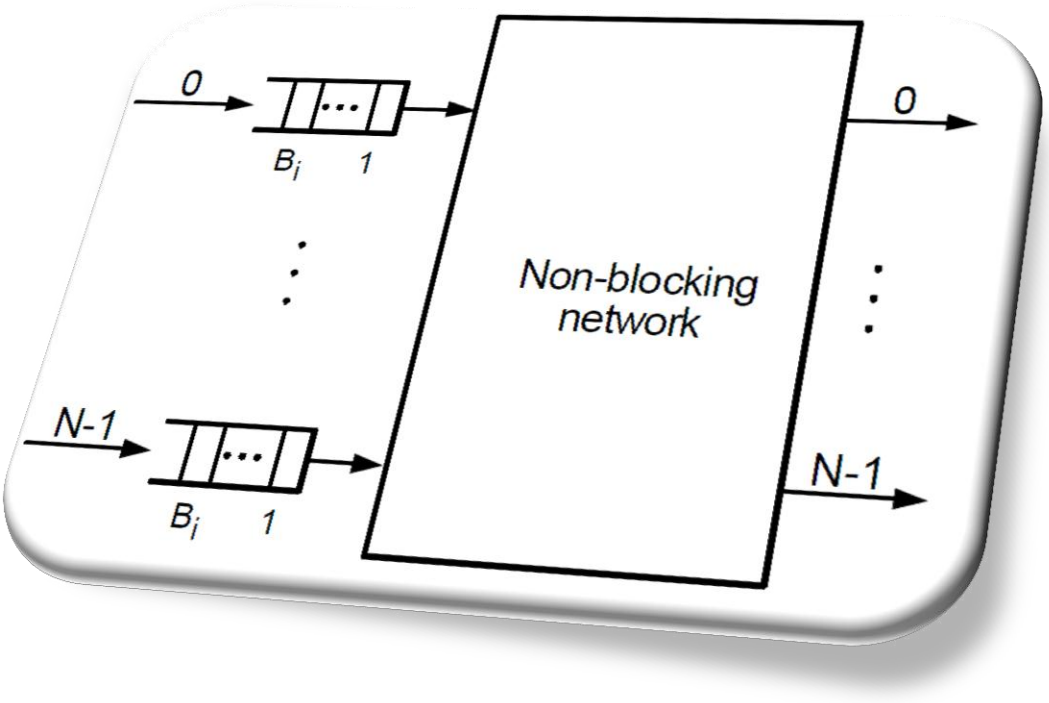
Simulatore di reti con buffer in ingresso



Corso	Reti di Connessione e Instradamento
Docente	Achille Pattavina
A.A.	2010-2011
Gruppo di lavoro	Diego Tuzi
	Larte Saliai
	Marco Visin
	Berner Panti
Assegnazione progetto	22/12/2010
Consegna progetto	05/2011

POLITECNICO DI MILANO

Sommario

Introduzione

La tecnologia di switching a lunghezza fissa è largamente riconosciuta come un consolidato approccio che permette di raggiungere un'elevata efficienza per le reti di commutazione a pacchetto. I pacchetti IP di lunghezza variabile che si presentano all'ingresso di una generica rete, sono segmentati in celle di lunghezza fissa e sono re-assemblati all'uscita della rete. I primi switches ad alta velocità per sistemi di commutazione a pacchetto usavano buffer in ingresso o condivisi, tali soluzioni, però, soffrono di limitazioni di throughput. A causa di tali limitazioni, nel passato, le ricerche hanno incentrato lo studio su architetture con buffer in uscita. La domanda iniziale per la capacità degli switch era contenuta nell'intervallo delimitato tra centinaia di Mbit/s sino ad arrivare a pochi Gbit/s. Per tali capacità il buffer in uscita sembrava essere la scelta migliore, grazie alle elevate performance del rapporto throughput/delay e la bassa quantità di memoria utilizzata. Comunque, con la richiesta di switches con capacità sempre maggiore, si è individuato il punto debole delle architetture con buffer in output ovvero la velocità di scrittura sulla memoria. Sebbene gli switches con buffer in uscita abbiano un ottimo rapporto throughput/delay per tutte le distribuzioni di traffico, lo speed-up richiesto alla velocità per le operazioni sulla memoria limita la scalabilità di quest'architettura. Al fine di costruire switch ad alta velocità, i ricercatori stanno focalizzando la loro attenzione su architetture con buffer in ingresso, o strutture ibride con buffer in ingresso e in uscita.

Gli switches con buffer in input sono utilizzati per ottenere elevate velocità, grazie al fatto che la velocità delle operazioni interne è moderatamente superiore alla velocità del flusso in ingresso. Nell'utilizzo di switch con buffer in ingresso sorgono due problemi:

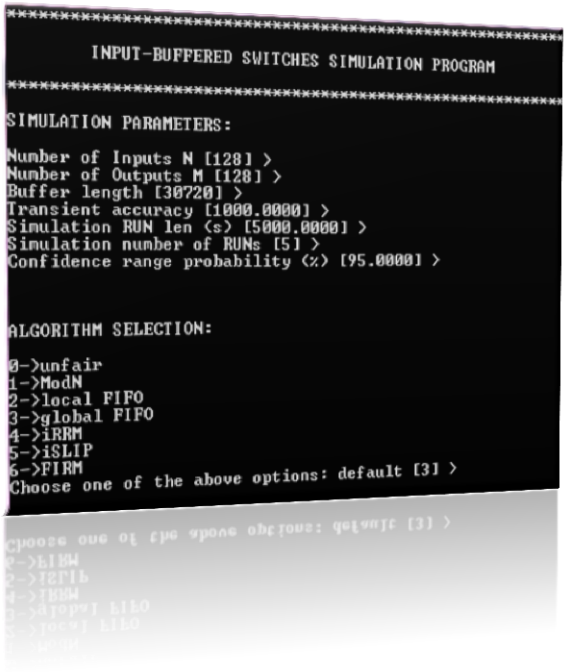
1. La limitazione del throughput a causa dell'Hol (head of line blocking che limita il throughput al 58.6% per politiche di tipo fifo).
2. La necessità di utilizzare una politica che regoli la contesa di un'uscita tra più celle.

Il primo problema può essere aggirato incrementando moderatamente la velocità della switch fabric oppure aumentando il numero di percorsi per ogni uscita. Il secondo problema può essere risolto utilizzando tecniche avanzate di scheduling.

Il progetto presentato in questo documento ha i seguenti obiettivi:

- Realizzazione di un simulatore che riproduca il comportamento di un'architettura con buffer in ingresso e verifichi i problemi di cui sopra;
- Implementazione di tecniche che riescano a limitare i problemi di cui sopra (come l'introduzione dello speed-up o l'adozione di tecniche avanzate di scheduling);
- Analisi dei risultati.

Il simulatore



Il simulatore realizzato in questo progetto permette di riprodurre una struttura di rete non bloccante con numero di ingressi e uscite variabili, inoltre in esso vi è la possibilità di variare numerosi altri parametri.

All'interno del simulatore sono implementati differenti algoritmi, che nonostante utilizzino la stessa rete e abbiano in ingresso lo stesso traffico, generano prestazioni molto differenti.

Le principali ipotesi effettuate nella realizzazione del simulatore sono le seguenti:

- asse temporale discreto con unità minima pari ad uno slot;
- tempo di servizio di un pacchetto, pari a uno slot;
- traffici bilanciati sugli ingressi;
- carico di ogni ingresso uniformemente distribuito su tutte le uscite.

Gli outputs del simulatore sono throughput, ritardo medio, statistica dei pacchetti persi e altro. Inoltre sono creati dei files di testo che costituiscono l'input a degli scripts matlab per la realizzazione di grafici comparativi.

Per comprendere meglio il funzionamento del simulatore si possono individuare le seguenti aree:

- Parametri di input;
- Selezione algoritmo;
- Dati di uscita.

Parametri di Input

Input	Default	Range	Descrizione
N	128	2-256	Numero di ingressi
M	128	2-256	Numero di uscite
Load	1	0.1-1	Carico di traffico su ogni ingresso
Input Buffer length	8	0-2048	Capacità (numero di celle) del buffer di un ingresso
Output Buffer length	8192	0-8192	Capacità (numero di celle) del buffer di una uscita
Transient accuracy	1000	10-10000	Accuratezza del transitorio. Influisce sulla durata del transitorio.
Run length	10000	100-20000	Lunghezza (in slot) di un singolo run di simulazione.
Number of runs	10	2-20	Numero di runs.
Confidence range probability	95	1-100	Valore percentuale dell'intervallo di confidenza.
Output speed-up	1	2-8	Speed up su ogni singola uscita (solo per algoritmi 0,1,2,3).

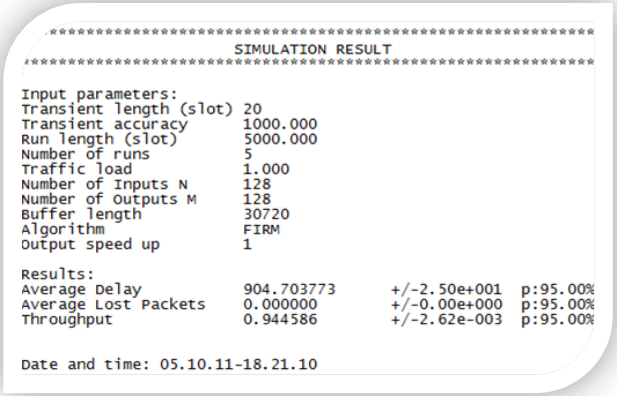
Selezione algoritmo

Numero	Algoritmo	Descrizione
0	**Unfair**	Algoritmo con priorità decrescente a partire dall'ingresso con indice più basso. Quest'algoritmo è scorretto perché alcuni ingressi sono avvantaggiati rispetto ad altri e c'è la possibilità che alcuni rimangano in attesa per un tempo indefinito. Soffre di HOL blocking.
1	**modN**	Algoritmo con priorità circolare. È abbastanza corretto poiché la priorità di trasmissione è concessa in modo circolare a tutti gli ingressi. Non possono verificarsi situazioni di starvation. Soffre di HOL blocking.
2	**localFifo**	Algoritmo che valuta tutti i pacchetti pronti per essere trasmessi di ogni ingresso e assegna la priorità a quei pacchetti che sono in posizione hol da più tempo. Soffre di HOL blocking.
3	**globalFifo**	Algoritmo che valuta tutti i pacchetti pronti per essere trasmessi di ogni ingresso e assegna la priorità a quei pacchetti che sono nel buffer da più tempo. È in assoluto l'algoritmo che nel maggior modo rispetta la politica fifo. Soffre di HOL blocking.
4	**iRRM**	È un algoritmo di sheduling avanzato. Risolve le contese tra ingressi e uscite utilizzando un metodo di selezione basato su schedulers round robin. Ogni iterazione consiste in tre passi (request, grant e accept). Ci possono essere fino a N iterazioni in ogni slot di tempo. Grazie all'uso di schedulers round robin, l'iRRM raggiunge una corretta allocazione della banda tra tutti i flussi. Raggiunge throughput unitario con N iterazioni. Soffre del fenomeno di sincronizzazione delle uscite. Non tiene conto in nessun modo di seguire una politica FIFO, quindi può generare fuori sequenza.
5	**iSLIP**	Molto simile a iRRM ma con dei miglioramenti che eliminano il fenomeno della sincronizzazione delle uscite.
6	**FIRM**	Molto simile a iRRM e a iSLIP ma grazie ad una differente gestione degli schedulers round robin riesce ad ottenere throughput unitario con una sola iterazione e in più, tra gli algoritmi avanzati, è quello che in maggior modo rispetta una politica FCFS (first come first serve).

Dati di uscita

Il simulatore produce in uscita alcuni files di testo:

- simResults_"nomeAlgoritmo".txt
- traceRun_"nomeAlgoritmo".txt
- delayValues_"nomeAlgoritmo".txt
- simParameters_"nomeAlgoritmo".txt



```
**********************************************************
                    SIMULATION RESULT
**********************************************************

Input parameters:
Transient length (slot) 20
Transient accuracy      1000.000
Run length (slot)       5000.000
Number of runs          5
Traffic load            1.000
Number of Inputs N      128
Number of Outputs M     128
Buffer length           30720
Algorithm               FIRM
Output speed up         1

Results:
Average Delay           904.703773      +/-2.50e+001  p:95.00%
Average Lost Packets    0.000000        +/-0.00e+000  p:95.00%
Throughput              0.944586        +/-2.62e-003  p:95.00%


Date and time: 05.10.11-18.21.10
```

simResults_"nomeAlgoritmo".txt

Il file dei risultati contiene un riepilogo dei dati di input e fornisce i seguenti risultati:

Risultati	Descrizione
Average input delay	Ritardo medio di permanenza di un pacchetto nel buffer di ingresso, intervallo di confidenza e probabilità di appartenere all'intervallo specificato.
Average output delay	Ritardo medio di permanenza di un pacchetto nel buffer di uscita, intervallo di confidenza e probabilità di appartenere all'intervallo specificato. Tale dato è diverso da zero solo con speed-up>1.
Average total delay	Ritardo medio totale, formato da: ritardo nel buffer di ingresso, ritardo di attraversamento della rete (posto per semplicità ad uno slot) e ritardo sul buffer di uscita.
Average lost packets	Valore medio di pacchetti persi, intervallo di confidenza e probabilità di appartenere all'intervallo specificato.
Throughput	Valore medio del throughput, intervallo di confidenza e probabilità di appartenere all'intervallo specificato.
Input packet loss probability	Probabilità che un pacchetto è perso perché trova il buffer pieno, calcolata come rapporto tra totale pacchetti persi e totale pacchetti generati dalla rete.
Average iteration	Numero medio di iterazioni necessarie in ogni slot di tempo agli algoritmi iRRM e iSLIP per ottenere il migliore matching possibile tra ingressi e uscite.

Il contenuto del file dei risultati contiene le stesse informazioni che sono visualizzate a video al termine della simulazione.

traceRun_"nomeAlgoritmo".txt

Il file traceRun riporta gli stessi valori di simResults ma relativi ad ogni singolo run, in tal modo si può apprezzare l'evoluzione dei dati di uscita alla fine di ogni run.

delayValues_"nomeAlgoritmo".txt

Il file delayValues contiene un vettore colonna, in cui la riga i-esima rappresenta il ritardo in slot di tempo, mentre il valore contenuto nel vettore rappresenta la frequenza di occorrenza di quel determinato valore di ritardo.

Esempio:

0
15
32
…

Tabella 1 – Esempio file delayValues

La tabella 1 ci dice che ci sono 0 pacchetti che hanno avuto ritardo pari 0 slot, 15 che hanno avuto ritardo 1 slot e 32 che hanno avuto ritardo 2 slot.

simParameters_"nomeAlgoritmo".txt

Il file simParameters contiene un vettore colonna i cui valori numerici corrispondono ad alcuni dei parametri di input. Tale file non contiene dei campi di descrizione per renderlo facilmente maneggiabile con altri linguaggi di programmazione. Lo struttura del file è mostrata in tabella 2.

Transient accuracy
Run length
Number of runs
Load
N
M
Input buffer size
Output buffer size
Speed-up

Tabella 2 – Struttura file simParameters

Scripts Matlab

Una volta eseguita la simulazione per tutti gli algoritmi, selezionando gli stessi parametri di input, è possibile ottenere automaticamente dei grafici comparativi utilizzando due scripts matlab. Inoltre gli scripts eseguono un controllo di coerenza dei dati ovvero, se i files che gli scipts utilizzano sono stati generati con parametri di input diversi il procedimento non viene avviato ma è segnalato un errore.

Gli scripts in questione sono i seguenti:

- distribuzioneRitardi_algBase.m;
- distribuzioneRitardi_algAvanzati.m.

I due scripts, in modo analogo, prendono in input i files "delayValues" e "simParameters" e forniscono come output un grafico che sulle asse delle ascisse riporta il ritardo in slot e sulle ordinate la probabilità di coda. Il valore corrispondente alla ascissa x1 e ordinata y1 è da intendersi come segue:

un pacchetto che entra nella rete ha probabilità y1 di sperimentare un ritardo maggiore di x1.

In figura 1 e 2 sono mostrati dei grafici esempio generati automaticamente dagli scripts.

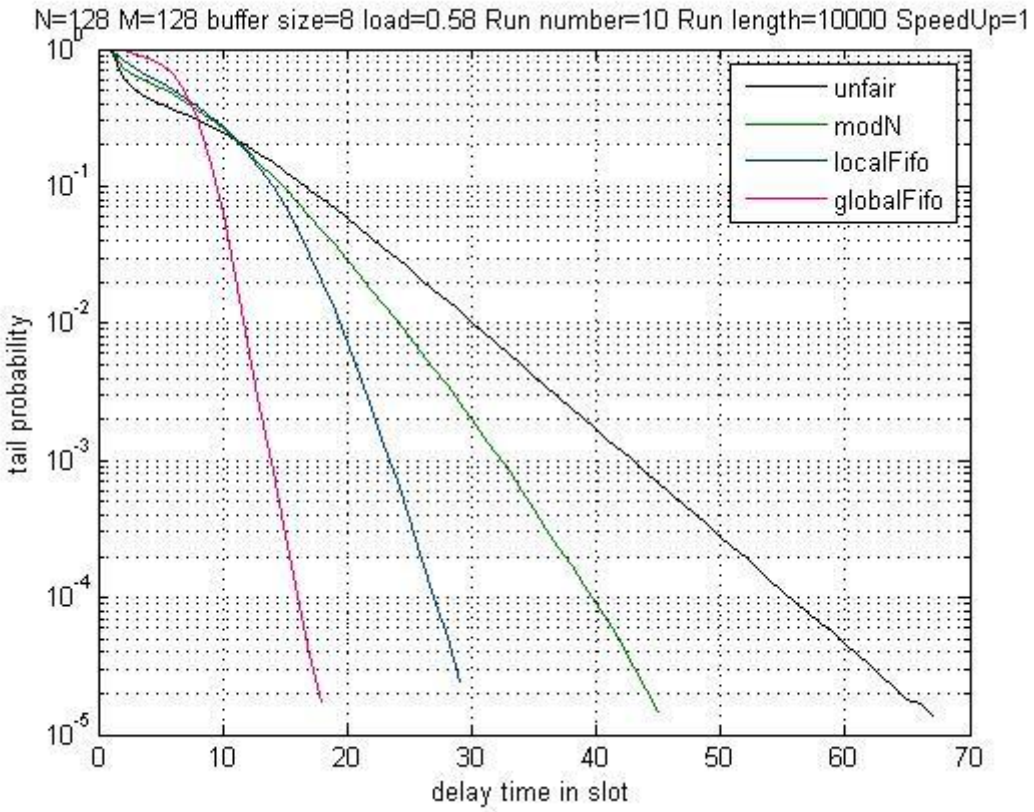


Figura 1 – Grafico generato automaticamente con lo script matlab "distribuzioneRitardi_algBase.m"

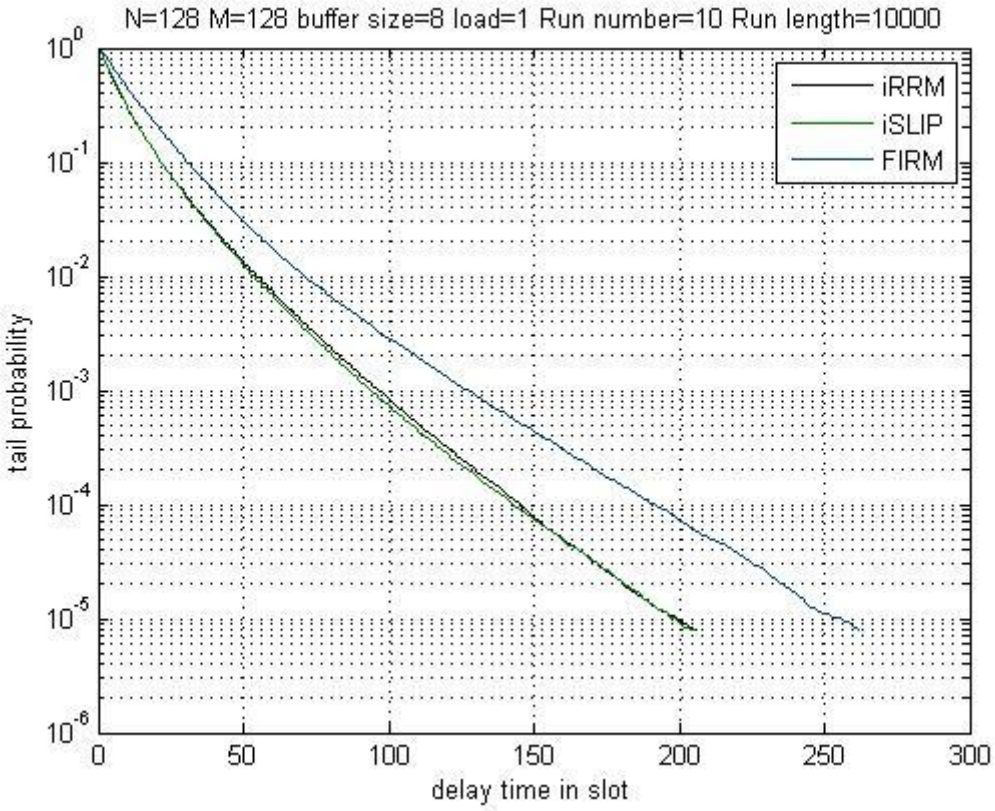


Figura 2 – Grafico generato automaticamente con lo script matlab "distribuzioneRitardi_algAvanzati.m"

Analisi dei risultati

Algoritmi base

Gli schemi di rete classici con buffer in ingresso sono caratterizzati da una coda fifo per ogni ingresso in grado di memorizzare i pacchetti. Ogni time slot è richiesta l'esecuzione di un algoritmo di scheduling per trovare il matching tra un ingresso e una uscita, è chiaro che solo una cella in testa al buffer (in posizione hol) può essere associata ad una determinata uscita ogni time slot. Un problema delle code in ingresso è il così detto "head of line" (HOL) blocking.

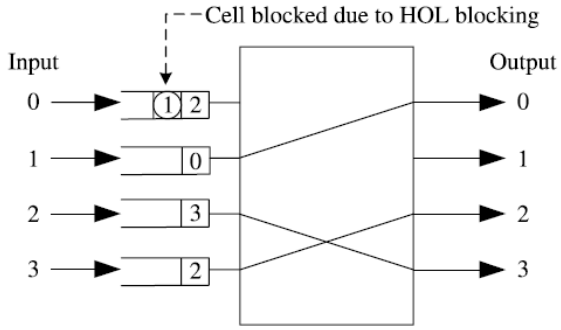


Figura 3 – Hol blocking in una rete con code in input

Un esempio di HOL blocking è mostrato in figura 3, l'ingresso 0 non può trasmettere il suo pacchetto in testa perché l'uscita 2 è già stata assegnata all'ingresso 3 nel time slot considerato. Il pacchetto in coda nell'ingresso 0 verso l'uscita 1 non può essere trasmesso anche se l'uscita 1 nel time slot considerato è libera. Questo comportamento degrada molto il throughput, che non può andare oltre il 58.6%.

Caso 1 – Analisi del throughput

Obiettivo:

Verificare che utilizzando diversi tipi di algoritmi il throughput massimo tende allo stesso valore a causa del noto problema dell' HOL blocking, inoltre verificare che i risultati prodotti siano vicini ai bounds derivanti della teoria.

Nella produzione dei risultanti si sono utilizzati i seguenti parametri di input:

- load: 1.0;
- numero di run di simulazione: 10;
- lunghezza run di simulazione: 10000;
- lunghezza dei buffer in ingresso e buffer di uscita: 8192 (elevata per simulare condizione di buffer infinito);
- accuratezza del transitorio: 1000.

Caso M=N

In tabella 3, sono riportati i valori di throughput massimo raggiunti dagli algoritmi, nel caso N=M, per differenti valori di N (che in questo caso indica sia il numero degli ingressi che il numero delle uscite). L'andamento grafico è mostrato in figura 4.

N	Theorical	unfair	modN	localFifo	globalFifo
2	0.7500	0.7497	0.7513	0.7499	0.7490
4	0.6553	0.6540	0.6563	0.6547	0.6578
8	0.6184	0.6191	0.6193	0.6190	0.6203
16	-	0.6013	0.6022	0.5999	0.6024
32	-	0.5930	0.5927	0.5933	0.5933
64	-	0.5900	0.5894	0.5899	0.5905
128	-	0.5879	0.5880	0.5883	0.5887
256	-	**0.5868**	**0.5869**	**0.5865**	**0.5869**
∞	0.5858	-	-	-	-

Tabella 3 – Throughput massimo, M=N

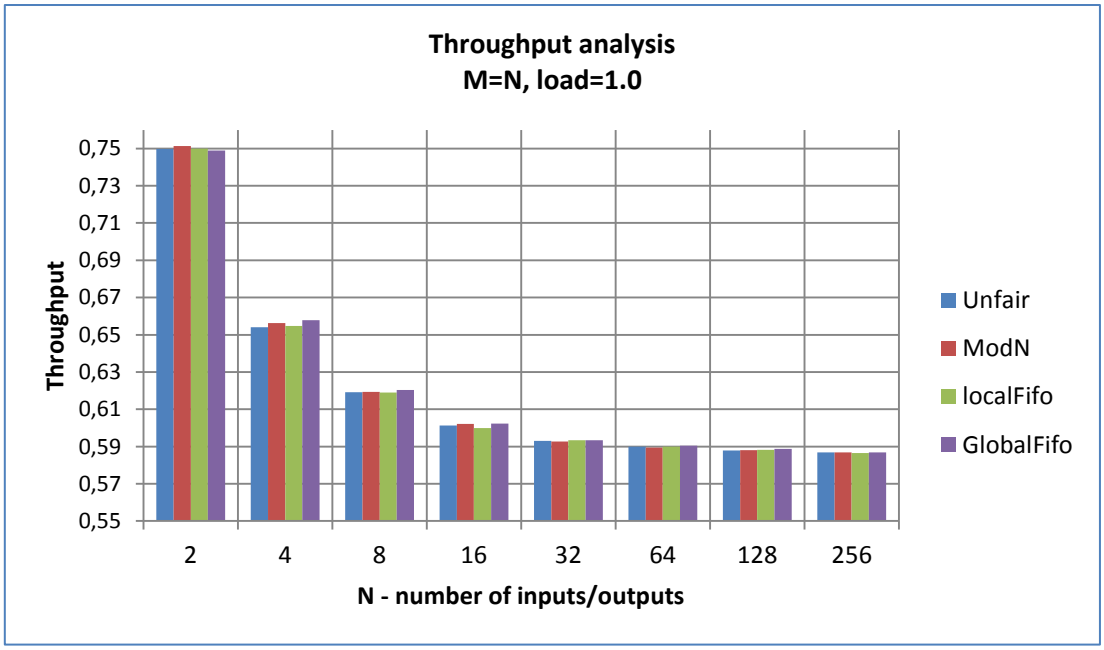


Figura 4 – Andamento del throughput, caso M=N

Caso M>N

In tabella 4, sono riportati i valori di throughput massimo raggiunti dagli algoritmi, fissando il parametro M e variando N. L'andamento grafico è mostrato in figura 5.

M	N	Theorical	unfair	modN	localFifo	globalFifo
256	**256**	0.586	0.5868	0.5869	0.5865	0.5869
256	**128**	0.764	0.7653	0.7656	0.7659	0.7660
256	**64**	0.877	0.8784	0.8786	0.8791	0.8789
256	**32**	0.938	0.9400	0.9400	0.9400	0.9401
256	**16**	0.969	0.9708	0.9708	0.9709	0.9712
256	**8**	0.984	0.9868	0.9866	0.9862	0.9864

Tabella 4 – Throughput massimo, M fissato e N variabile

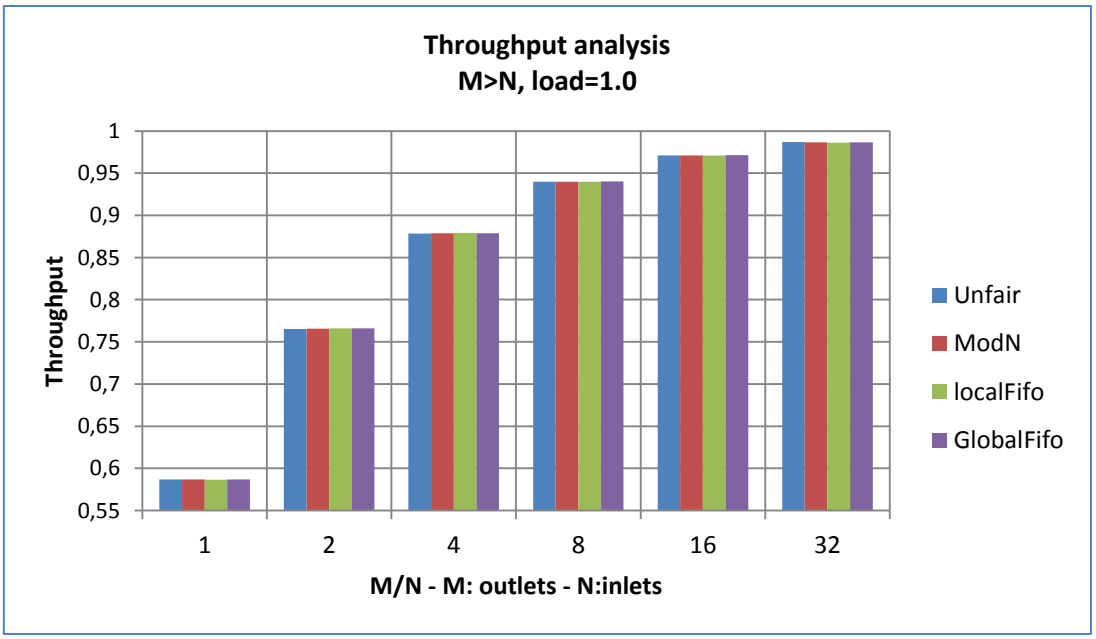


Figura 5 – Andamento del throughput, caso M>N

Caso M<N

In tabella 5, sono riportati i valori di throughput massimo raggiunti dagli algoritmi, fissando il parametro N e variando M. L'andamento grafico è mostrato in figura 6.

M	N	Theorical	unfair	modN	localFifo	globalFifo
256	**256**	0.586	0.5868	0.5869	0.5865	0.5869
128	**256**	0.382	0.3827	0.3825	0.3832	0.3833
64	**256**	0.219	0.2198	0.2196	0.2198	0.2198
32	**256**	0.117	0.1178	0.1174	0.1173	0.1173
16	**256**	0.061	0.0608	0.0606	0.0606	0.0606
8	**256**	0.031	0.0310	0.0308	0.0308	0.0308

Tabella 5 – Throughput massimo, N fissato e M variabile

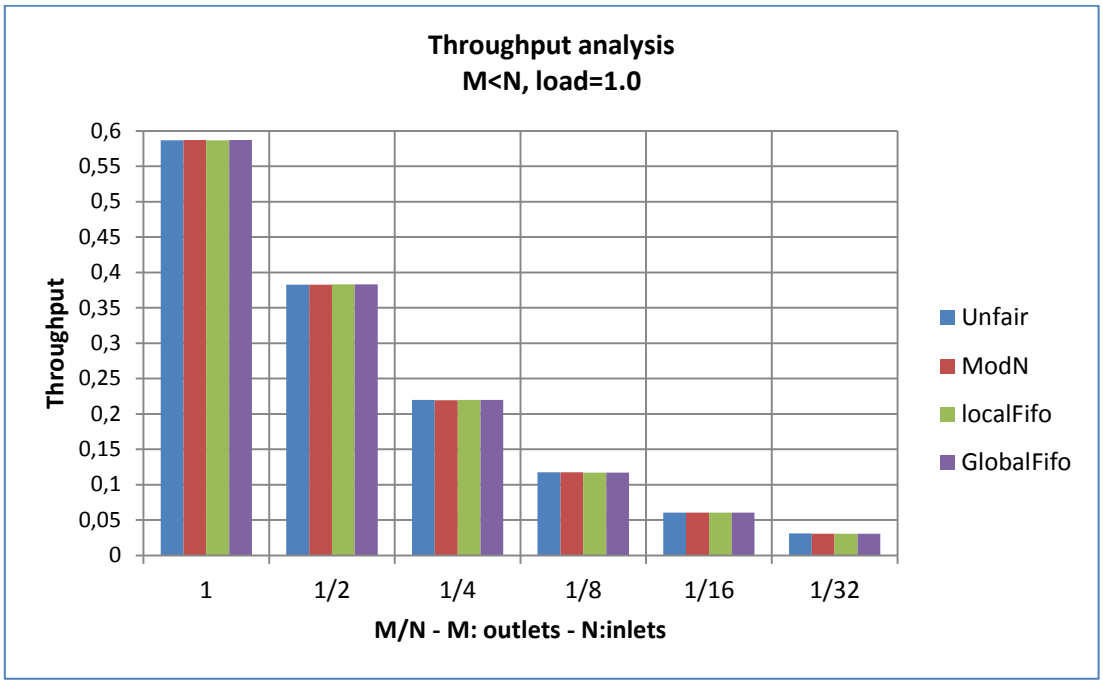


Figura 6 – Andamento del throughput, caso M<N

Caso 2 – Analisi della probabilità di perdita

Nel caso dell'analisi del throughput si è mostrato che i differenti algoritmi base hanno un comportamento molto simile. Le differenze tra gli algoritmi emergono nell'analisi dei pacchetti persi. Come ci si può aspettare gli algoritmi che garantiscono una politica fifo hanno prestazioni migliori sotto questo punto di vista. E' interessante notare che, mentre l'algoritmo global fifo ha prestazioni migliori in maniera indipendente dalla dimensione dei buffer in ingresso, dalle simulazioni si nota che l'algoritmo modN superata una certa dimensione del buffer ottiene prestazioni migliori del fifo locale.

In tabella 6 è mostrato l'andamento della probabilità di perdita al variare del carico, per i differenti algoritmi base nel caso in cui la lunghezza del buffer in ingresso sia limitata a 8 slot. L'andamento grafico è mostrato in figura 7.

Load	Unfair	ModN	localFifo	GlobalFifo
0.4	0.000072	0.000026	0.000004	-
0.5	0.002387	0.001086	0.000369	-
0.52	0.003628	0.001876	0.000781	0.000001
0.58	0.00838	0.005871	0.004064	0.000332
0.6	0.010148	0.007591	0.005773	0.002441

Tabella 6 – Probabilità di perdita, M=N=128, input buffer length=8

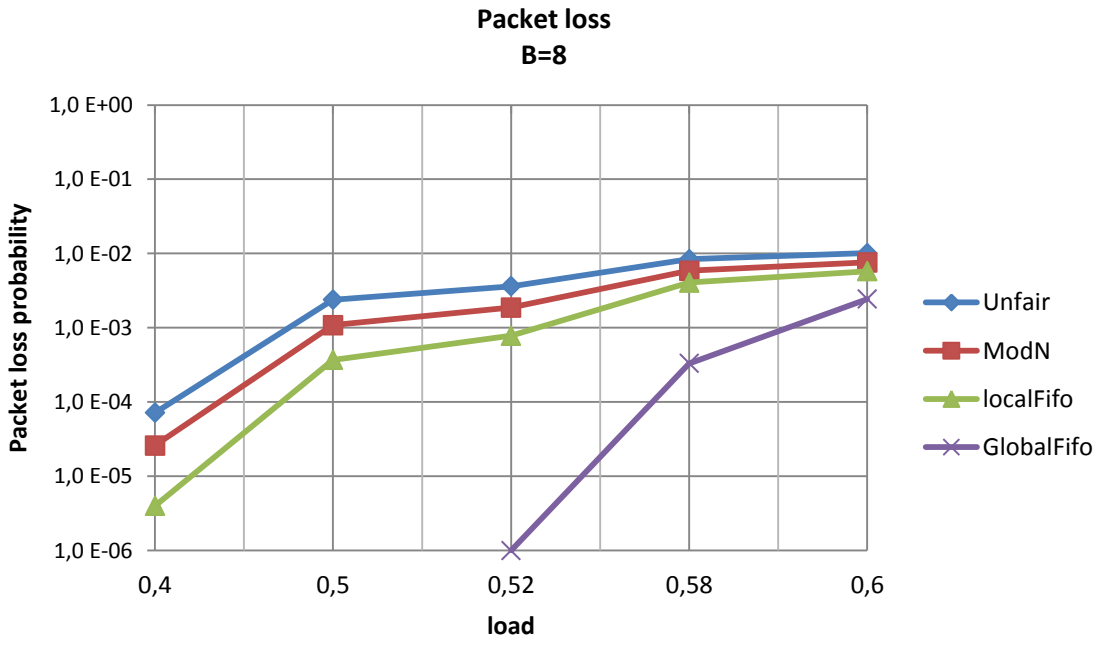


Figura 7 – Comparazione probabilità di perdita al variare del carico, M=N=128, B=8

In tabella 7 è mostrato l'andamento della probabilità di perdita al variare del carico, per i differenti algoritmi base nel caso in cui la lunghezza del buffer in ingresso sia limitata a 32 slot. L'andamento grafico è mostrato in figura 8. Sin da questo valore è possibile notare l'aumento di prestazioni dell'algoritmo modN.

Load	Unfair	ModN	localFifo	GlobalFifo
0.5	0.001403	-	-	-
0.55	0.005067	0.000003	0.000568	-
0.588	0.008473	0.00109	0.0033	0.000019
0.59	0.008733	0.001343	0.003502	0.000357
0.6	0.009571	0.002473	0.004525	0.002017

Tabella 7 – Probabilità di perdita, M=N=128, input buffer length=32

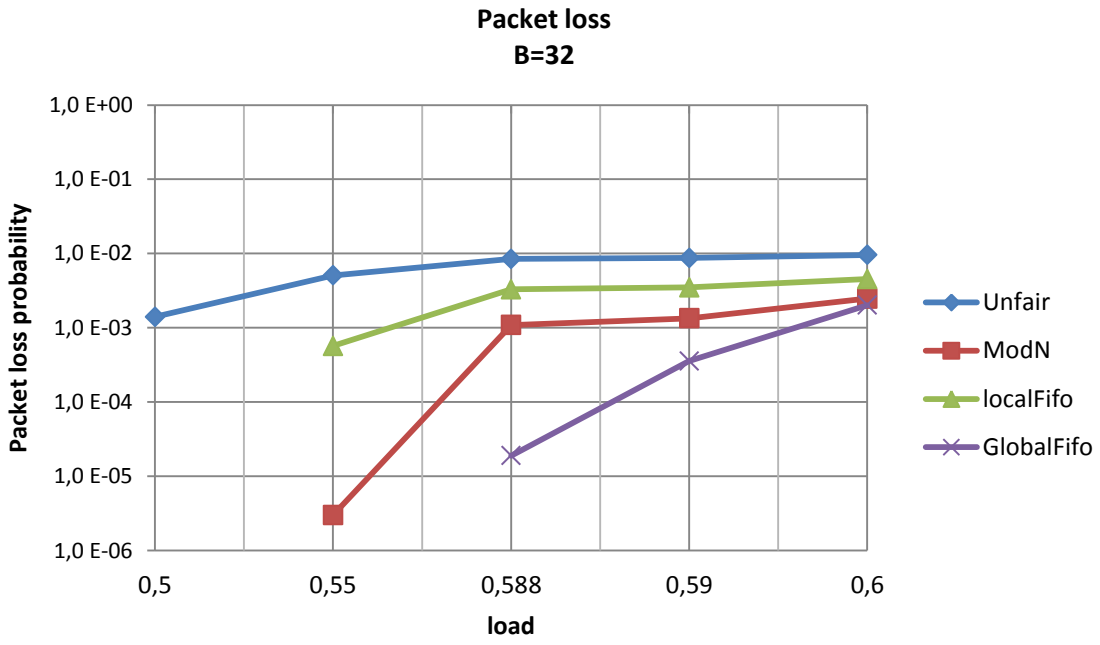


Figura 8 – Comparazione probabilità di perdita al variare del carico, M=N=128, B=32

In tabella 8 è mostrato l'andamento della probabilità di perdita al variare del carico, per i differenti algoritmi base nel caso in cui la lunghezza del buffer in ingresso sia limitata a 256 slot. L'andamento grafico è mostrato in figura 9.

Load	Unfair	ModN	localFifo	GlobalFifo
0.465	0.000047	-	-	-
0.54	0.004096	-	0.000076	-
0.588	0.00829	0.000045	0.003029	-
0.591	0.008558	0.000346	0.003306	0.000098
0.6	0.009439	0.001715	0.004231	0.001662

Tabella 8 – Probabilità di perdita, M=N=128, input buffer length=256

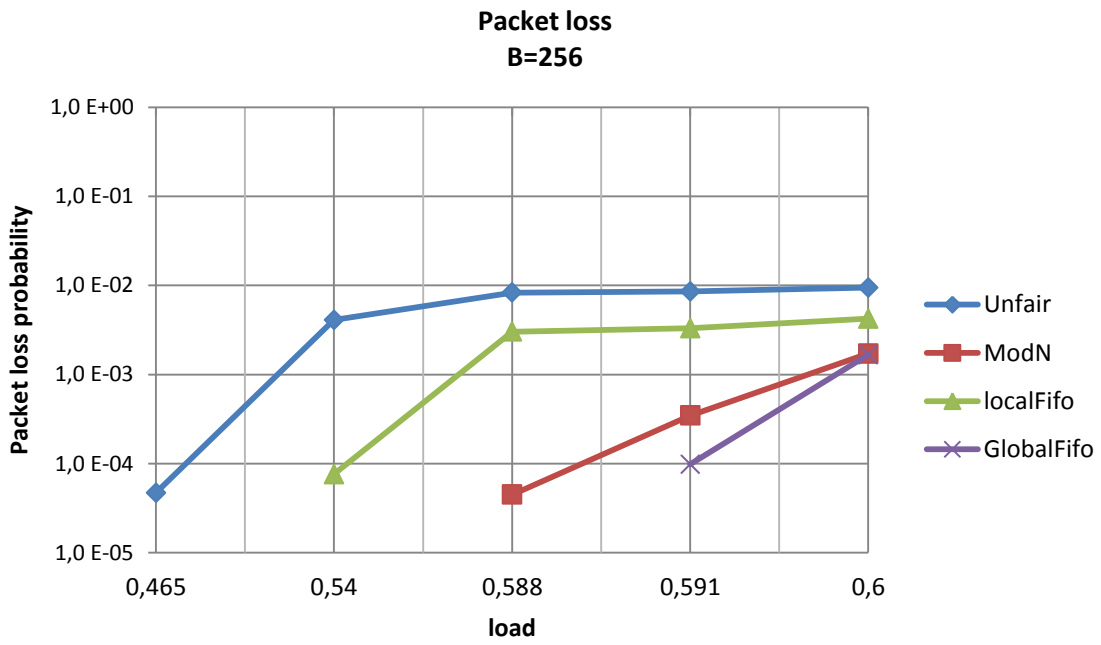


Figura 9 – Comparazione probabilità di perdita al variare del carico, M=N=128, B=256

In tabella 9 è mostrato l'andamento della probabilità di perdita al variare del carico, per i differenti algoritmi base nel caso in cui la lunghezza del buffer in ingresso sia limitata a 1024 slot. L'andamento grafico è mostrato in figura 10.

Load	Unfair	ModN	localFifo	GlobalFifo
0.465	0.000047	-	-	-
0.54	0.004096	-	0.000076	-
0.588	0.00829	0.000045	0.003029	-
0.591	0.008558	0.000346	0.003306	0.000098
0.6	0.009439	0.001715	0.004231	0.001662

Tabella 9 – Probabilità di perdita, M=N=128, input buffer length=1024

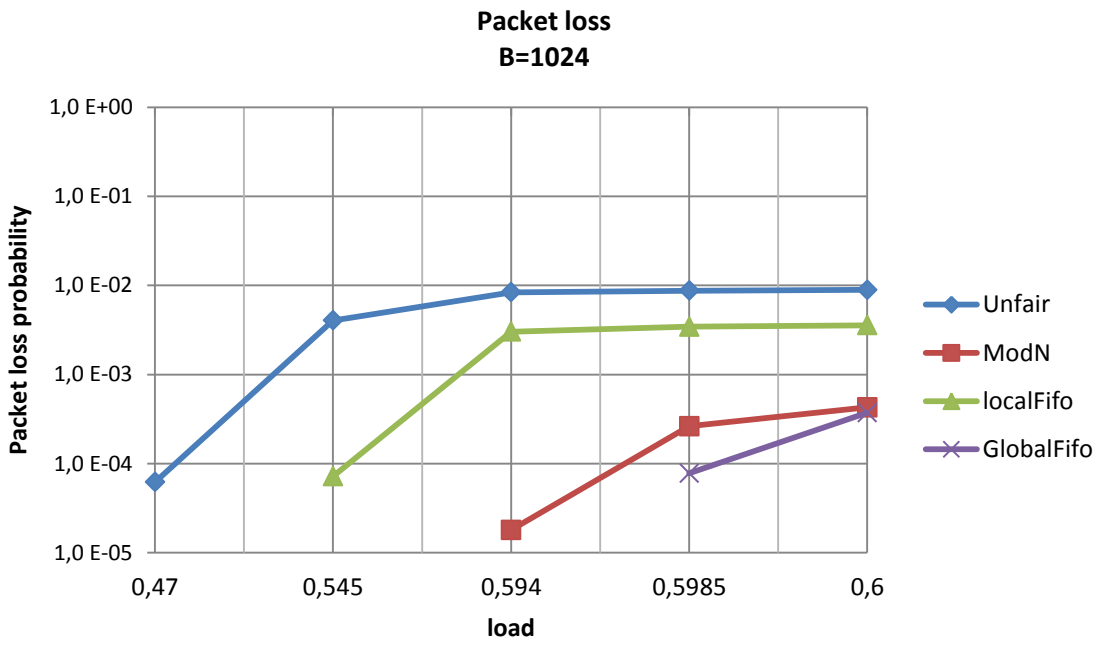


Figura 10 – Comparazione probabilità di perdita al variare del carico, M=N=128, B=1024

Caso 3 – Analisi del ritardo medio

In modo analogo all'analisi della probabilità di perdita, l'analisi del ritardo medio di permanenza di un pacchetto nel buffer di ingresso evidenzia le differenze tra i vari algoritmi base.

In tabella 10 sono mostrati gli andamenti del ritardo medio di permanenza nel buffer di ingresso al variare del carico. Anche in questo caso si può notare la bontà degli algoritmi modN e globalFifo. I valori in rosso indicano simulazioni in cui si è verificata perdita di pacchetti. L'andamento grafico è in figura 11.

load	Unfair	modN	localFifo	globalFifo
0.40	2	2	1.8	1.6
0.42	2.3	2.1	2	1.7
0.44	3	2.4	2.2	1.8
0.46	12	2.8	2.5	2
0.48	185	3.5	2.9	2.2
0.50	406	4.4	3.7	2.4
0.52	635	5.7	5.7	2.8
0.54	825	7.9	67.8	3.3
0.56	1003	11.8	446.1	4.2
0.57	1101	16.2	673.4	5
0.58	1177	31.6	898.2	6.9
0.59	1249	282	1109.9	194
0.60	1329	1061	1300	1042

Tabella 10 – Ritardo medio nel buffer di ingresso, M=N=128, input buffer length=2048

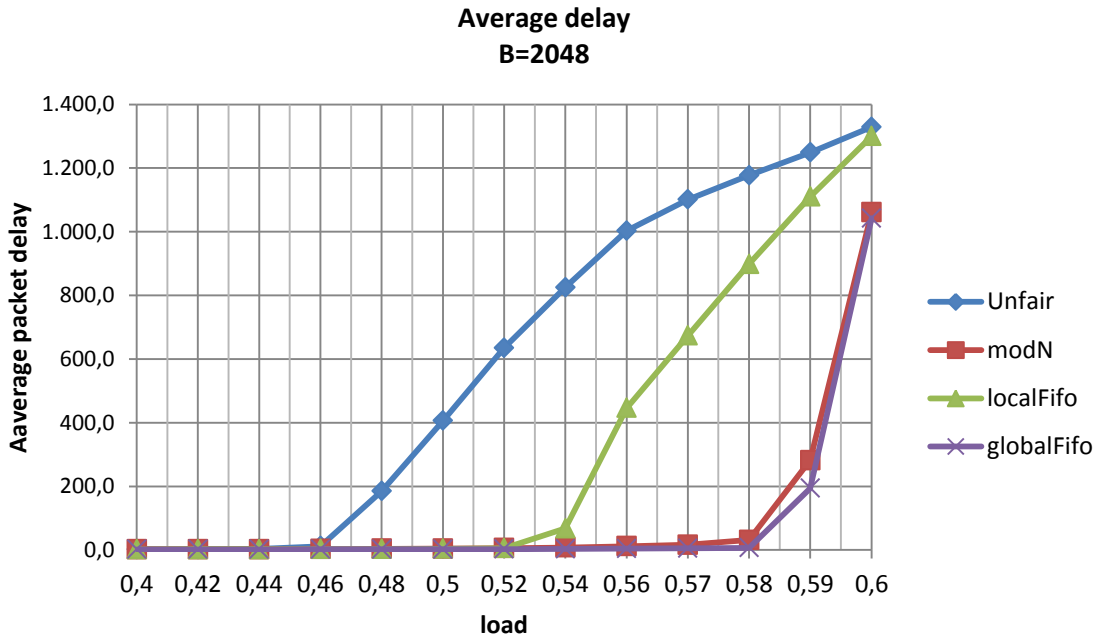


Figura 11 – Comparazione del ritardo medio al variare del carico, M=N=128, B=2048

Caso 4 – Output speed-up

Come evidenziato dal paragrafo sull'analisi del throughput, gli algoritmi base raggiungono throughput massimo pari a circa il 58%. Per superare questo problema una della possibilità è quella di ricorrere all'utilizzo dello speed-up in uscita, ovvero permettere a più ingressi di trasmettere verso una singola uscita nello stesso tempo di slot. Una volta che i K pacchetti (ove K è il fattore di speed-up) hanno raggiunto lo stadio di uscita si possono adottare le seguenti politiche:

- utilizzare una linea di uscita con capacità K-volte superiore a quella da un singolo ingresso;
- mantenere la stessa capacità del link ma utilizzare dei buffers su ogni uscita per evitare la perdita di pacchetti.

E' da notare che utilizzare una delle due metodologie sopra descritte non varia i risultati dal punto di vista del throughput.

In tabella 11 sono riportarti i valori di throughput ottenuti variando il fattore di speed-up per i differenti algoritmi. L'andamento grafico è mostrato in figura 12.

K	Theorical	unfair	modN	localFifo	globalFifo
1	0.586	0.5877	0.5878	0.5875	0.5876
2	0.885	0.8861	0.8863	0.8862	0.8862
4	0.996	0.9947	0.9947	0.9947	0.9947
8	1.000	0.9976	0.9976	0.9976	0.9976

Tabella 11 – Throughput con speed-up in uscita, M=N=128, bufferInput=1024

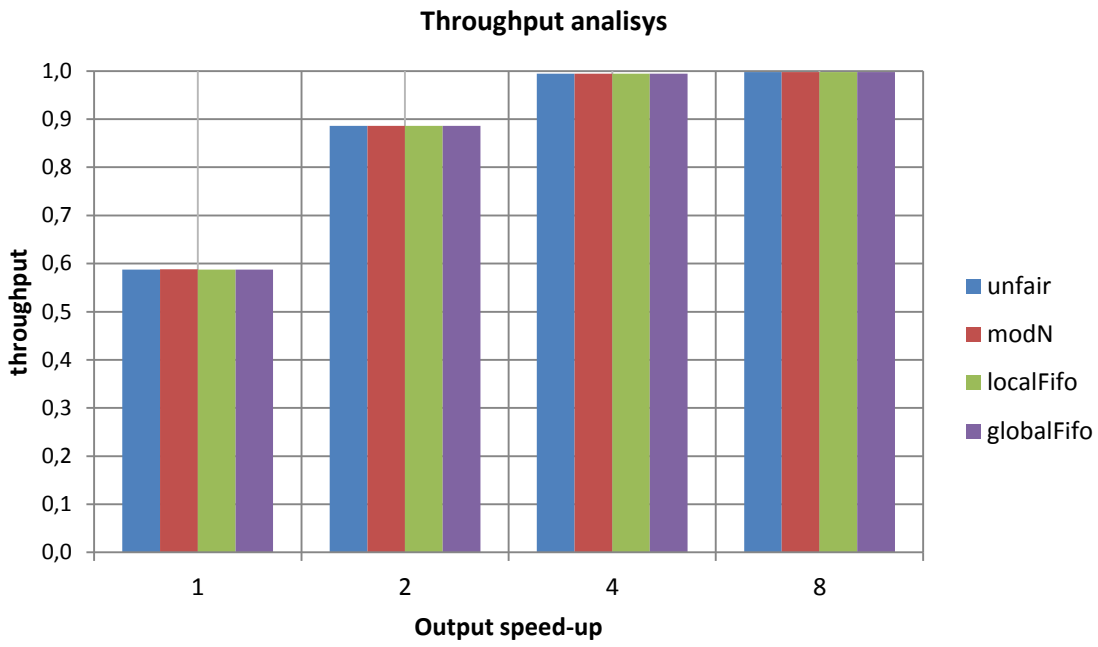


Figura 12 – Throughput massimo al variare del fattore di speed-up

Nel caso in cui si addotti la politica dei buffer in uscita è interessante notare la parte di ritardo generata dall'accodamento nei buffers di ogni uscita. In tabella 12 sono riportati i dati medi del ritardo sperimentato da un pacchetto nel buffer di uscita in caso di valori diversi di speed-up. L'andamento grafico è mostrato in figura 13.

K	unfair	modN	localFifo	globalFifo
1	0	0	0	0
2	3.73	3.73	3.75	3.74
4	90.33	90.62	91.03	90.35
8	164.88	163.84	163.83	163.84

Tabella 12 – Ritardo medio nel buffer di output in funzione di K, M=N=128, bufferInput=1024, bufferOutput=8192

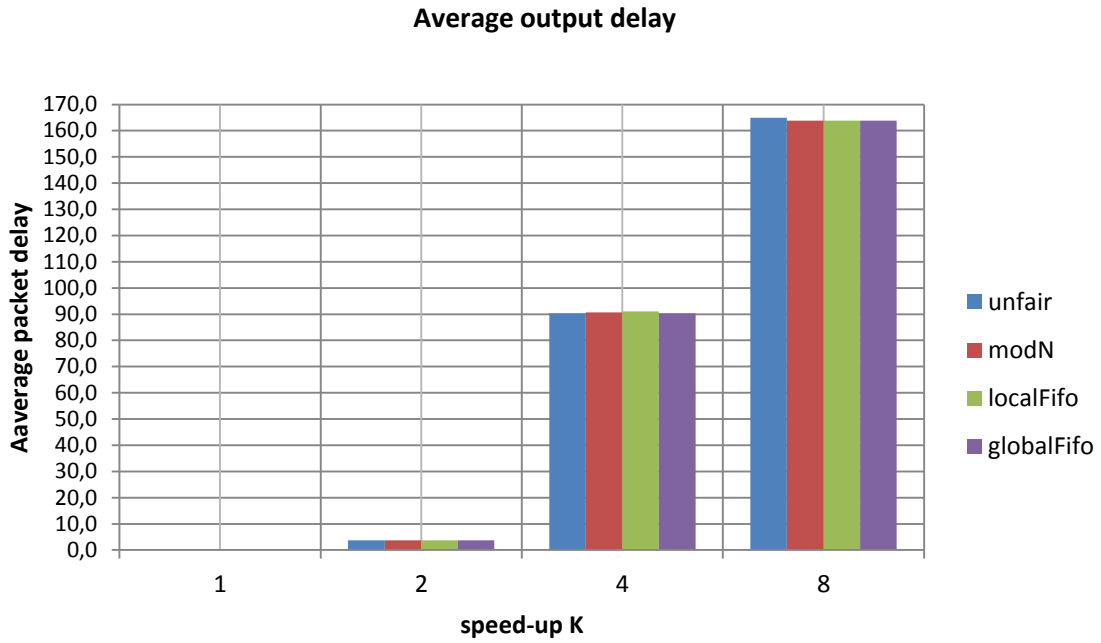


Figura 13 – Ritardo medio di permanenza nel buffer di uscita al variare del fattore di speed-up

Algoritmi avanzati

A causa del problema dell'Hol Blocking, come evidenziato dal capitolo sugli algoritmi base, il throughput è limitato al 58,6% in caso di traffico uniforme. La limitazione del throughput risiede nella struttura FIFO delle code, per risolvere tale problema è largamente utilizzata la tecnica delle code virtuali in uscita (VOQ), vedi figura 14. Si considera che in ogni ingresso ci siano N code virtuali, ognuna corrispondente ad un uscita, in totale ci sono N^2 code FIFO. In altre parole i pacchetti che arrivano dall'ingresso i e destinati per una uscita j sono memorizzati nella coda $VOQ_{i,j}$. Le celle in posizione hol di ogni VOQ devono essere schedulate per la trasmissione in ogni time slot. Comunque, al massimo una cella di una delle N VOQ di un ingresso può essere selezionata per la trasmissione.

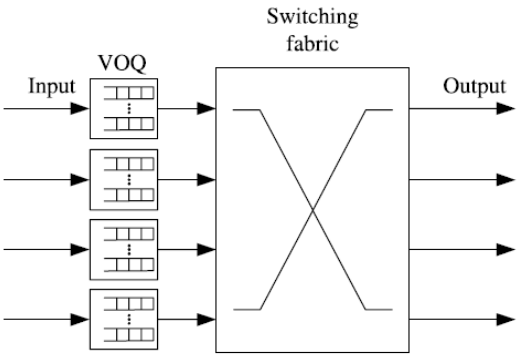


Figura 14 – Struttura VOQ

I differenti criteri con cui sono schedulate le celle in posizione hol delle N^2 code virtuali costituiscono l'algoritmo di scheduling.

Gli algoritmi di scheduling avanzati implementati nel programma di simulazione sono i seguenti:

- iRRM;
- iSLIP;
- FIRM.

I tre algoritmi avanzati possono essere descritti mediante tre fasi:

- *Request:* ogni ingresso invia una richiesta a tutte le uscite per cui ha pacchetti in coda.
- *Grant:* se un'uscita riceve più richieste, sceglie quella più vicina in base ad uno scheduler di tipo round-robin. L'uscita notifica ad ogni ingresso se è stato concesso il grant o meno. A questo punto viene aggiornato il grant pointer in modalità differenti a seconda dell'algoritmo.
- *Accept:* Se un ingresso riceve più grant da uscite diverse, essa accetta quella più vicina al suo scheduler round-robin. Analogamente alla fase di grant, l'accept pointer è aggiornato in maniera diversa da ogni algoritmo.

La principale differenza tra gli algoritmi è la tecnica di aggiornamento degli indici dei puntatori agli scheduler round-robin. Tali differenze portano alle seguenti conclusioni:

- iRRM e iSLIP hanno bisogno di iterare fino a N volte in ogni time slot le tre fasi sopra viste per raggiungere throughput unitario.
- FIRM raggiunge throughput unitario anche con una sola iterazione.

Caso 1 – Analisi del throughput

Obiettivo:

Verificare che gli algoritmi avanzati non soffrono dell'HOL blocking. Verificare che il throughput maggiore è raggiunto dall'algoritmo iRRM e dall'algoritmo iSLIP. FIRM può avere un throughput minore perché cerca di implementare una politica FCFS.

In tabella 13 sono riportati i valori di throughput massimo ottenibile con gli algoritmi di scheduling avanzati iRRM, iSLIP e FIRM. L'andamento grafico è mostrato in figura 15.

N	iRRM	iSLIP	FIRM
2	0.997	0.998	0.997
4	0.986	0.977	0.995
8	0.995	0.981	0.993
16	0.994	0.986	0.99
32	0.994	0.989	0.984
64	0.994	0.99	0.97
128	0.994	0.991	0.941

Tabella 13 – Throughput massimo al variare di M=N

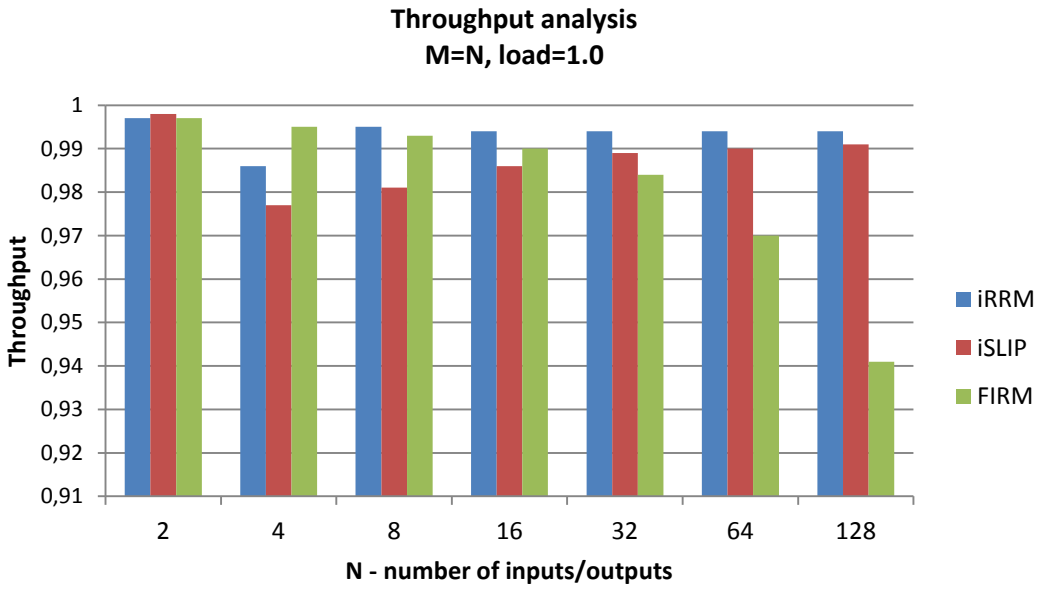


Figura 15 – Grafico comparativo del throughput massimo per differenti algoritmi

Caso 2 – Analisi della probabilità di perdita

In tabella 14 è mostrato l'andamento della probabilità di perdita al variare del carico, per i differenti algoritmi avanzati nel caso in cui la lunghezza del buffer in ingresso sia limitata a 8 slot. L'andamento grafico è mostrato in figura 16.

load	iRRM	iSLIP	FIRM
0.55	-	-	0.000013
0.7	-	-	0.013708
0.8	0.000032	0.000031	0.023880
0.9	0.003076	0.003211	0.032254
1.0	0.011015	0.011163	0.039220

Tabella 14 – Probabilità di perdita, M=N=128, input buffer length=8

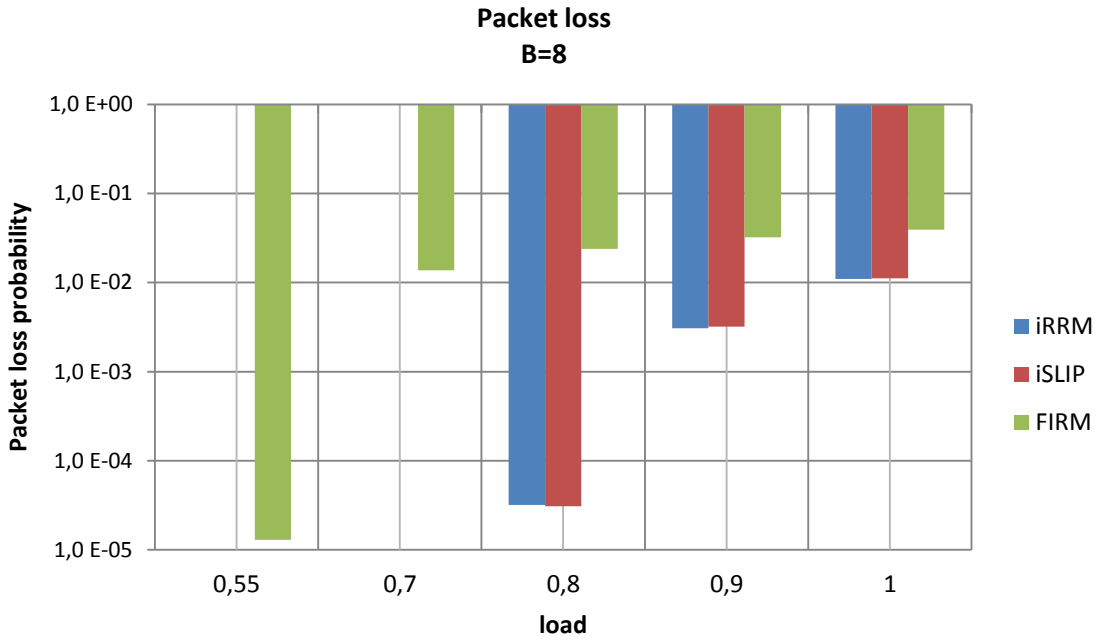


Figura 16 – Comparazione probabilità di perdita al variare del carico, M=N=128, B=8

In tabella 15 è mostrato l'andamento della probabilità di perdita al variare del carico, per i differenti algoritmi avanzati nel caso in cui la lunghezza del buffer in ingresso sia limitata a 32 slot. L'andamento grafico è mostrato in figura 17.

load	iRRM	iSLIP	FIRM
0.6	-	-	0.000126
0.9	-	-	0.030576
0.95	0.000010	0.000033	0.034245
1.0	0.003538	0.003664	0.037535

Tabella 15 – Probabilità di perdita, M=N=128, input buffer length=32

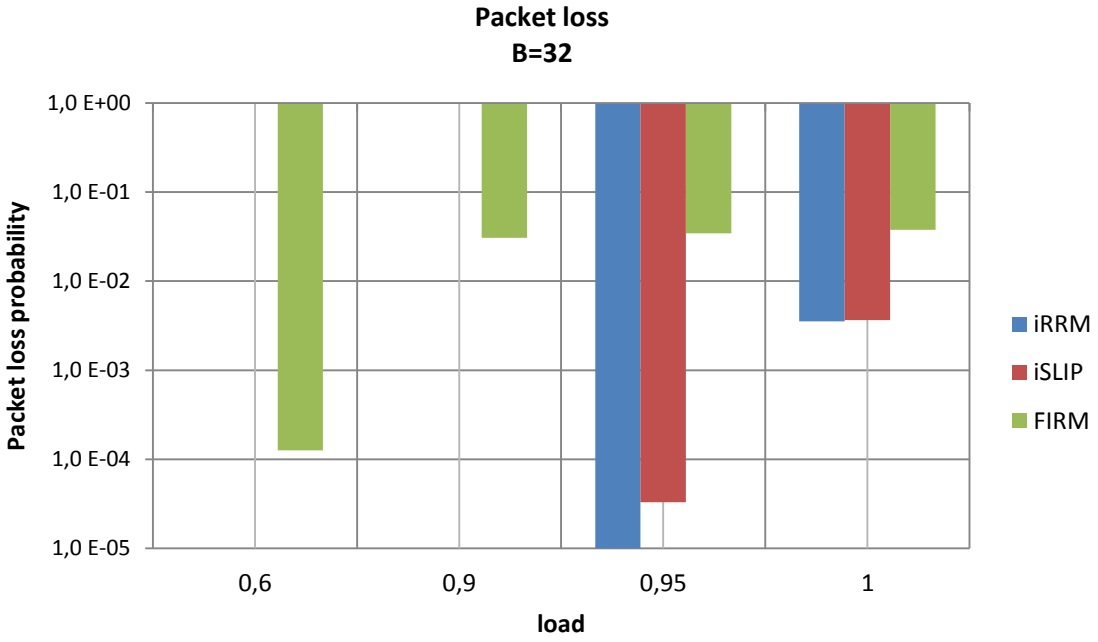


Figura 17 – Comparazione probabilità di perdita al variare del carico, M=N=128, B=32

In tabella 16 è mostrato l'andamento della probabilità di perdita al variare del carico, per i differenti algoritmi avanzati nel caso in cui la lunghezza del buffer in ingresso sia limitata a 256 slot. L'andamento grafico è mostrato in figura 18.

load	iRRM	iSLIP	FIRM
0.77	-	-	0.000015
0.98	-	0.000009	0.010809
0.99	0.000046	0.000379	0.018907
1.0	0.000572	0.001523	0.019651

Tabella 16 – Probabilità di perdita, M=N=128, input buffer length=256

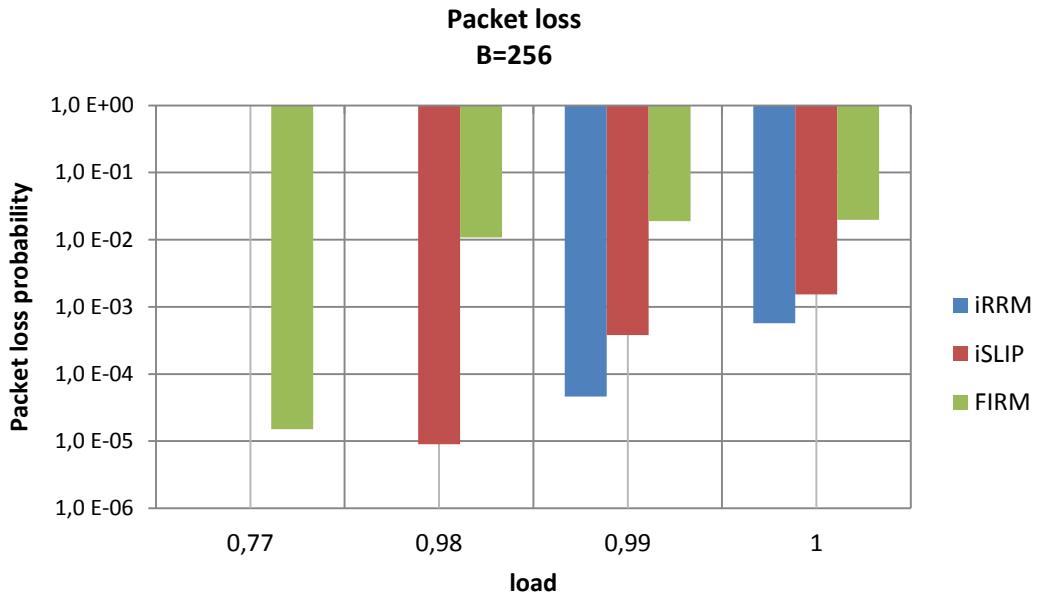


Figura 18 – Comparazione probabilità di perdita al variare del carico, M=N=128, B=256

In tabella 17 è mostrato l'andamento della probabilità di perdita al variare del carico, per i differenti algoritmi avanzati nel caso in cui la lunghezza del buffer in ingresso sia limitata a 1024 slot. L'andamento grafico è mostrato in figura 19.

load	iRRM	iSLIP	FIRM
0.95	-	-	0.000423
0.99	-	0.000102	0.003945
1.0	0.000090	0.000554	0.004836

Tabella 17 – Probabilità di perdita, M=N=128, input buffer length=1024

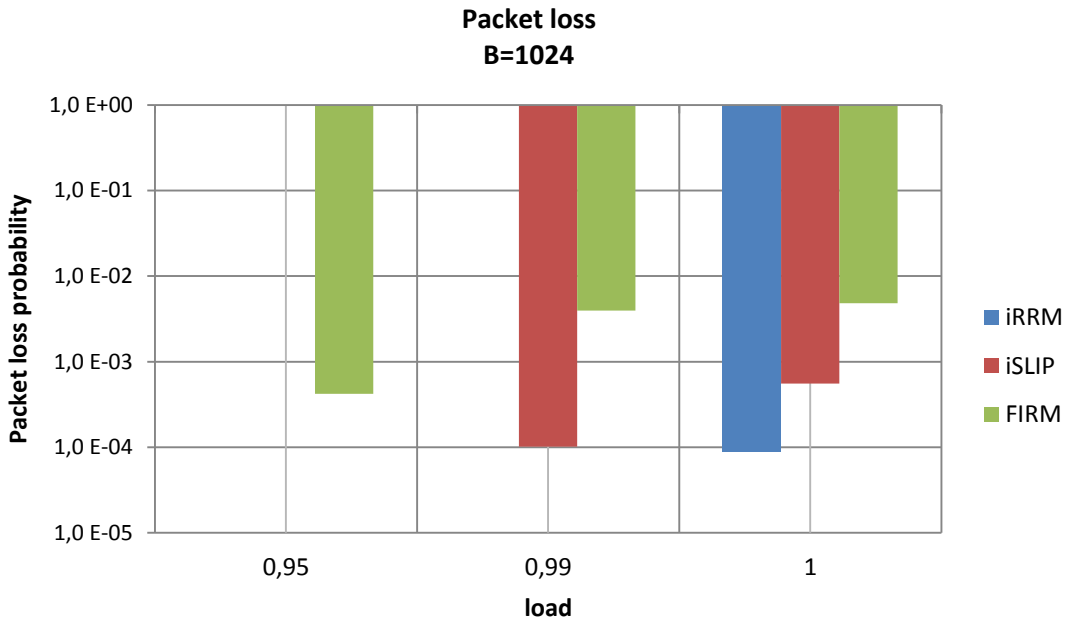


Figura 19 – Comparazione probabilità di perdita al variare del carico, M=N=128, B=1024

Caso 3 – Ritardo medio

In tabella 18 sono mostrati gli andamenti del ritardo medio di permanenza nel buffer di ingresso al variare del carico. I valori in rosso indicano simulazioni in cui si è verificata perdita di pacchetti. L'andamento grafico è in figura 20.

load	iRRM	iSLIP	FIRM
0.5	1.754	1.755	2.714
0.6	2.220	2.222	38.342
0.7	3.057	3.062	180.885
0.8	4.819	4.844	312.918
0.9	10.457	10.540	596.231
1.0	353.446	273.656	1689.164

Tabella 18 – Ritardo medio nel buffer di ingresso, M=N=128, input buffer length=2048

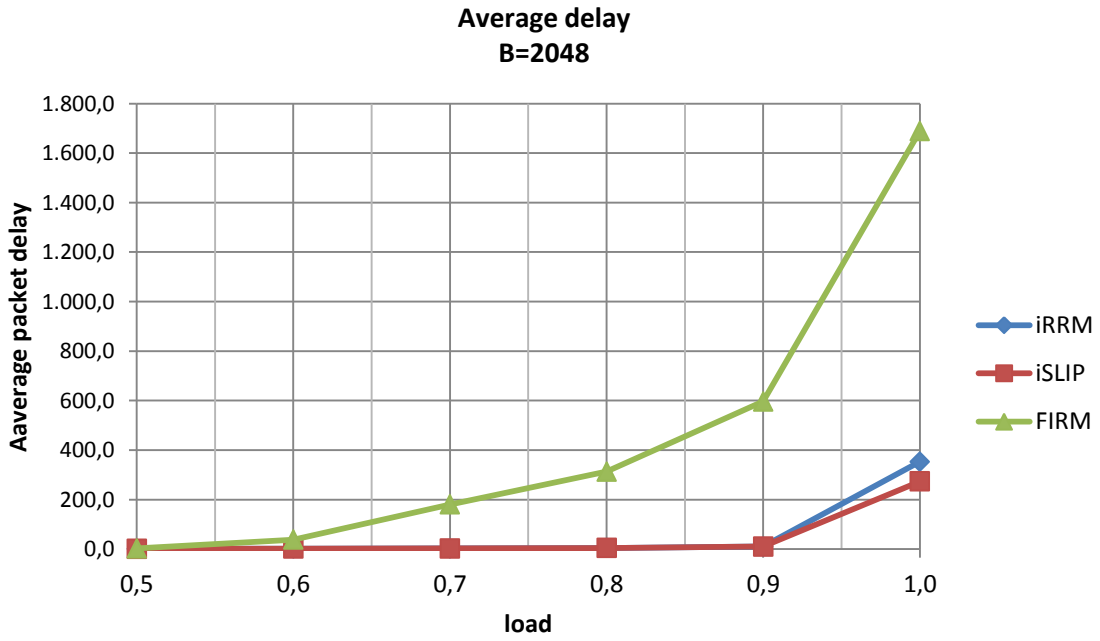


Figura 20 – Comparazione del ritardo medio al variare del carico, M=N=128, B=2048

Caso 4 – Numero medio di iterazioni

In tabella 19 sono riportati i dati inerenti il numero medio di iterazioni necessarie ad ogni slot di tempo per utilizzare al meglio gli algoritmi iRRM e iSLIP in funzione del numero di ingressi della rete. L'andamento grafico è mostrato in figura 21.

N	iRRM	iSLIP
2	2.00	2.00
4	3.10	3.35
8	4.11	4.67
16	4.98	6.20
32	5.00	7.76
64	6.46	9.06
128	6.92	9.76
256	7.17	9.91

Tabella 19 – Numero medio di iterazione per slot in funzione di M=N

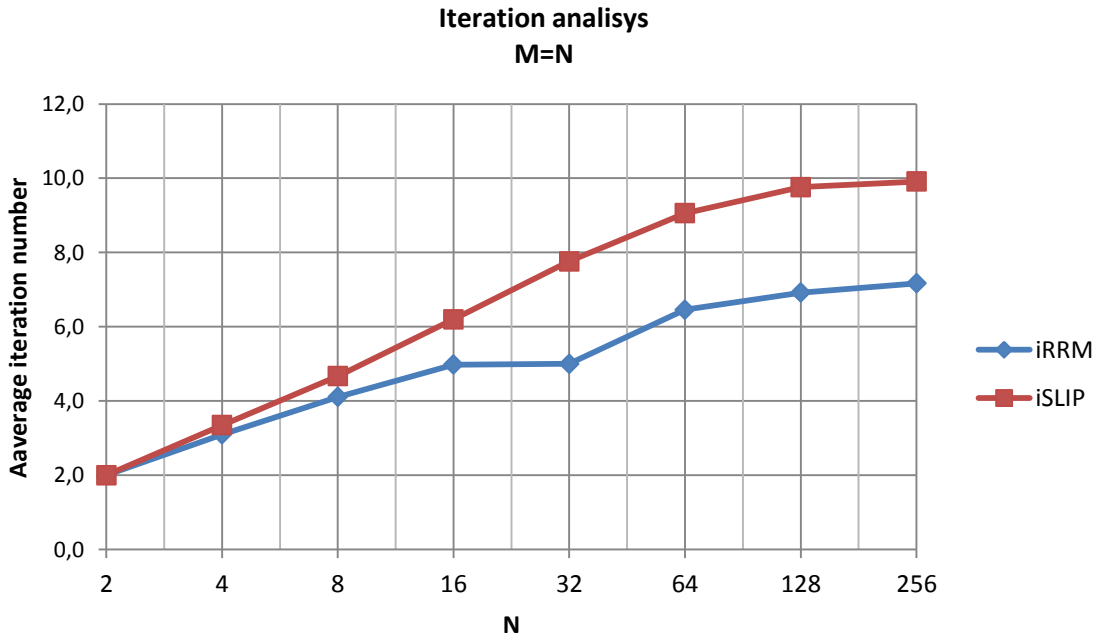


Figura 21 – Comparazione numero medio di iterazioni per iRRM e iSLIP

In tabella 20 sono riportati i dati inerenti il numero medio di iterazioni necessarie ad ogni slot di tempo per utilizzare al meglio gli algoritmi iRRM e iSLIP in funzione del carico in ingresso. L'andamento grafico è mostrato in figura 22.

load	iRRM	iSLIP
0.5	2.98	2.98
0.6	3.09	3.09
0.7	3.38	3.41
0.8	3.89	3.92
0.9	4.38	4.48
1.0	6.92	9.76

Tabella 20 – Numero medio di iterazioni per slot, M=N=128, carico variabile

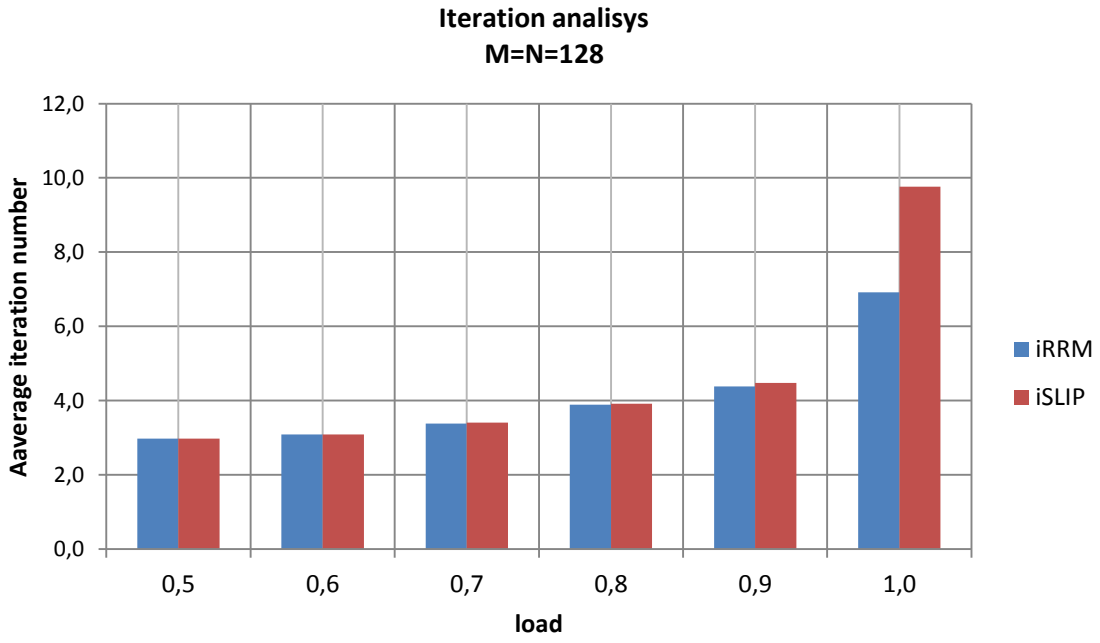


Figura 22 – Comparazione numero medio di iterazioni per slot, M=N=128, carico variabile

Riferimenti

- Il codice del simulatore è stato sviluppato partendo da un simulatore denominato “G/G/1 QUEUE SIMULATION PROGRAM” fornito dall’Ing. Massimo Tornatore.

Libri di testo:

- H.J. Chao, B. Liu, High Performance Switches and Routers, John Wiley & Sons, 2007.
- A. Pattavina, Switching Theory, Architectures and Performance in Broadband ATM Networks, John Wiley & Sons, 1998.
- P. Deitel, H. Deitel, C++ How to Program, 7/E, Deitel, 2010.