Il progetto di Laboratorio di Reti è stato sviluppato, scritto e testato su macchine **Intel x86** con sistema **Ubuntu Linux** sulle macchine del **Laboratorio Ercolani**.

Sono stati realizzati i due Load Balancer rispettivamente rappresentati nei seguenti file :

- **Load Balancer Mobile** : gestore della rete Wireless
- **Load Balancer Fixed** : gestore della rete Wired

Per realizzarli sono state affrontate varie fasi implementative, affrontando le problematiche inerenti alla progettazione: nelle prime si sono viste l'analisi del comportamento delle applicazioni e i diversi tipi di connessione, mentre nelle successive si sono realizzate le strutture dati e gli algoritmi in grado di fronteggiare tutti i problemi posti dal realtà simulata dal Monitor.
In particolare è stato riscontrato un bug nelle applicazioni (AppMobile e AppFixed) sull'invio e sulla ricezione dei pacchetti.
Infatti i pacchetti voce sono stati implementati come dei vettori di 100 interi aventi dimensione 400 byte (*sizeof(uint32_t buf[100])*), mentre la dimensione passata come parametro alle funzioni di lettura e scrittura (*readn*, *sendn*) era una costante di 100 byte (*PKTSIZE*).

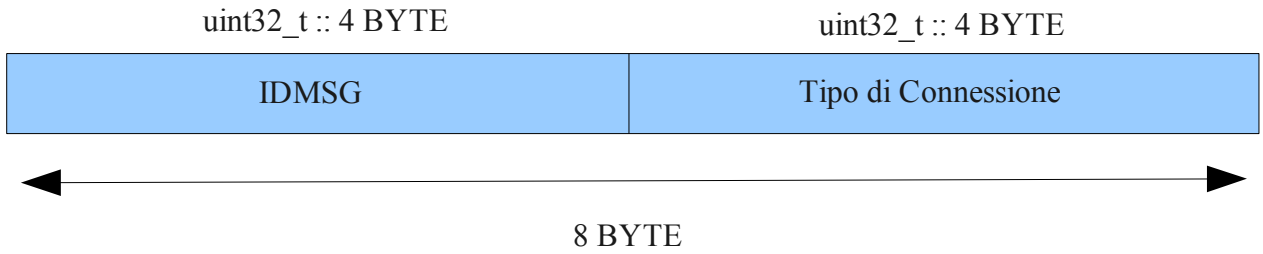
Entrambi i file sono suddivisi a loro volta in tre parti e ognuna di esse gestisce un tipo di connessione; infatti è stato possibile controllare le varie connessioni in maniera completamente indipendente l'una dall'altra.
I Load Balancer sono predisposti a coordinare le seguenti trasmissioni di pacchetti voce :

1. **Muta** : le applicazioni creano un pacchetto ogni 10 secondi
2. **Lenta o Non Continua** : le applicazioni creano un pacchetto in un arco di tempo che va dai 40 millisecondi ai 10 secondi
3. **Veloce o Continua** : le applicazioni generano un pacchetto ogni 40 millisecondi

Ciò che accomuna i due Load Balancer è il controllo dell'arrivo dei pacchetti voce dalle applicazioni, in modo da poter capire il tipo di connessione su cui si sta lavorando.

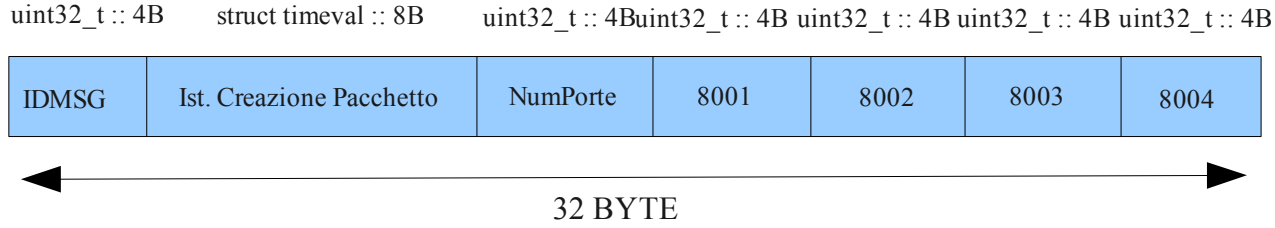
Inizialmente, dopo aver impostato i socket, viene controllato il primo pacchetto spedito dall'applicazione tenendo conto del tempo di arrivo: in base al ritardo del pacchetto si riesce a capire il tipo di connessione utilizzata dalle applicazioni.
Il Load Balancer Mobile avvisa il Fixed inviando tale informazione in un pacchetto su tutte le porte attive precedentemente impostate.

Pacchetto pktTipoConn



Dopo aver inviato il tipo di connessione al Fixed, il Mobile riceve la configurazione delle porte direttamente dalla connessione TCP del Monitor: anche in questo caso viene inviata l'informazione mediante il pacchetto *cfgPKT* al Load Balancer Fixed.

Pacchetto cfgPKT



A questo punto viene scelto la parte di codice relativa al tipo di connessione presente.

Muta

Lettura dei pacchetti UDP provenienti dal Monitor :

come prima cosa il Load Balancer Mobile controlla l'arrivo di nuovi pacchetti inviati dal Fixed: se quel che riceve sono PING (vedesi Load Balancer Fixed) allora calcola il ritardo apposito e controlla se sono arrivati in orario o meno.
Lo stesso avviene per i pacchetti voce che però vengono inoltrati all'applicazione.
Se entrambi i tipi di pacchetto sono in ritardo sulla porta da cui il Load Balancer Mobile sta inviando, la cambia scorrendo il vettore di porte attive circolarmente nel seguente modo:

index=(index+1)%numporte;

ossia viene incrementato l'indice calcolando il modulo per il wrap-around.
Invece se sono in orario su una porta differente, si procede a sincronizzare il Mobile su tale porta, tramite la funzione *trova_porta()*.

Lettura dei pacchetti TCP provenienti dal Monitor :

dopodiché vengono letti i pacchetti **ACK**, **NACK** e di **Configurazione** delle porte.
Se il Mobile riceve un pacchetto Configurazione esegue la seguente procedura:

1. aggiorna l'indice della porta da cui si stavano inviando i pacchetti;
2. aggiorna il vettore di porte con la nuova configurazione;
3. genera il pacchetto *cfgPKT*, mediante la funzione *config_pkt_porte()*;
4. invia tale pacchetto al Fixed tramite la funzione *send_config()*.

Se invece viene ricevuto un pacchetto ACK, si controlla se esiste all'interno della lista di pacchetti (vedesi Lettura dei pacchetti provenienti dall'applicazione) e se il risultato è positivo si procede alla sua eliminazione.
Infine se il Mobile riceve un pacchetto NACK, eseguirà la seguente procedura :

1. controlla che il pacchetto scartato sia presente nella lista di pacchetti;
2. controlla se la porta da cui è stato inviato sia uguale a quella attuale e se lo è la cambia scorrendola nel vettore di porte;
3. rinvia il pacchetto al Fixed.

Lettura dei pacchetti provenienti dall'applicazione :

infine il Mobile legge i pacchetti provenienti dall'applicazione, spedendoli immediatamente al Load Balancer Fixed.
Man mano che i pacchetti vengono spediti, il Mobile li salva all'interno della lista di pacchetti.
Infatti è usata una lista dinamica contenente i pacchetti e alcune loro informazioni tramite l'utilizzo della seguente struttura :

```
struct listaPKT {
        uint32_t pkt[PKTSIZE];
        uint16_t portaPKT;
        struct listaPKT *next;
};
```

Lenta

Lettura dei pacchetti UDP provenienti dal Monitor :

gestita nello stesso modo della connessione Muta.

Lettura dei pacchetti TCP provenienti dal Monitor :

gestita nello stesso modo della connessione Veloce se non per il fatto che viene data la seconda chance una volta ogni due NACK.

Lettura dei pacchetti provenienti dall'applicazione :

gestita allo stesso modo della connessione Muta.

Veloce

Lettura dei pacchetti UDP provenienti dal Monitor :

gestita nello stesso modo della connessione Muta.

Lettura dei pacchetti TCP provenienti dal Monitor :

questa procedura è gestita nello stesso modo della connessione Muta a differenza dell'uso della variabile *contACK*.
Avendo casi in cui il Monitor può scartare dei pacchetti inviati sulla porta *OK* ritornando un NACK al Mobile, questo ultimo conta quanti ACK ha pervenuto nel frattempo: infatti se sono arrivati più di due ACK dietro fila dal Monitor, vuol dire che la porta scelta dal Mobile è quella giusta.
In questo modo si da una seconda chance al pacchetto scartato rinviandolo sulla medesima porta e azzerando poi il contatore *contACK*: se il Monitor ritornerà un ACK di avvenuta consegna, allora il Mobile proseguirà l'invio sulla stessa porta, altrimenti la cambierà dato che la porta sarà divenuta *LOSS*.

Lettura dei pacchetti provenienti dall'applicazione :

gestita allo stesso modo della connessione Muta.

Dopo aver impostato correttamente i socket, come prima cosa il Load Balancer Fixed rimane in ascolto della connessione UDP che ha col Monitor, in attesa del pacchetto del tipo di connessione. Una volta stabilita la connessione in uso, vengono attuate le tre procedure descritte di seguito.

Muta

Lettura dei pacchetti provenienti dall'applicazione :

appena il Load Balancer Fixed riceve dei pacchetti dall'applicazione, li spedisce immediatamente al Mobile.

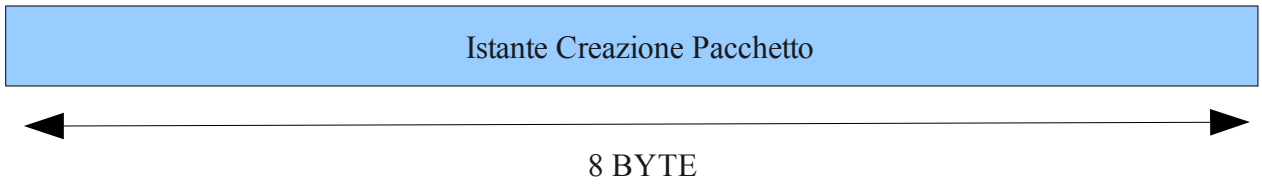
Lettura dei pacchetti UDP provenienti dal Monitor :

dalla connessione UDP col Monitor, il Fixed può ricevere due tipi di pacchetti: voce e configurazione (*cfgPKT*).
I primi sono inoltrati all'applicazione mentre con i secondi vengono aggiornati la porta attuale e il vettore di porte.
In entrambi i casi viene calcolato il ritardo con cui sono arrivati: se sono in orario su una porta diversa, il Fixed ci si aggiorna tramite la funzione *trova_porta()*.

Extra :

dopo la lettura dei socket, si procede ad inviare ogni 500 millisecondi un PING su tutte le porte attive mediante la funzione *send_ping()*, per far conoscere al Mobile la porta corretta.

Pacchetto PING



Lenta

Lettura dei pacchetti provenienti dall'applicazione :

questa procedura viene gestita come nella connessione Veloce, se non che l'invio dei PING avviene in questo frangente se la flag *pktOnDelay* è attiva.

Lettura dei pacchetti UDP provenienti dal Monitor :

se il Fixed riceve dei pacchetti di configurazione (*cfgPKT*), aggiorna l'indice dell'attuale porta d'invio e il vettore di porte attive.
Se invece riceve un pacchetto voce controlla il tempo di arrivo: se è in ritardo invia i PING su tutte le porte attive e attiva la flag *pktOnDelay* caso mai non lo fosse.

Per i pacchetti in orario, il Fixed aggiorna la flag per il randomport disattivandola e aggiornandosi sulla porta da cui è arrivato il pacchetto caso mai fosse diversa dall'attuale d'invio.
Tutti i pacchetti voce sono inoltrati all'applicazione, mentre gli altri vengono filtrati.

Veloce

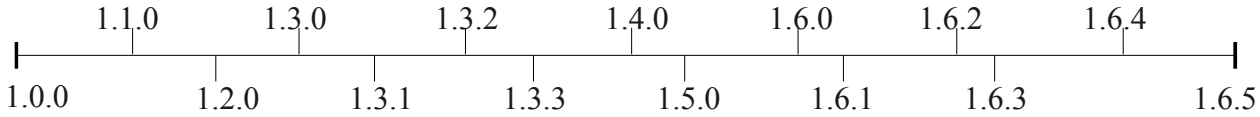
Lettura dei pacchetti provenienti dall'applicazione :

gestita come nella connessione Muta, tranne che per l'uso della flag *pktOnDelay*.
Se tale flag è attiva la porta corrente viene cambiata, scorrendo il vettore di porte.

Lettura dei pacchetti UDP provenienti dal Monitor :

gestite allo stesso modo della connessione Lenta.
Di nuovo vi è il controllo dei pacchetti non in sequenza, ossia se arriva un pacchetto con ID minore dell'ultimo arrivato, semplicemente viene scartato.
Inoltre i PING vengono inviati ad ogni pacchetto voce ricevuto dal Mobile, generando particolarmente poco overhead.
Per i pacchetti ricevuti si usano la flag *pktOnDelay* e il contatore *contLenti*.
La flag *pktOnDelay* serve per attivare il cosiddetto randomport, ossia se arrivano pacchetti in ritardo sulla porta da cui il Fixed sta inviando, comincia a scorrere il vettore di porte inviando i successivi pacchetti in maniera del tutto casuale.
Ovviamente la porta da cui stava ricevendo è divenuta lenta e perciò è giusto cercarne un'altra in attesa di un pacchetto in orario da parte del Mobile.
Il contatore *contLenti* però permette di attivare tale flag solo se arrivano più di due pacchetti in ritardo dietro fila: ciò permette un uso meno spudorato del randomport, aumentando così la percentuale di pacchetti inviati correttamente.
Se invece il Fixed riceve dei pacchetti in orario diminuisce *contLenti*, disattiva la flag per il randomport e controlla se sono arrivati su una porta differente da quella d'invio attuale: se è diversa, il Load Balancer Fixed ci si aggiorna tramite la funzione *trova_porta()*.
Tutti i pacchetti voce sono inoltrati all'applicazione, mentre gli altri vengono filtrati.

Project Timeline



Cartelle

Il progetto è suddiviso nelle seguenti cartelle :

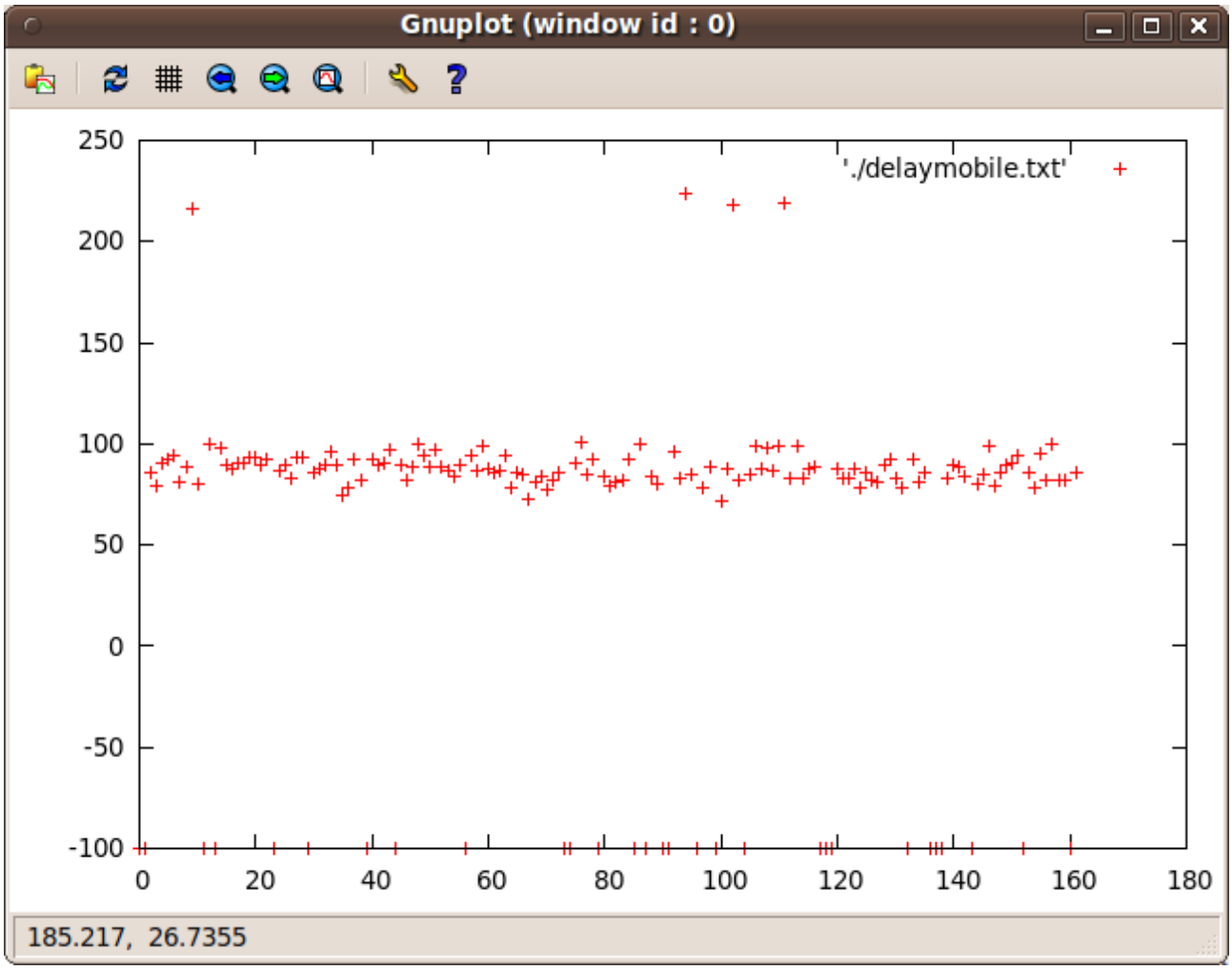
- **./.** : contiene i file di creazione “Makefile” e di informazione “AUTHORS”
- **./doc** : contiene i file della documentazione creata con Doxygen
- **./include** : contiene i file header
- **./src** : contiene i file sorgenti “.c”
- **./sorgente-originale** : contiene i file sorgenti originali del progetto
- **./script** : contiene gli script per l'avvio dell'applicazioni

File

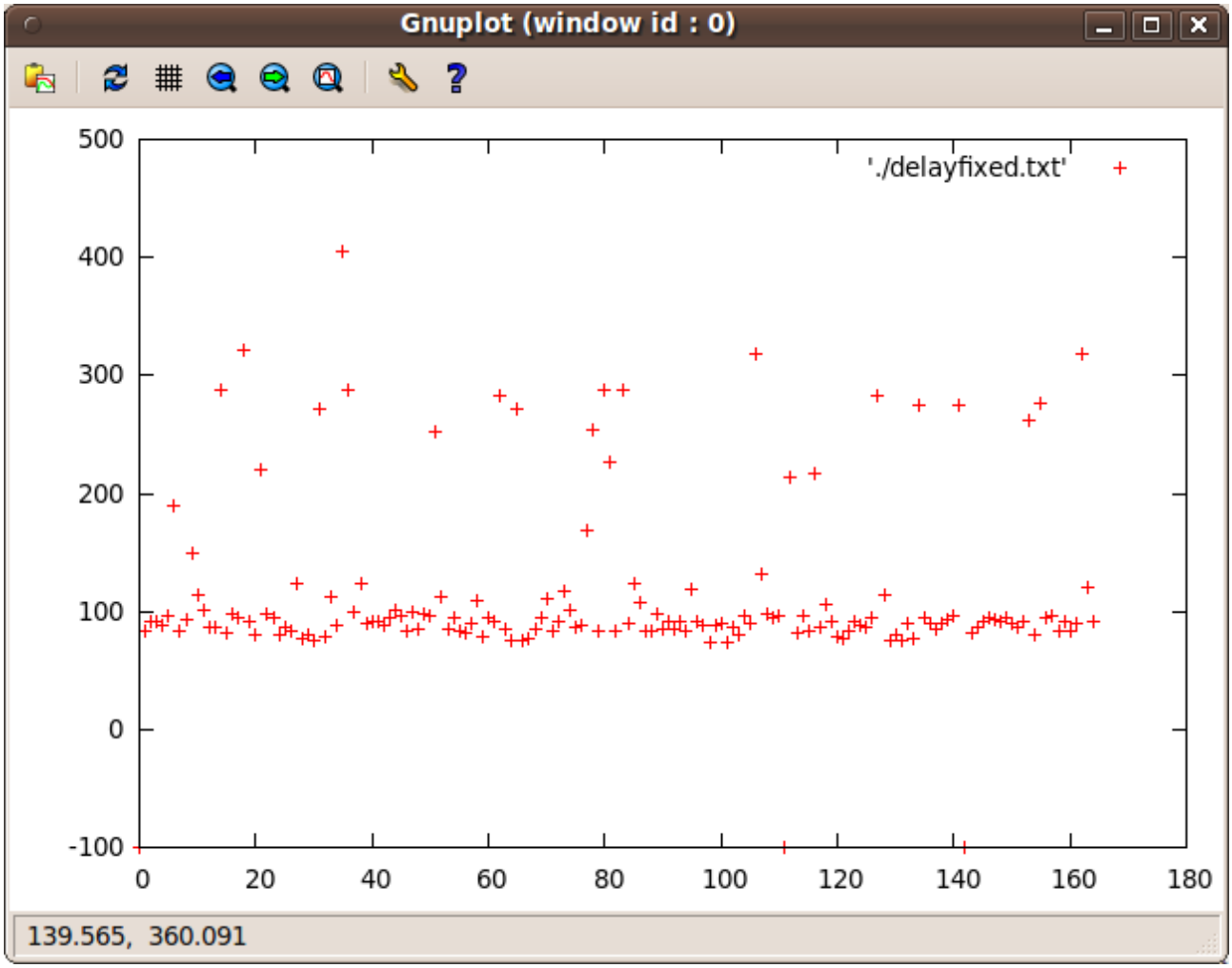
Di seguito sono descritti i vari file realizzati durante il progetto :

- **./AUTHORS** : contiene una descrizione sugli autori
- **./Doxyfile** : file di configurazione per la documentazione con Doxygen
- **./README** : file d'istruzione per l'uso del programma
- **./src/LB.c** : contiene una collezione di funzioni utili ad entrambi i Load Balancer
- **./src/LBfixed.c** : contiene il codice sorgente del gestore della parte Wired
- **./src/LBliste.c** : contiene una collezione di funzioni per l'utilizzo di liste
- **./src/LBmobile.c** : contiene il codice sorgente del gestore della parte Wireless
- **./include/const.h** : contiene la dichiarazione delle costanti
- **./include/LB.h** : file header di LB.c
- **./include/LBliste.h** : file header di Lbliste.c
- **./script/script_five_terminal.sh** : script per l'avvio dei 5 terminali d'uso del programma

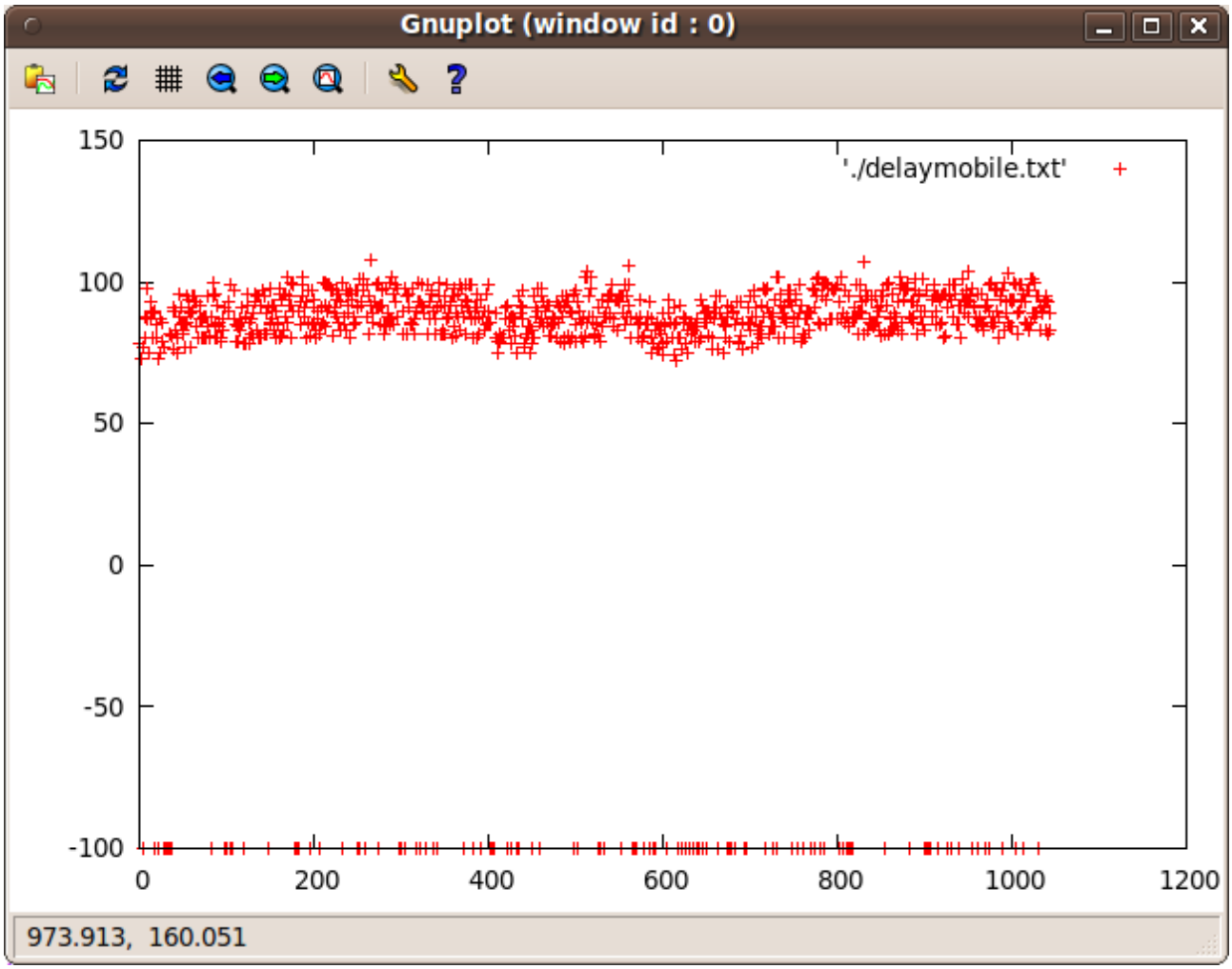
Load Balancer Mobile



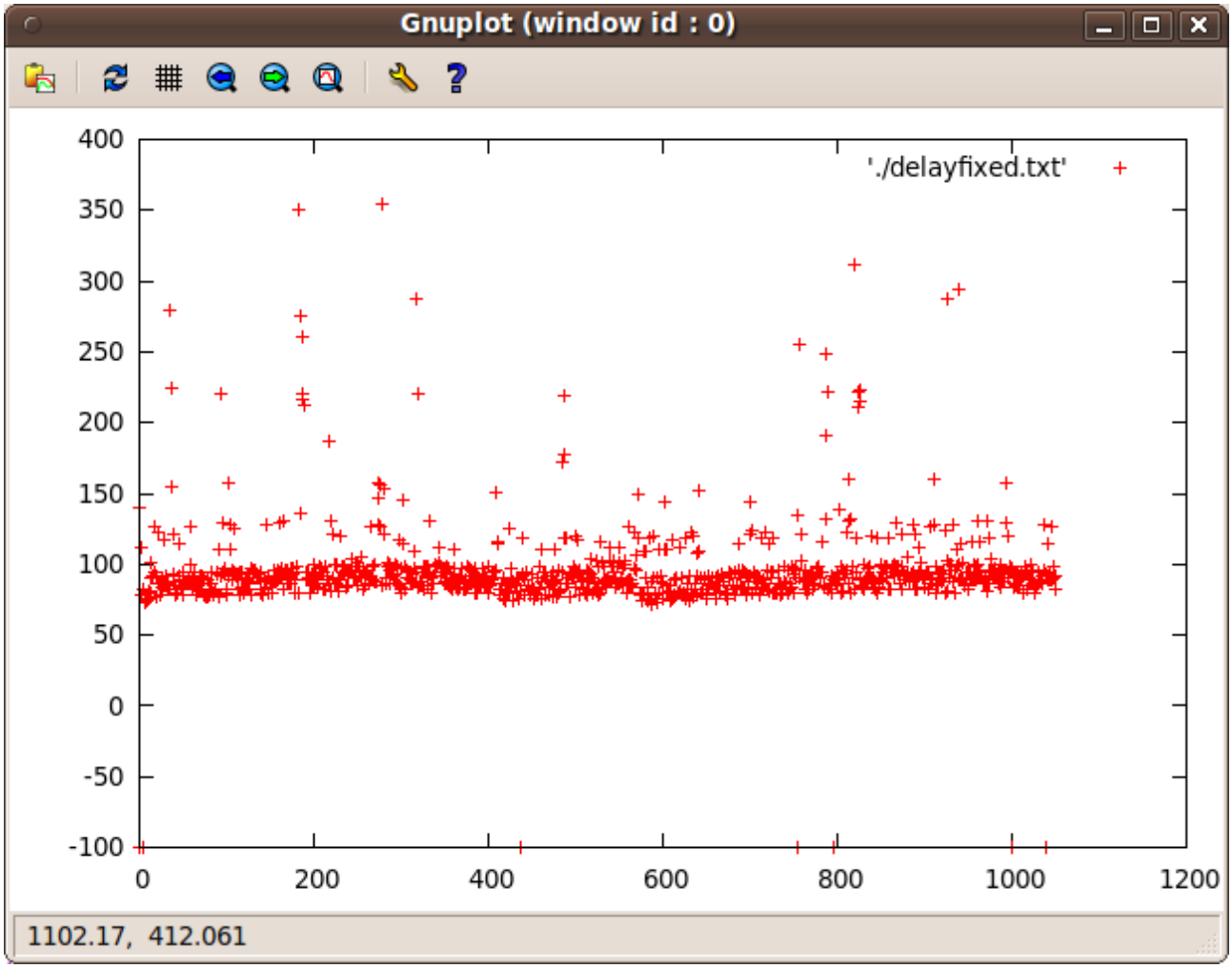
Load Balancer Fixed



Load Balancer Mobile



Load Balancer Fixed



Statistiche VoWLAN

MUTA (~ 15 pacchetti)

Numeri	% Pacchetti Ricevuti		% Pacchetti Scartati		% Overhead	
	M2F	F2M	M2F	F2M	M2F	F2M
0	56%	66%	44%	34%	50%	15%
1	59%	60%	41%	40%	54%	15%
2	67%	69%	33%	31%	52%	15%
3	57%	62%	43%	38%	49%	16%
4	60%	60%	40%	40%	51%	16%

LENTA (~ 150 pacchetti)

Numeri	% Pacchetti Ricevuti		% Pacchetti Scartati		% Overhead	
	M2F	F2M	M2F	F2M	M2F	F2M
0	87%	75%	13%	25%	29%	2%
1	89%	75%	11%	25%	24%	3%
2	86%	74%	14%	26%	27%	1%
3	82%	72%	18%	28%	24%	5%
4	85%	81%	15%	19%	34%	3%

VELOCE (~ 1000 pacchetti)

Numeri	% Pacchetti Ricevuti		% Pacchetti Scartati		% Overhead	
	M2F	F2M	M2F	F2M	M2F	F2M
0	95%	85%	5%	15%	12%	7%
1	96%	88%	4%	12%	15%	1%
2	91%	83%	9%	17%	17%	7%
3	96%	87%	4%	13%	14%	5%
4	95%	85%	5%	15%	16%	5%