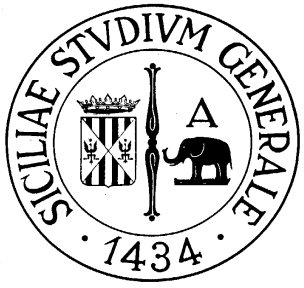

Università degli Studi di Catania

Dipartimento di Matematica e Informatica

Corso di Laurea Magistrale in Informatica

Parasiliti Parracello Cristina

Tesi di Laurea

PRoPHET : attacco alla privacy del protocollo

Relatore:

Salvatore Antonio Riccobene

Anno accademico 2014/2015

Indice

1 INTRODUZIONE

Implementazione e studio del protocollo di routing PRoPHET per dimostrare che, tramite le probabilità di incontro tra nodi su cui si basa il protocollo, si possono ricostruire le conoscenze tra essi.

In particolare viene inserito un nodo spia, che gli utenti vedono come un semplice nodo che rispetta le politiche del protocollo, ma che in realtà tramite le informazioni che ottiene, riesce a ricostruire le interconnessioni tra gli utenti della rete.

L'obiettivo della tesi è quindi dimostrare che la privacy degli utenti che utilizzano il protocollo PRoPRET è violabile.

2 RETI WIRELESS

2.1 Introduzione

Il crescente successo dei dispositivi mobili, come telefoni cellulari, tablet PC e laptop computer apre un nuovo scenario in cui gli utenti richiedono l'utilizzo di servizi internet in ogni momento ed indipendentemente dalla propria posizione fisica.

Le tecnologie wireless risultano essere particolarmente adatte per supportare la mobilità dei dispositivi o per l'installazione di infrastrutture di comunicazione in ambienti difficili.

Le reti wireless sono caratterizzate da comunicazioni broadcast, dove un insieme dinamico di nodi si muove liberamente senza link prestabiliti di interconnessione tra di loro.

Dato che la topologia della rete può cambiare continuamente, bisogna gestire in modo appropriato l'instradamento dei dati tra i nodi, il controllo degli accessi, l'assegnazione degli indirizzi IP ai nodi che si

aggiungono alla rete.

Altre problematiche per questo tipo di rete riguardano la bassa larghezza di banda, disconnessioni frequenti a causa del movimento dei nodi, canali con elevato tasso d'errore e limitata energia a disposizione.

2.2 Topologia delle reti wireless

Le reti wireless possono essere classificate in base a determinati requisiti come riportato dalla figura 1

Reti wireless	WAN	MAN	LAN	PAN	BAN
Applicazione	Reti cellulari	Reti di trasporto e backbone per le reti cellulari	Reti interne agli edifici	Rimpiazzo dei cavetti e piccole reti	Reti di dispositivi d'utente
Distanza/ Copertura	Anche centinaia di km	Fino a qualche decina di km	Fino al centinaio di metri	Fino ad alcuni metri	Centimetri o meno
Larghezza di banda	Dai kbit/s ai Mbit/s	Dai Mbit/s alle decine di Mbit/s	Decine di Mbit/s	Da centinaia di kbit/s ai Mbit/s	Fino a qualche Mbit/s
Esempi di sistemi standardizzati	GSM/UMTS	WiMAX	Wi-Fi	Bluetooth	Non disponibili

figura 1

2.3 Tipi di reti wireless

Esistono molti tipi di reti che possono essere classificate in due grandi categorie in base a come la rete è organizzata:

1) *rete basata sulle infrastrutture*, come ad esempio le reti cellulari, dove la stazione base, generalmente connessa ad una rete cablata, connette i vari dispositivi mobili.
2) *rete ad-hoc*, dove i nodi sono mobili, non ci sono stazioni base che coordinano le attività di un sottoinsieme di nodi.

   Le reti ad-hoc sono adatte per le missioni militari, per le operazioni di soccorso in situazioni di emergenza, etc.

2.4 Tassonomia delle reti wireless

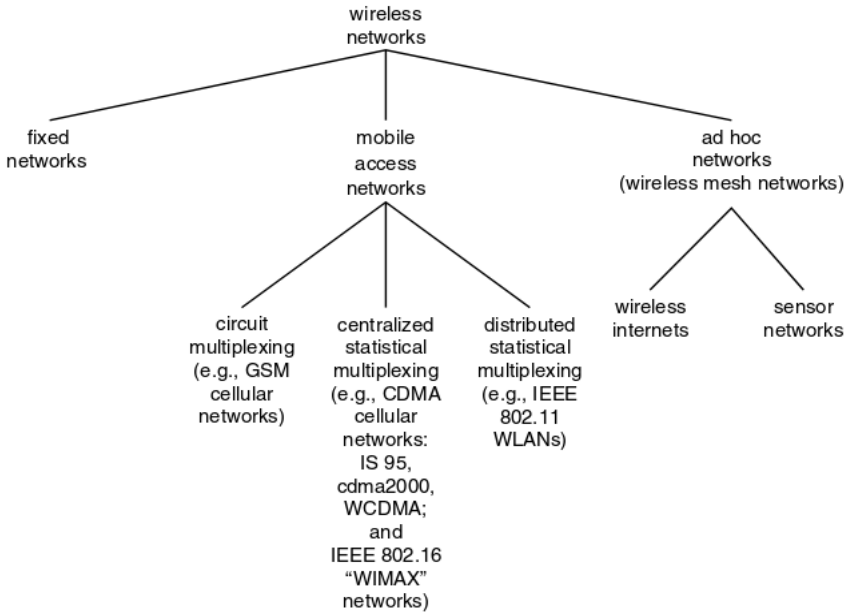

figura 2

Fixed networks: reti con collegamenti a microonde molto utilizzate per trasmissioni a lunga distanza. Normalmente sono formate da collegamenti punto a punto creati tramite antenne ad alto guadagno, posizionate in punti molto elevati come ad esempio lunghi alberi. Questi collegamenti possono essere visti come delle bit pipes punto a punto, quindi con gli stessi problemi dei collegamenti via cavo, ma con bit error rate più elevato;

Mobile access networks: reti ad accesso multiplo nelle quali diversi nodi condividono la stessa porzione di spettro radio per comunicare tra loro.

Le mobile access networks si suddividono a loro volta in:

- *Circuit multiplexing (reti cellulari GSM)*: l'area coperta dalla

rete è suddivisa in celle ed in ognuna di queste c'è una stazione base collegata alla rete cablata, attraverso la quale i nodi mobili all'interno della cella comunicano.

I nodi condividono lo spettro disponibile utilizzando la tecnologia FDM-TDMA (frequency division multiplexed time division multiple access);

- *Centralized statistical multiplexing (reti cellulari CDMA, reti WIMAX)*: in questo tipo di reti l'intero spettro non è suddiviso in base alle frequenze ma è interamente riutilizzato in ogni cella. Questo sistema prende il nome di CDMA (code division multiple access);
- *Distributed packet scheduling cellular networks*: tipologia di rete in cui la condivisione di un canale viene risolta tramite protocollo distribuito di acquisizione. Molto utilizzate nelle WLAN (Wireless Local Area Network).

Ad Hoc Networks: reti formate da diversi device (nodi) sparsi nello spazio, equipaggiati con ricetrasmittenti radio, che condividono la stessa radio frequenza.

2.5 Caratteristiche e vantaggi delle reti mobili ad-hoc (MANET)

Le MANET si basano sulla comunicazione cooperativa tra nodi possibilmente omogenei, senza la necessità di punti di accesso fissi e permettono la formazione di configurazioni dinamiche e sempre in mutamento.

Ogni nodo non opera più solo come host, ma anche come router, occupandosi quindi dell'instradamento dei pacchetti.

Le reti MANET sono reti *multi hop*, dove le informazioni vengono trasferite attraverso diversi nodi per inoltrarle a un destinatario finale che può essere anche molto lontano.

Ogni nodo utilizza gli altri dispositivi presenti nell'area per far arrivare le informazioni al destinatario. Se uno dei nodi cade perché, ad esempio, ha la batteria scarica o si è spostato in una zona troppo lontana o il proprietario ha deciso di spegnerlo, la rete è capace di riconfigurarsi e trovare comunque una via per far comunicare tra loro i vari nodi della rete.

3 DELAY TOLERANT NETWORKS

Delay tolerant networks (DTN) è un'architettura di rete ad-hoc, il cui obiettivo è far comunicare tra loro reti indipendenti e non appartenenti ad una rete globale continuamente interconnessa.

Queste reti sono caratterizzate da ritardi di trasmissione, da perdite di connessioni e da alti tassi di errore.

DTN utilizza un trasferimento di tipo *store and forward message switching*, dove ogni nodo memorizza in modo persistente l'intero contenuto del messaggio (s*tore message)* o una sua segmentazione, per poi trasferirlo in blocco (*forward message)* al nodo successivo.

Il meccanismo di trasferimento è basato su sessioni, dove lo scambio può continuare senza richiedere necessariamente un messaggio di conferma da parte di un nodo ricevente.

Si definisce un livello, end-to-end e orientato ai messaggi, chiamato *bundle layer,* che stà al di sotto del livello applicativo e al di sopra di quello di trasporto. Ogni device che implementa questo livello è chiamato nodo DTN.

Il bundle layer lavora con memoria persistente, in modo da essere affidabile in caso di interruzioni di rete.

4 PRoPHET

4.1 Introduzione

Anche se i modelli di mobilità random dei nodi sono molto utilizzati per valutare protocolli di rete ad hoc, nelle situazioni reali gli utenti non hanno comportamenti casuali, anzi al contrario, si muovono in modo prevedibile. Se una persona ha visitato un luogo più volte precedentemente, è probabile che lo rivisiti in futuro.

PRoPHET (Probabilistic Routing Protocol using History of Encounters and Transitivity) è un protocollo di routing, che fa parte della famiglia delle DTN e tiene conto della storia degli incontri tra i nodi e ne calcola le probabilità, in modo da prevedere i possibili incontri futuri tra i nodi.

Contiene degli elementi simili al protocollo Epidemic Routing, riducendone però il flooding.

4.2 Epidemic Routing

Epidemic Routing è un protocollo DTN che si basa sul flooding dei messaggi. Ogni nodo ha una tabella chiamata *summary vector* contenente i messaggi ricevuti.

Quando due nodi mobili si incontrano, si scambiano le proprie summary vector per determinare se esistono messaggi non ancora conosciuti da una delle due parti ed in caso positivo inserirli nella propria tabella. Quindi fino a quando i nodi hanno spazio nel proprio buffer di messaggi, essi vengono diffusi come un'epidemia.

Ogni messaggio contiene un ID globalmente univoco, in modo da

determinare se è stato già incontrato o meno, contiene inoltre i campi sorgente, destinatario ed un campo contatore di salti, in modo da stabilire un numero totale di inoltri del messaggio.
L'epidemia di messaggi è quindi regolata sia dal numero di salti che dalla grandezza del buffer di messaggi di ogni nodo.

4.3 PRoPHET Routing

PRoPHET si basa su una metrica probabilistica chiamata *delivery predictability, P (a,b)* ∈ *[0, 1]*, per ogni nodo *a (source)* ed ogni sua destinazione conosciuta *b.*
Le operazioni sono simili a quelle dell'Epidemic Routing, in particolare quando due nodi si incontrano, si scambiano sia le *summary vector* che le *delivery predictability*, in modo che ognuno di essi aggiorni il vettore di delivery predictability, utilizzato per decidere quali messaggi possono esseri inoltrati al nodo.
Il calcolo della metrica su cui si basa PRoPHET è suddivisa in tre parti:

Aggiornamento dopo ogni incontro

Viene aggiornata la probabilità ogni volta che un nodo viene incontrato, in modo che i nodi che si incontrano spesso abbiamo probabilità alta:

$$P_{(a,b)} = P_{(a,b)old} + (1 - P_{(a,b)old}) \times P_{init} \qquad (1)$$

dove $P_{init} \in [0, 1]$ è una costante di inizializzazione.

Aging

Se una coppia di nodi non si incontra poco dopo, loro probabilmente non saranno dei “buoni” nodi di inoltro; si deve quindi aggiornare la loro probabilità, diminuendone il valore:

$$P_{(a,b)} = P_{(a,b)old} \times \gamma^{k} \qquad (2)$$

dove $\gamma \in [0, 1)$ è una costante di aggiornamento e k è il numero di unità di tempo che sono passate dall'ultima volta che la metrica è stata ridotta.

Transitiva

Se un nodo A incontra frequentemente un nodo B e il nodo B incontra frequentemente un nodo C, allora il nodo C probabilmente è un buon nodo per inoltrare messaggi destinati al nodo A:

$$P_{(a,c)} = P_{(a,c)old} + (1 - P_{(a,c)old}) \times P_{(a,b)} \times P_{(b,c)} \times \beta \qquad (3)$$

dove $\beta \in [0, 1]$ è una scala costante che decide quanto grande impatto la transitività deve avere sulla delivery predictability.

5 PRoPHET: attacco alla privacy del protocollo

L’obiettivo della tesi è quello di attaccare la privacy del protocollo provando a ricostruire il grafo delle conoscenze dei vari nodi, tramite un nodo spia che partecipa al protocollo comportandosi come un qualsiasi altro nodo.

5.1 Ambiente di sviluppo

Per implementare e testare il protocollo è stato utilizzato un simulatore di rete NS2 (network simulator 2) in ambiente Linux.

Ns è un simulatore ad eventi discreti destinato alla ricerca, che fornisce un buon supporto per molti tipi di protocolli sia su reti cablate che wireless.

E' stato sviluppato presso l'University of Southern California's Information Sciences Institute (ISI).

E' composto da un motore di simulazione scritto in C++ e da un modulo di interazione utente-simulatore che utilizza il linguaggio Otcl (Object-oriented Tool Command Language).

Le classi C++ che costituiscono NS2 implementano l'insieme dei protocolli disponibili, fornendo così allo sviluppatore la capacità di programmare i nodi gestendo algoritmi efficiente, header, strutture dati, ecc.

L'ambiente Otcl permette la definizione di classi Otcl direttamente connesse alle classi C++.

Fornisce i metodi per l'utilizzo delle classi e consente di creare gli oggetti della rete ed i relativi collegamenti fra gli stessi.

È stato necessario implementare da zero il protocollo PRoPHET dato che tra le librerie di NS2 non era presente, ed è stato fatto in Otcl per facilitare la velocità di test del codice, dato che Otcl è interpretato senza necessità di compilazione ad ogni modifica.

5.2 Implementazione di PRoPHET

I tipi di nodi utilizzati nella simulazione sono nodi mobili con interfaccia wireless, configurati nel seguente modo:

```
set val(chan) Channel/WirelessChannel    ;#Channel Type
set val(prop) Propagation/TwoRayGround ;#radio-propagation model

set val(netif) Phy/WirelessPhy  ;# network interface type

set val(mac) Mac/Simple


set val(ifq) Queue/DropTail/PriQueue  ;# interface queue type
set val(ll) LL  ;# link layer type
set val(ant) Antenna/OmniAntenna  ;# antenna model
set val(ifqlen) 50   ;# max packet in ifq
```

figura 3

Sono stati utilizzati 8 nodi, di cui un nodo *spia*, che servirà per ottenere tutte le informazioni relative alle comunicazioni tra i nodi della rete e ricostruire il grafo delle conoscenze tra essi.

Lo scenario è costituito da un'area totale di 1600x1600 metri, i nodi si possono muovere in un'area di 800x800 metri, suddivisa in quattro zone, scelte ai quattro angoli dell'area di movimento.

Sono stati utilizzati diversi file di trace, per tracciare le principali operazioni della simulazione e salvare delle informazioni utili per l'attacco alla privacy del protocollo, come segue:

```
set ftracer [open wireless-prophet-tracer.tr w] ;#trace all operation
set fspyer [open wireless-prophet-spyer.tr w] ;#trace all the \
              delivery predictability of nodes known by the spy

set fgraphSpy [open wireless-prophet-graph_spy.xml w] ;#create \
                                        xml used for graph creation

proc writeFT { data } { #proc for write on tracer file
       global ftracer
       puts -nonewline $ftracer $data
}
proc writeFS { data } { #proc for write on spyer file
       global fspyer
       puts -nonewline $fspyer $data

}
```

figura 4

È stata creata una nuova classe *Prophet*, sottoclasse della classe *MessagePassing* e definite le tre procedure per il calcolo della delivery predictability nel seguente modo:

Aggiornamento dopo ogni incontro (Eq.1)

```
#PRoPHET delivery predictability update when node a meet node b
Agent/MessagePassing/Prophet instproc meetNodeProb {nodea nodeb } {
       global delivery_pred Pinit ns k a

       set delivery_pred($nodea,$nodeb) [format "%.4f" [expr \
                        ($delivery_pred($nodea,$nodeb) + ((1 - \
              $delivery_pred($nodea,$nodeb) )) * $Pinit)]]

       set current_time [format "%.2f" [expr {double(round(100* \
                                              [$ns now]))/100}]]

       set delivery_pred($nodea,$nodeb,"time") $current_time
       writeFT "\n$delivery_pred($nodea,$nodeb,"time") \
                     -meetNodeProb- => prob($nodea,$nodeb) \
                     :$delivery_pred($nodea,$nodeb)"
}
```

figura 5

Aging (Eq.2)

```
#PRoPHET delivery predictability aging update
Agent/MessagePassing/Prophet instproc agingNodeProb {nodea nodeb } {
      global ns k Gamma delivery_pred sim_stop
      set current_time [format "%.2f" [expr {double(round(100*[$ns \
                                                now]))/100}]]

      set delivery_pred($nodea,$nodeb) [format "%.4f" [expr \
                  ($delivery_pred($nodea,$nodeb)    *    $Gamma**$k)]]

      set delivery_pred($nodea,$nodeb,"time") $current_time

      writeFT "\n$delivery_pred($nodea,$nodeb,"time") \
            -agingNodeProb- => prob($nodea,$nodeb)\
                  :$delivery_pred($nodea,$nodeb)"
}
```

figura 6

La procedura di aging viene richiamata ogni *k* unità di tempo, dalla procedura *agingNodeProbManager* che gestisce l'aging per ogni probabilità, della delivery predictability del nodo, come segue:

```
#Aging manager for every delivery predictability
Agent/MessagePassing/Prophet    instproc    agingNodeProbManager
{nodea} {
      global ns k delivery_pred a num_nodes sim_stop
      $self instvar node_

      for {set i 0} {$i<$num_nodes} {incr i} {

            if {( $i != $nodea ) && \
                  ( $delivery_pred($nodea,$i) != 0 )} {

                  $a($nodea) agingNodeProb $nodea $i

            }
      }
```

figura 7

Questa procedura viene richiamata solo una volta, quando un nodo riceve un pacchetto, per poi essere richiamata ricorsivamente dopo *k* unità di tempo

```
        #every k unit time aging function is called
        if {$sim_stop == 0} {
              $ns after $k "$a($nodea) \
               agingNodeProbManager $nodea"
        }
}
```

figura 8

3. **Transitiva** (Eq. 3)

```
#PRoPHET delivery predictability transitive update when
                                          node a meet node b
Agent/MessagePassing/Prophet instproc transitiveNodeProb \
                                        {nodea nodeb nodec} {
        global ns k Gamma Beta delivery_pred a

        set current_time [format "%.2f" [expr \
                    {double(round(100*[$ns now]))/100}]]
        set delivery_pred($nodea,$nodec) [format "%.4f" \
              [expr ($delivery_pred($nodea,$nodec) + (1 - \
                          $delivery_pred($nodea,$nodec)) * \
                            $delivery_pred($nodea,$nodeb) * \
                  $delivery_pred($nodeb,$nodec) * $Beta )]]
        set delivery_pred($nodea,$nodec,"time") $current_time

        writeFT "\n$delivery_pred($nodea,$nodec,"time") - \
        transitiveNodeProb- => prob($nodea,$nodec)\
                    :$delivery_pred($nodea,$nodec)"
}
```

figura 9

Sono state poi definite le due procedure che permettono lo scambio di pacchetti tra i nodi della rete, in particolare:

- L'**invio dei pacchetti** in broadcast da parte di un nodo è definita dalla procedura *send_tables*, che invia il messaggio all'indirizzo di broadcast (BROADCAST_ADDR) tramite la funzione *sendto* e scrive nel file di trace l'invio di esso.

```
#send packet function
Agent/MessagePassing/Prophet instproc send_tables \
                        {size packet_id data port} {
      $self instvar messages_seen node_
      global ns MESSAGE_PORT BROADCAST_ADDR
      lappend messages_seen $packet_id
      set current_time [format "%.2f" \
      [expr {double(round(100*[$ns now]))/100}]]
      writeFT "\n$current_time -sendPacket- => pkt_id\
            :$packet_id pkt_sender:[$node_ node-addr]"

      #send packet in broadcast
      $self sendto $size "$packet_id:$data" \
                        $BROADCAST_ADDR $port
}
```

figura 10

- La **ricezione dei pacchetti** è definita dalla procedura *recv* che viene richiamata automaticamente quando un nodo riceve un pacchetto. Questa procedura si comporta in modo differente, in base al tipo di nodo che riceve il pacchetto.

Se il nodo ricevente è un nodo normale, allora vengono richiamate tutte le procedure per il calcolo della delivery predictability (Eq.1, Eq.2, Eq.3) come segue:

```
#check if node is normal(type=0) or spy(type=1)
if {$a($packet_receiver,'type') == 0} {
	#normal node

	writeFT "\n$current_time -recvPacket- => pkt_id\
		:$packet_id (NormalNode) pkt_source\
		:$source pkt_receiver:$packet_receiver"



	#delivery predictability algorithm start

	#start aging only if is first time that node (receiver) \
				receive a packet (flag_aging = 0)
	if {$delivery_pred($packet_receiver,"flag_aging") != 1} {
		set delivery_pred($packet_receiver,"flag_aging") 1
		$a($packet_receiver) agingNodeProbManager \
					$packet_receiver
	}

#set meet delivery predictability
$a($packet_receiver) meetNodeProb $packet_receiver $source
# add node met in friends list
if { [lsearch $friends($packet_receiver) $source] == -1 } {
	if {$friends($packet_receiver) == "null"} {
		set friends($packet_receiver) $source
	} else {
		lappend friends($packet_receiver) $source
	}
	writeFT "\n$current_time -addNewFriend- => node\
	:$packet_receiver friends:$friends($packet_receiver)"
}
 #set transitive delivery predictability for every node \
 				that aren't receiver friend
 #(not set if is pkt_sender or is receiver's friend or \
 							is it self)
 for {set x 0} {$x < $num_nodes} {incr x} {
 	if { ([lsearch $friends($packet_receiver) $x] == -1)\
 	 && ( $x != $source) && ( $x != $packet_receiver ) \
 	 && ( $delivery_pred($source,$x) != 0)} {
 		#set transitive delivery predictability
 		$a($packet_receiver) transitiveNodeProb \
 				$packet_receiver $source $x

 	}
 }


 #delivery predictability algorithm finish
```

figura 11

Se invece il nodo ricevente è il nodo spia, viene trascritta la tabella relativa alla delivery predictability del nodo sorgente (chi ha inviato il messaggio) in un file di trace, così che il nodo spia alla fine abbia tutti i dati per poter ricostruire il grafo delle conoscenze tra i nodi della rete

```
else { #spy node
      writeFT "\n$current_time -recvPacket- => pkt_id\
            :$packet_id (SpyNode) pkt_source\
            :$source pkt_receiver:$packet_receiver"

      #save all delivery predictability of node met
      for {set i 0} {$i<$num_nodes} {incr i} {
            if {$array_spy($source) == "null"} {
                  set array_spy($source) \
                        $delivery_pred($source,$i)
            } else {
                  lappend array_spy($source) \
                        $delivery_pred($source,$i)
            }
      }
      set var_source $source
```

figura 12

5.3 Scenario di simulazione

La configurazione generale della rete utilizzata nei test del protocollo è la seguente:

- Area totale: 1600x1600 metri
- Area di spostamento: 800x800metri
- Numero di nodi: 8 (7 normali, 1 spia)
- Durata simulazione: 150.0 secondi
- Dimensione pacchetti: 50 byte

Le costanti per il calcolo della *delivery predictability* sono state settate con i seguenti valori:

- P_{init} : 0.75
- β : 0.25
- γ : 0.98
- k : 2

L'area di spostamento è stata suddivisa in quattro zone, identificate dalle coordinate *x* e *y* come segue:

```
#simulation zones
set z0_x 1
set z0_y 1

set z1_x 1
set z1_y 799

set z2_x 799
set z2_y 799

set z3_x 799
set z3_y 1
```

figura 13

La posizione iniziale dei nodi è la seguente:

$Z_0 \rightarrow$ node 0, node 1

$Z_1 \rightarrow$ node 2

$Z_2 \rightarrow$ node 3, node 4, node 5

$Z_3 \rightarrow$ node 6

E' possibile vedere in figura lo scenario iniziale dopo l'assegnazione della posizione iniziale dei nodi

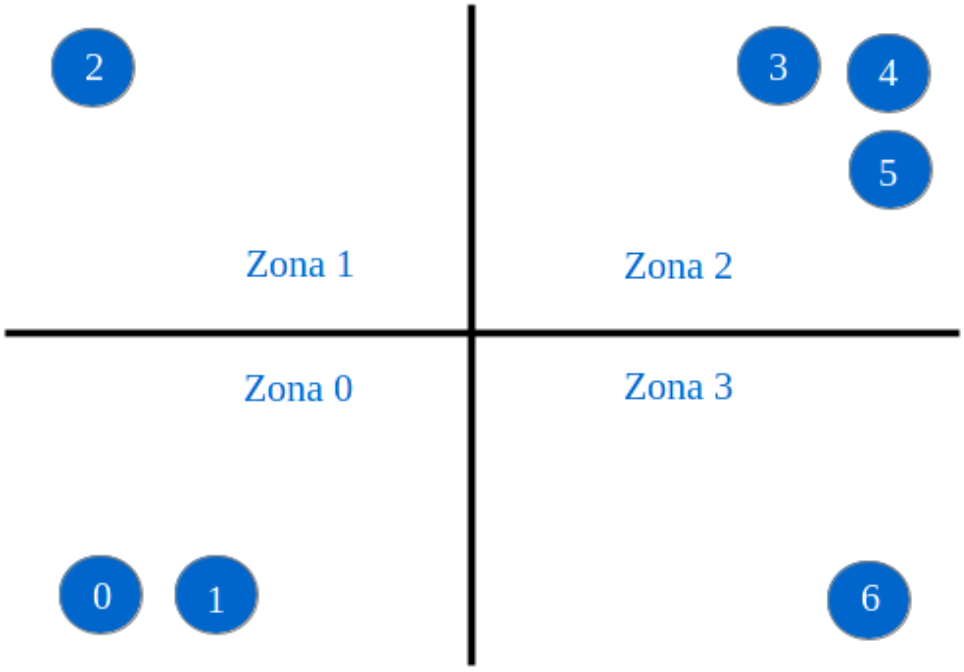

figura 14

I nodi si muovono in diverse zone, facendo “amicizia” (scambiandosi quindi le tabelle) con i nodi che incontra, durante tutto il periodo della simulazione.

In particolare gli incontri che ogni nodo fa durante la simulazione, sono i seguenti:

link	**# incontri**	**link**	**# incontri**
0 – 1	8	1 – 2	0
0 – 2	1	1 – 3	1
0 – 3	2	1 – 4	4
0 – 4	5	1 – 5	1
0 – 5	2	1 – 6	0
0 – 6	0		

link	**# incontri**	**link**	**# incontri**
2 – 3	4	3 – 4	15
2 – 4	2	3 – 5	15
2 – 5	2	3 – 6	0
2 – 6	2		

link	**# incontri**
4 – 5	15
4 – 6	0
5 – 6	0

I link con i valori a zero, rappresentano quei nodi che non si sono mai incontrati durante tutta la simulazione.

Si ha così un numero totale di 15 link, che rappresentano i collegamenti fra i nodi.

Il grafo delle conoscenze tra i nodi della rete, alla fine della simulazione è il seguente:

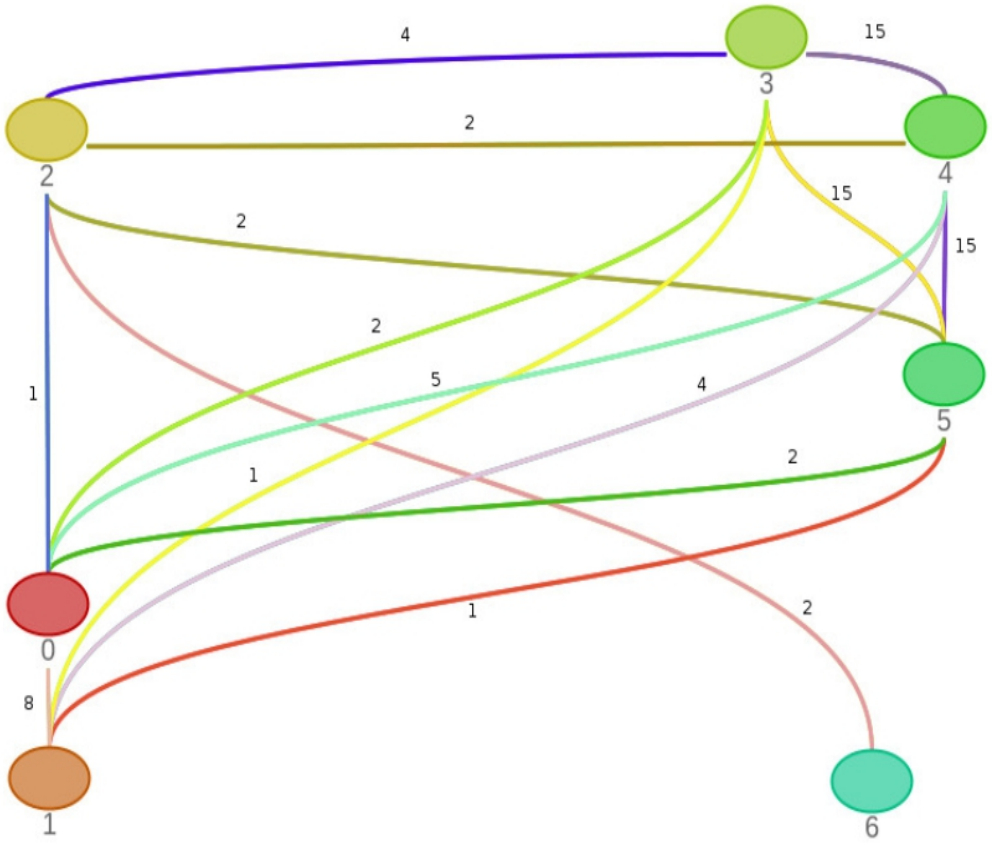

figura 15

dove i pesi sui link indicano il numero di volte che quei nodi si sono incontrati e quindi il numero di volte che si sono inviati pacchetti e aggiornate le proprie delivery predictability.

E' stato inserito all'interno dello scenario il nodo spia, posizionato inizialmente al di fuori dell'area di spostamento.

Alla fine di tutti i movimenti dei nodi, il nodo spia è stato fatto spostare in due differenti zone (zona 3 e zona 1) ottenendo le delivery predictability dei nodi 6 e 2 rispettivamente.

Queste informazioni vengono successivamente utilizzate per cercare di ricostruire in modo più fedele possibile il grafo delle conoscenze tra i nodi.

In particolare il nodo spia verifica se ogni probabilità all'interno della delivery predictability, che ha ottenuto, è maggiore di una certa soglia. Se l'esito è positivo allora considera i due nodi "amici" e crea un link tra essi, ottenendo alla fine un grafo delle conoscenze tra i nodi, tramite file xml

```
	#create xml file for rappresent graph of spy's known
	puts -nonewline $fgraphSpy "<nodes>\n \
	            <node>$var_source</node> \n"

	for {set j 0} {$j<$num_nodes} {incr j} {
		puts -nonewline $fspyer \
		"prob($source,$j)=$delivery_pred($source,$j) "

		if {$delivery_pred($source,$j)>$threshold} {

			puts -nonewline $fgraphSpy "  \n   \
				<friend>$j</friend> \n"
		}
	}
	puts -nonewline $fgraphSpy "</nodi> \n\n"
}
```

figura 16

5.4 Risultati ottenuti

Sono state effettuate diverse prove, al variare del valore della soglia, per verificare quanto il nodo spia riesca a ricostruire in modo fedele il grafo delle conoscenze, visitando solamente due zone dell'intera area di spostamento.

Su un totale di 15 link reali (figura 15) che rappresentano i veri incontri tra i nodi, il nodo spia è riuscito ad ottenere:

- con una **soglia di 0.10** → 8 link di cui 4 amicizie reali e 4 amicizie transitive (2 – 1, 6 – 3, 6 – 4, 6 – 5)

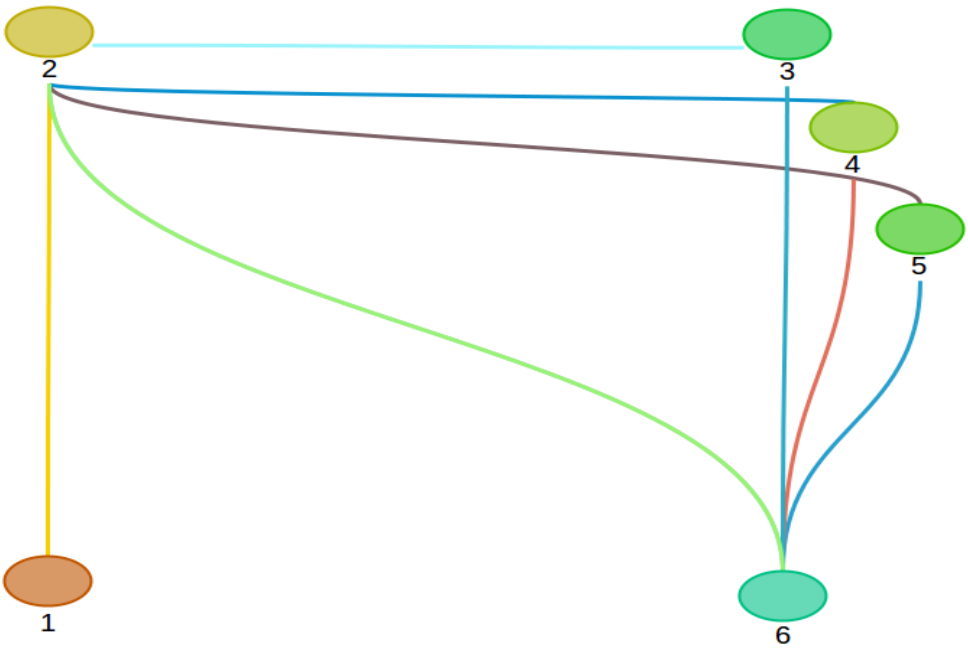

figura 17

La soglia scelta risulta essere troppo bassa, in quanto si ottengono anche dei falsi positivi.

- con una **soglia di 0.30** → 4 link di cui 4 amicizie reali

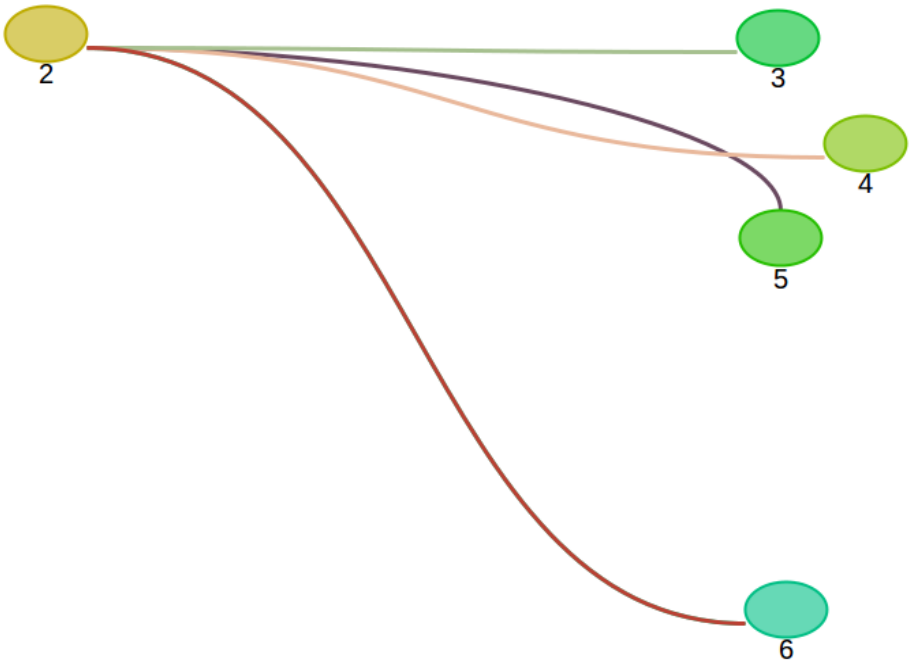

figura 18

Si ottengono delle amicizie reali, in particolare si riferiscono ad incontri frequenti, ma non fatti di recente.

- con una **soglia di 0.38** → 3 link di cui 3 amicizie reali

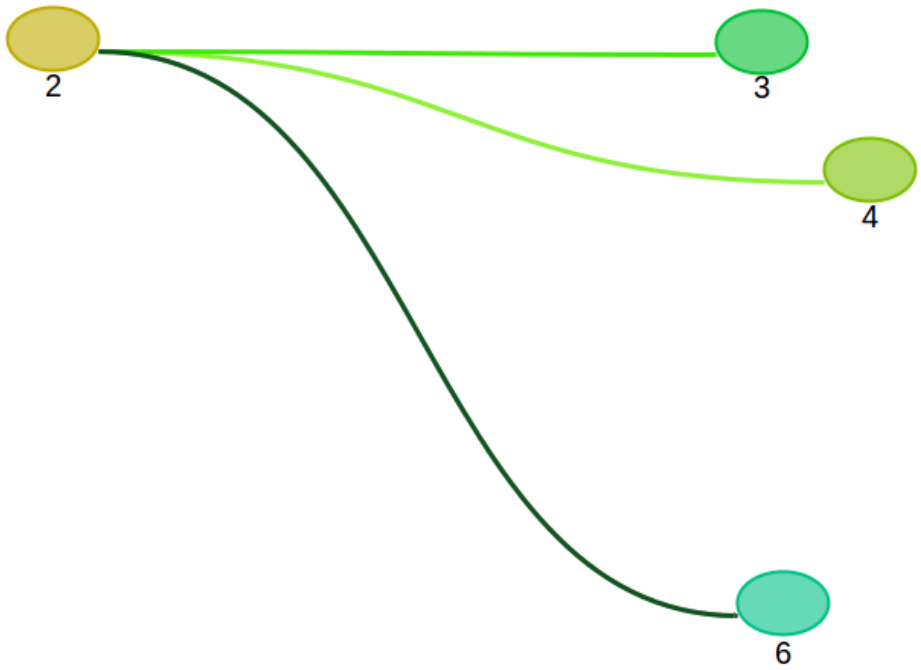

figura 19

Si ottengono delle amicizie reali, create di recente.

I risultati ottenuti evidenziano che a partire da una soglia di 0.30 si ottengono solamente amicizie reali, fedeli al grafo delle conoscenze, in particolare vengono scoperte le amicizie più frequenti tra i nodi.
Incrementando il valore della soglia, da un valore di 0.38 in poi, le amicizie risultano diminuire in quanto vengono considerate solo le conoscenze recenti tra i nodi, rimanendo comunque fedele a quelle reali.

La panoramica generale rispetto ai risultati ottenuti dalle varie simulazioni fatte, è rappresentata dai seguenti grafici:

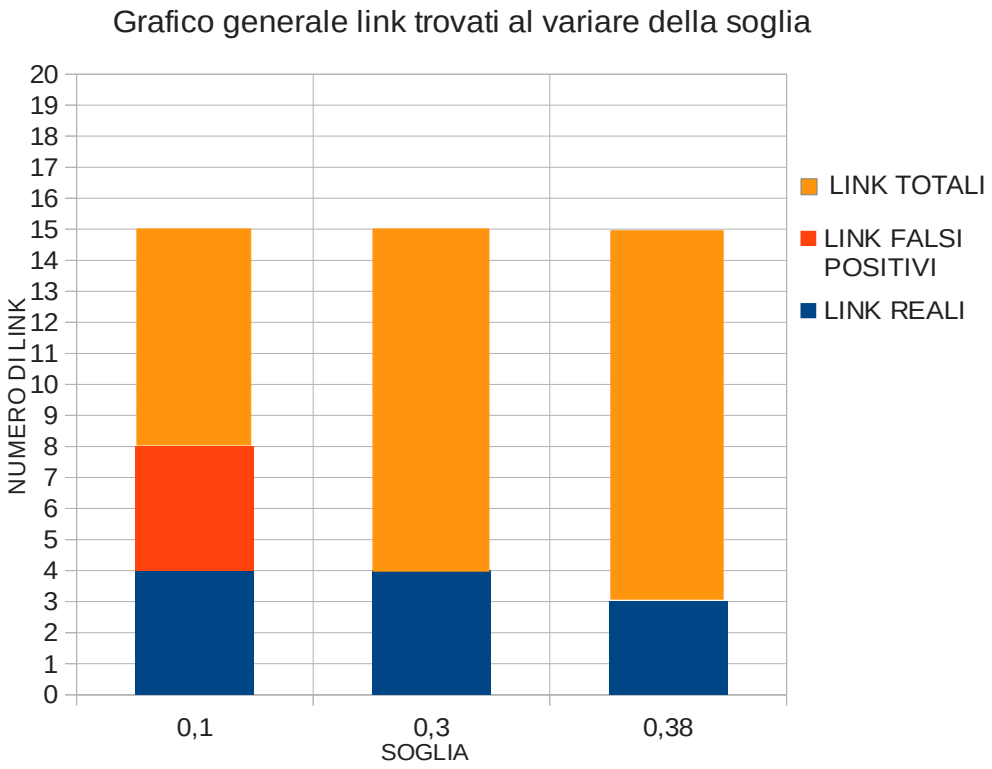

figura 21

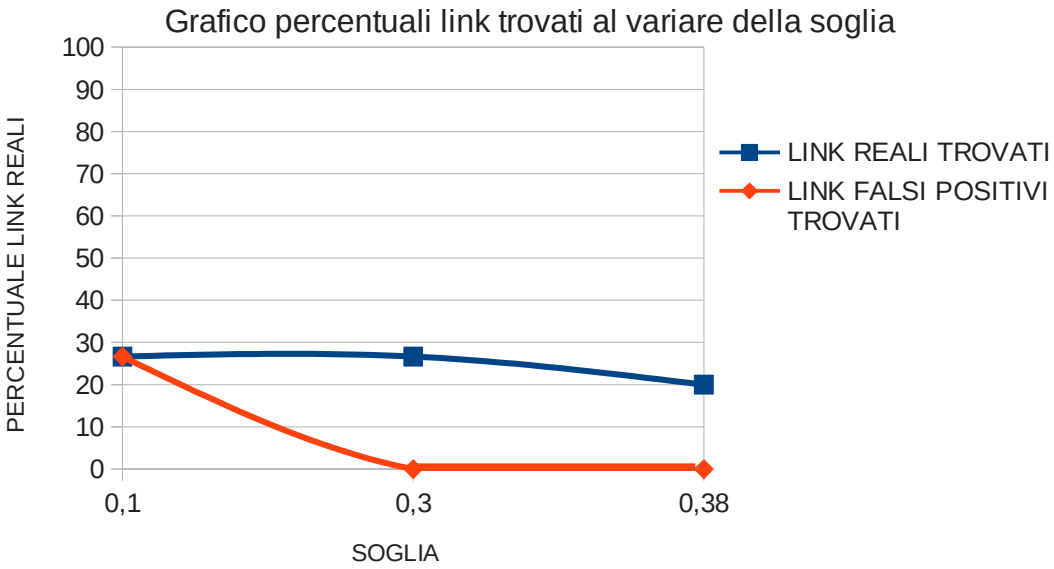

figura 22

5 CONCLUSIONI

Il nodo spia riesce ad avere informazioni private su nodi che non ha mai conosciuto, pur rispettando le politiche del protocollo PRoPHET.

In questo specifico scenario proposto, dove il nodo spia visita solamente due zone, ottiene informazioni sui nodi 3, 4 e 5 mai conosciuti e dalle informazioni ricavate, scopre che questi tre nodi frequentano spesso il nodo 2, può presupporre quindi che questi nodi frequentino la stessa zona.

E' quindi evidente che utilizzando il protocollo PRoPHET, la privacy dell'utente è attaccabile, per questo l'uso del protocollo dovrebbe essere limitato a situazioni in cui la posizione o le conoscenze tra i vari utenti può essere conosciuta da estranei, come ad esempio sistemi di sensori di monitoraggio per emergenze ambientali o militari, dove l'obiettivo non è di certo la privacy dei sensori.

Ovviamente estendendo lo spostamento del nodo spia in tutte le zone, si potrebbero ottenere più informazioni e quindi avere una prospettiva più dettagliata delle conoscenze tra i nodi della rete.

Si potrebbero anche eseguire dei test in cui il nodo spia partecipa più a lungo allo scenario, fermandosi anche per più tempo nelle zone dei nodi, in modo da riuscire ad ottenere informazioni più dettagliate sui posti frequentati e sulle conoscenze di un determinato nodo.

6 RIFERIMENTI E CODICE SORGENTE

- Prophet - Probabilistic Routing in Intermittently Connected Networks - Anders Lindgren, Avri Doria, Olov Schelen :
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.85.4880&rep=rep1&type=pdf

- DTN Rfc :
https://tools.ietf.org/html/rfc4838

- Manuale NS2 :
http://www.isi.edu/nsnam/ns/

- Codice sorgente:
https://github.com/CryPara/ProphetPrivacyAttack.git