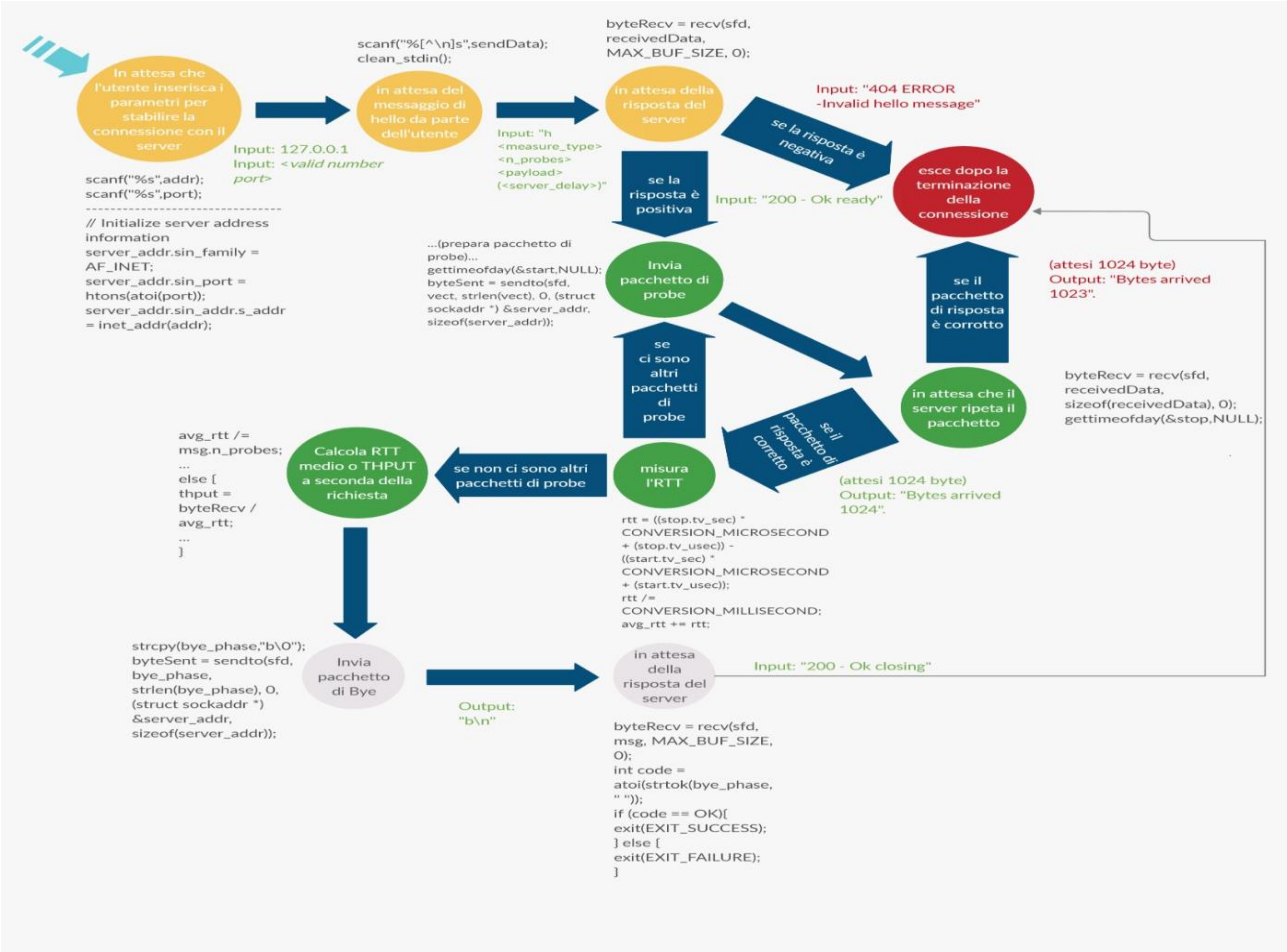
TASK 1

FSM Client



Il client per prima cosa prende in input l'indirizzo ip e la porta del server.

Una volta stabilità la connessione TCP il client invia il messaggio di Hello adottando il seguente formato: "<*protocol_phase*> <*sp*> <*measure_type*> <*sp*> <*n_probes*> <*sp*> <*msg_size*> <*sp*> <*server_delay*>\n".

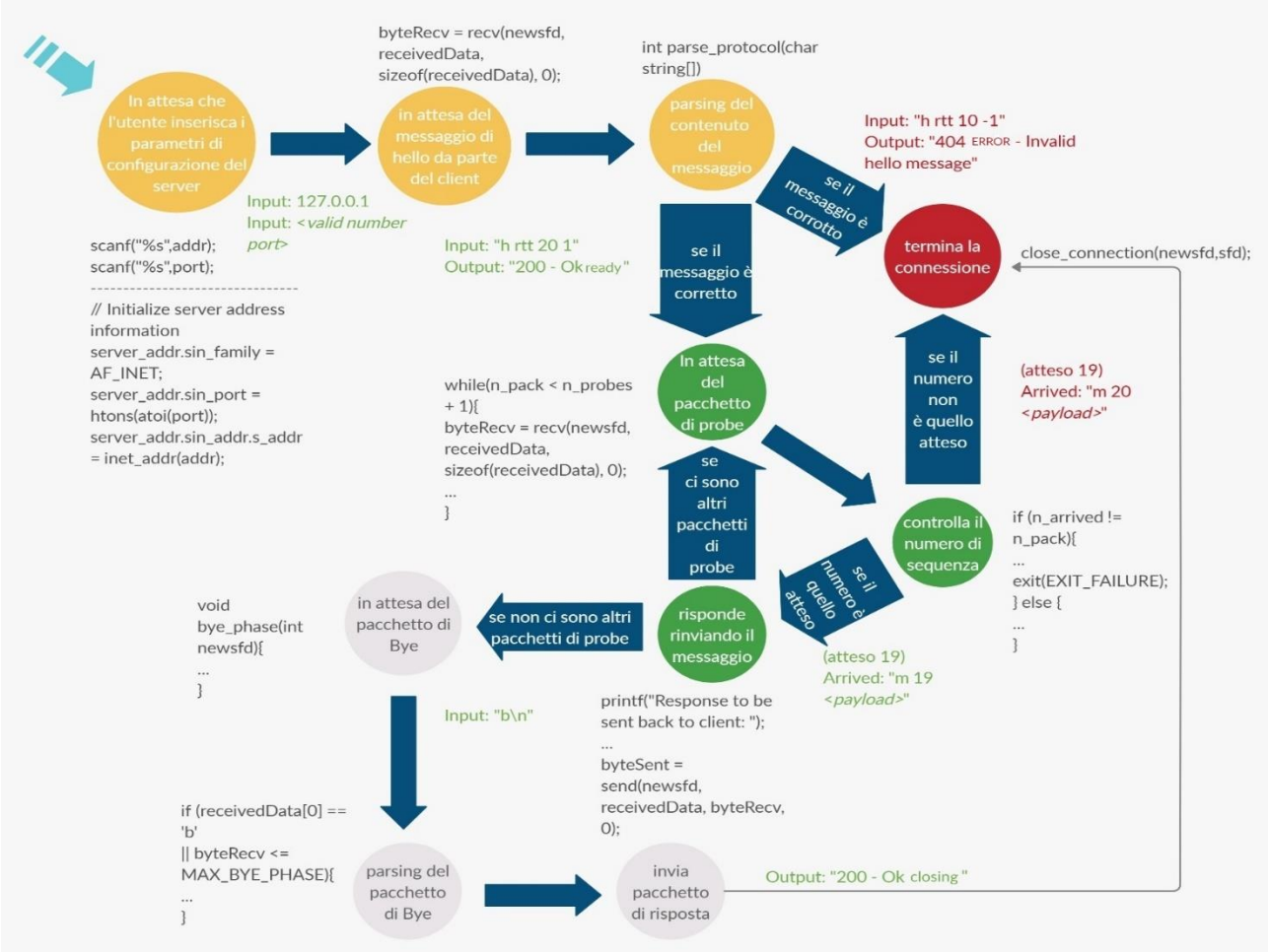
Completata l'Hello phase si avvia la fase di measurement: il client inizia ad inviare dei messaggi di probe al server per fare le misure richieste ovvero calcolare l'RTT medio o il throughput del canale fra loro.

Il client invia uno specifico numero di messaggi al server con un numero di sequenza incrementale che parte da 1, per il messaggio di probe è adottato il formato seguente:" <*protocol_phase*> <*sp*> <*probe_seq_num*> <*sp*> <*payload*>\n".

Qualora la connessione non fosse stata terminata a causa di un errore nella fase precedente il client può dare avvio alla fase finale ovvero quella di bye: viene inviato una richiesta di terminazione al server e si attende risposta. Una volta ricevuta questa il client termina la connessione TCP.

Il messaggio di bye: "<*protocol_phase*>\n".

FSM Server



Il server riceve in input l'indirizzo ip e la porta.

Una volta ricevuto il messaggio di Hello il server controlla e convalida quanto ricevuto.

Se il messaggio è valido il server risponde con un messaggio di testo contenente il codice di conferma permettendo al client di procedere alla fase successiva, mentre in caso contrario viene inviato un codice di errore e la connessione terminata.

In fase di measurement, il server riceve i messaggi dal client, controlla il numero di sequenza di ogni pacchetto e il numero di probe inviate e risponde ogni volta rinviando al mittente il messaggio ricevuto. Qualora arrivi un messaggio di probe errato il server chiude la comunicazione e la connessione dopo aver inviato un messaggio di errore al client.

Giunto alla bye phase, il server controlla che il formato dell'ultimo pacchetto sia corretto e provvede a comunicare la chiusura della connessione al client.

TASK 2

STRATEGIA RISOLUTIVA

Sia Client che Server prendono in input i parametri necessari per stabilire la connessione TCP (indirizzo ip locale 127.0.0.1 e la porta del server).

Il messaggio di Hello che dà avvio alla comunicazione, secondo il protocollo implementato, viene inserito attraverso standard input e inviato al server, che provvederà poi al parsing del messaggio. I vari parametri del messaggio, funzionali alla comunicazione, vengono registrati in una struttura dati appositamente implementata, la quale ha facilitato le operazioni della measurement phase. Infatti quest'ultima fase si è concretizzata in un ciclo all'interno del quale vengono spediti tanti messaggi quanto indicato nel parametro < n_probes > del messaggio di Hello, valore reperito da un campo della struttura. Cosa simile avviene per la misurazione del Throughput/RTT: quale risultato ricavare al termine della measurement phase è anch'esso indicato nel messaggio di Hello e memorizzato in un campo della struttura. L'RTT è stato calcolato come la differenza fra il momento in cui viene ricevuto dal client il messaggio replicato dal server e l'istante in cui il client invia il messaggio di probe: per memorizzare l'orario di invio dei vari messaggi si è utilizzata la struttura dati timeval, che si avvale sia del secondo che del microsecondo come unità di tempo, e la funzione gettimeofday(); il risultato è stato poi convertito in millisecondi.

Al termine della measurement phase, si è calcolato l'RTT medio o il Throughput a seconda della richiesta: l'RTT è stato ottenuto come la media aritmetica degli RTT dei singoli pacchetti, mentre il Throughput è il risultato del rapporto tra il numero massimo di byte inviati e l'RTT medio; i risultati sono stati poi loggati su file.

In seguito all'invio dell'ultimo pacchetto di probe, l'ultimo messaggio fra client e server è quello di bye e a questo segue la terminazione della connessione: qualora al server fosse stato inviato un messaggio corrotto in una delle fasi precedenti all'ultima ed essa avesse risposto con un messaggio di errore, la terminazione della connessione sarebbe stata anticipata.

Ogni fase è stata divisa in funzioni separate in modo da isolare e separare le singole operazioni.

Per quanto riguarda il server, anche per questo si è effettuata la divisione delle fasi in funzioni: in particolare sono stati divisi i controlli effettuati sui vari messaggi in modo che l'attività di parsing si adatti alla determinata fase senza appesantire il codice e che venga facilitata la manutenzione di quest'ultimo. A due funzioni separate è stato poi deputato il compito di inviare i messaggi di risposta al client al termine delle fasi, gestendo con un maggiore ordine la replica del server, effettuata una volta ultimate le operazioni di checking sui messaggi del client.

La nostra implementazione del server ha gestito anche il caso in cui il messaggio di Hello contenga il parametro di delay (che deve essere maggiore o uguale a 0, pena l'interruzione della comunicazione), e che i messaggi di probe arrivino nell'ordine corretto (ovvero con numero di sequenza in ordine crescente) e che non eccedano la soglia indicata nel parametro < n_probes > nel pacchetto di Hello (soglia che non deve essere comunque inferiore a 20 probes).

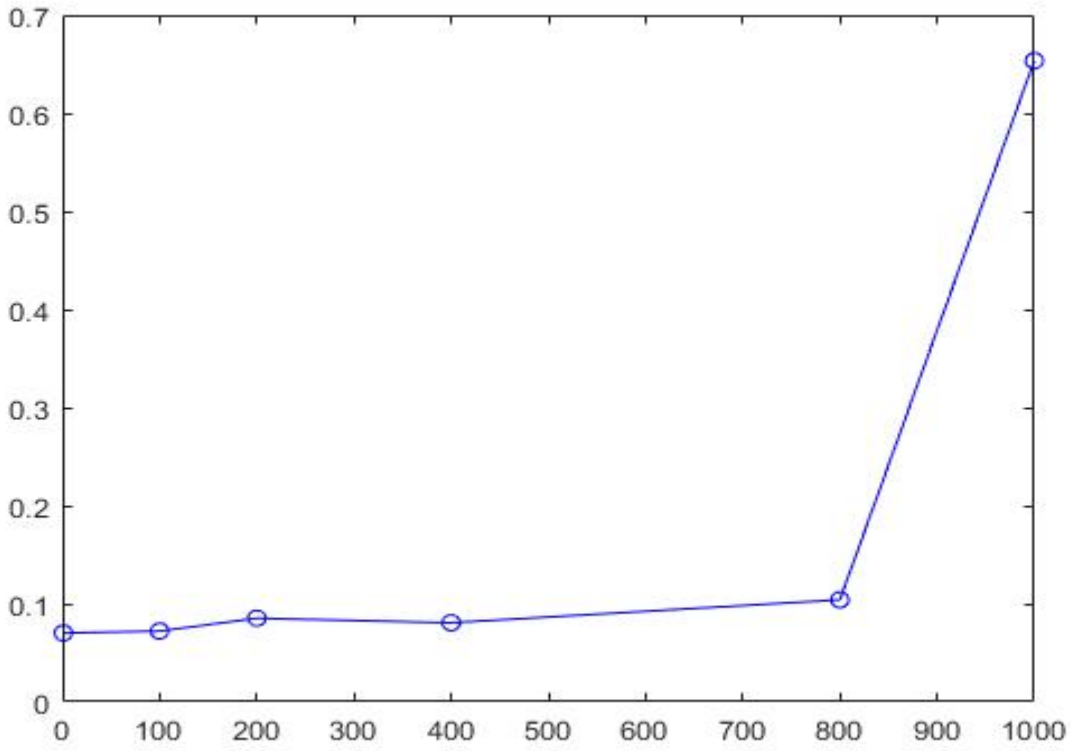
Nel caso in cui il delay sia specificato, il server attende su una sleep() per il quantitativo di tempo indicato nel parametro e poi invia la replica del messaggio al client.

GRAFICI

Abbiamo misurato l'RTT inviando pacchetti di grandezza: 1, 100, 200, 400, 800, 1000 byte. L'unità di misura è il millisecondo.

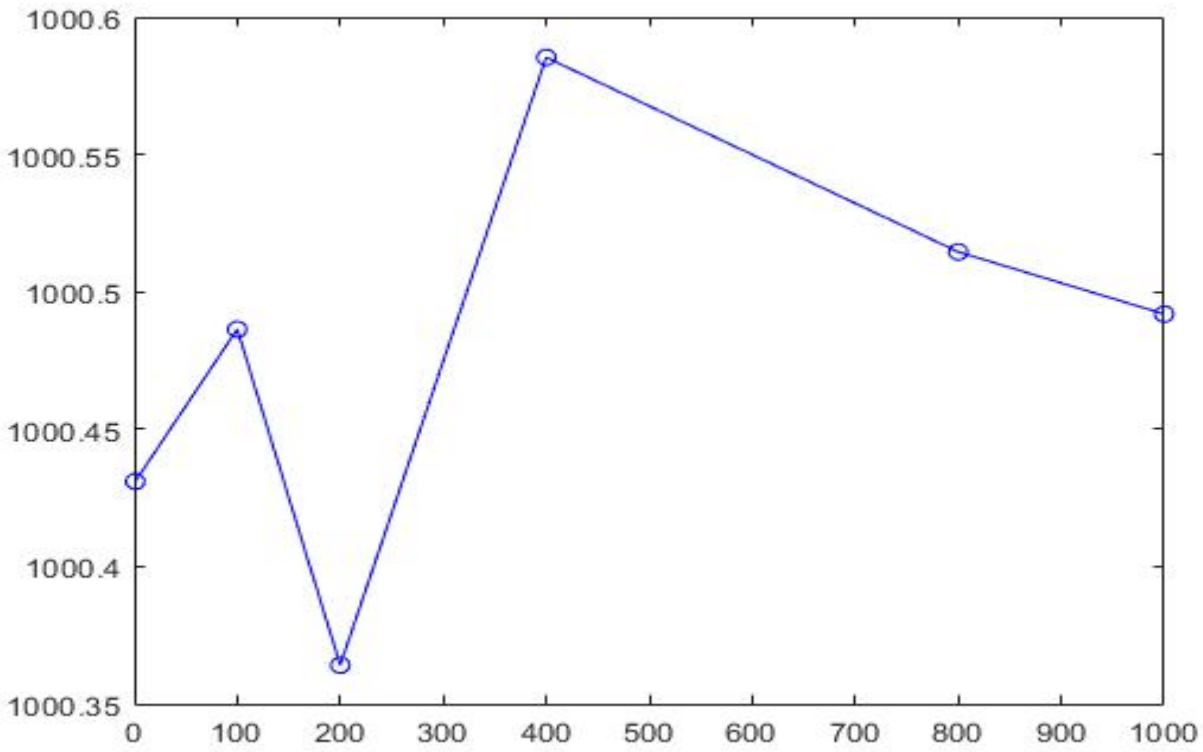
Il Throughput è stato calcolato con pacchetti di: 1K, 2K, 4K, 16K, 32K byte. L'unità di misura è il KB/s.

RTT MEDIO SENZA SERVER DELAY



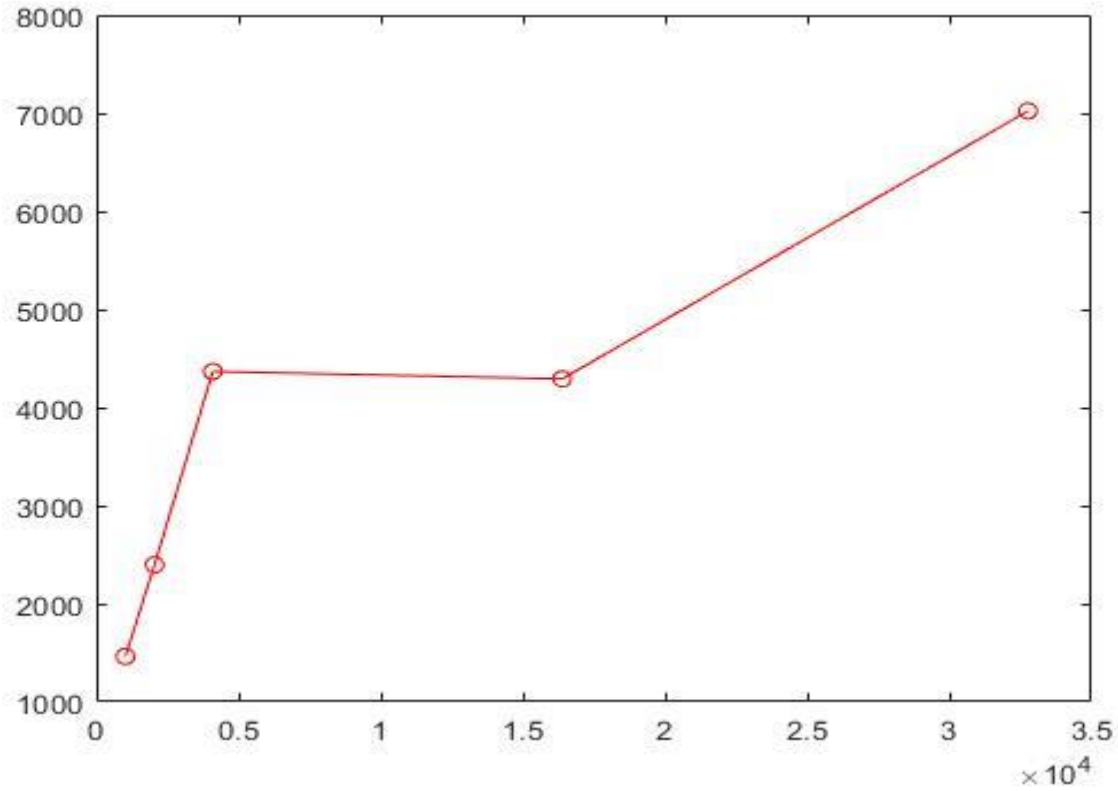
Il grafico sopra fa riferimento ad una misura effettuata in assenza di server delay: è possibile infatti notare come l'RTT non superi il secondo ma cresca comunque nell'intervallo 0-0.7 con l'aumentare della dimensione del pacchetto di probe inviato.

RTT MEDIO CON SERVER DELAY DI 1 SECONDO



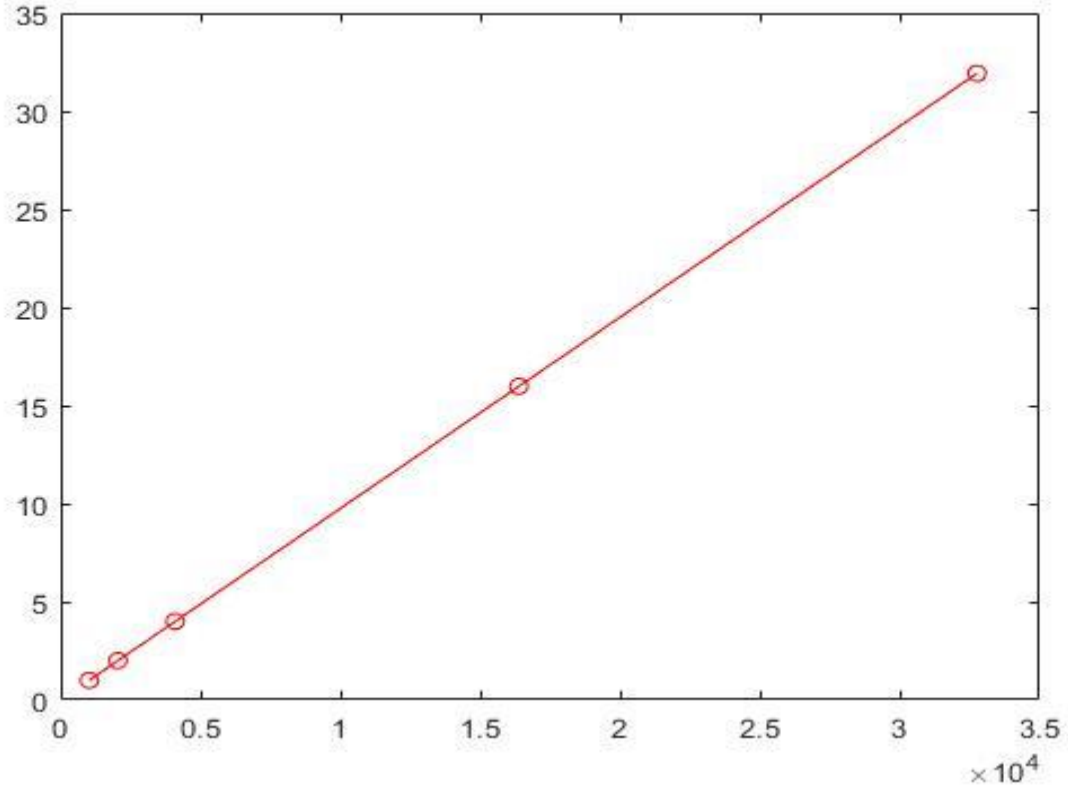
La presenza del server delay influenza l'RTT come si può notare dal grafico: infatti abbiamo valori che non scendono mai sotto al secondo, tempo per cui il server attende prima di inviare il pacchetto di risposta nella nostra rilevazione.

THROUGHPUT SENZA SERVER DELAY



Nel grafico soprastante, il throughput è stato misurato con un server delay pari a 0: l'aumentare delle dimensioni dei probe inviati, a fronte di un RTT medio molto piccolo, fa sì che il throughput cresca proporzionalmente al numeratore del rapporto e che lo faccia considerevolmente (al di sopra infatti dei 1000KB/s).

THROUGHPUT CON SERVER DELAY DI 1 SECONDO



Nel grafico soprastante, il throughput è stato misurato con un server delay di 1 secondo: questo ha comportato una crescita sempre in funzione dell'aumento del numero di bytes scambiati fra client e server, comunque inferiore alla situazione di assenza di server delay, perché maggiore è l'RTT medio (denominatore nel rapporto).