utente.
* Nel caso di questa notify() sono ricevuti solo oggetti di classe sensore, non quelli della classe utente. Questo perché è un attuatore che accende/spegne riscaldamento in base alla temperatura dell’ambiente

### Vantaggi / svantaggi

Nel modello type-based si sta già pensando a livello di classe, mentre nel content base si pensa in astratto, è più flessibile. Dato che il codice va realizzato in modo da poter essere scritto nella memoria limitata dei sensori, l’approccio type-based permette di sapere a priori quando occuperà il codice.

## Utilità w.r.t. sensori

* **Topic:** arrivano tutte le informazioni di temperatura anche se sono di valore maggiore della soglia
* **Type:** arrivano tutte le informazioni dei sensori, anche se sono di valore > soglia perché si ricevono tutti gli oggetti di un tipo
* **Content:** arrivano solo le informazioni di interesse  risparmio in banda ed energia

# Slide 3

## I sensore

Un sensore è un apparato il cui primo obbiettivo è rilevare dei segnali / grandezze dal mondo fisico in cui è immerso e quantificarli. Spesso hanno un sistema operativo ad hoc come TinyOS, Contiki o RIOT. Sono sistemi piccoli e molto performanti, focalizzano l’attenzione sul risparmio energetico

### Componenti sensori

* **Sensori veri e propri:** oggetti in grado di misurare una grandezza fisica analogica
  + Passivi: rilevano qualcosa nell’ambiente senza interagire con esso
  + Unidirezionali: puntano in una direzione (videocamera)
  + Omnidirezionali: misurano in tutte le direzioni (termometro, microfono)
  + Attivi: agiscono sull’ambiente per rilevare una grandezza (sonar, radar)
* **ADC:** analog digital converter, trasforma il segnale analogico in segnale digitale tramite campionamento
* **Micro-controller:** elaborazione
* **External-memory**
* **Transceiver:** radio dell’apparecchio, trasmette e riceve
* **Power source:** spesso la batteria

## Altri dispositivi

**Attuatore:** dispositivo che può modificare l’ambiente in cui è immerso (es. irrigatori, impianto di riscaldamento, motori...)

**Sink:** è un apparato che riceve tutte le informazioni rilevate dai sensori. Possono esserci più sink nell’ambiente che raccolgono dati differenti o dati dello stesso tipo. Il sink è cablato alla rete fissa e porta i dati raccolti su un cloud o ad un server. Possono elaborare parzialmente o totalmente i dati.

**Coordinatore:** in base a quello che ha raccolto dai sensori il coordinatore può mandare agli attuatori o anche ai sensori per farli funzionare in modo differente (es. aumentare la frequenza delle misurazioni)

Sensori, attuatori e sink possono essere fissi o mobili. Se sono fissi i rapporti di vicinato cambiano solo in base al duty cycle. Se sono mobili ci sono più complicazioni perché la mobilità contribuisce alla variazione della topologia.

## Caratteristiche dispositivi in WSN

Apparati anche di piccole dimensioni:

* Alimentati a batteria, rete elettrica o tramite l’ambiente (energia eolica, solare…)
* Distribuiti sul territorio da monitorare. I sensori sono sparpagliati in modo casuale, i sink e i controllori vengono piazzati. Spesso i sensori sono difficili da raggiungere una volta piazzati (es. in fondo al mare, in montagna...)
* Muniti di interfaccia wireless per la comunicazione
  + Eterogenei, i sink sono diversi dai sensori
  + Diversa disponibilità di energia e quindi diversa potenza di antenna
  + Tecnologia wireless diversa (bluetooth, wifi...)

## Multipath fading

Tutte le volte che si usa un canale wireless utilizzando un’antenna multidirezionale, il segnale viene propagato in tutte le direzioni. Il segnale può essere attenuato anche dall’acqua (quindi anche le persone, la pioggia), oppure essere riflesso dal metallo (quindi anche le automobili).

Quando le onde sono riflesse si ha il multipath fading, cioè un segnale riflesso si sovrappone al segnale originale. Se i due segnali hanno fase diversa, la somma / sottrazione algebrica delle onde sinusoidali può indebolire o interferire con il segnale originale.

## Requisiti WSN

### Accuratezza monitoraggio

Le reti di sensori misurano determinate grandezze ambientali, per poi memorizzare i valori o avvisare quando viene superato un valore di soglia preimpostata. Questi sensori devono essere piazzati in modo che tutta la superficie sia coperta dai sensori, in modo tale da verificare eventi con alta probabilità (non 100% perché i sensori si possono rompere).

### Autonomic / self organization

La rete deve potersi riconfigurare da sola perché spesso i sensori di cui è composta sono spesso difficili da raggiungere. I nodi devono quindi controllare costantemente le condizioni della rete di sensori (chi sono i vicini). Nella progettazione si deve pensare in anticipo agli ostacoli ambientali (alte temperature, campi elettromagnetici) per scrivere codice sui sensori che permette loro di riconfigurarsi.

### Energy saving

Se non possiamo collegare i sensori alla rete elettrica è saggio prevedere di costruire la parte di computazione e comunicazione per risparmiare energia. Il consumo energetico è estremamente dipendente dall’hardware e dall’uso del dispositivo.

Nella pila ISO /OSI ogni livello è una scatola nera che offre servizi al livello superiore e chiede servizi a quello inferiore. C’è un livello di astrazione tale per cui i livelli più alti sono indipendenti dall’implementazione dei livelli più bassi. Ad esempio ai livelli superiori non interessa se la comunicazione è fatta su fibra ottica o cavo in rame. Nelle WSN questo non è più vero. Lo stack va progettato tutto insieme, si sceglie l’hardware in base alle necessità della rete, e si sviluppa il software in base all’hardware. Si parla di cross layer design. Alcuni hardware consumano di più a mandare che a ricevere, altri consumano uguale. Il costo della computazione invece è poco più alto rispetto a quando il sensore dorme.

Ci sono però diversi livelli di dormienza:

* completamente dormente: CPU e radio ferma, c’è solo un orologio che fa interrupt per svegliare il sensore
* dorme poco: il sensore tiene acceso il transceiver in modalità ricezione solo per i messaggi indirizzati a lui. Ogni scheda di rete ha un indirizzo fisico. Quando arriva un nuovo messaggio,la scheda di rete controlla il mac address di destinazione, se corrisponde al suo, la scheda di rete fa un interrupt per svegliare il sensore.
* Ricezione promisqua: qualunque dato ricevuto indipendentemente dal mac address di destinazione. Il dato viene ritenuto di interesse per il sensore che viene svegliato.

É vero che in dormienza i sensori consumano molto meno rispetto a riceve e trasmettere, ma quando il sensore viene risvegliato il consumo di energia ha un picco di risveglio che poi si abbassa per tornare sul livello di consumo energetico medio

### lifetime sensori e rete

È meglio massimizzaere il lifetime della rete e non del singolo sensore.

I sensori fanno anche da router usando energia ( un po’ come nelle content sensor network)

Se voglio massimizzare il consumo il sensore comunica solo con il sink, e trascura qualunque messaggio che non ha il suo MAC address (nodo selfish). Facendo così non fa da router. Il problema che in questo modo gli altri sensori potrebbero essere costretti a usare più energia per comunicare a distanze più grandi. Quindi potrebbe essere interessante massimizzaere la vita della rete e non del singolo sensore. Ogni sensore fa da router, e tutti durano di più. Load balancing, non c’è nessun sensore che lavora più di altri. Se un sensore finisce l’energia potrebbe diventare un punto di partizione nella rete. E potrebbe esserci un buco nel monitoraggio.

### Scalabilità

Tanti nodi, alta densità.

### Robustezza

I sensori devono resistere a polvere, pioggia, alta temperature. La rete deve prevedere sensori che si scaricano, vengono spostati o vengono rotti. Il sistema deve essere robusto per adattarsi in modo autonomo, mantenendo l’accuratezza di monitoraggio.

### Consenso

Il dato riportato da un singolo sensore non è significativo come l’insieme dei dati raccolti da tanti sensori.

## Deployed vs unplanned

### Deployed

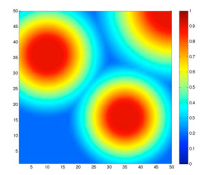
Si studia come è fatta l’area in cui si vogliono piazzare i sensori e si sceglie accuratamente la posizione di ogni sensore. È un problema di max/min, si vogliono massimizzare / minimizzare delle grandezze rispettando dei vincoli.

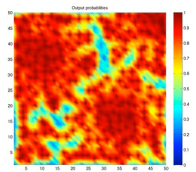
Esempio: massimizzare accuratezza e contemporaneamente minimizzare costi.

Normalmente queste reti sono composte da oggetti piuttosto grandi.

### Unplanned

I sensori sono sparpagliati a caso nell’area interessata

I colori indicano l’interesse di rilevare un certo evento. Rosso indica che è più importante rilevare l’evento in quella zona.

L’immagine indica come sono seminati i sensori.

Ci sono chiazze di sensori più dense che corrispondono alle aree della figura precedente.

Lo sparpagliare non è perfetto perché è importante creare anche dei cammini di comunicazione tra una chiazza e l’altra per fare arrivare i dati al sink.

## Localizzazione e sincronizzazione

Se i sensori sono stati piazzati si conoscono le coordinate, ma se i sensori sono stati sparpagliati devono essere in grado di individuare la loro posizione, assoluta (lat / long) oppure in relazione ad altri sensori.

Il GPS non funziona indoor, in foreste, strade strette e consuma molta batteria.

I sensori possono usare il segnale dei sink che sono dispositivi fissi per capire dove sono. Utilizzando caratteristiche come il tempo di propagazione del segnale.

La posizione è utile per aggregare i dati dei sensori vicini, e per capire in che posizione un certo evento si sta verificando.

Può essere utile tenere sincronizzati gli orologi dei sensori per etichettare i dati anche in modo temporale.

## Naming e routing

Non interessa andare verso un certo indirizzo o sensore, ma andare alla ricerca di un dato o inviare un dato ad un dispositivo sink qualsiasi che può raccogliere i dati.

## Range di comunicazione

### Hop lunghi

i sensori comunicano con il sink, i sensori lontani dal sink esauriranno prima le batterie perché la potenza necessaria di trasmissione varia con il quadrato della distanza. C’è probabilità di interferenza tra i sensori che stanno comunicando contemporaneamente.

### hop corti

i sensori comunicano con altri sensori, i sensori lontani dal sink comunicano comunque a breve distanza. Il problema è che un singolo sensore potrebbe diventare un punto fragile perché magari si trova al centro e riceve messaggi da tutti gli altri sensori.

Ci sono due soluzioni a questo problema:

1. potenza variabile per la trasmissione
2. aggregazione dei dati con ottimizzazione della bandwidth usata

Rispetto agli hop lunghi c’è meno probabilità di interferenza. Gli hop corti sono preferiti nelle WSN.

## Data Aggregation

* **La stessa segnalazione può arrivare da più sensori** → in questo caso si eliminano i duplicati inviando al sink una volta sola l’evento
* **gli eventi possono essere** **aggregati** → Certe volte non è necessario inviare tutti i dati che sono stati letti, alcuni nodi potrebbero fare un’aggregazione dei dati processandoli prima di inviarli al sink. Il tipo di processing dipende dal tipo di rete che si vuole sviluppare (es. un sensore riceve tante misure di temperature, fa media e varianza e inoltra solo quei dati).

Messaggi piccoli vs grandi → i messaggi piccoli sprecano meno energia e hanno maggior possibilità di successo. C’è poca probabilità di sovrapporsi con altri nodi se comunico per un breve intervallo temporale.

I nodi devono essere capaci di aggreggare → application awareness, intelligenza dei nodi per capire la semantica dei dati che sono trasmessi nella rete

## Aspetti secondari WSN

### Database

Raccogliere dati prodotti dai sensori, deve essere distribuito, i sensori devono memorizzare poche informazioni.

### Sicurezza

Il canale è wireless e il segnale può essere recepito da tutti quelli che sono in grado di sentirlo. La soluzione più semplice è cifrare i dati, un’operazione molto costosa in termini di computazione e memoria.

### Attuatori

Possono influenzare il sensing dei sensori mobili.

## Nodi statici vs mobili

**Ferries** → nodi di cui si può controllare il movimento per facilitare la comunicazione o per coprire aree rimaste scoperte o partizionate. I ferries potrebbero anche non cambiare la topologia della rete, ma essere dei magazzini di dati che vengono spostati da un’area all’altra (paradigma store, carry and forward).

Il movimento dei ferries può essere periodico e regolare, controllabile, o libero perché installati su veicoli o animali. Nell’ultimo caso la topologia cambia in modo quasi imprevedibile, c’è una maggiore probabilità di partizione della rete.

## Architetture gerarchiche

### Three-tiered architecture

* **Sensori** → alcuni sono più potenti in termini di calcolo, antenna e disponibilità energetica
* **Microserver** → maggiore capacità di comunicazione, OS tradizionale, più risorse in batteria (Raspberry e simili). Si parla di edge computing, elaborazione ai limiti della rete di sensori
* **Sink** → punto di connessione con internet. Invia i dati o riceve direttive come query di dati, cambio di configurazione dei sensori, ecc...

# Slide 4

## Tassonomia minima

### Hierarchical

Ci sono nodi di tipo diverso, alcuni più potenti. Sono reti più semplici da implementare perché spostano la computazione e il coordinamento su nodi centrali nella rete

### Reverse path forwarding

L’approccio ipotizza che il sink fa delle query che informano la rete di sensori di quello che il sink vuole sapere e quanto spesso. Le query fanno dei cammini che sono ricordati dai nodi, quando un nodo ha una risposta per la query inoltra seguendo il cammino al rovescio. Questi approcci assumono implicitamente che il canale sia full duplex (bidirezionali).

### Cost – field

I sensori sono preprogrammati per riportare il verificarsi di un evento, seguono dei cammini precostituiti.

## Assunzione sul sistema

* sensori distribuiti sul territorio in modo unplanned, i sensori conoscono la loro posizione che però non è usata nell’algoritmo di instradamento ma solo per arricchire il dato rilevato.
* I sensori hanno ID univoci e orologi sincronizzati
* i sensori non sono mobili
* la rete è densa, ci sono path ridondanti, topologia significativamente connessa
* i sensori sono intelligenti, sanno ragionare sui dati
* il canale è wireless, le antenne sono omnidirezionali
* preferiti hop corti

## Directed diffusion

### Assunzioni sui dati

Rientra nella categoria del reverse path forwarding.

Il sink invia delle query chiamate interest. Si dice interest e non query perché il sink annuncia di essere interessato ad un evento, ma non sa se lo riceverà mai. Diverso dalle query che spesso ritornano qualcosa

L’interest è composto da coppie attributo valore:

type = four-legged animal

interval = 20 ms

rect = [-100, 100, 200, 400]

timestamp = 01:20:40

expiresAt = 01:30:40

* **type** → dice al sensore di stare attento ad un tipo di evento. Se un sensore non è capace di rilevare un evento farà solo da relay per i messaggi. Il duty cycle è diverso per i sensori che fanno solo relay
* **interval** → i sensori riconfigurano il loro duty cicle per misurare l’evento con la frequenza definita
* **rect** → area rettangolare in cui cercare l’evento. Le coordinate non sono coordinate assolute di lat e long, ma relativa ad un punto della rete
* **timestamp** → quando è stato fatto l’interest, gli orologi devono essere sincronizzati
* **expiresAt** → quando scade l’interest

### Caratteristiche

**canali bi-direzionali**

**reverse path forwarding**

Gli interest installano informazioni di stato nei sensori che ricordano quali cammini sono stati seguiti dagli interest.

**Reactive routing**

Non vengono mantenute informazioni di instradamento o rapporti di vicinato. Le informazioni di stato vengono costruite solo nel momento in cui il sink inoltra un interest, e i cammini sono mantenuti attivi solo fino a quando questo interest è attivo. Le sorgenti sono i sensori, le destinazioni sono i sink.

**Flooding degli interessi**

L’algoritmo si divide in due fasi. La prima è la diffusione degli interest. Il flooding è la politica di instradamento più banale: la sorgente invia il messaggio a tutti i suoi vicini, i nodi che ricevono il messaggio lo reinoltrano a tutti i suoi vicini escludendo il nodo da cui l’ha ricevuto.

Pila ISO/OSI:

1. Livello fisico
2. Livello data link → mac address / hardware address
3. Livello rete → indirizzo ip
4. livello trasporto → numero di porta

L’hardware address può essere unicast o broadcast. L’informazione ad alto livello della interest che è destinata solo ad alcuni sensori, deve essere instradato usando l’hardware address. In queste reti wireless non si può fare il flooding classico diffondendo il segnale a tutti i sensori. Questo perché il segnale non è detto che venga rilevato da tutti i nodi a causa del duty cycle.

Non è obbligatorio fare tutte le comunicazione broadcast, quando le notifiche tornano indietro si indirizza a livello due unicast.

Il cammino di ritorno non è unico ma si memorizzano tutti i cammini possibili, e poi si usa il migliore per il ritorno.

**Aggregazione dati**

In direct diffusion i sink possono essere multipli, possono diffondere interessi diversi ma c’è la possbilità che sink differenti diffondano lo stesso interest.

I sensori capiscono la semantica dei dati fanno aggregazione filtrando i duplicati o adattando la frequenza con cui i dati vengono mandati verso i sink.

Crying baby → se due sink diversi chiedono lo stesso evento a frequenza diversa i sensori devono adattare il duty cycle per quello a frequenza maggiore. I sensori che hanno visto solo l’interest con frequenza più bassa però non vogliono riceve l’interest a frequenza più alta. A questi sensori può arriva una notifica ogni *n* notifiche, oppure si fa la media di *n* notifiche e la si invia al sensore.

### Diffusione interesse

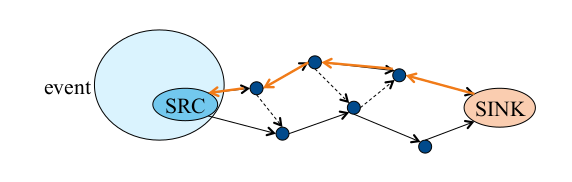
Il sink periodicamente guarda nella propria cache gli interessi attivi, rimuove quelli scaduti e invia ogni interesse attivo. Dato che il broadcast non è completo a causa del duty cycle dei sensori, è fondamentale diffondere l’interest periodicamente in broadcast. Il primo broadcast usa un alto interval (ogni quanto un sensore invia il dato) perché il sink non sa se l’evento a cui è interessato succederà mai. Infatti l’azione di risveglio del sensore è molto costosa. Nel momento in cui l’evento si verifica e un sensore comincia a inviare notifiche al sink, allora il sink cambia l’intervallo di monitoraggio dei sensori.

Ogni nodo ricorda l’interest nella cache, ogni entry è formata da:

* interest
* timestamp → da quando l’interest è attivo
* serie di gradienti:
  + neighbor ID (mac address)
  + data rate → rate a cui devo mandare le notifiche al vicino
  + duration → ricavata da timestamp e expiration.

I gradienti sono i nodi vicini a cui devo reinoltrare un evento. Posso avere più vicini e inviare i dati ad ogni vicino con data rate diverso

Gli interest duplicati sono scartati mantenendo solo l’ultimo. Nel gradiente non c’è scritto l’identificatore e la distanza dal sink. Un nodo che riceve un evento sa solo quali sono i next hop, non sa nemmeno se il next hop è un sink.



A destra c‘è il sink, il cerchio azzurro è la zona in cui il sink vuole monitorare l’evento. SRC è la sorgente dell’evento

1. il sink manda l’interest la prima volta ai propri vicini
2. i primi 2 nodi capiscono che non possono acquisire quella misurazione, si ricordano però come gradiente il sink, senza sapere se è davvero il sink o semplicemente un nodo intermedio
3. Il nodo in alto fa broadcast
4. i due sensori a sinistra ricevono l’interest, e ricordano come next hop il sensore precedente
5. anche il sink riceve il broadcast dell’interest, è implicitamente un ACK per il sink. In più si vogliono supportare sink multipli. Anche il sink arancione ha una cache e si ricorda che se riceve un interest lo rimanda al sensore.
6. Anche il sensore in basso a sinistra del sink fa la stessa cosa di quello in alto
7. il nodo centrale avrà due gradienti, ma lo stesso interest. Non fa di nuovo flooding perché è un duplicato.
8. I nodi facendo broadcast costruiscono anche dei gradienti all’indietro, sembrano controintuitivi ma in realtà servono per diffondere i dati anche a sink diversi.

### Routing eventi

I gradienti indicano quali vicini sono interessati ad un evento, possono essere sink o nodi più vicini ai sink. C’è ridondanza di gradienti e di path per essere in grado di riparare cammini rotti verso il sink.

Quando si verifica un evento per cui esiste un interesse attivo:

* il nodo SRC manda l’evento a tutti i vicini di cui ha un gradiente, è una trasmissione broadcast con indirizzi unicast
* il sensore più vicino alla SRC riceve l’evento e lo rimanda al nodo alla sua destra e a quello in basso.
* La freccia è tratteggiata perché il nodo in basso ha già ricevuto la notifica dell’evento da SRC. Scarta quindi l’evento, sa che il cammino verso l’alto è più lento
* così via per tutte le frecce tratteggiate
* il sink riceve eventi da entrambi i cammini, uno dei due sarà migliore perché più veloce o con meno interferenze.

### Varianti per la diffusione interest

* **Flooding →** versione base
* **Routing geografico →** Al posto che mandare in flooding l’interesse lo mando in direzione dei sensori più vicini all’area target per l’interest
* **direzione cached da risposte precedenti →** se il sink in passato aveva diffuso lo stesso interesse, e quella volta aveva scoperto che le notifiche gli arrivavano solo da un sottoinsieme di nodi, allora il sink può ricordare questo sottoinsieme e in futuro diffondere l’interest solo ai vicini da cui ha ricevuto risposte la volta precedente

### Rinforzo

Manda un interest unicast soltanto al vicino da cui riceve meglio le notifiche, con un interval più basso e quindi una frequenza più bassa. Il nodo vicino al sink sceglierà il proprio vicino migliore e manda indietro il nuovo interest del sink. Si continua fino alla sorgente, rinforzando solo il cammino migliore.

Si può fare anche il contrario, dal cammino in cui riceve meno bene può fare 3 cose:

1. mandare interest sul cammino peggiore con intervallo più alto. Si mantiene comunque il cammino peggiore perché se quello migliore si rompe c’è quello peggiore di backup