Università degli studi di Catania

Corso di Laurea Magistrale in Informatica

P2P and wireless networks A.A. 2014-15

Danilo Cantarella e Sebastiano Siragusa

Configurazione e rilevamento di nodi malevoli nel protocollo DSR

Introduzione

Lo scopo del progetto è stato quello di modificare il protocollo DSR per aggiungere dei nodi malevoli e rilevarli successivamente attraverso l'osservazione diretta da parte degli altri nodi della rete.
Per ogni nodo infatti viene settata una percentuale di essere malevolo e quindi di scartare i pacchetti ricevuti dagli altri nodi. Osservando il comportamento di ogni nodo si potrà quindi capire se si sta comportando normalmente o meno.

L'idea di base è quella di leggere da un file di testo le percentuali di essere malevolo per ogni nodo. Questa percentuale indica la quantità di pacchetti che il nodo dovrà scartare invece di effettuare il normale forwarding verso il next hop della route.

Ogni nodo della rete ascolterà tutti gli altri nodi nel suo raggio di ricezione per cercare di capire come questi si stanno comportando.
Da questa osservazione sarà possibile estrarre delle stime sul reale comportamento dei nodi della rete.

L'obiettivo è quello di ottenere delle stime che siano prossime alle percentuali di malignità settate a priori per i nodi.

Modifiche al DSR

Per prima cosa all'interno del DSRAgent sono stati aggiunti:

1. una variabile **perc_malicious** di tipo double;
2. un oggetto **myBank** di tipo Bank;
3. un oggetto **myMonitor** di tipo Monitor.

Queste variabili vengono inizializzate nel costruttore del DSRAgent.
In particolare, la variabile **perc_malicious** viene settata nel **DSRAgent::command**, richiamato da TCL attraverso il comando:

```
$ns at 0 "[$node($i) set ragent_] malicious $line"
```

dove $line rappresenta la percentuale letta direttamente da un file di testo.
In questo modo ogni nodo saprà la sua percentuale di essere malevolo e potrà comportarsi di conseguenza.

Per la gestione della perdita dei pacchetti, è stato inserito il seguente codice nella funzione **handleForwarding,** che a sua volta viene richiamata dalla funzione **recv** nel momento in cui il DSRAgent di un nodo riceve un qualsiasi pacchetto.

```
1022    // ******************** MALICIOUS ********************
1023
1024    if(perc_malicious > 0){
1025      srand(time(NULL));
1026      float r = (float) drand48();
1027      if(r < perc_malicious){
1028        if(srh->route_reply() || srh->route_request() || srh->route_error()){
1029          sendOutPacketWithRoute(p,false);
1030          if(DEBUG_MALICIOUS){
1031            cout << "[" << Scheduler::instance().clock() << "] node #" << net_id.dump()
1032          }
1033          return;
1034        }
1035        else{
1036          drop(p.pkt, DROP_RTR_ROUTE_LOOP);
1037          if (DEBUG_MALICIOUS){
1038            cout << "[" << Scheduler::instance().clock() << "] node #" << net_id.dump()
1039          }
1040          return;
1041        }
1042      }
1043    }
1044    if(srh->route_reply() || srh->route_request() || srh->route_error()){
1045      if(DEBUG_MALICIOUS){
1046        cout << "[" << Scheduler::instance().clock() << "] node #" << net_id.dump() << "
1047      }
1048      sendOutPacketWithRoute(p,false);
1049      return;
1050    }
1051    sendOutPacketWithRoute(p, false);
1052
1053    // ******************** END MALICIOUS ********************
```

Se il nodo è malevolo, allora con una certa probabilità scarterà un pacchetto dati, mentre lascerà sempre passare pacchetti di **route_reply**, **route_request** e **route_error**.
Quando un nodo è malevolo, viene generato, attraverso la funzione **drand48()**, un numero random compreso tra 0 e 1. Se questo numero è minore della percentuale di essere malevolo di quel nodo, allora il pacchetto, se è di tipo dati, verrà scartato attraverso la funzione **drop**.
In tutti gli altri casi invece il pacchetto verrà trattato normalmente, passandolo dunque alla funzione **sendOutPacketWithRoute** che si occuperà di fare il corretto forwarding del pacchetto verso il nodo successivo.

Ogni nodo possiede un oggetto **my_monitor** che permette di mettere in cache ogni pacchetto mandato ad ogni altro nodo. Vengono memorizzati solamente i pacchetti dati, ignorando quindi tutti i pacchetti di route_reply, route_request e route_error.
In questo modo il nodo manterrà in cache i pacchetti finché il next hop lo inoltrerà a sua volta.
Questo meccanismo di conferma del forwarding dei pacchetti è possibile grazie alla funzione **tap**. Questa funzione infatti permette ad ogni nodo di ascoltare tutti i pacchetti presenti a livello MAC. In questo modo ogni nodo, analizzando ogni pacchetto che ascolta, può capire se il pacchetto che ha mandato ad un nodo è stato a sua volta forwardato all'hop successivo.

Quando un pacchetto arriva alla funzione **tap**, il nodo controlla dapprima se esso ha un header valido. Poi, attraverso l'uso delle funzioni **get_prev_addr()** e **get_prev_prev_addr()**, aggiunte nel file **hdr_sr.h**, controlla quale nodo ha mandato questo pacchetto e soprattutto da quale nodo lo aveva ricevuto.
Infatti, se quest'ultimo è il nodo stesso, allora vorrà dire che un pacchetto mandato da esso è stato effettivamente inoltrato.
Il pacchetto viene dunque mandato alla funzione **handleTap()** del monitor. Questo cercherà il pacchetto nella cache del nodo, per poterlo cancellare e liberare memoria.

Ogni nodo, oltre a tenere in cache ogni pacchetto dati che manda, tiene anche conto di quanti ne ha mandati. Questo è possibile grazie all'oggetto **my_bank**.
La banca può essere vista come una lista di oggetti, ognuno della quale rappresenta il conto di pacchetti inviati e ricevuti verso un altro nodo della rete.
Quando un pacchetto viene mandato ad un nodo, viene richiamata la funzione **incSendedPackets()**, che incrementa il numero di pacchetti inviati per quello specifico nodo.

Oltre a questo, viene tenuto conto anche di quanti pacchetti sono stati effettivamente inoltrati dai nodi riceventi. Questo è possibile sempre grazie alla funzione tap. Infatti, quando un nodo capisce che il pacchetto che ha ascoltato corrisponde a quello che aveva mandato in precedenza, allora chiama la funzione **incConfirmedPackets()** che incrementa il numero di pacchetti confermati per quel determinato nodo.

Con questo meccanismo è possibile stimare quanti pacchetti vengono inoltrati e quanti ne vengono scartati, riuscendo a determinare quale sia il comportamento degli altri nodi della rete.

A fine simulazione ogni nodo scrive su un apposito file di testo tutta la propria bank.
Per ogni altro nodo della rete, con la quale c'è stato scambio di pacchetti dati, vengono stampati in un file di testo ("**result.txt**") il numero di pacchetti inviati e il relativo numero di pacchetti confermati.
Grazie a queste informazioni, per ogni nodo della rete, è possibile calcolare una stima del suo comportamento, tenendo in considerazione tutte le informazioni contenute nel file result.txt.
Queste stime vengono poi salvate in un altro file di testo ("**valuation.txt**"). Nella riga n-esima sarà dunque presente la media delle stime effettuate da ogni nodo verso l'n-esimo nodo della rete.

Simulazione

Per testare le modifiche effettuate è stato creato un ambiente wireless con le seguenti caratteristiche:

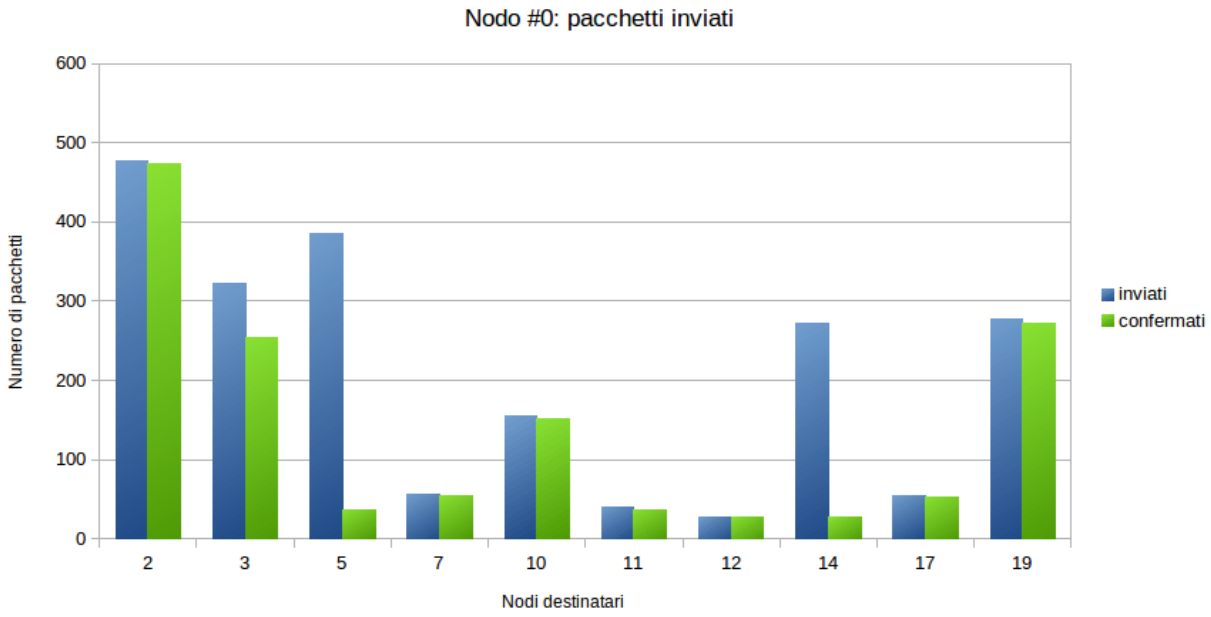
Numero di nodi	20
Dimensione della griglia	1200 m * 1200 m
Movimento dei nodi	Casuale
Tipo di traffico	CBR
Massimo numero di connessioni	10
Dimensione dei pacchetti	512 bytes
Rate di trasmissione	5 pacchetti/secondo
Tempo di simulazione	500 secondi

In questo primo esempio i nodi settati come malevoli sono:

Nodo #5	90%
Nodo #9	90%
Nodo #14	90%
Nodo #18	90%

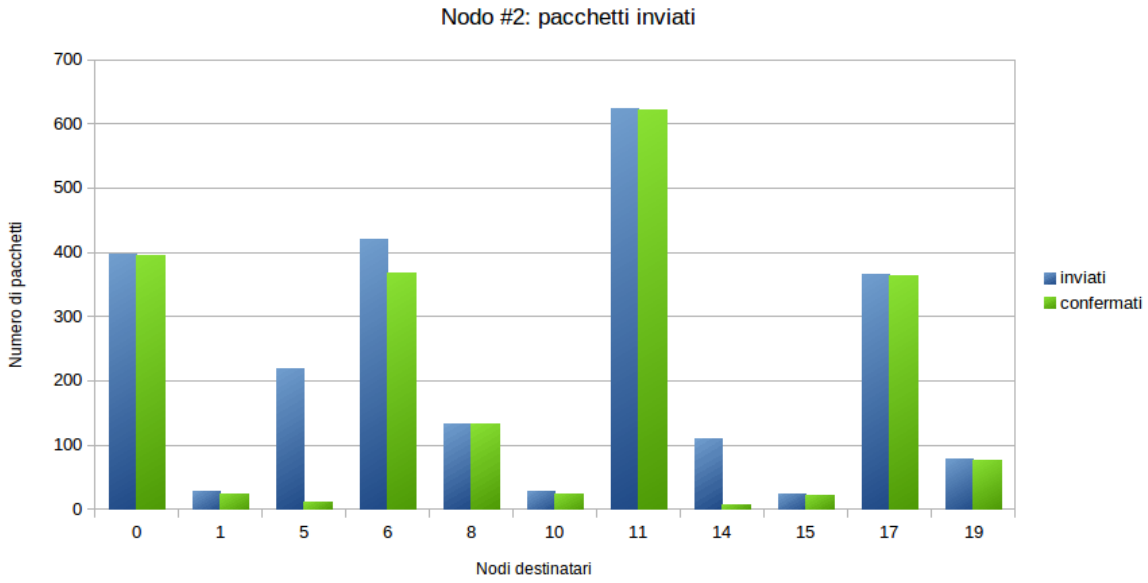
Tutti i restanti nodi hanno invece una percentuale nulla di essere malevoli.

Prendendo in esempio il nodo #0, la sua banca indica i seguenti dati:

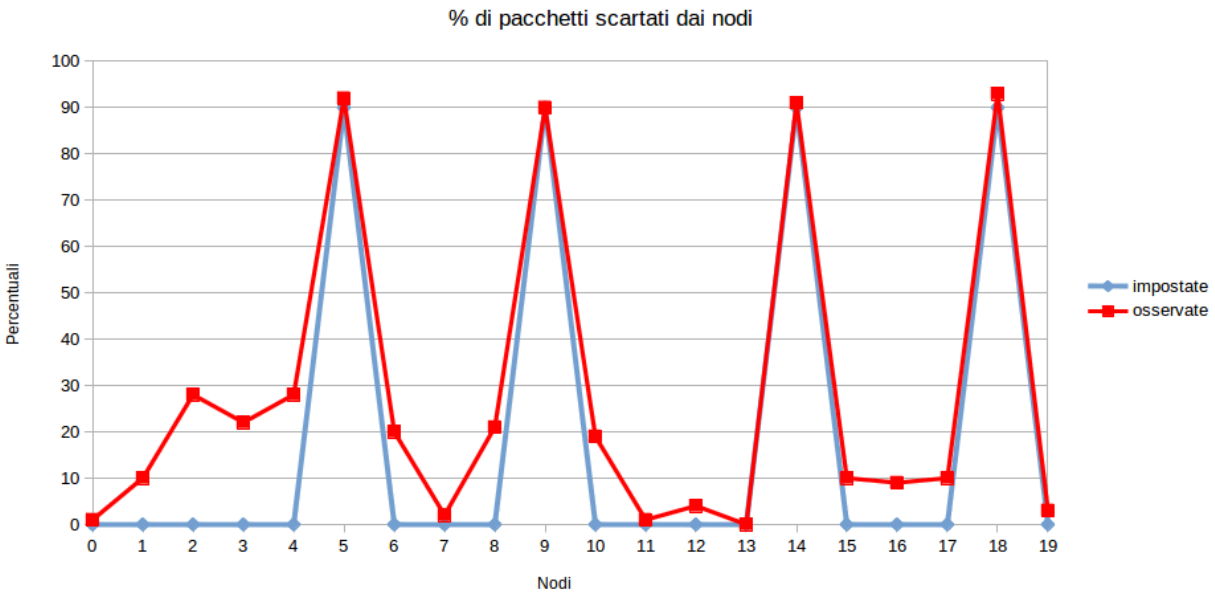


Questo nodo comunica con molti altri nodi della rete.
Da come si può notare i nodi #5 e #14 scartano praticamente circa il 90% dei pacchetti che ricevono da esso e quindi non vengono confermati. Questo in linea con quanto settato in precedenza attraverso le percentuali di essere malevoli.

Prendendo in considerazione un altro nodo, ad esempio il #2, si può osservare come i nodi malevoli #5 e #14 si comportino uguale.

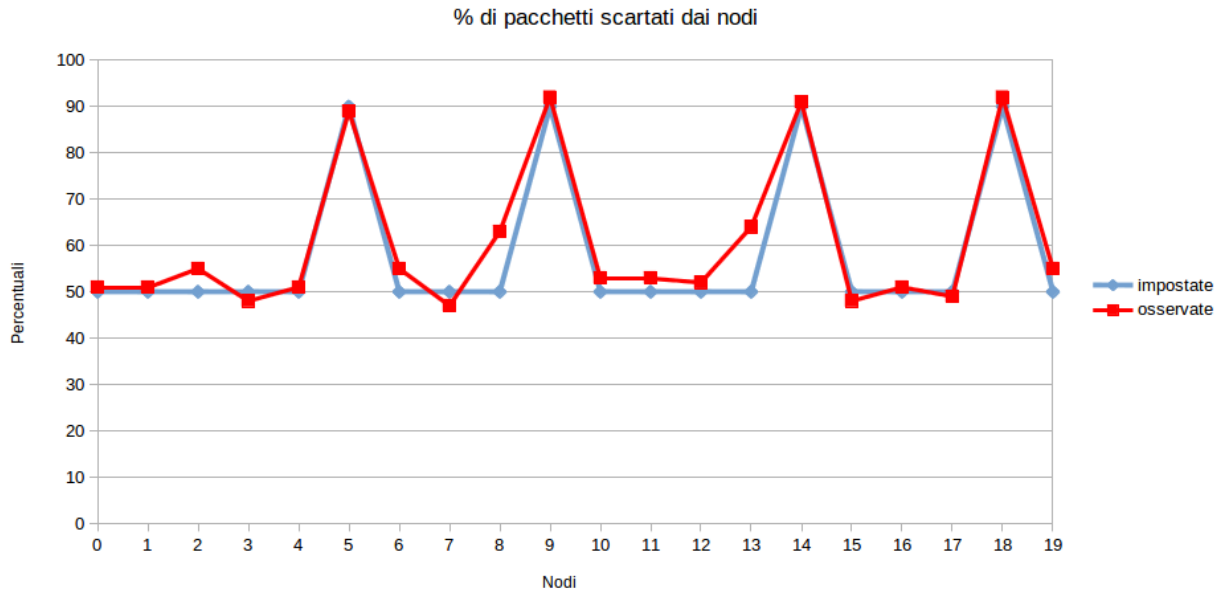


Analizzando queste osservazioni di tutti i nodi, si possono estrarre le stime sulle percentuali di pacchetti scartati dai nodi.
Il grafico che riassume queste statistiche è il seguente:



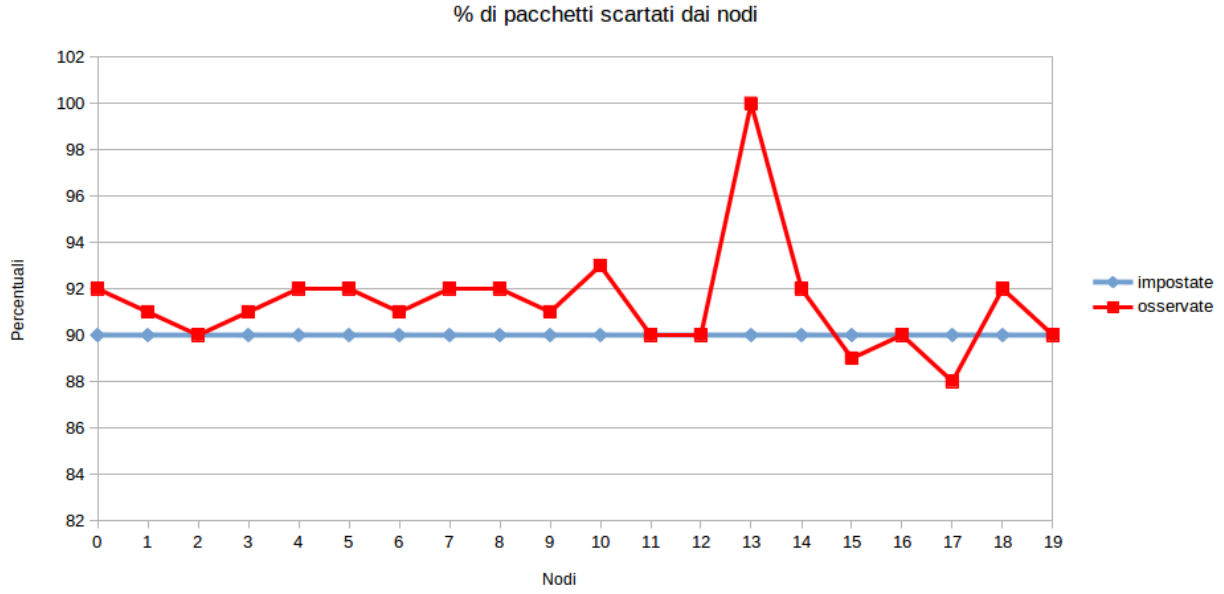
Tutti i nodi malevoli vengono rilevati correttamente.
Ovviamente quasi nessun nodo ha perdite pari a zero perché bisogna considerare anche che ci sono perdite dovute magari alla coda del nodo oppure al canale di comunicazione.

Per dimostrare che i nodi malevoli vengono correttamente rilevati anche in altre condizioni, tutti gli altri nodi sono stati settati malevoli al 50%.
I risultati ottenuti sono i seguenti:



Tutti i nodi quindi si comportano esattamente come previsto.

Infine, tutti i nodi sono stati settati con percentuali di essere malevoli pari al 90%.
I risultati sono i seguenti:

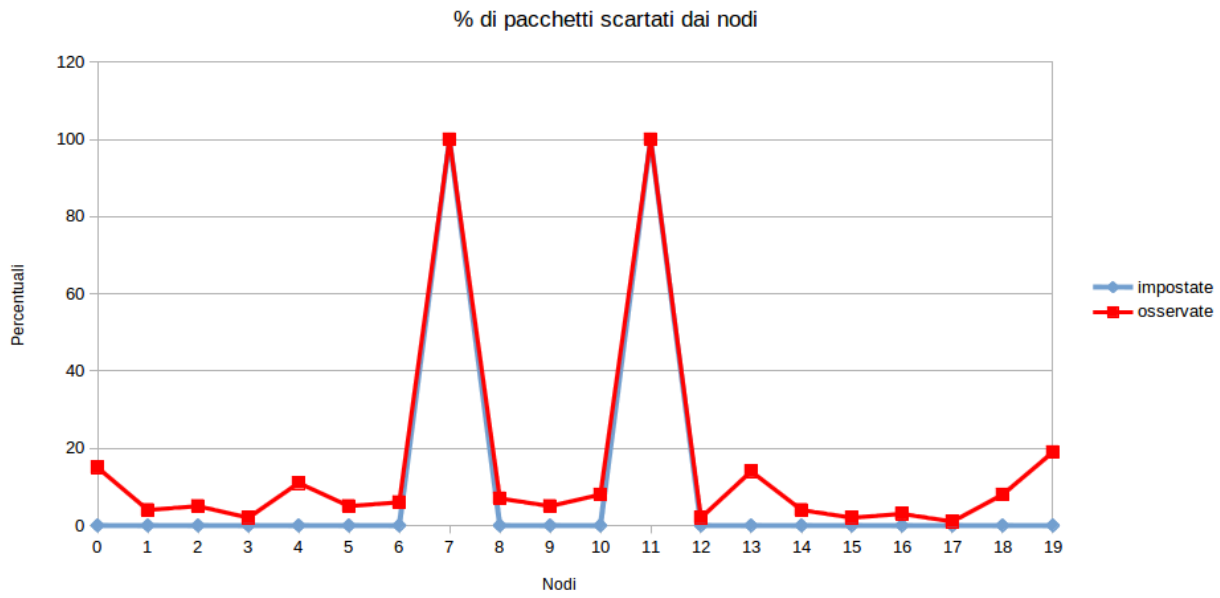


Anche in questo caso tutti i nodi si comportano come previsto ad eccezione del 13 perchè riceve pochissimi pacchetti da parte degli altri nodi, scartandoli tutti.

In questo secondo esempio, è stato utilizzato lo stesso scenario del primo cambiando però la configurazione dei nodi malevoli.
In particolare, solamente due nodi sono stati settati come malevoli al 100%.

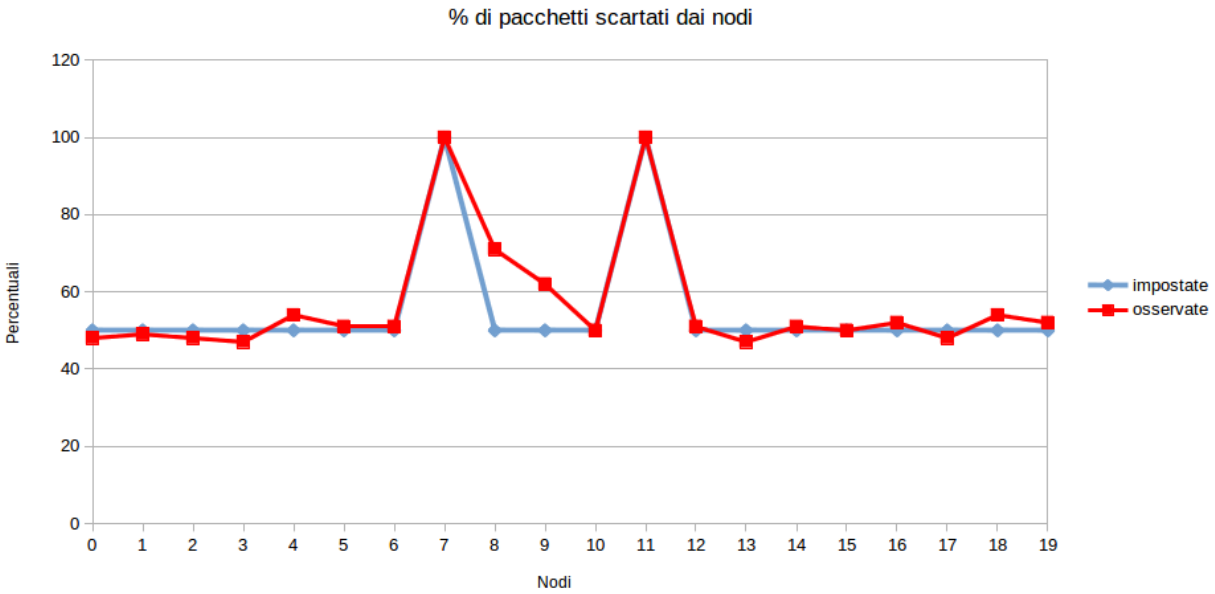
Nodo #7	100%
Nodo #11	100%

I risultati sono i seguenti:



I due nodi malevoli scartato il 100% dei pacchetti che ricevono e vengono rilevati correttttamente dagli altri nodi della rete.

Modificando le percentuali degli altri nodi al 50%, il risultato non cambia e anche in questo caso i due nodi malevoli al 100% vengono rilevati con precisione.



Conclusioni

Le modifiche effettuate al protocollo DSR hanno permesso di introdurre nella rete nodi con comportamento malevolo. Questi nodi rappresentano un danno per la rete e vengono sempre ben identificati dagli altri nodi.

Un possibile sviluppo futuro potrebbe essere quello di rendere i nodi più intelligenti, facendogli evitare quelli riconosciuti in precedenza come malevoli e incentivando dunque a non assumere un comportamento di questo tipo, poichè dannoso per la rete.