Politecnico di Milano

Facoltà di Ingegneria

Corso di Laurea in Ingegneria delle Telecomunicazioni

Dipartimento di Elettronica e Informazione

Valutazione delle prestazioni e ottimizzazione di un protocollo di routing sicuro per reti mobili ad hoc

Relatore: Chiar.mo Prof. **Antonio Capone**

Correlatori: Dott. Ing. **Davide Cerri**

Dott. Ing. **Alessandro Ghioni**

Tesina di Laurea di:

Alex Mufatti

Matricola 638664

Anno Accademico 2005/2006

Ai miei genitori.

Ringraziamenti

Eccoci giunti alla fine di questo percorso; non è stato facile, ma niente che lasci il segno lo è.

Devo ringraziare tantissime persone con l'aiuto delle quali sono potuto arrivare fino a qui, ma non voglio fare un elenco di nomi in cui sicuramente dimenticherei qualcuno che avrebbe meritato di esserci. In fondo penso che chi tengo veramente a ringraziare sappia già di appartenere a quell'elenco senza aver bisogno di leggere il proprio nome su questa pagina.

Vorrei però ringraziare i miei genitori che mi hanno accompagnato e sostenuto fino a qui: "se la laurea è un traguardo importante e non facile da raggiungere quello che avete sempre fatto voi per me lo è ancora di più e non basteranno certo queste parole per ringraziarvi".

Non avrei potuto arrivare fino in fondo senza i miei Amici: quelli che ci sono da sempre, quelli incontrati al poli e quelli conosciuti al cefriel; i ragazzi del calcetto, i tesisti e i masterandi: "Spero davvero di non perdervi mai e se dovesse succedere, mio malgrado, di incontrarvi ancora".

Un grazie con tutto l'affetto possibile a chi mi è stata vicino in questo lungo viaggio e ora non lo è più: "Dicono che non si scordi mai . . . penso abbiano ragione."

In fine, un ringraziamento di cuore davvero speciale a chi mi è accanto ora e lo sarà in futuro: ". . . il mio percorso è finito, il nostro invece è appena iniziato . . . "

Sommario

Le reti mobili ad hoc (Manet) sono una tipologia di reti wireless di nuova generazione dove non vi è un'infrastruttura pre-costituita e dove tutti i nodi svolgono le funzioni di host e di router. Risulta dunque necessaria la definizione di protocolli di routing appositi, adatti alle caratteristiche di queste reti. Nello stesso tempo le prerogative delle Manet le portano a essere soggette più di altre ad attacchi da parte di nodi *ostili*. Per questo si deve porre attenzione anche al problema della messa in sicurezza del protocollo utilizzato senza però dimenticarsi delle prestazioni, che sono fondamentali per le reti Manet in quanto le funzioni di routing sono svolte indifferentemente da tutti i nodi della rete. Da queste considerazioni deriva il lavoro svolto, il cui scopo è l'ottimizzazione delle prestazioni di un protocollo di routing sicuro per le Manet.

Dopo una breve introduzione sulle Manet (capitolo 1) e sui protocolli di routing presenti in letteratura (capitolo 2) si passerà ad analizzare il problema della sicurezza e i protocolli sicuri (capitolo 3). Ci si concentrerà poi sul protocollo AODV e su una sua versione sicura, SAODV, descrivendone il funzionamento; si vedranno dunque i problemi di prestazioni di quest'ultimo rispetto alla versione non sicura per poi passare alla proposta di due possibili ottimizzazioni volte a diminuire questo gap (sezione 4.4). Si valuterà quindi l'effettiva efficacia delle ottimizzazioni proposte tramite uno studio simulativo condotto su diversi scenari applicativi (paragrafo 4.5). Si arriverà poi a trarre le conclusioni del lavoro svolto dalle quali sono emersi anche possibili spunti per lavori ulteriori (capitolo 5).

Indice

Capitolo 1

Introduzione

Una rete wireless ad-hoc è costituita da un insieme di terminali autonomi, detti nodi, che comunicano tra di loro attraverso un'interfaccia radio formando una rete *multihop*[1]1.1 e mantenendo la connettività in maniera completamente distribuita. Siccome la comunicazione avviene attraverso canale radio, i nodi devono far fronte ai tipici problemi che questo comporta: rumore, fading, interferenza, limitata banda a disposizione rispetto al comune canale cablato.

Le caratteristiche principali delle reti ad-hoc sono quelle di rendere distribuite su tutti i nodi le funzionalità fondamentali della rete stessa, per esempio la risoluzione dei nomi o l'instradamento dei pacchetti, e di essere reti fortemente collaborative. Durante una comunicazione multihop, infatti, un nodo sorgente invia i dati ad un nodo destinazione servendosi della collaborazione degli altri nodi lungo il percorso, detti *nodi intermedi*, che usano parte delle proprie risorse per portare a termine tale trasmissione. Ogni nodo in questo modo ricopre una doppia funzione: quella di host, agendo da nodo sorgente o destinazione di una comunicazione, e quella di router, agendo da nodo intermedio.

La topologia di rete è tipicamente dinamica in quanto la connettività tra i nodi può variare a causa dell'uscita di un nodo dalla rete o, viceversa, dell'ingresso di

[1]si definisce multihop la comunicazione tra due nodi che non sono in visibilità diretta, in contrapposizione alla comunicazione *singlehop* o *diretta* che si instaura tra due nodi in visibilità radio.

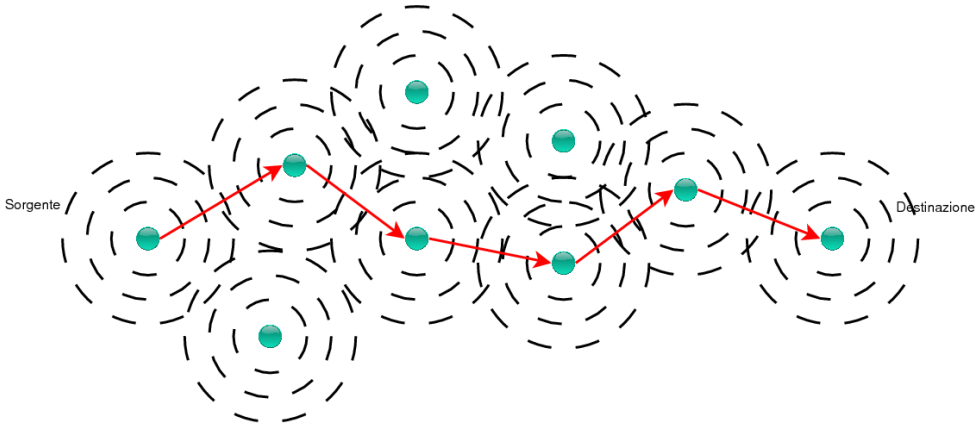


Figura 1.1: Comunicazione multihop

un nuovo terminale mobile e, ancora, dalla possibile mobilità degli stessi. Vi è dunque la necessità di un efficiente meccanismo di routing per consentire ai nodi di instaurare canali di comunicazione multihop, composti anche da molti link, in modo, se possibile, da non richiedere un eccessivo consumo di risorse (banda, potenza di calcolo).

Le tipologie di reti ad-hoc sono principalmente due: le reti di sensori e le manet.

1.1 Manet

Negli ultimi anni si è andata delineando una nuova generazione di sistemi di comunicazione wireless. Questi sistemi sono caratterizzati da un'elevata mobilità e nascono dall'esigenza di un inserimento rapido e autonomo in rete di terminali indipendenti e dalla necessità di rendere la rete indipendente da qualsiasi infrastruttura preesistente. Questa nuova tipologia di rete viene definita *MANet*[2].

I primi studi sulle manet risalgono agli anni '70-'80 in ambito militare, mentre nel 1997 è stato istituito un gruppo di lavoro in IETF[3], il *mobile ad-hoc network working group* (manet WG). Il manet WG fa parte dell'area routing e si occupa

[2]Mobile Ad-hoc Network

[3]Internet Engeneering Task Force

principalmente della definizione di uno o più protocolli di routing adatti a divenire standard per questa tipologia di rete.

Lo spazio di applicazioni delle manet varia da reti di piccole dimensioni quasi statiche in cui pochi nodi entrano ed escono dalla rete, a reti su larga scala caratterizzate da un'alta mobilità dei nodi. Gli scenari di utilizzo sono quelli in cui sia sconveniente, impossibile o scomodo creare un'infrastruttura di rete fissa. Esempi significativi di utilizzo di queste strutture di rete possono essere l'instaurazione di una rete robusta, efficiente e dinamica sui luoghi di catastrofi, in scenari di emergenza o di operazioni militari in campo ostile. Altri scenari possibili sono quelli di conferenze, eventi o riunioni nelle quali i partecipanti vogliano instaurare in modo semplice e spontaneo una rete per condividere informazioni o scambiare documenti. Come ben si comprende, negli scenari citati non è possibile basarsi su un'unità centrale o un'infrastruttura precostituita per gestire la connessione tra i nodi. È inoltre necessario che questi possano entrare e uscire liberamente e in modo rapido dalla rete senza rendere necessario nessun intervento speciale sulla rete stessa e senza precluderne nessuna funzionalità essenziale. A causa di queste caratteristiche la definizione dei protocolli di rete è un problema complesso. In particolare, l'estrema variabilità della topologia della rete rende il routing un elemento di cruciale importanza nelle manet.

1.2 Le reti di sensori

Una *rete wireless ad-hoc di sensori* consiste in un certo numero di sensori disposti su di un'area geografica. Ogni sensore possiede capacità di comunicazione wireless e un certo grado di intelligenza per l'elaborazione dei segnali e la trasmissione dei dati. Alcuni esempi di reti di sensori possono essere:

- reti militari per ottenere la maggior parte di informazioni possibili sui movimenti nemici, esplosioni e altri fenomeni di interesse;

- reti per individuare e caratterizzare contaminazioni o attacchi chimici, biologici, nucleari, esplosivi;
- reti di sensori di traffico per monitorare il traffico su autostrade o sulle vie di comunicazioni maggiormente congestionate;
- sensori per la sorveglianza di negozi, case, etc.

Gli scopi di una rete di sensori sono altamente dipendenti dall'applicazione della stessa, ma solitamente possono essere catalogati come uno tra:

- determinare il valore di un parametro in una determinata posizione;
- rilevare l'occorrenza di un determinato fenomeno di interesse e rilevarne alcuni parametri;
- classificare un oggetto rilevato;
- tenere traccia degli spostamenti di un oggetto.

Da questo si evince che uno degli obiettivi principali delle reti di sensori è quello di far giungere il dato richiesto all'utente. In certi casi vi sono anche dei limiti di tempo abbastanza stretti da rispettare in questa comunicazione.

A differenza delle manet, i nodi di una rete di sensori sono pressoché statici e quindi la topologia non è così variabile come nel caso precedente.

1.3 Sicurezza nelle Manet

Una rete wireless è per natura maggiormente vulnerabile rispetto ad una normale rete cablata, in quanto il canale di comunicazione radio offre maggiori opportunità ad un eventuale attaccante. Il motivo principale è che il campo di copertura non può essere completamente circoscritto; infatti le antenne utilizzate per la trasmissione radio sono solitamente omnidirezionali ed è quindi impossibile limitare la ricezione di una trasmissione al solo destinatario quando vi siano

diversi riceventi nel raggio di copertura. Inoltre nel caso di reti indoor è altrettanto impossibile limitare la copertura radio solo all'interno dell'edificio interessato; un eventuale attaccante può quindi avere accesso al canale anche non trovandosi nell'area di utilizzo lecito della rete.

1.3.1 Requisiti di sicurezza

In una rete mobile ad-hoc oltre ai requisiti di sicurezza tradizionali, come autenticazione e integrità dell'informazione, si aggiungono requisiti dettati dal fatto che la rete è collaborativa. I requisiti di sicurezza di una manet possono essere dunque così riassunti:

Disponibilità - In una rete wireless è molto facile portare a termine attacchi *DoS* (*Denial of Service*) immettendo una grande quantità di pacchetti non richiesti in rete e rendendo così inutilizzabile il canale. È altrettanto facile disturbare la trasmissione a livello fisico con forti segnali radio. Inoltre, a causa della limitata autonomia dei dispositivi tipicamente utilizzati in queste reti, è possibile portare un attacco di tipo *battery exhaustion*. Questi sono tutti attacchi che minano la disponibilità della rete.

Autenticazione - Poiché in una rete wireless chiunque si trovi nel raggio di copertura radio può inviare messaggi e riceverne, è essenziale che, tramite meccanismi di autenticazione, le due parti coinvolte in una comunicazione siano certe delle rispettive identità prima di iniziare qualsiasi scambio di dati.

Autorizzazione - L'autorizzazione è il passo successivo all'autenticazione. Infatti, avvenuta l'autenticazione, viene concesso al nodo di usufruire unicamente dei servizi per i quali è stato autorizzato.

Confidenzialità - Poiché la comunicazione si svolge su canale radio e, come detto in precedenza, è impossibile limitare la copertura al solo destinatario,

occorre, a volte[4], cifrare tutto il traffico per non permettere ad un nodo in ascolto di ottenere informazioni intercettando i pacchetti in arrivo.

Integrità - A causa della tipica configurazione collaborativa delle Manet è essenziale per il ricevente essere sicuro che, in una comunicazione multihop, il messaggio arrivato sia il medesimo spedito dalla sorgente; in altre parole si vuole evitare che nodi avversari possano modificare il contenuto dei messaggi lungo il tragitto. Per questo alcuni protocolli di rete possono allegare al messaggio dei *MAC* (Message Autentication Code). Una volta che il pacchetto è giunto a destinazione il nodo può verificare l'integrità del contenuto tramite appositi algoritmi applicati al messaggio stesso.

Non ripudio - Il non ripudio è la caratteristica per la quale un nodo non può negare di aver generato un pacchetto. Grazie a questo requisito si possono individuare i nodi colpevoli di un tentativo di attacco DoS e escluderli dalla rete. Si possono inoltre identificare nodi mal funzionanti che immettono in rete messaggi inconsistenti.

Privacy e anonimato - In una rete wireless, a differenza di una rete cablata, è importante, in certi scenari di utilizzo, mantenere privata la posizione dei nodi all'interno della rete anche per non facilitare l'attacco da parte di un nodo avversario. Infatti, conoscendola, un nodo avversario potrebbe facilmente immettersi lungo il percorso dei pacchetti per effettuare un tentativo di attacco.

1.3.2 Attacchi alle Manet

In una Manet si creano collegamenti radio, potenzialmente multihop, tra nodi. Questo compito è assolto principalmente da protocolli di livello due (*link layer*), che garantiscono la connettività a un hop, e protocolli di livello tre (*network layer*), che estendono questa connettività a percorsi multihop. Essendo l'ambiente

[4]quando la comunicazione contenga dati sensibili

completamente distribuito la cooperazione tra i nodi è strettamente necessaria per il corretto funzionamento della rete. Nella prima fase di sviluppo dei protocolli distribuiti per le manet si è dunque assunto che tutti i nodi fossero collaborativi. Quest'assunzione è ovviamente, e sfortunatamente, errata in un ambiente *ostile*. Poiché si è assunta la perfetta cooperazione dei nodi ma non la si è in realtà garantita, un attaccante può facilmente disturbare le normali operazioni di rete violando le specifiche dei protocolli.

Le principali funzioni svolte dai protocolli di livello rete nelle manet sono il *routing ad-hoc* e l'instradamento dei pacchetti dati; questi interagiscono tra loro per portare a compimento il trasporto dei pacchetti dalla sorgente alla destinazione. Tramite il protocollo di routing ad-hoc si scambiano messaggi tra i nodi volti a mantenere informazioni sullo stato delle connessioni e ad instaurarne di nuove. Basandosi su queste rotte i pacchetti vengono istradati attraverso i nodi intermedi fino a compiere l'intero percorso end-to-end. Entrambe queste operazioni (routing e instradamento) sono soggette ad attacchi da parte di nodi ostili che possono portare a varie disfunzioni nel livello rete. Gli attacchi al protocollo di routing ah hoc verranno trattati in dettaglio nel capitolo 3.1.

Gli attacchi alle operazioni di instradamento dei pacchetti dati sono volti a far in modo che i pacchetti vengano immessi in rete in maniera inconsistente rispetto alle informazioni di routing presenti nei nodi. Un attaccante che si venisse a trovare lungo una rotta utilizzata potrebbe scartare dei pacchetti destinati ad essere inoltrati, modificarne il contenuto o, anche, duplicare dei pacchetti che erano già stati inoltrati. Un altro tipo di attacco che può essere effettuato è, come accennato in precedenza, il DoS; lo scopo di questo attacco è quello di saturare le code dei nodi, e perciò la rete, con pacchetti fasulli bloccando così l'elaborazione dei pacchetti legittimi. In questo modo si ha anche un eccessivo consumo delle già ridotte risorse di rete.

Per quanto riguarda gli attacchi che è possibili portare al livello due recenti ricerche hanno evidenziato delle vulnerabilità dei protocolli, specialmente del protocollo MAC IEEE 802.11[1], lo standard di fatto per le Manet. È ben noto che

il *WEP*[5] del protocollo 802.11 è vulnerabile a vari tipi di attacchi causati dall'uso improprio delle primitive crittografiche come esposto in [2]. Questo protocollo è anch'esso soggetto ad attacchi di tipo DoS diretti ai meccanismi di contesa e prenotazione del canale wireless. L'attaccante può modificare il proprio meccanismo di *backoff* per impedire l'accesso al canale da parte dei suoi vicini [3, 4]. In questo protocollo il meccanismo di scelta del nodo destinato a trasmettere favorisce sempre l'ultimo vincitore rispetto agli altri contendenti, così un nodo in continua trasmissione può ottenere sempre l'uso del canale e mantenere gli altri nodi in attesa perenne. Questo attacco può provocare reazioni a catena nei livelli superiori in protocolli che utilizzano un meccanismo di prevenzione della congestione (per esempio le finestre del TCP). Un'altra vulnerabilità del protocollo 802.11 è dovuta al campo *NAV* della trama per la richiesta di trasmissione che indica la durata della prenotazione della risorsa. Un vicino avversario (o lo stesso mittente o destinatario) possono trascurare il contenuto del campo e introdurre intenzionalmente un bit errato nella trama della vittima attraverso un'interferenza wireless. Il frame così corrotto deve essere scartato dal ricevente dopo la rivelazione dell'errore. Questo tipo di attacco può essere considerato anch'esso di tipo DoS.

Anche a livello uno, *physical layer*, possono essere portati attacchi volti a minare la comunicazione tra i nodi di una manet. L'attacco più semplice è quello in cui un nodo nemico irradia con segnali radio di grande potenza[6] [5] uno o più nodi di una rete, rendendo in questo modo impossibile la comunicazione da e verso questi utenti, nonché tutte le comunicazioni che hanno questi nodi come intermedi. Purtroppo non esistono strategie difensive nei confronti di attacchi di questo genere.

1.3.3 Problemi nel definire un meccanismo di sicurezza

Una delle vulnerabilità fondamentali delle Manet sta nella loro architettura *peer to peer* aperta. Infatti, a differenza delle reti cablate che hanno router de-

[5] Wired Equivalent Privacy

[6] questi segali vengono detti di *jamming*

dicati, ogni nodo in un manet svolge la funzione di router inoltrando pacchetti per altri nodi. Inoltre il canale radio è accessibile sia a utenti legittimi che malintenzionati. Come conseguenza non vi è una chiara linea di difesa che può essere intrapresa dal punto di vista della progettazione per rendere sicura la rete. Infatti la distinzione tra rete interna e mondo esterno non è ben definita e non vi è un punto preciso in cui si possa posizionare un meccanismo di difesa specifico. Oltre a questo i dispositivi mobili, e perciò i dati in essi contenuti, sono soggetti alla compromissione e al furto vero e proprio, specialmente se si parla di dispositivi di basso profilo con protezioni hardware e software deboli; di conseguenza degli attaccanti potrebbero penetrare all'interno della rete per mezzo di questi dispositivi che restano l'anello debole della catena di sicurezza.

D'altra parte, la ristrettezza delle risorse nelle Manet costituisce anch'essa un problema non trascurabile nella progettazione dei meccanismi di sicurezza. Il canale radio dispone di una banda limitata e condivisa tra tutti i dispositivi e non è adatto a meccanismi di sicurezza che richiedano un grosso scambio di dati. La potenza di calcolo dei nodi e anch'essa limitata; si pensi a dispositivi come PDA che non riescono a compiere agevolmente operazioni ad alto costo computazionale come il calcolo di firme o verifiche tramite crittografia asimmetrica. Poiché i dispositivi mobili sono tipicamente alimentati a batteria, anche le risorse in termini di energia e autonomia possono essere limitate.

Per la progettazione dei meccanismi di sicurezza si deve tenere conto anche che il tipo di comunicazione wireless e l'alta mobilità presente nelle Manet evidenziano una dinamica molto più elevata se messa a confronto con quella delle reti cablate. Inoltre il canale radio è soggetto a interferenza ed errori e mostra caratteristiche di banda e ritardo variabili nel tempo. Nonostante queste caratteristiche dinamiche un utente può, mentre si sposta da un capo all'altro della rete, richiedere in qualsiasi momento e in ogni punto i requisiti di sicurezza.

Queste caratteristiche proprie delle manet portano alla definizione di una soluzione per la sicurezza che consegua sia una protezione globale che buone prestazioni di rete.

- Per prima cosa il meccanismo di sicurezza deve essere dislocato su molte individualità. La strategia per la sicurezza adottata da ognuno deve essere proporzionata alle sue limitate risorse in termini di capacità computazionali, memoria, capacità di comunicazione ed energia disponibile.

- In secondo luogo un meccanismo di protezione, per essere completo, deve essere necessariamente dislocato su diversi livelli dello stack protocollare in modo che ogni livello contribuisca con una linea difensiva. Non è possibile ottenere una protezione totale agendo soltanto su di un livello.

- Terzo, la soluzione per la sicurezza della rete deve potere difendersi sia da agenti esterni che lanciano attacchi alla Manet sia da nodi interni alla rete che siano stati compromessi da un attaccante. In questo secondo caso la modalità di difesa deve essere differente, poiché l'attaccante potrebbe essere venuto a conoscenza di informazioni a cui un agente esterno non poteva aver accesso.

- Quarto, il meccanismo di sicurezza deve coprire tutti gli aspetti di prevenzione, rilevamento e reazione che lavorano all'unisono per garantire il non collasso del sistema.

- Per ultimo, ma non per questo meno importante, la soluzione deve essere applicabile alle caratteristiche estremamente dinamiche e di limitata disponibilità di risorse tipiche delle manet.

Capitolo 2

Routing nelle manet

In questo capitolo verranno esposte le problematiche specifiche del routing nelle reti ad-hoc nonché gli attacchi che uno o più nodi nemici possono portare a termine nei riguardi del protocollo di routing. Verranno inoltre descritti i protocolli di routing in fase di standardizzazione dall'IETF e alcuni di quelli ancora in fase di studio. Ci si soffermerà poi nella descrizione del protocollo AODV che è quello di maggior interesse nell'ottica del lavoro svolto.

2.1 Caratteristiche del routing nelle Manet

In una rete ad-hoc non vi è distinzione alcuna tra host e router; ogni nodo ricopre entrambe le funzionalità che sono invece ben distinte in una rete cablata. Questa prerogativa delle reti manet fa si che un nodo, per poter partecipare alla rete, debba occuparsi non solo dell'invio e ricezione dei propri pacchetti ma anche dell'inoltro dei pacchetti provenienti da altri nodi. Per fare questo, ogni nodo deve destinare parte delle sue risorse[1] alla collaborazione con gli altri nodi. Questa caratteristica è fondamentale nelle reti ad-hoc perché al loro interno non vi è un'infrastruttura di rete fissa che provvede al mantenimento delle rotte tra i nodi, e sono invece i nodi stessi a creare l'infrastruttura di rete. Nelle Manet,

[1] banda, capacità di calcolo, energia

inoltre, si aggiunge un altra complicazione di cui tenere conto: l'estrema mobilità dei nodi che rende ancora più complesso il compito dell'algoritmo di routing.

Un nuovo nodo X può entrare nel raggio di copertura di un nodo facente parte della rete e quindi diventarne anch'esso parte. Per poter iniziare a comunicare all'interno di essa è però necessario che tale nodo X ottenga informazioni sulla topologia della rete; deve inoltre comunicare il proprio ingresso in rete ai suoi diretti vicini in modo che questi possano recapitare i pacchetti a lui destinati, siano essi pacchetti dati o di routing.

Al contrario un nodo Y può uscire dalla rete non essendo più nel raggio di copertura di alcun nodo. Per far fronte a questa modifica di topologia i nodi che per ultimi hanno avuto contatti con lui devono diffondere l'informazione in rete.

In generale un qualsiasi movimento di nodi all'interno della rete ne modifica la topologia e quindi richiede la modifica, la creazione o la cancellazione di cammini di routing.

A differenza delle reti cablate, dove le informazioni di routing rimangono fisse per molto tempo e possono quindi essere gestite con protocolli tradizionali, nelle Manet occorre studiare protocolli nuovi che facciano fronte alle esigenze di queste topologie così variabili nel tempo.

È possibile suddividere i protocolli di routing ad-hoc in due grandi famiglie: *proattivi* e *reattivi*. In un protocollo *proattivo* i nodi si scambiano continuamente informazioni sul routing. Un nodo conosce in ogni momento la topologia della rete e le rotte verso tutti gli altri nodi che la compongono. Questa tipologia di protocolli si adatta molto bene a situazioni in cui vi siano molte comunicazioni tra molte coppie diverse di nodi. D'altra parte il continuo scambio di informazioni porta ad un cospicuo consumo di risorse di rete che non sarebbe giustificato in caso di poche comunicazioni. Per questo motivo si sono studiati anche protocolli *reattivi*. Questi protocolli ricavano le informazioni di routing *on-demand*, cioè creano una rotta tra due nodi solo nel momento in cui vi sia una richiesta di comunicazione da parte del nodo sorgente. Quando un nodo S vuole comunicare con un nodo D inizia il processo detto di *route discovery* con il quale, se il nodo

D è effettivamente raggiungibile, viene creata la rotta e vengono inserite le informazioni relative nelle tabelle di routing di tutti i nodi lungo il cammino. Finché questa procedura non è terminata il nodo S non può iniziare la trasmissione dei dati. Nei protocolli reattivi un nodo ha unicamente una conoscenza parziale della topologia di rete, a differenza dei protocolli proattivi dove ogni nodo conosce le informazioni di instradamento verso tutti i nodi. Nei protocolli reattivi inoltre le informazioni di routing ottenute con la discovery non vengono mantenute costantemente aggiornate ma, una volta terminata la comunicazione, vengono lasciate scadere. Questo perché, a causa del movimento dei nodi, le informazioni potrebbero non essere più valide trascorsi anche solo pochi secondi dalla fine della comunicazione. Per questo motivo nelle reti mobili ad-hoc si associa ad ogni riga della tabella di routing un timeout, scaduto il quale la rotta viene considerata non più valida. A seconda del protocollo di routing utilizzato possono esserci gestioni più complesse della validità delle rotte. Tutti i valori riguardanti i timeout di validità sono comunque un parametro chiave per gli algoritmi di routing delle manet. Infatti un tempo di validità eccessivamente elevato potrebbe portare ad avere informazioni di routing *"vecchi"*, cioè informazioni che non rispecchiano più la mutata topologia. Viceversa tempi di validità troppo brevi possono portare a far scadere rotte ancora valide e perciò, in caso si avvii un'altra comunicazione sullo stesso percorso, a far cominciare un nuovo e dispendioso processo di discovery. Il processo di discovery infatti comporta un certo ritardo dovuto al tempo di propagazione delle informazioni, che è estremamente variabile e dipende da molti fattori quali la grandezza della rete, la mobilità dei nodi e, ovviamente, il tipo di protocollo adottato. In generale il tempo di creazione delle rotte è uno dei fattori principali nella scelta del protocollo di routing adottato da una rete.

Negli ultimi anni sono state pubblicate numerosissime proposte di protocolli per il routing nelle reti mobili ad hoc: protocolli proattivi, reattivi, misti (in parte proattivi e in parte reattivi) e protocolli che sfruttano la conoscenza della posizione spaziale dei nodi. Tra tutti questi il gruppo di lavoro IETF che si occupa del routing nelle manet ha scelto tre protocolli che sono stati pubblicati

come *experimental RFC*[2]. Altri tre invece sono ancora al vaglio del gruppo e ne è stato reso disponibile per il momento un *Internet Draft*.

2.2 I protocolli in fase di standardizzazione

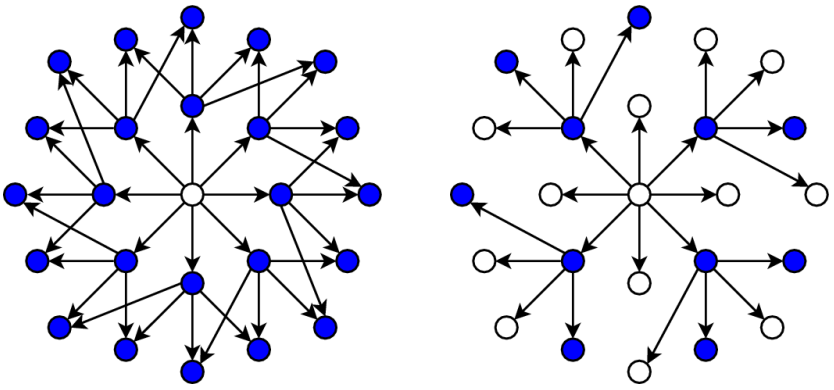
In questo paragrafo si descriveranno brevemente due dei tre protocolli in fase di standardizzazione da parte dell'IETF. Il terzo, AODV, verrà trattato più in dettaglio nel paragrafo 2.4 in quando su di esso si concentra l'attenzione di questo lavoro.

2.2.1 OLSR

OLSR [6] (Optimized Link State Routing) è un protocollo proattivo ed è quindi adatto a topologie particolarmente mobili in cui i nodi comunicano molto tra loro creando molti cammini diversi. Questo protocollo è un'ottimizzazione per le Manet del classico algoritmo *link state* (utilizzato in protocolli per reti cablate come OSPF [7]). Il concetto chiave alla base dell'ottimizzazione effettuata è quella dei *multipoint relays* (MPR), ovvero l'insieme di vicini ad un passo che permettono ad un nodo di raggiungere tutti i suoi vicini a due passi. Per creare questo sottoinsieme di vicini si utilizzano i messaggi di *hello* inviati periodicamente a tutti i vicini a uno e due passi. Una volta ottenuto l'insieme di *tutti* i vicini, il nodo estrae da esso gli MPR e la lista dei nodi dai quali esso è stato eletto MPR (detti *Multipoint Relay Selector set*). Utilizzando queste informazioni OLSR introduce tre principali ottimizzazioni:

- Se un nodo riceve un messaggio in broadcast ne fa il forward solo se appartiene all'MPR del nodo mittente. Si limita in questo modo il *flooding* nell'intera rete (si veda figura 2.1);
- le informazioni sullo stato dei collegamenti vengono generate solo da nodi che sono stati eletti come MPR da almeno uno dei propri vicini

[2]Request For Comment, reperibili all'indirizzo http://www.ietf.org/rfc



(a) processo di flooding tradizionale

(b) processo di flooding in OLSR

Figura 2.1: I MultiPoint Relays: solo i pacchetti evidenziati effettuano il forward dei messaggi

- quando un nodo crea un messaggio di *topology update*[3] "può"[4] decidere di inserirvi solo le informazioni riguardanti i collegamenti con i nodi dai quali è stato eletto MPR. In questo caso, a differenza dei comuni protocolli link state, si diffondono solo informazioni parziali sulla rete.

Il gruppo di lavoro Manet sta vagliando una seconda versione del protocollo OLSR, *OLSRv2*. Questa nuova versione, disponibile ora sotto forma di draft, si basa sugli stessi elementi chiave del suo predecessore, incorporando però una struttura di segnalazione più flessibile, alcune semplificazioni nella struttura dei messaggi che vengono scambiati dai nodi e una codifica degli indirizzi IP più compatta. Inoltre questa seconda versione rispetta il formato standard dei pacchetti di routing nelle Manet introdotto da [9].

2.2.2 TBRPF

Il protocollo TBRPF[10](Topology Dissemination Based on Reverse Path Forwarding) è anch'esso di tipo proattivo ed è composto da due moduli indipendenti che svolgono funzioni differenti:

[3]questi sono i messaggi con i quali si diffondo le notizie sui cambiamenti di topologia

[4]MAY secondo la terminologia standard dettata da [8]

Neighbour discovery - Questo modulo viene utilizzato dai nodi per creare una lista dei propri vicini. A differenza di OLSR vengono mantenute informazioni sui soli vicini ad un passo. Queste informazioni vengono recuperate attraverso messaggi di hello differenziali, cioè che contengono solo il cambiamento di stato della connessione tra i due nodi. La connessione può essere di tre tipi:

- *1 way* se X riceve un messaggio di hello da Y il quale non pubblicizza X come suo vicino;
- *2 way* se X riceve un messaggio di hello da Y il quale nello stesso messaggio di hello indica X come suo vicino 1-way o 2-way;
- *lost* se dopo un certo intervallo di tempo i 2 nodi non ricevono messaggi di hello l'uno dall'altro.

Se non vi sono cambiamenti di stato da segnalare nell'ambito dei propri vicini i messaggi di hello vengono comunque inviati, in modo da mantenere attive le connessioni e segnalare la propria presenza. In questo caso i messaggi vengono inviati vuoti in modo da risparmiare banda.

Routing Module - Questo è il vero e proprio modulo che gestisce l'invio e l'elaborazione delle informazioni di routing. Il routing module gestisce due differenti tipi di messaggi di topology update:

- *normali* - Questi messaggi vengono inviati periodicamente in broadcast (per esempio ogni 5 secondi), vengono elaborati dai nodi vicini dopodiché vengono inoltrati e si propagano per tutta la rete. In questi messaggi il nodo sorgente include una parte del suo albero di routing. Per limitarne la dimensione ogni nodo che elabora il messaggio sottrae alcune informazioni usando la tecnica del *reverse path forwarding* [11].
- *straordinari* - Questi messaggi vengono invece utilizzati quando il collegamento tra un nodo e un suo vicino cambia stato. In questo caso la

sorgente costruisce un messaggio di routing update straordinario contenente i cambiamenti dovuti alla modifica della connettività col vicino; i nodi che lo ricevono modificano le loro tabelle di routing, se necessario, e ritrasmettono il messaggio solo in caso questo abbia portato a cambiamenti nelle loro rotte.

Il fatto di avere una struttura modulare permette a TBRPF di poter, in un futuro, utilizzare una diversa strategia di gestione dei vicini con il medesimo modulo di routing.

2.3 I protocolli in fase di studio

Questi protocolli sono ancora al vaglio della comunità scientifica. Per il momento ne è stato proposto e pubblicato un draft[5].

2.3.1 DYMO

Il protocollo DYMO [12] (Dynamic Manet On-demand) è l'ultimo nato tra i protocolli di routing delle manet ed è di tipo reattivo. Le operazioni principali svolte da DYMO sono la *route discovery* e il *route management*.

Nel il processo di route discovery il nodo sorgente S, volendo comunicare con il nodo D, manda in broadcast un pacchetto di *Route Request* (RREQ) nell'intera rete. Durante questo processo di flooding ogni nodo intermedio inserisce nella sua tabella di routing le informazione sul percorso verso il nodo S[6]. Quando il nodo destinazione D riceve la RREQ risponde al nodo sorgente con un messaggio di Route Reply (RREP) che viene spedito in *unicast* verso S. Ogni nodo intermedio, prima di fare il forward della RREP, memorizza nella sua tabella di routing una riga con le informazioni sulla rotta verso D (*forward route*) e provvede all'inoltro del pacchetto verso S sfruttando la rotta inserita al passaggio della RREQ. Quan-

[5] reperibile sul sito http://www.ietf.org/ID.html

[6] chiamato *reverse route*

do S riceve la RREP la rotta tra i due nodi è stabilita e la sorgente può iniziare l'invio dei pacchetti dati.

DYMO utilizza i *Sequence Number* per evitare di creare rotte cicliche come specificato in [13]. Inoltre questi permettono anche di evitare l'utilizzo da parte dei nodi di informazioni di routing vecchie. Dell'ultilizzo dei sequnce number si parlerà più dettagliatamente nel paragrafo 2.4 quando lo si introdurrà per il protocollo AODV.

Per reagire in maniera corretta ai frequenti cambiamenti della topologia di rete i nodi monitorano i collegamenti per i quali sono nodi intermedi. Per fare questo utilizza i messaggio di hello con i quali viene verificata l'effettiva connettività con i nodi vicini. Quando viene ricevuto un pacchetto per una rotta che non è più disponibile il nodo sorgente della comunicazione viene informato dell'impossibilità di effettuare la consegna tramite un messaggio di *Route Error* (RERR). Al ricevimento del messaggio di RERR il nodo sorgente si preoccupa di iniziare nuovamente un processo di discovery.

Una caratteristica importante di DYMO è quella di prevedere delle estensioni. Infatti il formato dei pacchetti è stato studiato in modo tale da permettere in un futuro la gestione da parte del protocollo di estensioni per migliorare o aggiungere nuove funzionalità al protocollo come per esempio la gestione della sicurezza del routing[7].

Un'altra caratteristica innovativa di DYMO è quella di avere una struttura dei messaggi modulare. Infatti ogni messaggio è composto da una parte fissa (l'*header*) e da una parte che dipende dal tipo di messaggio, il *Routing Element* (RE). Ogni RE è a sua volta composto da uno o più *Routing Block* (RBlock). Gli unici messaggi per cui questo schema non è applicato sono i messaggi di Routing Error che hanno un formato proprio.

Quando un nodo S vuole iniziare un processo di discovery verso un nodo D crea un nuovo messaggio DYMO all'interno del quale crea un RBlock indicando

[7]è già stata proposta un'estensione per gestire la sicurezza del protocollo, SDYMO che verrà descritta nel paragrafo 3.2.4

D come destinazione della RREQ e invia il messaggio in broadcast. Quando D riceve il messaggio risponde con un ulteriore pacchetto DYMO creando a sua volta un RBlock contenente le informazioni riguardanti la rotta e lo spedisce, questa volta in unicast, verso S. Durante questo processo di discovery ogni nodo intermedio che riceve i messaggi di RREQ o RREP elabora tutti i Rblock in esso contenuti inserendo o aggiornando le rotte da essi descritte. Inoltre può a sua volta creare un nuovo RBlock contenente le *sue* informazioni di routing. Queste nuove informazioni verrano così elaborate da tutti i nodi successivi che aggiorneranno le proprie rotte con dati che potrebbero abbreviare i successivi processi di discovery.

2.3.2 DSR

Il protocollo DSR [14] (Dynamic Source Routing) è un protocollo reattivo di tipo *source routing*, ottimizzato per essere utilizzato in ambienti particolarmente dinamici e collaborativi come le reti wireless ad-hoc.

È possibile suddividere il protocollo in due fasi: *route discovery* e *route maintenance*. La prima viene eseguita quando un nodo sorgente A deve cercare un percorso per raggiungere un nodo destinazione E; la seconda invece è sempre attiva durante il trasferimento dei dati e si occupa di monitorare il cammino verso E per verificare (direttamente o attraverso dei messaggi di errore provenienti dagli altri nodi della rete) che sia sempre valido. Se A riscontra dei problemi sul percorso utilizzato per raggiungere E, può consultare la sua tabella di routing per cercare un percorso alternativo oppure può iniziare una nuova fase di route discovery.

Durante la fase di route discovery il nodo sorgente A cerca una rotta verso E mandando in broadcast un messaggio di *Route Request* (RREQ). I nodi vicini di A (nell'esempio di figura 2.2 B) aggiungono il loro indirizzo in coda a quello di A e rispediscono in broadcast il messaggio. Ogni altro nodo intermedio aggiunge in coda al messaggio RREQ il proprio indirizzo e ritrasmette il messaggio ai suoi

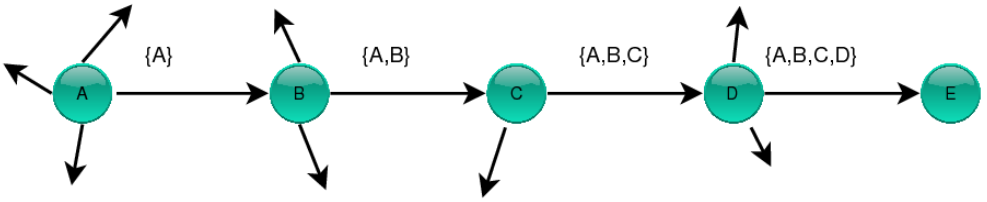


Figura 2.2: Il processo di route discovery in DSR

vicini[8]. Il nodo destinazione E si preoccupa di rispondere ad A con un messaggio *Route Reply* (RREP). D può scegliere di adottare tre diversi tipi di risposta:

- utilizzare un percorso di routing già presente nella sua tabella di routing;
- invertire il percorso che è stato accumulato durante il flooding del messaggio RREQ ed utilizzarlo per inviare la RREP[9];
- iniziare un'ulteriore route discovery per la destinazione A; in questo caso E allega al nuovo messaggio RREQ anche tutto il percorso accumulato dalla route discovery iniziata precedentemente da A.

In DSR ogni nodo è responsabile della connettività verso i suoi vicini ad un passo. Infatti, una volta inviato un pacchetto ad un qualsiasi vicino deve accertarsi che sia stato ricevuto correttamente. Per fare questo può agire con diverse modalità:

- può utilizzare un meccanismo di ack a livello 2, se è supportato dal protocollo di livello inferiore;
- può rimanere in ascolto in modalità promiscua sul canale per captare se i nodi vicini effettuano o meno l'inoltro dei pacchetti verso il nodo destinazione. Questa strategia non richiede l'utilizzo dei messaggi di ack[10], ma

[8]Il fatto di avere nel messaggio di request l'intera lista dei nodi attraversati è la caratteristica distintiva dei protocolli source routing

[9]in questo caso si presuppone che i link siano tutti bidirezionali

[10]questa tecnica però viene a volte chiamata *ack passivo*

comporta una maggiore occupazione delle risorse dei nodi poiché sono costretti ad elaborare anche pacchetti non destinati a loro; inoltre non si può essere certi che il canale radio sia bidirezionale e quindi se un nodo X trasmette un pacchetto a Y ed Y lo inoltra verso Z, è possibile che X non si accorga della ritrasmissione verso Z;

- può utilizzare degli appositi messaggi ack di livello 3 che sono specificati nel draft del protocollo DSR.

Se, dopo un certo numero di tentativi, non si è riusciti a portare a buon fine la trasmissione allora il link viene considerato inutilizzabile; il nodo che se ne accorge deve aggiornare la sua tabella di routing ed informare, con un messaggio di *Route Error* (RERR), tutti i suoi vicini che utilizzano tale link.

Il nodo, seguendo le linee guida del draft, può memorizzare le informazioni di routing in due modalità:

path cache che consiste nel memorizzare una riga per ogni percorso di routing; se un nodo conosce due o più percorsi per una stessa destinazione, ci saranno due o più righe associate a quella destinazione. Questa modalità richiede una elevata capacità di memorizzazione, ma consente una veloce ricerca del cammino di routing all'interno della memoria del nodo.

link cache in cui il nodo costruisce in memoria un grafo della topologia della rete e, al momento di cercare un percorso, utilizza un algoritmo specifico di ricerca (ad esempio l'algoritmo di Djikstra [15]). Questa modalità ha minori requisiti in fatto di capacità di memorizzazione, ma ne ha maggiori per quel che riguarda la capacità di calcolo del nodo, che deve essere in grado di implementare l'algoritmo di ricerca scelto; inoltre la ricerca del cammino di routing all'interno del grafo introduce un ritardo maggiore rispetto alla modalità path cache.

La caratteristica di DSR di avere in tabella più di un percorso per ogni destinazione pone un nuovo problema: la scelta del percorso *migliore*. In una rete

mobile ad hoc è infatti difficile stabilire quale sia il percorso migliore sul quale instradare i pacchetti dati poiché non è detto che sia sempre quello a distanza minima. Utilizzando un protocollo source routing un nodo può valutare, a differenza dei protocolli nei quali il nodo non ha la visibilità totale sul percorso che effettueranno i pacchetti dati, i costi di tutti i singoli link tra i nodi, come il ritardo stimato o la banda disponibile; inoltre il nodo sorgente può valutare l'affidabilità e/o la fiducia dei nodi intermedi, preferendo ad esempio un percorso privo di nodi sconosciuti o poco fidati. Nella sezione 2.4 si vedrà che per il protocollo AODV i percorsi migliori sono quelli più nuovi, anche se questi a volte non sono quelli a lunghezza minima; la motivazione di questa scelta sta nel fatto che i messaggi più nuovi rispecchiano meglio la topologia della rete.

Alcune considerazioni importanti su DSR sono:

- la dimensione media dei messaggi RREQ, RREP e dei pacchetti dati cresce al crescere della dimensione della rete perché si allunga la lunghezza media dei percorsi di routing e, di conseguenza, anche le liste di nodi contenute nei messaggi;

- alla luce di quanto detto al punto precedente, è altamente sconveniente utilizzare indirizzi IPv6[11] con il protocollo DSR poiché si rischia di avere un eccessivo overhead in ogni pacchetto di segnalazione oltre che nei pacchetti dati;

- il nodo destinazione e tutti i nodi intermedi estraggono molte informazioni da ogni messaggio di segnalazione, perché tali messaggi contengono la lista di tutti i nodi attraversati e non soltanto gli indirizzi dei nodi sorgente e destinazione: nell'esempio in figura 2.2 il nodo C, per esempio, oltre a individuare una rotta verso S, impara che può raggiungere anche A attraverso il suo vicino B;

- per quanto detto al punto precedente, il numero di route request inviate dai

[11] Gli indirizzi IPv6 hanno una lunghezza di 128 bit rispetto ai 32 bit degli indirizzi IPv4

nodi tende a diminuire nel tempo poiché essi tendono ad acquisire sempre più informazioni sulla topologia della rete senza la necessità di iniziare un nuovo processo di discovery;

- grazie ai messaggi di segnalazione e alle informazioni *ridondanti* che essi contengono ogni nodo può costruirsi un grafo parziale della rete e scegliere di volta in volta il cammino *migliore* per raggiungere una determinata destinazione;

- per il medesimo motivo ogni nodo può memorizzare percorsi alternativi per raggiungere gli altri nodi della rete: se durante un trasferimento di dati un percorso non è più utilizzabile, il nodo sorgente può immediatamente sceglierne un altro senza dover perdere tempo e risorse per effettuare una route discovery.

Per ovviare ai problemi esposti nei primi due punti il protocollo DSR può utilizzare una particolare opzione descritta nel suo draft: il *flow state*

Nella modalità flow state il primo pacchetto dati generato contiene tutta la lista dei nodi che devono essere attraversati; inoltre contiene un numero univoco scelto dalla sorgente (*Flow ID*), che identifica il flusso di dati che fluirà in seguito dalla sorgente alla destinazione. Tutti i nodi attraversati da questo primo pacchetto dati memorizzano la n-upla <*Nodo sorgente, Nodo destinazione, Flow ID, Next Hop*> in una tabella temporanea. Tutti i pacchetti successivi al primo non dovranno contenere esplicitamente l'intero percorso di routing da seguire, ma potranno essere instradati grazie al solo identificativo Flow ID. La modalità flow state viene richiesta dal nodo sorgente e deve essere supportata da tutti i nodi intermedi attraversati: se un nodo non supporta tale modalità avvisa il nodo sorgente, il quale utilizzerà in seguito la versione base del protocollo DSR, allegando cioè ad ogni pacchetto dati l'intera lista di nodi da attraversare.

2.4 il protocollo AODV

In questo paragrafo si descriveranno le azioni fondamentali del protocollo AODV che è la base sulla quale si poggia l'estensione di sicurezza per il routing nelle manet studiata in questa tesi.

2.4.1 Descrizione generale

Il protocollo AODV [13](Ad-hoc On-demand Distance Vector) è di tipo reattivo ed è pensato per essere usato in reti manet composte da un numero variabile di nodi tra le decine e le migliaia. Essendo stato appositamente studiato per l'ambito ad hoc offre un rapido adattamento alle condizioni variabili dei link, un basso utilizzo della rete (essendo reattivo produce traffico solo quando viene richiesta una rotta se questa non è già presente nella tabella di routing o quando si deve segnalare la caduta di un link precedentemente utilizzato per il trasporto dati), una richiesta non eccessiva di memoria (non richiede ai nodi di mantenere le rotte verso destinazioni non coinvolte in comunicazioni attive) e di potenza di calcolo. Si assicura l'assenza di rotte cicliche con l'utilizzo dei sequence number (si veda paragrafo 2.4.1.1) e evitando il problema del *counting to infinity* si assicura una veloce convergenza quando la topologia di rete viene modificata.

Ogni nodo mantiene in memoria una tabella di routing nella quale vengono memorizzate le seguenti informazioni (oltre ad altri campi di controllo che si ometteranno):

- indirizzo IP del nodo destinazione (*destination IP*),
- numero di sequenza del nodo destinazione (*destination sequence number*),
- flag di validità della rotta (indica se la rotta è valida o invalida o in riparazione),
- distanza in numero di hop (*hop count*),
- lista dei precursori per quella rotta,

- tempo di validità (*lifetime*),

Si supponga ora che un nuovo nodo X entri a far parte della rete e voglia comunicare con A; la figura 2.3 illustra il funzionamento del protocollo AODV in questo caso:

(a) poiché S non ha una riga nella sua tabella di routing riferita ad X, dà inizio alla procedura di *Route Discovery*: per prima cosa genera un messaggio RREQ[12] nel quale inserisce il suo indirizzo IP nel campo *Originator IP Address* e quello di X nel campo *Destination IP Address*; quindi spedisce il messaggio in broadcast locale, inserendo il valore `255.255.255.255` nel campo *Destination Address* dell'header IP. I nodi E, F e G ricevono il messaggio di S e aggiornano le loro tabelle di routing avendo ottenuto informazioni nuove sulla rotta per raggiungere S.

(b) I nodi E, F e G rispediscono la RREQ in broadcast locale. I nodi G, F , E e S ricevono una seconda copia dello stesso messaggio RREQ, se ne accorgono[13] e lo scartano. I nodi A e D apprendono una rotta verso S inserendo in tabella come next hop G e E rispettivamente. Il nodo C riceve un primo messaggio da E ed aggiunge una rotta verso S passando per E; quando riceve il secondo messaggio da F, lo scarta.

(c) I nodi A, C e D rispediscono la RREQ in broadcast locale. B e X la ricevono e aggiornano le loro tabelle aggiungendo una rotta verso S passante per C (ipotizzando che arrivino prima i messaggi provenienti da C piuttosto che quelli provenienti da A e D). Tutti gli altri nodi scartano il messaggio ricevuto avendolo ricevuto anche in precedenza. Il nodo X si accorge che il suo indirizzo IP coincide con il Destination IP Address indicato nel messaggio

[12] il formato dei pacchetti RREQ, RREP, RERR verrà descritti in dettaglio nei paragrafi 2.4.2, 2.4.3 e 2.4.5 rispettivamente

[13] i nodi mantengono una lista di RREQ ricevute per evitare di elaborare due volte lo stesso messaggio

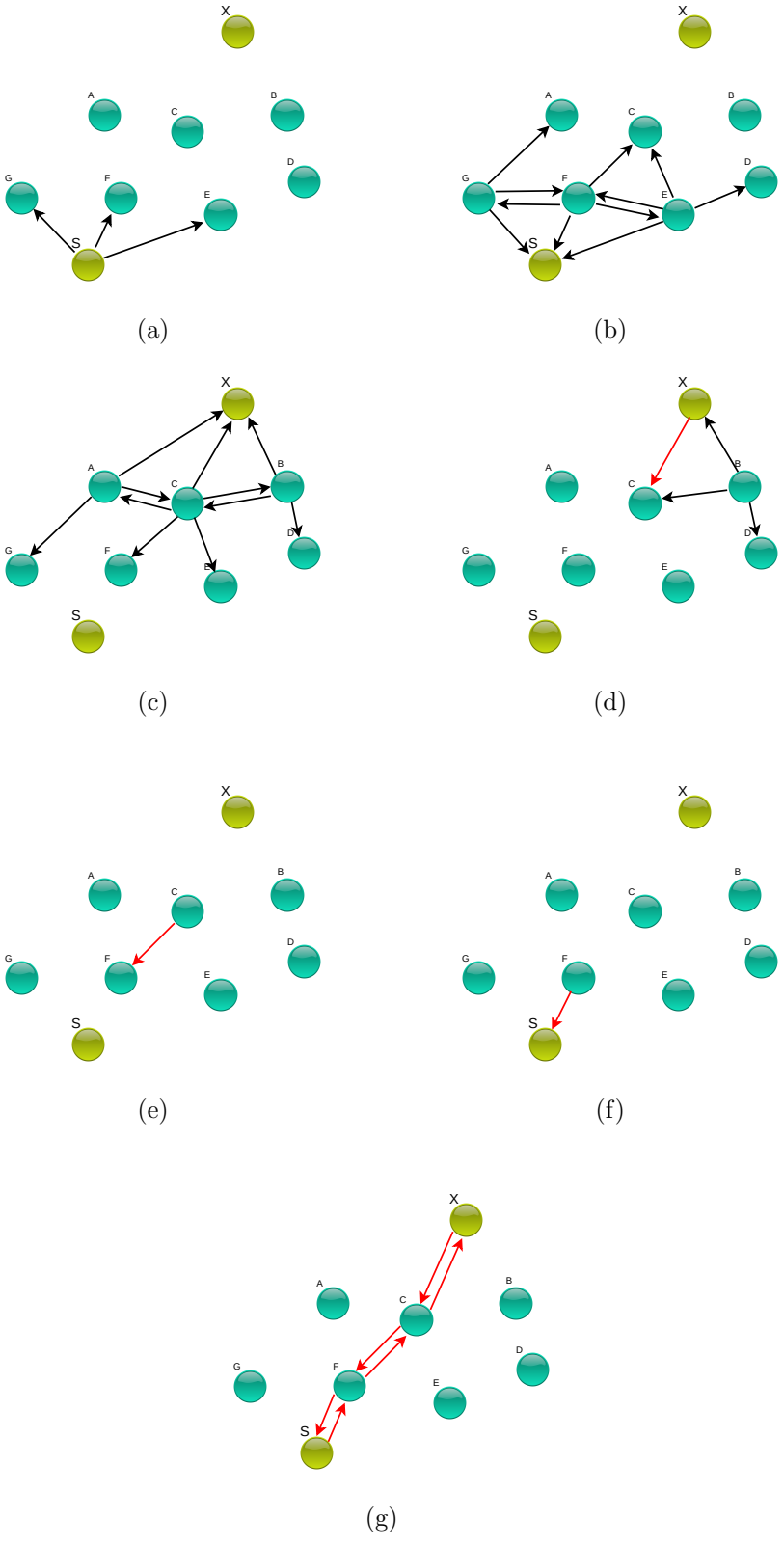


Figura 2.3: Un esempio di Route Discovery

RREQ e prepara quindi un messaggio di risposta (RREP) copiando il proprio indirizzo nel campo Destination IP Address e quello di S nel campo Originator IP Address.

(d) Questa RREP deve essere spedita in unicast a S, quindi X controlla nella sua tabella di routing verso quale nodo spedire il messaggio e, trovando l'informazione inserita poco prima, imposta come next hop C; il messaggio RREP viene quindi inserito in un pacchetto IP e spedito a C in unicast. Nel frattempo il nodo B inoltra la RREQ a X, D e C che la scartano essendo un duplicato di quella precedente.

(e) Il nodo C, ricevendo la RREP da parte di X, aggiorna la rotta verso quest'ultimo nella sua tabella di routing. Inoltre, poiché ha ricevuto dal suo vicino X la RREP destinata a S, inserisce X nella lista dei precursori associata al nodo S[14]. A questo punto inoltra il messaggio RREP verso il nodo F.

(f) F riceve la RREP, aggiorna la rotta verso X inserendo C come next hop, aggiunge C nella lista dei precursori per la destinazione S ed inoltra il messaggio RREP a S.

(g) A questo punto è stato instaurato un cammino da S verso X e viceversa e S può iniziare la comunicazione.

2.4.1.1 Gestione del numero di sequenza

AODV è un protocollo di tipo *Distance vector*, cioè usa come fattore discriminante tra due rotte verso la stessa destinazione il numero di hop che la separa dalla sorgente. Questo sarebbe adeguato per un ambito classico (reti cablate). In ambito manet invece, a causa della scarsa affidabilità del canale radio e, soprattutto, della mobilità dei nodi, la topologia può cambiare imprevedibilmente e quindi la scelta della rotta più breve potrebbe non essere la migliore. Per questo motivo in AODV si sono introdotti i numeri di sequenza. In generale il numero

[14] il significato e l'uso di questa lista verrano descritti nel capitolo 2.4.5

di sequenza associato ad un nodo è monotonamente crescente; per questo motivo un messaggio M_1 con numero di sequenza maggiore di un messaggio M_2 contiene informazioni più aggiornate. Ne risulta che la scelta tra due rotte verso la stessa destinazione in AODV viene fatta prima valutando il numero di sequenza della destinazione e poi, in caso questi siano uguali, si discrimina in base al numero di hop. Per questo motivo i numeri di sequenza sono importanti anche per evitare che si creino cicli nei cammini di routing. Un nodo aggiorna il proprio numero di sequenza in due casi:

- prima di generare una richiesta (RREQ) destinata ad un qualsiasi altro nodo della rete;
- prima di generare un messaggio RREP, in risposta ad un RREQ proveniente da un altro nodo della rete.

In caso un nodo perda il proprio numero di sequenza (per esempio in caso di riavvio della macchina) il protocollo prevede una procedura per ripristinarlo senza correre il rischio di compromettere le funzionalità di rete (si veda il paragrafo 2.4.1.3).

2.4.1.2 Gestione del campo time to live

Per evitare di sovraccaricare la rete con richieste RREQ, nell'RFC di AODV è consigliato[15] che il nodo sorgente che voglia iniziare una route discovery imposti il campo TTL dell'header IP secondo le seguenti regole:

- se è noto un precedente valore dell'hop count per la destinazione desiderata (derivante da un rotta presente in tabella ma invalidata perché non utilizzata per un periodo troppo lungo), viene usato tale valore, opportunamente incrementato per contemplare un eventuale allontanamento tra i nodi;
- altrimenti il TTL viene impostato a 1.

[15]SHOULD secondo la terminologia standard reperibile in [8]

Se il nodo sorgente non riceve risposta alla richiesta entro un intervallo di tempo pari a $2 \cdot NODE\_TRAVERSAL\_TIME \cdot (TTL\_VALUE + TIMEOUTBUFFER)$, allora spedisce un nuovo messaggio RREQ incrementando di 1 il campo RREQ ID e incrementando di 2 il campo TTL. $NODE\_TRAVERSAL\_TIME$ è il tempo stimato per l'attraversamento di un singolo nodo della rete e nell'RFC è indicato come default un valore di 40 millisecondi; TTL_VALUE è il valore utilizzato nel campo TTL dell'header IP; TIMEOUT BUFFER serve per contemplare eventuali ritardi dovuti a congestione ed è impostato di default a 2. Quando il campo TTL raggiunge la soglia $TTL\_THRESHOLD$ e non vengono ricevuti messaggi RREP di risposta, il nodo sorgente imposta il TTL al valore massimo possibile, cioè a $NET\_DIAMETER$. Questa tecnica di controllo del campo TTL è chiamata *expanding ring search* e viene utilizzata principalmente per due motivi:

- evitare una propagazione dei messaggi RREQ in tutta la rete, quando il nodo cercato è nelle vicinanze della sorgente;
- sfruttare una eventuale informazione già nota agli altri nodi della rete. Infatti, come illustrato in 2.4.3, se un nodo intermedio conosce una rotta valida verso la destinazione, può rispondere lui stesso con un messaggio RREP evitando ulteriori discovery.

2.4.1.3 Operazioni al riavvio di un nodo

A causa di un riavvio del dispositivo, un nodo può perdere il proprio numero di sequenza. Tale valore, come si è visto, è importante per distinguere i messaggi *nuovi* da quelli *vecchi*; quindi il nodo X, appena riavviato, non può riprendere il normale funzionamento impostando il suo numero di sequenza a zero poiché in questo modo rischierebbe di utilizzare numeri di sequenza vecchi (troppo bassi); se così fosse i messaggi di segnalazione da lui generati non sarebbero presi in considerazione dagli altri nodi della rete, che li scarterebbero in quanto considerati obsoleti. Per evitare questa situazione il nodo X che si riavvia può decidere di:

- aspettare un intervallo di tempo relativamente lungo per far scattare in questo modo tutti i timeout nelle tabelle di routing degli altri nodi della rete; così facendo non vi sarà più nessuna rotta verso X memorizzata ed il numero di sequenza di X può ripartire da zero;

- *imparare* il proprio numero di sequenza dai messaggi RREQ a lui destinati; infatti il nodo sorgente che genera i messaggi RREQ inserisce l'ultimo valore a lui noto del Sequence Number nel campo Destination Sequence Number e i nodi intermedi lo possono a loro volta aggiornare, se ne conoscono uno più recente.

Questa seconda possibilità sembra più conveniente in quanto X non deve aspettare che scadano i timeout per poter ricominciare ad operare. Ovviamente però se gli altri nodi della rete non generano RREQ per rotte verso X, X è costretto comunque ad aspettare; il nodo X dipende in questo modo dagli altri nodi della rete e non può gestire autonomamente il proprio numero di sequenza. Nell'RFC di AODV è inoltre specificato che se un nodo riceve un messaggio RREQ (a lui destinato) con Destination Sequence Number maggiore di quello da lui memorizzato, deve aggiornare il suo numero di sequenza copiando quello contenuto nel messaggio RREQ; questo fa sì che ogni nodo abbia sempre memorizzato un numero di sequenza maggiore, o tuttalpiù uguale, a quello presente nelle tabelle di routing degli altri nodi della rete.

2.4.2 I messaggi di Route Request (RREQ)

Il formato dei pacchetti RREQ è mostrato in figura 2.4. I campi principali presenti nel messaggio sono:

Type impostato a 1 per le RREQ;

J R G D U flag con varie funzioni; J e R sono riservati per il multicast, G e D vengono descritti nella sezione 2.4.2.2, U viene posto a 1 quando il messaggio porta un valore del numero di sequenza della destinazione non valido;

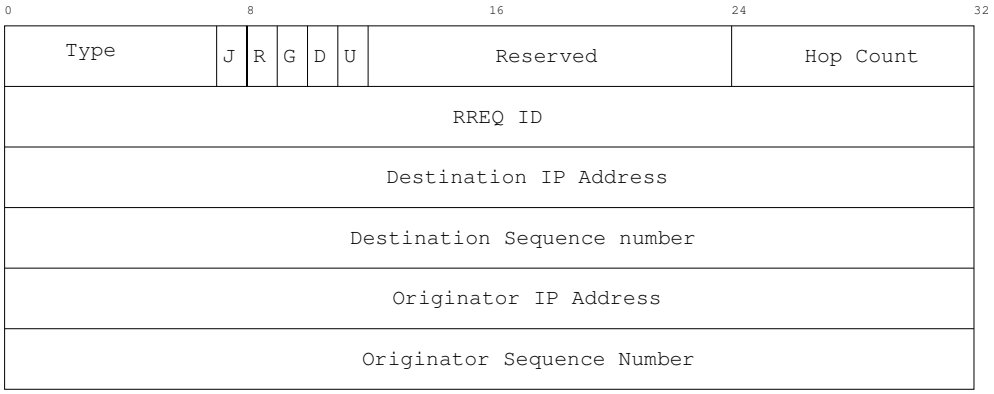


Figura 2.4: Il pacchetto RREQ

Hop Count viene inizializzato a zero e verrà incrementato dai nodi intermedi;

RREQ ID è il numero che identifica univocamente ogni messaggio RREQ generato dal nodo sorgente e viene sempre incrementato prima di essere inserito nel messaggio. Viene utilizzato dai nodi intermedi per identificare messaggi RREQ già ricevuti;

Destination IP Address indirizzo IP del nodo per il quale si sta cercando una rotta;

Destination Sequence Number è il numero di sequenza del nodo destinazione più aggiornato che il nodo sorgente conosce, se ne conosce uno. Poiché tale numero è in realtà gestito dal nodo destinazione, è possibile che il numero di sequenza inserito non sia in realtà quello attuale. Se S non ha una riga nella sua tabella di routing riferita a D con un numero di sequenza valido, inserisce tutti zeri in questo campo e imposta a 1 il flag U;

Originator IP Address è l'indirizzo IP del nodo sorgente;

Originator Sequence Number è il numero di sequenza del nodo sorgente della RREQ; viene incrementato ad ogni nuova route discovery ma non alla ritrasmissione di una RREQ in cui viene invece incrementato solo il RREQ ID.

Tutti i messaggi di AODV vengono scambiati sulla porta UDP numero 654; inoltre il nodo sorgente imposta il campo *Source address* al proprio indirizzo IP, il *destination address* all'indirizzo di broadcast locale `255.255.255.255` e il campo *time to live* seguendo lo schema visto in 2.4.1.2.

Una volta spedito il messaggio RREQ ai suoi vicini, il nodo sorgente resta in attesa di una risposta; se non viene ricevuta entro un timeout prestabilito, il nodo genera un nuovo messaggio RREQ incrementando i campi RREQ ID e TTL (come visto in 2.4.1.2).

Per evitare di sovraccaricare la rete di messaggi, si sono stabiliti nell'RFC del protocollo dei parametri: *RREQ RATELIMIT* e *RREQ RETRIES* che indicano rispettivamente la massima frequenza alla quale un nodo può inviare RREQ successive e il massimo numero di tentativi di discovery con TTL massimo consentite prima di segnalare la destinazione come irraggiungibile.

2.4.2.1 Operazioni al ricevimento di una RREQ

Ogni messaggio RREQ è univocamente identificato dalla coppia: <*Originator IP Address*, *RREQ ID*>

Poiché i messaggi RREQ si propagano in flooding in tutta la rete ogni nodo riceve più copie dello stesso messaggio. Per evitare di elaborare inutilmente tutte le copie ricevute, ogni nodo mantiene in una memoria temporanea le coppie <*Originator IP Address*, *RREQ ID*>; quando un nodo riceve un messaggio RREQ, per prima cosa controlla se la coppia di valori del messaggio è già presente nella sua memoria e, se la trova, scarta la RREQ senza ritrasmetterla. Le stesse operazioni vengono effettuate anche dal nodo sorgente stesso. Se un nodo riceve un messaggio RREQ nuovo allora controlla nella sua tabella di routing la presenza di una riga avente come Destination Address l'indirizzo del nodo sorgente e si comporta seguendo la seguente regola:

- se trova una corrispondenza controlla il valore dell'Originator Sequence Number del messaggio: se è maggiore del numero di sequenza memoriz-

zato nella tabella di routing aggiorna nella sua tabella la rotta verso il nodo sorgente S, e cioè:

  - vi copia il valore dell'Originator Sequence Number contenuto nel messaggio RREQ,
  - vi copia l'hop count contenuto nel messaggio RREQ,
  - vi copia l'indirizzo sorgente dell'header IP nel campo next hop,
  - aggiorna il valore del *Lifetime*.

- se non trova una corrispondenza inserisce una nuova riga nella sua tabella di routing con le informazioni riportate dalle RREQ.

La rotta verso il nodo S è così aggiornata e utilizzabile per instradare pacchetti dati. A questo punto il nodo intermedio può inoltrare la RREQ oppure, se conosce una rotta valida per raggiungere D, può rispondere lui stesso con un messaggio RREP come si vedrà nel paragrafo 2.4.3; nel primo caso:

- incrementa il valore dell'hop count del messaggio RREQ;
- decrementa il valore del TTL (nell'header IP);
- inserisce il proprio indirizzo nel campo Source Address (nell'header IP);
- aggiorna il campo Destination Sequence Number nel messaggio RREQ se conosce un numero di sequenza più aggiornato per la destinazione;
- rispedisce il messaggio in broadcast locale.

2.4.2.2 I flag D G

Durante l'elaborazione di un messaggio RREQ un nodo intermedio può decidere di rispondere con un messaggio RREP se, nella sua tabella di routing, ha una rotta valida e recente verso la destinazione cercata; se decide di farlo il nodo intermedio genera la RREP, la inoltra verso il nodo sorgente (Originator IP

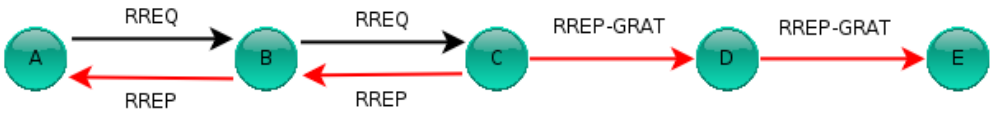


Figura 2.5: Il nodo C risponde come nodi intermedio: manda la RREP verso A e anche una RREP gratuita a E

Address) e scarta la RREQ ricevuta senza ritrasmetterla in flooding. Il nodo sorgente potrebbe però non volere che i nodi intermedi rispondano alla sua richiesta (per esempio perché non si fida delle loro informazioni o ha modo di credere che non siano aggiornate); il nodo sorgente ha quindi la facoltà di impostare i seguenti due flag nel messaggio RREQ:

D - destination only - solo la destinazione può rispondere con un messaggio RREP.

G - gratuitous RREP - quando un nodo intermedio decide di rispondere ad una RREQ deve necessariamente inviare un messaggio RREP anche al nodo destinazione per assicurarsi che la rotta sia ancora disponibile e che le informazioni siano presenti in tutti i nodi intermedi. Tale messaggio è detto *gratuito* perché il nodo destinazione non lo ha richiesto con nessuna RREQ. Il meccanismo è mostrato in figura 2.5.

I messaggi di RREP gratuiti sono dei comuni messaggi RREP che il nodo intermedio costruisce impostando:

Hop Count alla distanza nota a C verso A;

Destination Address al valore dell'indirizzo IP della sorgente del messaggio RREQ;

Destination Sequence Number al corrispondente valore presente nella riga della tabella di routing riferita al nodo sorgente della RREQ;

Originator IP Address all'indirizzo del nodo che effettua la risposta come nodo intermedio;

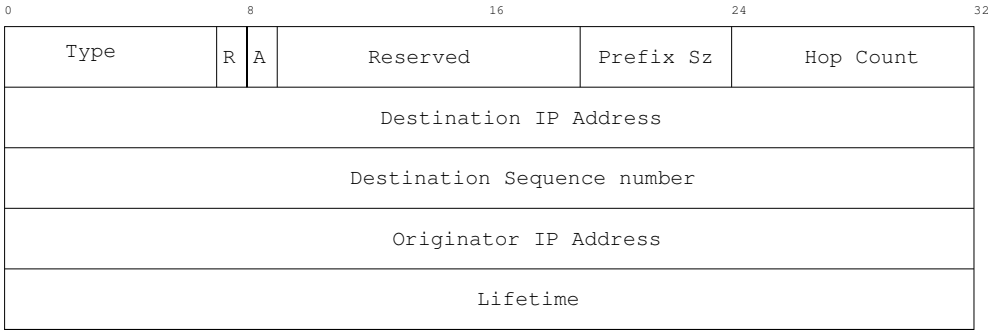


Figura 2.6: Il pacchetto RREP

Lifetime al valore corrispondente della tabella di routing riferita al nodo sorgente della RREQ.

2.4.3 I messaggi di Route Reply (RREP)

Il formato del pacchetto RREP è descritto in figura 2.6. Un nodo genera un messaggio RREP in risposta ad un messaggio RREQ se:

- è lui stesso il nodo destinazione della RREQ;
- è un nodo intermedio in possesso di una rotta sufficientemente aggiornata per la destinazione e il messaggio RREQ non ha impostato il flag D (si veda 2.4.2.2);

I campi principali del pacchetto RREP sono:

Type viene posto uguale a 2 per i messaggi RREP;

A R il flag R è riservato per il multicast, mentre A viene impostato per ricevere un riscontro (ack) dell'avvenuta ricezione del messaggio RREP, come spiegato in 2.4.6;

Prefix Size - Se il nodo destinatario della RREQ funge da gateway per una sottorete questo campo indica in numero di bit della maschera di rete;

Hop Count - Viene posto a zero se la RREP è generata dal nodo destinazione; se invece è generata da un nodo intermedio viene posto al valore dell'hop count della riga della tabella di routing riferita al nodo destinazione;

Destination IP Address - È l'indirizzo IP del nodo destinazione della richiesta, copiato direttamente dal campo omonimo del messaggio RREQ[16];

Destination Sequence Number - È il numero di sequenza più aggiornato associato al nodo destinazione della RREQ. Può essere quello reale se a rispondere è il nodo destinazione stesso o l'ultimo visto dal nodo intermedio che si appresta a rispondere;

Originator IP Address - È l'indirizzo IP del nodo che ha generato la richiesta e viene copiato direttamente dal campo omonimo della RREQ;

Lifetime - È il tempo di validità delle informazioni di routing trasportate dal messaggio. Se è il nodo destinazione a rispondere viene impostato ad un valore predefinito. Se è un nodo intermedio a farlo viene impostato al lifetime residuo della rotta a cui il messaggio si riferisce.

I messaggi RREP sono sempre inoltrati in *unicast* verso il nodo *originator*, quello cioè che ha formulato la richiesta di route discovery tramite un messaggio RREQ. Come già detto in precedenza, un messaggio RREP può essere generato dal nodo destinazione oppure da un nodo intermedio; le differenze riguardano solamente i campi Hop Count, Destination Sequence Number e Lifetime. Se poi la RREQ ha il flag *G* impostato, il nodo intermedio deve spedire un ulteriore messaggio *RREP gratuito* al nodo destinazione, come descritto nella sezione 2.4.2.2.

2.4.3.1 Operazioni all'arrivo di un messaggio RREP

Quando un nodo riceve un messaggio RREP, estrae dall'header IP l'indirizzo del nodo vicino dal quale ha ricevuto il messaggio e aggiorna la riga della sua

[16] originator e destination si riferiscono sempre al messaggio RREQ

tabella di routing relativa a tale nodo vicino; se il nodo è un nuovo vicino, allora viene aggiunta una nuova riga impostando come *invalido*[17] il numero di sequenza. Dopo aver fatto questo, il nodo intermedio incrementa il valore del campo hop count ed aggiorna la rotta verso il nodo destinazione[18], solo se si verifica uno dei seguenti casi:

- il numero di sequenza nella tabella di routing è invalido;
- il Destination Sequence Number nel messaggio RREP è maggiore del numero di sequenza contenuto nella tabella di routing;
- i numeri di sequenza sono uguali, ma la rotta nella tabella di routing è marcata come inattiva;
- i numeri di sequenza sono uguali, ma il nuovo hop count è inferiore.

In caso aggiorni la tabella di routing imposta la rotta come valida, modifica il *lifetime* utilizzando quello specificato nel pacchetto RREP e modifica i precursori aggiungendo il nodo da cui ha ricevuto il messaggio alla lista. Se il nodo non è il destinatario della RREP, cioè se non è il nodo *originator* del messaggio, allora consulta la tabella di routing per trovare le informazioni necessarie all'instradamento del pacchetto. A questo punto invia il pacchetto al next hop appena trovato inserendo l'indirizzo di quest'ultimo nodo nella lista dei precursori per la rotta verso il nodo Destination.

2.4.4 I messaggi di Hello

AODV è un protocollo reattivo, ma nell'RFC viene descritta la possibilità di mantenere proattivamente la connettività locale con i nodi vicini, utilizzando messaggi di Hello. Un messaggio Hello non è altro, in realtà, che un messaggio RREP spedito in broadcast locale (`255.255.255.255`) e con `TTL=1` in modo che

[17] infatti nel messaggio RREP non compare il valore del numero di sequenza del nodo vicino ma solo quello del nodo destinazione

[18] è il nodo destinazione della RREQ relativa alla RREP ricevuta

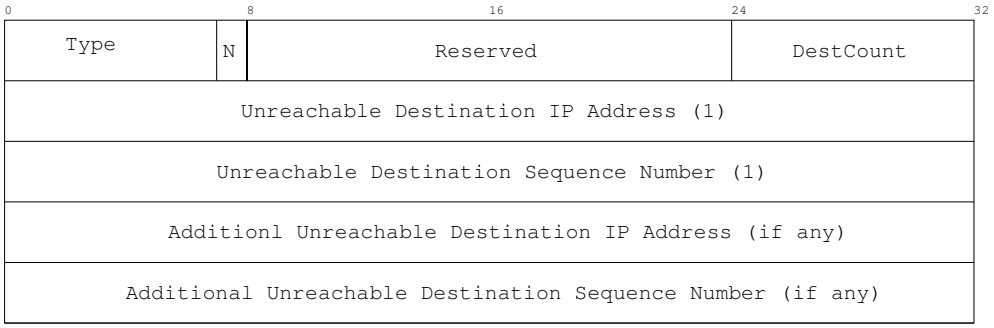


Figura 2.7: Il pacchetto RERR

non si propaghi per l'intera rete. Grazie a questi messaggi ogni nodo può sapere quali sono i suoi nodi vicini e può accorgersi quando uno di essi si allontana e diviene non più raggiungibile. Infatti se dopo un certo intervallo di tempo non vengono ricevuti messaggi di hello da un nodo la connessione verso di esso viene considerata invalida e viene spedito un messaggio di errore con le modalità descritte in 2.4.5.

Nelle specifiche del protocollo viene descritta un'altra modalità per il monitoraggio dello stato della connessione coi vicini: il *link layer feedback* brevemente descritto in 2.3.2.

2.4.5 I messaggi di Route Error (RERR)

I messaggi di errore vengono generati dai nodi della rete quando si verifica la rottura di una rotta. Un nodo invia un messaggio di errore nei seguenti casi:

- si accorge che il nodo successivo al quale deve inoltrare un pacchetto (next hop) non è più raggiungibile;
- arriva un pacchetto che non sa dove inoltrare poiché non ha la corrispondente riga nella sua tabella di routing;
- riceve a sua volta un messaggio di errore da uno dei suoi vicini.

Se un nodo rileva la presenza di un errore deve eseguire le seguenti operazioni:

- comporre una lista di tutti i nodi che non sono più raggiungibili;
- impostare, nella sua tabella di routing, come invalide le rotte verso i nodi non più raggiungibili;
- determinare a quali nodi vicini deve comunicare i cambiamenti controllando la lista dei precursori dei nodi non più raggiungibili;
- spedire i messaggi RERR a tali vicini

Una rotta non più raggiungibile non è cancellata immediatamente, ma viene impostata come invalida, per non perdere le informazioni sull'Hop Count e sul Sequence Number associati ad essa; il campo Lifetime nella tabella di routing viene impostato ad un valore predefinito e indica, in questa fase, dopo quanto tempo la rotta deve essere definitivamente cancellata dalla tabella di routing. I messaggi RERR vengono spediti in unicast se la lista dei precursori comprende pochi nodi (1 o 2), oppure in broadcast locale se i precursori sono la maggioranza dei nodi vicini.

Come mostrato in figura 2.7, nei messaggi RERR sono elencate le destinazioni non più raggiungibili ed i numeri di sequenza ad esse associati. Quando un nodo riceve un messaggio RERR aggiorna la sua tabella di routing impostando le rotte come invalide e vi copia il nuovo numero di sequenza; quando invece è il nodo stesso a generare un RERR (quando cioè si accorge che un link non è più funzionante o che il nodo destinazione di un pacchetto non è presente nella sua tabella di routing) allora incrementa di 1 il valore Destination Sequence Number precedente prima di spedire il messaggio ai nodi vicini per fare in modo che venga preso in considerazione e non scartato come *vecchio*. Quando il nodo intermedio che ha generato il messaggio di errore ha anche realizzato un local repair (sezione 2.4.5.1), il flag N (No delete) viene posto a 1.

2.4.5.1 Local Repair

La caduta di un link tra un nodo A e un nodo B può rendere alcune rotte non più raggiungibili. Prima di generare un messaggio di errore uno dei due nodi può provare ad effettuare la *riparazione* di una, alcune o tutte le rotte che sono state invalidate dalla caduta del link. La procedura che permette questo viene detta *Local repair*.

Se A si trova non troppo lontano della destinazione D può iniziare una procedura di route discovery, generando un messaggio RREQ destinato a D; mentre A aspetta un messaggio RREP di risposta, deve immagazzinare tutti i pacchetti destinati al nodo D che non è in grado di instradare. A questo punto è possibile che si verifichi una delle seguenti situazioni:

- A non riceve nessun messaggio RREP entro un intervallo di tempo prestabilito;
- A riceve uno o più messaggi RREP, ma con un valore di hop count che è superiore a quello precedentemente memorizzato;
- A riceve almeno un messaggio RREP con un valore di hop count minore o uguale a quello a lui precedente noto.

Nel primo caso il nodo A considera D come irraggiungibile e genera un normale messaggio di errore, come descritto in 2.4.5. Nel secondo caso la rotta viene considerata ancora valida ma siccome l'hop count è aumentato A genera un messaggio di errore ponendo a 1 il flag N: i nodi che ricevono tale messaggio RERR non devono cancellare dalla loro tabella di routing la riga associata a D, ma possono decidere di effettuare una nuova route discovery per la destinazione D per cercare una rotta più breve. Infine, nel terzo caso, la nuova rotta trovata da A per la destinazione D è equivalente o addirittura più corta di quella precedente, e per questo motivo non vengono generati messaggi di errore.

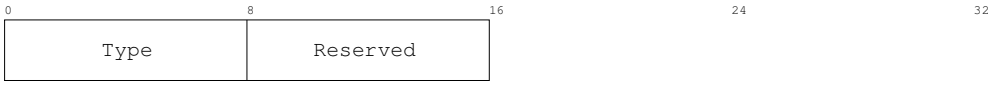


Figura 2.8: Il pacchetto RREP-ACK

2.4.6 I messaggi di Route Reply Acknowledge (RREP-ACK)

Nel messaggio RREP è possibile impostare il flag *A* per richiedere il riscontro dell'avvenuta ricezione (come visto in 2.4.3. Il riscontro viene effettuato con il semplice messaggio la cui struttura è mostrata in figura 2.8; il campo *Type* viene posto a 4, mentre i rimanenti 8 bit sono posti a zero e vengono ignorati in ricezione. Grazie al riscontro, chi trasmette il messaggio RREP è sicuro che della bidirezionalità del collegamento tra lui ed il vicino Questo messaggio è molto semplice e compatto e non reca informazioni sul messaggio RREP al quale si riferisce poiché si presuppone che l'intervallo di tempo tra spedizione del messaggio RREP ed arrivo del RREP-ACK sia trascurabile.

Capitolo 3

La sicurezza del routing nelle Manet

In questo capitolo verranno discussi i problemi di sicurezza specifici dei protocolli di routing nelle manet. Verranno quindi descritte le principali soluzioni presenti in letteratura con un'attenzione particolare verso il protocollo SAODV sul quale si basa il lavoro di questa tesi.

3.1 Principali problemi di sicurezza del routing

In nessuno dei protocolli presi in considerazione dall'IETF sono presenti soluzioni ai problemi di sicurezza del routing. Questo perché, nella prima fase dello sviluppo dei protocolli destinati a diventare standard, sì è puntata l'attenzione principalmente sulle specifiche architetturali e prestazionali (in termini di utilizzo di risorse di rete e locali) piuttosto che su quelle di sicurezza. Al punto di sviluppo attuale è diventato di fondamentale importanza integrare i protocolli studiati nella prima fase con modifiche specifiche volte a garantire i requisiti di sicurezza esposti in 1.3.1. Per fare questo occorre conoscere nei dettagli quali sono le vulnerabilità e i pericoli a cui è sottoposto il routing nelle manet.

Poiché in un ambiente collaborativo come quello delle reti ad hoc ogni nodo

partecipa al routing e poiché il canale di trasmissione radio permette l'accesso alla rete a chiunque, bisogna garantire che le informazioni relative al routing non vengano alterate da un eventuale nodo attaccante. Rendere sicuro un protocollo di routing significa garantire:

- l'autenticità dei nodi, i cui identificativi sono inseriti nei pacchetti;
- l'integrità dei principali campi dei pacchetti (gli identificativi dei nodi, i numeri di sequenza, il numero di nodi attraversati dai pacchetti);
- la freschezza dei dati, per evitare attacchi di tipo *replay*;
- la veridicità dei messaggi generati: ogni nodo è responsabile del suo comportamento e se genera messaggi non veritieri deve poter essere identificato ed escluso dalla rete.

In letteratura si tende ad utilizzare una classificazione degli attacchi che si possono portare ad una rete mobile ad hoc suddividendoli in base al tipo di disfunzione che recano alla rete. Parlando di attacchi specifici al protocollo di routing si individuano due categorie principali:

Routing distruption: sono attacchi che puntano a fare in modo che pacchetti di traffico legittimi vengano instradati in maniera errata;

Resource consumption: sono invece attacchi che si pongono come obiettivo quello di consumare risorse vitali per la rete, come la banda, o per i nodi, come memoria e potenza di calcolo.

Si può attuare un'altra grande divisione nei tipi di attacchi basata sulle caratteristiche dell'attaccante stesso. Secondo questa divisione gli attacchi si dividono in *interni* ed *esterni*, *passivi* e *attivi*. Si definiscono interni gli attacchi portati da un utente che *legittimamente* fa parte della manet, in contrapposizione con quelli esterni che sono effettuati da utenti non facenti parte della rete ma che si mettono in ascolto e trasmettono essendo nell'area di copertura di uno o più nodi.

- Un attacco *esterno* è facilmente identificabile se un nodo trasmette utilizzando un indirizzo IP o MAC non facente parte del *pool* di indirizzi destinato alla rete. Il controllo del solo indirizzo non è però sufficiente a garantire la difesa da questo tipo di attacco. Infatti un nodo potrebbe *sniffare* così il traffico di rete e ricavare le informazioni di cui necessita per impersonificare un appartenente alla rete. Per difendersi da questo tipo di attacco occorre che ogni nodo dimostri di conoscere un segreto; un segreto può essere, per esempio, una chiave privata con la quale firmare i propri messaggi, una chiave di gruppo nota solamente ai nodi autorizzati o una parte di una chiave distribuita.

- In un attacco *interno* invece, improvvisamente, un nodo che si è sempre comportato in modo corretto inizia a violare i protocolli di rete. Questo può essere ricondotto al problema dei *generali bizantini* [16]; la pericolosità di un attacco del genere sta nel fatto che è difficile stabilire con precisione se sia casuale, cioè causato da un malfunzionamento del nodo, oppure sia voluto, cioè sia l'effetto di un tentativo di attacco. Per il corretto funzionamento dell'intera rete, è opportuno implementare uno o più meccanismi per individuare ed escludere i nodi che si comportano in maniera bizantina; questa non è un'operazione facile ed è tuttora oggetto di ricerca; infatti, per esempio, in protocolli reattivi i nodi non possiedono l'intera informazione sulla topologia di rete e non possono quindi decidere quali nodi utilizzare per il transito dei loro pacchetti.

La divisione tra attacchi attivi e passivi si basa invece sul tipo di interazione che avviene tra nodo attaccante e rete:

- in un attacco passivo il nodo attaccante capta i messaggi provenienti dalla rete ma non ne immette di propri. La pericolosità di questo attacco è abbastanza limitata, infatti un avversario che si metta in ascolto sul canale radio può catturare i messaggi di routing transitanti ed estrarre da essi informazioni sulla topologia della rete e sull'identità dei nodi partecipanti; questi

dati però sono sensibili solo in particolari applicazioni delle manet come il campo militare. In protocolli reattivi non source routing come AODV, inoltre, i pacchetti dati non contengono informazioni sulla rete e anche i messaggi RREQ e RREP ne contengono solo di parziali; in un protocollo come DSR invece i messaggi dati contengono l'intero percorso di routing e portano perciò un contenuto di informazione maggiore; se pensiamo poi a protocolli proattivi l'informazione che può essere captata da un attaccante passivo riguarda l'intera topologia di rete.

- gli attacchi *attivi* sono maggiormente pericolosi. Infatti in questi attacchi un nodo, oltre a captare i messaggi provenienti dalla rete, ne invia di propri. Un nodo può quindi disturbare il corretto funzionamento della rete ed impedire ai nodi di usufruire dei servizi che essa fornisce. La maggior parte degli attacchi descritti in seguito possono essere catalogati come attivi.

Un attacco *Denial of Service* (DoS) è un attacco di tipo attivo; il suo scopo è di creare delle situazioni di *fuori servizio* ed è finalizzato a colpire un singolo nodo, un gruppo di nodi oppure l'intera rete. Come detto in 1.3.2 lo si può effettuare a diversi livelli dello stack protocollare.

Un'altra tipologia di attacco è il così detto *black hole*. Il nodo attaccante, violando le regole dei protocolli, potrebbe riuscire a diventare nodo intermedio per una o più rotte. Una volta conquistata questa posizione potrebbe scartare tutti i pacchetti dati che transitano attraverso di lui, rendendo la comunicazione impossibile. Questo attacco è particolarmente facile da portare in protocolli non sicuri. In AODV, per esempio, è sufficiente che un nodo generi messaggi RREP inserendo come Destination IP Address l'indirizzo del nodo di cui vuole controllare il traffico e inserisca un Destination Sequence Number superiore a quello attualmente in uso dal suddetto nodo. Così facendo tutti i nodi che ricevono questi messaggi RREP inseriscono il nodo attaccante come next hop verso il nodo vittima e le successive comunicazioni saranno così dirottate.

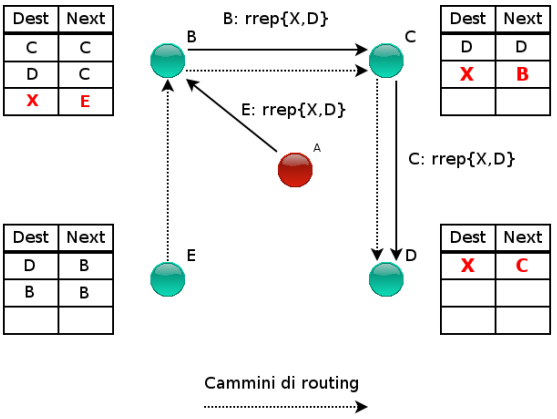
Una variante di questo attacco è il *gray hole*, che si differenzia dal precedente

in quanto tratta in modo diverso i singoli pacchetti. Alcuni vengono lasciati passare inalterati, altri vengono bloccati. Questo attacco è più difficile da rilevare in quanto i suoi sintomi possono essere confusi con quelli di un canale radio molto disturbato.

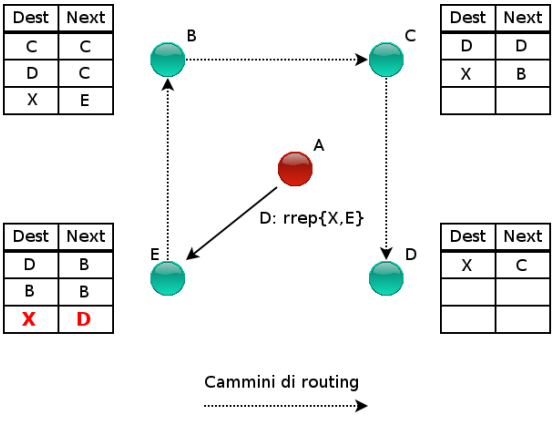
Un nodo può attuare un attacco "opposto" al black hole cioè, anziché attirare su di sé più rotte possibili, può decidere di non collaborare al routing della rete. L'attacco è molto semplice da attuare (basta che il nodo non invii alcun messaggio di routing), ma è anche di scarsa utilità in quanto non porta ad alcun malfunzionamento della rete ma solo ad un vantaggio per il nodo che risparmia potenza di calcolo e batterie.

Un altro attacco possibile è quello chiamato *loop*, portato da un nodo che, inviando pacchetti di routing contraffatti, tenti di creare delle rotte cicliche all'interno delle tabelle di routing dei nodi della rete. Questo attacco ha la doppia valenza di impedire il corretto recapito dei messaggi e di consumare risorse di rete poiché i pacchetti continuano a circolare. Ci possono essere due tipi di attacco loop, a seconda che il nodo attaccante sia esso stesso parte del loop o che non lo sia. Nella prima modalità l'attaccante deve *alimentare* il loop ad ogni passaggio instradando a sua volta i pacchetti; la seconda modalità invece è più pericolosa in quanto il nodo può immettere un solo pacchetto nel loop e questo continuerà a circolare finché non sarà scaduto il suo TTL. Anche questo attacco è facilmente attuabile contro una rete che utilizza come protocollo di routing AODV; per portarlo a termine il nodo non deve fare altro che impersonare due nodi della rete pubblicizzando rotte fasulle verso un nodo immaginario X. Un esempio è mostrato in figura 3.1

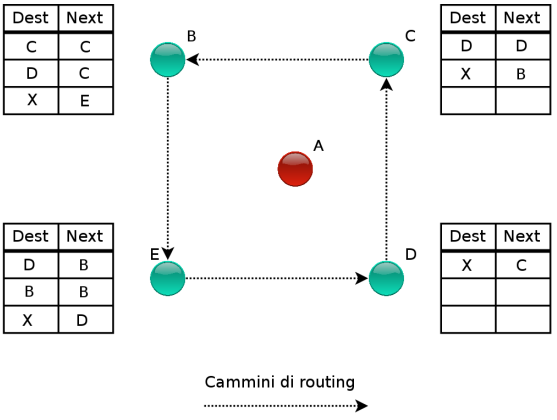
a - Il nodo attaccante A, impersonando il nodo E, forgia un messaggio RREP fasullo diretto a D nel quale pubblicizza l'esistenza di una rotta verso il nodo immaginario X. I nodi B, C, D di conseguenza propagano il messaggio e inseriscono nelle loro tabelle di routing la rotta verso X passante rispettivamente per E, B, C.



(a)



(b)



(c)

Figura 3.1: Attacco loop portato dal nodo A

b - A manda un'ulteriore messaggio fasullo a E, impersonando D, nel quale pubblicizza una rotta verso X. E modifica la sua tabella di routing aggiungendo una rotta per X passante per D.

c - il loop è stato creato. Ora A può inviare pacchetti dati verso X che verranno inoltrati lungo la rotta circolare fino allo scadere del loro TTL.

Vi è un'altra tipologia di attacco possibile in protocolli che implementano un meccanismo di rilevamento di nodi maligni; in questo caso infatti un nodo potrebbe far reputare un nodo fidato come malvagio. Si pensi ancora al protocollo AODV: il meccanismo di controllo della frequenza di invio delle RREQ può rivelarsi in questo caso un'arma a doppio taglio. Sebbene sia stata studiata per rilevare nodi mal funzionanti potrebbe portare all'esclusione dalle comunicazioni di un nodo legittimo; infatti, se un attaccante impersonasse un nodo X e inviasse a suo nome un grande quantitativo di RREQ in sequenza, il nodo X verrebbe escluso dalla rete sebbene non sia il responsabile del problema.

Un attacco ancora diverso è il *worm hole*. Questo attacco risulta essere molto pericoloso per il routing e, soprattutto, di difficile eliminazione. Per essere messo in opera necessita di due nodi complici (A e B) che dispongano di un collegamento privato tra essi (per esempio un collegamento cablato o un tunnel IP-over-IP). Il concetto alla base di questo attacco è quello di prelevare i pacchetti ad un capo del tunnel (A) e rispedirli all'altro capo (B). In base ai pacchetti che si inoltrano lungo il collegamento esclusivo e a quelli che invece vengono instradati normalmente dalla manet questo attacco può portare a diverse conseguenze:

- se si inoltrano tutti i pacchetti di routing per uno o più collegamenti si fa in modo che la rotta passante per i due nodi attaccanti sia considerata da tutti come la migliore.

- se invece si inoltrano attraverso il tunnel solo le richieste direttamente al destinatario ma si lasciano trasportare dalla manet le relative risposte la

conseguenza è che si impedirà a tutti i nodi distanti più di due hop dal nodo attaccato di costruire rotte verso di esso.

In letteratura sono presenti alcune contromisure contro questo tipo di attacco che si basano sul concetto di *packet leashes*. Attraverso i packet leashes (letteralmente i guinzagli dei pacchetti) si vuole aggiungere un controllo che possa stabilire se il percorso che sembra essere stato seguito da un pacchetto sia realmente quello effettuato. I packet leashes possono essere di due tipi:

spaziali se l'informazione che apportano riguarda la posizione del pacchetto. Per poter utilizzare questa tecnologia occorre che nella rete vi sia una certa sincronizzazione temporale in modo da poter calcolare con una certa precisione se il pacchetto ha viaggiato lungo un percorso irrealistico (per esempio aver percorso una distanza troppo elevata se confrontata con il tempo impiegato)

temporali se l'informazione è unicamente di tipo temporale. In questo caso viene approssimato il tempo di viaggio del pacchetto come differenza tra tempo di arrivo e di partenza; questo verrà poi confrontato con la lunghezza del percorso in hop per valutare se sia realistico o meno. In questo caso la sincronizzazione temporale deve essere molto elevata (mentre nel caso precedente, essendo presente anche un'informazione sulla posizione geografica dei nodi, è sufficiente una sincronizzazione blanda).

Queste due tipologie di difesa si basano entrambe sulla sincronizzazione temporale dei nodi. In ambito manet però sembra poco realizzabile in quanto un nodo può entrare e uscire dalla rete senza preavviso. Inoltre la prerogativa di queste reti, se non la loro caratteristica principale, è quella di essere spontanee e di rapida instaurazione, caratteristiche entrambe in contrasto con la richiesta di una precisa sincronizzazione temporale. Sulla stessa prerogativa si basano anche alcune soluzioni per la messa in sicurezza dei protocolli di routing proposte in letteratura che si vedranno in dettaglio nel paragrafo 3.2.4.

Un altro tipo di attacco è il *rushing attack* che può essere portato contro protocolli on-demand che utilizzano un meccanismo di soppressione dei messaggi

duplicati (come AODV). Un nodo avversario che invii velocemente una serie di messaggi di richiesta impersonando il nodo X può fare in modo che le successive richieste legittime da parte del nodo X vengano scartate come duplicati. Infatti i nodi che ricevono un messaggio di richiesta, per evitare di rielaborare pacchetti già precedentemente presi in considerazione, scartano i messaggi che ritengono duplicati basandosi sull'indirizzo del mittente e sul RREQ_ID della richiesta, numeri che il nodo attaccante può conoscere e utilizzare.

3.2 I protocolli di routing sicuri

Mentre l'attenzione dell'IETF è ancora puntata sulla standardizzazione di un protocollo di routing senza preoccuparsi, per il momento, della sicurezza, la comunità scientifica sta invece ponendo molta attenzione a questo problema proponendo soluzioni diverse ed eterogenee. Si esporranno però solo le maggiormente rilevanti. Prima di tutto si tratteranno le estensioni di sicurezza per tre protocolli la cui versione non sicura è in fase di studio da parte dell'IETF. Poi si tratteranno altri protocolli che utilizzano idee originali per la messa in sicurezza.

3.2.1 Secure Ad hoc On-demand Distance Vector routing (SAODV)

SAODV è un'estensione volta a garantire la sicurezza del protocollo AODV già da tempo in fase di studio. Le modalità di funzionamento dell'estensione sono raccolte in un *draft individuale* [17] proposto da Guerrero Zapata nell'ottobre 2001 e giunto nel febbraio 2006 alla quinta revisione. Si tratterà con attenzione questo protocollo poiché è la base della trattazione sperimentale di questa tesi.

Essendo basata su AODV, che è da tempo allo stato di *RFC*, SAODV ne eredita la stabilità. Inoltre è una delle prime proposte per la messa in sicurezza del routing nelle manet e per questo motivo ne esistono già anche alcune implemen-

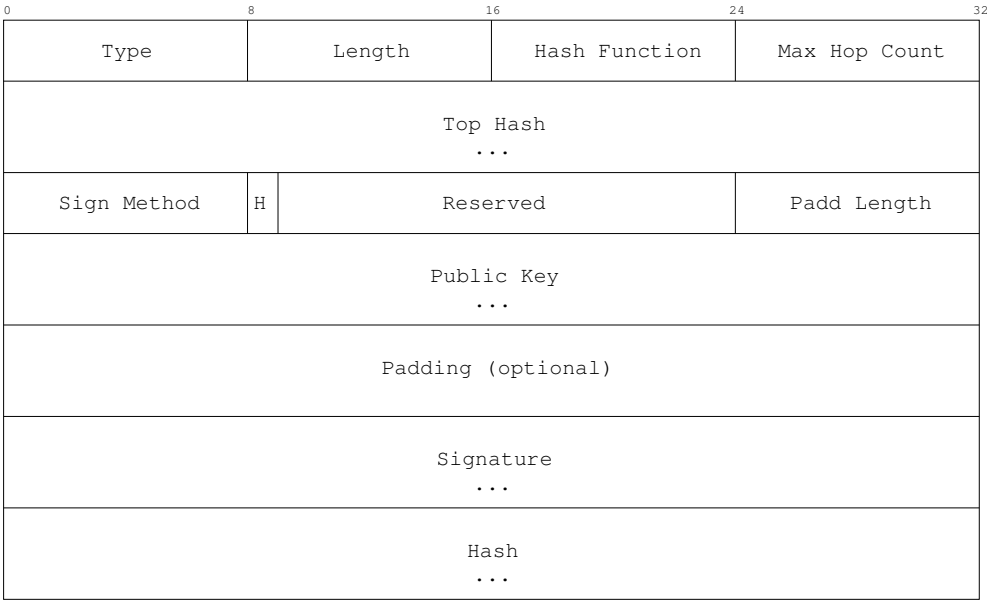


Figura 3.2: L'estensione SAODV per il messaggio RREQ

tazioni. Non ne esiste invece alcuna versione per ambienti simulati come ns2[1], opnet [2] o GloMoSim[3].

Questo protocollo implementa un meccanismo di protezione dei messaggi di routing che garantisce la corretta instaurazione delle rotte anche in campo *ostile*, mentre non si occupa in alcun modo del successivo scambio di dati, che potrà essere reso sicuro utilizzando protocolli specifici. Il meccanismo messo a punto in SAODV si basa sull'osservazione che lungo il tragitto dei messaggi RREQ e RREP del protocollo AODV solo alcuni campi vengono modificati ad ogni hop, mentre la maggior parte resta invariata lungo tutto il percorso. Per questo motivo si possono attuare due tipi di protezione diversi per le due tipologie di campo. I due metodi di protezione di questi campi verranno descritti nei paragrafi 3.2.1.1 e 3.2.1.2.

Nelle figure 3.2, 3.3 e 3.4 viene mostrato il formato delle estensioni SAODV per i messaggi RREQ, RREP e RERR rispettivamente.

[1]http://www.isi.edu/nsnam/ns/

[2]http://www.opnet.com/

[3]http://pcl.cs.ucla.edu/projects/glomosim/

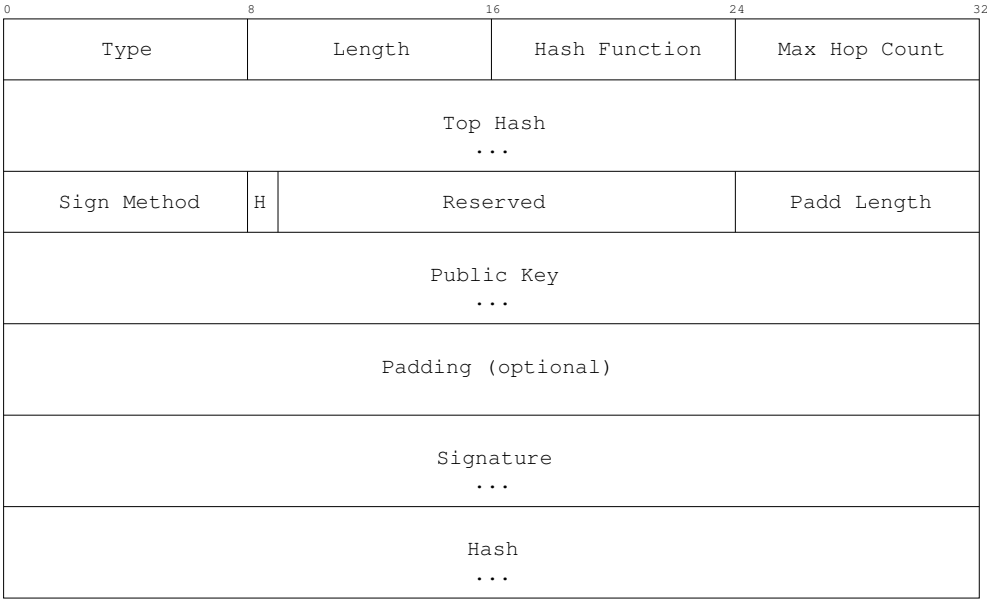


Figura 3.3: L'estensione SAODV per il messaggio RREP

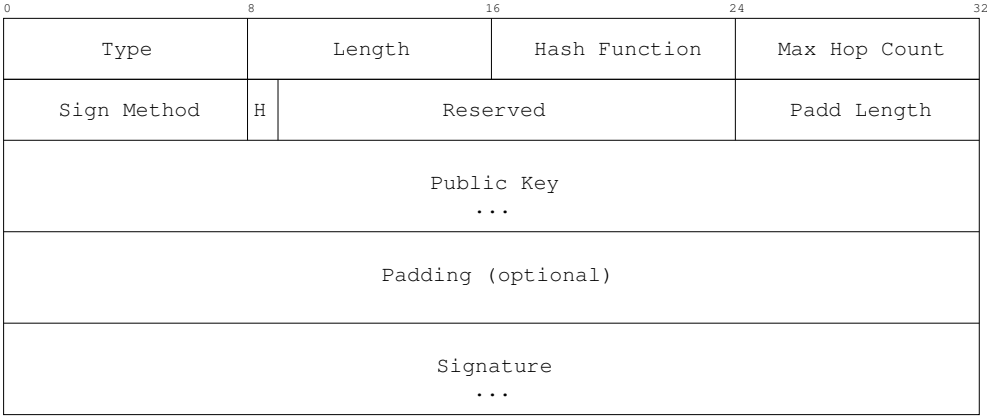


Figura 3.4: L'estensione SAODV per il messaggio RERR

3.2.1.1 Protezione dei campi fissi nei messaggi RREQ e RREP

Il meccanismo di protezione dei campi fissi nei messaggi AODV si basa sulla crittografia asimmetrica. Un nodo che genera la RREQ o la RREP utilizza la sua chiave privata per generare una firma dell'intero messaggio[4], che verrà poi inserita nell'apposito campo dell'estensione SAODV. I campi *variabili* vengono azzerati durante l'operazione di firma in modo da non compromettere la successiva verifica della firma. Ogni nodo lungo il percorso del pacchetto può dunque verificare sia l'integrità che l'autenticità delle informazioni controllando la firma con la chiave pubblica del nodo sorgente. Vi è una precisazione da fare riguardo ai campi considerati fissi: in AODV il capo Destination Sequence Number non è in realtà immutabile lungo il percorso. Questo perché ogni nodo intermedio che conosca un numero di sequenza più aggiornato di quello presente nel messaggio può modificare il contenuto del campo. In SAODV questo non è più possibile poiché si andrebbe contro la "filosofia" del protocollo che è quella di considerare vere le informazioni su di un nodo solo se queste sono fornite dal nodo stesso, e perciò firmate da esso.

3.2.1.2 Protezione del campo hop count

L'hop count, ovviamente, è un campo che viene modificato ad ogni hop lungo il percorso del messaggio SAODV. Per questo motivo non può essere firmato come avviene per i campi fissi. Occorre però proteggere anche questo campo per impedire ad un nodo intermedio di aumentarlo (rendendo così la rotta risultante più lunga di quanto in realtà sia) o diminuirlo (rendendola più corta e favorendo attacchi di tipo black o gray hole come visto in 3.1) a piacere. In SAODV a questo scopo si utilizzano le *catene di hash* [18]. Alla creazione di un messaggio un nodo deve:

- generare un numero casuale detto *seme*;

[4] si firma sia la parte AODV che tutti i campi che precedono il campo Signature dell'estensione SAODV

- scegliere il valore del campo Time to live dell'header IP (come descritto in 2.4.1.2 per AODV) e impostare Max Hop Count allo stesso valore.
- copiare il valore del seme nel campo Hash;
- scegliere una funzione di hash (h) da utilizzare[5] e impostare il campo HashFunction di conseguenza;
- calcolare il valore TopHash applicando MaxHopCount volte la funzione di hash al seme, cioè:

  $TopHash = h^{MaxHopCount}(seme)$

Durante il percorso del pacchetto ogni nodo intermedio deve:

- decrementare il campo TTL dell'header IP;
- incrementare il valore del hop count del pacchetto AODV;
- applicare la funzione di hash al campo Hash dell'estensione SAODV $Hash = h(Hash)$.

In questo modo in ogni punto del percorso distante n hop dal nodo origine dei messaggi il campo Hash avrà valore:

$Hash = h^n(seme)$

A questo punto ogni nodo intermedio può verificare l'integrità del campo Hop Count; infatti, applicando al campo Hash la funzione di hash $MaxHopCount - HopCount$ volte deve ottenere di nuovo il valore TopHash.

quindi se:

$TopHash == h^{MaxHopCount-HopCount}(Hash)$ allora il pacchetto è valido; se l'uguaglianza non è soddisfatta il pacchetto è stato corrotto.

Grazie alla proprietà di unidirezionalità delle funzioni di hash, nessun nodo può diminuire il conteggio del numero di salti effettuati. Un nodo attaccante

[5] nel draft ne vengono proposte alcune tra cui SHA1 che deve essere supportata da tutte le implementazioni

può tuttalpiù decidere di inoltrare un pacchetto senza aggiornare i valori dell'hop count, rendendo così la rotta più breve di un salto, oppure può calcolare la funzione di hash più di una volta aumentando così il valore dell'hop count; il cammino risulta quindi più lungo e perciò non ci sono vantaggi per il nodo attaccante. È anche inutile per un attaccante aggiornare solo uno dei due campi HopCount o Hash perché così genera un pacchetto non valido che viene scartato dal nodo successivo.

3.2.1.3 Protezione dei messaggi di errore

I messaggi di errore, a differenza degli altri messaggi, hanno unicamente validità "hop by hop", infatti ogni nodo che riceve un messaggio RERR può modificarlo aggiungendo altre destinazioni non raggiungibili prima di reinoltrarlo in unicast o in broadcast, ma sempre con TTL uguale ad 1. In SAODV quindi si è optato per firmare l'intero messaggio di errore. Ogni nodo che riceve una RERR deve verificare la firma del mittente; a questo punto crea un nuovo messaggio RERR aggiungendo altre eventuali rotte non raggiungibili, lo firma a sua volta e lo trasmette in unicast o in broadcast ai suoi vicini. Con questo meccanismo viene garantito il non ripudio per i nodi vicini, ma non per il nodo che ha generato il primo messaggio RERR. Questa scelta è in accordo con il funzionamento del protocollo AODV perché ogni nodo ha visibilità soltanto dei suoi vicini e non sa dove si trovano gli altri nodi della rete; infatti per raggiungere un nodo che si trova a più di un hop di distanza, un nodo AODV sa da quale vicino deve passare, ma non sa quali altri nodi verranno attraversati dai suoi pacchetti. Quindi, se il nodo X genera un messaggio RERR, solo i suoi vicini hanno interesse nel verificarne la firma e, se il nodo X genera troppi RERR, i suoi vicini lo possono escludere perché lo reputano un nodo corrotto o mal funzionante; anche se un nodo lontano sapesse che X è corrotto, non saprebbe come sfruttare questa informazione perché non sarebbe in grado di escluderlo dalle operazioni di routing.

Secondo l'RFC di AODV, il nodo che riceve un messaggio RERR deve con-

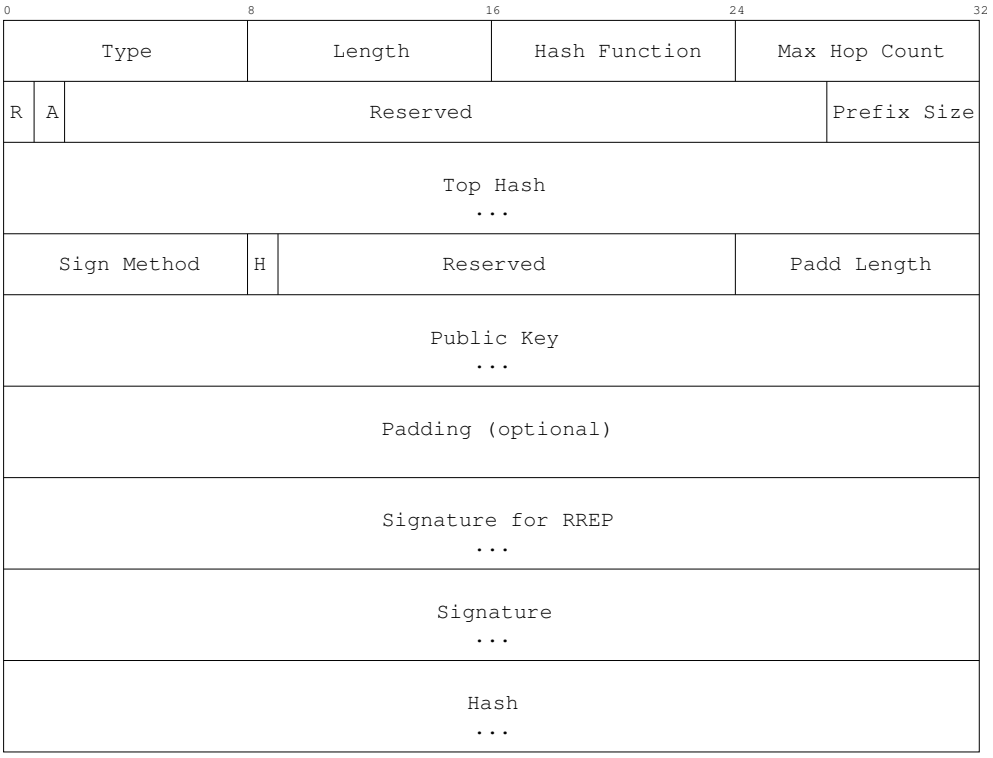


Figura 3.5: L'estensione SAODV per il messaggio RREQ con doppia firma

siderare valido ogni Destination Sequence Number in esso contenuto; purtroppo però tale valore non è generato dal nodo destinazione, ma da uno dei nodi intermedi, i quali teoricamente possono scegliere un valore qualsiasi. Per questo motivo in SAODV è consigliato di ignorare tali valori perché non provengono dal nodo legittimo e non sono firmati da esso, il che va contro la politica di SAODV per la quale solo il nodo stesso è garante per le informazioni che lo riguardano.

3.2.1.4 Il meccanismo di doppia firma

Il metodo di autenticazione e di verifica dell'integrità dei messaggi tramite crittografia asimmetrica descritto nel paragrafo precedente non permette al protocollo di usufruire del meccanismo di collaborazione tra nodi che è alla base di AODV, non permette cioè ai nodi intermedi di poter rispondere a richieste in caso essi abbiano le informazioni necessarie per farlo. In SAODV infatti per esse-

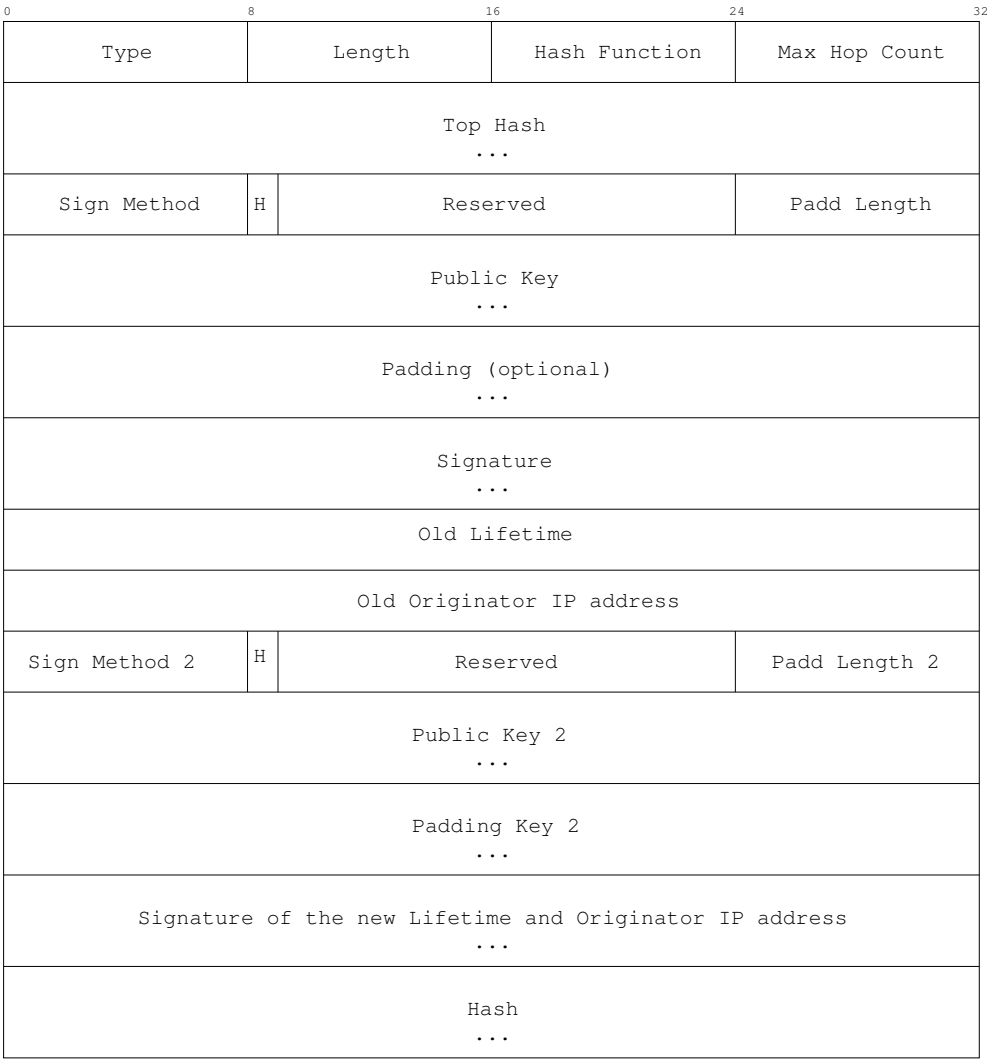


Figura 3.6: L'estensione SAODV per il messaggio RREP con doppia firma

re considerata valida una RREP deve contenere la firma del nodo destinazione[6]; per permettere la collaborazione è quindi stato proposto in SAODV l'utilizzo del *meccanismo di doppia firma*. Quando un nodo genera un messaggio di RREQ può decidere di permettere ai nodi che lo riceveranno di rispondere a successive eventuali RREQ a lui destinate. Per permettere questo il nodo sorgente allega al messaggio (utilizzando un'estensione diversa di SAODV rispetto a quella a singola firma, mostrata in figura 3.5 e chiamata RREQ *Double Signature Extension*, RREQ-DSE), oltre alla firma del messaggio RREQ, un'ulteriore firma di una RREP *fittizia*[7] da lui generata; i nodi lungo il percorso di diffusione della RREQ memorizzano nella tabella di routing oltre alle canoniche informazioni anche questa seconda firma che utilizzeranno per rispondere alle successive RREQ destinate al nodo sorgente della RREQ con doppia firma. Con questo meccanismo i nodi intermedi possono rispondere a richieste per le quali hanno le informazioni necessarie (oltre alla rotta valida anche la seconda firma prelevata da una RREQ con estensione a doppia firma del nodo destinazione) senza alterare il meccanismo di sicurezza. Infatti un nodo che riceve una RREP generata da un nodo intermedio può essere certo che quest'ultimo abbia realmente una rotta verso il nodo richiesto in quanto altrimenti non sarebbe potuto essere in possesso della firma della RREP fittizia generata dal nodo destinazione della RREQ. Per consentire a questo meccanismo di funzionare il nodo intermedio deve aggiungere alla RREP anche i campi utilizzati dal nodo sorgente della RREP fittizia che il nodo origine della RREQ non può dedurre facendo così in modo che possa verificarne la firma. Per fare questo utilizza un'estensione diversa di SAODV detta RREP *Double Signature Extension* (RREP-DSE) mostrata in figura 3.6.

La seconda firma presente nel messaggio RREQ-DSE e memorizzata dai nodi servirà anche in caso il nodo sorgente richieda la generazione di messaggi RREP gratuiti.

[6] il nodo per il quale si richiede la rotta

[7] chiamiamo fittizia questa RREP perché è generata senza che vi sia una vera e propria richiesta

3.2.1.5 Overhead introdotto da SAODV

Ogni nodo deve possedere una coppia di chiavi pubblica/privata e, inoltre, tutti nodi devono poter conoscere le chiavi pubbliche degli altri nodi. Devono anche essere certi della corretta associazione tra chiave e identità del nodo. In SAODV viene proposto un meccanismo per ottenere l'indirizzo IP attraverso un algoritmo applicato alla chiave pubblica [19]; in questo modo ogni nodo può verificare in modo semplice la corretta associazione indirizzo IP/chiave. Resta comunque il problema di come assegnare le chiavi e di come distribuirle nella rete.

SAODV non richiede dei messaggi ulteriori rispetto a quelli necessari al normale funzionamento di AODV; i messaggi di SAODV sono però di dimensioni maggiori perché comprendono l'estensione che è relativamente pesante contenendo campi come hash, firma e chiave.

Dal punto di vista computazionale i costi aumentano, infatti:

- il nodo sorgente deve generare una firma e calcolare MaxHopCount hash;
- ogni altro nodo deve verificare una firma, calcolare MaxHopCount-HopCount hash e calcolare un ulteriore hash se inoltra il messaggio.

Se vengono utilizzati messaggi a doppia firma il carico aumenta ancora, infatti:

- il nodo sorgente deve generare due firme e calcolare MaxHopCount hash;
- ogni nodo intermedio deve verificare due firme, calcolare MaxHopCount-HopCount hash e calcolare un ulteriore hash se inoltra il messaggio.
- ogni nodo intermedio che risponde ad una RREQ deve verificare due firme e calcolarne una (due se è richiesto l'uso di RREP gratuite) e calcolare MaxHopCount-HopCount hash

3.2.2 ARAN, AUthenticated Routing for Ad hoc Network

ARAN [20] è un protocollo reattivo che non è un'estensione di AODV, ma che utilizza gli stessi tipi di messaggi (RREQ, RREP e RERR) per il suo funzionamento.

Il nodo sorgente S che voglia trovare una rotta verso la destinazione D inizia una route discovery:

- S trasmette in broadcast il seguente pacchetto

  $[RDP, IP_d, cert_s, N_s, t]_{k_s}$

  contenente un identificativo del tipo di messaggio RDP (Route Discovery Packet), l'indirizzo IP del nodo per il quale si richiede la rotta IP_d, il certificato del nodo sorgente $cert_s$, un numero casuale N_s e un timestamp t. Il tutto viene firmato con la chiave privata del nodo S (k_s).

- un nodo intermedio (nodo A) controlla la firma sul pacchetto e aggiorna la sua tabella di routing (tenendo in memoria gli indirizzi dei nodi sorgente, destinazione ed il vicino dal quale il pacchetto è arrivato, e considera la riga come temporanea[8]); dopodiché genera il seguente nuovo pacchetto:

  $[[RDP, IP_d, cert_d, N_d, t]_{k_s}]_{k_a}, cert_a$

  In pratica è lo stesso pacchetto ricevuto dal nodo sorgente, ulteriormente firmato dal nodo A; in coda al pacchetto è inserito anche il certificato del nodo A;

- un successivo nodo B che riceve il pacchetto dal nodo A, ripete gli stessi passi descritti al punto precedente. Ovviamente in questo caso il nodo deve verificare due firme (con la chiave pubblica di A e poi con quella del nodo sorgente) prima di poter considerare valido il contenuto. A questo

[8] il percorso tra sorgente e destinazione non è ancora stato completato ed il nodo aspetta un'eventuale reply per considerarlo valido

punto elimina la firma ed il certificato del nodo precedente[9] e trasmette in broadcast il seguente pacchetto:

$[[RDP, IP_d, cert_s, N_s, t]_{k_s}]_{k_b}, cert_b$

- ricevuto il messaggio, il nodo destinazione D risponde con un messaggio Route Reply (RREP), anch'esso firmato; il pacchetto ripercorre in direzione opposta lo stesso percorso (in unicast) e tutti i nodi intermedi aggiornano le loro tabelle di routing rendendo valide le rotte inserite nel percorso di andata; analogamente ai messaggi RDP anche i RREP vengono firmati dai nodi intermedi con il medesimo meccanismo.

Sono previsti messaggi di errore per segnalare un eventuale malfunzionamento di un link; il nodo che genera un messaggio di errore lo deve firmare e trasmettere ai nodi sorgente interessati[10], per avvisarli che una o più destinazioni non sono ora raggiungibili. Tale messaggio di errore viene inoltrato senza alcuna modifica (a differenza del comportamento di AODV) perché in ARAN è importante memorizzare quale nodo ha generato il messaggio; un nodo che emette "troppi" messaggi di errore potrebbe essere un nodo attaccante, quindi è classificabile come sospetto. Ogni nodo tiene quindi memorizzata una lista di nodi sospetti, ed ogni messaggio generato da un nodo sospetto non viene né elaborato né inoltrato. Se un nodo A considera un suo vicino B inaffidabile, può decidere di escluderlo dai cammini di routing; se invece il nodo non fidato C è più lontano non è garantita la sua esclusione dai cammini di routing, poiché solo i vicini hanno la visibilità su di lui. Per questo motivo il non ripudio dei messaggi di errore ha un'utilità limitata, a differenza di quanto avviene con i protocolli source routing, dove il nodo sorgente decide completamente il cammino che devono compiere i suoi pacchetti. In questa ottica è più saggia la scelta fatta in SAODV di mantenere la validità dei messaggi di errore unicamente per i nodi vicini.

[9] non essendo un protocollo source routing i nodi hanno visibilità solo dei diretti vicini, non sono quindi considerate le informazioni sui nodi più lontani

[10] cioè quelli che hanno quel link all'interno di almeno una rotta

ARAN non è un vero e proprio protocollo distance vector in quanto la scelta sul cammino migliore viene fatta in base al tempo di arrivo del pacchetto e non in base al numero di hop[11]; si presuppone infatti che i pacchetti che arrivano prima abbiano seguito un percorso migliore.

3.2.2.1 Overhead introdotto da ARAN

ARAN richiede molte operazioni crittografiche asimmetriche: il nodo sorgente S deve effettuare una firma sul messaggio, i nodi vicini di S devono verificare una firma e generarne una, mentre tutti gli altri nodi devono verificare 2 firme e generarne una. Il peso di questo protocollo è quindi maggiore per i nodi intermedi che per il nodo sorgente; un protocollo efficiente dovrebbe invece caricare il meno possibile i nodi intermedi che sono molti e lasciare eventualmente un maggior carico computazionale ai nodi estremi che sono i due nodi coinvolti nella comunicazione.

Rispetto ad AODV non richiede messaggi aggiuntivi, anche se la dimensione degli stessi è maggiore a causa delle firme e dei certificati che vi sono inseriti. In [20] sono presentati alcuni risultati delle simulazioni effettuate con GloMoSim; le simulazioni sono state effettuate prima con 20 nodi e poi con 50 nodi; sono state utilizzate chiavi a 512 bit e nel simulatore è stato introdotto un ritardo di 8.5 ms per la generazione e di 0.5 ms per la verifica di una firma. In assenza di nodi attaccanti sono state misurate le prestazioni di ARAN rispetto ad AODV ed è stato trovato che coincidono i valori di:

- numero medio di pacchetti arrivati a destinazione;
- lunghezza media dei percorsi di routing;
- overhead misurato in pacchetti di segnalazione;
- ritardo end-to-end dei pacchetti contenenti i dati.

[11] infatti non esiste nessun campo a riguardo nei pacchetti di richiesta o risposta

A causa delle operazioni crittografiche aggiuntive, ARAN ha un maggiore overhead misurato in byte ed un maggior ritardo end-to-end durante la fase di route discovery.

Una seconda simulazione riguarda il caso in cui una parte dei nodi della rete si comporta in maniera non corretta, azzerando il campo hop count di tutti i messaggi di segnalazione che transitano da essi, con lo scopo di essere presenti nei cammini di routing scelti dagli altri nodi. Nel protocollo ARAN non esiste il campo hop count perché vengono scelti sempre i pacchetti di segnalazione più veloci (quelli che arrivano per primi), per questo motivo ARAN si comporta meglio di AODV durante questo tipo di attacco[12]. In questa simulazione sono stati misurati la lunghezza dei cammini di routing e la percentuale di pacchetti che transitano attraverso i nodi attaccanti. In AODV si nota un (limitato) allungamento dei cammini di routing ed una maggiore percentuale di pacchetti transitanti attraverso i nodi attaccanti.

3.2.3 Secure DSR (SDSR)

Questo protocollo [21] è un'estensione sicura del protocollo DSR. La messa in sicurezza si basa su due presupposti: l'esistenza di soli link bidirezionali e il possesso da parte di ogni nodo di una coppia di chiavi pubblica/privata (detta MANET-ID) firmata da un'entità di certificazione in modo da non consentire la creazione di nuove identità. A differenza degli altri protocolli visti finora, SDSR integra un meccanismo di distribuzione delle chiavi. Si supponga che il nodo S voglia entrare in comunicazione con il nodo D. La comunicazione si svolge in questo modo:

1. S crea un messaggio di route request contenente il suo indirizzo IP, l'indirizzo IP di D, un identificativo unico della richiesta (ID), una chiave pubblica di tipo Diffie-Hellmann ($DHPK_s$), un numero casuale N_1 e una lista iniziale

[12] si tenga presente anche che AODV non è stato progettato per ovviare a nessun tipo di attacco

di nodi da percorrere che per il momento comprende solo il nodo S stesso ($SR(S)$). A questo punto il nodo crea una firma di tutto il messaggio con il suo MANET-ID. Questa firma protegge tutti i dati statici presenti nella richiesta (S,D,ID,$DHPK_s$); D potrà verificarne l'integrità per mezzo della firma allegata da S.

2. ogni nodo intermedio K lungo la rotta verso D aggiunge se stesso alla lista di nodi attraversati SR(S,K..) e trasforma il numero casuale N_1 in N_2. Per fare questo il nodo K cifra N_1 con una chiave simmetrica che solo lui conosce. Questa procedura viene detta *nonce trasformation*.

3. quando la richiesta raggiunge D questo genera un messaggio di route reply che contiene tutte le informazioni statiche della request più l'intero percorso creato durante la discovery, la firma che era stata generata da S e una nuova firma fatta da D. Questa nuova firma include i campi fissi ma anche la rotta e la firma di S. In questo modo lungo il percorso inverso le informazioni sulla rotta sono protette. Infine D allega anche la sua chiave pubblica Diffie-Hellman [22] e una chiave cifrata con la chiave di sessione SK_{sd}.

4. tutti i nodi K lungo il percorso inverso annullano la trasformazione fatta al numero casuale N (decifrandolo con la chiave simmetrica), allegano una chiave Diffie-Hellman e un chiave cifrata con la chiave di sessione SK_{sk}.

Il meccanismo di *nonce trasformation* permette di identificare eventuali modifiche alla rotta durante il percorso *di andata* della request. Infatti il nodo S si aspetta di ritrovare N_1 nel messaggio di reply; se questo non accade significa che il messaggio è stato alterato. Per prevenire modifiche durante il percorso *di ritorno* invece viene utilizzata la firma del nodo D che può essere verificata dai nodi per testare l'integrità del pacchetto.

Le chiavi Diffie-Hellman scambiate durante la procedura di reply fanno in modo che i nodi possano scambiarsi con il protocollo di scambio di Diffie-Hellman delle chiavi di sessione per rendere sicuro il successivo scambio di dati.

Per autenticare i nodi intermedi invece vengono utilizzate le chiavi cifrate allegate dai nodi durante il percorso di ritorno. In questo modo si può anche verificare che siano state scambiate le corrette chiavi per la sessione dati.

3.2.3.1 Overhead introdotto da SDSR

In SDSR i messaggi che vengono scambiati tra i nodi sono molto più pesanti che nei protocolli visti finora. Questo perché, oltre al carico dovuto al fatto di essere un protocollo source routing, vi è anche un maggiore uso di firme e chiavi. Il messaggio di richiesta contiene la chiave Diffie-Hellman e la firma del nodo S; i messaggi di risposta, invece, contengono una firma e una chiave Diffie-Hellman e una chiave cifrata per ogni nodo attraversato. Inoltre questo protocollo richiede anche computazionalmente un maggiore dispendio di risorse in quanto il nodo sorgente deve effettuare un'operazione di firma asimmetrica e ogni nodo sul percorso della richiesta deve fare una verifica e un'operazione di cifratura simmetrica (per la modifica del nonce). Durante la risposta invece il nodo destinazione deve effettuare una firma e una cifratura e ogni nodo lungo il percorso di ritorno deve effettuare un'operazione di decifratura e una di cifratura.

3.2.4 Altri protocolli di routing sicuri

Hu, Perrig e Johnson propongono Ariadne [23], un protocollo source routing nel quale si utilizzano gli hash al posto degli algoritmi crittografici asimmetrici per alleggerire il carico dei nodi. Nell'articolo si presuppone che tutti i nodi della rete ad hoc siano sincronizzati in modo da utilizzare TESLA [24, 25] per gestire le chiavi necessarie all'autenticazione degli hash. Durante il flooding dei messaggi RREQ, ogni nodo intermedio aggiunge il proprio indirizzo IP in coda alla lista dei nodi attraversati ed aggiorna il campo Hash nel seguente modo: il nodo intermedio X preleva il campo hash dal messaggio RREQ ricevuto (sia h_i tale valore) e calcola il nuovo valore dell'hash $h_{i+1} = H[IP_x, h_i]$, dove H[·] è una funzione di hash e IP_x è l'indirizzo IP del nodo X. Inoltre il nodo intermedio

aggiunge un MAC sull'intero messaggio, autenticandolo con la propria chiave TESLA (che non è ancora stata resa pubblica). Una volta arrivato il messaggio RREQ a destinazione, il nodo destinatario *congela* la lista dei nodi intermedi con un MAC[13] e risponde con un messaggio RREP. Il messaggio RREP percorre il cammino inverso a quello seguito dal messaggio RREQ ed ogni nodo intermedio accoda la chiave TESLA che aveva utilizzato per calcolare il MAC nel messaggio di richiesta. Quando il nodo riceve la RREP, controlla la validità delle chiavi TESLA, la correttezza del MAC prodotto dal nodo destinazione e la correttezza di tutti i MAC dei nodi intermedi. Se tutti i controlli vanno a buon fine, il nodo sorgente aggiunge il percorso di routing nella sua tabella. In Ariadne sono previsti anche dei messaggi di errore simili a quelli del protocollo DSR, che devono essere firmati dal nodo che li genera, per garantire autenticità, integrità e non ripudio. Anche nel protocollo Ariadne per ogni route discovery è il solo nodo sorgente ad imparare un cammino di routing.

Guerrero Zapata, lo stesso autore di SAODV, propone anche un'estensione per il protocollo DYMO (paragrafo 2.3.1): SDYMO. Questa estensione si basa fortemente sull'estensione già proposta per SAODV. Infatti si utilizzano firme per autenticare e proteggere la parte *fissa* dei messaggi di routing mentre si utilizzano le catene di hash per proteggere i campi variabili come l'hop count. Ogni nodo che vuole cercare una rotta verso una destinazione D crea un messaggio RREQ e lo firma utilizzando la sua chiave privata. Il messaggio viene spedito in flooding. I nodi intermedi prima di aggiornare la propria tabella di routing e inoltrare il messaggio verificano tale firma. Una volta che il messaggio giunge al nodo D viene creata una RREP, firmata da D con la sua chiave privata e inviata in unicast lungo la rotta appena creata. Per quanto riguarda i messaggi di route error questi vengono firmati dal nodo da cui sono generati e mandati in broadcast con TTL uguale a uno. I nodi riceventi verificano la firma prima di modificare le proprie tabelle di routing e creano un nuovo messaggio di errore aggiungendo eventuali

[13] viene utilizzata per questo una chiave condivisa tra nodo sorgente e destinazione

nuove destinazioni non raggiungibili e firmandolo a loro volta prima di rispedirlo in broadcast.

Il protocollo SEAD [26] si propone di autenticare sia il contatore del numero di salti (hop count) che il contatore del numero di sequenza (sequence number) utilizzando tecniche crittografiche come le catene di hash che sono maggiormente efficienti rispetto alla crittografia asimmetrica. Per fare questo occorre che ogni nodo, per creare la propria catena di hash, stimi a priori il valore massimo che l'hop count può assumere (*m-1*)[14] e il numero di sequenza massimo ammissibile (k). A questo punto il nodo può generare la sua catena di hash calcolando: $h^{k \cdot m}(seed) = h_0 = v_{k \cdot m}$. Il valore h_0 va autenticato e mandato in broadcast nella rete per renderlo noto a tutti gli altri nodi. Quando il nodo vuole diffondere un aggiornamento di routing con numero di sequenza i e hop count iniziale nullo, costruisce un messaggio di segnalazione e vi inserisce il seguente valore: $h^{(k-i) \cdot m}(seed) = v_{(k-i) \cdot m}$ che indica: *hopcount=0* e *numero di sequenza=i*. Un nodo intermedio che riceve tale messaggio deve verificare i valori ricevuti di HopCount e numero di sequenza applicando al valore ricevuto la funzione di hash i volte e riottenendo, se l'informazione è corretta, h_0. Prima di inoltrare il messaggio applica nuovamente la funzione di hash. Per creare una nuova catena di hash, il nodo sorgente deve calcolare un elevato numero di hash fino ad arrivare al valore h_0, dopodiché lo deve firmare. Tale valore firmato deve poi essere trasmesso a tutti i nodi della rete, creando così un overhead di traffico. Ogni nodo che riceve un messaggio di routing deve calcolare fino a m funzioni di hash per autenticare il valore ricevuto; inoltre, prima di ritrasmettere il messaggio, deve calcolare un hash per aggiornare tale valore.

[14] lo si fa stimando la grandezza della rete

Capitolo 4

Valutazione ed ottimizzazione delle prestazioni di SAODV

In un lavoro di tesi precedente [27] è stato fatto un paragone tra i protocolli AODV e SAODV valutando alcuni parametri fondamentali per le prestazioni della rete. In particolare sono stati presi in considerazione:

- il ritardo nella creazione di una rotta;
- il ritardo tra l'invio di un pacchetto dati e l'arrivo di quest'ultimo a destinazione;
- la percentuale di connessioni effettuate rispetto a quelle richieste;
- l'overhead in pacchetti e byte introdotto dal protocollo.

Le valutazioni sono state effettuate facendo variare diversi parametri della rete e del protocollo stesso, come

- i tempi impiegati dal nodo per effettuare firme e verifiche;
- la mobilità dei nodi in termini di velocità media e tempi di pausa tra un movimento e il successivo[1];

[1] è stato utilizzato il modello *random waypoint*

- il raggio di copertura dei nodi.

I risultati ottenuti hanno mostrato un comportamento migliore da parte del protocollo non sicuro per quanto riguarda tutti i parametri valutati con differenze più o meno marcate in base al tipo di scenario simulato. Questi risultati sono in gran parte prevedibili in quanto le procedure di sicurezza introducono un overhead notevole. Questo overhead d'altra parte non porta a vantaggi visibili in ambienti privi di utenti *ostili* come quelli simulati finora. In scenari in cui vi siano dei tentativi di attacco alla rete invece il protocollo sicuro potrebbe risultare migliore rispetto a quello non sicuro. Nonostante SAODV sia progettato per la sicurezza non vanno trascurate le prestazioni soprattutto alla luce del fatto che, in situazioni *normali*, per la maggior parte del tempo di utilizzo il protocollo non è soggetto ad attacchi.

L'attenzione di questo lavoro è stata dunque puntata sull'ottimizzazione delle prestazioni del protocollo per tentare di limitare il gap introdotto dalle funzionalità di sicurezza.

4.1 Il problema

SAODV rispecchia molto da vicino la modalità di funzionamento del protocollo da cui deriva, conservandone le principali caratteristiche. In particolare è stata preservata la collaborazione tra i nodi che permette, in AODV, ai nodi intermedi lungo un percorso di poter rispondere se hanno le informazioni necessarie per farlo. In questo modo si hanno principalmente due vantaggi:

- un vantaggio dal punto di vista del carico della rete, in special modo se si utilizza la tecnica dell'*expanding ring*, limitando così infatti il flooding dei messaggi di richiesta nell'intera rete;

- un vantaggio in termini di tempo impiegato per effettuare una discovery in quanto la richiesta non deve percorrere l'intero percorso ma solo la parte di esso fino al nodo intermedio che si appresta a rispondere.

Per conservare questi vantaggi introdotti da AODV, SAODV utilizza il meccanismo della doppia firma visto nel paragrafo 3.2.1.4. Questo meccanismo però è particolarmente pesante dal punto di vista computazionale e, come logica conseguenza, anche da quello dei ritardi con i quali una rotta viene creata; per questo motivo non è sempre vantaggioso, come invece lo era nel protocollo non sicuro, permettere ai nodi intermedi di rispondere. Infatti, se si utilizza *sempre* il meccanismo a doppia firma, un nodo intermedio che vuole rispondere ad una richiesta deve:

- verificare le due firme contenute nel messaggio di richiesta;
- creare una firma per proteggere il messaggio di risposta;
- creare un'ulteriore firma per il messaggio di risposta gratuito da inviare al nodo destinazione (in caso sia stato posto il flag G[2] ad uno)[3].

Oltre a questo anche il nodo sorgente è soggetto ad un carico maggiore dovendo effettuare due firme, una per proteggere il messaggio di richiesta ed una per il messaggio di risposta fittizio che verrà utilizzato dai nodi intermedi per rispondere a suo nome.

Se invece si lascia proseguire il flooding del messaggio permettendo solo al nodo destinazione di rispondere:

- il nodo sorgente deve effettuare una sola firma;
- i nodi intermedi devono verificare una firma;
- il nodo destinazione deve verificare una firma ed effettuarne una.

In questo lavoro sono state proposte due possibili ottimizzazioni per il protocollo SAODV con l'intento di migliorarne le prestazioni senza per questo compromettere la sicurezza del protocollo: una versione del pretocollo adattativo (descritta nel paragrafo 4.4.1) e il forward anticipato (descritte nel paragrafo 4.4.2).

[2] si veda 2.4.2.2

[3] se, come lecito, si ipotizzano connessioni bidirezionali tra i nodi è auspicabile per il nodo sorgente porre il flag G uguale ad uno

Si sono poi simulate le due modifiche singolarmente per verificarne gli effetti su diversi scenari e con diversi parametri. Infine sono state effettuate delle simulazioni attivando entrambe le ottimizzazioni per valutare le possibili interazioni tra le due.

4.2 Strumenti e parametri utilizzati

Per effettuare le simulazioni è stato utilizzato *Network Simulator 2* (ns2); ns2 è un simulatore ad eventi discreti, sviluppato per simulare sia reti fisse che reti wireless con nodi mobili come le manet; è un progetto open source in continua evoluzione che si avvale dei contributi di università e ricercatori. I protocolli principali sono infatti implementati direttamente nel simulatore, mentre molti altri sono reperibili in rete come estensioni. In questo lavoro è stata utilizzata la versione 2.28 del simulatore[4].

Purtroppo non esiste ancora un'implementazione per ns2 (né per altri simulatori di rete) di SAODV. Si è quindi partiti dall'implementazione di AODV apportando poi le necessarie modifiche per rendere il comportamento identico a quello del protocollo sicuro.

Poiché l'implementazione del protocollo AODV presente all'interno di questa versione di ns2 non è strettamente conforme alle specifiche dell'RFC è stato utilizzato il codice[5] prodotto dalla Uppsala University; tale implementazione, chiamata AODV-UU [28], è quella raccomandata dalla comunità scientifica che si occupa di AODV poiché è la maggiormente stabile e testata. Con questa implementazione è inoltre possibile utilizzare il protocollo sia in ambiente simulato tramite ns2 che in ambito reale.

Le modifiche apportate al codice di AODV-UU sono state le seguenti:

- sono stati aggiunti i ritardi necessari a simulare le firme e le verifiche crittografiche;

[4] reperibile presso http://www.isi.edu/nsnam/dist/

[5] reperibile presso http://core.it.uu.se/AdHoc/AodvUUImpl

	Dimensione in byte	
	AODV	**SAODV**
RREQ	24	176
RREQ-DSE		244
RREP	20	176
RREP-DSE		326
RERR	12	148

Tabella 4.1: Dimensioni dei messaggi SAODV

- è stata implementata la la versione *destination only* del protocollo, cioè quella in qui solo i nodi destinazione possono rispondere alle RREQ[6];
- sono state modificate le dimensioni dei messaggi per tenere conto delle estensioni di SAODV (si veda la tabella 4.1);
- è stato modificato il comportamento del protocollo per quanto riguarda la gestione dei nodi vicini;
- è stato modificata la logica usata per gestire il campo *sequence number*;
- è stata aggiunta una coda FIFO all'interno di ogni nodo per poter gestire al meglio i ritardi delle operazioni crittografiche e le rotte *non verificate*;
- è stato modificato il comportamento del protocollo ARP per ovviare ad un baco dell'implementazione del livello MAC di ns2.

Sono state inoltre inserite le due ottimizzazioni studiate in questa tesi:

- è stata implementata una logica *adattativa* per il protocollo che verrà descritta nel paragrafo 4.4.1;

[6]l'uso del flad D è già presente in AODV-UU ma non vi è la possibilità per l'utente di impostarlo a piacere

	$\mathbf{DSA_{512}}$		$\mathbf{RSA_{512}}$		**Hash**
Macchina	Firma	Verifica	Firma	Verifica	Hash (sha-1)
Pentium3 600Mhz	2	2.4	2.4	0.2	0.002
Pentium4 3.2Ghz	0.7	0.8	0.8	0.1	0.001
Centrino Duo	0.61	0.73	0.96	0.1	0.001
Amd 2800+	0.5	0.6	0.6	0.1	0.001

Tabella 4.2: Tempi di firma e verifica in millisecondi

- è stata implementata l'opzione *forward anticipato* che verrà descritta nel paragrafo 4.4.2.

Per valutare la connettività degli scenari in modo da non simulare reti partizionate o comunque poco connesse è stato utilizzato il tool iNSpect [29]. Grazie a questo tool è possibile ottenere una rappresentazione grafica delle posizioni e dei movimenti dei nodi negli scenari prodotti per le simulazioni.

In SAODV non vengono specificati gli algoritmi crittografici da utilizzare ma viene fornito solo un elenco di quelli che devono essere supportati dalle implementazioni del protocollo; così, per decidere il valore dei ritardi da inserire nel simulatore, sono state effettuate delle misurazioni dei tempi necessari per generare una firma, verificare una firma, calcolare un hash. Le misure sono state effettuate su quattro PC con caratteristiche differenti e sono state utilizzate le librerie crittografiche fornite da OpenSSL[7]. Dai risultati ottenuti (riassunti in tabella 4.2) emerge che:

- con l'algoritmo RSA la verifica di una firma richiede un tempo di circa un ordine di grandezza inferiore rispetto ad una sua generazione;
- firme e verifiche DSA richiedono tempi dello stesso ordine di grandezza (e dello stesso ordine di grandezza di una firma RSA), più precisamente una verifica richiede circa il 20% di tempo in più di una firma;

[7] http://www.openssl.org

- il calcolo di un hash richiede circa 1/1000 del tempo di una firma (RSA o DSA) ed 1/100 di una verifica RSA.

Alla luce dei valori riportati in tabella 4.2, l'algoritmo RSA è più indicato per il protocollo SAODV rispetto a DSA. Infatti un messaggio (RREQ, RREP o RERR) viene firmato da un solo nodo della rete mentre viene verificato da molti nodi (i soli vicini nel caso di messaggi RERR, i soli nodi lungo la rotta per i messaggi RREP o potenzialmente tutti i nodi della rete nel caso di RREQ). Nelle simulazioni effettuate si è utilizzato un tempo di firma (t_f) variabile tra i 40 e i 150 millisecondi e un tempo di verifica sempre pari ad $\frac{1}{10} \cdot t_f$ per rispecchiare le caratteristiche dell'algoritmo RSA. I tempi di firma in tabella sono stati misurati su dispositivi con prestazioni più elevate di quelli comunemente utilizzati in una rete mobile ad hoc[8], ma soprattutto tali misure sono state effettuate a macchine "scariche"; nella realtà un dispositivo che partecipa ad una rete mobile ad hoc viene utilizzato dall'utente e quindi il suo processore non è totalmente disponibile per eseguire le operazioni crittografiche richieste dal protocollo di routing. Per questi motivi le simulazioni sono state eseguite scegliendo tempi di firma anche molto maggiori rispetto a quelli misurati , ovvero sono stati usati i valori t_f=40, 60, 80, 100, 130, 150 ms.

Nella sezione 3.2.1.2 è stato descritto il metodo delle catene di hash utilizzato in SAODV per proteggere il valore del campo Hop Count nei messaggi RREP e RREQ. Negli scenari simulati in questa tesi il numero di hop medio non supera mai i 10 salti. Ogni nodo deve quindi calcolare circa 10 hash per pacchetto in modo da verificare la validità del campo Hop Count. Come si può notare dalle misurazioni effettuate, il tempo per il calcolo di un hash è di 2-3 ordini di grandezza inferiore a quello necessario per una verifica; il tempo utilizzato dal nodo per calcolare la catena di hash è dunque trascurabile rispetto alle altre operazioni di crittografia asimmetrica. Per questo motivo nelle simulazioni effettuate in questo lavoro non viene preso in considerazione nessun ritardo per queste operazioni.

[8]si presuppone che in una manet vengano utilizzati PDA, telefoni cellulari, tablet

	alta mobilità	**bassa mobilità**
Velocità minima	1 m/s	1 m/s
Velocità massima	10 m/s	4 m/s
Tempo di pausa	0 s	0 s

Tabella 4.3: I modelli di mobilità

Gli scenari utilizzati sono stati generati secondo il modello *random waypoint* [30]. Sono stati simulati diverse grandezze del territorio con un numero variabile di nodi e di connessioni. Gli scenari con cui sono stati ottenuti risultati più rilevanti sono stati:

- territorio di 200x200 metri con 50 nodi con raggio di copertura pari a 50 metri. Durante la simulazione vengono instaurate 50 connessioni TCP scegliendo a caso il nodo di partenza e d'arrivo.
- territorio di 1500x50 metri con 100 nodi con raggio di copertura 50 metri. Le connessioni TCP instaurate, sempre scegliendo casualmente nodi iniziale e finale, sono 100.

Per entrambi gli scenari sono state simulate due versioni cambiando la mobilità dei nodi come descritto in tabella 4.3.

Le simulazioni sono state eseguite su un tempo simulato di 500 secondi, le connessioni effettuate sono di tipo FTP durante le quali sono scambiati 128 pacchetti di 128 byte l'uno.

In tabella 4.4 sono riassunti i principali parametri adottati.

4.3 Implementazione del protocollo

In questo paragrafo si descriveranno brevemente alcune delle modifiche apportate all'implementazione AODV-UU per renderla compatibile con SAODV e con le ottimizzazioni studiate in questo lavoro.

Parametro	**Valore**
numero di nodi	50..100
dimensione dello scenario	200x200, 1500x50
durata della simulazione	500 secondi
Modello di mobilità	Random Waypoint
Velocità minima	1 m/s
Velocità massima	4 .. 10 m/s
Tempo di pausa	0 secondi
Antenna	omnidirezionale
Modello di propagazione	Two Ray Ground
Livello MAC	IEEE 802.11b
Raggio di copertura radio	50 metri
Numero di connessioni	50 .. 100
Tipo di traffico	FTP (su TCP)
Dimensione dei pacchetti	128 byte
Numero di pacchetti/connessione	128
Tempo di firma	40 .. 150 millisecondi
Tempo di verifica	4 .. 15 millisecondi

Tabella 4.4: Principali parametri delle simulazioni

In AODV quando un nodo riceve un qualsiasi messaggio di routing utilizza le informazioni presenti nell'header IP di tale messaggio per aggiornare le rotte relative ai suoi vicini. Lo standard prevede che il nodo controlli il campo *source address* ed agisca nel seguente modo:

- se ha già una rotta verso quel nodo deve aggiornare il lifetime di tale rotta;
- se il nodo non esiste nella sua tabella di routing allora deve aggiungere la relativa riga.

Questo comportamento non è in linea con la policy di sicurezza di SAODV, infatti così facendo ogni nodo potrebbe forgiare messaggi di routing aventi un header IP fasullo per far aggiungere informazioni non reali nelle tabelle di routing dei propri vicini. In questo modo, per esempio, un nodo attaccante potrebbe far credere che il nodo destinazione D di una comunicazione sia un vicino del nodo sorgente S. S, consultando la propria tabella di routing, invierebbe i pacchetti dati senza effettuare una discovery e questi non sarebbero recapitati in modo corretto. Nonostante non sia presente nel draft di SAODV alcun riferimento a questo problema si è scelto di modificare il comportamento del protocollo in modo da evitare il buco di sicurezza. Nell'implementazione fatta di SAODV i nodi non utilizzano più le informazioni presenti nell'header IP per aggiungere nuove righe nelle loro tabelle di routing; per fare questo possono utilizzare solo i dati presenti nella parte *firmata* dei pacchetti.

Da questa considerazione deriva anche la scelta effettuata di utilizzare i messaggi di Hello per mantenere la connettività ad un hop. In AODV è possibile utilizzare alternativamente i *link layer feedback* che però trasportano informazioni non sicure in quanto non firmate e perciò inattendibili. I messaggi di Hello invece vengono sempre firmati dal nodo che li genera ed il nodo ricevente può essere certo dell'identità del suo vicino controllandone la firma. Per i messaggi di Hello è stata utilizzata l'estensione a singola firma del protocollo SAODV in quanto non porta alcun vantaggio permettere ai nodi di rispondere per rotte verso i propri vicini; infatti il maggior numero di operazioni crittografiche richieste per

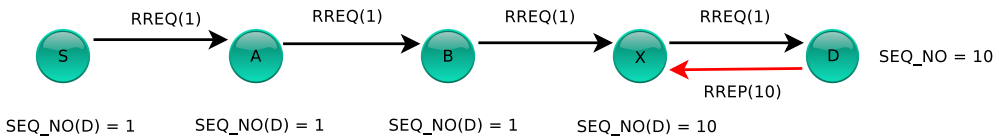


Figura 4.1: Scenario nel quale la gestione del sequence number di AODV non è conforme con i comportamenti di SAODV

la risposta come nodo intermedio non è compensato da un effettivo guadagno in termini di tempo di propagazione della richiesta fino al nodo destinazione.

Un altro problema riscontrato che non viene trattato dal draft di SAODV è quello della gestione del *Destination Sequence Number* durante la fase di discovery. In AODV infatti ogni nodo lungo la rotta può aggiornare tale campo (che viene inizializzato dal nodo sorgente al valore più recente da lui visto) al valore presente nella sua tabella di routing. Questo in SAODV non è possibile in quanto il messaggio di richiesta è firmato e una sua modifica da parte di un nodo intermedio ne invaliderebbe la firma. Questo porta ad una possibile situazione critica poiché AODV specifica che un nodo, prima di rispondere con una Route Reply a una Route Request a lui destinata, incrementi il proprio sequence number *solo se* il Destination Sequence Number della Route Request è maggiore o uguale al proprio. Così facendo si è certi che tutti i nodi che elaboreranno la Route Reply di risposta aggiorneranno la propria tabella di routing e inoltreranno il messaggio; infatti questi avranno sicuramente memorizzato un sequence number minore di quello presente nella Route Reply inviata dal nodo destinazione. In SAODV questo non è più vero.

Si supponga di essere nella situazione rappresentata in figura 4.1, nella quale i nodi utilizzano SAODV: il nodo S invia una Route Request per D, il Destination Sequence Number della Route Request sarà 1 (il valore piu recente che S conosce), il messaggio arriva con quel numero di sequenza fino al nodo D nonostante il nodo intermedio X conosca un sequence number maggiore. D dopo aver verificato che il numero di sequenza è inferiore al proprio, invia una Route Reply con il valore corrente del proprio numero di sequenza non incrementato. Il nodo X dunque

riceve la Route Reply, ma vedendo un numero di sequenza già noto scarta il messaggio; non e stato quindi possibile stabilire una rotta.

La soluzione che è stata utilizzata nell'implementazione è quella di far incrementare sempre il numero di sequenza al nodo destinatario della Route Request prima di generare una Route Reply di risposta. Il comportamento esterno nel caso in cui si usino i messaggi di Hello per mantenere l'elenco dei nodi vicini è uguale a quello di AODV. Questo perché i vicini del nodo destinatario della Route Request hanno sempre il numero corretto di sequenza del nodo, e aggiornano sempre un eventuale numero di sequenza inferiore presente nella Route Request; il nodo destinazione quindi riceve sempre una Route Request col numero di sequenza corretto, e di conseguenza incrementa sempre il proprio numero di sequenza prima di generare la Route Reply. Nel caso in cui invece vengano usati dei meccanismi a livello 2 al posto dei messaggi di Hello, si avrebbe un incremento accelerato dei sequence number.

Si è apportata un'ulteriore modifica al protocollo per quanto riguarda la gestione della tabella di routing. In SAODV un nodo che riceve un messaggio di routing lo accoda in attesa che possa essere elaborato[9]. Quando sono stati gestiti tutti i messaggi precedentemente arrivati si elabora il pacchetto e si inseriscono le informazioni nella tabella di routing. Nella versione implementata per questo lavoro ogni nodo, *prima* di accodare il messaggio per la verifica della firma crittografica, pre-elabora il pacchetto inserendo le eventuali informazioni nella tabella di routing come *non verificate*. In questo modo se dovessero arrivare pacchetti dati che necessitano tali informazioni questi non verrebbero scartati come succederebbe senza questa modifica ma verrebbero invece mantenuti in *cache* per poi essere elaborati nel momento in cui il messaggio di routing sia stato verificato. Questa modifica è stata apportata anche perché risulta estremamente utile quando utilizzata con una delle ottimizzazioni proposte: il *forward anticipato* descritto

[9]essendo necessarie operazioni di crittografia con tempi di esecuzione non trascurabili per elaborare i messaggi è possibile che il nodo non possa gestire immediatamente tutti i pacchetti che gli arrivano

in 4.4.2.

Un'altra modifica effettuata all'implementazione di AODV-UU è stato l'adattamento dei timeout per tenere conto dei maggiori ritardi dovuti alle operazioni crittografiche. Più precisamente per quei timeout che sono funzione dalla grandezza della rete si è imposta una dipendenza dal NODE_TRAVERSAL_TIME. Questo parametro è stato fissato in modo da tenere conto dei tempi impiegati dai nodi per le operazioni di firma o verifica e del tempo di propagazione del pacchetto. Per i timeout indipendenti dalla grandezza della rete invece si è cercato un valore ragionevolmente superiore a quello precedentemente fissato in AODV-UU.

4.4 Le ottimizzazioni

In questo paragrafo verranno descritte le ottimizzazioni proposte per il protocollo SAODV.

4.4.1 L'adattatività

Nel protocollo SAODV si è mantenuta la caratteristica, presente in AODV, di consentire ai nodi intermedi di rispondere alle richieste a nome dei nodi destinatari. Per fare questo si è utilizzato il meccanismo di doppia firma descritto nel paragrafo 3.2.1.4. Questo meccanismo, per non intaccare le caratteristiche di sicurezza di SAODV, fa un massiccio utilizzo di operazioni di crittografia asimmetrica. Infatti il nodo intermedio che vuole rispondere ad una richiesta deve effettuare la verifica del messaggio e due firme: una per il messaggio di risposta e una per il messaggio di risposta gratuito da inviare al nodo destinazione[10]. Inoltre, per consentire la risposta ai nodi intermedi, i nodi che generano messaggi RREQ devono allegare due firme: una per il messaggio di richiesta e una per il successivo messaggio di risposta.

[10]in AODV è consigliato l'utilizzo dei messaggi di RREP gratuiti nel caso di connessioni bidirezionali come quelle utilizzate nelle simulazioni di questo lavoro

Per questo motivo non è sempre conveniente consentire ai nodi intermedi di rispondere. Infatti, se la rete è particolarmente carica, si obbligano i nodi intermedi e i nodi sorgenti a effettuare un maggior numero di firme e verifiche, aumentando ancora di più il carico già critico della rete. Inoltre, anche prendendo in considerazione la singola discovey,le operazoni crittografiche aggiuntive necessarie possono richiedere un tempo maggiore rispetto a quello necessario al pacchetto per giungere fino al destinatario. Si è quindi pensato di applicare al meccanismo di risposta una *logica adattativa*.

All'arrivo di una richiesta per la quale il nodo possiede le informazioni necessarie per rispondere viene valutato il numero di firme e verifiche in attesa di essere effettuate dal nodo e e di conseguenza si calcola il tempo che questo impegherà per effettuare le operazioni richieste per la risposta:

- se questo tempo supera una soglia prestabilita allora il nodo non risponde ed effettua il normale forward del messaggio;
- se invece il tempo è inferiore alla soglia il nodo collabora.

In questo modo, oltre a velocizzare l'operazione di discovery in atto, viene anche alleggerito il carico del nodo intermedio che si trova già in situazione di congestione.

La generazione di messaggi di richiesta a doppia firma viene effettuata con lo stesso meccanismo: al momento di effettuare una discovery il nodo sorgente valuta la lunghezza della propria coda crittografica. Se questa supera una soglia prestabilita allora il nodo genera un messaggio a singola firma altrimenti genera un messaggio a doppia firma. Anche in questo caso, oltre a velocizzare il tempo di discovery, si allevia il carico computazionale del nodo che deve così generare una sola firma al posto delle due necessarie per il messaggio RREQ-DSE.

4.4.2 Il forward anticipato

Durante il processo di route discovery in SAODV ogni nodo che riceve il messaggio di richiesta o quello successivo di risposta deve verificare la firma effettuata dal nodo sorgente prima di elaborare il pacchetto e di rispedirlo, se necessario. Questo introduce un certo ritardo nell'acquisizione da parte del nodo sorgente delle informazioni di routing di cui necessita. Questo ritardo è maggiormente elevato se le code dei pacchetti che necessitano di operazioni crittografiche sono particolarmente lunghe, come nel caso di reti con tante connessioni tra coppie di nodi diverse; queste infatti necessitano ogni volta di una nuova discovery essendo SAODV un protocollo reattivo. Per cercare di ridurre il ritardo dovuto alla verifica delle firme crittografiche effettuata da tutti i nodi lungo il percorso di diffusione dei messaggi di routing si è introdotto il *forward anticipato*. Questa ottimizzazione consiste nel consentire ai nodi di effettuare il forward del pacchetto (in caso sia necessario) *prima* che la relativa firma venga verificata. Un nodo in cui sia abilitato il forward anticipato agisce nel seguente modo in base al tipo di pacchetto:

route request - il pacchetto viene pre-elaborato e vengono inserite le informazioni relative nella tabella di routing come *non verificate*. A questo punto

- se il nodo è la destinazione della richiesta o un nodo intermedio con le informazioni necessarie per rispondere allora:
  1. mette il pacchetto in coda per la verifica;
  2. crea il pacchetto di risposta e lo mette immediatamente in coda per effettuare la firma, senza attendere che il pacchetto di richiesta sia effettivamente verificato in modo da non introdurre ulteriori ritardi se dovessero arrivare nel frattempo altri pacchetti;
  3. quando il pacchetto di richiesta viene verificato:
     - se risulta valido allora vengono rese *verificate* le relative infor-

mazioni nella tabella di routing e viene firmato il pacchetto di risposta;

- se risulta fasullo viene scartato e viene eliminato dalla coda anche il relativo pacchetto di risposta e le informazioni nella tabella di routing inserite precedentemente.

- se è un nodo intermedio e non ha le informazioni necessarie per rispondere allora:
  1. mette il pacchetto in coda per la verifica;
  2. effettua l'inoltro del pacchetto in broadcast;
  3. quando il pacchetto di richiesta viene verificato:
     - se risulta valido allora vengono rese *verificate* le relative informazioni nella tabella di routing;
     - se risulta fasullo viene scartato così come le informazioni precedentemente inserite come *non verificate*.

route reply - il pacchetto viene pre-elaborato e vengono inserite le informazioni relative nella tabella di routing come *non verificate*. A questo punto:

- se il nodo è la destinazione della risposta il pacchetto viene messo in coda e viene attesa la sua verifica.
  - se risulta valido vengono rese verificate le informazioni relative nella tabella di routing e si dà inizio al trasferimento di dati;
  - se risulta fasullo viene scartato così come le informazioni precedentemente inserite come *non verificate*.
- se il nodo non è la destinazione della risposta
  - il pacchetto viene messo in coda per la verifica;
  - viene effettuato l'inoltro del pacchetto lungo la rotta verso la destinazione;
  - quando il pacchetto viene verificato

- se risulta valido vengono rese verificate le informazioni relative nella tabella di routing;
- se risulta fasullo viene scartato così come le informazioni precedentemente inserite come *non verificate*.

route error - ai messaggi di errore non è possibile applicare questa ottimizzazione in quanto devono essere firmati da tutti i nodi che necessitano di inoltrarli ai propri vicini.

Questa ottimizzazione, a differenza dell'adattatività, potrebbe introdurre però una problematica di sicurezza. Infatti, se un nodo attaccante generasse RREQ con firme fasulle in una rete dove fosse attivo il forward anticipato, la diffusione del messaggio non verrebbe limitata ai soli vicini del nodo ostile come succederebbe nella versione del protocollo senza ottimizzazioni o con la sola adattatività, ma continuerebbe fino all'esaurirsi del TTL quindi, presumibilmente, per tutta la rete se il nodo è intenzionato a procurare danno. Iterando questo comportamento il nodo attaccante potrebbe portare le code di tutti i nodi a saturarsi di richieste che risulterebbero oltrettutto essere fasulle. Nelle versioni senza ottimizzazioni o con la sola ottimizzazione adattativa, invece, un comportamento del genere potrebbe, nel caso peggiore, far saturare unicamente le code dei nodi vicini del nodo attaccante che, dovendo verificare la correttezza della firma prima di inoltrarlo, blocchereberro la propagazione del messaggio fasullo.

4.5 Risultati ottenuti e commenti

In questo paragrafo verranno esposti i risultati ottenuti con la simulazione del protocollo SAODV e delle ottimizzazioni ad esso applicate. Verranno mostrati prima i risultati delle singole ottimizzazioni e successivamente i risultati di simulazioni in cui vengono applicate entrambe le ottimizzazioni.

4.5.1 Simulazioni del protocollo SAODV adattativo

Le prime simulazioni effettuate sulla versione adattativa del protocolloi, descritta nel paragrafo 4.4.1, si basano su uno scenario quadrato di lato 200 metri in cui si muovono 50 nodi seguendo lo schema di mobilità *random waypoint* con velocità minima di 1 m/s e massima di 10 m/s e nessun tempo di pausa tra uno spostamento e il successivo. Sono stati simulati nodi con un raggio di copertura radio di 50 metri che cercano di instaurare un totale di 50 connessioni TCP scegliendo ogni volta casualmente nodo sorgente e destinazione. I tempi di firma sono stati posti a 40, 60, 80, 100, 130 e 150 millisecondi.

Sono state simulate tre versioni del protocollo:

- una *non collaborativa* dove i nodi intermedi non rispondono *mai*;
- una *completamente collaborativa* dove i nodi rispondono *sempre* a patto di avere le informazioni necessarie;
- una *adattativa* dove i nodi rispondono in base ai criteri dettati dall'ottimizzazione studiata.

La soglia in base alla quale scatta l'adattatività è stata impostata ad un valore pari a due volte il tempo necessario ad effettuare un'operazione di firma. Per prima cosa si è deciso di rendere la soglia dipendente dal tempo di firma in quanto la lunghezza delle code crittografiche dei nodi è anch'essa strettamente dipendente da tale valore, oltre che dalle caratteristiche dello scenario. Scegliere un valore indipendente dal tempo impiegato per effettuare le operazioni crittografiche avrebbe portato ad avere situazioni diverse al variare del tempo di firma: in simulazioni con tempo di firma basso l'adattatività sarebbe scattata solo quando i nodi fossero stati molto congestionati mentre con tempi di firma alti si sarebbe evitata la collaborazione anche in situazioni di congestione bassa se non nulla. Il coefficiente di proporzionalità tra soglia e tempo di firma è stato scelto in modo che il meccanismo adattativo scattasse un numero ragionevole di volte. Valori minori portano a risultati molto simili al protocollo nella versione non collaborativa in

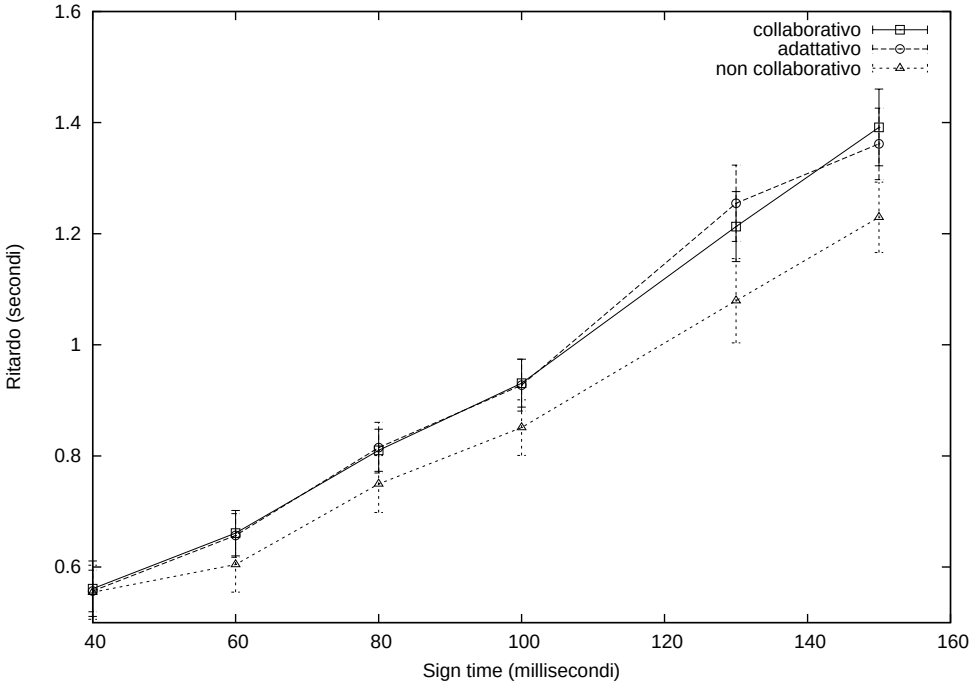


Figura 4.2: Ritardo del primo pacchetto dati

quanto il meccanismo adattativo scatta quasi sempre; valori maggiori, viceversa, portano a comportamenti molto simili al protocollo completamente collaborativo.

In figura 4.2 è mostrato l'andamento del ritardo accusato dal primo pacchetto dati di ogni connessione dal momento della sua creazione al momento della ricezione da parte del nodo destinazione. Come si può notare le versioni collaborativa e adattativa del protocollo portano a tempi maggiori rispetto alla versione non collaborativa. Inoltre i valori del ritardo per le due versioni che prevedono collaborazione tra i nodi sono molto simili.

Questi risultati sono dovuti principalmente al fatto che in uno scenario come quello simulato le rotte sono molto brevi. Infatti, come si vede in figura 4.3, si hanno mediamente rotte di lunghezza compresa tra i due e i tre hop. In questo caso i protocolli collaborativi non hanno risultati apprezzabili in quanto lo svantaggio dato dalle più numerose operazioni crittografiche richieste non è compensato dal vantaggio di non dover effettuare l'intero percorso da nodo sorgente a nodo destinazione. Inoltre l'adattatività dell'ottimizzazione proposta scatta

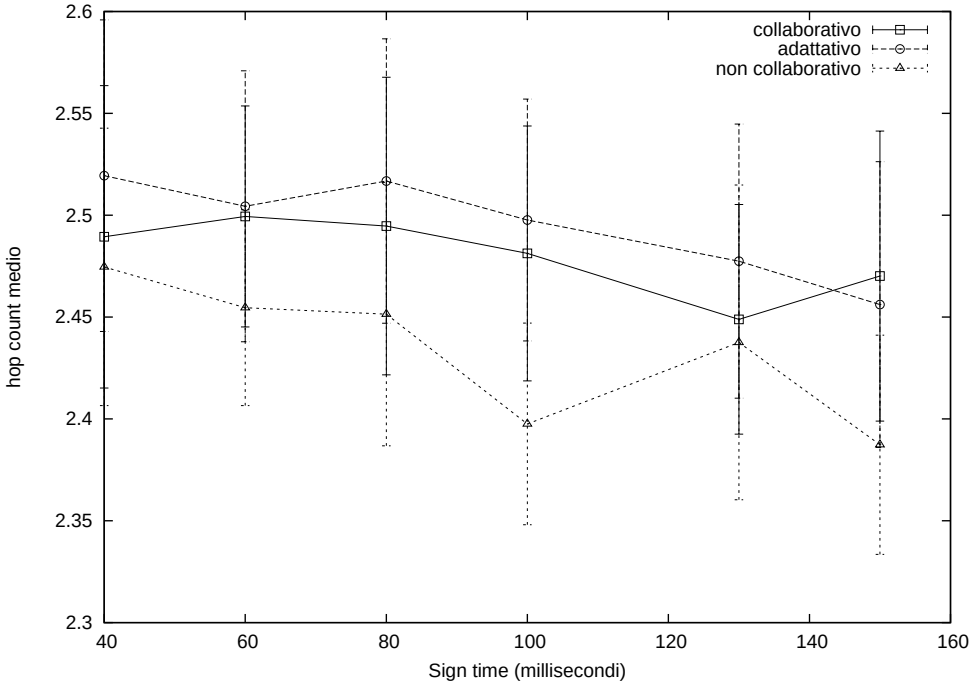


Figura 4.3: Hop count medio

molto raramente poiché i nodi non sono eccessivamente carichi.

In scenari così piccoli è anche possibile che alcune delle informazioni di routing necessarie siano in realtà già presenti nella tabella di routing del nodo sorgente e che quindi i pacchetti dati vengano inviati immediatamente senza dover attendere il processo di discovery. Per questo motivo si è misurato un ulteriore parametro: il tempo di instaurazione della rotta, calcolato come differenza tra l'istante di ingresso in coda del primo pacchetto dati di una comunicazione e l'istante in cui questo esce dalla coda eliminando però tutti quei pacchetti per i quali questa differenza è nulla cioè quelli per cui non scatta il processo di route discovery. In figura 4.4 vengono mostrati gli andamenti di questo parametro. Si può notare che in questo caso vi è un leggero miglioramento delle prestazioni per il protocollo adattativo rispetto a quello completamente collaborativo anche se il divario dovuto alla brevità delle rotte è ancora evidente. La differenza di prestazioni tra i protocolli collaborativi e non collaborativi può essere dovuta principalmente a due cause:

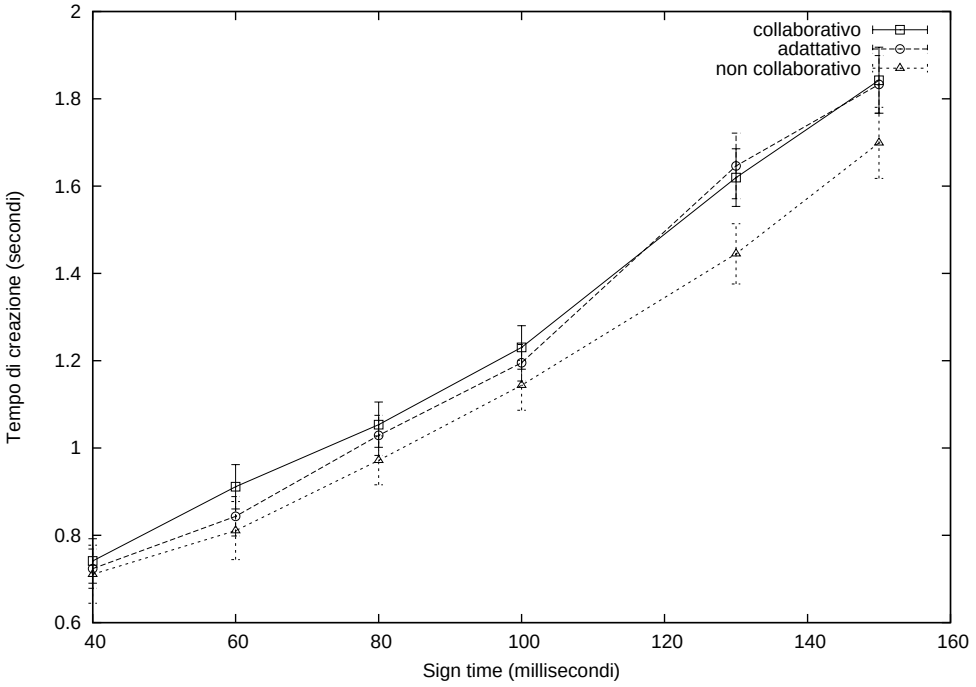


Figura 4.4: Tempi necessario per effettuare la discovery

- le maggiori operazioni crittografiche effettuate dal nodo sorgente di una RREQ-DSE;
- l'ulteriore firma necessaria al nodo intermedio che intende rispondere ad una richiesta.

Per capire quale delle due cause fosse responsabile delle peggiori prestazioni dei protocolli collaborativi si è implementata un'ulteriore piccola modifica che verrà descritta nel paragrafo 4.5.1.1.

Si è quindi deciso di effettuare le simulazioni su scenari nei quali le rotte fossero più lunghe. Per questo sono state effettuate diverse prove con scenari quadrati più grandi, cambiando il numero di nodi e il raggio di copertura. Anche in questo modo non si sono ottenuti innalzamenti apprezzabili nel numero di hop medio delle rotte. Si è quindi passati a scenari rettangolari nei quali la disposizione maggiormente "allungata" dei nodi può portare a rotte più lunghe senza in questo

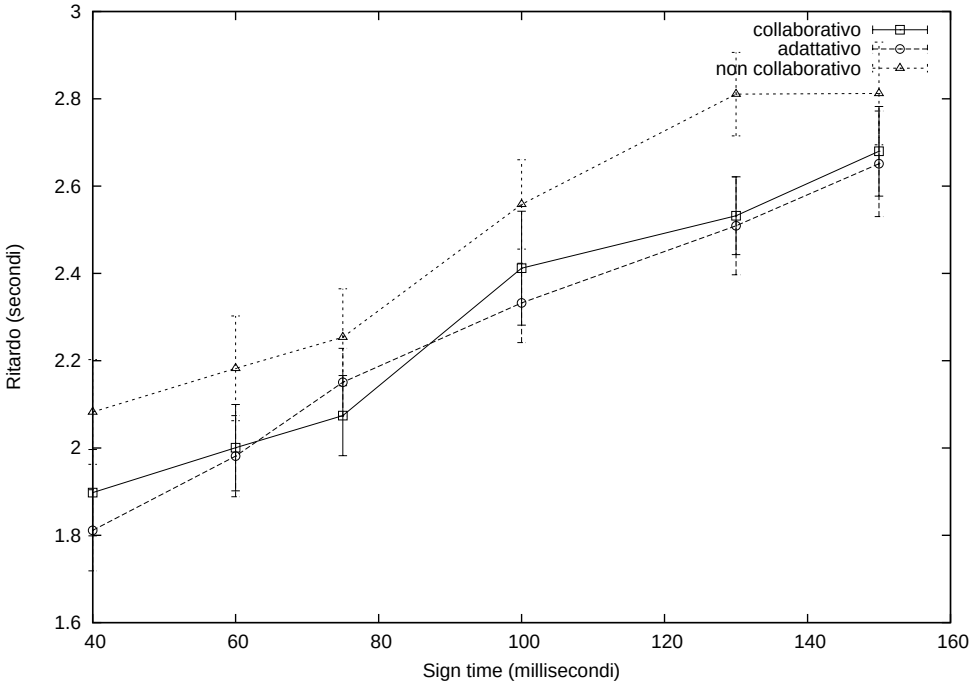


Figura 4.5: Ritardo del primo pacchetto dati

modo dover aumentare troppo il numero di nodi simulati[11].

Nelle figure 4.5 e 4.6 si possono vedere i risultati ottenuti per quanto riguarda rispettivamente il tempo impiegato dal primo pacchetto TCP ad arrivare a destinazione e il tempo necessario per la discovery in uno scenario di 1500x50 metri con 100 nodi che effettuano 100 connessioni TCP e si muovono seguendo il medesimo modello di mobilità precedente. Come si può vedere in questo caso i protocolli collaborativi hanno prestazioni migliori rispetto a quello non collaborativo al contrario di quanto avveniva nel caso precedente. Questo è dovuto alla maggiore lunghezza delle rotte che in questo scenario si aggira mediamente sui 5 hop come si può vedere in figura 4.7. Il particolare andamento decrescente dell'hop count medio può essere giustificato dal fatto che, con l'aumentare del tempo di firma, a causa della grande mobilità dei nodi, non tutte le rotte riescono ad essere create. In particolare si ha una probabilità maggiore che non vengano create le rotte più

[11] l'aumento del numero di nodi comporta un aumento più che proporzionale del tempo necessario ad effettuare la simulazione

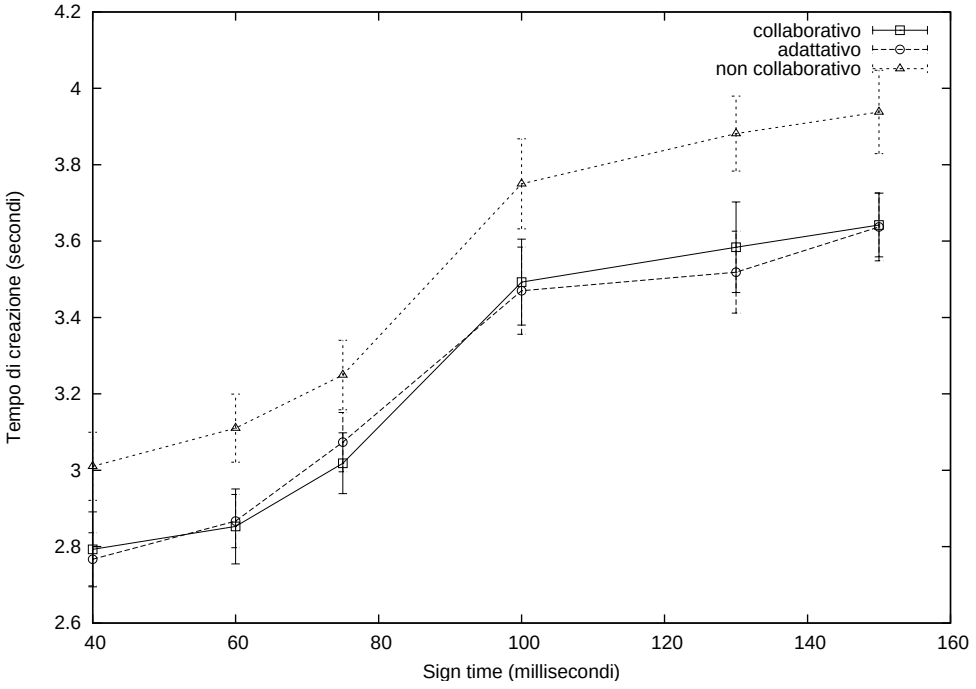


Figura 4.6: Tempo necessario per effettuare la discovery

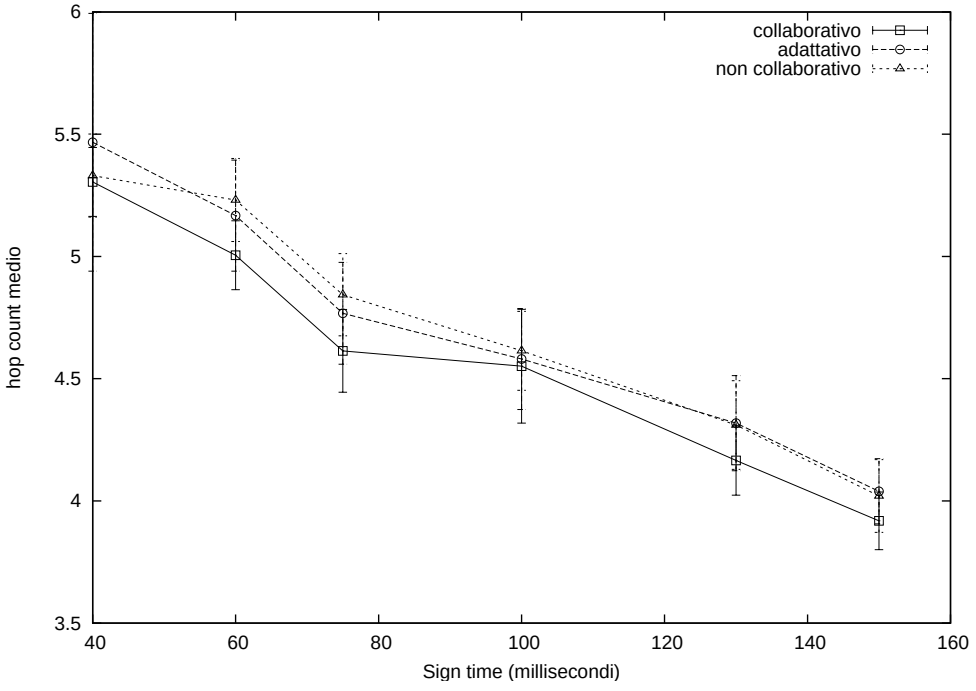


Figura 4.7: Hop count medio

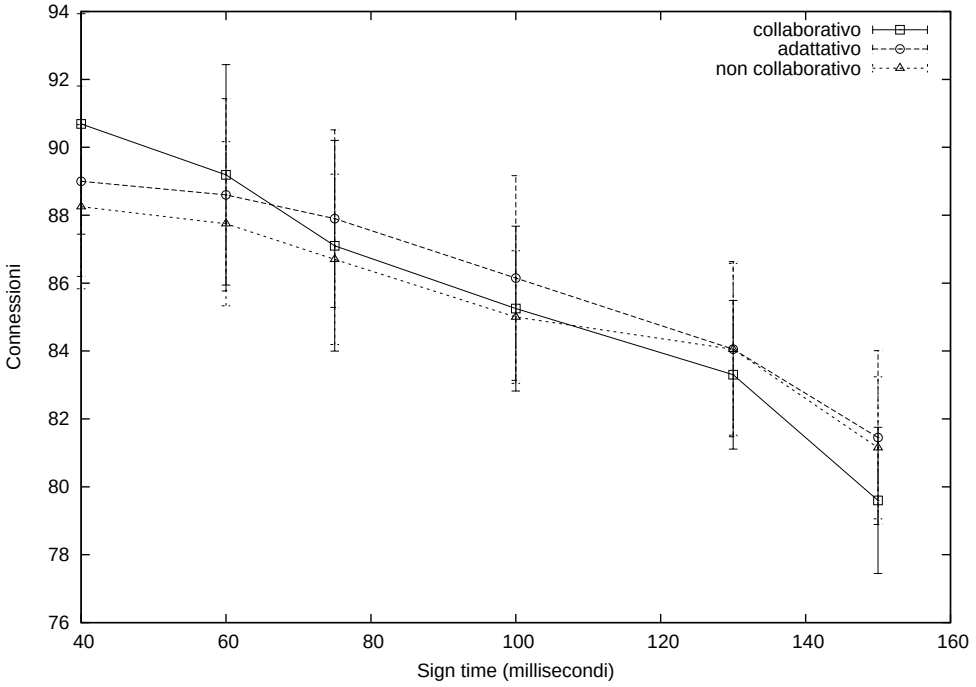


Figura 4.8: Numero di connessioni instaurate

lunghe, sulle quali la mobilità dei nodi ha maggiore effetto in quanto è richiesto un tempo maggiore per la loro instaurazione. Come si può vedere in figura 4.8 il numero di connessioni andate a buon fine è diverso nei tre protocolli simulati e decresce all'aumentare del tempo di firma. In particolare si vede che il protocollo adattativo riesce ad instaurare un numero maggiore di connessioni rispetto alle altre due varianti. Questa osservazione permette di vedere sotto una luce diversa i risultati di figura 4.5 e 4.6. Infatti i valori riportati sono influenzati dal numero di connessioni andate a buon fine. Le connessioni più lunghe, cioè quelle la cui discovery richiede un tempo maggiore, sono quelle con maggior probabilità di non essere instaurate. Il fatto quindi di avere un ritardo medio maggiore nella creazione delle rotte è influenzato non solo dal diverso comportamento del protocollo ma anche dal diverso numero di rotte realmente scoperte. Si è quindi valutato un ulteriore parametro per tenere conto di questa dipendenza dal numero di connessioni instaurate e dalla loro lunghezza media.

In figura 4.9 è mostrato l'andamento del tempo impiegato dal primo pacchetto

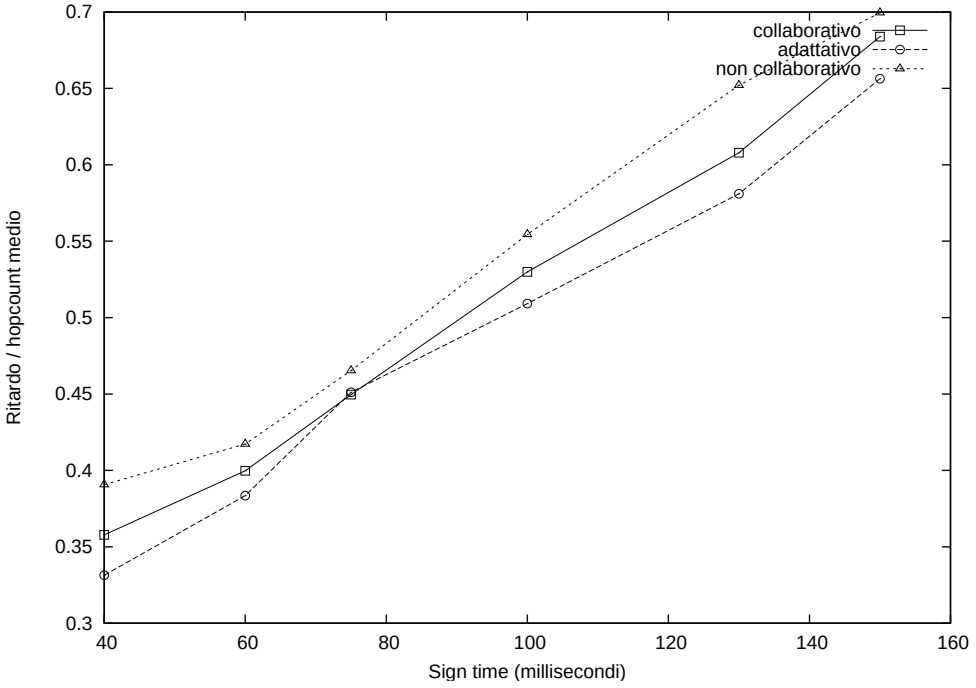


Figura 4.9: Ritardo del primo pacchetto TCP normalizzato rispetto all'hop count medio delle rotte

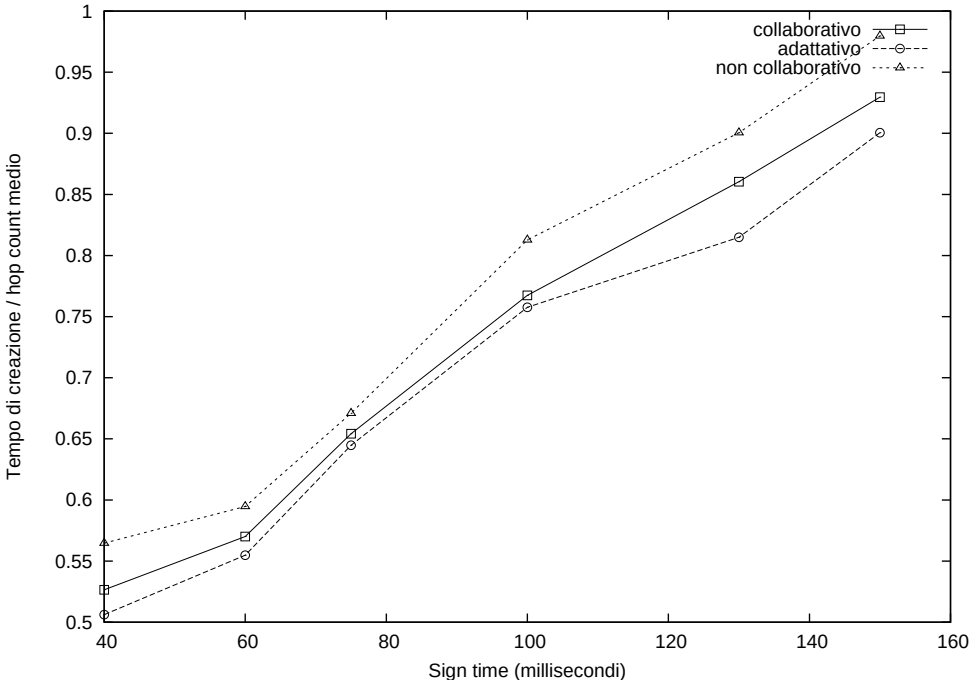


Figura 4.10: Tempo di creazione della rotta normalizzato rispetto all'hop count medio delle rotte

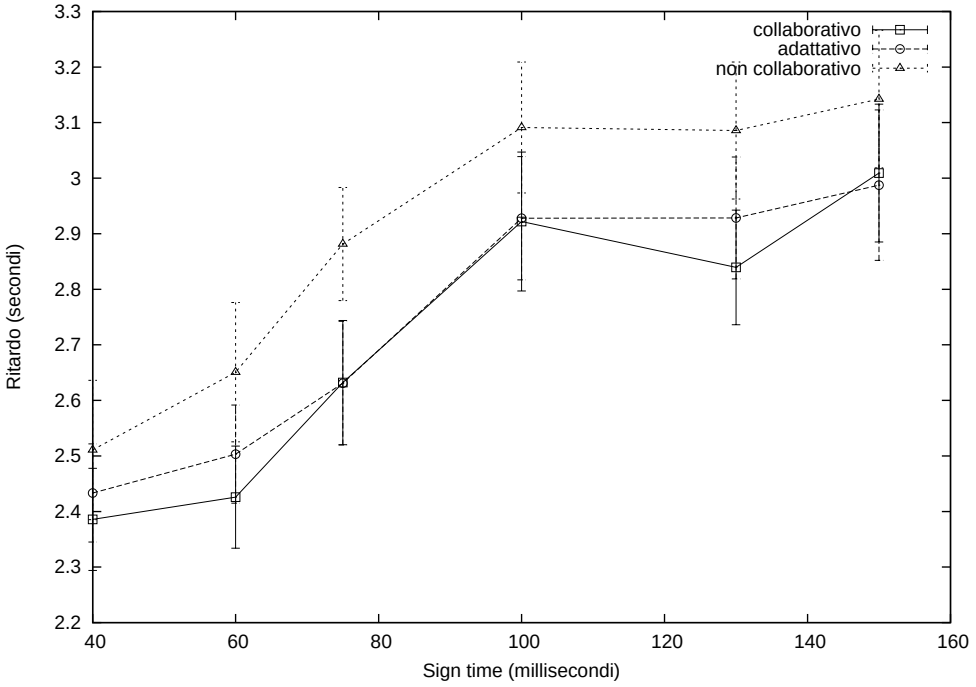


Figura 4.11: Ritardo del primo pacchetto TCP

TCP a giungere a destinazione normalizzato rispetto all'hop count medio delle rotte. Come si può vedere, in questo caso il protocollo adattativo risulta migliore delle altre due varianti. In figura 4.10 è mostrato l'andamento del tempo di instaurazione della rotta anch'esso normalizzato rispetto all'hop count medio delle rotte. Le migliori prestazioni del protocollo adattativo evidenziate valutando questo nuovo parametro sono quindi dovute, oltre ad una più veloce instaurazione delle rotte, anche ad un maggior numero di connessioni andate a buon fine.

Sono state effettuate ulteriori simulazioni su scenari rettangolari, ma utilizzando una mobilità inferiore per analizzare il comportamento del protocollo. In questo caso la lunghezza media delle rotte è aumentata ancora leggermente in quanto è facilitata la discovery per le rotte più lunghe. In questo caso i parametri del modello random waypoint utilizzati sono stati: velocità minima di 1 m/s e velocità massima si 4 m/s, con tempo di pausa tra un movimento ed il successivo sempre nullo.

Nelle figure 4.11, 4.12, 4.13, 4.14 sono mostrati i risultati ottenuti per quanto

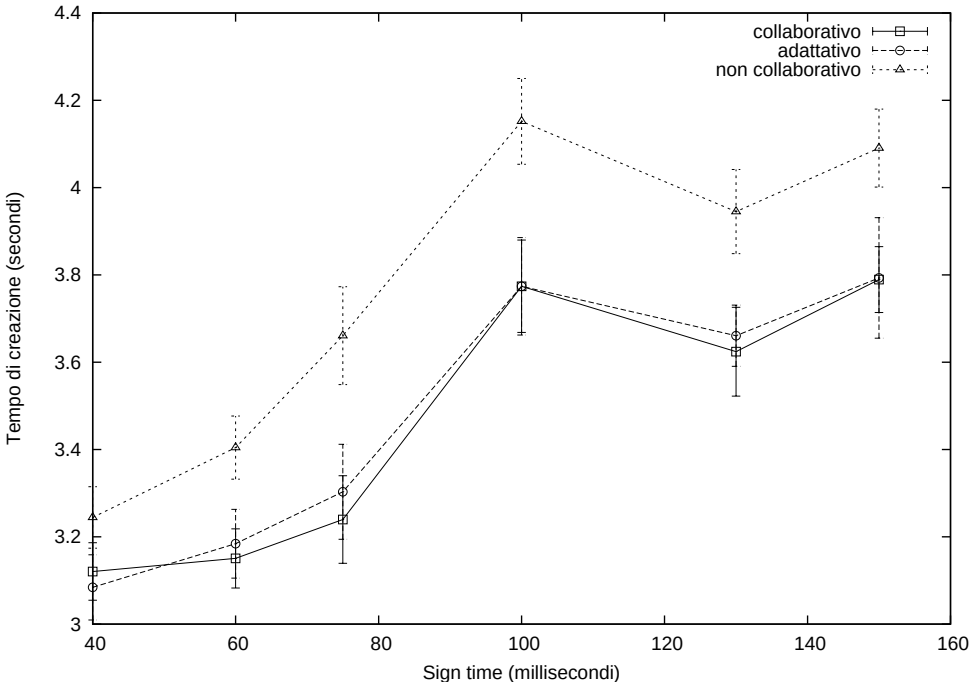


Figura 4.12: Tempo di creazione della rotta

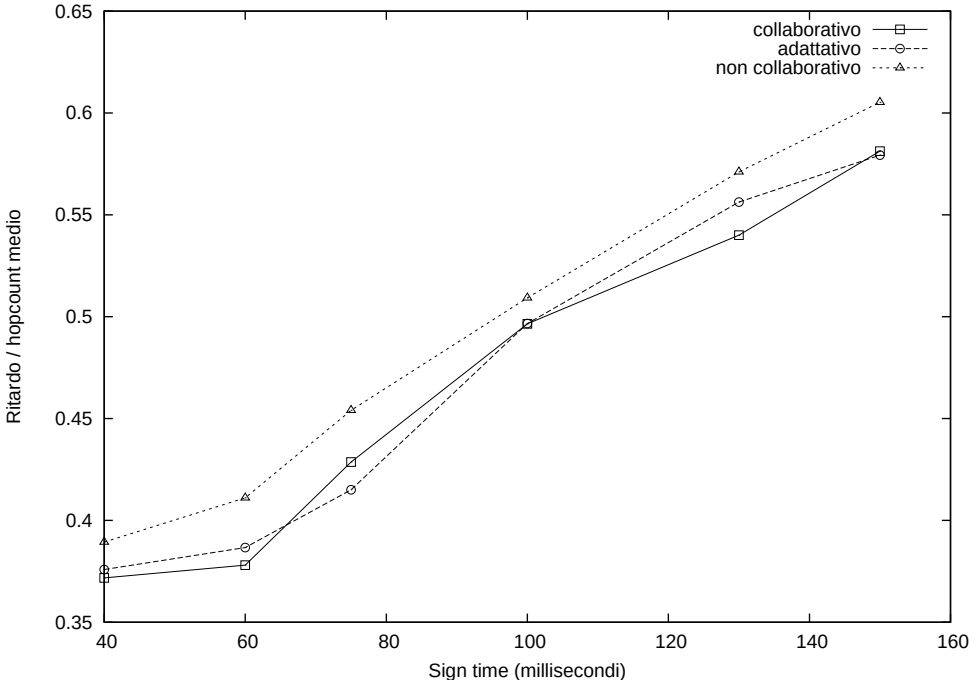


Figura 4.13: Ritardo del primo pacchetto TCP normalizzato rispetto all'hop count medio delle rotte

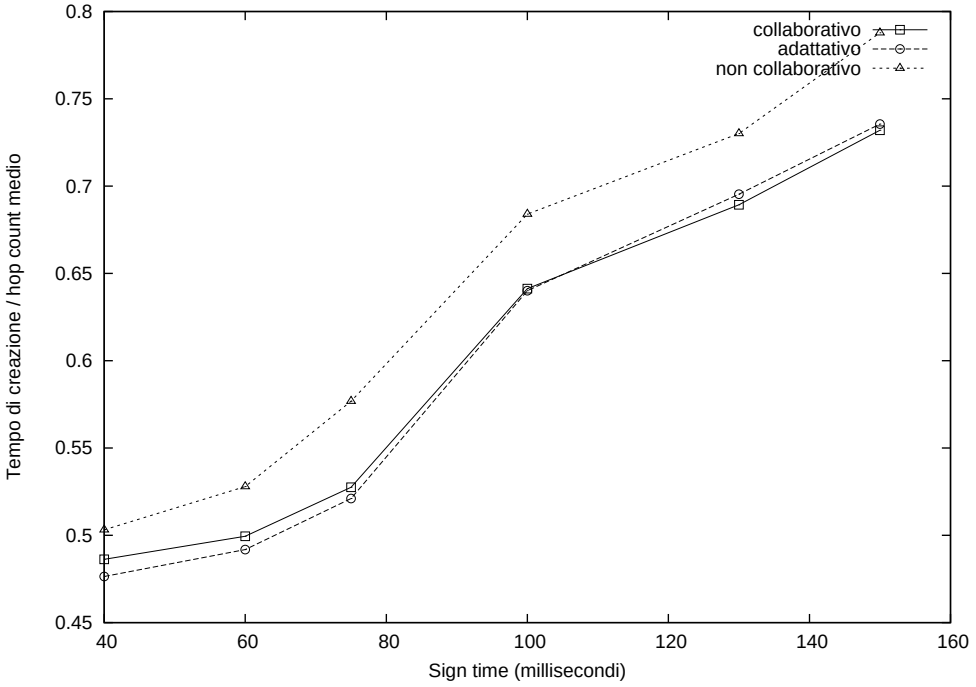


Figura 4.14: Tempo di creazione della rotta normalizzato rispetto all'hop count medio delle rotte

riguarda i valori di ritardo con e senza normalizzazione rispetto all'hop count. In questo caso si può notare come la differenza data dalla normalizzazione sia inferiore in quanto la minore mobilità porta a una più facile instaurazione di rotte lunghe che erano le responsabili del divario visto in precedenza.

Per tutte le simulazioni effettuate si sono valutati anche il numero di pacchetti persi e l'overhead generato dal protocollo di routing. Gli andamenti di questi parametri non sono molto significativi in quanto non si hanno grosse differenze tra le varie versioni simulate. In figura 4.15 è mostrato l'andamento del numero di pacchetti generati dal protocollo di routing nello scenario rettangolare con alta mobilità. Tale numero è stato ricavato contando ogni singolo hop effettuato da un pacchetto di controllo. Per questo motivo il parametro è fortemente influenzato dal numero di RREQ che, essendo inviate in flooding, effettuano un numero di hop molto maggiore rispetto a RREP e RERR.

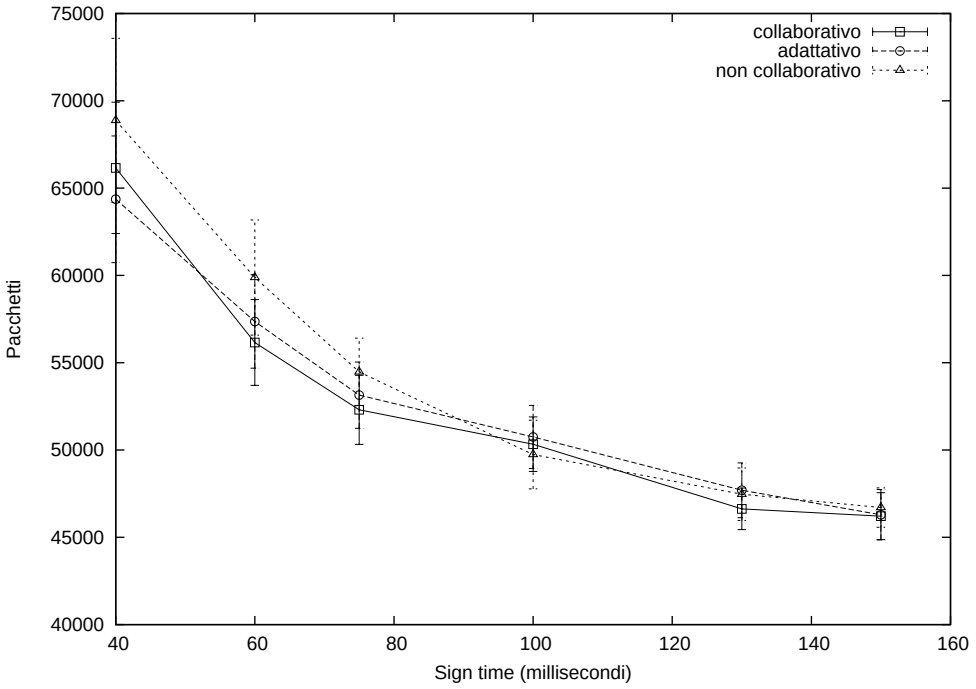


Figura 4.15: Overhead introdotto in numero di pacchetti

4.5.1.1 Controllo dell'hop count

Il gap prestazionale osservato in scenari con rotte molto brevi può essere dovuto, come detto, a due cause: la doppia firma delle RREQ-DSE o la seconda firma necessaria per il pacchetto RREP gratuito che il nodo intermedio deve inviare se decide di rispondere.

Per individuare a quale delle due cause sia dovuta l'evidente differenza nei ritardi di instaurazione di rotta è stato modificato ulteriormente il protocollo adattativo inserendo una logica in base alla quale un nodo intermedio decide se rispondere valutando anche la propria distanza dal nodo destinazione della richiesta.

L'ottimizzazione agisce in questo modo: all'arrivo del messaggio di richiesta il nodo intermedio verifica di possedere tutte le informazioni necessarie per rispondere. In caso affermativo controlla nella propria tabella di routing qual è l'hop count della rotta relativa al nodo destinazione della richiesta:

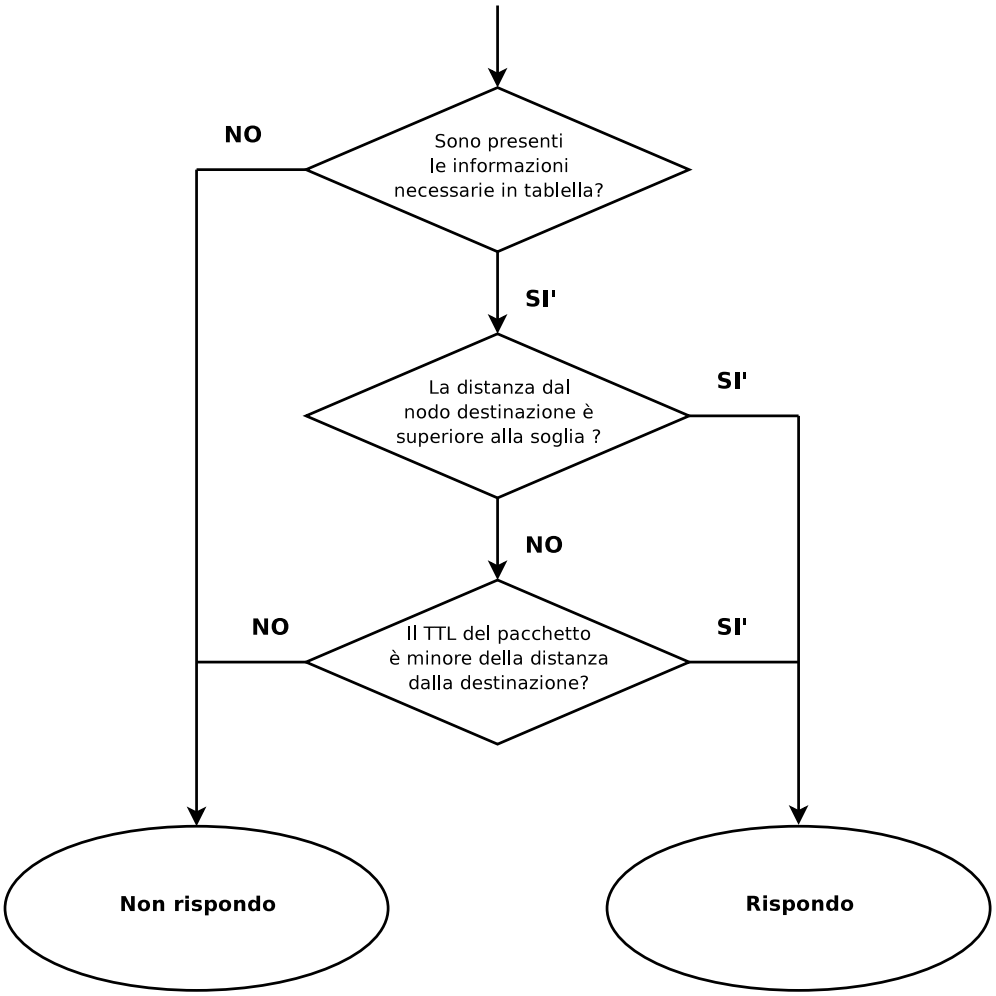


Figura 4.16: Diagramma di flusso

- se questa è maggiore di una soglia prestabilita (impostata nelle simulazioni a due hop) allora risponde;
- se la distanza è minore della soglia allora controlla il campo TTL del pacchetto RREQ:
  - se il valore del TTL è maggiore o uguale alla distanza dalla destinazione allora il pacchetto viene inoltrato senza rispondere;
  - altrimenti il nodo risponde.

L'ultimo controllo serve, in caso di utilizzo dell'expanding ring, ad evitare che un nodo con le informazioni necessarie non risponda e costringa il nodo sorgen-

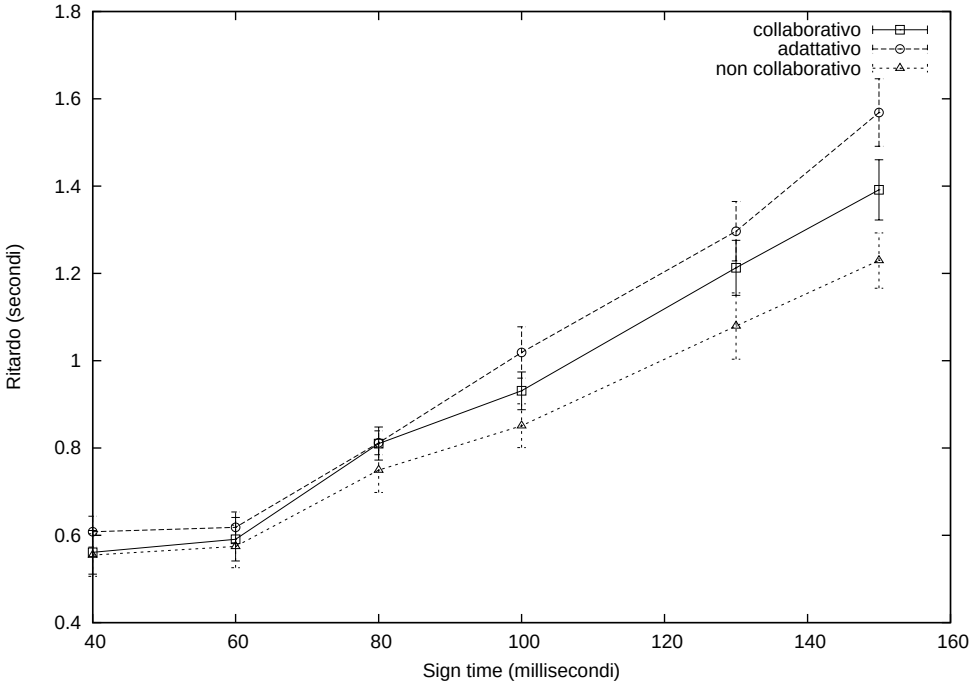


Figura 4.17: Ritardo del primo pacchetto tcp

te della richiesta a fare un'ulteriore discovery. In figura 4.16 viene mostrato il meccanismo implementato. Va precisato che questo meccanismo sta a valle del meccanismo adattativo che quindi effettua una prima selezione delle richieste alle quali sarà possibile rispondere.

In figura 4.17 sono riportati i risultati delle simulazioni effettuate con questa modifica. Come si vede l'effetto ottenuto è quello di un ulteriore peggioramento delle prestazioni del protocollo adattativo. Questo significa che, in realtà, a pesare sui tempi di discovery non è il tempo impiegato dal nodo intermedio per generare l'ulteriore firma per i messaggi RREP gratuiti ma il tempo utilizzato dal nodo sorgente per generare i messaggi RREQ-DSE; questa giustificazione da sola non spiega l'aumento dei ritardi quindi il fatto di evitare che il nodo intermedio risponda in alcuni casi non è un vantaggio ma bensì uno svantaggio. Purtroppo non è possibile per il nodo sorgente sapere la lunghezza delle rotte per il quale il suo messaggio RREQ-DSE verrà utilizzato; non è dunque individuabile una strategia per fare in modo che il nodo generi messaggi a doppia firma solo nel

caso in cui le rotte siano lunghe.

In conclusione possiamo dire che l'ottimizzazione adattativa del protocollo porta a leggeri miglioramenti rispetto alla versione collaborativa per quanto riguarda il tempo di discovery in scenari in cui vi siano rotte lunghe. Entrambe le versioni sono inoltre decisamente migliori rispetto a quella non collaborativa. In scenari invece dove le rotte sono più brevi i protocolli adattativo e collaborativo hanno prestazioni pressoché identiche e risultano inoltre peggiori del protocollo non collaborativo. La causa di questo gap è stata individuata nel tempo impiegato dal nodo sorgente per effettuare le due firme necessarie alla RREQ-DSE che purtroppo non può essere mitigato da una strategia adattativa. Per quanto riguarda gli altri parametri valutati nelle simulazioni non vi sono grosse differenze tra le tre versioni del protocollo.

4.5.2 Simulazione del protocollo SAODV con forward anticipato

In questo paragrafo mostreremo i risultati ottenuti con la seconda ottimizzazione proposta. Gli scenari simulati sono i medesimi visti nel paragrafo 4.5.1 per la versione adattativa. In questo caso sono simulate due versioni del protocollo, una nella quale è abilitato il forward anticipato dei messaggi ed una dove questa ottimizzazione è disabilitata. In entrambe le versioni il comportamento del protocollo per quanto riguarda la risposta dei nodi intermedi è di collaborazione completa.

I risultati ottenuti non mostrano significative differenze date dalla grandezza dello scenario se non un distacco leggermente minore tra le curve negli scenari con rotte molto brevi. Viene quindi mostrato in figura 4.18 l'andamento del ritardo del primo pacchetto TCP al variare del tempo di firma per lo scenario rettangolare ad alta mobilità. L'andamento evidenziato dal grafico non è particolarmente significativo infatti, anche in questo caso, il risultato è influenzato dal diverso numero di connessioni instaurate dalle due versioni del protocollo (mostrate in figura

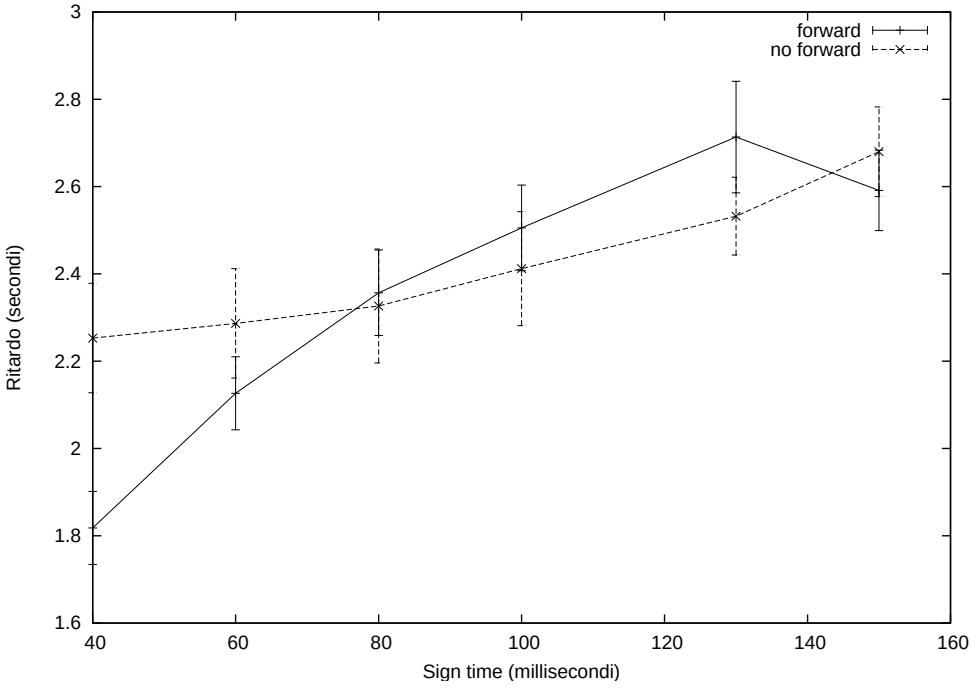


Figura 4.18: Ritardo del primo pacchetto TCP

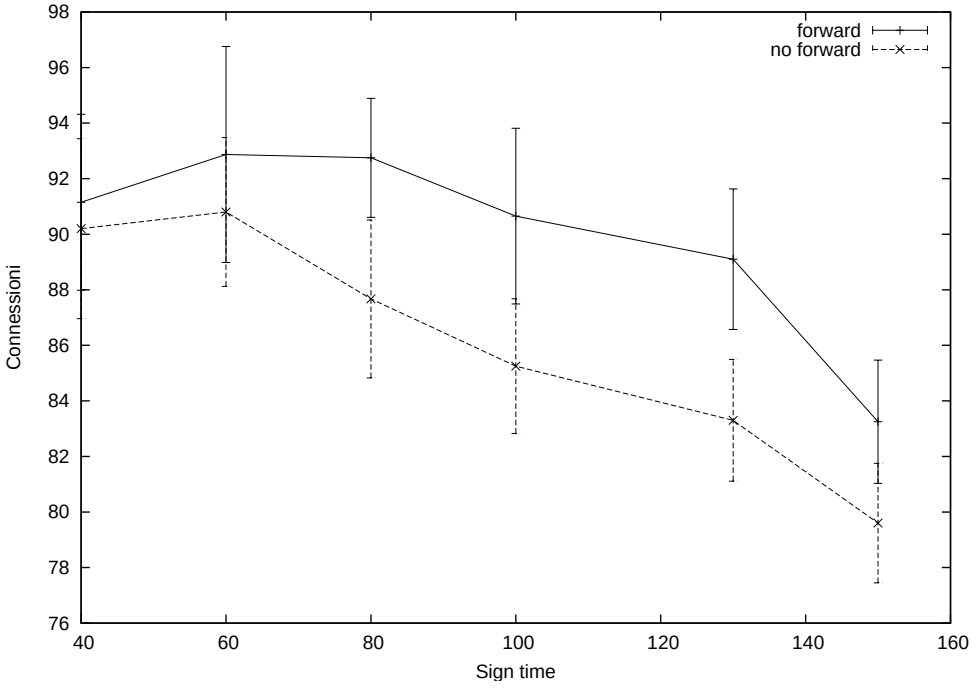


Figura 4.19: Numero di connessioni instaurate

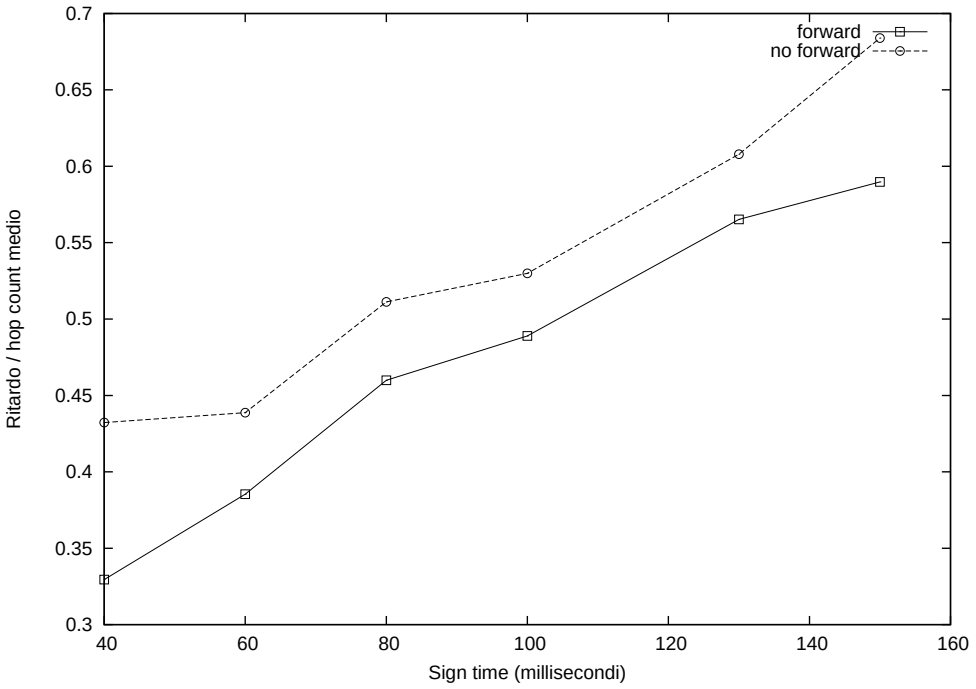


Figura 4.20: Ritardo del primo pacchetto tcp normalizzato rispetto all'hop count medio delle rotte

4.19). Il grafico di figura 4.20 mostra come l'andamento del ritardo normalizzato rispetto al numero di hop medio delle rotte sia molto più interessante. Come si può vedere la versione del protocollo con il forward anticipato dei pacchetti dà sempre risultati migliori rispetto a quella con l'opzione disabilitata. Inoltre, come si può vedere in figura 4.21 anche il numero di pacchetti di controllo generati dal protocollo di routing è nettamente inferiore per quanto riguarda la versione con forward anticipato. Risultati molto simili sono stati ottenuti anche per scenari con bassa mobilità e con geometria quadrata di piccole dimensioni.

Dai risultati ottenuti si può dunque affermare che l'utilizzo del forward anticipato è vantaggioso sia perché porta ad un minor ritardo nell'instaurazione delle rotte, evitanto che il pacchetto resti in attesa della verifica della propria firma, sia perché produce un minor overhead di pacchetti sia perché consente di individuare un numero maggiore di rotte. Gli stessi vantaggi visti sullo scenario rettangolare sono stati misurati anche su quello quadrato nel quale vi sono meno connessioni e nodi.

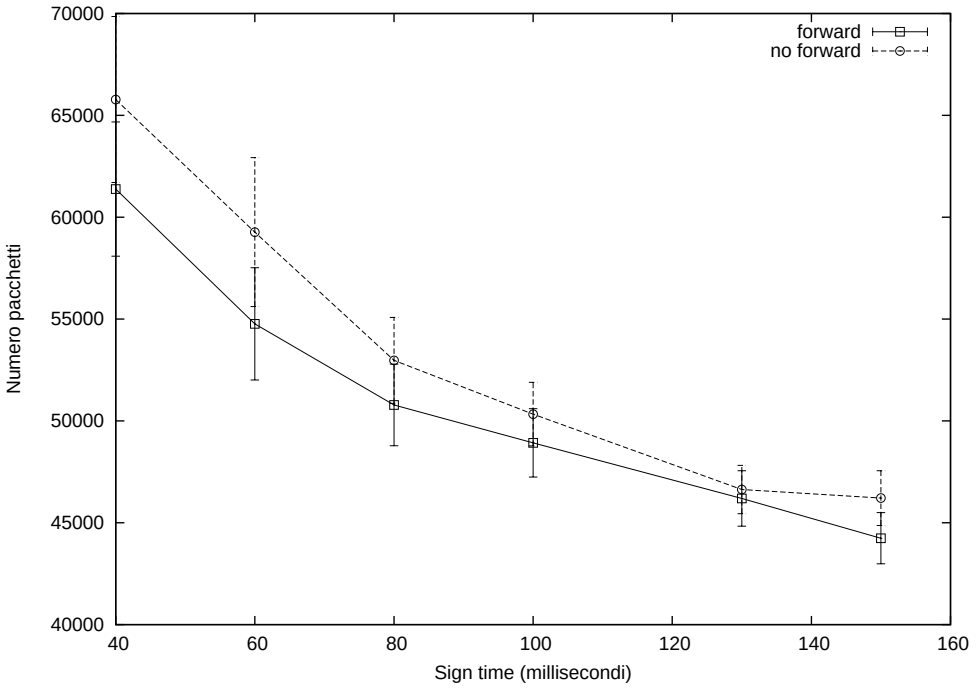


Figura 4.21: Overhead in numero di pacchetti

4.5.3 Simulazione del protocollo SAODV con entrambe le ottimizzazioni

Come si è visto nei precedenti paragrafi le due ottimizzazioni proposte portano, singolarmente, a dei vantaggi nel tempo di discovery. Quello che si vuole studiare ora è l'applicazione contemporanea delle due ottimizzazione per verificare quali possano essere le loro interazioni e se queste portano a vantaggi o svantaggi per i paramentri valutati nelle simulazioni.

Dal grafico di figura 4.22, che si riferisce alle simulazioni su scenario rettangolare con alta mobilità, si può vedere come le versioni adattativa e collaborativa del protocollo siamo molto simili in quanto a ritardo e nettamente migliori della versione non collaborativa. Se però si analizza il parametro normalizzato (come fatto per la valutazione delle altre versioni del protocollo) si può notare che in realtà la versione collaborativa risulta migliore (figura 4.23). L'introduzione del forward limita quindi il guadagno dovuto all'adattatività del protocollo

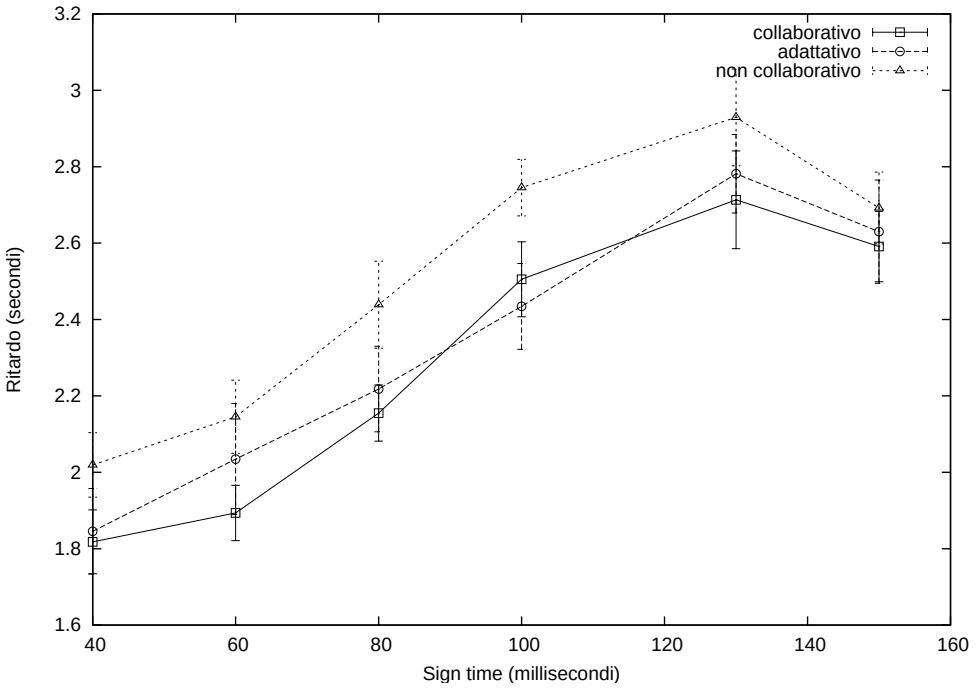


Figura 4.22: Ritardo del primo pacchetto TCP

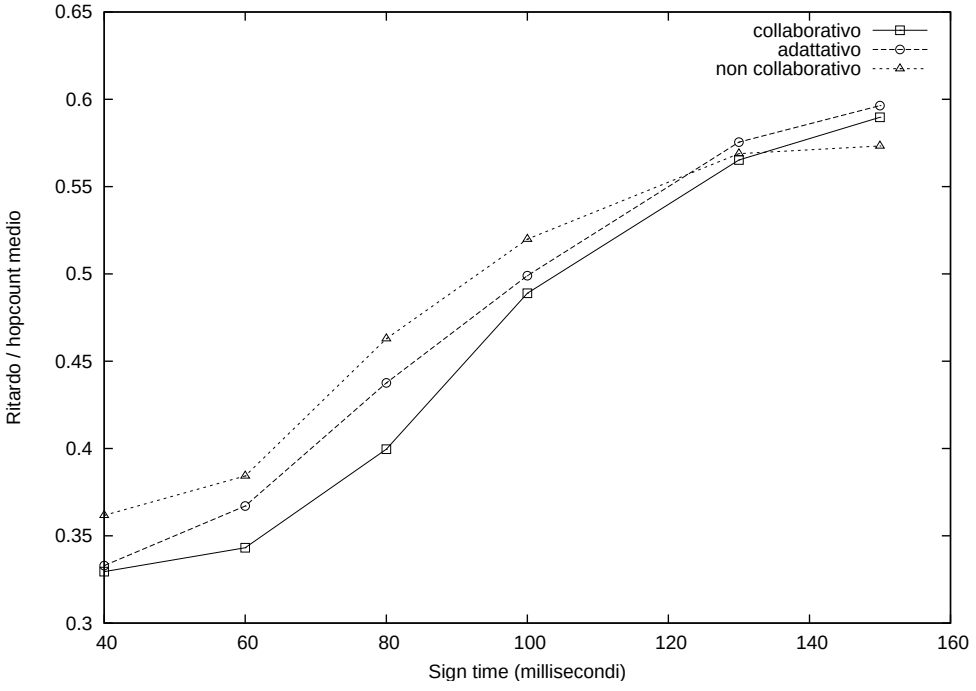


Figura 4.23: Ritardo del primo pacchetto tcp normalizzato rispetto all'hop count medio

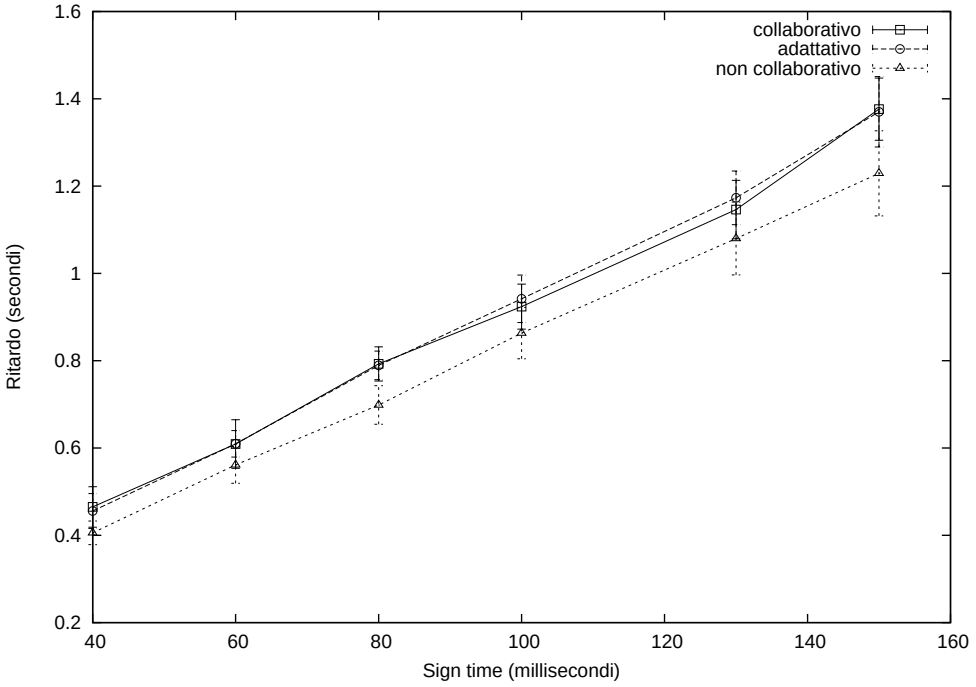


Figura 4.24: Ritardo del primo pacchetto TCP

portando quest'ultima versione ad avere prestazioni minori rispetto alla versione completamente collaborativa.

In figura 4.24 si può vedere come anche su scenari di piccole dimensioni i protocolli collaborativi si comportino in maniera molto simile. Anche in questo caso l'interazione delle due ottimizzazioni non porta ad evidenti guadagni dal punto di vista del ritardo.

Va notato che il forward anticipato non peggiora le prestazioni di nessuna versione del protocollo, bensì le migliora ma in misura diversa. Infatti il minor divario tra le versioni adattativa e collaborativa in presenza di forward anticipato non è dato da un peggioramento della prima variante ma da un miglioramento più deciso della seconda rispetto ad essa.

Dai risultati ottenuti possiamo dunque concludere che è auspicabile l'utilizzo del solo forward anticipato. Infatti questo porta a miglioramenti sia al protocollo nella sua versione collaborativa che in quella adattativa ma in percentuale diversa. Il protocollo collaborativo riceve un vantaggio tale da risultare migliore anche in

senso assoluto rispetto alla versione adattativa, invertendo la situazione del caso in cui non si ha forward anticipato. Inoltre, con il forward anticipato, si hanno prestazioni migliori rispetto al protocollo non collaborativo in tutti gli scenari visti, non risentendo della diversa lunghezza delle rotte nei diversi scenari.

Capitolo 5

Conclusioni

Il protocollo AODV è da tempo stato pubblicato come *Experimental RFC* dal gruppo di lavoro IETF che si occupa di routing nelle Manet. In questo protocollo non vi è tuttavia alcun riferimento alle problematiche di sicurezza che sono però obbligatoriamente da prendere in considerazione quando ci si occupa di reti wireless. Per questo motivo è nato SAODV, un'estensione di AODV, che implementa un meccanismo per rendere sicuro il protocollo.

Il meccanismo su cui si basa SAODV fa uso della crittografia asimmetrica per rendere sicura l'instaurazione di rotte tra i nodi della rete. Come AODV, anche SAODV ha la caratteristica di implementare un meccanismo grazie al quale si può permettere ai nodi della rete di collaborare nel processo di *discovery*, permettendo così, soprattutto se usato in abbinamento con tecniche come l'*expanding ring* per limitare il flooding dei messaggi di richiesta, di rendere più breve e meno dispendiosa per la rete stessa la ricerca di nuove rotte. La strategia utilizzata da SAODV per permettere la collaborazione tra nodi richiede però un maggior numero di operazioni crittografiche, sia a carico del nodo sorgente che dei nodi intermedi, rispetto a una variante *destination only* in cui cioè solo i nodi destinazione della richiesta possono rispondere. Questo porta a uno svantaggio sia in fatto di tempo impiegato per effettuare la discovery sia per quanto riguarda il carico computazionale imposto ai nodi. Si è quindi cercato di ottimizzare il pro-

tocollo in modo da alleggerire il carico della rete e, nel contempo, velocizzare il processo di discovery.Si sono proposte a questo scopo due ottimizzazioni diverse: l'adattatività e il forward anticipato.

La prima ottimizzazione consiste nel consentire ai nodi intermedi di decidere quando rispondere ad una richiesta in base alla valutazione di un parametro che tenga conto del loro carico di lavoro. Nelle simulazioni effettuate si è preso in considerazione come parametro il tempo impiegato dal nodo a smaltire la coda di operazioni crittografiche che ancora necessitano di essere eseguite. Quando questo tempo è superiore ad una soglia allora il nodo non collabora, in caso contrario collabora. Nello stesso modo l'adattatività si applica anche al nodo sorgente nel momento di decidere se produrre messaggi a doppia o singola firma.

Il forward anticipato invece agisce sul meccanismo con i quali i nodi gestiscono i messaggi che devono essere inoltrati. In SAODV ogni messaggio ricevuto da un nodo deve essere verificato prima che possa essere inoltrato e che possano essere fatte altre operazioni da esso dipendenti. Con il forward anticipato si cerca di parallelizzare queste verifiche effettuando l'inoltro del pacchetto prima che sia effettivamente stata verificata la sua autenticità, consentendo al pacchetto di propagarsi nella rete senza dover attendere che il nodo sia in grado di verificare la firma su di esso.

I risultati ottenuti mostrano che entrambe le ottimizzazioni sono vantaggiose. L'adattatività porta a vantaggi in termini di tempi di discovery su scenari con rotte piuttosto lunghe dove, effettivamente, la risposta dei nodi intermedi ha un certo peso. Su scenari con rotte brevi invece si è visto che questa versione del protocollo accusa i medesimi svantaggi della versione collaborativa rispetto alla versione in cui solo i nodi destinazione hanno facoltà di rispondere. Questo è dovuto al fatto che su rotte di due o tre hop la risposta del nodo intermedio non porta a un risparmio significativo in numero di hop percorsi, ed il maggior carico richiesto per effettuarla rispetto alla risposta del nodo destinatario fa sì che le prestazioni decadano. In questi scenari gli altri valori misurati (overhead, pacchetti persi, etc) sono sostanzialmente simili tra le diverse varianti del protocollo.

Il forward anticipato invece si comporta bene in tutti gli scenari simulati. Oltre a portare un vantaggio in termini di tempo di discovery in questo caso si è osservato anche un minor overhead in termini di pacchetti circolanti in rete dovuti al routing.

Visti i risultati positivi delle ottimizzazioni prese singolarmente si sono dunque effettuate delle simulazioni nelle quali fossero entrambe attive. In questo caso si è avuta un'inversione di tendenza rispetto alle simulazioni della sola versione adattativa. Infatti, abilitando anche il forward anticipato, questa versione del protocollo è risultata avere prestazioni minori rispetto a quella completamente collaborativa. Il motivo di questa inversione sta nel fatto che il forward anticipato introduce un miglioramento di prestazioni in percentuale diversa tra le due versioni del protocollo; avvantaggia cioè maggiormente la versione collaborativa rispetto a quella adattativa.

Visti tutti i risultati ottenuti, sia valutando le ottimizzazioni singolarmente che contemporaneamente, si può affermare che è auspicabile l'utilizzo del solo forward anticipato. Infatti, nonostante risultino entrambe soluzioni vantaggiose, il forward anticipato fornisce un vantaggio maggiore in termini di ritardo dovuto alla discovery. Inoltre introduce anche un vantaggio in termini di overhead di traffico dovuto al routing che non è presente nella versione adattativa. L'applicazione contemporanea delle due ottimizzazioni non porta ad un ulteriore vantaggio; infatti l'applicazione del forward al protocollo in versione adattativa porta sì ad un miglioramento del tempo di discovery, ma in maniera più limitata rispetto a quello che introduce nella versione completamente collaborativa, portando quest'ultima a prestazioni migliori.

5.1 Sviluppi futuri

Il lavoro di questa tesi potrebbe essere ancora sviluppato su diversi fronti. Da un lato si potrebbero simulare scenari diversi facendo variare altri parametri di rete (come per esempio il tempo di pausa tra un movimento e il successivo, i

raggi di copertura, etc) per verificare il comportamento delle due ottimizzazioni. Si potrebbero anche effettuare simulazioni volte ad ottimizzare i valori di soglia del caso adattativo, rendendo magari tale valore non fisso ma variabile, dipendente dalla storia della coda del singolo nodo.

Una linea di sviluppo diversa è invece quella di simulare reti non omogenee, cioè nelle quali i nodi abbiano capacità computazionali molto diverse tra loro. In questo senso sarebbe significativo anche non utilizzare tempi di firma e di verifica costanti ma dipendenti dal carico effettivo del dispositivo; per fare questo occorre modellare più in dettaglio il carico delle CPU dei nodi in funzione delle attività svolte per trovare un legame tra i cicli di clock necessari a compiere un operazione crittografica e il tempo realmente impiegato in funzione degli altri processi presenti nel dispositivo.

Inoltre sarebbe interessante simulare il comportamento del protocollo, delle sue varianti e questa volta anche del protocollo non sicuro AODV in scenari in cui siano presenti un o più nodi *attaccanti*. In quest'ottica si potrebbero simulare diversi tipi di attacchi per verificare l'effettiva efficacia del protocollo anche dal punto di vista delle prestazioni. Da questo punto di vista sarebbe interessante verificare l'effettivo impatto delle ottimizzazioni proposte sulla sicurezza. L'adattatività non sembra introdurre problemi legati alla sicurezza; il forward anticipato invece potrebbe esporre la rete ad attacchi di tipo DoS. Infatti, un ripetuto invio di RREQ contraffatte potrebbe saturare le code dei nodi dell'intera rete causando un degrado delle prestazioni la cui entità sarebbe da valutare. Sarebbe quindi interessante studiare il reale impatto di questo problema e proporre eventuali contromisure.

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] IEEE std. 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1997.

[2] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile comunications: the insecurity of 802.11. *ACM MOBICOM*, 2001.

[3] V. Gupta, S. Krishnamurthy, and M. Faloutsos. Denials of service attacks at the MAC layer in wireless ad-hoc networks. *IEEE MILCOM*, 2002.

[4] P. Kyasanur and N. Vaidya. Detection and handling of MAC layer in wireless ad hoc networks. *DCC*, 2003.

[5] R. Negi and A. Perrig. Jamming analysis of MAC protocols. *Carnegie Mellon Technical Memo*, 2003.

[6] T. Clausen and P. Jacquet. Optimized LinkState Routing (OLSR) protocol. RFC 3626, IETF, 2003.

[7] J. Moy. OSPF version 2. RFC 2328, IETF, Aprile 1998.

[8] S. Brandner. Key words for use in RFCs to indicate requirement levels. RFC 2119, IETF, 1997.

[9] T. Clausen, J. Dean, C. Dearlove, and C. Adjih. Generalized manet packet/message format. Internet-Draft Version 01, IETF, Giugno 2006.

[10] R. Ogier, F. Templin, and M. Lewis. Topology dissemination Based on Reverse-Path Forwarding (TBRPF). RFC 3684, IETF, Febbraio 2004.

[11] Y.K. Dalal and R.M. Metcalfe. Reverse Path Forwarding (RPF) of broadcast packets. *CACM*, 21(12):1040–1048, 1978.

[12] I. Chakeres and C. Perkins. Dynamic Manet On-demand (DYMO) routing. Internet-Draft Version 05, IETF, Giugno 2006.

[13] C. Perkins and E.M. Belding. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, IETF, Luglio 2003.

[14] J. Broch, D.B. Johnson, and D.A. Maltz. The Dynamic Source Routing Protocol (DSR) for mobile ad hoc networks. Internet-Draft Version 03, IETF, Ottobre 1999.

[15] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[16] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem,trans. prog. lang. and sys. *Communications of the ACM*, 4(3):382–401, 1982.

[17] M. Guerrero Zapata. Secure Ad hoc Distance Vector (SAODV) routing protocol. Internet-Draft Version 05, IETF, 2006.

[18] M. Guerrero Zapata and N. Asokan. Securing ad hoc protocols. *WiSe, Wireless Security*, Settembre 2003.

[19] G. Montenegro and C. Castelluccia. Statistically Unique and Cryptographically Verifiable (SUCV) identifiers and addresses. *NDSS*, 2002.

[20] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E.M. Belding-Royer. A secure routing protocol for ad hoc network. *IEEE International Conference on Network Protocols (ICNP)*, Novembre 2002.

[21] F. Kargl, A. Geiss, S. Schlott, and M. Weber. Secure dynamic source routing. In *proceedings oh the 38th Hawaii International Conference on System Sciences*, pages 1–15, 2005.

[22] W. Diffie and M. Hellmann. New directions in cryptography. *IEEE Transactions on Information Theory IT*, 22(6):644–654, 1976.

[23] Y.C. Hu, A. Perrig, and D.B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth ACM International Conference on Mobile Computing and Networking (Mobicom2002)*, Settembre 2002.

[24] A. Perrig, R. Canetti, D. Song, and J.D. Tygar. Efficient and secure authentication for multicast. *Network and Distributed System Security Symposium*, pages 35–46, Febbraio 2003.

[25] A. Perrig, R. Canetti, J.D. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. *IEEE Symposium on Security and Privacy*, Maggio 2000.

[26] Y.C. Hu, D.B. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing in mobile wireless ad hoc networks. *Fourth IEEE Workshop on Mobile Computing Systems and Applications*, Giugno 2002.

[27] N. Alagia. Valutazione delle prestazioni di protocolli per il routing sicuro nelle reti mobili ad hoc. *Tesi di laurea in ingegneria delle telecomunicazioni, Politecnico di Milano*, 2003.

[28] B. Wiberg. Porting AODV-UU implementation to NS-2 and enabling trace-based simulation. *Tesi di dottorato di ricerca*, 2002.

[29] S. Kurkowsky, T. Camp, N. Mushell, and M. Colagrosso. A visualization and animation tool for NS-2 wireless simulations: iNSpect. In *Proceedings*

of the 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pages 503–506, Ottobre 2005.

[30] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *Mobile Computing, IEEE Transactions on*, 2003.