MSA - Mobility Management in Internet: no infrastructure networks

- Soluzioni viste fino ad ora assumono l'esistenza di un'infrastruttura di appoggio che offre supporto per la risoluzione del problema della mobilità
- Un'infrastruttura può non esistere nativamente per:
  - contesto nuovo
  - infrastruttura distrutta
  - diritti di accesso di varia natura
- Soluzione utile in ambito di reti cellulari per cercare di risolvere il problema causato dai picchi improvvisi di traffico che si possono generare all'interno di una cella. Se le coomunicazioni riguardano nodi ceh si trovano nella stessa cella o celle vicini, i nodi potrebbero provare a parlarsi direttamente tra di loro

Manet

- Mobile ad-hoc networks, si assume che sia altamente probabile che una connessione end-to-end tra due nodi esista. La configurazione della connesione può cambiare nel tempo, ma assumiamo che mediamente sia altamente probabile ceh ci sia. E' uno scenario in cui ci sono vincoli di banda e di energia, visto ceh parliamo tipcicamente di nodi mobili alimentati da batteria e dunque è necessario cercare di risparmiare più energia possibile.
- Reti che possiamo vedere come di due tipi:
  - **Reti di transito:** i punti di comunicazione finali di una comunicazione stanno fuori dalla rete MANET
  - **Stub network:** il punto di partenza e terminazione della comunicazione appartiene alla rete MANET. Assumeremo di essere in questa situazione
- Caratteristica fondamentale è il *dinamismo*:
  - Non c'è una topologia fissa della rete. Se prendo una qualsiasi coppia di nodi es osservo lo stato della connessione nel corso del tempo e la qualità della connessione, questa varia in maniera più o meno amplia. La qualità della comunicazione potrebbe non essere la stessa nelle due direzioni, ma potrebbe anche mancare del tutto. Il fatto che un nodo riesca a parlare con un altro non significa necessariamente l'inverso. Un meccanismo di instradamento per queste reti dorebbe essere in grado di gestire situazioni di asimmetria nei link tra due nodi qualunque
- Algoritmi di routing tradizionali hanno tempi di convergenza lunghi, non sono adatti ad uno scenario molto dinamico come quello della rete MANET, sono inefficienti in quanto la quantità di messaggi scambiati è molto grande o anche perché non sono in grado di gestire la presenza di link asimmetrici
- *Tipi di approccio:*
  - **Gerarchici:** si definisce una gerarchia tra i nodi del sistema, quindi alcuni alla base della gerarchia sono liberati dal carico, dall'incombenza di risolvere l'instradamento, e solo un sottoinsieme dei nodi ha questo compito. C'è il

problema aggiuntivo di come selezionare i nodi che dovranno contribuire a risolvere il problema
    - **Non gerarchici:** tutti i nodi del sistema sono alla pari, si fanno carico di dare il loro contributo all'inoltro dei messaggi
    - **Reattivi e proattivi**
- **Valutazione:**
    - **Misure esterne:** ritardo che subiscono i pacchetti, throughpu
    - **Misure interne:** prezzo, intermini di overhead, da pagare per garantire le misure esterne

OLSR (Optimized Link State Routing)

- Approccio pro attivo, si vuole tenere aggiornata la posizione di un nodo mobile, rischiando di tenere aggiornate informazioni che magari non verranno utilizzate
- Il risparmio di energia è vitale nelle reti MANET, come posso risolvere il problema dello sforzo inutile? Lo si fa agendo in due direzioni:
    - Cercando di ridurre la quantità di messaggi di controllo, di aggiornamento della posizione dei nodi, che vengono fatti circolare nella rete
    - Cercando di ridurre la dimensione che hanno i messaggi, in modo da minimizzare l'impatto del traffico di controllo sui consumi energetici del sistema
- **Multipoint Relay Set:** dato un nodo A, questo set è l'insieme minimo di vicini che stanno a distanza 1 da A, ed attraverso i cui A è in grado di inviare un messaggio a tutti i suoi vicini di distanza 2. Solo i nodi che fanno parte del *multipoint relay set* di qualche nodo saranno quelli coinvolti nello scambio di messaggi
    - **Messaggi di hello:** inviati da tutti i nodi, ogni nodo in questo pacchetto mmette la lista dei nodi che stanno a distanza 1 da lui. Di questi nodi specifica anche queal'è il sottoinsieme che costituisce il suo multipoint relay set. Se ogni nodo manda un pacchetto di hello concepito in questo modo, ascoltando i messaggi di tutti i vicini è possibile calcolare il *multipoint relay set*
    - **Messaggi di Topology Control (TC):** inviati solo dai nodi che fanno parte del *multiplayer relay set* e contengono la lista di quali sono i nodi per cui il nodo mittente del pacchetto è membro del loro MPR. I nodi di questo tipo sono sostanzialmente i router del sistema
- **Algoritmo:**
    - Ogni nodo determina quali sono i nodi a distanza 1, a distanza 2 e calcola il suo MPR, costruisce il suo messaggio di hello e lo invia periodicamente
    - Se un nodo riceve un pacchetto di tipo hello, si chiede se fa parte del set MPR del nodo mittente. Se scopro di fare parte del set MPR di qualche nodo, mi devo fare carico di inviare un pacchetto TC, in quanto scopro di essere eletto come router per il nodo che ha inviato il pacchetto
    - Se un nodo qualunque riceve un pacchetto di tipo TC, si chiede se fa parte del set MPR di qualcuno. Se la risposta è si, lo devo inoltrare a qualcun altro. In questo modo solo i nodi che fanno parte dell'insieme MPR ricevono questi

aggiornamenti sulla topologia della rete, riducendo la quantità e la dimensione dei messaggi che circolano sulla rete

DSR (Dinamic Source Routing)

- Protocollo di ripo reattivo
- Si basa sue due algoritmi:
  - **Scoperta del cammino:** nel momento in cui arriva un messaggio dal livello applicativo destinato verso una certa destinazione
  - **Manutenzione del cammino:** essendo un sistema dinamico, quest'informazione è soggetta ad obsolescenza, quindi è necessario fare un lavoro di manutenzione e verificare fino a che punto l'informazione raccolta nel passato più o meno recente sia ancora effettiva o se va in qualche modo aggiornata
  - **Source routing:** Non c'è in nessun punto della rete una tabella di instradamento. Ogni pacchetto che viaggia all'interno della rete ha scritto dentro di se qual'è il percorso che il pacchetto deve seguire
- **Strutture dati necessarie:**
  - **Cache dei percorsi:** mantiene informazioni sui percorsi scoperti in precedenza verso un certo numero di destinazioni
  - **Tabella di richieste:** mantiene le richieste in corso di risoluzione, i essaggi di *route request* che sono stati generati dal nodo perhé qalcuno a livello applicativo ha prodotto un messaggio che deve essere inviato verso una certa destinazione
  - **Buffer dei messaggi:** in attesa di essere trasmessi, perché è in corso di risoluzione il processo di scoperta del cammino finale
  - **Buffer di ritrasmissione:** pacchetti in attesa di ricezione dell'ACK di ritorno
- **Algoritmo scoperta del cammino:**
  - Se un nodo deve inviare un pacchetto verso qualcuno che non sta a distanza 1, invia un messaggio di broadcast di tipo *ROUTE REQUEST* con un ID associato, dove viene specificato il destinatario finale del messaggio
  - I messaggi inviati in broadcast iniziano a propagarsi per la rete
  - Se il nodo che riceve il pacchetto si accorge, tramite l'ID, che ha già gestito quel pacchetto, allora cestina il messaggio per evitare la generazione di cicli nella rete
  - Altrimenti il nodo aggiunge all'header del pacchetto il proprio identificativo e manda il messaggio in broadcast
  - Quando il messaggio arriva al nodo inale, grazie al meccanismo precedente quest'ultimo conterrà l'elenco di tutti i nodi che sono stati attraversati per raggiungerlo
  - Il nodo destinatario genera un messaggio di ROUTE REPLY che, nell'ipotesi di link bidirezionali, la strada che il messaggi odeve eseguire è l'inverso di quella scritta nel pacchetto ricevuto

- Se non è possibile fare questa ipotesi, allora il nodo di destinazione invierà il messaggio di ROUTE REPLY utilizzando il meccanismo visto prima
- **Ottimizzazioni possibili:**
  - Conoscendo il diametro della rete è possibile limitare il numero massimo di passi
  - Man mano che lo scambio di messaggi procede, l'informazione comincia a circolare e ad essere acquisita, anche se in modo parziale, da tutti i nodi coinvoti nello scambio. Se ad esempio A vuole parlare con B ed un nodo intermedio riceve il emssaggio di *ROUTE REQUEST* che da se stesso porta a B, piuttosto che seguire il meccanismo standard può inviare lui il messaggio di *ROUTE REPLY* riducendo la latenza del processo di scoperta

- **Manutenzione della route:**
  - Il mittente che ha ricevuto l'informazione sul percorso che deve essere seguito per arrivare a destinazione, prende il pacchetto in attesa ed aggancia un header che contiene il percorso da seguire e lo inolttra
  - Può capitare che questa configurazione sia diventata nel tempo non più valida
  - La validità delle informazioni può essere valutata in vari modi:
    - Guardando se dal livello sottostante attiva un ACK se il protocollo lo prevede
    - Se stiamo usando un protocollo link layer che non prevede ACK, posso ascoltare il messaggio che ho inviato e vedo se viene reinoltrato dal mio vicino a distanza 1
    - Posso prevedere la richiesta di un ACK dal nodo a cui ho indirizzato il pacchetto
  - A causa di un qualunque di questi motivi, chi ha inoltrato il pacchetto capisce che c'è un problema e posso:
    - Informare il mittente originario per indicare la non validità del percorso indicato nel pacchetto
    - Farmi carico in prima persona ed attivare localmente la procedura di riparazione, indivuando il nuovo percorso da seguire **Conclusioni:** si tratta di un protocollo semplice, la cui efficacia dipende da vari fattori tra i quali gioa un ruolo preponderante il livello di dinamismo della rete. Se la topologia cambia lentamente riesco a capitarlizzare molto lo sforzo atto in precedenza e quindi l'informazione che tengo nella cache saranno validi per intervalli di tempo lunghi.Se la rete è molto dinamica, il tipo di protocollo cambia, rischia di divetare un po troppo oneroso

Considerazioni sui due algoritmi