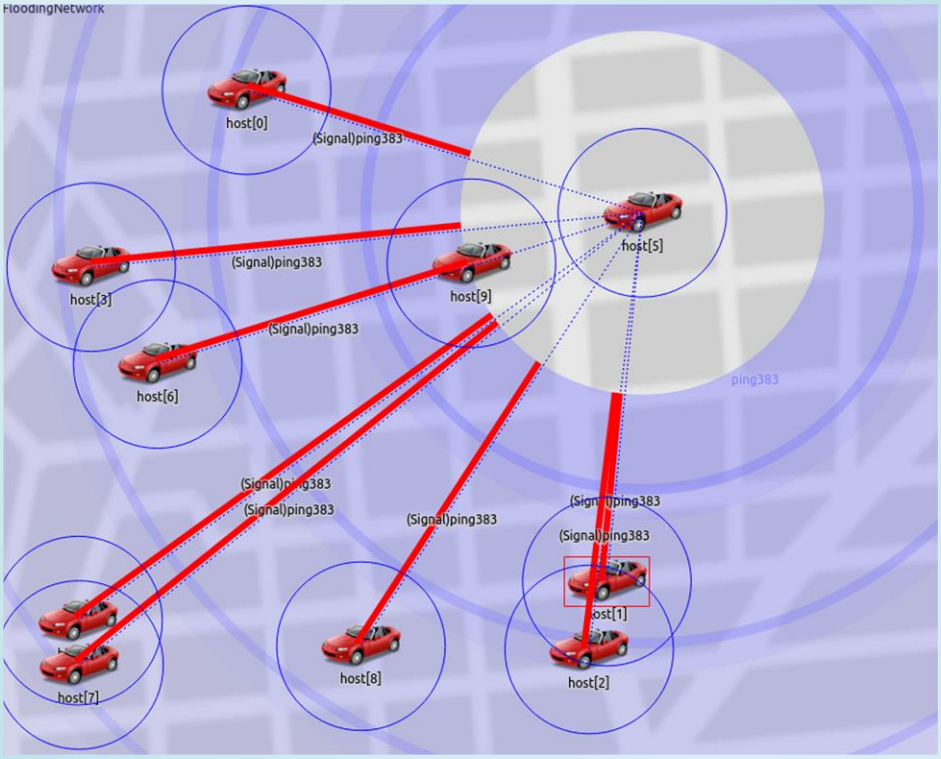
RELAZIONE

ELABORATO B: SIMULAZIONE DI UN ALGORITMO DI FLOODING SU UNA RETE WIRELESS



***Docenti**: Prof. Lucia Lo Bello*

Prof. Gaetano Patti

REALIZZAZIONE A CURA DI:

Arciprete Andrea 1000009556

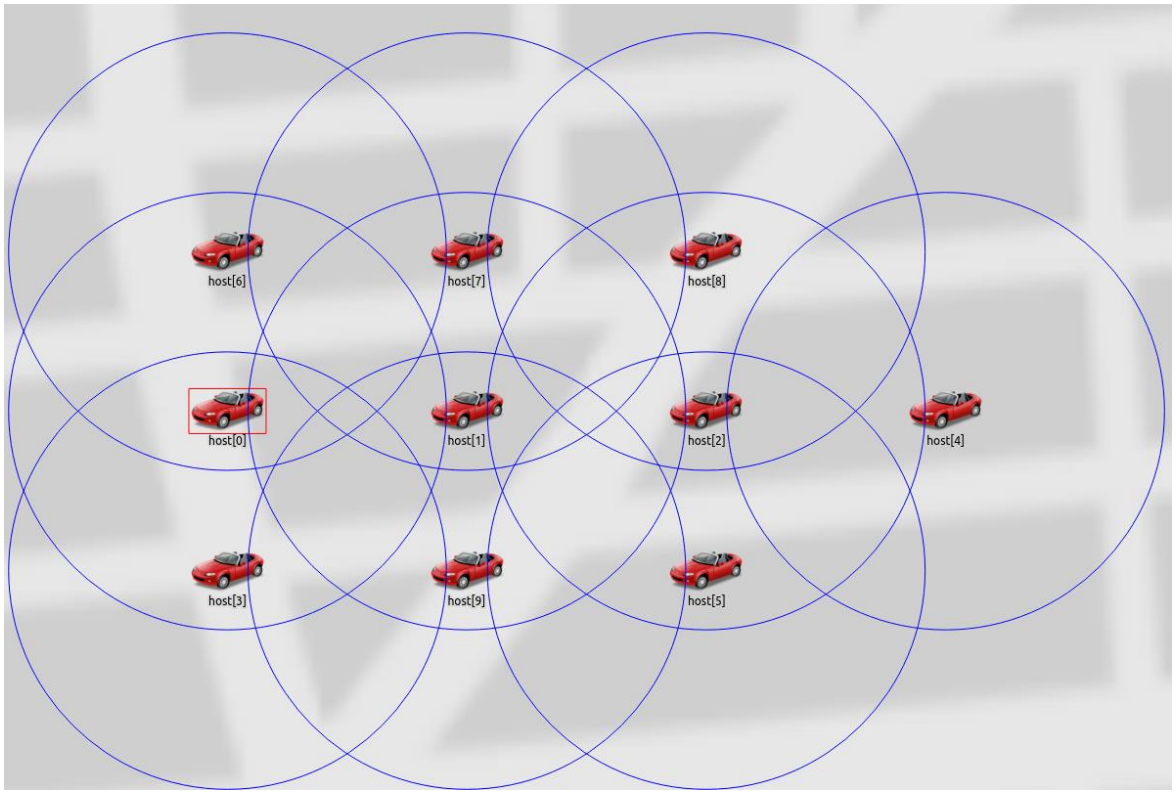
Chiavetta Daniela 1000009554

***Anno Accademico**:2019/2020*

INGEGNERIA INFORMATICA

1. INTRODUZIONE

Si vuole simulare un algoritmo di Flooding su una rete wireless in cui dei nodi mobili inviano periodicamente dei messaggi ad un nodo ricevitore.



Al fine di soddisfare la specifica inerente la realizzazione di un algoritmo di Flooding su una rete wireless abbiamo utilizzato il framework INET, una libreria open-source di modelli e protocolli di rete per la piattaforma di simulazione Omnet++.

Per la creazione del progetto potevamo seguire due diverse strade:

- Creazione del progetto all'interno della directory **examples** di INET.
- Creazione del progetto e successivo collegamento con la libreria INET.

Noi abbiamo scelto di seguire la seconda strada, dunque abbiamo creato il nostro progetto e abbiamo incluso la libreria INET al suo interno.

Nelle simulazioni abbiamo supposto che tutti i nodi, eccetto l'**host[9]**, si comportino da trasmettitori, mentre l'**host[9]** si comporti da nodo ricevitore. Tuttavia, è possibile impostare opportunamente i parametri nel file di configurazione **omnetpp.ini** al fine di settare un diverso nodo ricevitore o più nodi ricevitori. Ciascun nodo, infatti, presenta due diversi livelli applicativi, uno di tipo **PingApp,** utilizzato per generare messaggi, e un altro di tipo **UdpSink,** utilizzato per ricevere eventuali messaggi a lui destinati.

Il funzionamento dell'algoritmo di Flooding che abbiamo implementato è il seguente: ogni singolo nodo trasmettitore invia un pacchetto, indirizzato ad un determinato nodo ricevitore, agli altri nodi che rientrano nel range di trasmissione (copertura).

Tale pacchetto a livello network, che incapsula al suo interno il messaggio proveniente dal livello applicativo (PingApp), presenta i seguenti campi:

- Indirizzo sorgente di 2 byte.
- Indirizzo di destinazione di 2 byte.
- TTL (1 byte) che parte da un valore parametrizzato (settabile dal file **omnetpp.ini**) e decresce ad ogni hop. Se il TTL vale zero il pacchetto avente tale TTL deve essere scartato.
- Numero di sequenza (3 byte): è un numero sequenziale che viene incrementato da un nodo ogni volta che fa una trasmissione.

Trasmesso il pacchetto ai nodi che rientrano nel range di trasmissione del nodo sorgente, questo pacchetto può essere ricevuto da un forwarder node (nodo che rientra nel range di trasmissione del nodo sorgente ma non è il destinatario del pacchetto) o dal nodo destinatario:

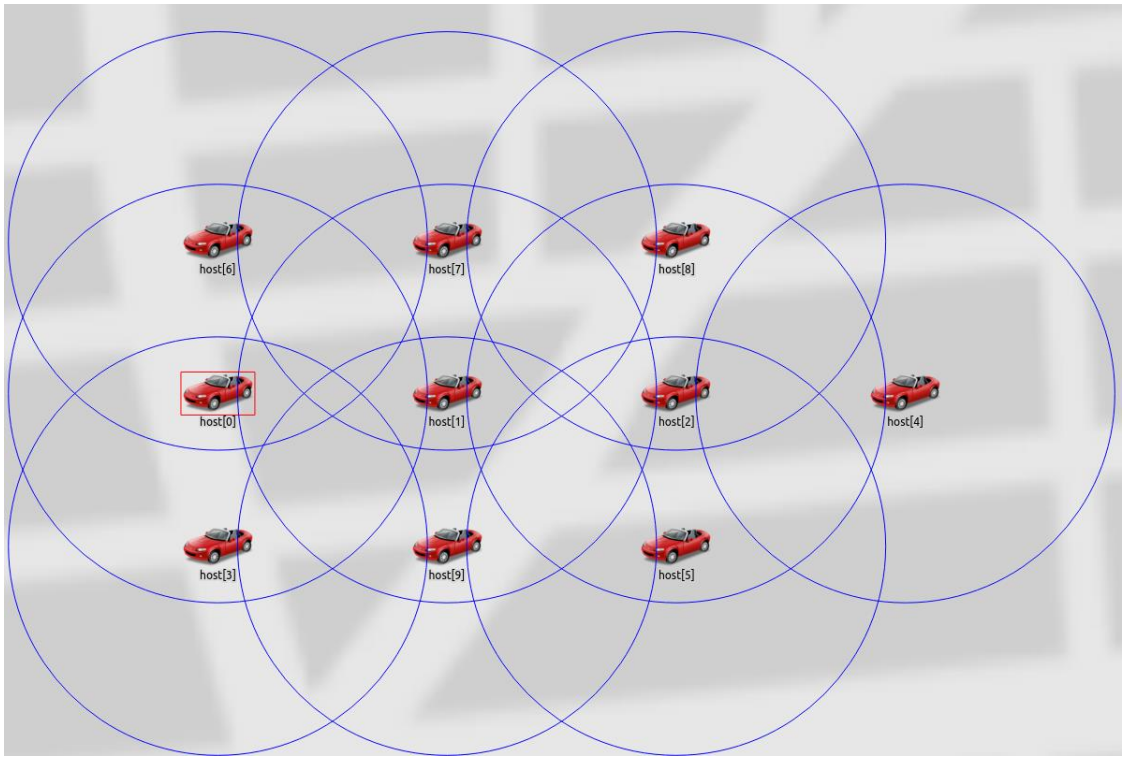
- Se è un forwarder node e il pacchetto ricevuto non lo ha mai ricevuto prima d'ora allora, a livello di rete, deve decrementare il TTL (se questo non vale zero) e forwardare il pacchetto a tutti i nodi che rientrano nel suo range di trasmissione. Ovviamente, se il pacchetto ricevuto presenta il TTL nullo allora esso viene scartato.
- Se è un forwarder node e ha ricevuto, in passato, lo stesso pacchetto allora, a livello di rete, il pacchetto deve essere scartato, a prescindere dal valore del TTL.
- Se è un nodo destinatario allora il primo pacchetto ricevuto avente TTL maggiore di zero viene decapsulato e il messaggio da esso estratto viene passato al livello applicativo, mentre tutti i successivi pacchetti aventi la stessa coppia (indirizzo sorgente, numero di sequenza) vengono scartati.

Inoltre, al fine di rendere la simulazione quanto più vicina al mondo reale abbiamo utilizzato come modello di canale il **LogNormalShadowing** con alpha = 4.03 e sigma = 4.98.
Ogni simulazione che si rispetti deve essere corredata da statistiche al fine di trarre da essa dei risultati. A tale scopo abbiamo valutato le seguenti metriche:

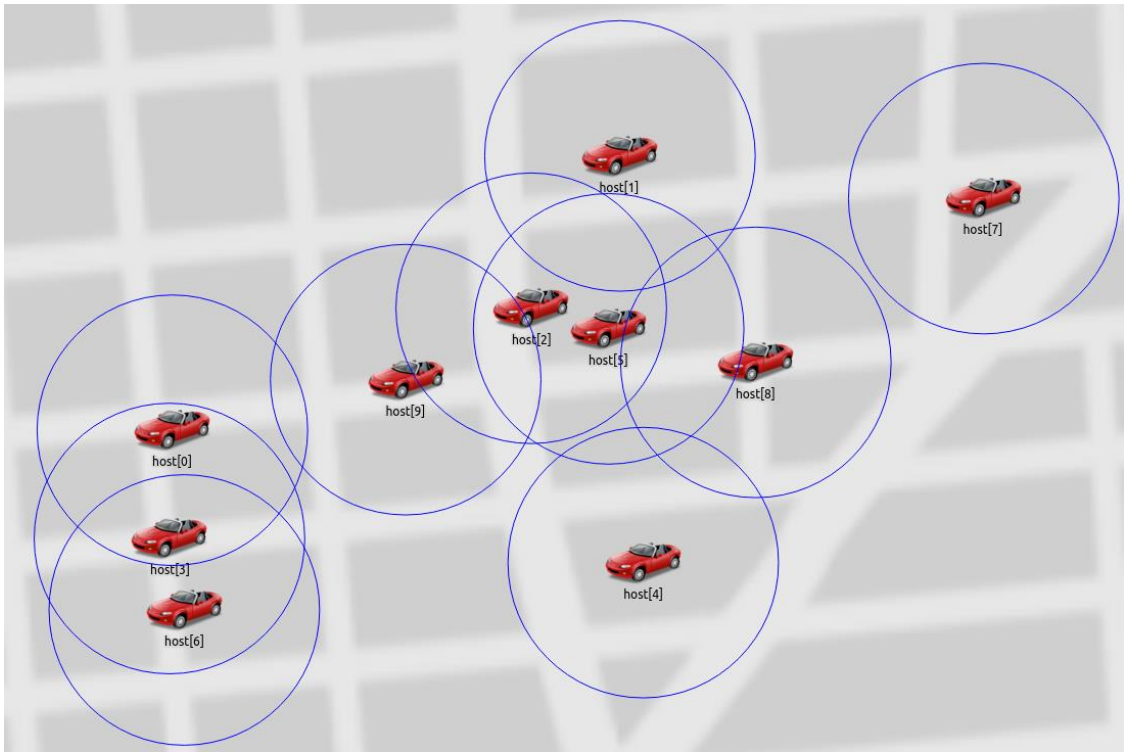
- Numero medio di hops necessari a trasmettere un messaggio tra sorgente e destinazione.
- Numero massimo di pacchetti nella coda di trasmissione del MAC di ogni nodo.

2. SCENARIO

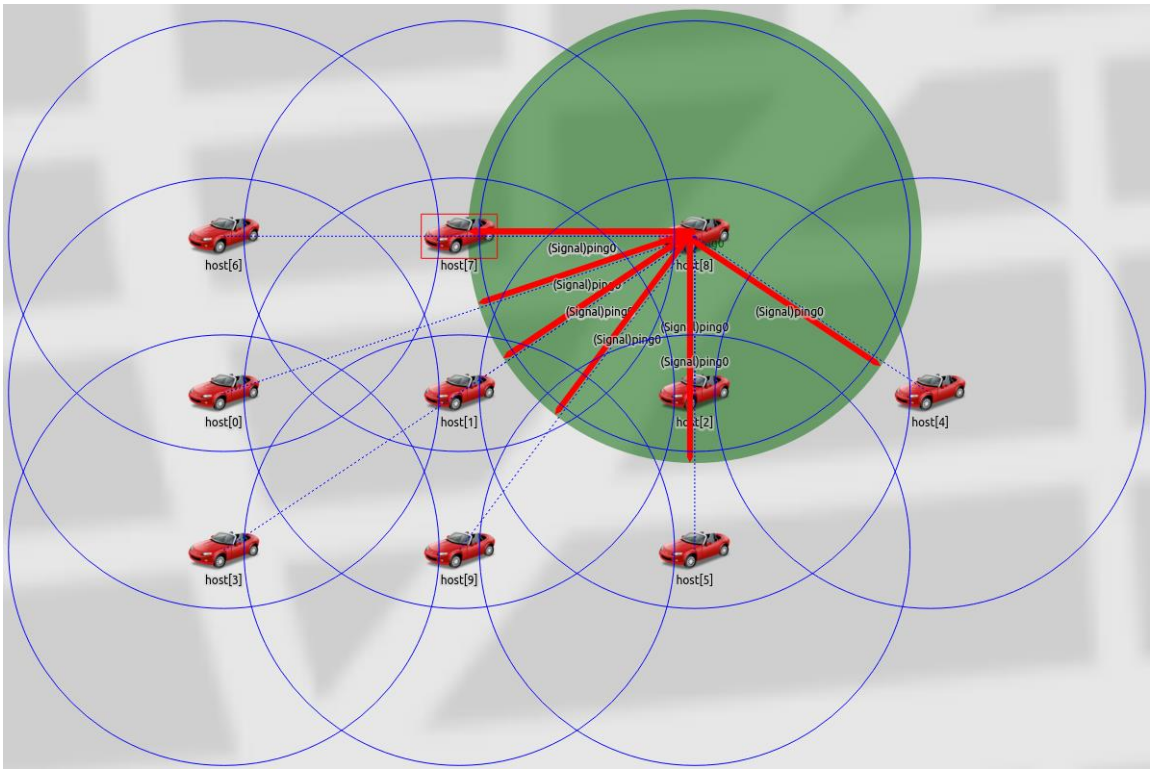
Per tutte le simulazioni effettuate abbiamo settato un unico nodo ricevitore (**host[9]**) e tutti i restanti nodi trasmettitori.
Inizialmente, al fine di testare il funzionamento corretto dell'algoritmo di Flooding abbiamo deciso di simulare una rete wireless costituita da 10 nodi stazionari.



Successivamente, abbiamo aggiunto la mobilità al fine di valutare le statistiche in presenza di 10 nodi mobili.



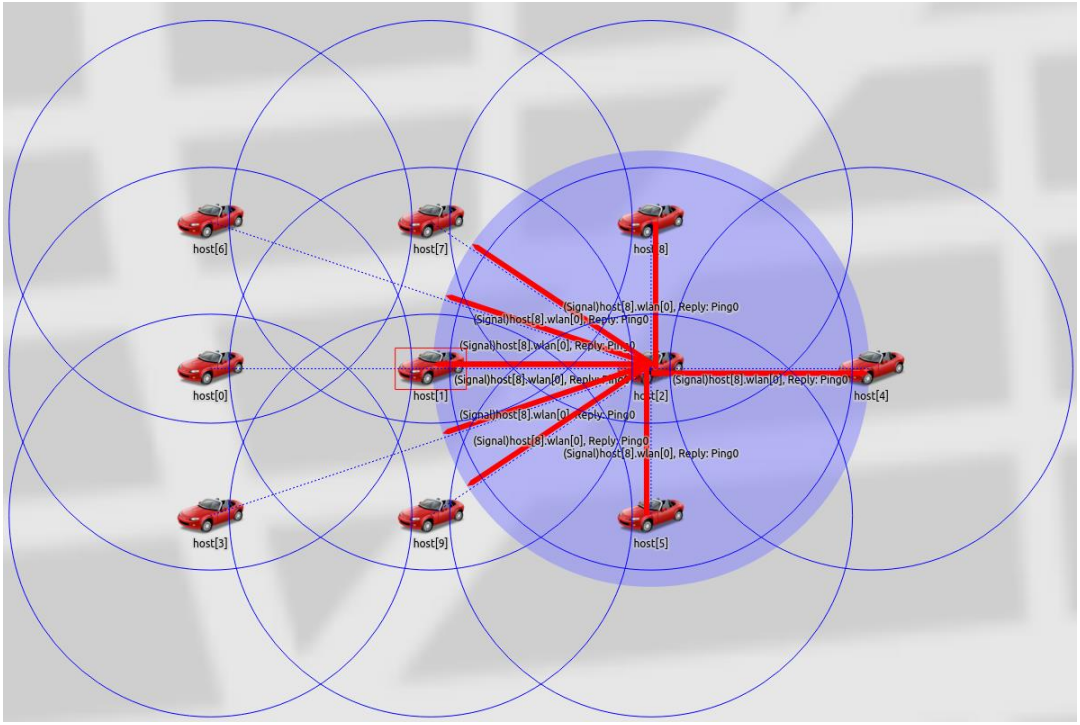
Nelle figure seguenti sono mostrate due istantanee della simulazione della rete wireless contenente 10 nodi mobili.



In questa figura si nota come il messaggio (Ping) inviato da un nodo sorgente (**host[8]**) viene trasmesso a tutti i nodi che rientrano nel range di trasmissione (copertura) del nodo stesso, range di trasmissione che è rappresentato dalla circonferenza di colore azzurro che avvolge il nodo sorgente. Ciò significa che i nodi che giacciono al di fuori di questa circonferenza scarteranno il pacchetto ricevuto.

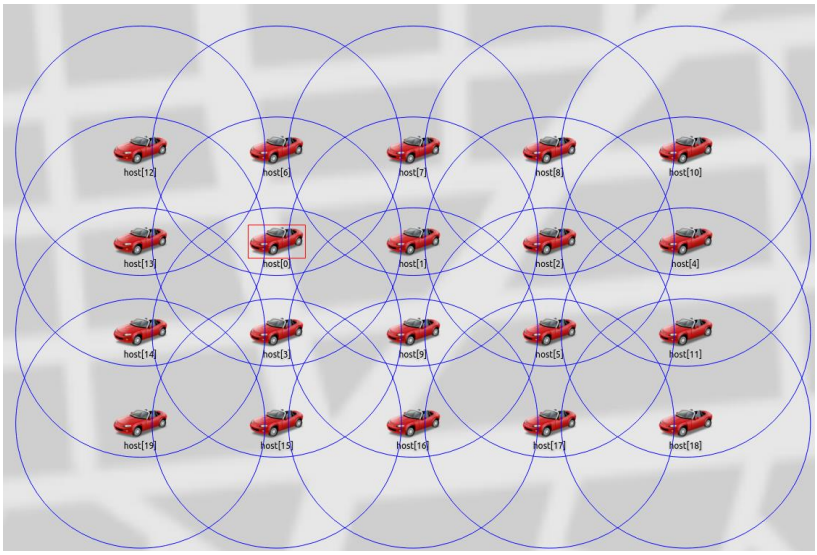
Come detto nell'introduzione, non appena un nodo riceve per la prima volta un pacchetto avente una determinata coppia (indirizzo sorgente, numero di sequenza) che non è a lui destinato e avente TTL maggiore di 0 allora ritrasmette il pacchetto ai nodi che rientrano nel suo range di trasmissione.

La figura successiva mostra proprio questo comportamento.

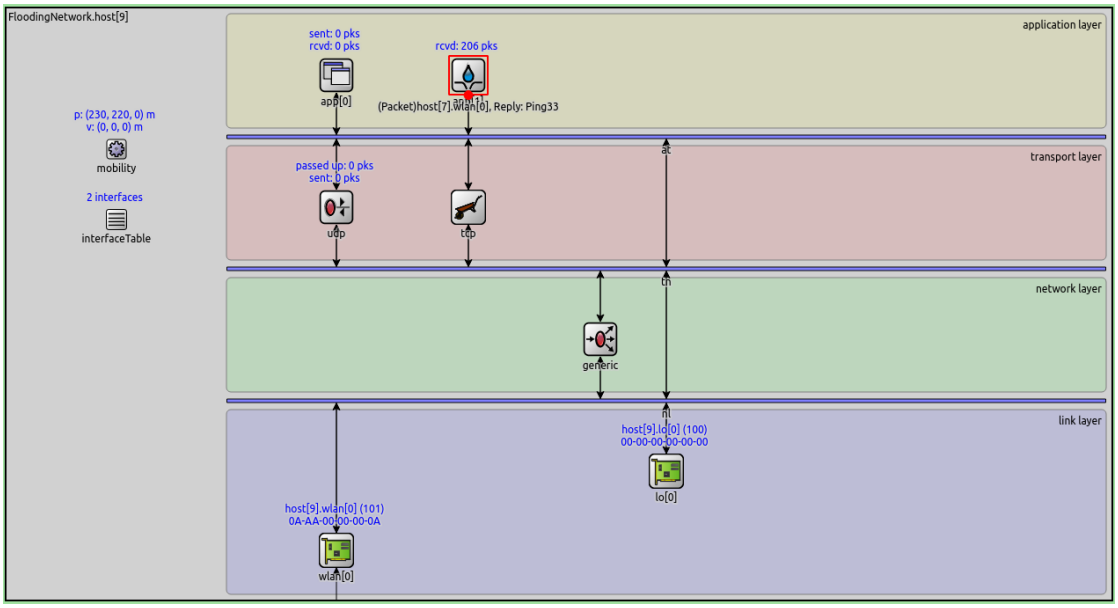


Come si nota dalla figura sovrastante l'**host[2]** ha ricevuto il pacchetto inviato dall'**host[8]** e, notando che non è a lui destinato (in quanto come detto nell'introduzione tutti i pacchetti sono destinati all'**host[9]**), che non lo ha già ricevuto e forwardato in precedenza, che il TTL è maggiore di zero, lo trasmette ai nodi che rientrano nel suo range di trasmissione (copertura). Questa procedura andrà avanti fintanto che o l'**host[9]** riceve il pacchetto o il TLL diventa pari a zero, o il nodo sorgente/forwarder trasmette il pacchetto ma nel suo range di trasmissione non è presente alcun nodo oppure il pacchetto si perde a causa del fatto che nella nostra simulazione il canale è abbastanza rumoroso (stiamo utilizzando il **LogNormalShadowing** come modello di canale).

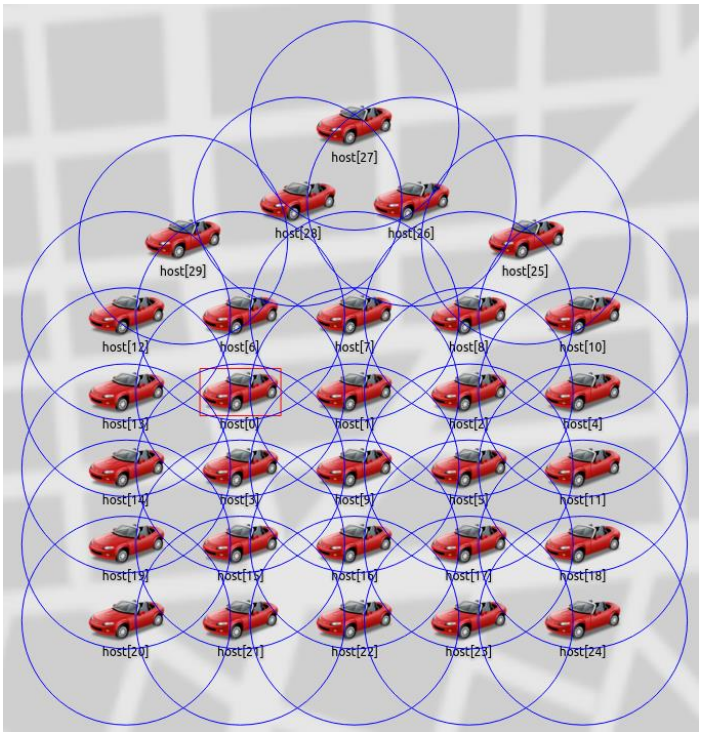
In un secondo momento abbiamo incrementato il numero di nodi passando, prima da 10 a 20, poi da 20 a 30 e infine da 30 a 40 nodi.
Nella figura seguente è mostrata un'istantanea della simulazione della rete wireless contenente 20 nodi mobili.



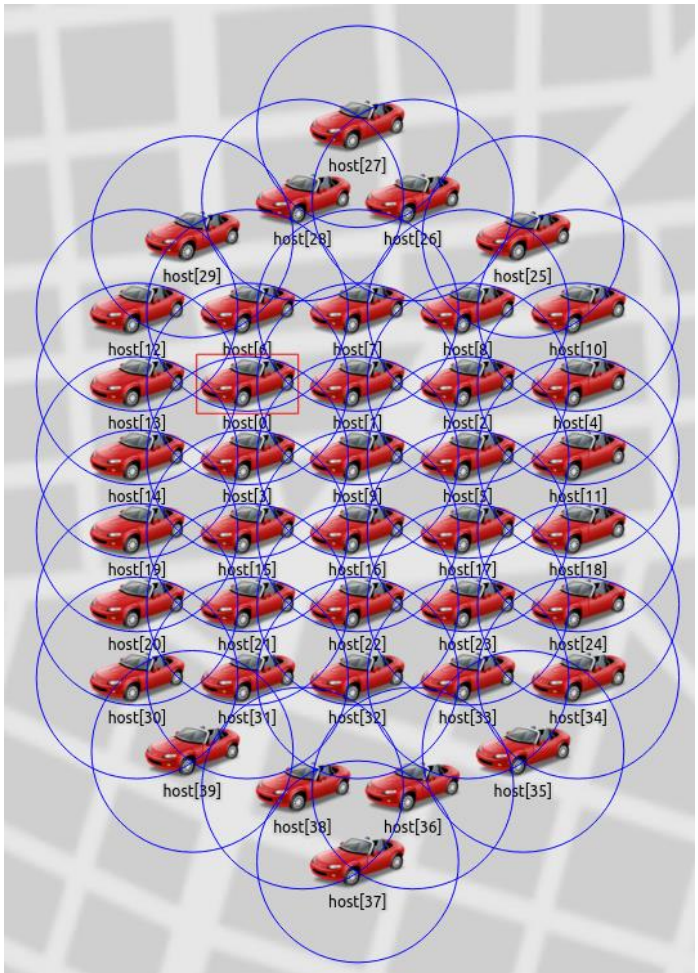
La figura sottostante mostra come il messaggio venga consegnato al livello applicativo del nodo destinatario che come detto è l'**host[9]**.



Nella figura seguente è mostrata un'istantanea della simulazione della rete wireless contenente 30 nodi mobili.

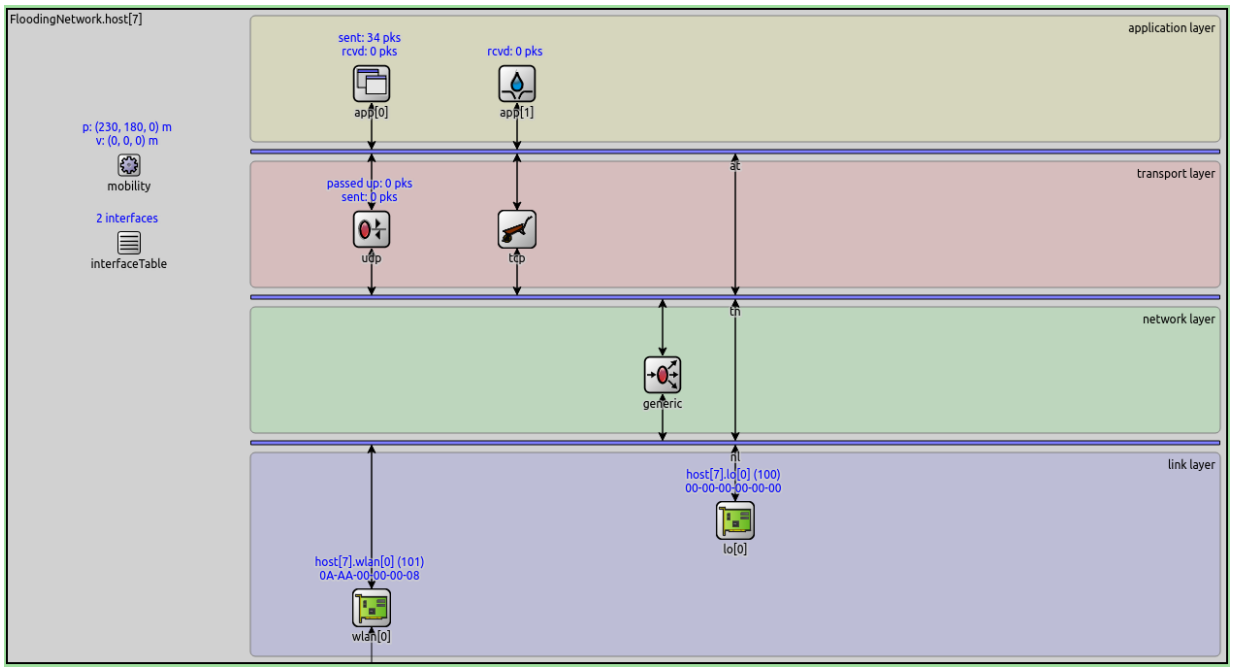


Nella figura seguente è mostrata un'istantanea della simulazione della rete wireless contenente 40 nodi mobili.



3. SCELTE IMPLEMENTATIVE

Per ciascun nodo abbiamo deciso di implementare due livelli applicativi diversi: un livello applicativo (PingApp) usato per generare messaggi e un livello applicativo (UdpSink) utilizzato per ricevere messaggi.

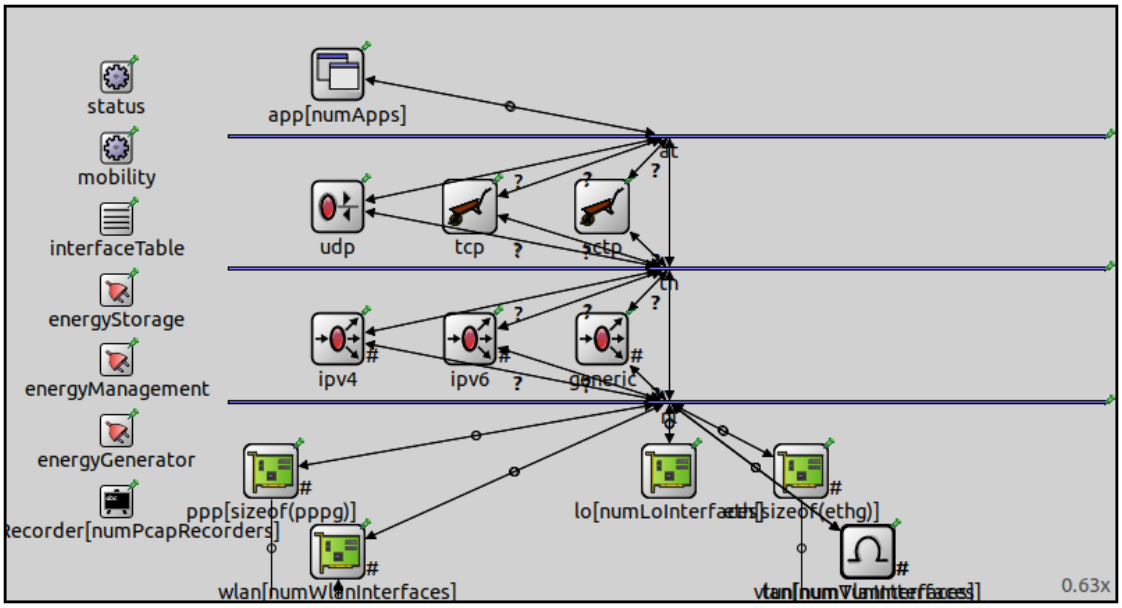


Grazie a questa scelta implementativa **siamo riusciti a trasmettere correttamente ciascun messaggio sino al livello applicativo del nodo destinatario**. Ciò significa che non appena un pacchetto arriva al livello di rete del nodo destinatario (host[9]), da esso viene estratto il messaggio che viene mandato al livello applicativo del nodo destinatario stesso.

Oltre all'aggiunta di un ulteriore livello applicativo (UdpSink) siamo stati costretti a modificare anche il livello di trasporto ed in particolare il simple module **udp**.

Per riuscire a modificare il simple module **udp**, dal momento che abbiamo utilizzato INET e dunque molti dei moduli messi a disposizione dalla libreria, siamo stati costretti a creare un nuovo simple module denominato **udp** che estende il modulo **udp** messo a disposizione da INET in modo da modificare le funzionalità in esso presenti adattandole alle nostre esigenze. A tal scopo abbiamo fatto l'override di alcuni dei metodi presenti all'interno del modulo **udp** messo a disposizione da INET.

Per poter utilizzare il nostro modulo **udp** che come detto estende il modulo **udp** messo a disposizione dalla libreria INET siamo stati costretti a creare un modulo **AdHocHostFlooding**, che, in pratica, è totalmente identico al modulo **AdhocHost** presente nella libreria INET, eccezion fatta per il simple module **udp** da noi implementato.



Come si nota da questa figura siamo riusciti con successo ad utilizzare, all'interno del compound module **AdHocHostFlooding**, il simple module **udp** da noi implementato al posto del simple module **udp** presente nella libreria INET.

Inoltre, per l'implementazione dell'algoritmo di Flooding abbiamo preso spunto dal modulo **Flooding** della libreria INET e abbiamo creato un nuovo simple module denominato **Flood,** che estende il modulo **Flooding** della libreria INET. Ancora una volta, per adattare le funzionalità presenti all'interno del modulo **Flooding,** all'interno del simple module **Flood** abbiamo fatto l'override di alcuni dei metodi presenti nel modulo **Flooding**.

Vogliamo precisare, inoltre, di aver scelto **LinearMobility** come mobilità e di aver effettuato le simulazioni impostando un TTL pari a 5 dal momento che, eseguendo le varie simulazioni, ci siamo resi conto, come testimoniano le statistiche a seguire, che il numero medio di hops si mantiene inferiore a 3. Per tale motivo, nel file **omnetpp.ini**, abbiamo deciso di effettuare le simulazioni con un valore basso di TTL.

4. STATISTICHE

Le statistiche raccolte sono le seguenti:

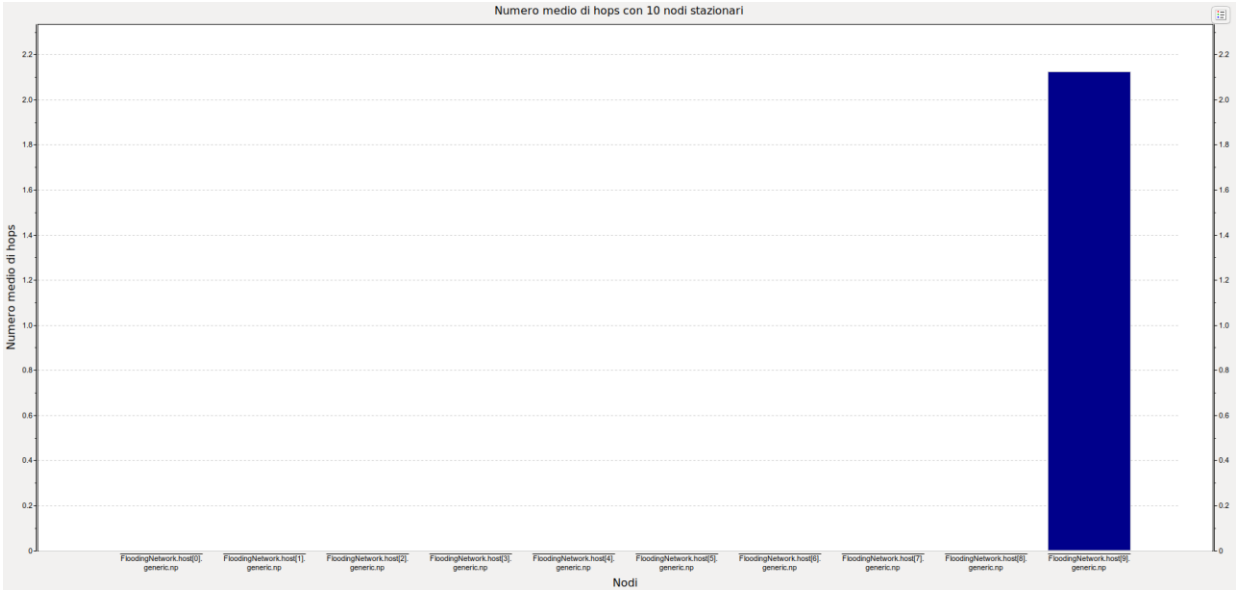
- Numero medio di hops necessari a trasmettere un messaggio tra sorgente e destinazione.
- Numero massimo di pacchetti nella coda di trasmissione del MAC di ogni nodo.

5. RISULTATI

Al fine di valutare le prestazioni della rete abbiamo considerato diverse configurazioni, che differiscono per il numero di nodi. Inoltre, abbiamo valutato cosa succede riducendo il **Send Interval** con cui l'applicazione genera messaggi.

a) Configurazione con 10 nodi stazionari e Send Interval pari a 1s

- *NUMERO MEDIO DI HOPS*

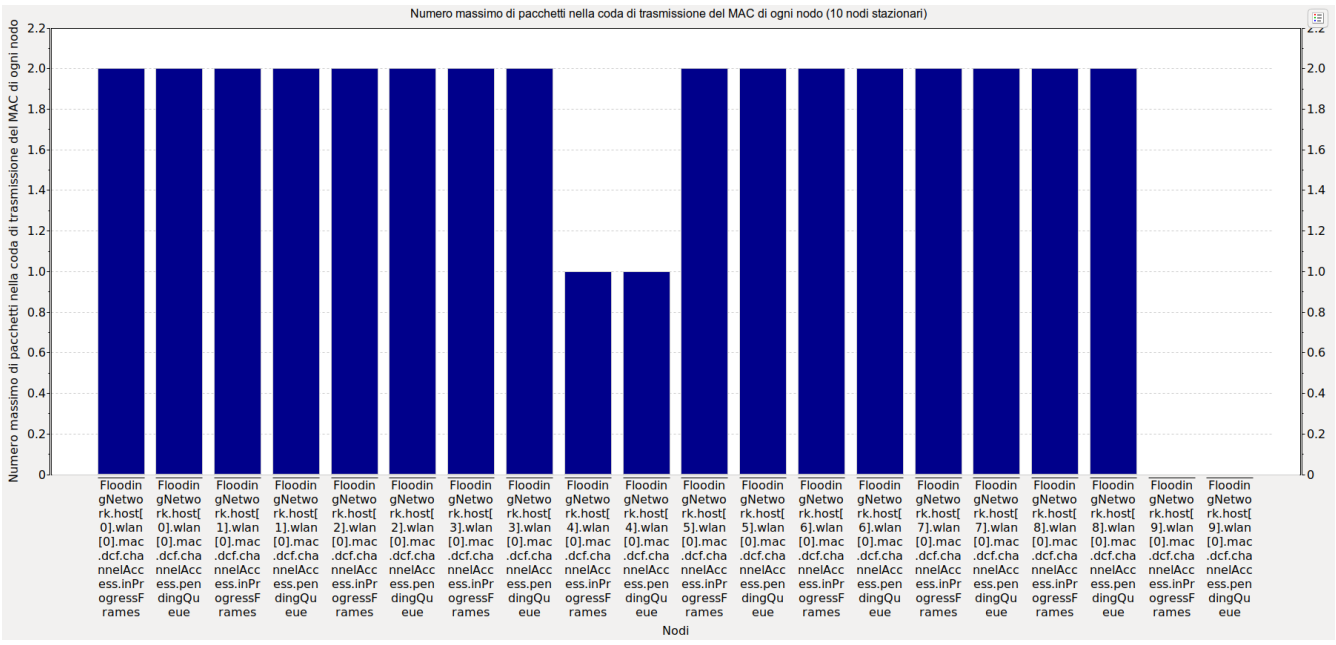


Questo grafico indica che in media è necessario un numero di hops pari a circa 2.1 affinché il messaggio giunga al nodo destinatario, ossia all'**host[9]**.

Tale valore ce lo aspettavamo in quanto per come abbiamo disposto i vari nodi l'host[9] è raggiungibile mediante 1 hop dai nodi trasmettitori più vicini e mediante 2-3 hops dai nodi più lontani. Dunque, ha senso che il numero medio di hops sia pari a circa 2.1 per come abbiamo disposto i nodi nella configurazione con 10 nodi stazionari.

A causa del fatto che abbiamo considerato uno scenario in cui l'unico nodo destinatario è l'**host[9]** è ovvio che nella figura il numero medio di hops è diverso da zero soltanto in corrispondenza dell'**host[9]**.

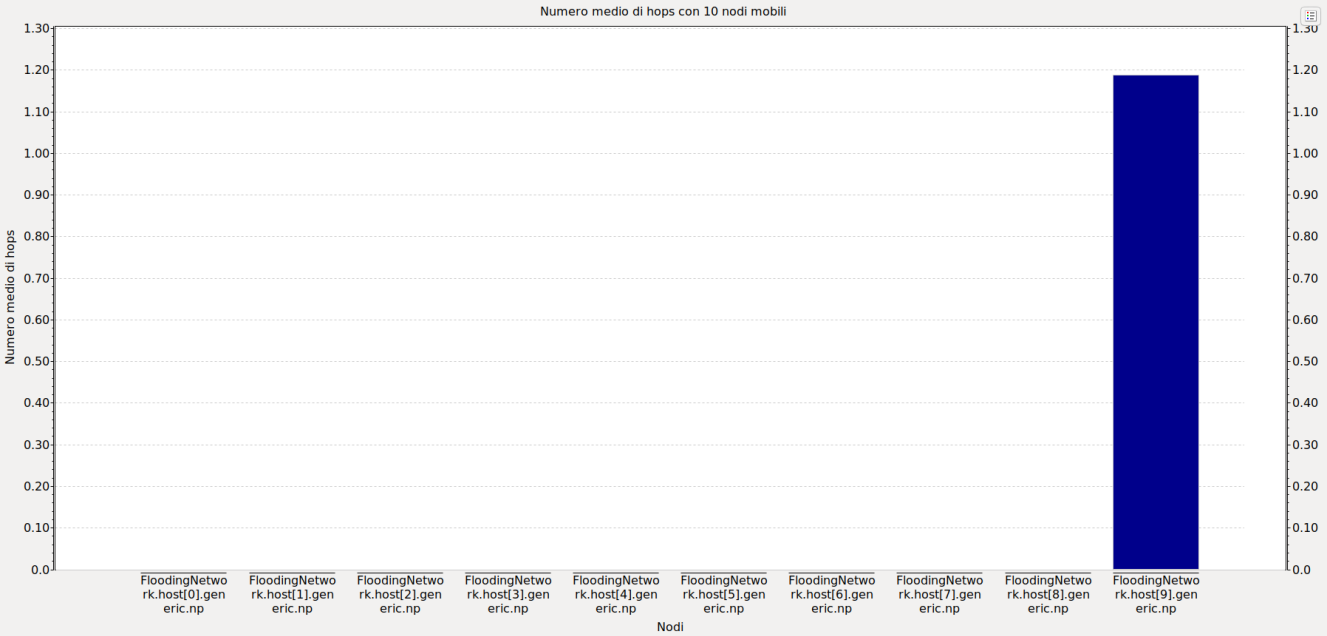
- *NUMERO MASSIMO DI PACCHETTI NELLA CODA DI TRASMISSIONE DEL MAC DI OGNI NODO*



Questo grafico indica il numero massimo di pacchetti accodati nella coda di trasmissione del livello MAC di ciascun nodo. A causa del fatto che abbiamo scelto di considerare lo scenario in cui tutti i nodi sono trasmettitori, eccetto l'**host[9]** che funge soltanto da ricevitore, nel grafico e in tutti quelli a seguire si vedrà come in corrispondenza dell'host[9] il numero massimo di pacchetti accodati nella coda del livello MAC di questo nodo sia nullo in quanto, come detto, tale nodo non trasmette.
Dal momento che abbiamo un numero di nodi abbastanza basso e l'applicazione genera messaggi con un Send Interval di 1s non accade mai (come ci aspettavamo) che la coda va in saturazione infatti il numero massimo di pacchetti accodati è pari a 2, mentre la dimensione massima della coda di trasmissione è pari a 100. Siamo, dunque, lontanissimi dal raggiungimento di questo valore massimo e dunque dalla saturazione.

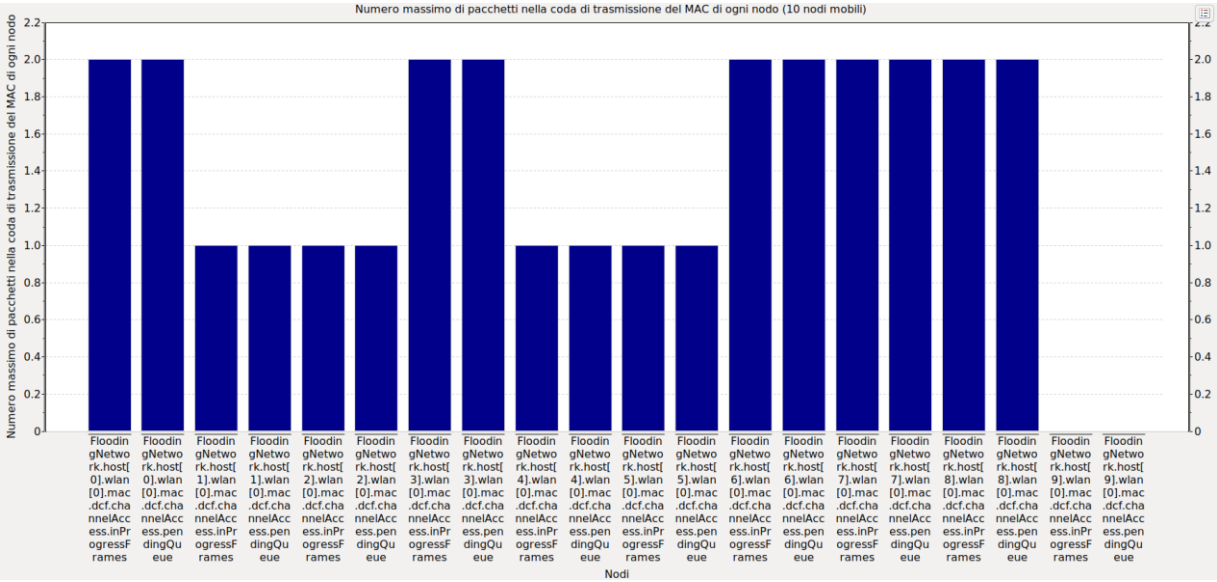
b) Configurazione con 10 nodi mobili e Send Interval pari a 1s

- *NUMERO MEDIO DI HOPS*



Prendendo in considerazione una configurazione di 10 nodi mobili abbiamo notato che il numero medio di hops necessari affinché il messaggio giunga al nodo destinatario, ossia all'host[9], si è leggermente ridotto passando da 2.1 a circa 1.19.

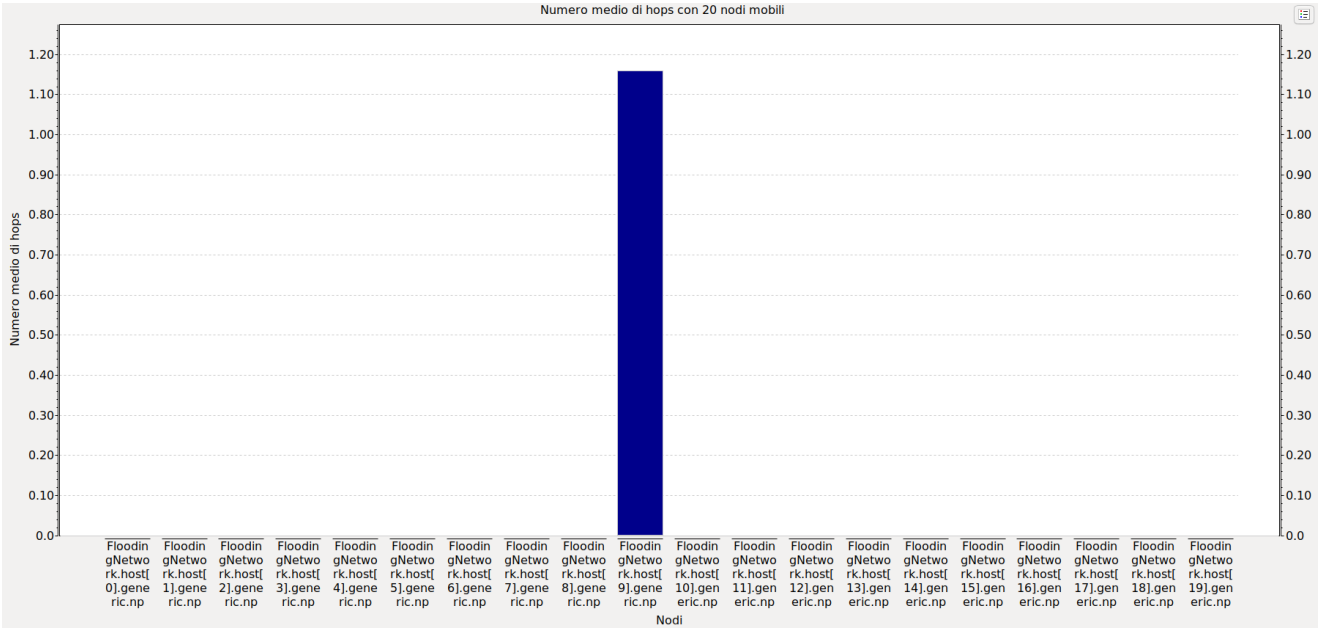
- *NUMERO MASSIMO DI PACCHETTI NELLA CODA DI TRASMISSIONE DEL MAC DI OGNI NODO*



Pur avendo aggiunto la mobilità nessuna delle code presenti nel livello MAC di ciascun nodo va in saturazione, anzi l'aggiunta della mobilità ha avuto un effetto benefico dal punto di vista del numero di pacchetti accodati infatti, rispetto al caso precedente, vi è un maggior numero di nodi che ha un solo pacchetto accodato nella coda di trasmissione del MAC.

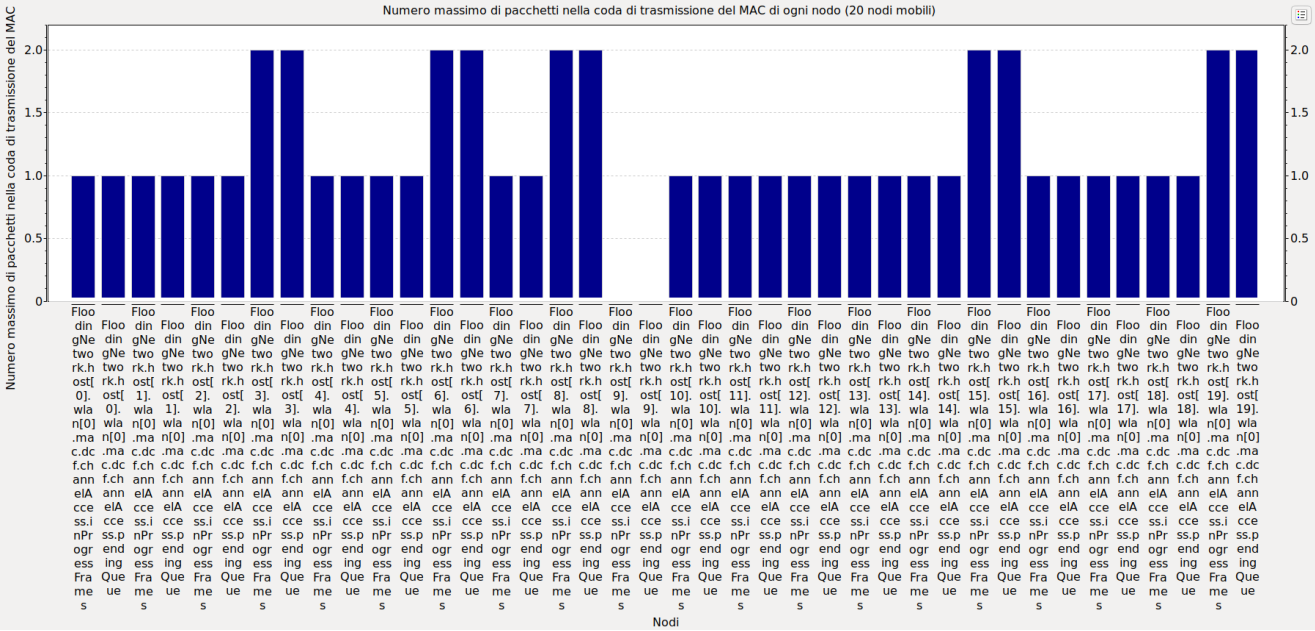
c) Configurazione con 20 nodi mobili e Send Interval pari a 1s

- *NUMERO MEDIO DI HOPS*



Abbiamo deciso di aumentare il numero di nodi da 10 a 20 per valutare l'impatto che quest'aggiunta comporta. Come si vede dal grafico rispetto al caso precedente il numero medio di hops si è ulteriormente abbassato raggiungendo il valore medio di circa 1.17.

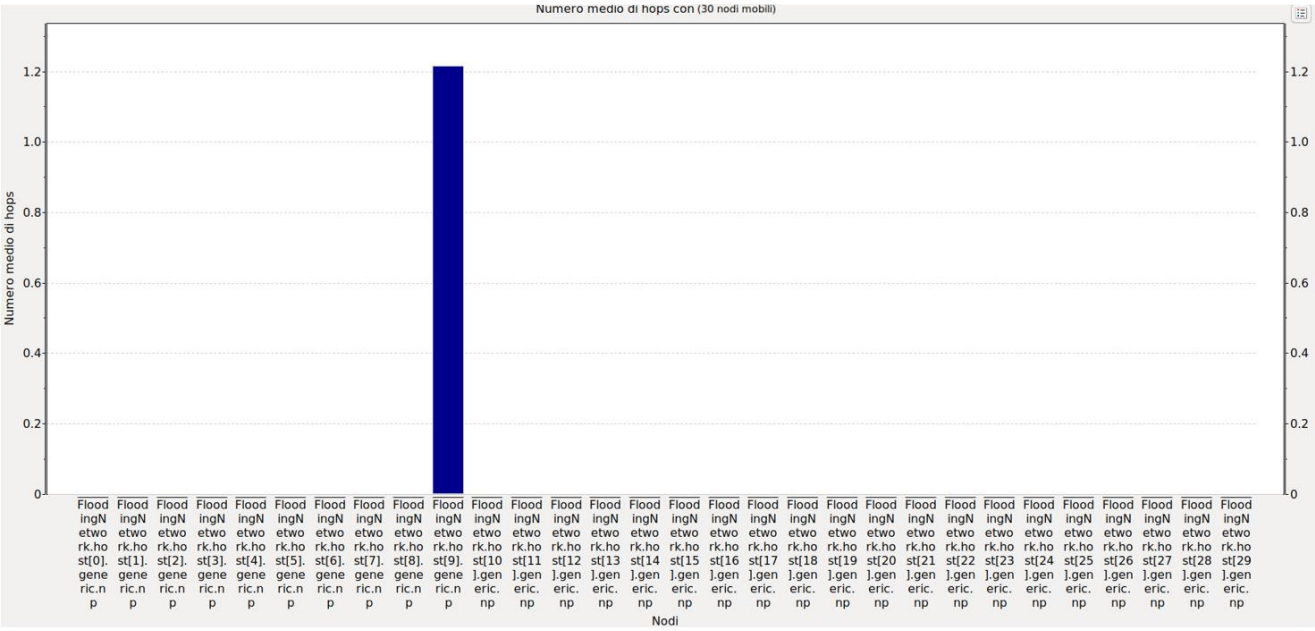
- *NUMERO MASSIMO DI PACCHETTI NELLA CODA DI TRASMISSIONE DEL MAC DI OGNI NODO*



L'aggiunta di ulteriori nodi ha comportato in questo caso un ulteriore beneficio: nella maggior parte dei nodi soltanto un pacchetto rimane accodato nella coda del livello MAC, mentre in soli 5 nodi ne restano accodati 2. Ancora una volta siamo ben lontani dalla saturazione delle code, dunque anche con 20 nodi mobili la rete sta funzionando egregiamente.

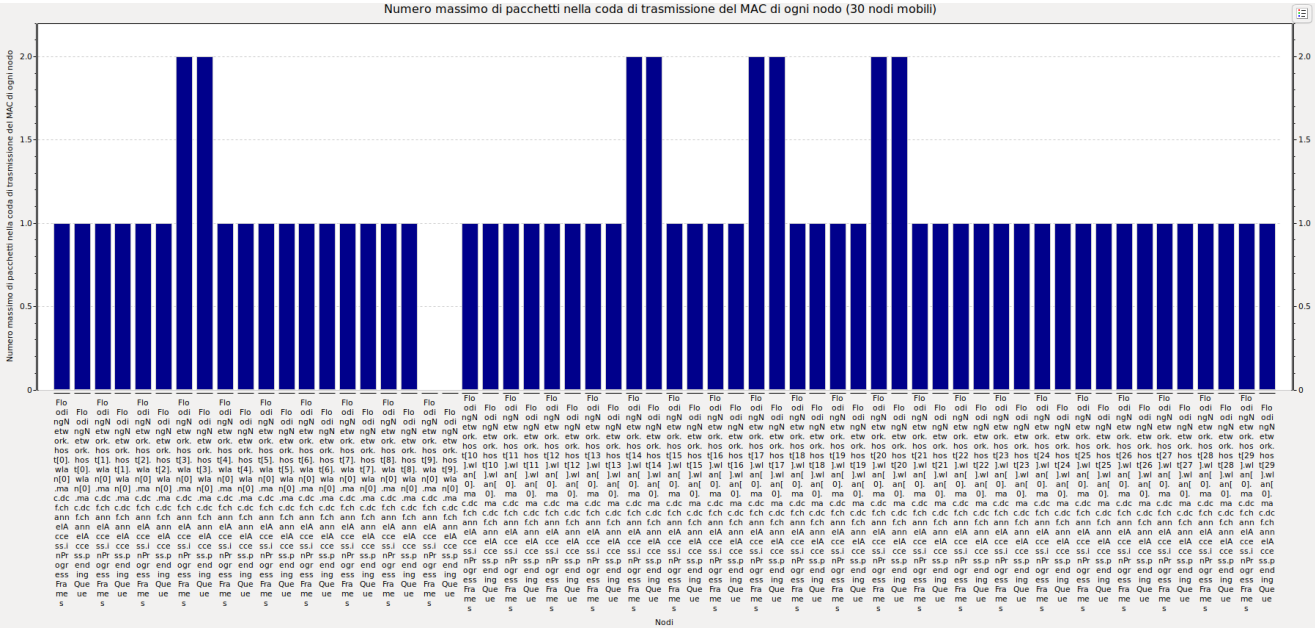
d) Configurazione con 30 nodi mobili e Send Interval pari a 1s

- *NUMERO MEDIO DI HOPS*



Il passaggio da 20 a 30 nodi mobili ha comportato l'incremento del numero medio di hops che adesso è diventato di poco superiore a 1.2.

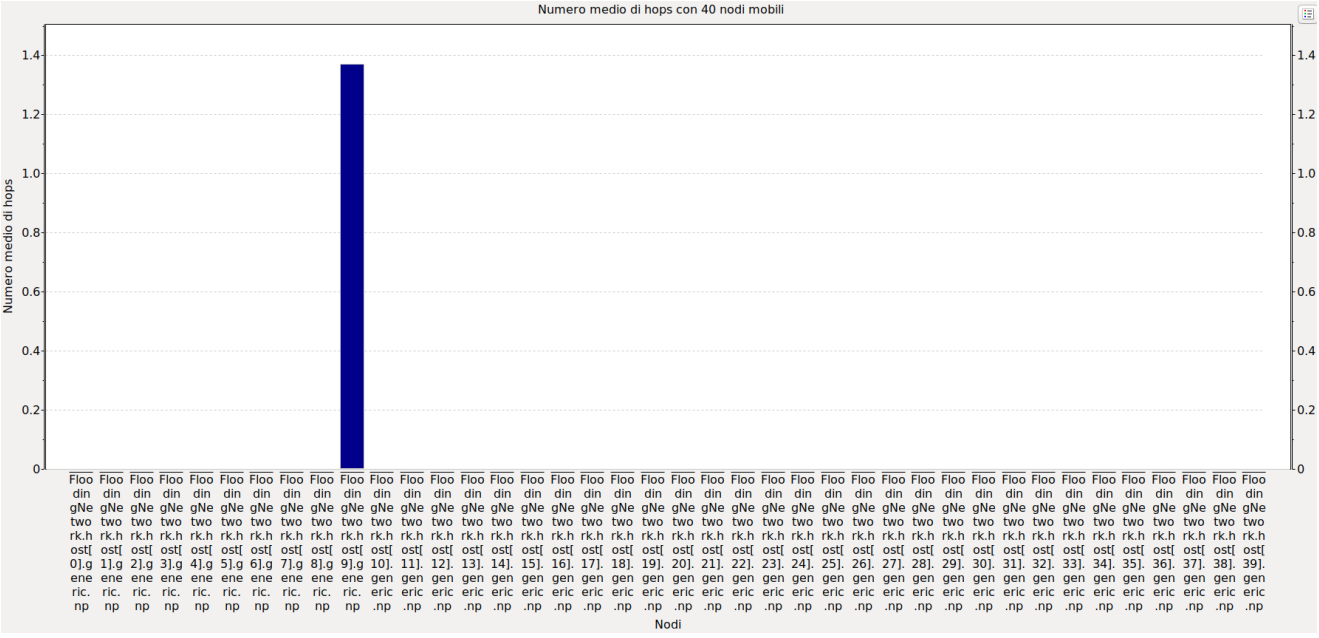
- *NUMERO MASSIMO DI PACCHETTI NELLA CODA DI TRASMISSIONE DEL MAC DI OGNI NODO*



L'aggiunta di ulteriori nodi ha comportato in questo caso un ulteriore beneficio: nella maggior parte dei nodi soltanto un pacchetto rimane accodato nella coda del livello MAC, mentre in soli 4 nodi ne restano accodati 2. Ancora una volta siamo ben lontani dalla saturazione delle code, dunque anche con 30 nodi mobili la rete sta funzionando egregiamente.

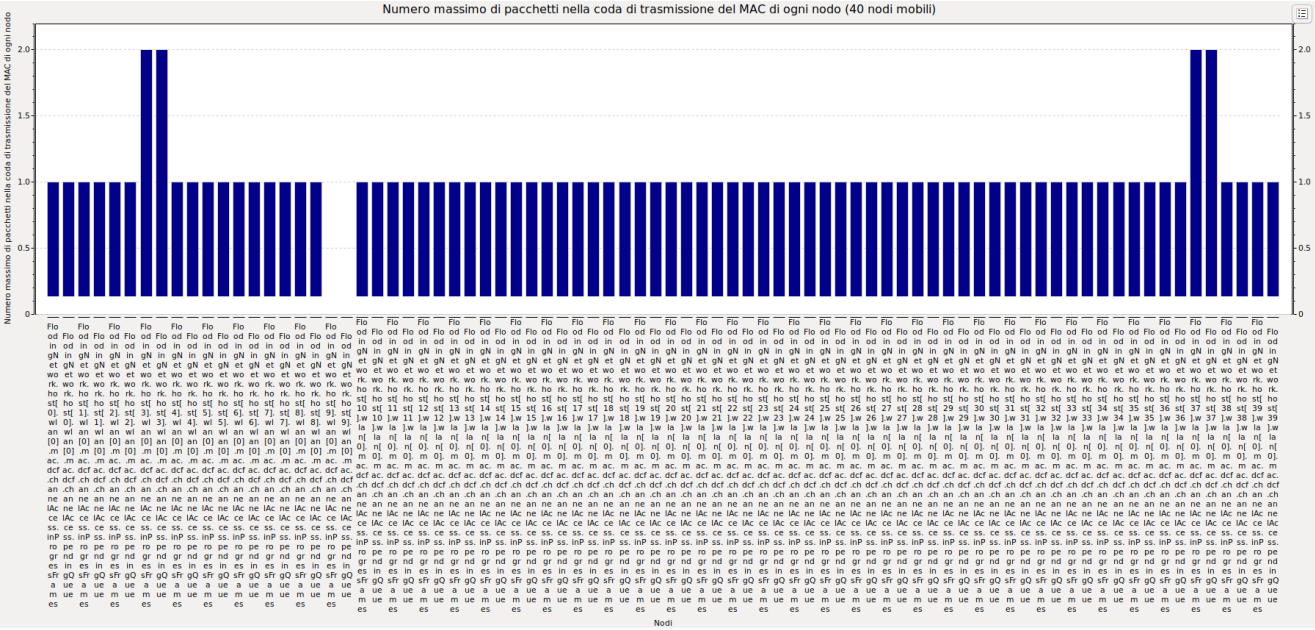
e) Configurazione con 40 nodi mobili e Send Interval pari a 1s

- *NUMERO MEDIO DI HOPS*



L'aggiunta di ulteriori nodi ha innalzato ancora di più il numero medio di hops necessari affinché il messaggio giunga al nodo destinatario, ossia all'**host[9]**. Qui il numero medio di hops è pari a circa 1.38.

- *NUMERO MASSIMO DI PACCHETTI NELLA CODA DI TRASMISSIONE DEL MAC DI OGNI NODO*



L'aggiunta di ulteriori nodi ha comportato in questo caso un ulteriore beneficio: nella maggior parte dei nodi soltanto un pacchetto rimane accodato nella coda del livello MAC, mentre in soli 2 nodi ne restano accodati 2. Ancora una volta siamo ben lontani dalla saturazione delle code, dunque anche con 40 nodi mobili la rete sta funzionando egregiamente.

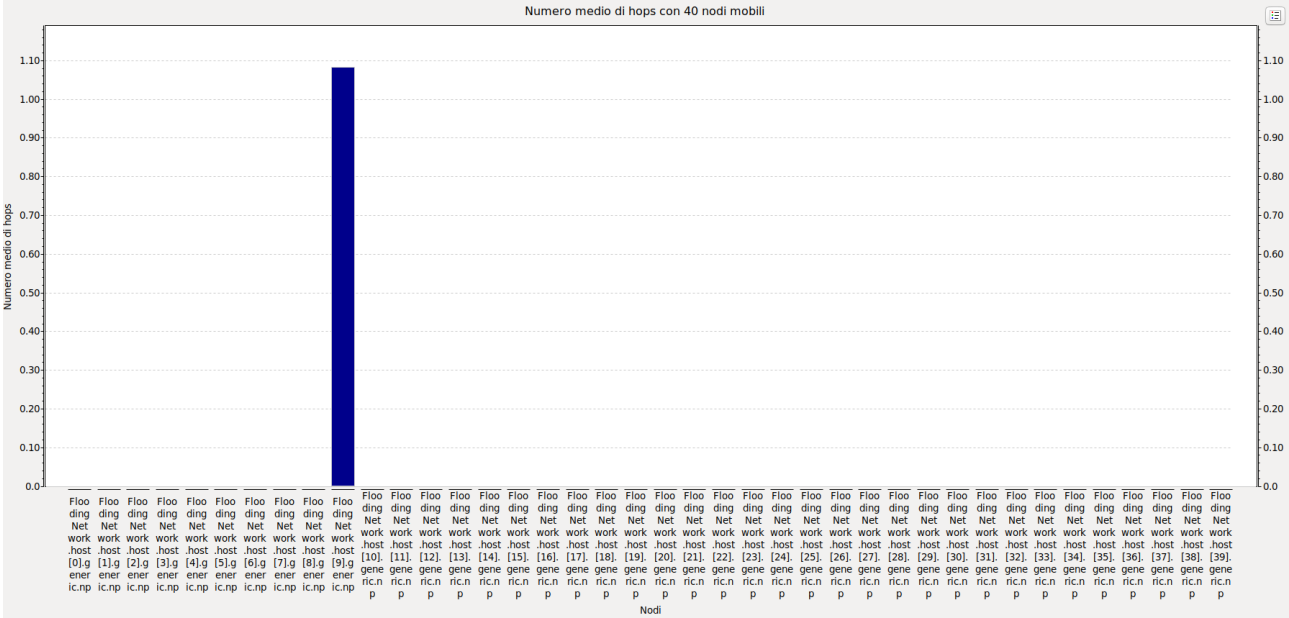
Da queste simulazioni ci siamo resi conto del fatto che anche incrementando di molto il numero di nodi le statistiche si sono mantenute pressocché costanti. Ciò significa che all'aumentare del numero di nodi la rete continua a funzionare egregiamente e questo è un fattore positivo che implica un'elevata scalabilità della rete. Abbiamo ottenuto questi risultati settando il Send Interval del livello applicativo di ciascun nodo pari a 1s.

Per comprendere come si comporta la rete al diminuire del Send Interval abbiamo deciso di effettuare un'ulteriore simulazione considerando uno scenario di rete costituita da 40 nodi mobili e settando il Send Interval pari a 1ms.

I risultati ottenuti sono di seguito riportati.

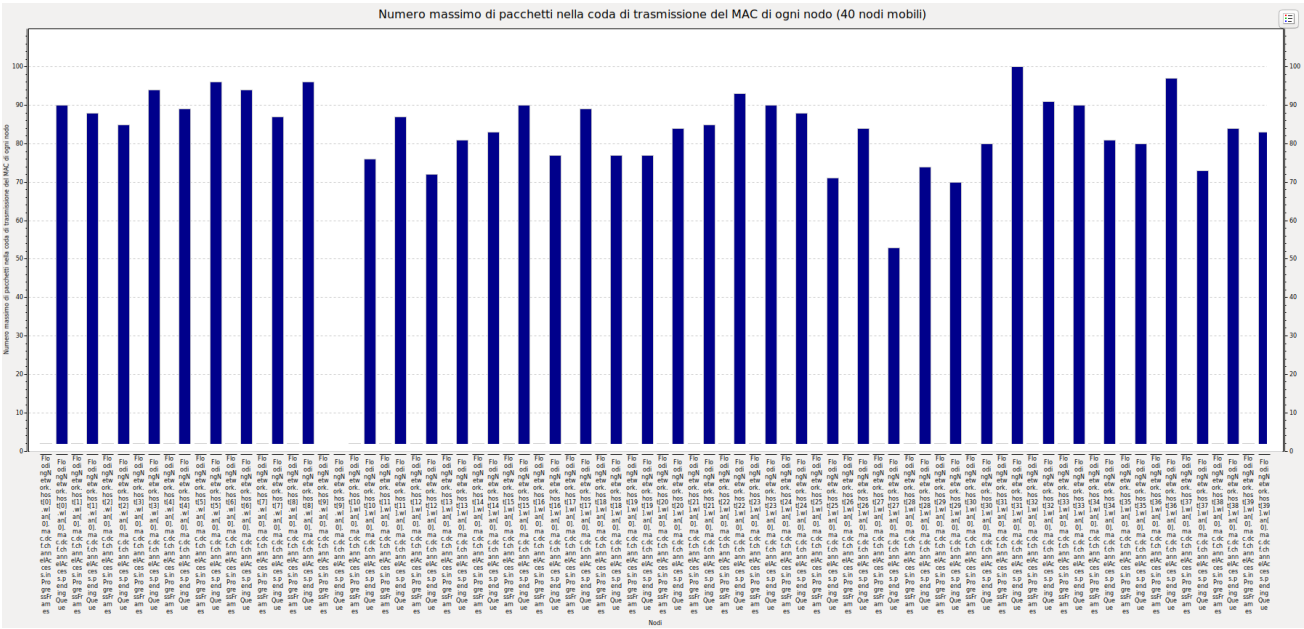
f) Configurazione con 40 nodi mobili e Send Interval pari a 1ms

- *NUMERO MEDIO DI HOPS*



Analogamente ai casi precedenti il numero medio di hops è rimasto pressoché invariato.

- *NUMERO MASSIMO DI PACCHETTI NELLA CODA DI TRASMISSIONE DEL MAC DI OGNI NODO*



Come ci aspettavamo l'aver abbassato il Send Interval ha comportato un incremento del numero di pacchetti accodati nelle code del livello MAC di ciascun nodo. In alcune code viene raggiunta addirittura la capacità massima di 100 pacchetti accodati. Ciò significa che le code che hanno raggiunto la capienza massima non potranno accogliere ulteriori pacchetti fintanto che non si libera almeno uno slot per ospitare un nuovo pacchetto. Dunque, fino a quando la coda ha un numero di pacchetti accodati inferiore a 100 può ricevere ulteriori pacchetti, ma raggiunta la capacità massima i successivi pacchetti devono essere necessariamente scartati.

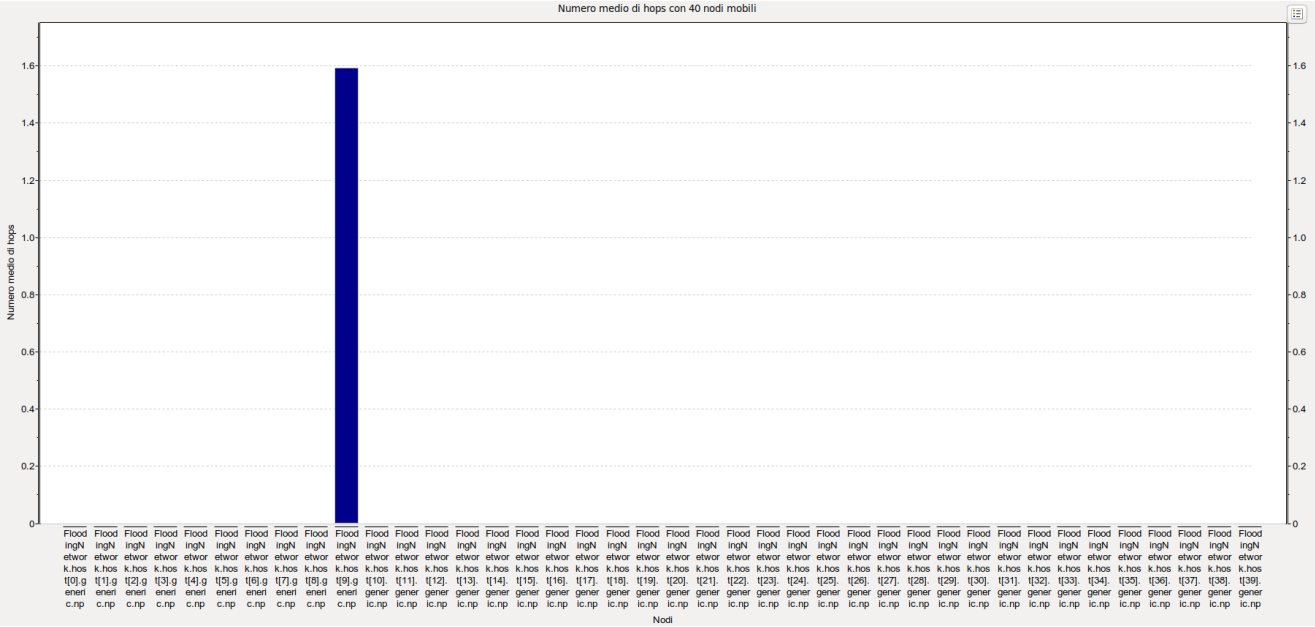
Ciò significa che con un Send Interval pari a 1 ms la rete non riesce a gestire tutto il traffico generato dai vari nodi.

Al fine di comprendere qual è il massimo valore di Send Interval in corrispondenza del quale la rete continua a funzionare egregiamente abbiamo effettuato ulteriori simulazioni e siamo giunti alla conclusione che fino ad un Send Interval dell'ordine di 10^{-2} la rete riesce a gestire tranquillamente il traffico generato dai vari nodi, mentre, come abbiamo visto nella simulazione precedente, con un Send Interval dell'ordine di 10^{-3} (1 ms) la rete è eccessivamente sovraccaricata dal punto di vista del numero di pacchetti presenti nella coda di trasmissione del MAC di ogni nodo e dunque le code di trasmissione dei nodi vanno in saturazione.

A seguire è riportata la simulazione in cui abbiamo 40 nodi mobili e un Send Interval dell'ordine di 10^{-2}.

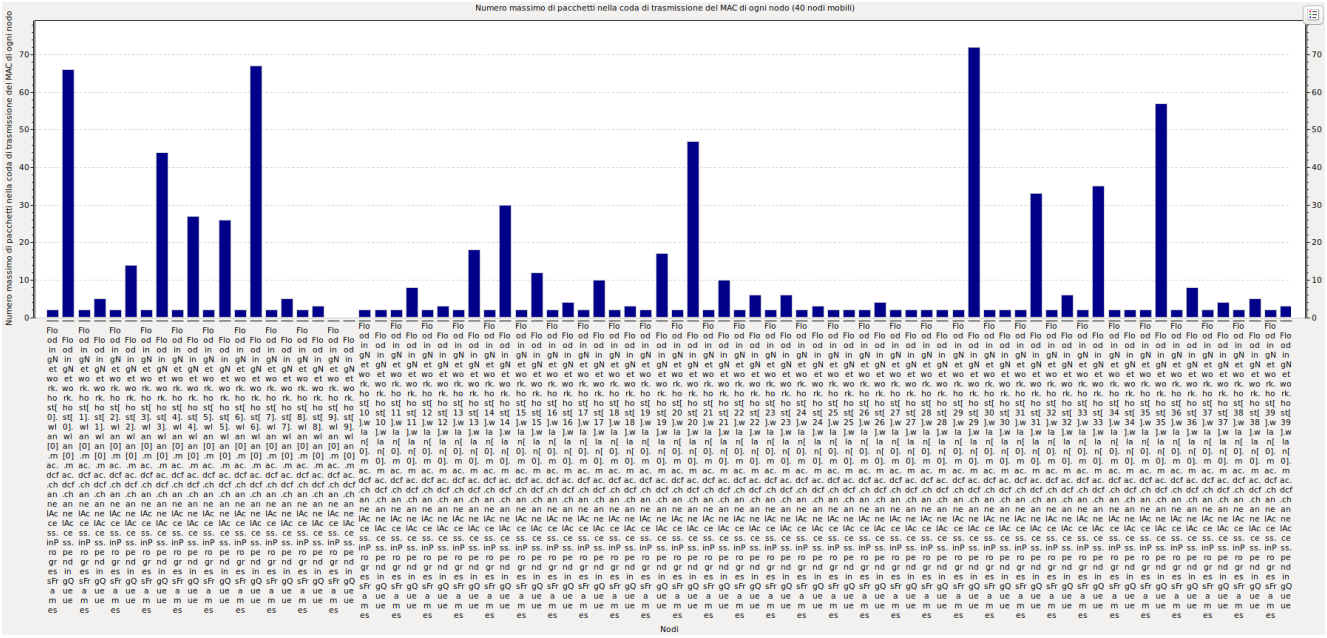
g) Configurazione con 40 nodi mobili e Send Interval pari a 10^{-2} s

- *NUMERO MEDIO DI HOPS*



Il passaggio da un Send Interval dell'ordine di 10^{-3} ad uno dell'ordine di 10^{-2} non ha influito in maniera significativa sul numero medio di hops che, invece, è influenzato fortemente dal numero di nodi presenti e da come questi sono disposti (la disposizione dei nodi mobili in un certo istante è casuale).

- *NUMERO MASSIMO DI PACCHETTI NELLA CODA DI TRASMISSIONE DEL MAC DI OGNI NODO*



In quest'ultima figura si nota come con un Send Interval dell'ordine di 10^{-2} nessuna coda è andata in saturazione. Ciò significa che, come abbiamo detto sopra, con un tale Send Interval la rete funziona egregiamente e riesce tranquillamente a gestire il traffico generato dai vari nodi. Ovviamente, dalla figura, si nota anche come in alcuni casi il numero massimo di pacchetti presenti nelle code di trasmissione è elevato e circa pari a 70, ma, poiché ogni singola coda ha una capienza massima di 100 pacchetti, non si ha alcuna perdita dei pacchetti dovuta al raggiungimento della capacità massima delle code di trasmissione e dunque la rete sta funzionando molto bene anche con un Send Interval dell'ordine di 10^{-2}.