# Reti di calcolatori e sicurezza

**tunerr**

**Miro Salvagni**

# Indice

# 1 Analisi

## 1.1 Descrizione del sistema

Si vuole realizzare un tunnel, a livello IP, che implementi le seguenti funzionalità:

- Perdita di pacchetti
- Introduzione di errori
- Imposizione di limiti di velocità al trasferimento dei dati.

## 1.2 Note implementative

L'intero sistema è sviluppato in c++.
Il sistema è pensato per funzionare su sistemi *linux* che utilizzano le librerie *libipq*. Con questa libreria è possibile scrivere programmi in *user-space* che ricevono i pacchetti dal kernel tramite l'uso di *iptables*.
Per eseguire l'applicazione è necessario avere i privilegi dell'utente *root*.

## 1.3 Funzionamento generale

Il sistema è pensato per rimanere sempre in esecuzione, come un qualsiasi *demone* linux.

All'avvio il programma legge dei file di regole, nei quali sono descritte le azioni da intraprendere per i diversi tipi di pacchetti che arrivano al programma.
La possibilità di creare diverse regole contemporaneamente, permette, ad esempio, di introdurre errori per una tipologia di pacchetti e di scartarne altri avendo in esecuzione un'unica istanza dell'applicazione.

L'utente interagisce con il sistema in modo analogo ad *iptables*: dopo la prima esecuzione è possibile cambiare i file delle regole, visualizzare le regole attive e terminare il programma specificando gli opportuni parametri nella riga di comando.

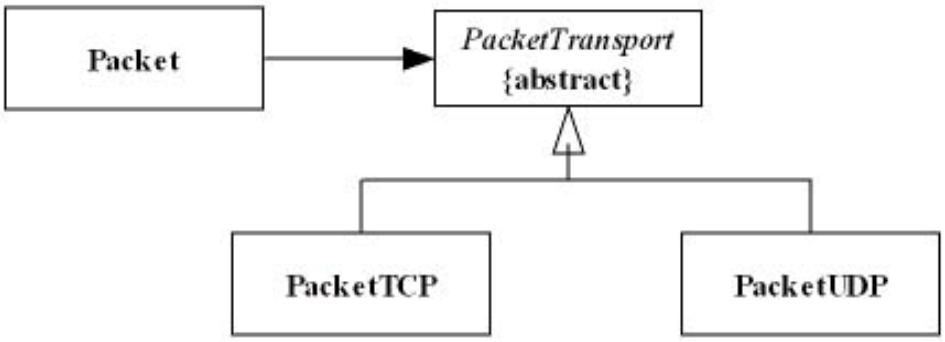
# 2 Design ed implementazione

## 2.1 Pacchetti

Il sistema deve supportare pacchetti di tipo ip, tcp ed udp.
Indipendentemente dal tipo del pacchetto deve essere possibile ottenere alcune informazioni, quali la dimensione, il mittente, il destinatario, le porte che lo interessano e leggere e scrivere ogni byte che lo compone.

Dal momento che ogni pacchetto udp e tcp è contenuto in un pacchetto ip, sono state scritte le seguenti classi:

- *Packet*: il pacchetto IP, contiene un oggetto PacketTransport
- *PacketTransport*: una classe virtuale, che rappresenta un pacchetto a livello di trasporto.
- *PacketUDP*: la specializzazione di PacketTransport per il protocollo UDP.
- *PacketTCP*: la specializzazione di PacketTransport per il protocollo TCP.




## 2.2 Matcher

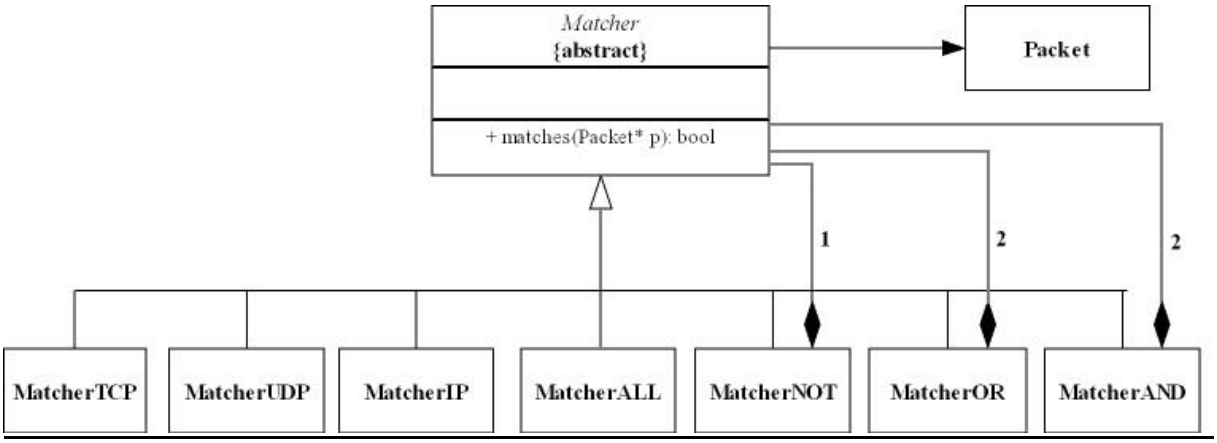
Poiché ogni regola è pensata in modo da influenzare solo una determinata categoria di pacchetti, è introdotto un meccanismo di matching che si propone di individuare, dato un pacchetto, se la regola deve o no influenzarlo.

Il meccanismo è implementato nelle classi *Matcher* (la classe astratta principale), *MatcherTCP*, *MatcherUDP*, *MatcherIP*. Il metodo, nella classe Matcher, che svolge questo compito è:

*virtual bool matches(Packet* p);*

Oltre a questi tre matcher che gestiscono i vari protocolli supportati sono stati creati dei matcher composti: *MatcherOR*, *MatcherNOT*, *MatcherAND* coi quali è possibile comporre i matcher semplici ottenendo matcher più complessi.

Infine, è stata creata la classe *MachetALL* (la cui funziona *matches(Packet*)* torna sempre *true*) per le regole che devono influenzare tutti i pacchetti che arrivano al sistema. Questa è particolarmente utile se si intende lasciare ad *iptables* il compito di determinare quali pacchetti devono passare nel tunnel.




## 2.3 Regole

Le regole sono implementate come segue:

*Rule* è la classe astratta principale, da cui tutte le regole ereditano. Contiene la funzione:

*virtual int apply(Packet* p);*

Che applica il comportamento selezionato al pacchetto.

La classe *RuleSet* implementa il design pattern *Composite*, permettendo di avere set di regole che vengono viste dal sistema come un'unica regola. Il sistema crea, all'avvio, il *RuleSet* che viene utilizzato per tutta la durata dell'esecuzione.

*RuleNetwork* è un'ulteriore classe astratta, ereditata da *Rule*, che contiene un *Matcher*. Qualora l'operazione di matching dia esito positivo, viene chiamata la funzione:

*virtual int do_apply(Packet* p);*

Che ha, per le classi che ereditano da *RuleNetwork*, lo stesso comportamento di *int Rule::apply(Packet*)*.

*RuleNothing* è il tipo di regola *RuleNetwork* più semplice, non ha nessun effetto sul pacchetto.

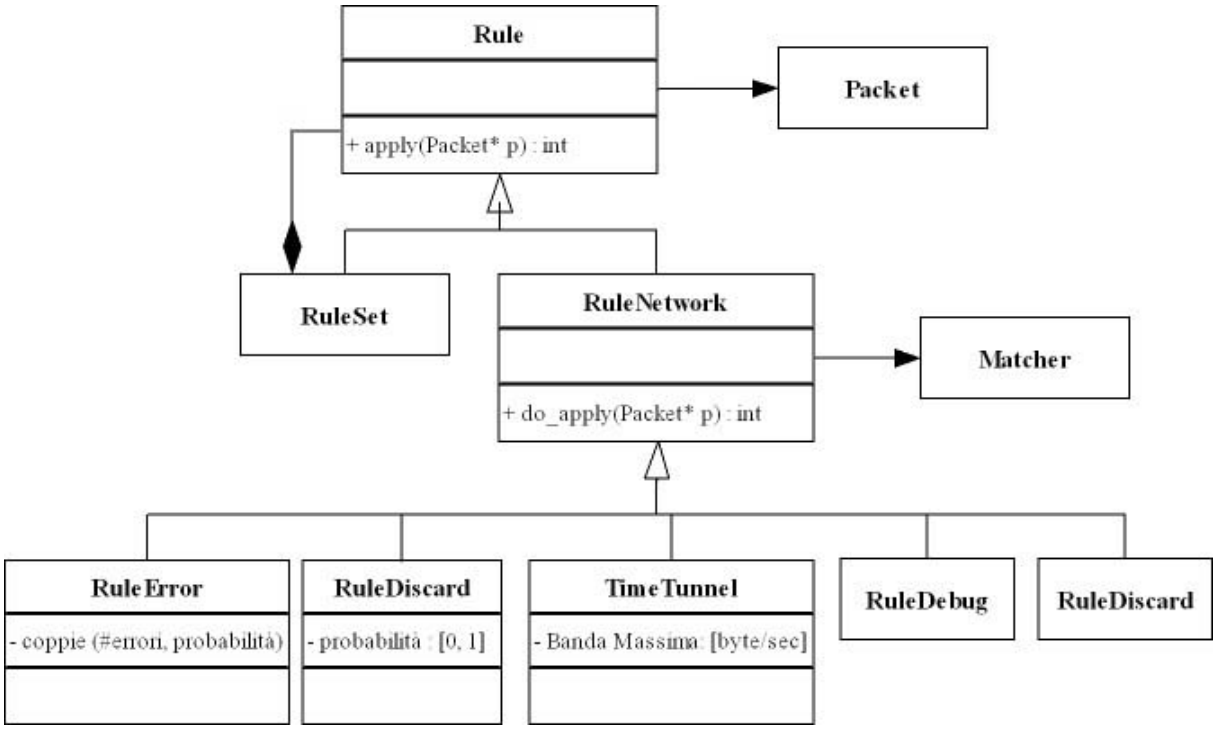
*RuleDebug* è implementata, a scopi di debug, per visualizzare in standard output il passaggio di un pacchetto.

*RuleDiscard* implementa la perdita di pacchetti. Il suo unico parametro è un numero decimale compreso tra 0 e 1 che indica la probabilità che un pacchetto venga scartato.

*RuleError* implementa l'introduzione casuale di errori nei pacchetti. Ha come parametri delle coppie (n, p), dove n è il numero di errori che verranno introdotti nel pacchetto con probabilità p. Chiaramente, la somma totale degli indici di probabilità deve essere uguale a 1 (se questa somma è minore di 1, la probabilità che nel pacchetto non vengano generati errori è impostata a 1-somma(pi)).

*TimeTunnel*: implementa l'introduzione di limiti di velocità alla trasmissione dei pacchetti. Ha come unico parametro la velocità massima che si vuole ottenere, espressa in byte per secondo.

Per creare gli oggetti che rappresentano le regole si usa unicamente la classe *RuleFactory*, che contiene metodi statici per leggere i parametri dai file.



## 2.4 Configurazione

La classe *Config* si occupa di gestire la configurazione. Dal momento che questa deve essere accessibile in più parti del programma, è implementata secondo il design pattern *Singleton*.

Nella versione finale del sistema, l'istanza dell'oggetto *Config* si occupa unicamente di memorizzare il nome della directory che contiene le regole e il *RuleSet* del programma.

All'interno della classe è presente la funzione:

*static bool reload();*

Che si occupa di caricare le regole dai file quando l'utente lo richiede.

## 2.5 La classe principale: TunErr

La classe principale gestisce la comunicazione con le librerie *libipq*.
Contiene, inoltre, delle funzioni per assicurarsi che non ci sia, al momento dell'esecuzione, un'altra istanza di *tunerr* in esecuzione. Qualora il sistema rilevi un altro processo di tipo *tunerr*, verrà analizzata la riga di comando fornita dall'utente, verranno modificati i file delle regole come richiesto e verrà inviato al processo in esecuzione un segnale per indicargli di rileggere i file di configurazione.

## 2.6 Programmi di test

Per le operazioni di test del programma sono stati sviluppati 2 coppie di due programmi: *tcpsrv*/*tcpclnt*, *udpsrv*/*udpclnt*.
Ogni coppia implementa un server e un client per i protocolli tcp e udp.
Il comportamento di questi programmi non cambia a seconda del protocollo usato:
il client, una volta connesso al server, invia dei pacchetti di dati che il server gli rispedisce così come li riceve.
Il client analizza i pacchetti ricevuti confrontandoli con quelli inviati ottenendo così il numero di pacchetti persi, il numero di errori contenuti nei pacchetti e le velocità di trasferimento medie e istantanee.

# 3 Manuale utente

## 3.1 Lancio dell'applicazione

Per avviare il software è necessario disporre dei permessi dell'amministratore di sistema (*root*).
Prima di avviare *tunerr*, si deve fare in modo che *iptables* instradi i pacchetti che si intendono trattare. E' disponibile, nella directory dei sorgenti, lo script d'esempio *iptables_init*:

```
#!/bin/bash

modprobe iptables_filter
modprobe ip_queue

iptables -A OUTPUT -p icmp -j QUEUE
iptables -A INPUT -p udp --dport 17777 -j QUEUE
iptables -A INPUT -p tcp --dport 17777 -j QUEUE
```

In questo esempio vengono instradati attraverso tunerr tutti i pacchetti tcp ed udp con porta di destinazione 17777 (per maggiori informazioni sulla sintassi di iptables si consultino le relative *man-pages*).

Successivamente, si può lanciare il programma vero e proprio:

```
# ./tunerr
```

Il programma a questo punto è in esecuzione, e tutti i pacchetti verranno trattati come descritti nei file delle regole.

## 3.2 La riga di comando

Il software è stato pensato in modo che l'utente non debba cambiare i file di configurazione manualmente. Tramite la riga di comando l'utente può terminare il programma, visualizzare, creare e cancellare le regole.

Tutti questi comandi sono disponibili solo quando *tunerr* è già in esecuzione. La riga di comando viene ignorata al lancio del programma.

### 3.2.1 Richiesta di aiuto

Per visualizzare una lista dei comandi disponibili, l'utente ha a disposizione il comando "help":

```
# ./tunerr --help
```

Verrà visualizzata una lista dei comandi disponibili.

### 3.2.2 Terminazione del software

Per visualizzare una lista dei comandi disponibili, l'utente ha a disposizione il comando "terminate":

```
# ./tunerr --terminate
```

Il processo *tunerr* principale verrà terminato.

### 3.2.3 Ricaricare i file delle regole

Qualora fosse necessario (ad esempio se i file delle regole vengono modificati manualmente, o da altri processi), l'utente può far si che *tunerr* ricarichi i file delle regole tramite il comando "reload":

```
# ./tunerr --reload
```

Il programma ricaricherà le regole.

### 3.2.4 Visualizzazione delle regole attive

Per visualizzare una lista delle regole attive, l'utente ha a disposizione il comando "show":

```
# ./tunerr --show
```

Verrà visualizzata una lista delle regole attive.

### 3.2.5 Creazione di una nuova regola

Per creare una nuova regola (e renderla attiva), è messo a disposizione dell'utente il comando "add":

```
# ./tunerr --add [matching <matcher>] policy [ debug | discard <prob> |
error <errors> | bandlimit <limit> | nothing]
```

Verrà creata una regola con le caratteristiche specificate.

#### 3.2.5.1 Campo Matching

Il campo matching è opzionale, qualora non specificato *tunerr* tratterà tutti i pacchetti che gli arriveranno da *iptables*.

In alternativa, è possibile specificare dei matcher per selezionare il filtraggio di pacchetti ip, tcp, e udp.

I matcher ip supportano i seguenti parametri:

- *daddr*: indirizzo di destinazione. Seguito da >, >=, <, <=, ==, != e da un numero di 32bit che specifica il valore.
- *saddr*: indirizzo di provenienza. Seguito da >, >=, <, <=, ==, != e da un numero di 32bit che specifica il valore.
- *ttl*: tempo di vita del pacchetto. Seguito da >, >=, <, <=, ==, != e da un numero di 32bit che specifica il valore.
- *length*: lunghezza del pacchetto ip. Seguito da >, >=, <, <=, ==, != e da un numero di 32bit che specifica il valore.

Ad esempio:

```
matching ip ttl > 1 length == 3200
```

Seleziona il filtraggio di tutti i pacchetti ip con campo ttl maggiore di uno e dimensione esattamente uguale a 3200.

I matcher tcp supportano i seguenti parametri:

- *dport*: porta di destinazione. Seguito da >, >=, <, <=, ==, != e da un numero di 16bit che specifica il valore.
- *sport*: porta di provenienza. Seguito da >, >=, <, <=, ==, != e da un numero di 16bit che specifica il valore.

Ad esempio:

```
matching tcp sport <= 1024 dport == 7777
```

Seleziona il filtraggio di tutti i pacchetti tcp con porta di provenienza minore o uguale a 1024 e porta di destinazione 7777.

I matcher udp supportano i seguenti parametri:

- *dport*: porta di destinazione. Seguito da >, >=, <, <=, ==, != e da un numero di 16bit che specifica il valore.
- *sport*: porta di provenienza. Seguito da >, >=, <, <=, ==, != e da un numero di 16bit che specifica il valore.

- *chk*: checksum. Seguito da >, >=, <, <=, ==, != e da un numero di 16bit che specifica il valore. (il campo è messo ai soli fini di completezza, qualora l'utente volesse utilizzarlo.)
- *length*: lunghezza del pacchetto ip. Seguito da >, >=, <, <=, ==, != e da un numero di 32bit che specifica il valore.

Ad esempio:

```
matching tcp dport == 7777 length > 10000
```

Seleziona il filtraggio di tutti i pacchetti tcp porta di destinazione 7777 e dimensione maggiore di 10000.

E' inoltre possibile comporre più matcher coi comandi AND, OR, NOT:

- NOT matcher: si selezionano i pacchetti che non vengono selezionati da matcher.
- matcher1 AND matcher2: si selezionano i pacchetti che vengono selezionati sia da matcher1 che da matcher2.
- matcher1 OR matcher2: si selezionano i pacchetti che vengono selezionati da matcher1 o da matcher2.

Ad esempio:

```
matching udp dport == 7777 AND matching ip length > 10000
```

Seleziona tutti i pacchetti che hanno dimensione nel campo ip maggiore di 10000, sono di tipo udp, e hanno specificata la porta di destinazione 7777.

#### 3.2.5.2 Campo Policy

Attraverso il campo policy si specifica come una regola agisce su un pacchetto che viene filtrato. E' possibile specificare:

- *debug*: per ogni pacchetto che viene selezionato dal matcher verrà visualizzato un avvertimento nello standard output.
- *nop*: non verrà eseguita nessuna operazione sui pacchetti.
- *discard*: seguito da un numero decimale *p* compreso tra 0 e 1. Ogni pacchetto avrà una probabilità *p* di essere scartato.
- *bandlimit*: seguito da un numero intero *b*. Il sistema farà in modo che la velocità massima, espressa in byte/secondo, per i pacchetti selezionati, sia *b*.
- *error*: seguito da stringhe "prob *n p*" dove n è un numero intero e p un numero decimale compreso tra 0 e 1. Il sistema introdurrà, con probabilità *p*, *n* errori nel pacchetto selezionato. Oltre a questo è possibile specificare il campo affect:
    - *affect all*: gli errori saranno introdotti nell'intero pacchetto.

- *affect transport*: gli errori saranno introdotti nell'header di trasporto e nel campo dati del pacchetto.
- *affect data*: gli errori saranno introdotti unicamente nel campo dati del pacchetto.

Ad esempio:

```
policy debug
```

Per ogni pacchetto verrà stampato un avviso.

```
policy discard 0.5
```

Ogni pacchetto avrà una probabilità di 0.5 (50%) di venir scartato.

```
policy error prob 1 0.2 prob 2 0.1 affect data
```

Ogni pacchetto avrà una probabilità di 0.2 (20%) di contenere 1 errore e 0.1 (10%) di contenerne 2. (Si noti che la probabilità che non vengano introdotti errori sarà 1-0.2-0.1=0.7, 70%).
Gli errori verranno introdotti solamente nel campo dati.

### 3.2.6 Cancellazione di una regola

Per cancellare una regola attiva, è messo a disposizione dell'utente il comando "delete":

```
# ./tunerr –-delete <numero_regola>
```

Verrà eliminata la regola numero <numero_regola> (la prima regola attiva ha numero 0).
Per conoscere il numero della regola che si intende eliminare, utilizzare il comando show.

### 3.2.7 Cancellazione di tutte le regole

Per cancellare tutte le regole attive, è messo a disposizione dell'utente il comando "clear":

```
# ./tunerr –-clear
```

Verranno eliminate tutte le regole attive.

# 4 Conclusioni

## 4.1 Risultati

Nei seguenti risultati, si vede tra parentesi la percentuale effettiva di pacchetti scartati.

### 4.1.1 Discard 0%

```
Speed = 1000, PacketSize = 1000, Pattern = 4289331370
packets:        (0.000000) 42 = 42 - 0
packets:        (0.000000) 0 = 3556 - 3556
packets:        (0.000000) 0 = 7038 - 7038
packets:        (0.000000) 0 = 10514 - 10514
packets:        (0.000000) 0 = 17550 - 17550
packets:        (0.000000) 0 = 27464 - 27464
packets:        (0.000000) 0 = 37306 - 37306
packets:        (0.000000) 0 = 47173 - 47173
packets:        (0.000000) 0 = 57053 - 57053
packets:        (0.000000) 0 = 66883 - 66883
packets:        (0.000000) 0 = 76731 - 76731
packets:        (0.000000) 0 = 86633 - 86633
```

### 4.1.2 Discard 20%

```
Speed = 1000, PacketSize = 1000, Pattern = 4289331370
packets:        (0.181818) 6 = 33 - 27
packets:        (0.199744) 938 = 4696 - 3758
packets:        (0.201838) 1889 = 9359 - 7470
packets:        (0.200699) 2814 = 14021 - 11207
packets:        (0.201009) 4940 = 24576 - 19636
packets:        (0.199811) 7393 = 37000 - 29607
packets:        (0.198619) 9812 = 49401 - 39589
packets:        (0.199237) 12322 = 61846 - 49524
packets:        (0.199001) 14777 = 74256 - 59479
packets:        (0.198804) 17221 = 86623 - 69402
packets:        (0.199461) 19767 = 99102 - 79335
packets:        (0.199738) 22282 = 111556 - 89274
```

### 4.1.3 Discard 50%

```
Speed = 1000, PacketSize = 1000, Pattern = 4289331370
packets:        (0.690614) 1317 = 1907 - 590
packets:        (0.559529) 3520 = 6291 - 2771
packets:        (0.536090) 5704 = 10640 - 4936
packets:        (0.525278) 7886 = 15013 - 7127
packets:        (0.515717) 9975 = 19342 - 9367
```

```
packets:        (0.513424) 12201 = 23764 - 11563
packets:        (0.507780) 18339 = 36116 - 17777
packets:        (0.506829) 24639 = 48614 - 23975
packets:        (0.504224) 30797 = 61078 - 30281
packets:        (0.503457) 37060 = 73611 - 36551
packets:        (0.503241) 43316 = 86074 - 42758
packets:        (0.502370) 49498 = 98529 - 49031
packets:        (0.502548) 55816 = 111066 - 55250
packets:        (0.502290) 62080 = 123594 - 61514
packets:        (0.502256) 68353 = 136092 - 67739
packets:        (0.502167) 74616 = 148588 - 73972
packets:        (0.502342) 80872 = 160990 - 80118
packets:        (0.502363) 87165 = 173510 – 86345
```

### 4.1.4 Discard 90%

```
Speed = 1000, PacketSize = 1000, Pattern = 4289331370
packets:        (0.860082) 209 = 243 - 34
packets:        (0.890976) 5075 = 5696 - 621
packets:        (0.895842) 15017 = 16763 - 1746
packets:        (0.896600) 29109 = 32466 - 3357
packets:        (0.897556) 43290 = 48231 - 4941
packets:        (0.898158) 57474 = 63991 - 6517
packets:        (0.899165) 71614 = 79645 - 8031
packets:        (0.899818) 85830 = 95386 - 9556
packets:        (0.899804) 99970 = 111102 - 11132
packets:        (0.899648) 114115 = 126844 - 12729
packets:        (0.899555) 128236 = 142555 - 14319
packets:        (0.899762) 142435 = 158303 - 15868
packets:        (0.899701) 156574 = 174029 - 17455
packets:        (0.899915) 170793 = 189788 - 18995
packets:        (0.900012) 185002 = 205555 - 20553
packets:        (0.899995) 199149 = 221278 – 22129
```

### 4.1.5 Discard 100%

```
Speed = 1000, PacketSize = 1000, Pattern = 4289331370
packets:        (1.000000) 128 = 128 - 0
packets:        (1.000000) 6020 = 6020 - 0
packets:        (1.000000) 11898 = 11898 - 0
packets:        (1.000000) 24007 = 24007 - 0
packets:        (1.000000) 40885 = 40885 - 0
packets:        (1.000000) 57858 = 57858 - 0
packets:        (1.000000) 74723 = 74723 - 0
packets:        (1.000000) 91560 = 91560 – 0
```

### 4.1.6 Introduzione di errori: 30% 1 errore, 10% 2.

In questo esempio è indicato tra parentesi il numero medio di errori per pacchetto misurata dal client.

```
Speed = 1000, PacketSize = 1000, Pattern = 4289331370
errors: (0.333333) 2

errors: (0.499281) 1736
errors: (0.502394) 3463
errors: (0.498452) 5152
errors: (0.500363) 6897
errors: (0.494927) 11170
errors: (0.494729) 16002
errors: (0.493180) 20754
errors: (0.496194) 25750
errors: (0.498015) 30733
errors: (0.496852) 35517
errors: (0.497891) 40483
errors: (0.497350) 45320
errors: (0.497359) 50192
errors: (0.497029) 55031
errors: (0.497345) 59951
errors: (0.497085) 64803
errors: (0.496546) 69571
```

### 4.1.7 Bandlimit 1.000.000b/sec

In questo esempio, e in quelli che seguono, si vede tra parentesi, nella colonna in, la banda massima percepita dal client.
Si noti che il limite è imposto sull'effettiva dimensione del traffico, che comprende gli header dei layer di rete e trasporto, mentre i valori riportati fanno riferimento solo alla dimensione del campo dati dei pacchetti ricevuti.

```
Speed = 1000, PacketSize = 1000, Pattern = 4289331370
Mean Band usage:       (in: 285714 Kb/s)      (out: 6.57143e+06 Kb/s)
Mean Band usage:       (in: 324.256 Kb/s)     (out: 324.256 Kb/s)
Mean Band usage:       (in: 966.306 Kb/s)     (out: 966.306 Kb/s)
Mean Band usage:       (in: 580.266 Kb/s)     (out: 580.266 Kb/s)
Mean Band usage:       (in: 965.298 Kb/s)     (out: 965.298 Kb/s)
Mean Band usage:       (in: 690.393 Kb/s)     (out: 690.393 Kb/s)
Mean Band usage:       (in: 965.295 Kb/s)     (out: 965.295 Kb/s)
Mean Band usage:       (in: 751.091 Kb/s)     (out: 751.091 Kb/s)
Mean Band usage:       (in: 964.794 Kb/s)     (out: 964.794 Kb/s)
Mean Band usage:       (in: 784.987 Kb/s)     (out: 1014.27 Kb/s)
Mean Band usage:       (in: 958.12 Kb/s)      (out: 1275.1 Kb/s)
Mean Band usage:       (in: 811.822 Kb/s)     (out: 1055.8 Kb/s)
Mean Band usage:       (in: 952.673 Kb/s)     (out: 1575.94 Kb/s)
Mean Band usage:       (in: 827.112 Kb/s)     (out: 1325.98 Kb/s)
Mean Band usage:       (in: 954.759 Kb/s)     (out: 1488.99 Kb/s)
Mean Band usage:       (in: 842.202 Kb/s)     (out: 1355.29 Kb/s)
Mean Band usage:       (in: 948.604 Kb/s)     (out: 1816.89 Kb/s)
```

```
Mean Band usage:       (in: 840.163 Kb/s)     (out: 2077.7 Kb/s)
Mean Band usage:       (in: 940.164 Kb/s)     (out: 2245.92 Kb/s)
Mean Band usage:       (in: 852.179 Kb/s)     (out: 1971.81 Kb/s)
Mean Band usage:       (in: 941.082 Kb/s)     (out: 2205.72 Kb/s)
Mean Band usage:       (in: 856.385 Kb/s)     (out: 2178.44 Kb/s)
Mean Band usage:       (in: 938.936 Kb/s)     (out: 2320.62 Kb/s)
Mean Band usage:       (in: 865.18 Kb/s)      (out: 2081.44 Kb/s)
Mean Band usage:       (in: 941.214 Kb/s)     (out: 2207.76 Kb/s)
Mean Band usage:       (in: 867.725 Kb/s)     (out: 2255.23 Kb/s)
Mean Band usage:       (in: 938.009 Kb/s)     (out: 2373.18 Kb/s)
Mean Band usage:       (in: 874.524 Kb/s)     (out: 2161.57 Kb/s)
Mean Band usage:       (in: 934.691 Kb/s)     (out: 2548.98 Kb/s)
Mean Band usage:       (in: 875.689 Kb/s)     (out: 2334.12 Kb/s)
Mean Band usage:       (in: 936.864 Kb/s)     (out: 2443.54 Kb/s)
Mean Band usage:       (in: 881.18 Kb/s)      (out: 2251.2 Kb/s)
Mean Band usage:       (in: 938.765 Kb/s)     (out: 2351.27 Kb/s)
Mean Band usage:       (in: 886.101 Kb/s)     (out: 2177.82 Kb/s)
Mean Band usage:       (in: 940.384 Kb/s)     (out: 2269.8 Kb/s)
Mean Band usage:       (in: 888.12 Kb/s)      (out: 2231.72 Kb/s)
Mean Band usage:       (in: 939.443 Kb/s)     (out: 2320.08 Kb/s)
Mean Band usage:       (in: 892.181 Kb/s)     (out: 2166.86 Kb/s)
Mean Band usage:       (in: 940.908 Kb/s)     (out: 2248.88 Kb/s)
Mean Band usage:       (in: 892.297 Kb/s)     (out: 2289.43 Kb/s)
Mean Band usage:       (in: 938.59 Kb/s)      (out: 2370.38 Kb/s)
Mean Band usage:       (in: 895.739 Kb/s)     (out: 2227.88 Kb/s)
Mean Band usage:       (in: 939.957 Kb/s)     (out: 2303.57 Kb/s)
Mean Band usage:       (in: 898.919 Kb/s)     (out: 2171.84 Kb/s)
Mean Band usage:       (in: 941.153 Kb/s)     (out: 2242.78 Kb/s)
Mean Band usage:       (in: 901.828 Kb/s)     (out: 2120.57 Kb/s)
Mean Band usage:       (in: 940.34 Kb/s)      (out: 2286.29 Kb/s)
Mean Band usage:       (in: 902.67 Kb/s)      (out: 2283.29 Kb/s)
```

### 4.1.8 Bandlimit 5.000.000b b/sec

```
Speed = 1000, PacketSize = 1000, Pattern = 4289331370
Mean Band usage:       (in: 0 Kb/s)   (out: 1.36571e+08 Kb/s)
Mean Band usage:       (in: 1605.15 Kb/s)     (out: 2186.36 Kb/s)
Mean Band usage:       (in: 4738.75 Kb/s)     (out: 5608.4 Kb/s)
Mean Band usage:       (in: 2835.07 Kb/s)     (out: 3183.47 Kb/s)
Mean Band usage:       (in: 4707.42 Kb/s)     (out: 5142.25 Kb/s)
Mean Band usage:       (in: 3368.19 Kb/s)     (out: 3616.93 Kb/s)
Mean Band usage:       (in: 4712.22 Kb/s)     (out: 5002.1 Kb/s)
Mean Band usage:       (in: 3669.47 Kb/s)     (out: 3862.9 Kb/s)
Mean Band usage:       (in: 4715.13 Kb/s)     (out: 4932.54 Kb/s)
Mean Band usage:       (in: 3861.49 Kb/s)     (out: 4019.72 Kb/s)
Mean Band usage:       (in: 4716.86 Kb/s)     (out: 4890.8 Kb/s)
Mean Band usage:       (in: 3993.32 Kb/s)     (out: 4127.19 Kb/s)
Mean Band usage:       (in: 4717.2 Kb/s)      (out: 4862.14 Kb/s)
Mean Band usage:       (in: 4090.9 Kb/s)      (out: 4206.91 Kb/s)
Mean Band usage:       (in: 4718.43 Kb/s)     (out: 4842.66 Kb/s)
Mean Band usage:       (in: 4165.63 Kb/s)     (out: 4267.99 Kb/s)
Mean Band usage:       (in: 4719.1 Kb/s)      (out: 4827.81 Kb/s)
Mean Band usage:       (in: 4224.31 Kb/s)     (out: 4315.89 Kb/s)
Mean Band usage:       (in: 4719.5 Kb/s)      (out: 4816.13 Kb/s)
Mean Band usage:       (in: 4271.9 Kb/s)      (out: 4354.75 Kb/s)
Mean Band usage:       (in: 4719.95 Kb/s)     (out: 4806.91 Kb/s)
Mean Band usage:       (in: 4310.78 Kb/s)     (out: 4386.42 Kb/s)
```

```
Mean Band usage:       (in: 4720.03 Kb/s)     (out: 4799.09 Kb/s)
Mean Band usage:       (in: 4343.99 Kb/s)     (out: 4413.58 Kb/s)
Mean Band usage:       (in: 4720.34 Kb/s)     (out: 4792.82 Kb/s)
Mean Band usage:       (in: 4372.13 Kb/s)     (out: 4436.57 Kb/s)
Mean Band usage:       (in: 4720.69 Kb/s)     (out: 4787.59 Kb/s)
Mean Band usage:       (in: 4396.46 Kb/s)     (out: 4456.45 Kb/s)
Mean Band usage:       (in: 4720.98 Kb/s)     (out: 4783.1 Kb/s)
Mean Band usage:       (in: 4417.65 Kb/s)     (out: 4473.77 Kb/s)
Mean Band usage:       (in: 4721.04 Kb/s)     (out: 4779.02 Kb/s)
Mean Band usage:       (in: 4435.97 Kb/s)     (out: 4488.69 Kb/s)
Mean Band usage:       (in: 4721.15 Kb/s)     (out: 4775.51 Kb/s)
Mean Band usage:       (in: 4452.48 Kb/s)     (out: 4502.19 Kb/s)
Mean Band usage:       (in: 4721.43 Kb/s)     (out: 4772.58 Kb/s)
Mean Band usage:       (in: 4467.26 Kb/s)     (out: 4514.28 Kb/s)
Mean Band usage:       (in: 4721.56 Kb/s)     (out: 4769.87 Kb/s)
Mean Band usage:       (in: 4480.42 Kb/s)     (out: 4525.02 Kb/s)
Mean Band usage:       (in: 4721.69 Kb/s)     (out: 4767.46 Kb/s)
Mean Band usage:       (in: 4492.29 Kb/s)     (out: 4534.72 Kb/s)
Mean Band usage:       (in: 4721.74 Kb/s)     (out: 4765.22 Kb/s)
Mean Band usage:       (in: 4502.88 Kb/s)     (out: 4543.34 Kb/s)
Mean Band usage:       (in: 4721.79 Kb/s)     (out: 4763.2 Kb/s)
Mean Band usage:       (in: 4512.79 Kb/s)     (out: 4551.45 Kb/s)
Mean Band usage:       (in: 4721.93 Kb/s)     (out: 4761.41 Kb/s)
Mean Band usage:       (in: 4521.77 Kb/s)     (out: 4558.78 Kb/s)
Mean Band usage:       (in: 4525.98 Kb/s)     (out: 4562.22 Kb/s)
Mean Band usage:       (in: 4529.94 Kb/s)     (out: 4565.44 Kb/s)
```

### 4.1.9 Bandlimit 100.000b/sec

```
Speed = 1000, PacketSize = 1000, Pattern = 4289331370
Mean Band usage:       (in: 285714 Kb/s)      (out: 5.05714e+07 Kb/s)
Mean Band usage:       (in: 101.611 Kb/s)     (out: 705.299 Kb/s)
Mean Band usage:       (in: 100.614 Kb/s)     (out: 367.59 Kb/s)
Mean Band usage:       (in: 97.6249 Kb/s)     (out: 934.409 Kb/s)
Mean Band usage:       (in: 97.6247 Kb/s)     (out: 703.296 Kb/s)
Mean Band usage:       (in: 90.8508 Kb/s)     (out: 1183.85 Kb/s)
Mean Band usage:       (in: 91.9797 Kb/s)     (out: 986.54 Kb/s)
Mean Band usage:       (in: 87.0937 Kb/s)     (out: 850.159 Kb/s)
Mean Band usage:       (in: 88.41 Kb/s)       (out: 756.093 Kb/s)
Mean Band usage:       (in: 89.4339 Kb/s)     (out: 682.929 Kb/s)
Mean Band usage:       (in: 90.2529 Kb/s)     (out: 624.399 Kb/s)
Mean Band usage:       (in: 87.1195 Kb/s)     (out: 1801.8 Kb/s)
Mean Band usage:       (in: 84.3423 Kb/s)     (out: 3701.1 Kb/s)
Mean Band usage:       (in: 82.6054 Kb/s)     (out: 4937.93 Kb/s)
Mean Band usage:       (in: 80.9743 Kb/s)     (out: 6096.13 Kb/s)
Mean Band usage:       (in: 78.2325 Kb/s)     (out: 7670.77 Kb/s)
Mean Band usage:       (in: 76.705 Kb/s)      (out: 8667.17 Kb/s)
Mean Band usage:       (in: 74.8885 Kb/s)     (out: 9780.97 Kb/s)
Mean Band usage:       (in: 73.3844 Kb/s)     (out: 10694.8 Kb/s)
Mean Band usage:       (in: 72.2485 Kb/s)     (out: 11430 Kb/s)
Mean Band usage:       (in: 70.9272 Kb/s)     (out: 12274.5 Kb/s)
Mean Band usage:       (in: 70.1113 Kb/s)     (out: 12761.1 Kb/s)
Mean Band usage:       (in: 68.7356 Kb/s)     (out: 13504.3 Kb/s)
Mean Band usage:       (in: 68.2592 Kb/s)     (out: 13871.9 Kb/s)
Mean Band usage:       (in: 67.1584 Kb/s)     (out: 14499.1 Kb/s)
Mean Band usage:       (in: 66.6237 Kb/s)     (out: 14879.2 Kb/s)
```

```
Mean Band usage:       (in: 65.8237 Kb/s)     (out: 15358.2 Kb/s)
Mean Band usage:       (in: 65.1568 Kb/s)     (out: 15781.2 Kb/s)
Mean Band usage:       (in: 64.7509 Kb/s)     (out: 16073.8 Kb/s)
Mean Band usage:       (in: 64.0296 Kb/s)     (out: 16503.6 Kb/s)
Mean Band usage:       (in: 63.7548 Kb/s)     (out: 16686.6 Kb/s)
Mean Band usage:       (in: 63.0478 Kb/s)     (out: 17137.1 Kb/s)
Mean Band usage:       (in: 62.8831 Kb/s)     (out: 17269.6 Kb/s)
Mean Band usage:       (in: 62.185 Kb/s)      (out: 17667.7 Kb/s)
Mean Band usage:       (in: 62.0554 Kb/s)     (out: 17803.1 Kb/s)
Mean Band usage:       (in: 61.4778 Kb/s)     (out: 18127.1 Kb/s)
Mean Band usage:       (in: 61.3197 Kb/s)     (out: 18269 Kb/s)
Mean Band usage:       (in: 60.847 Kb/s)      (out: 18538.7 Kb/s)
Mean Band usage:       (in: 60.6614 Kb/s)     (out: 18682.7 Kb/s)
Mean Band usage:       (in: 60.2809 Kb/s)     (out: 18909.2 Kb/s)
Mean Band usage:       (in: 60.0191 Kb/s)     (out: 19077 Kb/s)
Mean Band usage:       (in: 59.7701 Kb/s)     (out: 19243.1 Kb/s)
Mean Band usage:       (in: 59.4855 Kb/s)     (out: 19427.7 Kb/s)
Mean Band usage:       (in: 59.3067 Kb/s)     (out: 19545.9 Kb/s)
Mean Band usage:       (in: 59.0456 Kb/s)     (out: 19749.8 Kb/s)
Mean Band usage:       (in: 58.9289 Kb/s)     (out: 19819.5 Kb/s)
Mean Band usage:       (in: 58.5573 Kb/s)     (out: 20052.8 Kb/s)
```

## 4.2 Conclusioni

Come si vede dai risultati presentati, il software realizzato riesce a simulare un canale affetto da rumore.
In particolare possono essere simulati, tramite regole definite dall'utente, la perdita di pacchetti e l'introduzione di errori secondo modelli di probabilità.
Sono stati inoltre realizzati meccanismi che permettono l'introduzione di limiti di velocità a una connessione.