

Laureando:
Alexandru Rotariu



Relatore:
Prof. Giuseppe Di Battista

Correlatore:
Francesco Giacinto Lavacca

UNIVERSITÀ DEGLI STUDI ROMA TRE

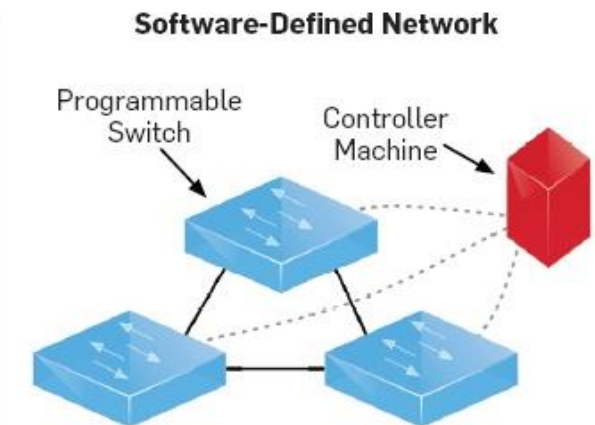
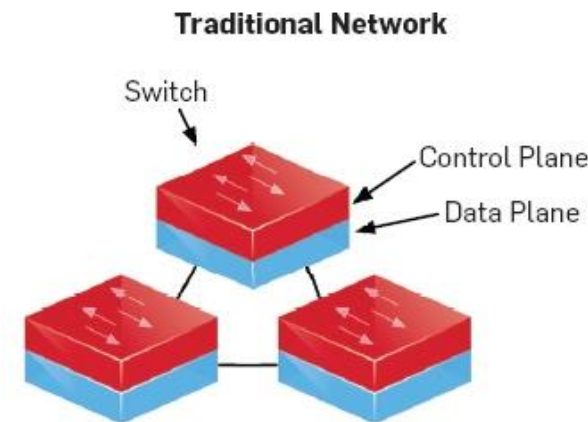
Dipartimento di Ingegneria
Corso di Laurea in Ingegneria Informatica

**PROGETTO E IMPLEMENTAZIONE DI
COMPONENTI DEL CONTROLLER SDN POX IN
AMBIENTE EMULATIVO MININET**

Anno Accademico 2019/2020

Architettura SDN

- Le attuali applicazioni informatiche e i nuovi servizi richiedono che le reti di telecomunicazioni siano sempre più dinamiche, e ciò ha portato allo sviluppo dell'approccio Software Defined Networking (SDN).
- Separazione del piano di controllo dal piano dati che attualmente convivono negli apparecchi di rete.
- Centralizzazione del controllo dell'intera rete in applicazioni dedicate che ne permettano la modifica software: essi sono detti controller.

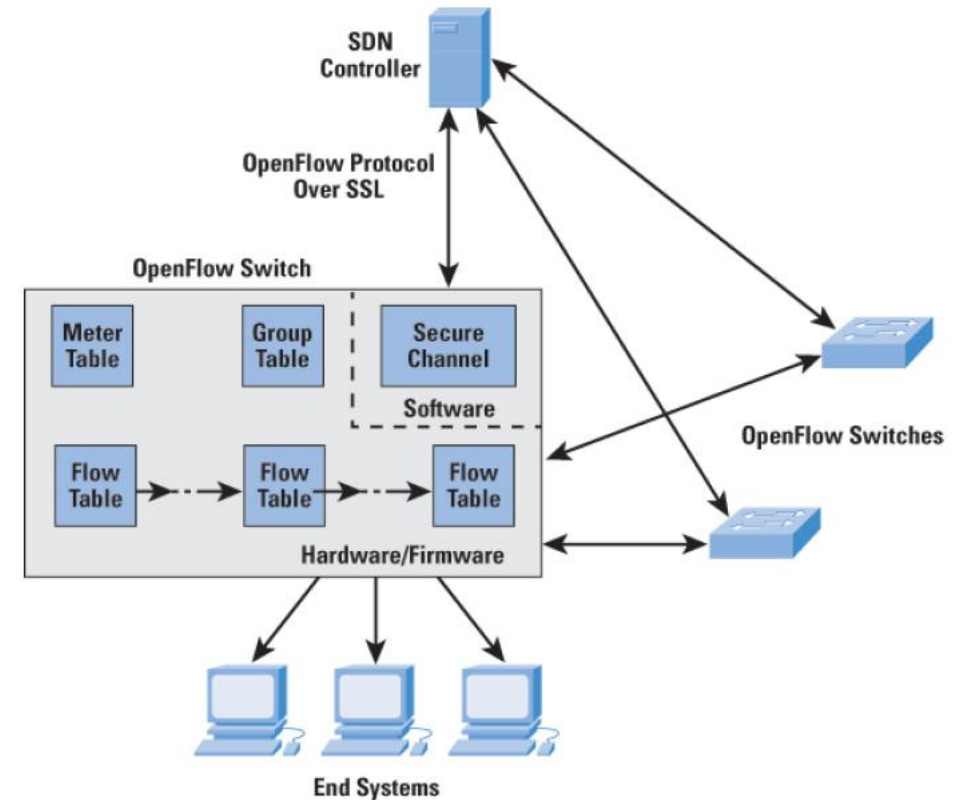


Controller

- Funziona come una sorta di sistema operativo per l'intera rete, tutte le applicazioni e le connessioni passano attraverso di esso.
- Le sue capacità si basano sui component ovvero programmi software che gli permettono di svolgere operazioni: più component vengono usati, più le sue funzionalità aumentano.

Switch OpenFlow

- Possiedono diverse tabelle tramite cui controllano e inoltrano i pacchetti.
- Il contenuto delle tabelle è detto entry mentre l'entità base che rappresenta il traffico dati è detto flow.
- Nelle flow table si individuano i flussi in ingresso che corrispondono ad una delle voci della tabella, a cui è legata l'azione da svolgere con essi.



Mininet

- Emulatore di reti SDN che permette la gestione di un insieme di host, switch, router e collegamenti in un unico ambiente Linux.
- Fornisce il controller SDN «POX» con alcuni component di base.
- Possibilità di creare component personalizzati (linguaggio Python).

Obiettivo

- Realizzare un componente per il controller SDN «POX» in ambiente emulativo Mininet.
- Deve essere in grado di far comunicare tra di loro, tutti i dispositivi presenti nella rete, indipendentemente dalla topologia e in maniera totalmente automatica.
- Tutti i pacchetti devono seguire il percorso più breve per la destinazione.

Procedure sviluppate

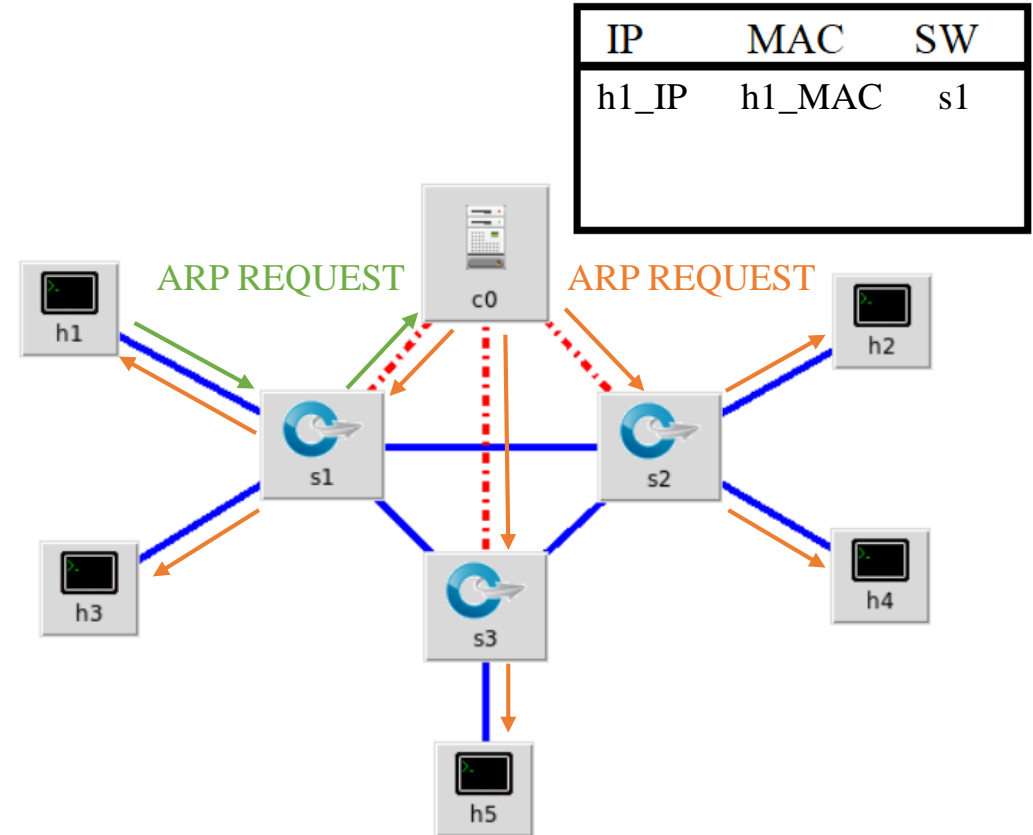
- Inizializzazione delle strutture dati;
- Gestione delle ARP e della comunicazione;
- Calcolo dello shortest path (con l'algoritmo di Dijkstra);
- Installazione delle regole per gestire il routing dei flussi.

Inizializzazione del component

- All'avvio di Mininet, partirà la fase di inizializzazione, in cui il controller rileva la presenza di tutti gli switch e di tutti i collegamenti tra di essi, grazie al Discovery.
- Il component permette di salvare tutte queste informazioni in adeguate strutture dati, in modo da poter essere utilizzare ai fini del calcolo dello shortest path.
- Dato che non viene usato lo spanning tree, tutte le interfacce degli switch che non comunicano direttamente con un host vengono impostate su NO FLOOD (vengono disabilitate), in modo da evitare il loop di pacchetti.
- Viene creata una tabella, utilizzata esclusivamente dal controller, per tenere traccia delle posizioni degli host. Questa tabella è inizialmente vuota.

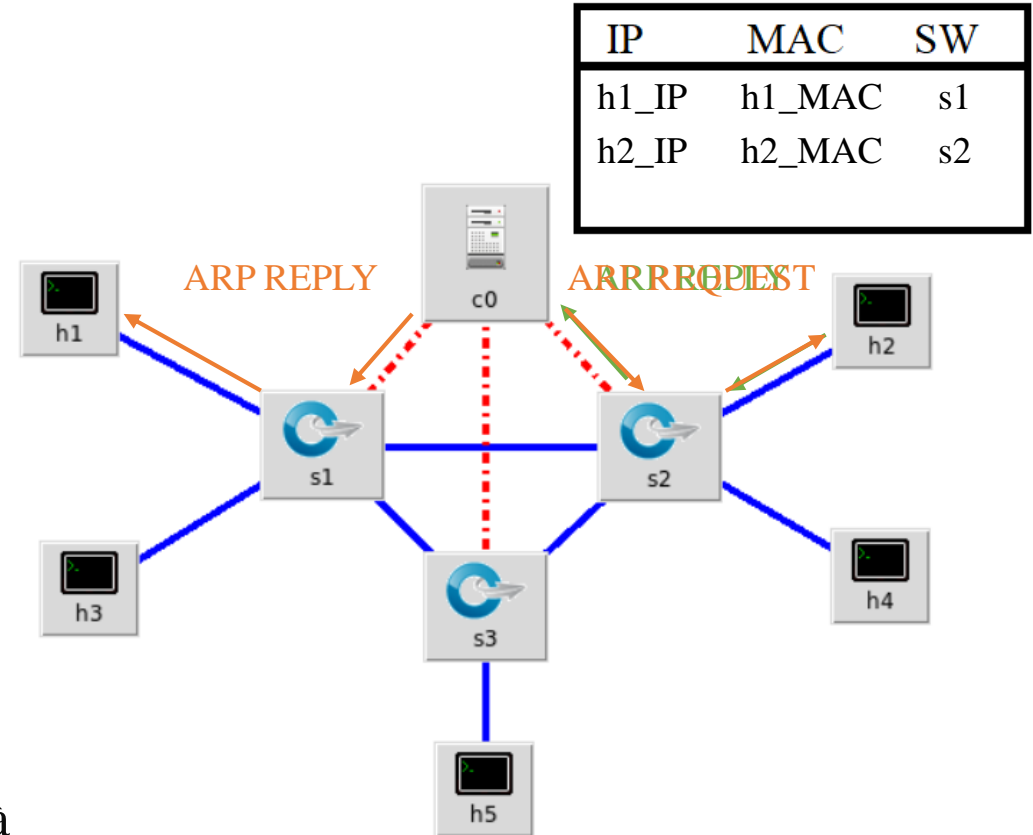
Gestione dell'ARP (caso 1)

- L'host h1 desidera comunicare con l'host h2;
- h1 manda un ARP request in broadcast;
- Il pacchetto arriverà allo switch s1, il quale avendo la flow table vuota, lo inoltrerà al controller;
- Il controller, scopre lo switch al quale è collegato h1, il suo MAC e il suo IP. Aggiunge queste informazioni alla sua tabella;
- Si passa alla gestione del pacchetto ARP: l'IP del destinatario non è presente nella tabella del controller, allora il pacchetto viene inoltrato a tutti gli switch presenti della rete;
- Dunque anche lo s2 che comunica con h2 riceve il pacchetto e lo inoltra su tutte le porte (tranne quelle impostate su NO FLOOD. L 'ARP request arriva a destinazione.



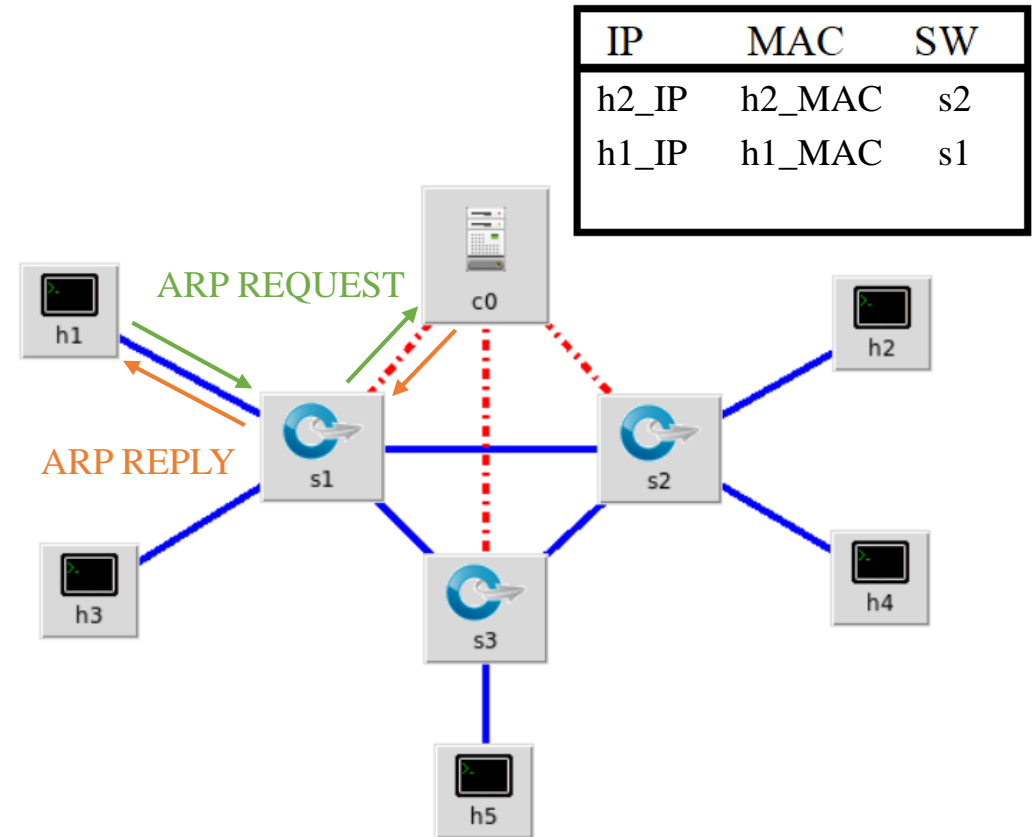
Gestione dell'ARP (caso 1)

- L'host h2 riceve l'ARP request di h1, inoltrata dal controller;
- h2 risponde con un ARP replay all'indirizzo IP di h1;
- Il pacchetto arriverà allo switch s2, il quale avendo le flow table vuote, lo inoltrerà al controller ;
- Il controller, scopre così la posizione di h2 (lo switch al quale è collegato), il suo MAC e il suo IP. Aggiunge queste informazioni alla sua tabella;
- Si passa alla gestione del pacchetto ARP: l' IP del destinatario sarà sicuramente nella tabella del controller quindi inoltra il pacchetto allo switch più vicino ad esso (s1);
- Lo switch inoltrerà il pacchetto anche ad h1 che scoprirà il MAC del destinatario (h2).



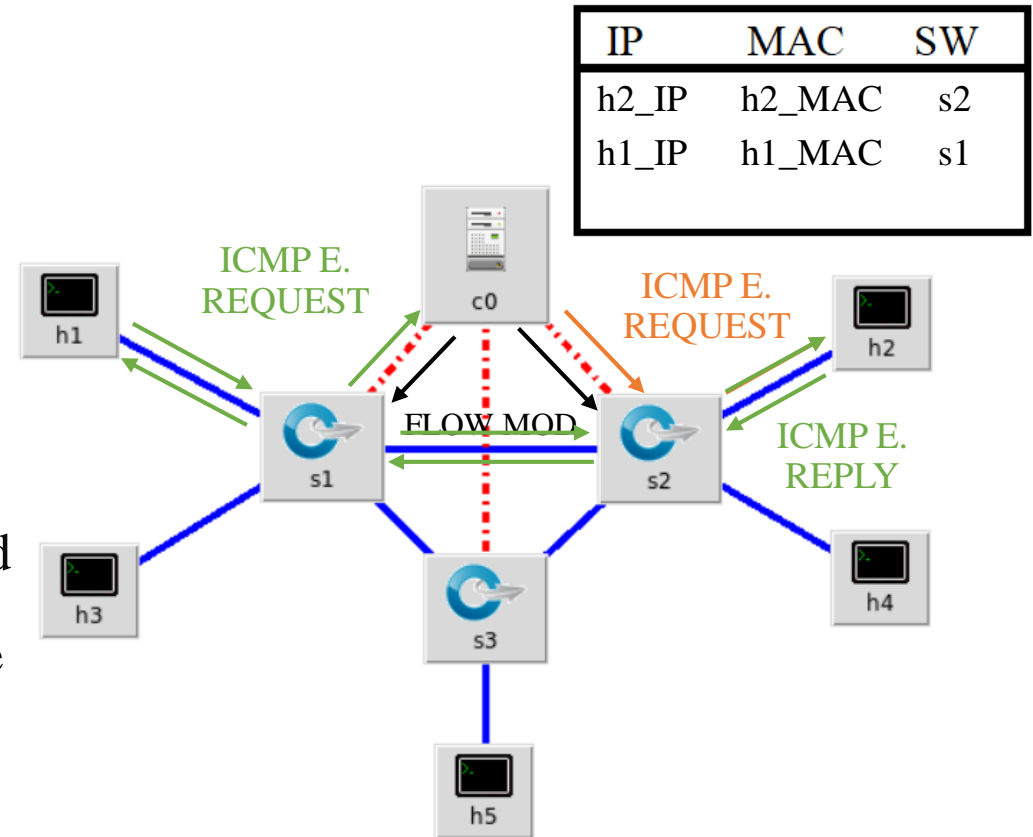
Gestione dell'ARP (caso 2)

- L'host h1 desidera comunicare con l'host h2;
- h1 manda un ARP request in broadcast;
- Il pacchetto arriverà allo switch s1, il quale avendo la flow table vuota, lo inoltrerà al controller;
- Il controller, scopre lo switch al quale è collegato h1, il suo MAC e il suo IP. Aggiunge queste informazioni alla sua tabella;
- Si passa alla gestione del pacchetto ARP: l'IP del destinatario è presente nella tabella del controller allora si ha anche il suo MAC, quindi il controller risponde con un ARP Reply per conto di h2, direttamente allo switch s1.

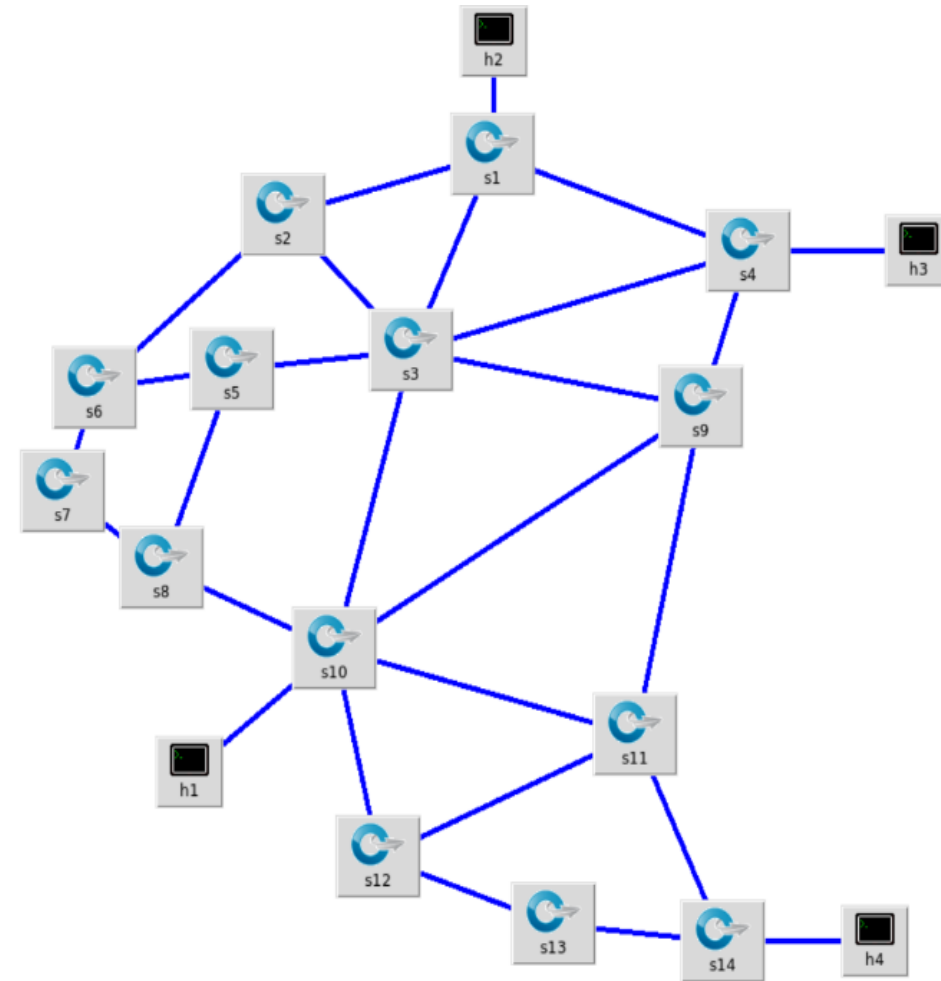


Per i pacchetti a seguire...

- Una volta scoperto il MAC di h2, h1 manderà il primo pacchetto della comunicazione (supponiamo sia un ICMP Echo Request) allo switch s1;
- S1 inoltrerà il pacchetto al controller, il quale, oltre a consegnarlo direttamente allo switch destinatario, trova il percorso più breve da h1 ad h2 (facendo uso dell'algoritmo di Dijkstra) ed installa le opportune FLOW RULE nelle flow table di ogni switch (ambe le direzioni);
- h2, risponde con un ICMP Echo Reply, il quale arriva ad s2, che questa volta però avrà una regola per i flussi verso h1, nelle sua flow table. Il pacchetto viene dunque inoltrato sull'interfaccia specificata;
- Si segue questa logica fino all'ultimo switch il quale lo inoltrerà direttamente al destinatario.

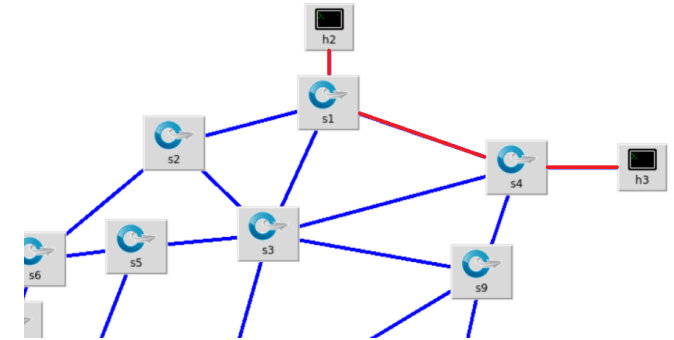


Condizioni iniziali per i test



Test 1

h2 \rightarrow h3 (2 hop)



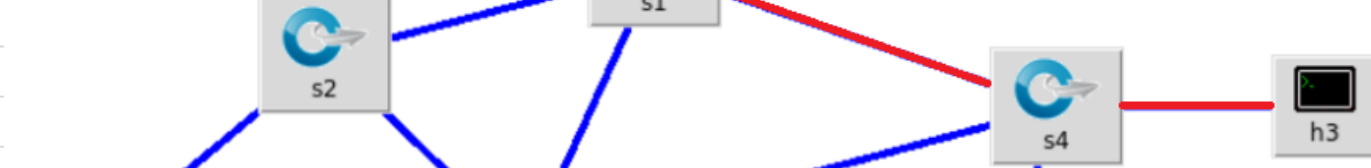
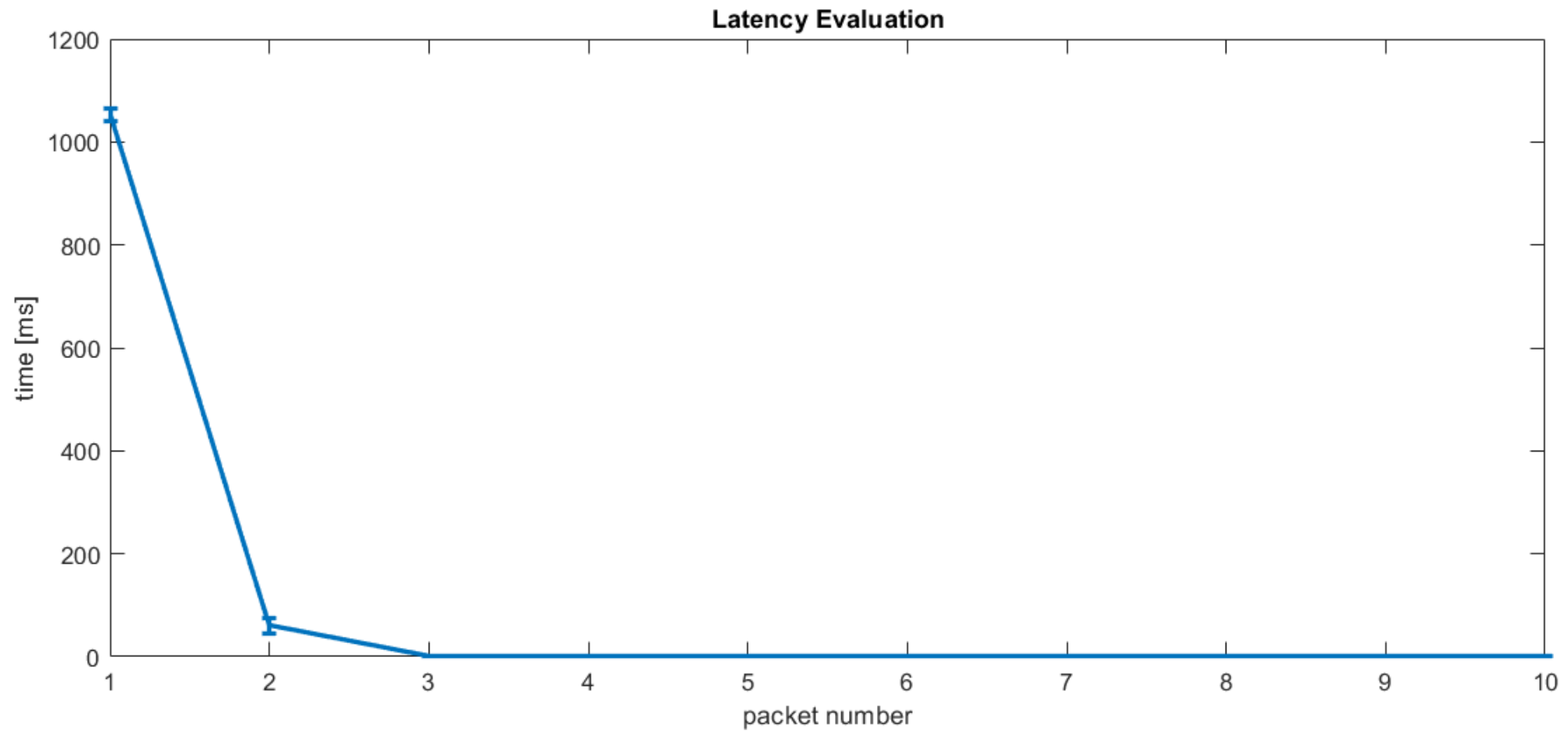
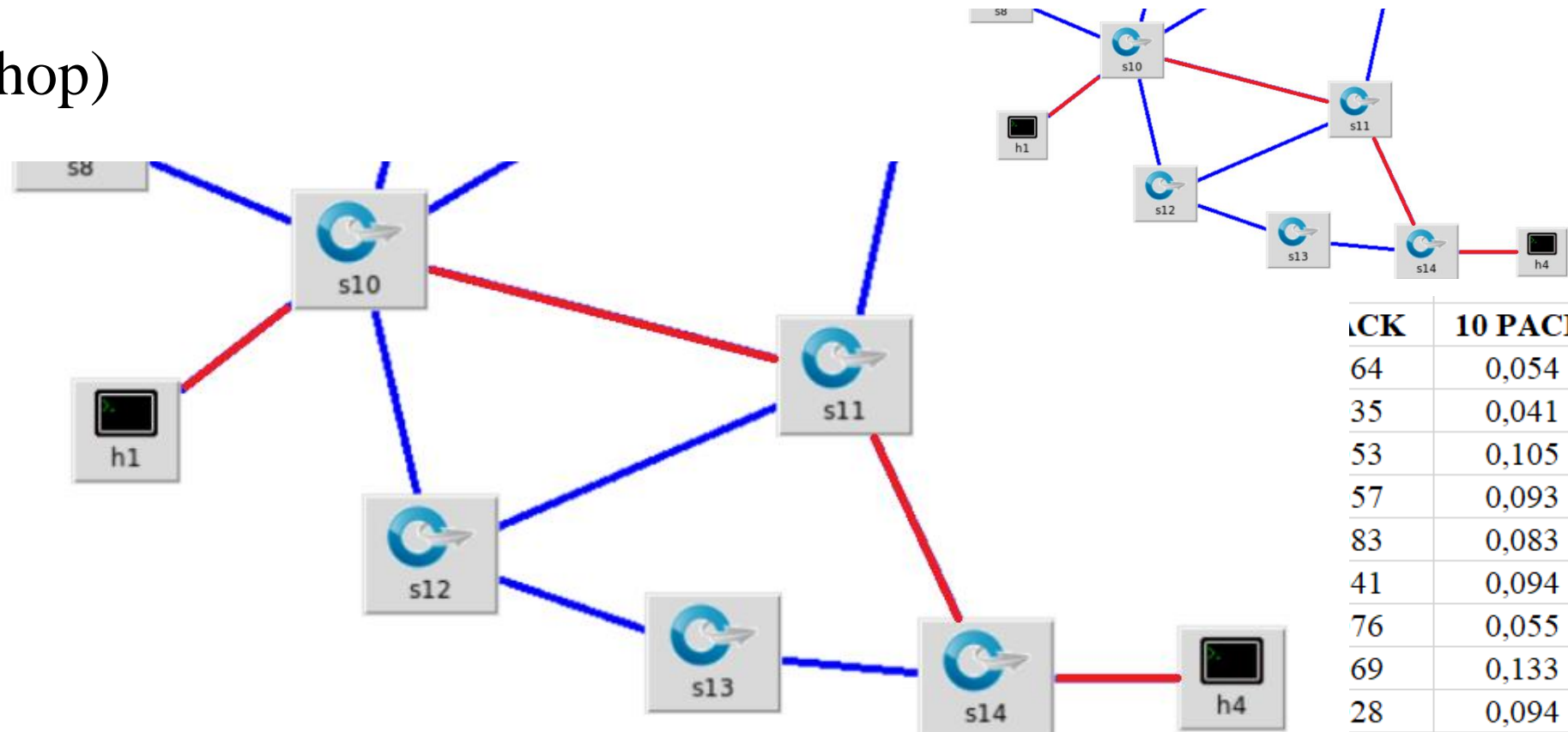
TEST 1	1 PACK								9 PACK	10 PACK
1 PING	1054								0,049	0,074
2 PING	1050								0,062	0,081
3 PING	1065								0,037	0,071
4 PING	1043								0,067	0,052
5 PING	1044								0,087	0,049
6 PING	1030								0,08	0,082
7 PING	1054								0,082	0,082
8 PING	1069								0,064	0,086
9 PING	1045								0,084	0,082
10 PING	1064								0,083	0,084
MEDIA	1051,8	60,53	0,4036	0,096	0,071	0,0814	0,0823	0,0798	0,0695	0,0743
VARIANZA	129,16	185,2901	0,0054416	0,0005982	0,0001546	5,84E-06	0,0006666	0,0002198	0,0002535	0,0001602

Grafico Test 1



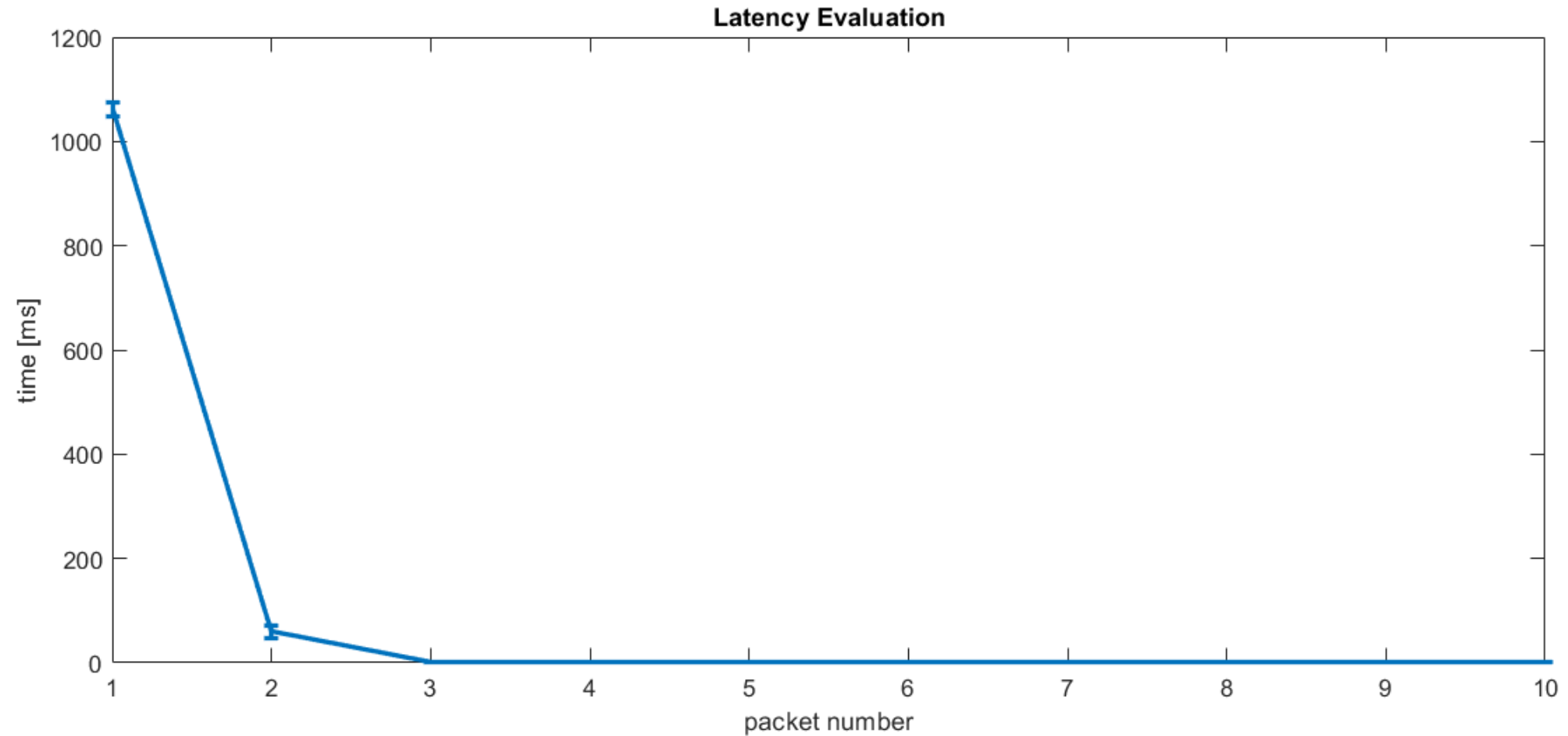
Test 2

h1 → h4 (3 hop)



TEST 2	1 P									ACK	10 PACK
1 PING	10									64	0,054
2 PING	10									35	0,041
3 PING	10									53	0,105
4 PING	10									57	0,093
5 PING	10									83	0,083
6 PING	10									41	0,094
7 PING	10									76	0,055
8 PING	10									69	0,133
9 PING	10									28	0,094
10 PING	1059	50,6	0,473	0,043	0,077	0,023	0,091	0,075	0,069	0,036	
MEDIA	1060,7	59,5	0,4669	0,0604	0,0726	0,0708	0,0694	0,05925	0,0575	0,0788	
VARIANZA	166,61	135,188	0,0154059	0,0005596	0,0002288	0,0005972	0,0004624	0,0005952	0,0002969	0,0008748	

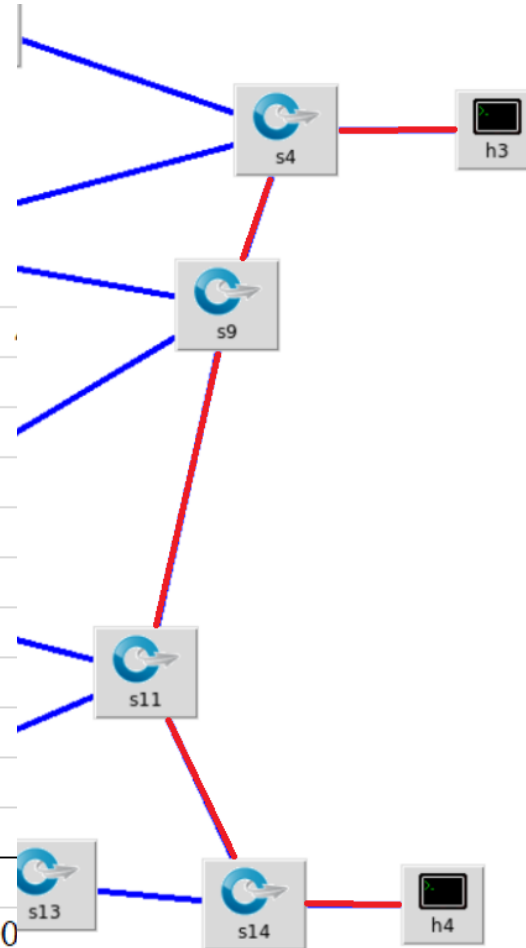
Grafico Test 2



Test 3

h3 → h4 (4 hop)

TEST 3	1 PACK	2 PACK	3 PACK
1 PING	1064	58,1	0,945
2 PING	1046	54,1	0,894
3 PING	1071	76,2	0,174
4 PING	1071	70,6	0,824
5 PING	1068	67,7	0,188
6 PING	1027	41,4	0,149
7 PING	1060	60,6	0,855
8 PING	1057	49,3	0,48
9 PING	1062	62,2	0,459
10 PING	1091	82,6	1,32
MEDIA	1061,7	62,28	0,6288
VARIANZA	255,21	139,6936	0,141967



7 PACK	8 PACK	9 PACK	10 PACK
0,044	0,097	0,036	0,04
0,026	0,061	0,105	0,049
0,07	0,044	0,07	0,097
0,054	0,055	0,154	0,038
0,099	0,092	0,074	0,041
0,085	0,068	0,064	0,056
0,063	0,09	0,074	0,081
0,062	0,069	0,08	0,068
0,058	0,102	0,042	0,067
0,065	0,86	0,075	0,087
0,0626	0,1538	0,0774	0,0624
0,0003668	0,05575	0,0009866	0,0003976

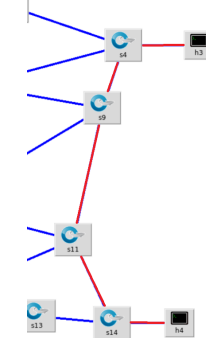
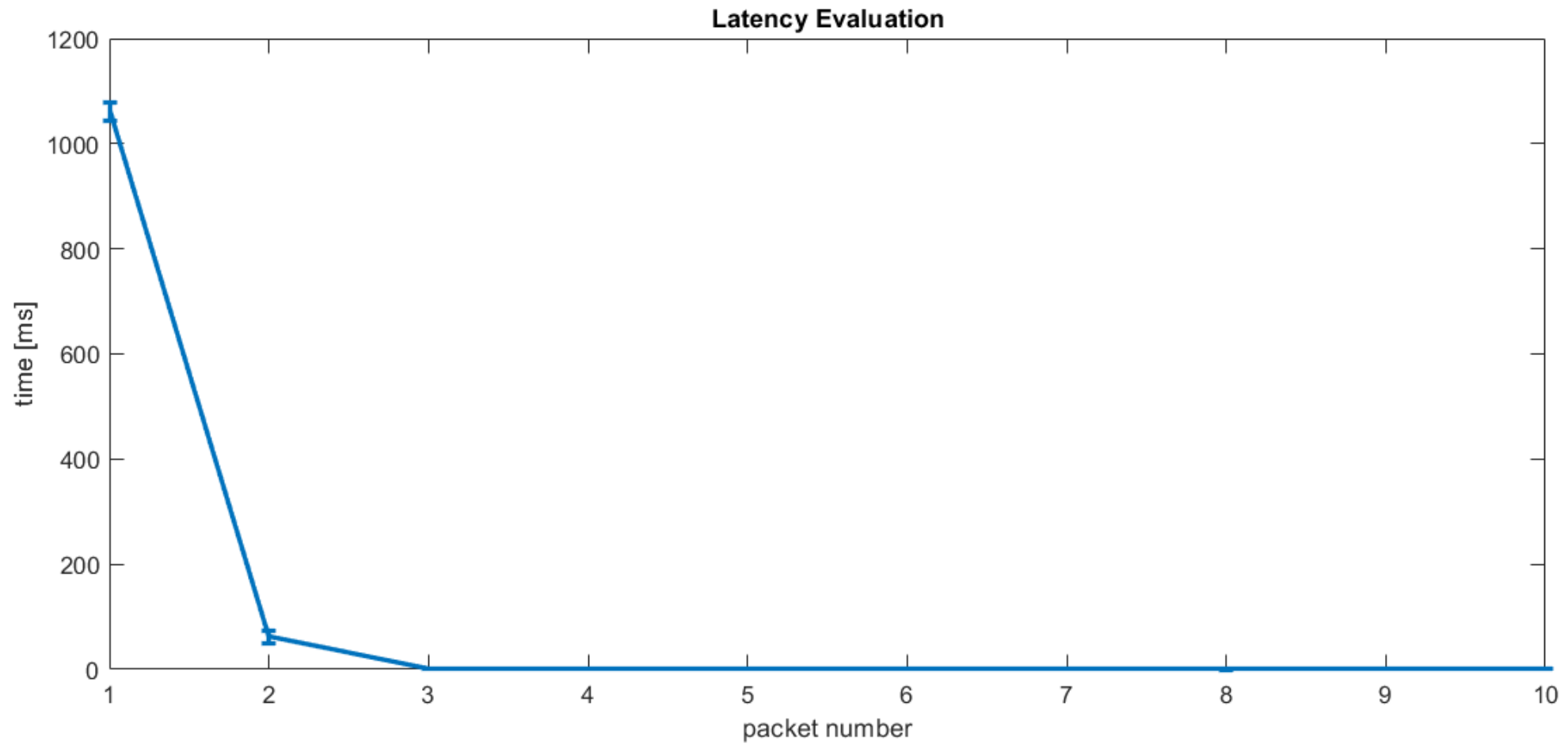


Grafico Test 3



Conclusione e Sviluppi futuri

- I risultati in tutti e tre i test sono molto simili, nonostante siano stati fatti con numeri di nodi intermedi diversi: il numero di hop non influenza il tempo di convergenza della rete.
- Implementazione dell' algoritmo di Yen, per avere K percorsi di backup;
- Definizione di algoritmi che permettano di abbattere i tempi di risposta dei primi pacchetti preinstallando delle regole negli switch;
- Tutto il materiale è disponibile al link:
<https://github.com/alexandrurotariu/ComponentPOX>

Grazie dell'attenzione.
