POLITECNICO DI TORINO

Facoltà di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Analisi statistica di traffico sulla rete Internet

Relatori:
Prof. Marco Ajmone Marsan
Dott. Marco Mellia
Dott.ssa Michela Meo

Candidato:
Andrea Carpani

Maggio 2001

Sommario

Negli ultimi anni si è avuto un forte incremento dell'importanza di Internet e, assieme a questo, un aumento esponenziale del numero di utenti collegati alla rete. Dal punto di vista della rete sono stati aggiornati e migliorati i protocolli, definiti all'inizio degli anni '80, che sono alla base dello scambio dei dati. Per garantire, però, un livello di efficienza adatto a sostenere lo sviluppo ulteriore di Internet è opportuno verificare come si comportano i protocolli nati quando Internet era poco più di un esperimento a livello universitario, in una rete complessa in cui gli utenti si aspettano una elevata qualità del servizio.

Oggetto del presente lavoro è lo sviluppo di un software di analisi di traffico TCP su rete Internet e il suo successivo utilizzo per lo studio del traffico sul link di accesso alla rete Internet del Politecnico di Torino.

L'obbiettivo della prima parte del lavoro è stato quello di catturare tracce di dati rappresentativi dell'intera gamma di utilizzo della rete Internet da parte degli utenti del Politecnico. A tale scopo, tramite un PC collegato allo stesso tratto di rete del router di accesso ad Internet dell'Ateneo, sono stati catturati dati relativi al traffico delle date che vanno dal 29 maggio alla fine di giugno 2000 e dal 20 gennaio al 11 febbraio 2001. A caratterizzare i due periodi non vi sono soltanto i nove mesi di distanza, che permettono di confrontare l'evoluzione dei servizi usati, ma anche una diversa configurazione di un backbone principale, in particolare del collegamento tra il GARR (rete universitaria della quale il Politecnico fa parte) e gli Stati Uniti, cresciuto da gli iniziali 45 Mbps agli attuali 622 Mbps. Questo, unito alla forte caratterizzazione degli utenti del Politecnico verso la consultazione di siti statunitensi, ha costituito un considerevole miglioramento delle prestazioni

della rete Internet.

Parallelamente a questo lavoro di raccolta, partendo da un software preesistente e ampliandolo, è stato sviluppato il programma di analisi `tcpstat` in grado di esaminare le tracce di dati ricostruendone la storia delle connessioni. La velocità di esecuzione di `tcpstat` e il moderato consumo di memoria ne permettono l'utilizzo sia, come nel nostro caso, per l'elaborazione di dati precedentemente raccolti, sia per l'esame in tempo reale di un link di capacità paragonabile a quella dell'Ateneo. `tcpstat` è in grado di fornire statistiche a livello di pacchetto IP e TCP permettendo un'analisi statistica dell'utilizzo di questi protocolli. È, inoltre, in grado di ricostruire l'evoluzione delle connessioni TCP e di analizzarne caratteristiche quali perdite di segmenti, ritrasmissioni e ritardi. Il tutto viene effettuato elaborando tracce di traffico contenenti unicamente gli header dei singoli pacchetti, accorgimento che si è reso necessario per garantire un esame non intrusivo e rispettoso della privacy dell'utenza della rete.

Una prima parte dell'analisi delle tracce è dedicata alle statistiche a livello IP elaborate dal programma: si tratta di un'analisi della distribuzione degli indirizzi rispetto al numero di pacchetti di cui sono destinatari e della percentuale di utilizzo dei protocolli trasportati da IP. Alcune caratteristiche del protocollo, in particolare il Time to Live, permettono di ricavare interessanti dati sulla posizione reciproca tra la rete dell'Ateneo e gli host contattati sulla rete Internet. Le misurazioni a questo livello si concludono con l'analisi dei dati relativi alla lunghezza dei pacchetti IP che conferma, come già noto in letteratura, una predominanza di pacchetti di dimensioni o molto piccole o della massima dimensione ammessa.

Il confronto dei risultati raccolti durante i due periodi temporali non evidenzia particolari differenze ad eccezione di una maggiore quantità di traffico generata, dovuta all'incrementata capacità di rete. Questa osservazione conferma l'ipotesi spesso fatta, che la disponibilità di risorse spinge gli utenti ad usarle.

Il traffico TCP rappresenta oltre il 90% del volume in Internet, pertanto un suo studio è di fondamentale importanza nell'esame delle prestazioni della rete.

La seconda parte dell'analisi delle tracce, quindi, è dedicata ai risultati

delle misurazioni a livello TCP distinguendo tra analisi di protocollo e analisi a livello di flusso. Con le prime si intende esaminare come venga effettivamente utilizzato il protocollo dai vari stack TCP con particolare riferimento al valore dell'*MSS* e alla percentuale di utilizzo di opzioni facoltative come *Sack*, *Timestamp* e *Window Scale*. Si vedrà come queste opzioni, pensate per migliorare l'efficienza del protocollo siano in realtà poco sfruttate.

Le statistiche TCP a livello di flussi di dati si propongono due scopi: in prima istanza si vuole descrivere la tipologia di connessioni che si incontrano in rete, esaminandone la lunghezza, la durata e l'asimmetria in termini di numero segmenti e quantità di dati scambiati. Un esame complessivo dei dati in tale ottica ha dimostrato come gran parte del traffico Internet sia composto da connessioni HTTP e come queste si basino sullo scambio di una notevole quantità di dati di piccole dimensioni in direzione server-client. Osservando invece il numero di pacchetti scambiati dai vari servizi, si evince come alcuni protocolli generino uno scambio elevato di pacchetti per ogni flusso, si veda ad esempio il protocollo FTP, situazione, questa, già nota in letteratura.

In maniera innovativa si è cercato di ottenere una descrizione del comportamento del protocollo sul campo esaminando parametri come finestre di congestione, finestre di ricezione, pacchetti fuori sequenza e pacchetti duplicati. Sfruttando le ipotesi con cui sono state prelevate le tracce dati è possibile ricavare parametri importanti per il funzionamento e la caratterizzazione dei flussi TCP, come la probabilità di perdita di dati e la sua distribuzione su più pacchetti. Analogamente uno studio dei pacchetti duplicati permette caratterizzare da un lato le ritrasmissioni di pacchetti inutili e dall'altro le percentuali di perdita.

Lo studio di TCP e dei parametri che ne caratterizzano il funzionamento, è di fondamentale importanza per conoscere e poter migliorare il protocollo. Il presente lavoro si propone come un primo passo in questa direzione e come punto di partenza per ulteriori futuri approfondimenti.

Ringraziamenti

Al termine del lavoro desidero ringraziare il dott. Marco Mellia e il dott. Renato Lo Cigno per il valido contributo tecnico e la disponibilità che mi hanno fornito in tutti questi mesi. Desidero inoltre ringraziare il Ce.S.I.T. per la gentile collaborazione e per l'ospitalità offerta.

Un ringraziamento particolare va ai miei genitori che con la loro pazienza e il loro affetto mi hanno sempre sostenuto, alla nonna Anna per i saggi consigli e a mia sorella per l'appoggio generazionale.

Indice

Capitolo 1

Introduzione

In questo capitolo si introdurranno brevemente i concetti alla base del funzionamento della rete Internet e verranno illustrate le motivazioni del presente lavoro. Verranno inoltre presentati riferimenti a studi precedenti e strumenti utili per la cattura e lo studio del traffico.

1.1 Motivazioni

La suite di protocolli TCP/IP permette la comunicazione tra computer di tutte le dimensioni, con hardware differenti sui quali girano sistemi operativi completamente diversi. Il fatto è tanto più notevole se si pensa che il progetto ha superato le più rosee aspettative: nato alla fine degli anni '60 come ricerca finanziata dal governo statunitense nell'ambito della trasmissione di dati su reti a commutazione di pacchetto, dopo 30 anni è diventato il mezzo più diffuso di comunicazione per la trasmissione dati. Si tratta di un sistema completamente aperto in quanto la definizione dei protocolli e la descrizione di molte implementazioni sono liberamente disponibili ad un costo irrisorio se non addirittura gratuitamente. Questa suite è alla base della wide area network chiamata Internet alla quale sono collegati milioni di utenti da ogni parte del globo.

Prima dell'intensa crescita della navigazione sul web, applicazioni quali FTP, mail e news dominavano su Internet. Per queste applicazioni parametri

di prestazione orientati all'utente, come ad esempio il tempo di risposta, erano decisamente meno importanti di quanto non lo siano adesso. Col passare degli anni e con la crescita esponenziale del numero di computer collegati alla rete, i protocolli hanno mostrato tutta la loro elasticità adattandosi a veicolare traffico in quantità sempre maggiori su reti spesso congestionate facendo fronte delle mutate esigenze di un'utenza orientata sempre più verso la navigazione e l'interazione in tempo reale.

Per permettere a IP e TCP di assecondare questi cambiamenti è opportuno un continuo lavoro di integrazione e di aggiornamento unito ad una approfondita conoscenza del loro funzionamento su una rete complessa in cui un'elevata qualità del servizio è requisito fondamentale.

In quest'ottica si inserisce il presente studio volto a fornire strumenti necessari ad una accurata analisi sia del comportamento degli utilizzatori della rete, sia dei protocolli che ne sono alla base.

1.2 Studi precedenti

Data la predominanza del traffico web su Internet molti studi si sono a lungo soffermati sulla sua analisi sia dal punto di vista della rete (connessioni tra il browser e il server, bytes trasmessi) che da quello della descrizione dell'utenza. Non si è a conoscenza di studi che, partendo da dati reali di tracce raccolte, abbiano cercato di ricostruire e analizzare il comportamento di TCP.

Prima della crescita esplosiva del web sono stati sviluppati due importanti lavori incentrati sulla caratterizzazione di modelli di traffico; sitratta dei lavori di Danzig e altri [24, 19, 18] e da Paxons e Floyd [20, 28]. Vi sono stati, in tempi più recenti, molti studi volti a definire l'utilizzo della rete sempre in relazione alla navigazione: si tratta di raccolte di dati utili alla calibrazione di software di *caching* e *content delivery* [5, 11, 1]. In questi casi i risultati sono per lo più incentrati sull'elaborazione di tracce e log raccolti presso proxy o servers http; utili, quindi, per una caratterizzazione delle abitudini di navigazione degli utenti, ma poco adatte agli scopi del presente lavoro. I dati raccolti a livello di rete riguardano unicamente l'interazione tra web

browsers e i server, e parametri quali frequenza di connessione, tempistiche e dimensioni delle richieste.

I software generatori di traffico web maggiormente usati attualmente, si basano su dati rilevati in seguito a due progetti di raccolta innovativi per il loro periodo: si tratta dei lavori di Mah [14] e di Crovella e altri[17, 16, 13, 6]. I risultati di questi studi sono alla base di alcuni generatori di traffico integrati con il simulatore di reti *ns* [10] utilizzato, ad esempio, in [29, 27]. Questi modelli sono stati anche utilizzati nella generazione di traffico web in ambiente di laboratorio [21, 15]. In entrambe i casi, tuttavia, il bacino di utenza della rete non era sufficientemente variegato da poter essere considerato rappresentativo di Internet e le tracce erano relativamente brevi. Si consideri inoltre come essendo i dati relativi al 1995 in un caso e al 1995 e 1998 nell'altro, si possano considerare, di fatto, obsoleti.

In tempi più recenti sono cinque i progetti, di cui si sia a conoscenza, concernenti l'analisi di tracce su larga scala. Si tratta, in molti casi, di ricerche indirizzate in maniera particolare sul protocollo HTTP.

Gribble e Brewer [25] nel 1996, partendo da 45 giorni di tracce raccolte sul link del campus dell'Università di Berkley, hanno recuperato dati utili allo sviluppo di servizi di web caching (località delle pagine, cache control headers). Balakrishnan e altri [7], hanno raccolto tracce relative al traffico del server del sito ufficiale delle Olimpiadi del 1996 per uno studio sul miglioramento prestazioni di TCP utilizzato per il trasporto di traffico web. Lo studio prende in considerazione aspetti di *loss recovery* compressione degli ACK e bottlenecks.

Uno studio di Cleveland e altri, [26], ha utilizzato 23 milioni di connessioni HTTP raccolte su un arco di tempo che va da novembre 1998 a luglio 1999 sul link di collegamento a Internet dei Bell Labs. I dati sono stati utilizzati per lo sviluppo di un modello statistico per la generazione di tempistiche di connessioni TCP.

Alcuni ricercatori dell'Università di Washington hanno raccolto tracce relative al traffico sullo switch principale dell'Ateneo in maniera continuativa per una settimana. I dati sono stati elaborati da un software proprietario in grado di identificare connessioni HTTP e di estrarne gli headers. I risultati sono apparsi in due studi sul web proxy caching [30, 2].

Un'altro studio sulla ricostruzione del traffico HTTP a partire da tracce raccolte è quello realizzato da Feldman [3] alla fine del 1999. Si tratta di un interessante analisi dei problemi riscontrati nella ricostruzione del flusso di dati sia a livello TCP che a livello HTTP.

1.3 Strumenti utili per la cattura e l'analisi del traffico

Gli strumenti utilizzati sia in fase di sviluppo del software che in fase di raccolta dei dati sono tutti liberamente distribuibili e modificabili. La scelta di usare questo tipo di programmi nasce da due esigenze fondamentali in campo accademico: la necessità di poter elaborare il codice e comprenderne il funzionamento senza dover dipendere da terze parti e la libertà di poter redistribuire le eventuali modifiche del codice.

Per la raccolta dei dati si è utilizzato `tcpdump 3.4` scritto da Steve McCanne, Craig Leres e Van Jacobson nel 1991 e continuamente aggiornato (*http://www.tcpdump.org*). Si tratta di uno dei software più autorevoli per la cattura e l'esame di pacchetti sulla rete. Per le chiamate a funzione di basso livello `tcpdump` si appoggia sulla libreria `libpcap 0.4` (*http://www.tcpdump.org*) che fornisce primitive di accesso alla rete su più architetture in maniera trasparente .

Per lo sviluppo di `tcpstat`, il software alla base del presente studio, si è partiti da `tcptrace 5.2` (*http//www.tcptrace.org*) scritto da Shawn Ostermann. Si tratta di un programma in grado di analizzare le tracce prodotte da `tcpdump` ed estrarne statistiche a livello TCP fornendo in output dati relativi alle connessioni e grafici rappresentativi di variabili quali throughput e round trip time.

Altro programma in grado di sfruttare le tracce di `tcpdump` è `tcpanaly`: si tratta di un software in grado di analizzare l'implementazione TCP, esaminando tracce di dati, confrontandolo con un database interno. Una volta riconosciuta l'implementazione è in grado di descrivere comportamento del protocollo in relazione alle tempistiche di invio dei pacchetti.

Per una rappresentazione grafica del flusso di pacchetti sulla rete, si è

utilizzato `ethereal` (*http://www.ethereal.com*) un analizzatore di protocollo liberamente distribuibile che, appoggiandosi a `libpcap`, permette sia di catturare i dati in tempo reale, sia di esaminare una traccia precedentemente raccolta con `tcpdump`.

È possibile ottenere una descrizione dettagliata in tempo reale del traffico sulla rete con `ntop` (*http://www-serra.unipi.it/~ntop/*) uno strumento in grado di illustrare l'occupazione della banda da parte delle varie connessioni e dei protocolli più diffusi.

Capitolo 2

Il software `tcpstat`

Il software `tcpstat` permette di ottenere dati statistici sull'utilizzo dei protocolli TCP e IP a partire dal capture di traffico di rete. Come si vedrà in seguito l'analisi di questi dati consente uno studio dell'utilizzo e dell'efficienza della rete in esame.

`tcpstat` nasce dallo sviluppo di `tcptrace` versione 5.2.1 il cui autore, Shawn Ostermann, ha rilasciato il codice sorgente con una licenza che ne permette la modifica e la redistribuzione a patto che la licenza rimanga inalterata.

2.1 Le tracce

Il programma non è in grado, al momento attuale, di catturare autonomamente la traccia da esaminare, ma prende in ingresso files dump prodotti dai seguenti programmi:

- tcpdump , grazie alla libreria pcap,
- snoop, diffuso su macchine Sun
- Macintosh Etherpeek

`tcpstat` è in grado di esaminare una qualunque traccia, ma il software è stato pensato per analizzare il traffico di rete in uscita da una LAN verso Internet, a condizione di essere vicini alla sorgente o alla destinazione del traffico. Ciò

permette di considerare uno dei due tratti di rete *perfetto* e di ipotizzare che su quel di esso non vi siano stati riodinamenti o perdite di pacchetti. La situazione migliore si ha quando la cattura del traffico avviene sullo stesso tratto di rete visto dal lato interno del router di uscita della LAN.

2.2 Guida all'uso

Il programma va richiamato con la seguente riga di comando

```
tcpstat [-n] -a<file con reti interne> traccia
```

L'ultimo parametro sulla riga di comando è il nome del dump file da esaminare che deve essere in uno dei formati elencati in precedenza.

Il comando *-a* è obbligatorio e il suo parametro deve essere il nome di un file di testo nel quale vanno elencate le sottoreti che vanno considerate interne durante l'analisi. Il file, in formato ASCII, deve contenere una serie di righe, terminate da *newline*, nel formato

```
indirizzo
netmask
indirizzo
netmask
...
```

Per ogni indirizzo viene identificata, in base alla netmask, una sottorete i cui indirizzi IP vanno considerati interni. Viene di seguito riportato un esempio di file:

```
130.192.0.0
255.255.0.0
193.42.134.0
255.255.255.0
193.204.112.0
255.255.255.0
193.205.59.0
255.255.255.0
```

Il programma attualmente ammette l'uso di un massimo di 20 sottoreti interne.

Lo switch opzionale `-n` impone al programma di non risolvere il nome degli host e di presentarne l'indirizzo IP numerico.

2.3 Output

Durante l'elaborazione dei dati `tcpstat` stampa sullo *standard output* alcuni messaggi relativi allo stato di esecuzione, utili per tener traccia del progredire delle operazioni[1]. L'output principale del programma, invece, è scritto direttamente in file separati. Per ogni traccia in ingresso tutti i dati relativi si trovano in una directory, chiamata directory base, che il programma crea all'avvio. Tale sotto-directory ha come nome lo stesso nome del file in ingresso con il suffisso `.out`. In questa directory, alla fine dell'elaborazione, si troveranno i seguenti files:

`addresses:` in questo file vengono elencati tutti gli IP di destinazione dei pacchetti visti, con l'indicazione del numero di pacchetti.

`log_complete:` un elenco di tutte le connessioni TCP chiuse con il corretto handshaking finale o con un reset. Per ognuna di queste connessioni vengono riportati tutti i dati calcolati dal programma.

`log_nocomplete:` in questo files vengono riportati i dati calcolati dal programma relativi alle connessioni che non rientrano nella categoria precedente in particolare quelle che dopo un intervallo di tempo T non hanno visto più pacchetti nelle tracce; di default T è pari a 30 minuti. Questo per evitare di occupare memoria inutile durante le analisi dei dati.

Parte dell'output viene scritto da `tcpstat` in una serie di sottodirectory ogni volta che il programma arriva ad analizzare un quarto d'ora di traffico. Le sottodirectory che contengono questi dati hanno il nome sequenziale `00`, `01`, `02`, etc ...

[1] Per maggiori informazioni circa questo output si veda l'Appendice A.

Alla fine dell'elaborazione in ognuna di queste directory si troveranno i file relativi a statistiche generali, statistiche sul protocollo IP, statistiche sul protocollo TCP e statistiche relative alle perdite. La mole di dati prodotti da ogni elaborazione è notevole: in ogni sottodirectory, in fatti, vengono scritti 77 file. Per facilitarne la consultazione si è cercato di usare nomi il più possibile autoesplicativi seguendo alcune semplici regole:

- il prefisso `ip_` e quello `tcp_` identificano statistiche relative al protocollo IP e TCP rispettivamente.
- i suffissi `_ent` e `_usc` identificano rispettivamente statistiche relative a connessioni in ingresso e in uscita rispetto alla divisione in reti interne e esterne.
- i suffissi `_a2b` e `_b2a` identificano statistiche relative a semi-connessioni rispettivamente provenienti e dirette verso l'host che ha iniziato la connessione.
- `_b_` identifica statistiche espresse in bytes.
- `_p_` identifica statistiche espresse in pacchetti.

Di seguito vengono elencati i vari files riportati in tabella per una migliore consultazione.

Statistiche generali:

`protocol:` numero di pacchetti IP suddivisi per protocollo.

`not_id_p:` (*not identified packet*) numero di pacchetti TCP non appartenenti a nessuna connessione riconosciuta dal programma, dovuti, ad esempio, a connessioni già iniziate al momento dell'inizio della cattura.

Statistiche relative al protocollo IP:

Misura	Suffisso	Direzione	Esempio
Lunghezza pacchetti	`len`	ent, usc, cumulativa	`ip_len_usc`
Time to Live	`ttl`	ent, usc, cumulativa	`ip_ttl`
Type of Service	`tos`	ent, usc, cumulativa	`ip_tos_ent`

Statistiche relative al protocollo TCP:

Misura	Suffisso	Direzione	Esempio
Servizio	port	n.d.	tcp_port
Lunghezza della connessione in byte (0 - 50KB)	`cl_b_s`	ent, usc, a2b, b2a	`tcp_cl_b_l_a2b`
Lunghezza della connessione in byte (50KB - 50 MB)	`cl_b_l`	ent, usc, a2b, b2a	`tcp_cl_b_s_a2b`
Lunghezza della connessione in pacchetti	`cl_p`	ent, usc, a2b, b2a	`tcp_cl_p_a2b`
Dimensione della congestion window	`cwnd`	-	`tcp_cwnd`
Maximim Segment Size (MSS)	`mss`	a,b,used	`tcp_mss_a`
Opzioni TCP	`opts`	-	`tcp_opts`
Dimensione della finestra di ricezione	`rwnd`	ini, max, avg	`tcp_rwnd_ini`

Statistiche relative al Round Trip Time:

Misura	Suffisso	Direzione	Esempio
Numero di pacchetti validi per il conteggio del RTT	`cnt`	ent, usc, a2b, b2a	`rtt_cnt_ent`
RTT massimo	`max`	ent, usc, a2b, b2a	`rtt_max_usc`
RTT minimo	`min`	ent, usc, a2b, b2a	`rtt_min_usc`
RTT medio	`avg`	ent, usc, a2b, b2a	`rtt_avg_a2b`
Deviazione standard	`stdev`	ent, usc, a2b, b2a	`rtt_stdev_b2a`

Statistiche relative alle perdite:

Misura	Suffisso	Direzione	Esempio
Burst di pacchetti Out Of Order	`ooob`	ent, usc, a2b, b2a	`ooob_a2b`
Burst di pacchetti Out Of Order con l'opzione SYN o FIN	`ooob_sf`	ent, usc, a2b, b2a	`ooob_sf_a2b`
Burst di pacchetti Out Of Order in corrispondenza di una finestra di ricezione pari a 0	`ooob_rwndz`	ent, usc, a2b, b2a	`ooob_rwndz_usc`
Burst di pacchetti duplicati	`dupb`	ent, usc, a2b, b2a	`dupb_a2b`
Burst di pacchetti duplicati con l'opzione SYN o FIN	`dupb_sf`	ent, usc, a2b, b2a	`dupb_sf_b2a`
Burst di pacchetti duplicati in corrispondenza di una finestra di ricezione pari a 0	`dupb_rwndz`	ent, usc, a2b, b2a	`dupb_rwndz_a2b`

2.4 Lo scenario di misura

Il software `tcpstat` nasce dall'esigenza di avere uno strumento adatto a descrivere il traffico tra una sottorete e il resto di Internet. Nel nostro caso la sottorete in questione corrisponde alla rete interna del Politecnico di Torino. Il posto migliore, dal punto di vista topologico, per osservare questo traffico è il link che collega la sottorete all'esterno. In Figura 2.1 è possibile osservare la struttura della rete nel nostro caso. Si noti come la rete dell'Ateneo comprenda sottoreti appartenenti alle sedi decentrate di Alessandria e Vercelli. Per una descrizione dettagliata della struttura della rete dell'Ateneo si faccia riferimento all'Appendice C.

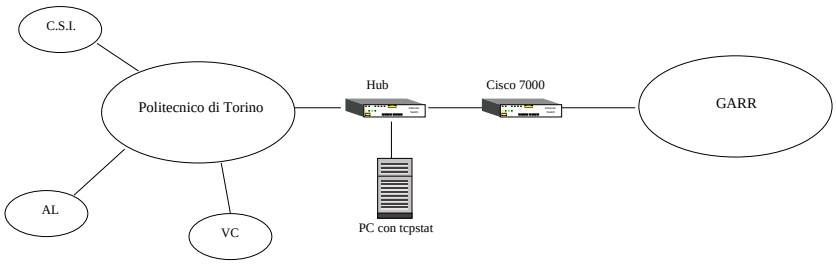


Figura 2.1. Topologia di rete

Il software non esamina direttamente il link fisico, ma è in grado di leggere i dati catturati tramite `tcpdump`.

Nel nostro caso, per effettuare la cattura, è stato utilizzato un personal computer di fascia media (AMD K6 con 128 MB di RAM) dotato di scheda di rete collegata tramite un hub allo stesso link fisico visto dalla porta interna del router dell'Ateneo. Come sistema operativo è stato scelto Linux.

2.4.1 Le tracce

Le misurazioni sono state effettuate sul link in uscita del Politecnico di Torino in tre momenti diversi. Si sono scelti periodi diversi per poter esaminare differenti tipologie di rete. Il link in uscita preso in considerazione è quello che collega l'Ateneo alla rete universitaria italiana GARR. Durante l'anno 2000, il collegamento GARR-USA è stato ampliato in due fasi successive, progredendo dagli iniziali 45 Mbps agli attuali 622 Mbps passando per una

fase in cui la banda era di 155 Mbps. Questa evoluzione è particolarmente interessante nel nostro caso, in quanto il traffico Internet generato e ricevuto dal Politecnico è fortemente diretto verso siti statunitensi. La tabella seguente riassume i vari momenti nei quali le tracce sono state catturate e la banda disponibile sul tratto GARR USA nei vari momenti.

Blocco di dati	data	Mbps sul link GARR-USA	Dimensione delle tracce raccolte
A	1.6.2000 - 11.6.2000	46	80 GB
B	19.6.2000 - 9.7.2000	155	150 GB
C	19.1.2001-11.2.2001	622	162 GB

Nell'ultima colonna è stata indicata la dimensione delle tracce raccolte nei vari periodi. Nel presentare i risultati si confronteranno, ove interessante, i cambiamenti osservati durante l'evoluzione della topologia.

Capitolo 3

Statistiche IP

In questo capitolo illustreremo come le statistiche a livello IP effettuate dal software `tcpstat` possano tornare utili per capire come viene utilizzata la rete. In un primo momento verranno presentati i gruppi di tracce prese in considerazione. La diversa configurazione di un backbone principale nei vari casi permetterà di osservare l'evoluzione dell'utilizzo della rete stessa. Verrà poi mostrata la prima parte delle statistiche ottenibili con il software, in particolare quelle relative al layer IP.

3.1 Il protocollo IP

IP (Internet Protocol) è un protocollo di comunicazione su rete a commutazione di pacchetto che si situa al livello 3 (network) del modello ISO/OSI e permette la trasmissione di blocchi di dati, detti *datagrammi*, da una sorgente ad una destinazione. Lo scopo del protocollo è quello di fornire le funzioni necessarie per recapitare questo blocco fornendo funzioni di indirizzamento e di frammentazione e ricomposizione dei dati. IP non prevede meccanismi per aumentare la affidabilità del canale di comunicazione: è *inaffidabile* e non *orientato alle connessioni*.

Con *inaffidabile* si intende dire che non vi è garanzia che il pacchetto arrivi regolarmente a destinazione: il protocollo farà del suo meglio, ma nell'eventualità di un problema durante il percorso (per esempio un router

congestionato) il pacchetto viene semplicemente scartato e alla sorgente viene inviato messaggio ICMP.

Non *orientato alle connessioni* vuol dire che ogni datagramma IP non contiene informazioni sullo stato di pacchetti successivi o precedenti e viene trattato in maniera indipendente: seguendo due percorsi differenti due datagrammi IP possono essere consegnati alla destinazione in ordine diverso da quello con cui sono stati creati.

Eventuali esigenze in termini di affidabilità e ordinamento vanno delegate al livello superiore (ad esempio TCP).

3.1.1 Header IP

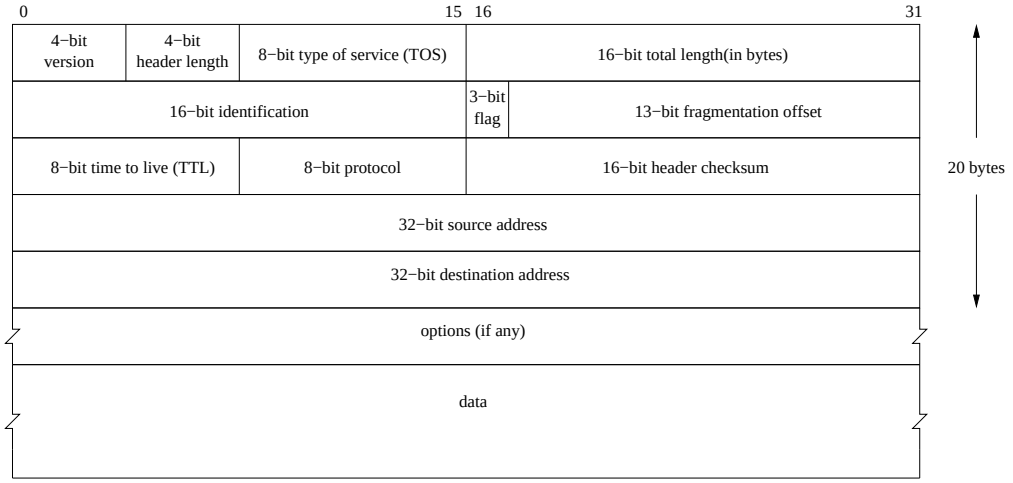


Figura 3.1. Datagramma IP

In Figura.3.1 viene mostrato il formato di un pacchetto IP [22]. La dimensione dell'header è di 20 bytes a meno di opzioni particolari e la lunghezza massima del pacchetto, indicata dai 16 bit del campo *length* è 65535 bytes. Benché sia tecnicamente possibile inviare pacchetti di tale dimensione essi verrebbero comunque frammentati dal layer di rete sottostante, per questo motivo la maggior parte delle implementazioni del protocollo usano come dimensione massima 8192 bytes.

L'attuale versione del protocollo è 4 e viene indicata nei 4 bit del campo *version*.

Il campo *header length* contiene 4 bit che indicano il numero di word da 32 bit presenti nell'header: se non vi sono opzioni presenti vale 5.

Gli 8 bit nell'header IP che definiscono il *type of service* [4] servono a indicare opzioni di routing alternative. Vengono settati dallo stack TCP del mittente e definiscono una serie di richieste fatte dall'applicazione. Possono venir usati dai router lungo il percorso per scegliere il cammino migliore in base a tali richieste. I primi tre bit (bit di precedenza) indicano l'importanza del pacchetto e ne aumentano la priorità. Sono utili, per esempio, in caso di congestione per permettere la circolazione dei pacchetti di congestion-control che altrimenti verrebbero bloccati dalla congestione stessa. I restanti quattro bit indicano il tipo di servizio e sono pensati per permettere al router di scegliere l'instradamento migliore per ogni pacchetto a seconda delle sue caratteristiche.

- *delay bit* (D) indica ai router di scegliere il percorso che minimizza il ritardo.
- *throughput bit* (T) indica ai router di scegliere il percorso che massimizza il throughput.
- *reliability bit* (R) indica ai router di scegliere il percorso che massimizza l'affidabilità.
- *cost bit* (C) indica ai router di scegliere il percorso che minimizza il costo economico.

I quattro bit di servizio vanno settati in esclusione.

Il campo *total length* indica la lunghezza totale del pacchetto espressa in bytes.

Il campo *identification* identifica univocamente ogni pacchetto inviato da un host e ha un ruolo importante nel processo di frammentazione e ricomposizione del datagramma durante percorso dalla sorgente alla destinazione.

Il campo *Time To Live* (*TTL*) impone un limite superiore al numero di router che il pacchetto può attraversare durante il percorso dalla sorgente alla

destinazione. Viene impostato in origine ad un valore dipendente dallo stack TCP (tipicamente 32, 62, 128 o 255) e decrementato ogni volta che passa attraverso un router. Il pacchetto il cui *TTL* raggiunge zero viene scartato e la sorgente viene avvisata con un pacchetto ICMP.

Il campo *protocol* contiene un indicazione del protocollo che, dal livello superiore, ha passato i dati ad IP.

Il campo *header checksum* viene calcolato in base ai soli bit dell'header IP tralasciando il resto del pacchetto: ciascuno dei protocolli che si affidano ad IP, infatti, ha un proprio controllo di integrità dei dati.

Ogni pacchetto IP contiene l'indirizzo della sorgente e quello della destinazione espressi ciascuno da 16 bit. I campi *source address* e *destination address* verranno descritti in Sezione 3.1.2.

Infine il campo *options* contiene una lista, di dimensione variabile, di informazioni aggiuntive che possono essere incluse nel datagramma. Le opzioni attualmente previste dal protocollo sono:

- restrizioni relative alla sicurezza (usate in campo militare)
- *record route*: ogni router attraversato dal pacchetto deve aggiungere il proprio indirizzo.
- *timestamp*: ogni router attraversato dal pacchetto deve aggiungere il proprio indirizzo e una timestamp.
- *loose source routing*: si tratta di un elenco di routers che vanno obbligatoriamente attraversati dal pacchetto.
- *strict source routing*: come nel caso precedente, ma il pacchetto non può attraversare routers che non siano elencati.

3.1.2 Indirizzi IP

Ad ogni interfaccia di rete di ogni macchina che si affaccia su Internet va assegnato un indirizzo IP: un numero su 32 bit che la contraddistingue univocamente. Gli indirizzi vengono di norma indicati con quattro numeri decimali, relativi ai singoli bytes, separati da punto. Il significato dei bit varia

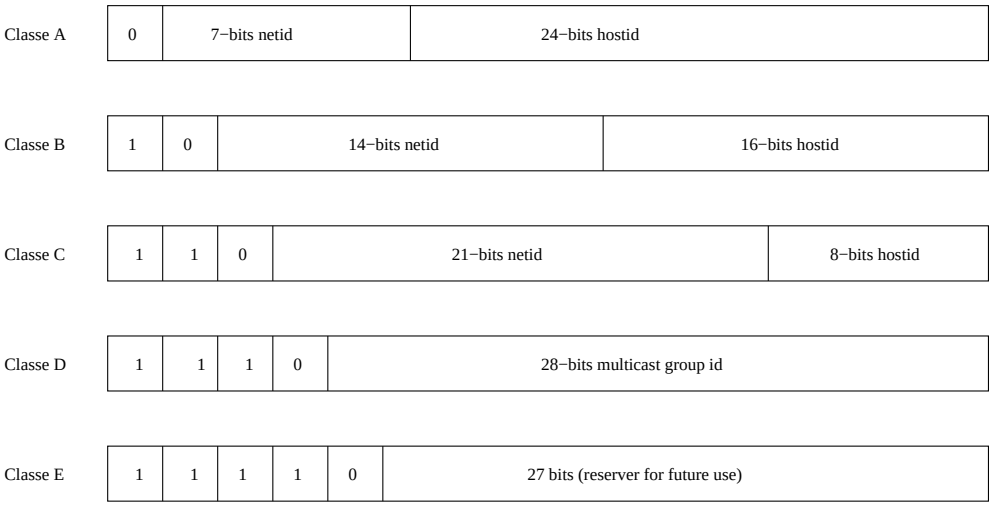


Figura 3.2. Classi di indirizzi IP.

Classe	Intervallo
A	da 0.0.0.0 a 127.255.255.255
B	da 128.0.0.0 a 191.255.255.255
C	da 192.0.0.0 a 223.255.255.255
D	da 224.0.0.0 a 239.255.255.255
E	da 240.0.0.0 a 247.255.255.255

Tabella 3.1. Intervalli degli indirizzi per le varie classi IP.

a seconda della classe alla quale l'indirizzo appartiene, secondo lo schema di Figura 3.2. Per poter conoscere la classe alla quale un indirizzo appartiene è sufficiente prendere in considerazione il valore del primo byte e confrontarlo con i valori in Tabella 3.1.

3.2 Analisi a livello IP

Le informazioni che il software `tcpstat` permette di ottenere a livelli IP sono classificabili in cinque categorie:

Indirizzi: numero di pacchetti per ogni indirizzo di destinazione.

Protocollo: numero di pacchetti raggruppati per protocollo usato (TCP, UDP, altro).

Type of Service: pacchetti raggruppati per type of service, in totale e suddivisi tra pacchetti in ingresso e in uscita.

Time to Live: pacchetti raggruppati per time to live, in totale e suddivisi tra pacchetti in ingresso e in uscita.

Lunghezza dei pacchetti: pacchetti raggruppati per lunghezza, in totale e suddivisi tra pacchetti in ingresso e in uscita.

3.2.1 Distribuzione degli indirizzi

Per facilitare l'analisi della distribuzione degli indirizzi in base alla quantità di pacchetti, è stato scritto il programma `get_addresses`[1] in grado di effettuare la risoluzione dei 200 indirizzi più contattati a partire dai risultati di `tcpstat`. Il programma si occupa, inoltre, di eliminare gli hosts interni all'Ateneo, permettendo così di avere una lista degli indirizzi verso i quali vi è maggior traffico. I primi 10 host più contattati nei periodi A e C sono elencati nelle Tabelle 3.2 e 3.3.

[1]Per maggiori informazioni si faccia riferimento alla descrizione del programma in Appendice B.2.

In tutti i periodi presi in considerazione l'host destinatario del maggior numero di pacchetti IP è *newsfeed.cineca.it*. Si tratta di uno dei maggiori news servers italiani che, come il server *news.polito.it*, fa parte del backbone responsabile della propagazione della gerarchia news italiana *it.**. L'indirizzo che appare come secondo nell'elenco, è il proxy *homer.cineca.it*. Questi due server, da soli, sono destinatari del 10% del traffico IP in uscita dall'Ateneo.

È interessante notare come non siano presenti nella lista motori di ricerca affermati come *Google*, *Altavista* o *Yahoo*. Si tratta di siti che, nonostante la grande quantità di accessi, non sono destinatari di grandi moli di dati.

In Figura 3.3 è stata evidenziata la distribuzione di indirizzi più contattati nei due periodi A e C. Si noti come, a distanza di 9 mesi, pur essendo gli host diversi, le distribuzioni siano simili, a parte l'aumento in valore assoluto dei contatti dovuto all'incremento della banda.

Host destinatario dei pacchetti	Numero di pacchetti
newsfeed.cineca.it	4306463
homer.cineca.it	2521750
163.162.4.11	2091547
giano.com.dist.unige.it	1331350
ns.studenti.torino.it	1312060
S140116.selcuk.edu.tr	1150656
255.255.255.255	832288
207.175.46.245	822906
colo00-167.xoom.com	754609
ftp1.sunet.se	751316

Tabella 3.2. Distribuzione dei primi 10 indirizzi più contattati nel primo periodo.

3.2.2 Utilizzo del protocollo

Nel grafico in Figura 3.4 è riportato il numero di pacchetti suddivisi per protocollo. Il grafico riporta il numero di pacchetti al minuto, misurato a intervalli di 15 minuti. La misura copre un arco temporale di una settimana presa a titolo di esempio. Si può notare come una notevole parte dei pacchetti, tra l'80% e il 90%, sia TCP. Il restante traffico UDP si può pensare

Host destinatario dei pacchetti	Numero di pacchetti
newsfeed.cineca.it	5601782
homer.cineca.it	2056089
213.167.195.41	1495639
adnetwork.kataweb.it	1345563
croci.unipi.it	1222341
buddy.i-plus.net	1209751
pc091.vpszk.bme.hu	1176500
www.repubblica.it	1002892
scastsrv3.shoutcast.com	992788
admedia.kataweb.it	989169

Tabella 3.3. Distribuzione dei primi 10 indirizzi più contattati nel terzo periodo.

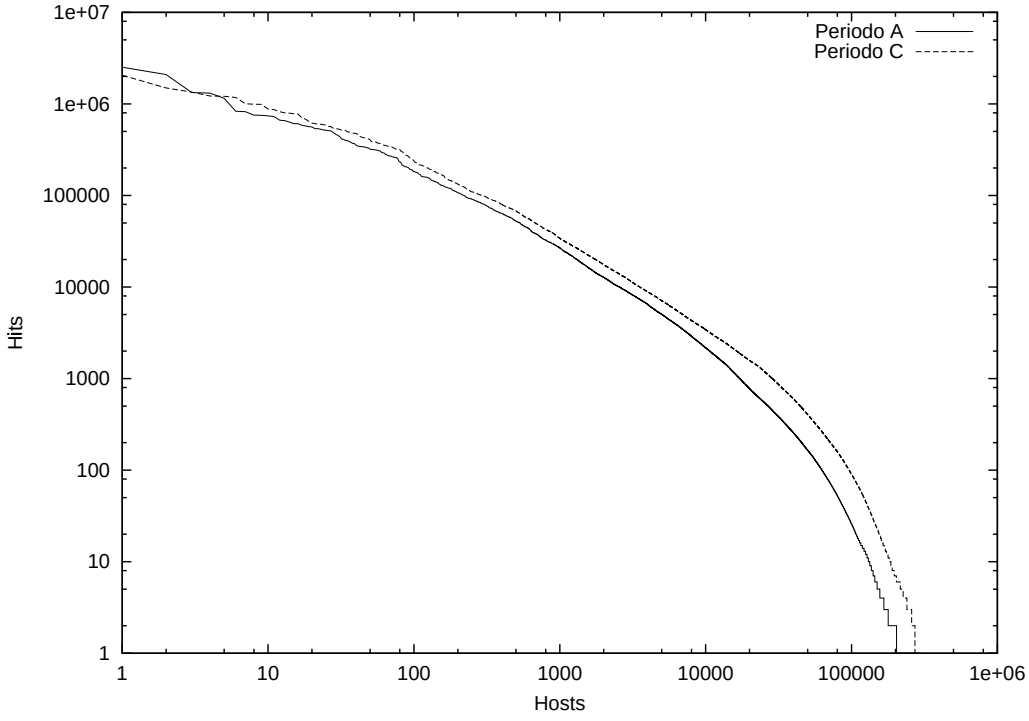


Figura 3.3. Distribuzione degli indirizzi contattati nei periodi A e C.

suddiviso principalmente tra richieste DNS e streaming di file multimediali. Una percentuale minima è stata classificata sotto *Other*. Come era facile prevedere la maggior parte del traffico avviene durante il giorno in particolare durante la settimana lavorativa; la sera e nei giorni festivi si riduce sensibilmente. Il flusso, nei momenti di maggior traffico, varia tra gli 80000 e i 100000 pacchetti al minuto.

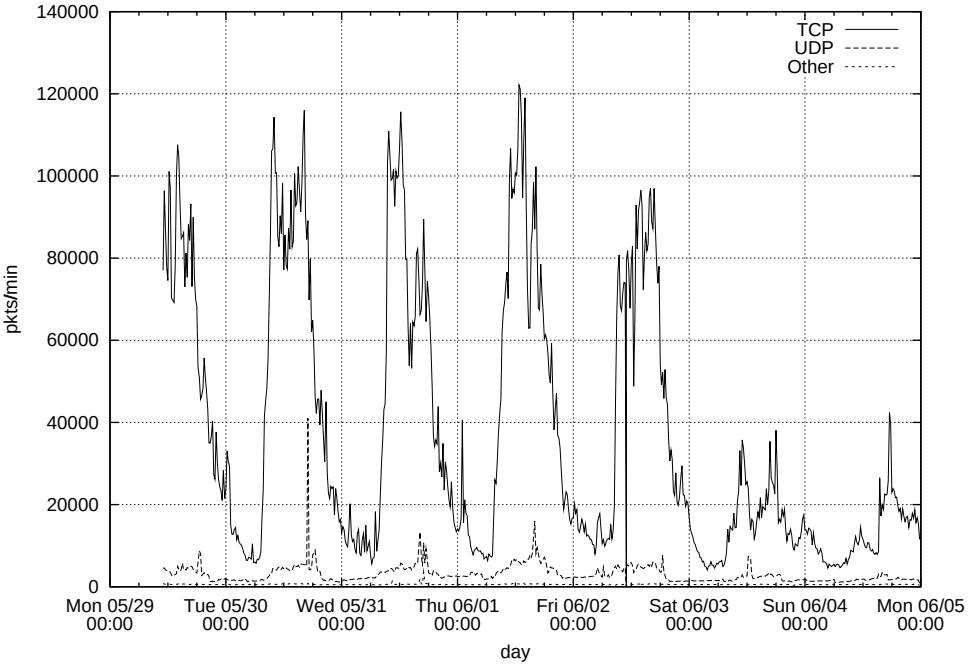


Figura 3.4. Utilizzo del protocollo nel periodo A

In Figura 3.5 sono rappresentati i dati relativi al traffico di una settimana nel periodo B: si tratta di dati catturati a partire dal decimo giorno dopo l'incremento di banda. A questo corrisponde un aumento del traffico nelle ore di punta, rispetto al caso precedente, che ora raggiunge picchi di 2.5 Milioni di pacchetti al minuto. Rispetto all'analisi precedente è meno visibile l'andamento periodico dovuto all'alternarsi di giorno e notte, mentre è sempre evidente la distribuzione del protocollo usato a favore di TCP.

Interessante è notare come, a partire da lunedì 3 luglio, la percentuale di

traffico UDP raggiunga livelli elevatissimi. Questo corrisponde a un fenomeno di flooding[2] del quale la rete informatica dell'Ateneo è stata involontaria protagonista durante le prime ore della giornata. Per comprendere ed illustrare il fenomeno è stato scritto il programma *udp-what* che permette di quantificare il traffico non UDP a partire dalla traccia catturata. Il risultato dell'esame dei dati del programma mostra, in Tabella 3.4, il tipo e la quantità di pacchetti non TCP nelle tracce prese in esame. Sono stati considerati i tracciati relativi ad un periodo di sei ore (dalle 22:00 fino alle 5:00) di tre giorni diversi di cui l'ultimo è quello durante il quale c'è stato l'attacco. Come si può notare la notte tra il 2 e il 3 luglio si registra un' anomala presenza di pacchetti legati ai protocolli chargen, icmp e frag.

30.6.2000		1.7.2000		2.7.2000	
domain	181240	domain	130057	chargen	36287820
udp	96034	udp	93959	icmp	9307932
igmp	32647	igmp	32421	frag	4299116
icmp	23393	icmp	21144	domain	162045
ntp	11085	ntp	12552	udp	54718
OSPF	8636	OSPF	8635	igmp	31663

Tabella 3.4. Traffico non TCP

Tornando all'analisi dei protocolli trasportati, la Figura 3.6 si riferisce ai dati raccolti nella settimana a cavallo tra gennaio e febbraio 2001. L'andamento del grafico è qualitativamente simile a quello di Figura 3.4 e 3.5 mentre va notato come in termini assoluti si registra un incremento nell'utilizzo della rete: durante i momenti di maggior utilizzo (le ore lavorative dal lunedì al venerdì) il flusso di pacchetti TCP varia tra 140000 e 160000 al minuto.

È di nuovo apprezzabile la differenza di utilizzo del canale tra giorno e notte, così come il predominio di utilizzo del protocollo TCP.

Le differenze tra il primo e l'ultimo periodo si vedono soprattutto in relazione alla quantità di banda utilizzata.

[2]Per flooding si intende un'anomala concentrazione di pacchetti dovuta non ad una effettiva esigenza di servizio, ma ad un tentativo doloso di saturazione della banda.

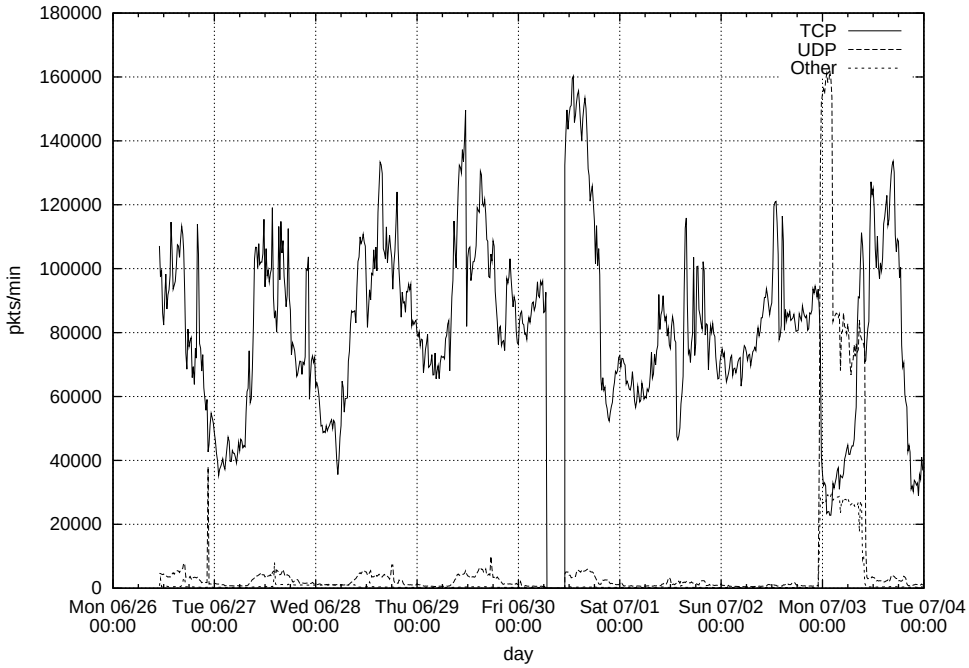


Figura 3.5. Utilizzo del protocollo nel periodo B

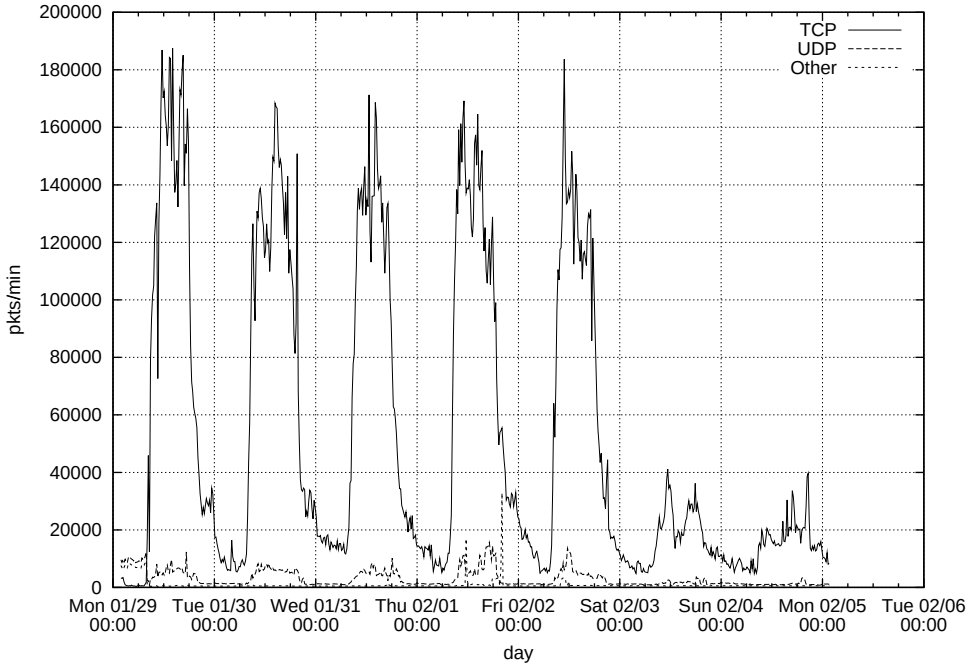


Figura 3.6. Utilizzo del protocollo nel periodo C

3.2.3 Type of Service

In tabella 3.5 sono riportate, per i tre periodi presi in considerazione, le percentuali di TOS [4] più frequenti. Come si può notare la grande maggioranza dei pacchetti ha un TOS normale e sono pochi i pacchetti che richiedono un trattamento preferenziale. Si notano differenze marginali nella distribuzione di TOS utilizzati nei tre periodi di analisi.

Type Of Service	Significato	Periodo A	Periodo B	Periodo C
0	Precedenza normale, servizio normale	90.5 %	95.2%	95.1%
16	Precedenza normale, min ritardo	5.6%	3.2%	2.8%
8	Precedenza normale, max throughput	1.9%	0.2%	0.8%
192	Precedenza alta, servizio normale	0.5%	0.2%	0.3%

Tabella 3.5. Percentuali di pacchetti per *TOS*.

3.2.4 Time To Live

L'analisi della distribuzione dei TTL dei pacchetti in ingresso permette di avere un'idea della distanza della rete dell'Ateneo dagli elaboratori che all'Ateneo accedono. Inoltre i pacchetti in uscita sono topologicamente vicini alla macchina che effettua le misurazioni: si tratta, in genere, di un uno o due hop di distanza. È quindi possibile avere una rappresentazione del tipo di macchine presenti all'interno dell'Ateneo, sapendo la relazione tra sistema operativo e TTL il cui valore di default è riportato in Tabella 3.6 per i sistemi operativi più diffusi.

In Figura 3.7 vengono mostrate le distribuzioni di TTL in ingresso e in uscita durante il primo periodo. Osservando i valori di TTL relativi ai pacchetti in uscita sono evidenti i picchi in corrispondenza dei sistemi operativi più diffusi: Microsoft Windows 95 (TTL 32), Linux (TTL 64), Microsoft

Sistema Operativo	TTL
Digital OSF/1	60
Linux	64
SunOS 2.x	255
SunOS 5.8	64
Windows 95	32
Windows 98	128
Windows 2000	128
Windows NT	128

Tabella 3.6. *Time To Live* iniziale dei principali sistemi operativi

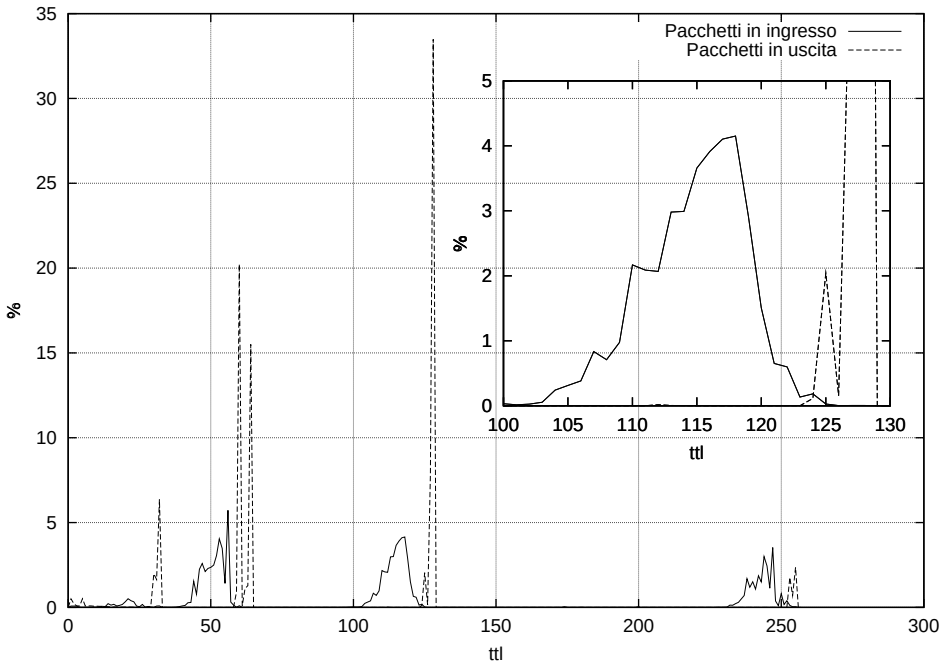


Figura 3.7. Distribuzione percentuale del TTL nel periodo A

Windows 95 e 2000 (TTL 128) e SunOS 2.x (TTL 255). È interessante notare il picco in corrispondenza del valore 60 dovuto ai pacchetti in uscita dai server pubblici principali del Politecnico: *www.polito.it*, *ftp.polito.it* e *news.polito.it*. Si tratta di macchine Digital il cui OSF/1 impone un TTL di 60.

Osservando la distribuzione dei *TTL* in ingresso, è possibile dedurre una distanza media di 9 o 10 hop per gli utenti Linux e Windows 98/2000 che accedono ai servizi del Politecnico. L'ingrandimento in Figura 3.7 è relativo all'intervallo tra i valori 100 e i 128, intervallo relativo al *TTL* usato di default da Microsoft Windows 98. È possibile notare come siano pochi i pacchetti in ingresso con un *TTL* compreso tra 123 e 128 questo perché prima di giungere sulla rete dell'Ateneo qualunque pacchetto deve attraversare almeno due o tre router di solo transito del backbone GARR che diminuiscono il valore di *TTL*.

Le distribuzioni dei TTL, e quindi delle distanze tra sorgenti e destinazioni, sono ovviamente molto simili e ripetute in corrispondenza dei valori iniziali. Unica anomalia si può osservare in corrispondenza di un TTL pari a 56: si nota un picco che può essere attribuito al feed notturno delle news in provenienza dal server *newsfeed.cineca.it*. Il valore della percentuale di TTL coincide infatti con quello dei pacchetti in provenienza da tale macchina e guardando la distribuzione di pacchetti per host di destinazione (vedi Sezione 3.2.1) si constata che il server *news.polito.it* durante la settimana presa in considerazione ha ricevuto un numero di pacchetti confrontabile con quello mostrato dal picco.

Come si può osservare in Figura 3.8 il pattern della distribuzione nel periodo C non varia sensibilmente. Osservando la distribuzione dei TTL uscenti è possibile notare una diminuzione dell'utilizzo di macchine Windows 95 identificabili con l'abbassamento dal 3% al 2% del valore di TTL relativo. La distribuzione dei TTL entranti rispecchia quella osservata nel periodo A.

3.2.5 Lunghezza dei pacchetti

Le statistiche relative alla lunghezza dei pacchetti IP, siano essi in ingresso o in uscita, mostrano un andamento influenzato dalla percentuale di pacchetti

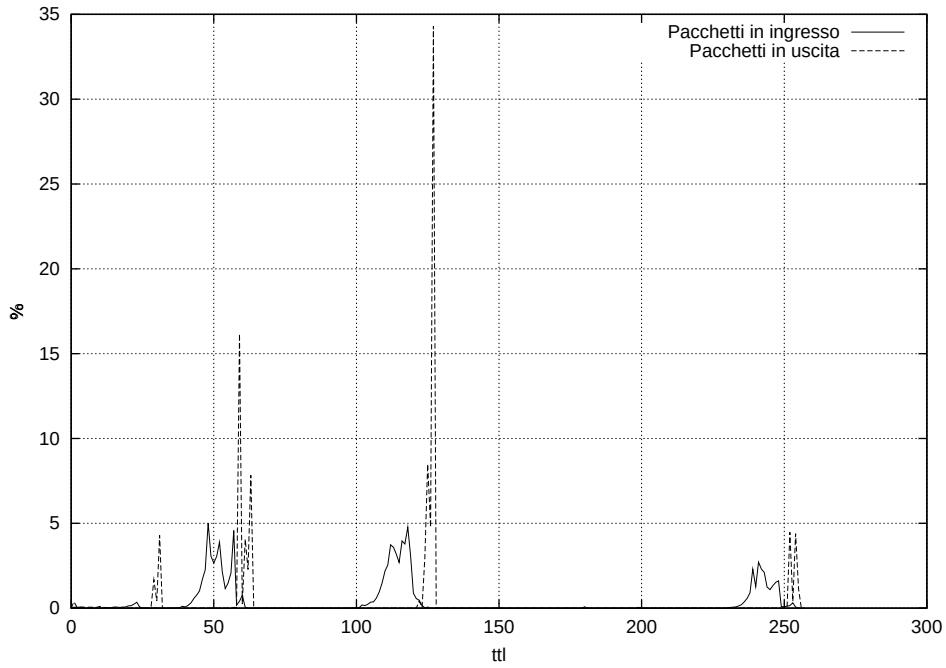


Figura 3.8. Distribuzione percentuale del TTL nel periodo C

TCP che, come si è visto, supera di gran lunga quella degli altri tipi di pacchetti IP. In Figura 3.9 è rappresentata la quantità totale di pacchetti in uscita suddivisa per lunghezza.

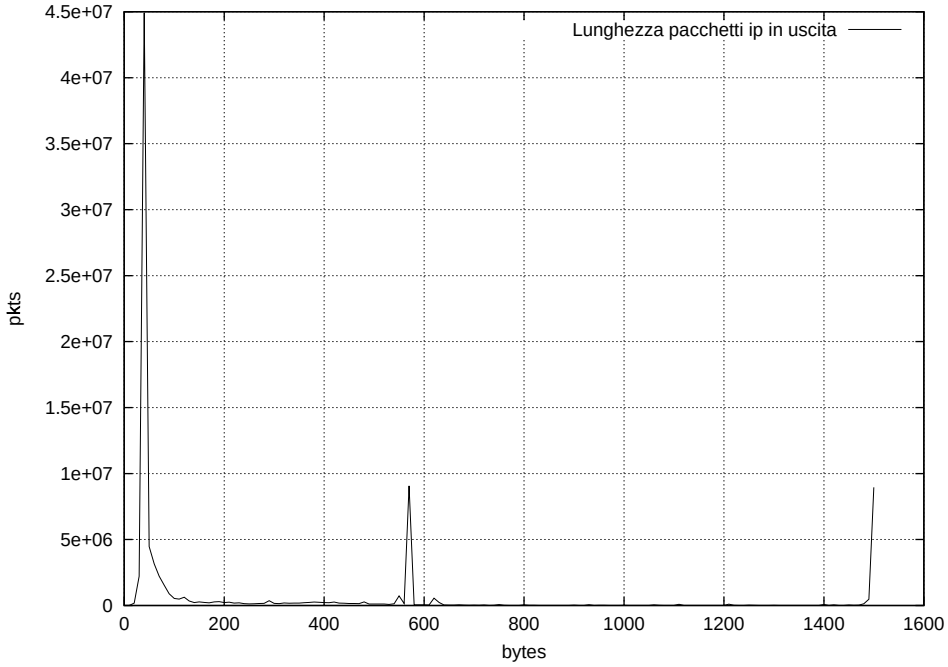


Figura 3.9. Pacchetti IP in uscita suddivisi per lunghezza

È possibile osservare come la maggioranza dei pacchetti abbia dimensione pari a 40 byte o 1500 byte: si tratta, quindi, di pacchetti senza payload (puro ack) e di pacchetti con dimensione massima tipicamente associati a trasferimenti importanti di dati. Un picco è, inoltre, osservabile in corrispondenza di un valore di lunghezza pari a 576 bytes: si tratta della dimensione massima che ogni stack TCP deve essere in grado di gestire per un singolo datagramma IP in base a RFC 791[22]. Il traffico in ingresso, ad esempio, è caratterizzato da trasferimenti di grandi dimensioni molto più di quanto non lo sia il traffico in uscita: questa caratteristica si può osservare in Figura 3.10.

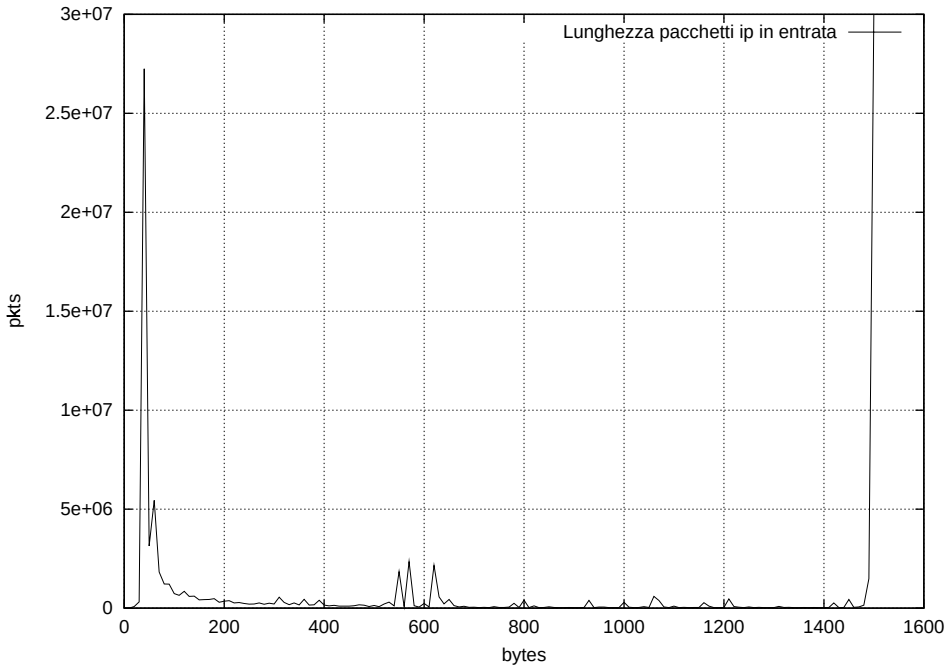


Figura 3.10. Pacchetti IP in ingresso suddivisi per lunghezza

Capitolo 4

Statistiche TCP

In questo capitolo verrà descritto brevemente il protocollo TCP con particolare attenzione alla struttura del singolo segmento. Verranno in seguito esaminate le statistiche a livello TCP prodotte dal programma `tcpstat` che possono essere divise in

- statistiche di protocollo, cioè relative all'utilizzo del protocollo stesso da parte delle varie implementazioni degli stack TCP.
- statistiche sulle connessioni TCP vale a dire statistiche relative a sessioni complete.

4.1 Il protocollo TCP

TCP [8] (Transmission Control Protocol) è un protocollo che permette una comunicazione *affidabile* tra due processi in ambiente interconnesso. Il protocollo si appoggia su IP per inviare i dati da una sorgente ad una destinazione. Per permettere il multiplexing del canale fisico TCP prevede una serie di indirizzi (*porte*) su ogni host.

Il protocollo si basa sul concetto di *connessione*, cioè sulla necessità degli stack TCP dei due host coinvolti, di inizializzare una serie di parametri e mantenerne aggiornato lo stato per la durata dello scambio di dati.

TCP fa parte della categoria dei protocolli a finestra: per garantire l'affidabilità il protocollo divide i dati in blocchi, di dimensione negoziata, tenendo

poi traccia del momento dell'invio. TCP si aspetta un *acknowledgment* di ogni blocco inviato entro un tempo limite trascorso il quale il blocco viene ritrasmesso.

Poiché i segmenti TCP vengono affidati ad IP, è possibile che arrivino con un ordine diverso da quello col quale sono stati trasmessi: per ricostruire il flusso di dati originale il protocollo utilizza dei *numeri di sequenza*. È previsto inoltre un meccanismo di *checksum* per garantire l'integrità dei segmenti.

TCP offre dunque un servizio orientato alla connessione, permette il controllo di errore, il controllo di sequenza, il controllo di flusso e il controllo di congestione.

4.1.1 TCP header

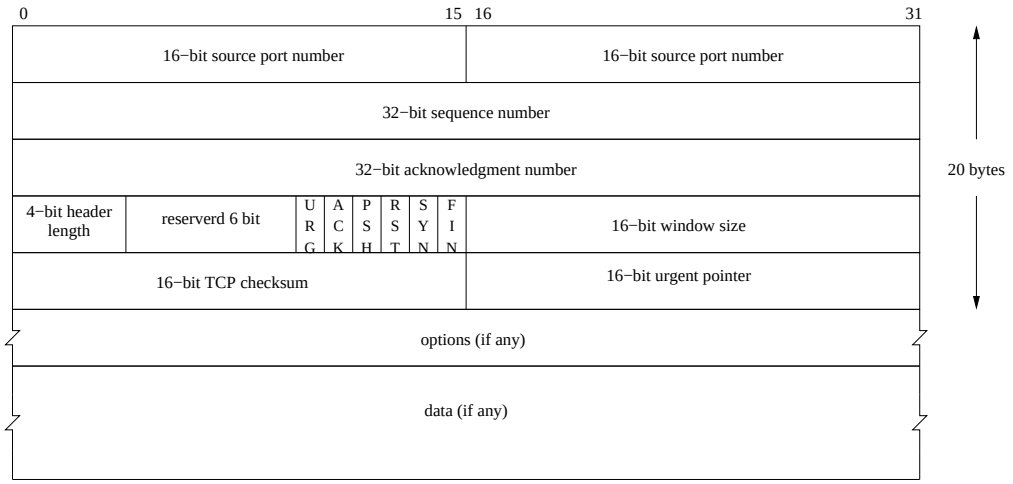


Figura 4.1. Datagramma TCP.

In Figura 4.1 viene mostrato il formato di un segmento TCP. Di seguito verranno descritti in dettaglio i vari campi.

In ogni segmento TCP sono indicati, su 16 bit, la *porta* sorgente e quella di destinazione. Questi due valori, assieme agli indirizzi IP identificano univocamente una *connessione*: due host possono avere più connessioni TCP istaurate e queste saranno distinguibili l'una dall'altra perché almeno uno di questi quattro campi sarà differente.

In ogni segmento TCP, il *sequence number* rappresenta la posizione del primo byte di dati del segmento, all'interno del flusso completo. Serve per ricostruire l'integrità del flusso effettuando il controllo di sequenza e identificando i segmenti mancanti.

Nel momento in cui viene stabilita una nuova connessione, il primo segmento deve avere il flag SYN settato. In un segmento di questo tipo il numero di sequenza contiene l'*initial sequence number* (ISN) scelto dalla sorgente per la connessione: ad ogni byte di dati inviato successivamente dovrà corrispondere un incremento del numero di sequenza. Si tratta di un intero su 32 bit che ritorna a a 0 una volta raggiunto $2^{32} - 1$.

L'*acknowledgment number* contiene il valore del prossimo *sequence number* che la sorgente del segmento si aspetta di ricevere; si tratta quindi dell'ultimo numero di sequenza ricevuto correttamente più 1. Questo campo è valido solo se il flag ACK vale 1.

Il campo *header length* contiene la lunghezza dell'header del segmento TCP espressa in word da 32 bit. Il campo è necessario poiché la lunghezza delle opzioni è variabile. I 4 bit dell'*header length* impongono un limite massimo di 60 byte all'intestazione. Un segmento TCP senza opzioni ha un header lungo 20 byte.

I 6 flag dell'header TCP possono valere 1 o 0 indipendentemente l'uno dall'altro. Essi sono:

URG: indica che il valore dell'*urgent pointer* è valido.

ACK: indica che il valore dell'*acknowledgment number* è valido.

PSH: indica che lo stack TCP dell'host che riceve il segmento deve passare i dati all'applicazione appena possibile.

RST: reset della connessione.

SYN: il segmento con il flag SYN con valore 1 è il primo segmento della connessione e va utilizzato per sincronizzare il *sequence number*.

FIN: un valore a 1 indica che la sorgente ha finito di inviare dati.

Il controllo di flusso del protocollo TCP avviene grazie all'utilizzo della *window size* proposta dal ricevitore. Si tratta di un indicazione del numero di byte che il ricevitore è in grado di accettare, a partire dal byte indicato nell' *acknowledgment number*. Tramite l'opzione *window scale*, è possibile variare questo valore.

Il valore del *checksum* copre l'intero segmento TCP, sia header che dati, e deve essere controllato dalla destinazione.

Si noti come in Figura 4.1 la parte di dati sia opzionale. Segmenti di puro header sono validi e sono utilizzati, per esempio, per iniziare e terminare una connessione o per effettuare l'acknowledge di un segmento se non vi sono dati da trasmettere in direzione opposta.

4.1.2 Opzioni TCP

Le opzioni TCP trovano posto alla fine dell'header, dopo i primi 20 byte fissi descritti in precedenza, e sono comprese nel checksum. Hanno lunghezza variabile da 1 a più byte, e, per permettere all'header di essere allineato a word da 32 bit, esistono speciali opzioni di padding. Si noti che, se è obbligatorio che l'header sia allineato, il protocollo però non impone la stessa cosa per le opzioni: è quindi opportuno che gli stack TCP siano preparati a gestire il caso in cui le queste si susseguano e il padding avvenga solo alla fine.

End of option list (1 byte): questa opzione, da usarsi come ultima della lista, ne indica la fine. Va utilizzata per allineare l'header, secondo quanto detto in precedenza.

No operation (1 byte): si tratta di un opzione di padding da utilizzarsi nel caso in cui altre opzioni seguano all'interno dello stesso segmento.

Maximum segment size (4 byte): l'opzione più comune nell'header TCP è il valore del *maximum segment size* (MSS) che, specificata obbligatoriamente nel primo segmento scambiato, indica la dimensione massima per segmento che il destinatario intende ricevere.

Window scale (3 byte): la finestra massima specificabile nell'header TCP è di $2^{16} = 65$K byte. Questo valore si è dimostrato inadeguato nel caso di comunicazioni su banda particolarmente larga con un round trip time elevato. Per ottenere finestre più larghe è prevista l'opzione *window scale* che indica un fattore di moltiplicazione sino a 2^{14}. Questa opzione va specificata esclusivamente nei segmenti SYN che iniziano la connessione e non può essere cambiata in seguito. Essa può venir proposta dalla sorgente con il primo segmento ed è valida solo se la destinazione risponde includendola a sua volta.

Timestamp (10 byte): questa opzione permette alla sorgente di includere una *timestamp* in ogni segmento, valore che verrà restituito dalla destinazione al momento dell'*acknowledge*. Si tratta di un'opzione simmetrica (entrambe gli host devono ammetterla) che permette ai due stack TCP di conoscere il *round trip time* di ogni ACK ricevuto. Si noti che è scorretto pensare a questa misura come un'indicazione del RTT dei segmenti in quanto spesso un ACK corrisponde a più di un segmento: nel caso di ACK cumulativi, vale a dire ack che coprono più segmenti, il valore di *timestamp* di risposta è quello del primo segmenti.

Selective Acknowledgment: il valore del campo *acknowledgment* nell'header TCP indica l'ultimo byte ricevuto correttamente più 1. Si tratta quindi di *acknowledgment* cumulativo che non tiene conto di eventuali segmenti persi all'interno della sequenza. Le opzioni di *selective acknowledgment* permettono di ovviare a questo problema richiedendo la ritrasmissione di blocchi specifici:

sack permitted (2 byte): questa opzione può essere inviata solo in segmenti con il flag SYN, cioè all'inizio della comunicazione e serve a richiedere e confermare la possibilità dell'utilizzo dell'opzione *sack* durante la connessione

sack option (lunghezza variabile): questa opzione viene utilizzata durante la connessione per informare la controparte di blocchi discontigui di dati ricevuti e accodati; quest'ultima dovrà provvedere

a rinviare i dati mancanti alla cui ricezione un adeguato *acnowledgement number*, indicherà lo spostamento della finestra dell'host ricevente. Si noti quindi come il *selective acknowledgment* non cambi il significato dell'*acknowledgment number* che continua a indicare il limite inferiore della finestra di ricezione.

4.2 Analisi a livello TCP

Di seguito verranno descritti i risultati ottenuti dall'esame delle statistiche TCP di due periodi della durata di una settimana ciascuno: la prima a maggio 2000, corrispondente al periodo A descritto nel capitolo precedente, e la seconda tra gennaio e febbraio 2001 (periodo C). Verrà considerato *client* (e indicato come *a*) l'host che inizia la connessione e come *server* (indicato come *b)* quello che la accetta.

In un primo momento verranno discusse statistiche di protocollo, in particolare verranno esaminate le percentuali di utilizzo delle opzioni più frequenti: *MSS*, *sack*, *timestamp* e *window scale*. In seguito si passerà allo studio di flussi completi di dati, vale a dire connessioni TCP per le quali abbiamo visto almeno un segmento di tipo SYN e concluse in uno dei seguenti modi:

- con uno scambio reciproco di segmenti di tipo FIN e relativo acknowledgment.
- con un segmento RST inviato da uno dei due host.
- per timeout dopo 30 minuti di inattività. Vengono infatti considerate concluse quelle connessioni per le quali negli ultimi 30 minuti non si sono visti transitare segmenti.

Per questo tipo di connessioni verranno esaminate informazioni circa il servizio trasportato, la lunghezza, la durata, le perdite e le finestre di congestione e di ricezione.

Richiesta del client		Richiesta del server	
MSS	%	MSS	%
1460	90.9%	1460	80.7%
530-540	6.0%	0-10	7.6%
1450-1460	1.8%	510-520	3.6%
0-10	0.4%	1380-1390	3.0%
510-520	0.2%	530-540	2.4%

Table 4.1. MSS periodo A

Richiesta del client		Richiesta del server	
MSS	%	MSS	%
1460-1470	88.6%	1460-1470	73.9%
530.0-540.0	6.9%	0-10	13.4%
510.0-520	2.2%	510-520	5.2%
1450-1460	0.8%	530-540	3.1%
1380-1390	0.3%	1380-1390	2.8%

Table 4.2. MSS periodo C

4.2.1 Opzioni TCP

4.2.1.1 MSS

Un esame degli MSS richiesti da client e server nei due periodi mostra come la situazione, a distanza di 9 mesi, non sia di fatto cambiata. Il valore di MSS più richiesto in entrambi i casi è 1460 che corrisponde all'MTU standard per Ethernet (1500) meno i 40 byte degli header minimi IP e TCP. Più raramente viene richiesto un valore compreso tra 1450 e 1460 da host che, presumibilmente, prevedono di utilizzare opzioni TCP.

Un valore compreso tra 530 e 540 è il secondo come percentuale di richieste per i client. Si tratta, con ogni probabilità, di 536 che corrisponde alla dimensione di un datagramma IP da 576 byte (meno i 40 byte dei due header minimi IP e TCP) vale a dire la dimensione minima di un segmento IP che deve essere supportata da ogni router [23]. Anche in questo caso si può notare una minima percentuale di valori leggermente inferiori dovuti a possibili opzioni TCP che limitano la dimensione dell'MSS.

È interessante notare come sia prassi comune per i server scegliere come

MSS un valore di 536 indicando 0: tale valore, infatti, è da considerarsi come un adeguamento al valore di default, 536.

4.2.1.2 Selective Acknowledgment

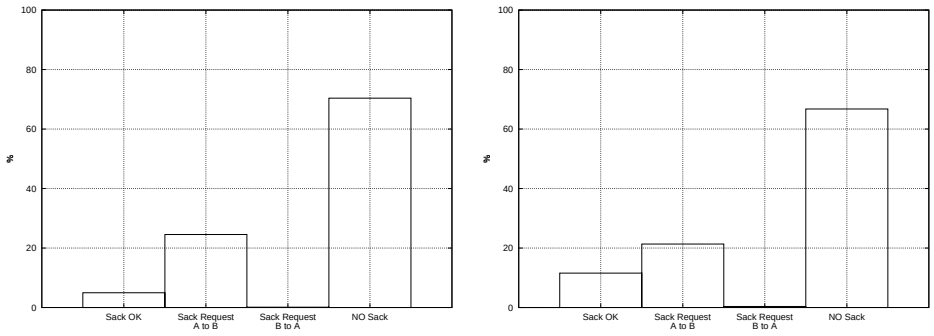
In Figura 4.2 sono riportate le percentuali di connessioni che fanno richiesta e/o uso dell'opzione di *Selective Acknowledgment*. In particolare l'istogramma riporta con l'etichetta *NO Sack* la percentuale di connessioni che non fanno uso di *Sack*. *Sack Request A to B* indica la percentuale di client che all'inizio della connessione si dichiarano disponibili a supportare *sack*. *Sack Request B to A* indica la percentuale di server che richiedono l'utilizzo dell'opzione *Sack* pur non avendone ricevuto richiesta. In entrambi i casi questa percentuale è nulla infatti, come previsto dal protocollo, se il client non richiede *Sack*, il server non può proporlo. Infine *Sack OK* indica la percentuale in cui la negoziazione dell'opzione *Sack* ha avuto successo.

Figura 4.2 mostra un notevole incremento del numero di connessioni che fanno uso dell'opzione passando dal periodo A al periodo C. Si passa infatti da una percentuale minima nel 2000, appena il 4%, ad un valore del 15% l'anno successivo. Si noti come rimane pressoché inalterata la quantità di connessioni che non usano l'opzione e come l'incremento sia dovuto al fatto che un numero maggiore di server rispondono positivamente alla richiesta dell'opzione. Il numero di client che dichiarano la disponibilità ad usare *Sack*, invece, è rimasta pressoché costante nei due periodi.

4.2.1.3 Window Scale

Similmente a quanto riportato in Figura 4.2, Figura 4.3 riporta la percentuale di richieste di utilizzo dell'opzione *Window Scale* nei periodi A e C. In particolare vengono riportate le percentuali di connessioni che non usano l'opzione, etichettate *No Wscale*, di richieste lato client, con etichetta *Wscale Request A to B*, dal lato server, etichetta *Wscale Request B to A*, e infine le connessioni dove entrambe i flussi hanno concordato l'utilizzo dell'opzione.

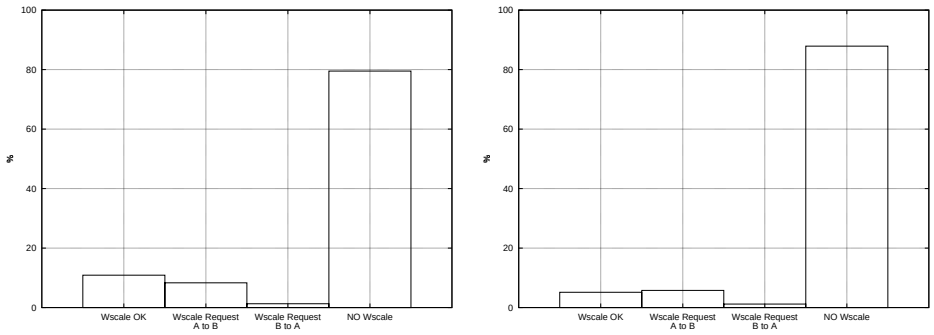
Il confronto tra i due periodi per quel che riguarda l'opzione *Window Scale,* al contrario, mostra una diminuzione nell'utilizzo. In questo caso la proporzione fra il numero di richieste e di risposte positive rimane costante,



(a) Periodo A (b) Periodo C

Figura 4.2. Percentuale di utilizzo dell'opzione di *Selective Acknowledgment.*

mentre è possibile notare come sia cresciuto il numero di connessioni che dell'opzione non fanno uso.



(a) Periodo A (b) Periodo C

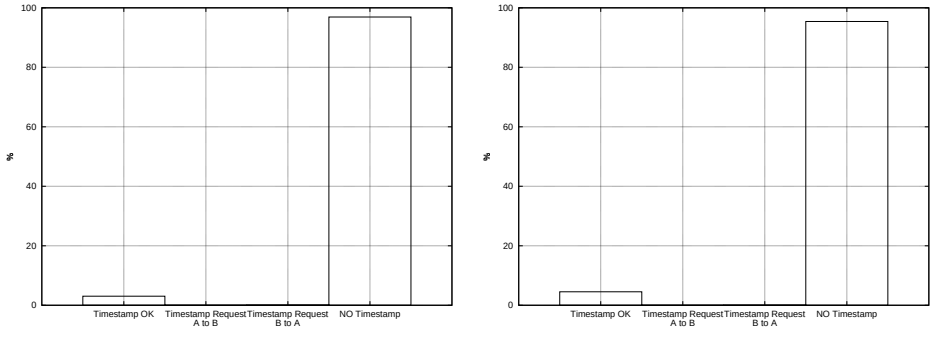
Figura 4.3. Percentuale di utilizzo dell'opzione *Window Scale.*

4.2.1.4 Timestamp

In Figura 4.4 sono riportate le percentuali di utilizzo in relazione all'opzione *Timestamp.* L'etichetta *No Timestamp* indica la percentuale di connessioni per le quali non è attiva l'opzione. Le etichette *Timestamp request A to B* e *Timestamp Request B to A* indicano rispettivamente la percentuale di

richieste dei client e dei server, mentre *Timestamp Ok* indica la percentuale di connessioni per le quali l'opzione è stata negoziata con successo.

Gli istogrammi relativi all'opzione indicano come, di fatto, l'opzione venga usata molto raramente nonostante sia stata standardizzata già nel 1992 [8].



(a) Periodo A

(b) Periodo C

Figura 4.4. Percentuale di utilizzo dell'opzione *Timestamp*.

4.2.2 Lunghezza delle connessioni

Come si vedrà in seguito in Sezione 4.3, una notevole componente del traffico TCP è dovuto alla navigazione sulla rete. Il traffico web è caratterizzato dalla presenza di molti elementi di piccole dimensioni trasmessi sia dai client (richieste http) sia dai server (pagine html composte da testi, immagini e in generale oggetti distinti) e le statistiche sulla lunghezza delle connessioni confermano questo comportamento. Si prenda ad esempio Figura 4.5 relativa al periodo A, dove il grafico tra 0 e 50KB a passo di 50 byte mostra una percentuale preponderante di semi-connessioni[1] di dimensione inferiore ai 1500 KB.

Si noti, nell'ingrandimento, la presenza consistente di semi-connessioni senza trasferimento di dati (lunghezza 0) la cui percentuale è costante nelle due direzioni: si tratta, con ogni probabilità, di tentativi di connessione

[1]Per semi-connessione si intente una delle due direzioni della connessione full-duplex TCP.

abortiti dopo i primi pacchetti. Si prenda ad esempio un tentativo di connessione ad una servizio non attivo sul server: quest'ultimo, al segmento SYN risponderà con un RST e in entrambe i casi non si avrà scambio di dati.

Una minima parte di questi segmenti senza dati, può inoltre essere spiegato con la presenza di connessioni fortemente sbilanciate quali, ad esempio, il trasferimento di dati tramite FTP: in questo tipo di servizio la semiconnessione dal client verso il server veicola soltanto segmenti di *acknowledgment* e nessun dato in quanto la connessione di controllo è separata da quella di scambio file.

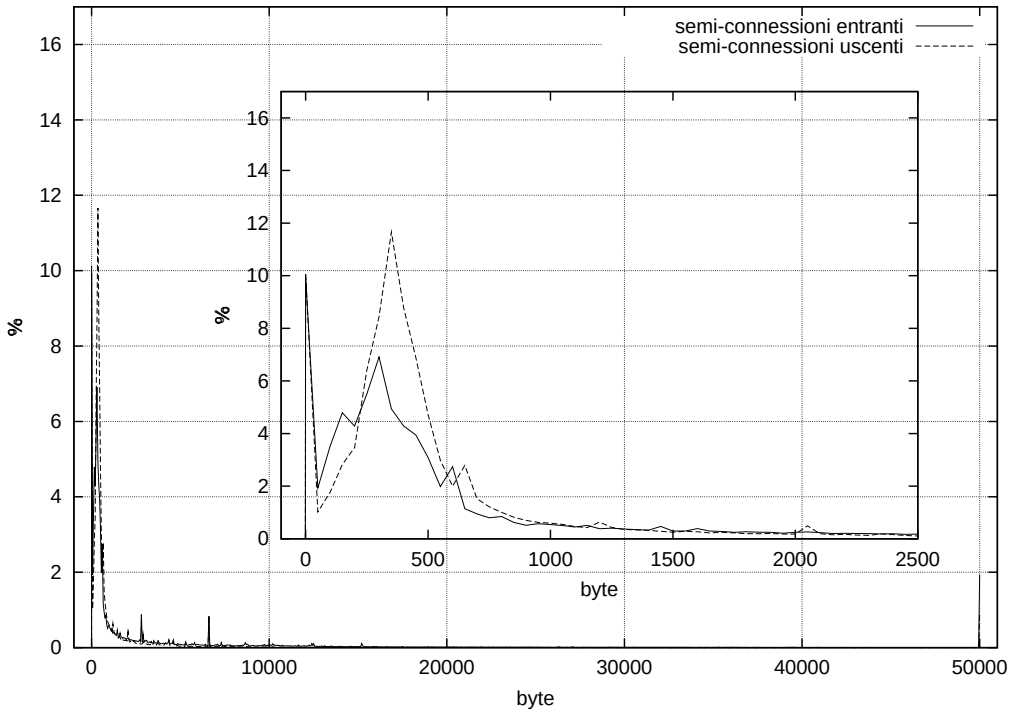


Figura 4.5. Lunghezza delle connessioni espressa in byte (Periodo A).

In figura 4.6 lo stesso grafico, riproposto per il periodo C, mette in evidenza una andamento simile a quello del periodo precedente in cui la percentuale maggiore delle connessioni ha lunghezza, in byte, inferiore ai 1500 KB. Si noti, nell'ingrandimento, come la percentuale di connessioni in precedenza identificate come non riuscite (quelle, cioè, con lunghezza 0) sia passata dal 10% al 16%.

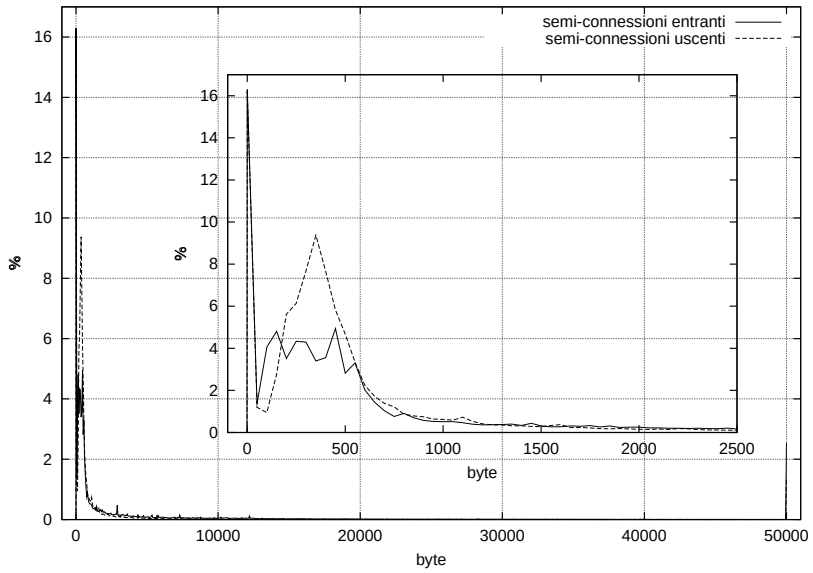


Figura 4.6. Lunghezza delle connessioni espressa in byte (Periodo C).

La figura 4.7 mostra la lunghezza delle connessioni nell'intervallo da 50kB a 50MB a passi di 50kB relativamente al periodo A. La scala logaritmica utilizzata mette in evidenza come al distribuzione di queste percentuali sia di tipo *heavy tailed*, ossia distribuzione la cui coda finale decresce in maniera polinomiale la cui rappresentazione in scala doppiamente logaritmica è chiaramente rappresentata da un andamento lineare. Si noti come, in generale, le semi-connessioni entranti abbiano una lunghezza maggiore di quelle uscenti. La durata della misura permette di stimare abbastanza bene la distribuzione delle connessioni fino a circa il valore di 10^{-4}, mentre oltre non sono stati registrati dati a sufficienza per poter avere una misura affidabile della distribuzione.

In Figura 4.8 è possibile confrontare le percentuali relative alle lunghezze, espresse in segmenti sia contenenti dati che di pura segnalazione per TCP, delle connessioni nei periodi A e C. A distanza di nove mesi la situazione non cambia significativamente ed è evidente la predominanza di connessioni con meno di 10 segmenti. Si sottolinea come oltre il 70% delle connessioni non scambi più di 10 segmenti dati in entrambe i periodi.

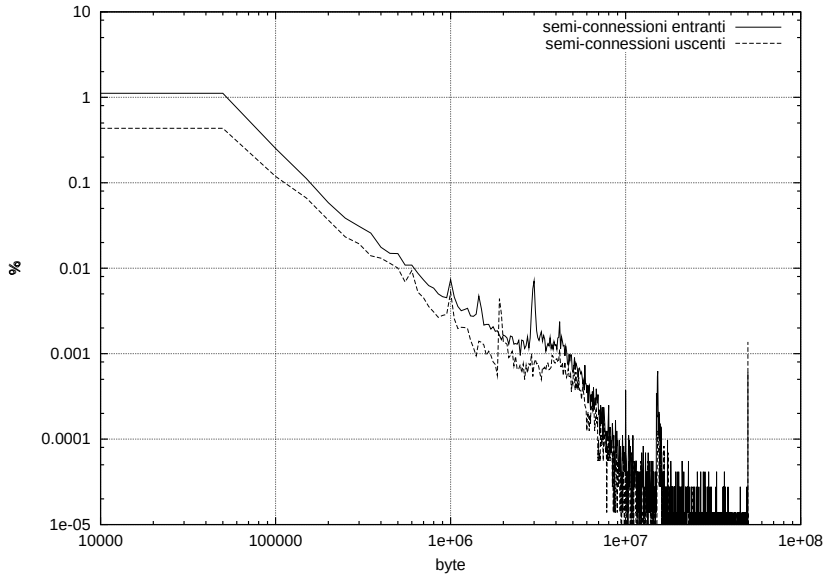
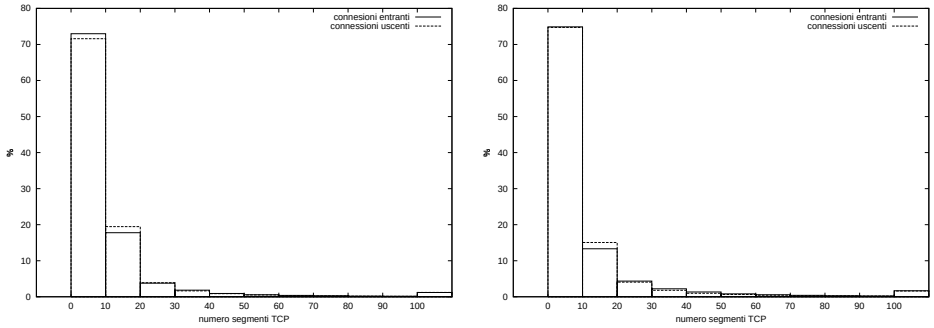


Figura 4.7. Lunghezza delle connessioni espressa in byte (Periodo A).



(a) Periodo A

(b) Periodo C

Figura 4.8. Lunghezza delle connessioni espressa in segmenti TCP.

4.3 Asimmetria delle connessioni

Una postelaborazione dei dati forniti da `tcpstat` ha permesso di ricavare interessanti statistiche in relazione all'asimmetria delle connessioni sia per quel che riguarda il numero di segmenti scambiati che di byte trasmessi. Per asimmetria delle connessioni si intende il rapporto tra i dati trasmessi dal client al server e viceversa.. I dati presentati di seguito fanno riferimento

a misurazioni relative ad un unico periodo, in particolare il periodo C, in quanto si è potuto constatare come la situazione non sia differente rispetto agli altri periodi di misura.

In un primo momento si è cercato di caratterizzare l'utilizzo del canale di trasmissione in relazione alla quantità di dati inviati dal client rispetto ai byte totali scambiati, secondo la formula:

$$x = \frac{dati(a2b)}{dati(b2a) + dati(a2b)}$$

Il valore di x è compreso tra un minimo di 0 (solo dati dal server) e un valore massimo di 1 (solo dati dal client), il valore 0.5 indica perfetta simmetria.

Il grafico ottenuto, si veda Figura: 4.9, mostra come le connessioni sono in generale sbilanciate dal lato server e un calcolo preciso rivela un rapporto 68% a 32% rispetto al valore 0.5 indicativo dell'equilibrio.

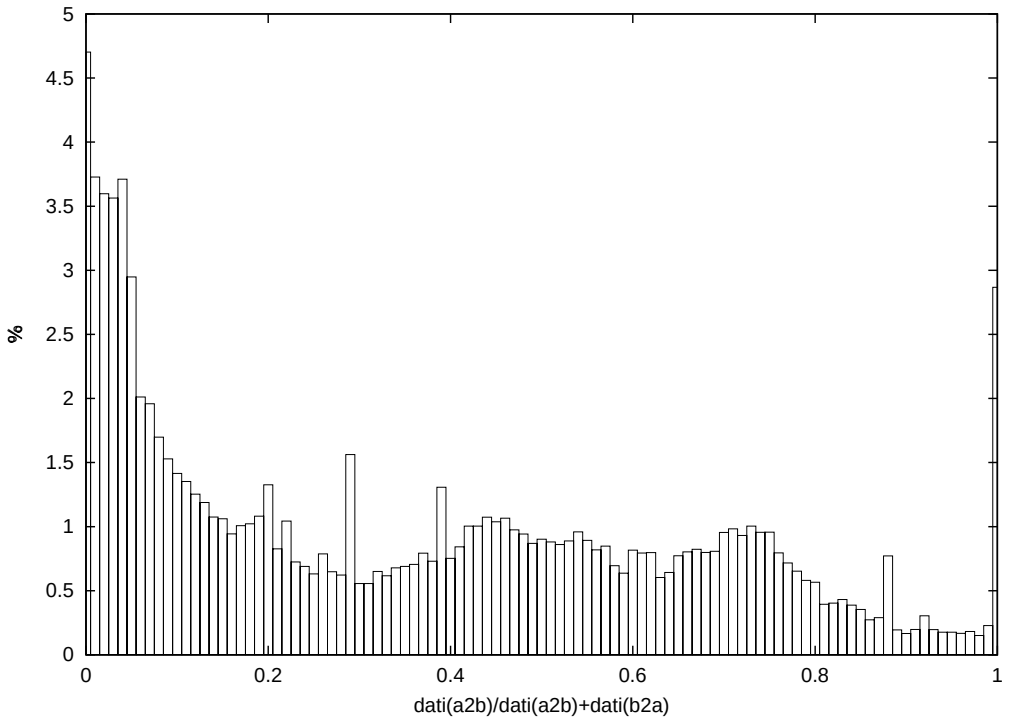


Figura 4.9. Asimmetria nella trasmissione di dati tra client e server.

Interessante è anche lo studio dell'asimmetria in termini di segmenti scambiati nel corso della comunicazione tra client e server e viceversa, compresi i segmenti dal protocollo TCP. Per il grafico in Figura 4.10, si è calcolato questo dato tramite la formula:

$$x = \frac{segmenti(a2b)}{segmenti(a2b) + segmenti(b2a)}$$

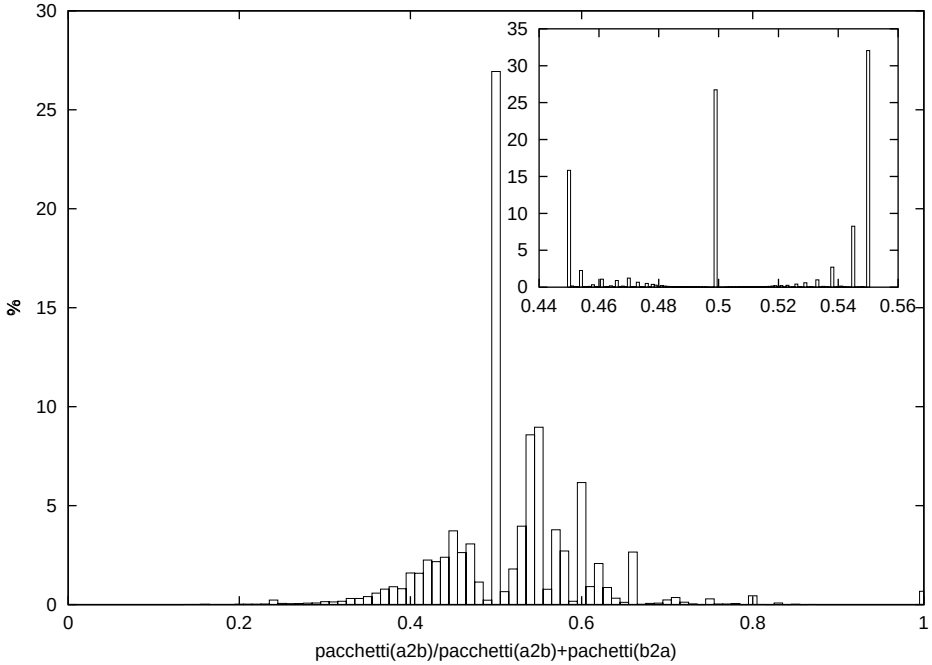


Figura 4.10. Asimmetria nella trasmissione di segmenti tra client e server.

Appare evidente come la maggior parte delle connessione siano bilanciate su un ugual numero di pacchetti nelle due direzioni. Si noti nell'ingrandimento come l'alta percentuale di flussi per i quali il rapporto suddetto vale 0.5 e che appare come il più frequente, è effettivamente indicativo di una perfetta simmetria.

Nonostante l'asimmetria nello scambio dati evidenziata in figura 4.9, l'overhead introdotto dal protocollo TCP rende invece in numero di segmenti scambiato in una connessione molto più simmetrico, come evidenziato in Figura 4.10.

In maggior dettaglio, come messo in evidenza dall'istogramma ingrandito, i picchi in corrispondenza di valori pari a 0.54 e 0.55 sono indicativi di come l'asimmetria del protocollo TCP pesi nel caso di connessioni con scambio di pochi segmenti vale a dire il 70% delle connessioni misurate come visto nel paragrafo 4.2.2. Si prenda ad esempio l'evoluzione della seguente richiesta HTTP catturata con `tcpdump`,

```
10.1.2.46.1122 > 194.244.52.66.80: S 1727657860:1727657860(0) win 32120 size 74
194.244.52.66.80 > 10.1.2.46.1122: S 1742786283:1742786283(0) win 32120 size 78
10.1.2.46.1122 > 194.244.52.66.80: . ack 1 win 32120 size 66
10.1.2.46.1122 > 194.244.52.66.80: P 1:70(69) ack 1 win 32120 size 135
194.244.52.66.80 > 10.1.2.46.1122: . ack 70 win 32120 size 70
194.244.52.66.80 > 10.1.2.46.1122: P 1:939(938) ack 70 win 32120 size 1008
10.1.2.46.1122 > 194.244.52.66.80: . ack 939 win 32120 size 66
194.244.52.66.80 > 10.1.2.46.1122: F 939:939(0) ack 70 win 32120 size 70
10.1.2.46.1122 > 194.244.52.66.80: . ack 940 win 32120 size 66
10.1.2.46.1122 > 194.244.52.66.80: F 70:70(0) ack 940 win 32120 size 66
194.244.52.66.80 > 10.1.2.46.1122: . ack 71 win 32120 size 70
```

Per comodità in Figura 4.11 abbiamo riportato la sequenza degli eventi della connessione.

Come si può notare a fronte dei 6 pacchetti inviati dal client sono solo 5 i segmenti inviati dal server e il rapporto precedentemente indicato vale, appunto, 0.54.

Per meglio comprendere il modo in cui il traffico WWW predomina tra i vari tipi di connessioni sulla rete, sono state state calcolate le percentuali relative di utilizzo dei vari servizi. Parallelamente sono state calcolate le percentuali di pacchetti appartenenti a queste connessioni e il traffico in byte generato. È possibile osservare il risultato in Tabella 4.3 dove risulta evidente il predominio della navigazione WWW in termini di numero di connessioni. Passando invece alla percentuale di segmenti o di byte trasmessi, il peso di altri servizi assume una certa importanza, è il caso, ad esempio, della connessione dati del protocollo FTP: quasi il 10% dei byte trasmessi è relativo al trasferimento di dati con questo protocollo trasportato da un esiguo 0.5% di connessioni.

Un altro dato interessante per comprendere il diverso utilizzo del protocollo da parte dei vari applicativi è illustrato in Tabella 4.4 dove sono riportate

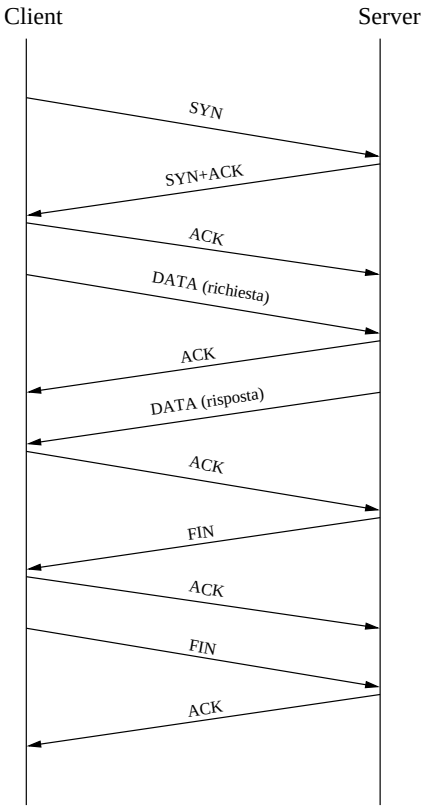


Figura 4.11. Scambio di segmenti in una connessione di richiesta HTTP tipica.

Servizio	Porta	% connessioni	% segmenti	% bytes
http	80	81.48	62.78	61.27
mail	25	2.98	2.51	2.04
https	443	1.66	0.87	0.52
pop	110	1.26	0.93	0.42
auth	113	0.54	0.07	0.00
ftp controllo	21	0.54	0.54	0.50
-	6346	0.53	2.44	1.58
ftp dati	20	0.51	6.04	9.46
-	992	0.45	0.13	0.00
dns	53	0.31	0.03	0.01

Tabella 4.3. Percentuali di connessioni, segmenti e byte per servizio, periodo C.

le dimensioni medie in segmenti ed in byte delle semiconnessioni client-server e server-client in relazione ai diversi servizi.

Come si può osservare in alcuni casi, ad esempio per la posta in uscita (mail), la connessione è sbilanciata verso il client che invia un numero di byte maggiore di quelli che riceve. Al contrario, per la maggior parte dei servizi è il server a occupare maggiormente la banda, si veda ad esempio il caso delle connessioni http e pop e delle richieste dns.

È possibile notare un comportamento particolare per quel che riguarda la porta dati del servizio FTP: in prima analisi sembra si tratti di un caso in cui il client invia più byte di quanti non ne riceva. La spiegazione risiede nel fatto che il trasferimento di dati in una connessione FTP, sia esso per lo scambio di un file che per il listing di una directory, avviene su una connessione TCP iniziata dal server su una porta scelta dal client. Per questo motivo nelle nostre misure, dove chi attiva la connessione viene denominato client, i dati sembrano invertiti. In questo tipo di connessioni, inoltre è possibile osservare un rapporto di 3 : 2 tra i pacchetti inviati da un estremo della connessione e l'altro. Si tratta, infatti, di flussi molto lunghi sui quali si può notare l'effetto del *delayed acknowledgment* che invece negli altri casi non risulta così evidente.

Servizio	Porta	Media			
		client to server		server to client	
		pacchetti	bytes	pacchetti	bytes
http	80	15.22	1189.95	19.55	15998.45
mail	25	21.19	15034.38	16.76	624.26
https	443	11.47	936.71	12.33	6255.79
pop	110	14.87	91.05	18.47	7489.01
auth	113	3.05	4.29	2.78	15.42
ftp controllo	21	23.72	11931.13	21.86	9254.27
-	6346	101.50	23806.91	105.71	44393.94
ftp dati	20	313.99	343921.34	223.45	82873.22
-	992	7.35	58.86	5.89	136.56
dns	53	2.14	34.19	2.29	571.54

Tabella 4.4. Media di byte e pacchetti per connessione, periodo C.

4.4 Durata delle connessioni

In Figura 4.12 è riportata la durata delle connessioni, in millisecondi, espressa come tempo intercorso tra il passaggio del primo pacchetto contenente il flag SYN e il segmento che conclude la connessione, secondo le modalità riportate in Sezione 4.2. È possibile constatare come la maggior parte delle connessioni si concluda entro i primi 700 ms o allo scadere del primo timeout di TCP, vale a dire dopo 3 secondi.

La scala logaritmica mette in risalto singolarità interessanti come ad esempio il picco in corrispondenza di un valore di durata pari a 30 secondi: un'analisi del fenomeno ha permesso di ricondurlo ad una proprietà del protocollo di identificazione proposto in RFC 1413 [9]. Il protocollo prevede un meccanismo per permettere ad un client di conoscere l'identificativo dell'utente proprietario di una data connessione sul server. Lo scambio di informazioni avviene dopo il collegamento del client alla porta 113 del server. Allo scadere di un timeout di 30, nel caso in cui il questi non risponda, la connessione viene chiusa dal client con un pacchetto di RST.

Sono presenti altre singolarità e picchi più o meno distribuiti, anche essi riconducibili a timer a livello applicativo o ai fenomeni legati ai meccanismi di backoff interni a TCP stesso. Se si analizza l'intervallo di durata oltre i 70 secondi la distribuzione ottenuta presenta altri fenomeni di concentrazione simili.

4.5 Meccanismi di controllo di flusso

Una delle caratteristiche fondamentali del protocollo TCP è la possibilità da parte del client e del server di controllare la quantità del flusso di dati che il ricevitore è in grado di smaltire, e di adeguare di conseguenza la velocità di trasmissione del server.Questo avviene grazie al fatto che il client comunica al server la dimensione della finestra ancora libera in ricezione. Altrettanto importante nel protocollo TCP è il controllo di congestione, effettuato regolando invece la finestra di trasmissione del server. Ad esempio nel caso in cui due host siano collegati da reti ad alta velocità e basso ritardo, è possibile che la sorgente dei pacchetti invii più di un segmento sino ad occupare la finestra

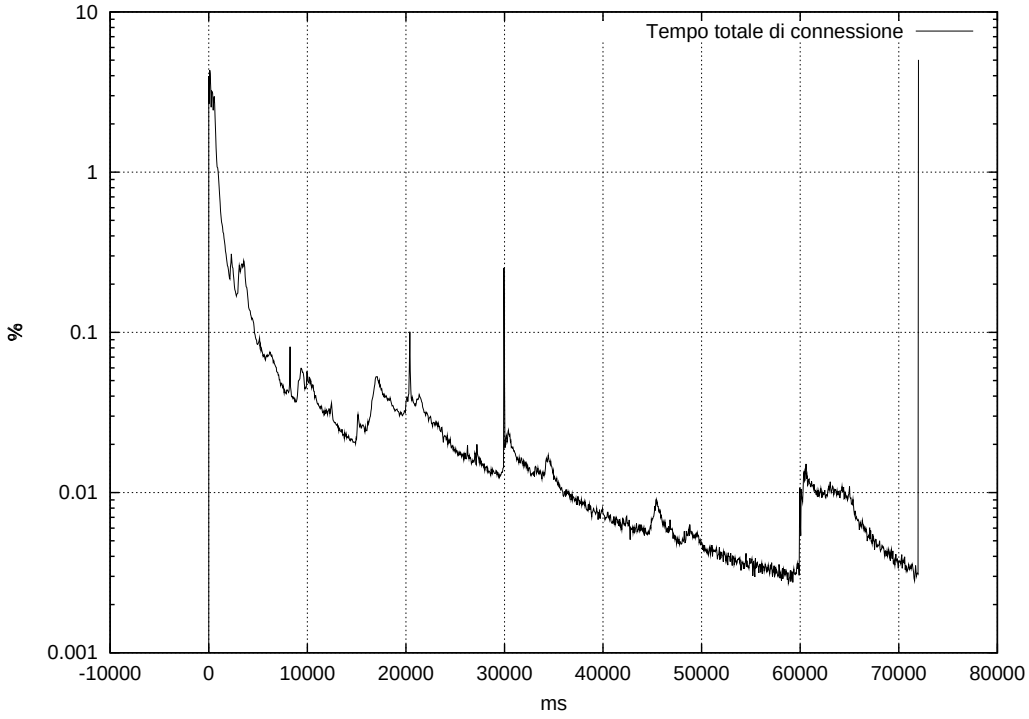


Figura 4.12. Tempo totale di connessione.

massima annunciata dal ricevitore. Questo comportamento, se non limitato, può causare problemi nel momento in cui i segmenti devono attraversare uno o più router che, a causa di congestioni, potrebbero dover scartare alcuni di questi pacchetti.

Per risolvere questo problema sono utilizzati gli algoritmi *slow start* e *congestion avoidance* [12] per il controllo della quantità di dati che la sorgente può inviare in ogni momento: all'inizio di ogni connessione ogni host inizializza la variabile *cwnd* (*congestion window*) nello stack TCP al valore corrispondente all'MSS annunciato dalla controparte. Questa variabile viene incrementata della dimensione di un segmento ad ogni ack ricevuto durante la connessione fino a quando non si causa la perdita di un pacchetto sulla rete. Da quel momento la sorgente ha una stima della finestra di trasmissione e può usare l'algoritmo congestion avoidance per regolare la sua velocità.

Ogni host può, in ogni momento, inviare una quantità di dati massima

pari al minimo tra i due valori della *congestion window* e della finestra annunciata dal ricevitore. La finestra di congestione, quindi, è un controllo di flusso imposto dalla sorgente mentre la quella di ricezione è una limitazione imposta dalla destinazione.

Il valore misurato da `tcpstat` per la *congestion window* non è, chiaramente, il valore effettivo calcolato dallo stack TCP dei due host, ma una sua stima effettuata in base al calcolo della differenza tra i valori degli ack visti transitare in una direzione e dei corrispondenti numeri di sequenza nell'altra: la differenza dunque tra numero di sequenza uscente ed *acknowledge number* entrante corrisponde al numero di byte già trasmessi ed ancora in transito nella rete, che coincide con una stima della finestra di trasmissione grazie al fatto che le misure sono state effettuate in prossimità di uno dei due host.

In Figura 4.13 viene mostrato il valore della *congestion window* calcolata nel periodo A, sempre distinguendo tra semiflussi entranti e uscenti. La percentuale misurata pari al 35% circa, corrispondente al valore 0 è dovuta a semiflussi che non trasmettendo dati ricevono *acknowledgment number* pari esattamente all'ultimo numero di sequenza inviato.

Si noti come i valori della *congestion window* siano relativamente bassi: ciò è dovuto alla dimensione ridotta dei flussi osservata in Sezione 4.2.2 che impedisce la crescita della finestra: il basso numero di segmenti scambiati, infatti, limita lo stack TCP nell'incremento della variabile *cwnd*. Sono particolarmente evidenti gli incrementi del valore della *congestion window* secondo multipli interi dell'MSS. Come precedentemente illustrato, infatti, TCP prevede che questa sia l'unità di misura da utilizzare per allargare o restringere la finestra.

Il grafico nelle Figure 4.14 e 4.15 mostra i valori minimo, massimo e medio della finestra di ricezione annunciata durante le connessioni nei periodi A e C. In questo caso la misura è direttamente ricavata dall'interstazione dei segmenti e non è quindi una stima. Come si può notare il valore più usato nei due periodi rimane 8K, ma è interessante osservare la sua diminuzione a distanza di nove mesi a fronte di un incremento dell'utilizzo di una finestra di 16K. Il valore limitato della finestra può seriamente interferire con i meccanismi di TCP per il riconoscimento della perdite ed in particolare rendere critico l'uso del *Fast Recovery* che rileva la perdita grazie alla ricezione

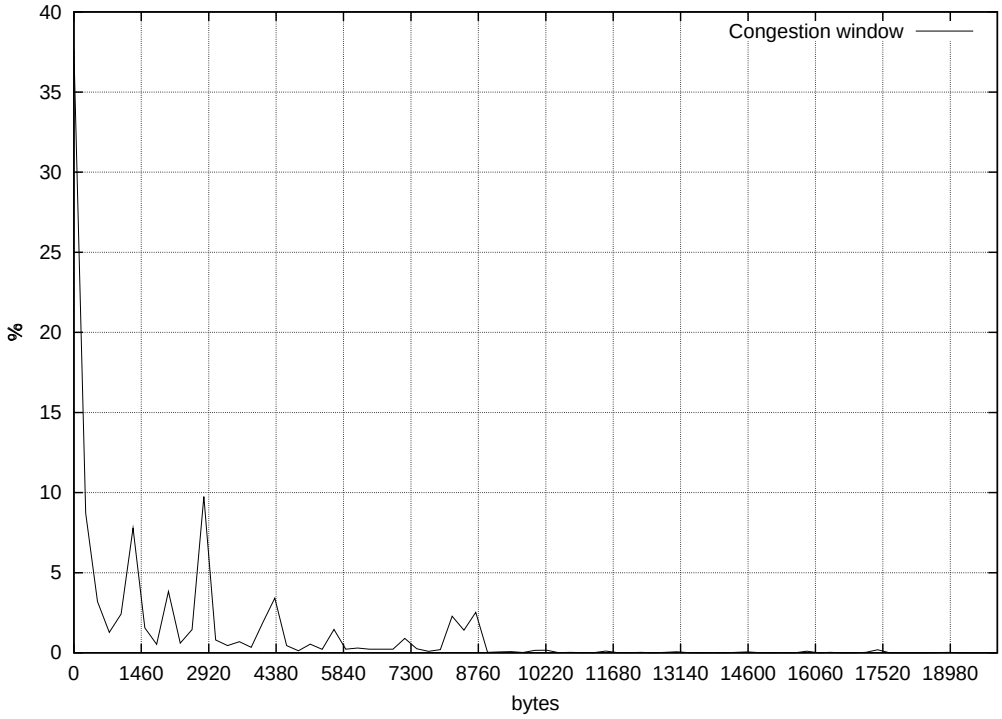


Figura 4.13. Congestion window stimata.

di tre *ack* duplicati. Infatti il valore di 8K unito a un MSS di 1460 byte, corrisponde ad una finestra di soli 5 pacchetti.

Si noti che le ultime versioni degli stack TCP dei sistemi operativi più diffusi (Windows2000, Linux ad esempio) utilizzano il 32K come valore iniziale per la finestra di ricezione.

4.6 Perdite

Una parte delle misurazioni prodotte da `tcpstat` assume un significato particolare considerando la topologia di rete nella quale sono state catturate le tracce. Se, infatti, il punto di raccolta è sufficientemente prossimo alla rete interna è possibile supporre che vi sia un solo cammino tra i vari host che ad essa appartengono e la macchina che cattura le tracce. In una simile configurazione è ipotizzabile che non vi possano essere fenomeni di riordinamento

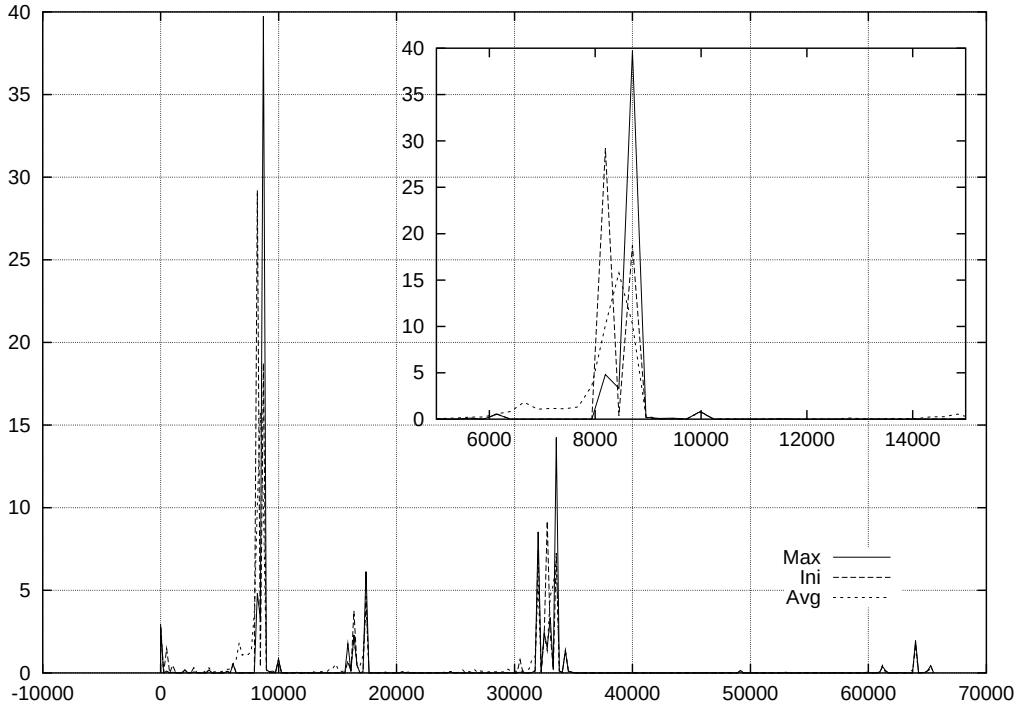


Figura 4.14. Receiver window annunciata periodo A.

di pacchetti a livello IP. Si noti come questa ipotesi non implica una rete perfetta in quanto sono comunque possibili eventuali perdite di pacchetti.

Nel nostro caso, la rete interna del Politecnico soddisfa queste condizioni, essendo basata su tecnologia *Ethernet*.

4.6.1 Out of order burst

Un *burst* di pacchetti fuori sequenza si riferisce ad un blocco di dati *out of order* contigui, potenzialmente più lungo di un singolo segmento.

Per *out of order* si intendono quei segmenti TCP che, durante il percorso dalla sorgente alla destinazione, subiscono un riordinamento. Questa situazione, dal punto di vista dell'osservatore fisso in un nodo della rete, può essere sintomo di due condizioni: un reale riordinamento a livello IP o la perdita e conseguente ritrasmissione ritardata di un segmento, perso in precedenza.

Le misurazioni relative a questo tipo di fenomeno sono ottenute da `tcpstat` confrontando i numeri di sequenza dei segmenti di dati che compongono il

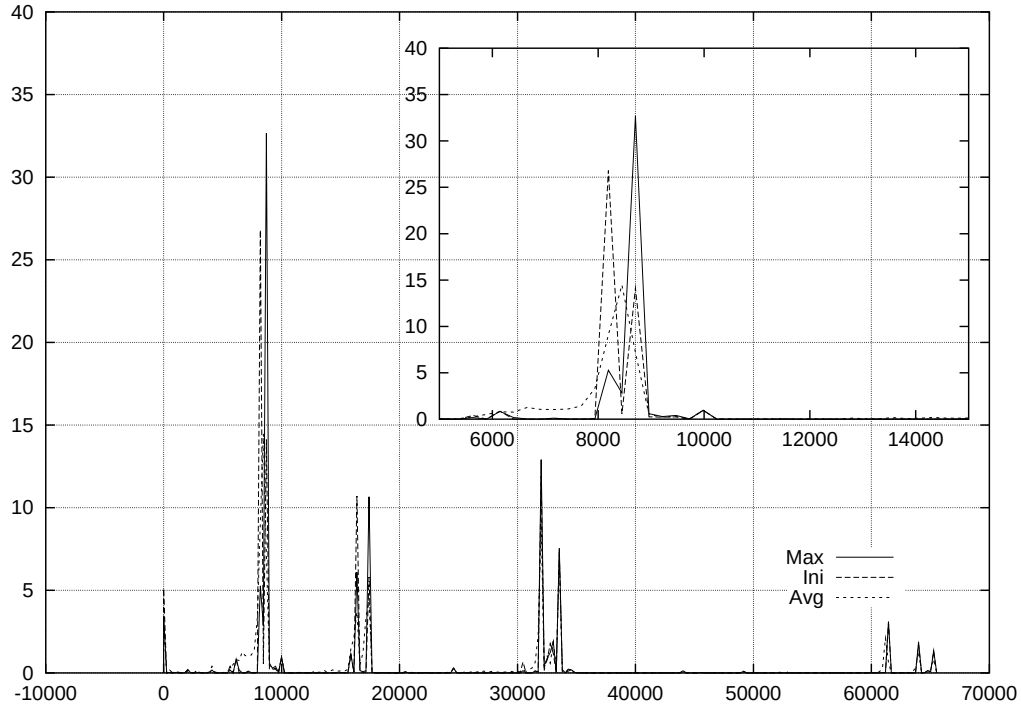


Figura 4.15. Receiver window annunciata periodo C.

flusso. Per poter identificare, quindi, il fenomeno è necessario che il pacchetto perso o fuori sequenza non sia l'ultimo, ma sia posizionato all'interno della sequenza e che almeno un segmento successivo sia ricevuto; ne consegue che la possibilità di misurazione del fenomeno è fortemente influenzata dalla lunghezza del flusso e di questo dato bisogna tenere conto nell'interpretazione dei dati.

Alla luce delle ipotesi fatte in precedenza a proposito della rete interna dell'Ateneo, i dati relativi ai *burst* di segmenti *out of order* su questa rete assumono un significato particolare: ipotizzando, infatti, l'assenza di riordinamento è possibile associare questi segmenti fuori sequenza sulle connessioni in uscita a segmenti persi sulla rete interna. Viceversa non è possibile distinguere l'evento *fuori sequenza* dall'evento *perdita dati* nel caso di flussi entranti, essendo essendo i due eventi possibili e identificati solo cone *out of orfder burst*.

Interessante è anche il valore assoluto della probabilità di evento *out of*

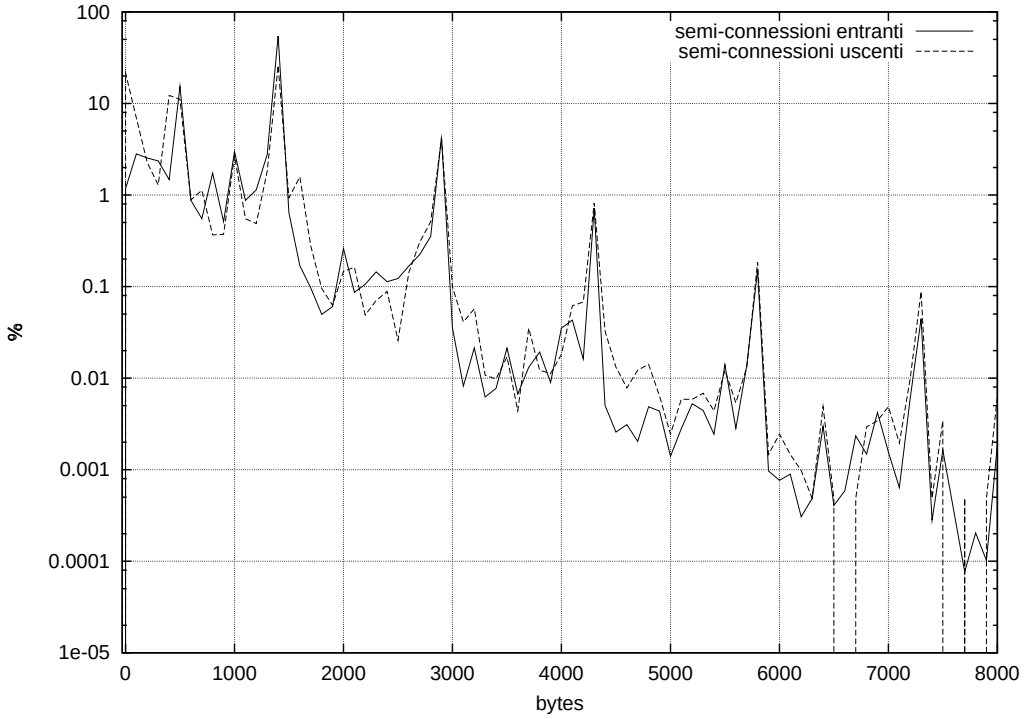


Figura 4.16. Out of order segment burst

	p{ooob} in byte	p{eventi ooob}
entranti	0.010	0.017
uscenti	0.001	0.001

Tabella 4.5. Probabilità di *out of order burst*.

order; tabella 4.5 riporta tale misura in termini di probabilità di *out of order burst* in byte, definita come

$$p\{ooob\} = \frac{totale\,bytes\,out\,of\,order}{totale\,bytes\,trasferiti}$$

e di probabilità di eventi di *out of order*, definita come

$$p\{eventi\,ooob\} = \frac{numero\,di\,ooob\,identificati}{numero\,totale\,di\,segmenti\,visti}$$

sia per semiflussi entranti che uscenti. Si noti come le due probabilità nel caso di flussi uscenti siano molto basse, in accordo con quanto ipotizzato

sulla rete interna.

In Figura 4.16 sono rappresentati, in scala logaritmica, i *burst* di segmenti *out of order* relativi ad una settimana nel periodo C. Sono evidenti picchi in corrispondenza di multipli interi di un MSS pari a 1460: i pacchetti di tali dimensioni, si veda in proposito la Sezione 3.2.5, sono i tra i più diffusi e a parità di probabilità di perdita è logico aspettarsi una loro predominanza.

4.6.2 Duplicate burst

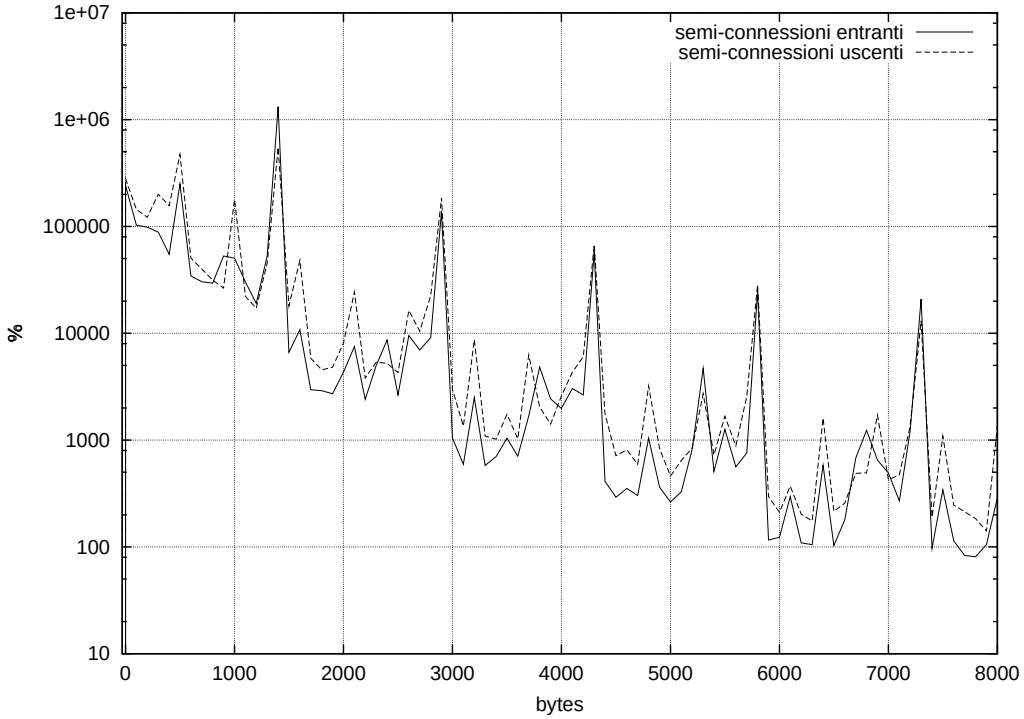


Figura 4.17. Duplicate packets burst

Un *burst* di pacchetti duplicati è una sequenza di segmenti che il programma vede ripetersi sulla rete. La sequenza può essere, come nel caso dei pacchetti *out of order,* più lunga di un singolo segmento. La probabilità di individuare correttamente l'evento di ritrasmissione è massima poiché non dipende da dati inviati successivamente successivi.

Anche in questo caso è possibile interpretare questi dati alla luce delle ipotesi fatte in precedenza sulla rete dell'Ateneo, in particolare i *burst* di

segmenti duplicati sulle semi-connessioni in uscita si possono considerare a tutti gli effetti come ritrasmissioni dovute a perdite di segmenti nel percorso verso la destinazione, mentre per quel che riguarda le semi-connessioni in entrata i pacchetti duplicati sono da considerarsi conseguenze di perdite sulla rete interna oppure a ritrasmissioni inutili da parte della sorgente.

Anche in questo caso vengono riportati i valori di probabilità di *duplicate burst* in byte calcolato come

$$p\{dupb\} = \frac{totale\,bytes\,duplicati}{totale\,bytes\,trasferiti}$$

e di probabilità *di eventi* di *duplicate burst* definito come

$$p\{eventi\,dupb\} = \frac{numero\,di\,dupb\,identificati}{numero\,totale\,di\,segmenti\,visti}$$

sia per semiflussi entranti che uscenti.

È interessante notare come, nonostante i semiflussi entranti e uscenti non abbiano nessuna correlazione, in quanto attraversano nodi indipendenti, la distribuzione sia di *out of order burst* sia di *duplicate burst* siano molto simili tra loro.

	p{dupb} in byte	p{eventi dupb}
entranti	0.009	0.012
uscenti	0.019	0.014

Tabella 4.6. Probabilità di *duplicate burst*.

In Figura 4.17 è possibile osservare, su scala logaritmica, la distribuzione di *burst* di segmenti duplicati. Si noti come, anche in questo caso, come era lecito aspettarsi vi sono picchi predominanti in corrispondenza di lunghezze pari a multipli interi di 1460.

4.7 RTT

Il *round trip time* misurato da `tcpstat` viene calcolato considerando il tempo che intercorre tra l'osservazione di un segmento in un senso e il corrispondente

acknowledgment nel verso opposto. In Figura 4.18 è possibile osservare la distribuzione del valore medio di questa misura, per flussi entranti e uscenti.

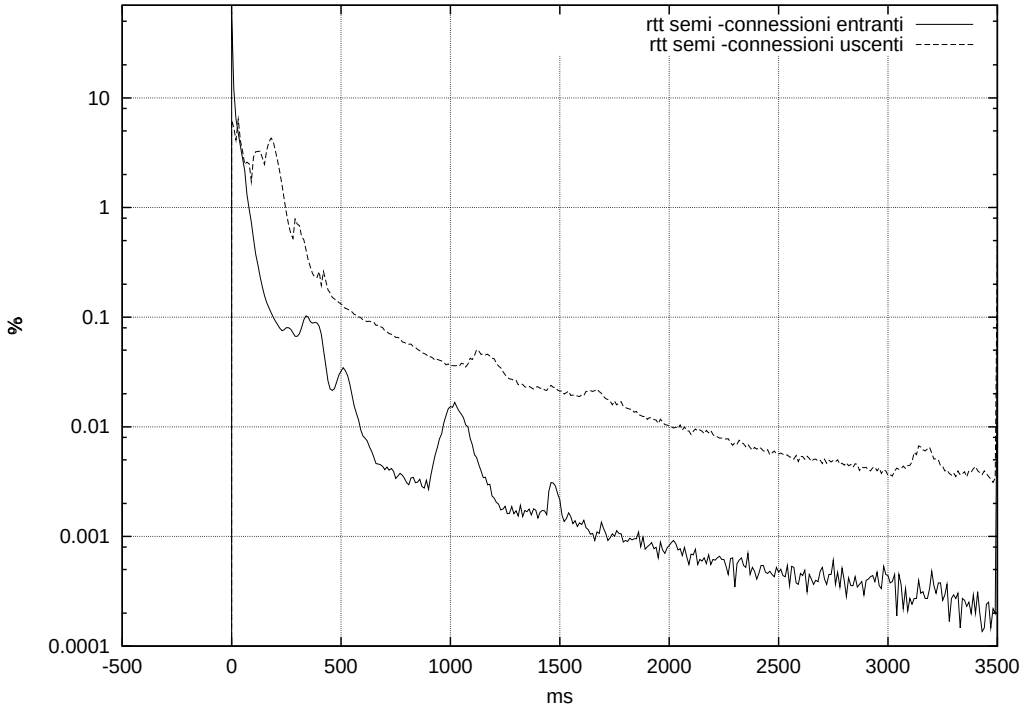


Figura 4.18. Distribuzione dei valori medi del *Round Trip Timr*

Il valore relativo alle semi-connessioni entranti si riferisce, in pratica, al tempo di latenza della rete interna mentre quello in direzione opposta va considerato come il tempo di risposta degli host sulla rete Internet rispetto al Politecnico di Torino. Si noti come alcune spiccate singolarità della curva relativa alle connessioni entranti si possano ritrovare sulla seconda curva spostate leggermente in avanti nel tempo. Si noti anche come il RTT misurato sulla rete interna sia molto più contenuto rispetto a quello misurato sulle connessioni uscenti. Anche in questo caso è posibile notare come risultino dei picchi nella distribuzione dei RTT dovuti all'uso di timer nei protocolli. ad esempio considerando la distribuzione dei flussi uscenti, la concentrazione di RTT appena oltre il valore 3 secondi è imputabile allo scadere del primo timer di ritrasmissione che è settato di default a tale valore: a seguito di una perdita il server ritrasmette il pacchetto dopo lo scadere del timer e il

successivo *acknowledge* viene associato da `tcpstat` alla prima trasmissione del segmento avvenuta, appunto, più di 3 secondi prima. Discorsi analoghi possono essere fatti per giustificare gli altri picchi.

Capitolo 5

Conclusioni

Il lavoro svolto si è dimostrato interessante sotto parecchi punti di vista.

La notevole quantità di tracce raccolte a parecchi mesi di distanza si sono rivelate un ottimo elemento di studio per comprendere l'evoluzione dell'utilizzo della rete dell'Ateneo e del funzionamento dei protocolli IP e TCP. La metodologia di raccolta non invasiva utilizzata, inolte, si è dimostrata efficace e rispettosa della privacy degli utenti senza interferire nel funzionamento della rete.

Il software sviluppato ha permesso un'analisi approfondita della tipologia del traffico e dei protocolli sui quali si basa Internet. La velocità di esecuzione e la scarsa quantità di memoria utilizzata lo rendono adatto ad un'elaborazione in tempo reale del traffico su collegamenti di rete di dimensioni paragonabili a quella del Politecnico di Torino e utile anche al di fuori di un ambito strettamente accademico.

Le misurazioni in relazione al protocollo IP hanno permesso di analizzare parametri quali la distribuzione degli indirizzi contattati e le percentuali dei protocolli trasportati da IP, confermando la predominanza del traffico TCP. Un'interessante esame del *time to live* dei singoli pacchetti ha permesso di caratterizzare la posizione reciproca tra la rete dell'Ateneo e gli host contattati sulla rete Internet. Infine lo studio della lunghezza dei segmenti ha confermato la predominanza di pacchetti o piccoli o della massima dimensione consentita.

Misure sul protocollo TCP hanno messo in evidenza caratteristiche legate

alle implementazioni degli stack dei sistemi operativi in particolare hanno mostrato come opzioni pensate per migliorar l'eficienza del protocollo siano in realtà poco sfruttate.

Analisi TCP a livello di flusso, introdotte per la prima volta in questo lavoro, hanno caratterizzato da un lato la tipologia delle connessioni che si incontrano in rete esaminandone la lunghezza, la durata e la asimmetria in termini di numero di segmenti e quantità di dati scambiati e dall'altro il comportamento del protocollo in una situazione reale esaminando parametri quali finestre di congestione e di ricezione, probabilità di pacchetti fuori sequenza e di pacchetti duplicati.

Appendice A

tcpstat

A.1 tcpstat

`tcpstat` è un software di analisi di traffico di rete. È in grado interpretare tracce raccolte tramite i più diffusi software di cattura (`tcpdump`, `snoop`, `etherpeek`) e ricavarne analisi statistiche a livello IP e TCP. Il programma nasce come evoluzione di `tcptrace 5.2` il cui autore Shawn Ostermann ne ha rilasciato il codice sorgente sotto una licenza che ne permette la modifica a patto che la licenza stessa rimanga inalterata.

A.2 Installazione

Prerequisiti necessari all'installazione del software sono:

- un compilatore `gcc` funzionante.
- gli interpreti `flex` e `bison` della gnu.
- la libreria `libpcap`.

Il programma si avvale di `autoconf` per la generazione del `Makefile`, quindi la compilazione avviene seguendo i seguenti passi:

```
./configure
make
make install
```

L'ultimo passo va eseguito dall'amministratore del sistema.

A.3 Utilizzo

Il programma va richiamato con la seguente riga di comando

```
tcpstat [-n] -a<file con reti interne> traccia
```

L'ultimo parametro sulla riga di comando è il nome del dump file da esaminare che deve essere in uno dei formati elencati in precedenza.

Il comando *-a* è obbligatorio e il suo parametro deve essere il nome di un file di testo nel quale vanno elencate le sottoreti che vanno considerate interne durante l'analisi. Il file, in formato ASCII, deve contenere una serie di righe, terminate da *newline*, nel formato

```
indirizzo
netmask
indirizzo
netmask
...
```

Per ogni indirizzo viene identificata, in base alla netmask, una sottorete i cui indirizzi IP vanno considerati interni. Viene di seguito riportato un esempio di file:

```
130.192.0.0
255.255.0.0
193.42.134.0
255.255.255.0
193.204.112.0
255.255.255.0
193.205.59.0
255.255.255.0
```

Il programma attualmente ammette l'uso di un massimo di 20 sottoreti interne.

Lo switch opzionale `-n` impone al programma di non risolvere il nome degli host e di presentarne l'indirizzo IP numerico.Il programma, una volta installato, va richiamato con la seguente riga di comando:

A.4 Esempio di output.

Per una descrizione dettagliata dei files di analisi prodotti da `tcpstat` si veda la Sezione 2.3.

A.5 Segnali

Il programma è in grado di intercettare segnalazioni inviate tramite il comando UNIX `kill`.

`SIGUSR1:` tale segnale inviato durante l'elaborazione dei dati, produce sullo standard error, una serie di mesaggi diagnostici sullo status del programma.

`SIGINT:` tale segnale impone la fine dell'elaborazione previo salvataggio di tutti i dati elaborati fino al momento dell'interruzione.

Appendice B

Strumenti informatici

L'ingente mole di informazioni generate da `tcpstat` durante l'esame delle tracce, pur se organizzata razionalmente, è spesso di difficile interpretazione perchè molto frammentata. Il software ogni quarto d'ora stampa i risultati ottenuti fino a quel momento in una serie di sotto-directory: per poter riaggregare i risultati e avere panoramiche che coprissero periodi più lunghi (tipicamente una settimana) è stato necessario lo sviluppo di una serie di strumenti che rendessero più fruibili questi dati. I programmi qui presentati si possono trovare nella directory `utils` dell'archivio contenente `tcpstat`.

Di seguito se ne da una breve descrizione, unitamente a esempi di utilizzo.

B.1 `plot6`

Il software `plot6` è in grado di ottenere istogrammi aggregati su un periodo a scelta, a partire dall'output, suddiviso in quarti d'ora, di `tcpstat`. L'esecuzione del programma senza argomenti genera le seguenti informazioni sulle modalità di l'utilizzo:

```
[carpani@gramigna carpani]$ ./plot6
Usage:
./plot6 <-d directory> <-v>  [<-l> | <-n> <-g> \
  <-b base_name> <-f data_file[,data_file]> "aaaa/mm/gg hh:mm" "aaaa/mm/gg hh:mm"]
-d: directory with the output of tcpstat
-v: verbose debugging
-l: list the available trace results, ordered by date, only
-n: print results in % (normalize)
```

```
-g: print results to a .gnu file (suitable for gnuplot)
-b: title
-f: output file from tcpstat you want to aggregate
```

Di seguito verranno brevemente descritti i comandi e le opzioni utilizzabili.

-d: il parametro è obbligatorio e deve essere formato da una stringa contenente l'elenco delle directories dove si trovano i dati di output di `tcpstat`. Il path deve essere relativo alla directory corrente.
Es. `-d ../dati_1,/mnt/altro_hd/dati_2`

-v: questa opzione abilita messaggi diagnostici.

-l: questa opzione permette di ottenere la lista delle date per le quali sono disponibili dei dati senza elaborarli effettivamente.

-n: questo parametro facoltativo impone al programma di normalizzare l'istoframma finale in percentuale.

-g: questa opzione genera due files, il primo con l'istogramma richiesto e il secondo con uno scheletro di file di input per `gnuplot`.

-b: l'opzione vuole come parametro il nome di base che avranno i files generati da `plot6`.

-f: l'opzione vuole come parametro il nome del file di output di `tcpstat` che si vuole aggregare; si faccia riferimento alle tabelle riportate in Sezione 2.3.

Gli ultimi due parametri della riga di comando devono essere le date iniziali e finali del periodo che si vuole aggregare. Il formato è

```
aaaa/mm/gg hh:mm
```

Si tratta dello stesso formato ottenuto con l'opzione `-l` a meno del giorno della settimana (si veda l'esempio di utilizzo).

Perchè il programma funzioni correttamente, è opportuno che le directory contenenti l'elaborazione di `tcpstat` abbiano, insita nel nome, l'indicazione del giorno e dell'ora di inizio della cattura; ad esempio per un file creato a partire dalle 5:00 am del giorno 20 giugno 2000 il nome corretto è: `05_00_20_Jun_2000.dump.gz.out`

B.1.0.1 Esempio di utilizzo

Con l'opzione `-l` è possibile osservare quali sono i risultati disponibili:

```
[carpani@verza carpani]$ plot6 -l -d ../diskone1_bis3/
2000/05/29 11:00 Mon
2000/05/29 17:00 Mon
2000/05/29 23:00 Mon
...
2000/05/31 17:00 Wed
2000/05/31 23:00 Wed
```

Per aggregare i dati relativi al file `time_tot` del periodo che va dalle 17:00 del 29 maggio alla stessa ora del 31 si utilizzi:

```
plot6 -g -n -b tot_time -f tot_time -d ../../../diskone1_bis3/ \
"2000/05/29 17:00" "2000/05/31 17:00"
```

La riga di comando va digitata tutta su un unica riga. Si otterranno due files in uscita

`tot_time.dat` conterrà i dati dell'istogramma aggregato.

`tot_time.cmd` conterrà un file di comandi per gnuplot adatto alla rappresentazione grafica dei dati.

B.2 get_address

`get_address` è in grado di analizzare i file `addresses` prodotti da `tcpstat` alla fine dell'elaborazione delle tracce. In questi file per ogni indirizzo viene indicato il numero di pacchetti di cui è destinatario. `get_address` è in grado di aggregare i dati di più files a partire da un albero di directory simile a quello descritto in Sezione B.1.

Il programma, eseguito senza parametri, fornisce la seguente schermata di aiuto:

```
[carpani@gramigna carpani]$ get_address
Choose at least one from -a -l or -r
Usage:
bin/get_address <-d directory> -v [-r|-l|-a] "aaaa/mm/gg hh:mm" "aaaa/mm/gg hh:mm"]
-d: directory with the output of tcptrace.ac
-v: verbose
-r: resolve the first 200 names
-l: list available dates
-a: dump all hits
```

Le opzioni sono simili a quelle utilizzate da `plot6`:

-d: è il parametro obbligatorio con l'indicazione delle directories con l'output di `tcpstat`.

-v: abilita messaggi diagnostici

-l: permette di ottenere la lista delle date per le quali sono disponibili dei dati.

-r: impone la risoluzione inversa del nome dei primi 200 indirizzi più contattati nel periodo indicato.

-a: stampa sullo standard output il totale degli host contattati e il numero di contatti.

B.3 udp_what

`udp_what` è in grado di elaborare l'output di `tcpdump` e di ricavarne statistiche in relazione di pacchetti UDP. Il programma non prevede opzioni. In output si ottengono un elenco dei vari tipi di pacchetti UDP riscontrati e il loro numero.

B.3.0.2 Esempio di utilizzo

```
[carpani@gramigna carpani]$ tcpdump -r traccia.dump udp | udp-what
******************
syslog = 2
zephyr = 10
```

```
socks = 4
bootpc = 117
radius = 1
frag = 572
udp (unidentified) = 27947
time = 43
datametrics = 1
ingreslock = 1
codasrv = 4
radacct = 1
ntp = 5645
snmp = 4291
...
```

B.4 mk_asym

Il file di output `log_complete` di `tcpstat` contiene tutti i dati ricavati dal programma relativamente alle connessioni TCP correttamente concluse. Lo scopo del file è quello di permettere una rielaborazione di queste informazioni senza per questo dovere eseguire nuovamente `tcpstat` sulle tracce in esame.

Nel caso specifico del presente lavoro, è stato necessario recuperare informazioni circa l'asimmetria delle connessioni TCP, informazione, questa, non prodotta direttamente da `tcpstat`. Allo scopo di facilitare questa operazione di postprocessing è stato creato `mk_asym`.

Il programma, eseguito senza parametri, fornisce la seguente schermata di aiuto:

```
[carpani@gramigna carpani]$ ./mk_asym
Usage:
./mk_asym <-d directory> <-v> <-l> "aaaa/mm/gg hh:mm" "aaaa/mm/gg hh:mm"]
-d: directory with the output of tcpstat
-v: verbose
-l: list available dates
```

Le opzioni sono del tutto simili a quelle di `plot6`:

-d: è il parametro obbligatorio con l'indicazione delle directories con l'output di `tcpstat`.

-v: abilita messaggi diagnostici

-l: permette di ottenere la lista delle date per le quali sono disponibili dei dati.

Il programma genera in uscita, nella directory corrente, i files `asym_1.dat` e `asym_2.dat` contenenti rispettivamente i dati relativi all'asimmetria delle connessioni in byte e in segmenti.

Il programma, scritto in linguaggio *perl*, è predisposto per l'analisi di files di tipo `log_complete` e viene qui presentato in quanto utile punto di partenza per eventuali successive necessità di postprocessing.

Appendice C

Rete informatica del Politecnico di Torino

C.1 Rete informatica del Politecnico di Torino

Il programma `tcpstat` è in grado di analizzare il traffico internet e di fornire statistiche a livello IP e TCP sul flusso di dati. Le misurazioni ottenute si basano sulla possibilità di poter considerare come *interne* una serie di sottoreti verso le quali ci si aspetta di non osservare riordinamento di pacchetti. Nel presente studio sono state considerate interne le seguenti sottoreti facenti parte della rete informatica del Politecnico di Torino. Si noti come esse comprendano anche strutture al di fuori degli edifici principali siti in C.so Duca degli Abbruzzi. A titolo informativo, il numero di host presenti sulla rete interna del politecnico è superiore a 5000.

Sottorete	Ubicazione	Subnet
130.192.1.0	Ingegneria	255.255.255.0
130.192.2.0	Ingegneria	255.255.255.0
130.192.3.0	Ingegneria	255.255.255.0
130.192.4.0	Ingegneria	255.255.255.0
130.192.5.0	Ingegneria	255.255.255.0
130.192.6.0	Ingegneria	255.255.255.0

130.192.7.0	Ingegneria	255.255.255.0
130.192.8.0	Ingegneria	255.255.255.0
130.192.9.0	Ingegneria	255.255.255.128
130.192.9.128	Ingegneria	255.255.255.128
130.192.10.0	Ingegneria	255.255.255.0
130.192.11.0	Ingegneria	255.255.255.0
130.192.12.0	Ingegneria	255.255.255.0
130.192.13.0	Ingegneria	255.255.255.0
130.192.14.0	Ingegneria	255.255.255.0
130.192.15.0	Ingegneria	255.255.255.0
130.192.16.0	Ingegneria	255.255.255.0
130.192.17.0	Ingegneria	255.255.255.0
130.192.18.0	Boggio	255.255.255.0
130.192.19.0	Ingegneria	255.255.255.0
130.192.20.0	Ingegneria	255.255.255.0
130.192.21.0	Ingegneria	255.255.255.0
130.192.22.0	Ingegneria	255.255.255.0
130.192.23.0	Ingegneria	255.255.255.0
130.192.24.0	Ingegneria	255.255.255.0
130.192.25.0	Ingegneria	255.255.255.0
130.192.26.0	Ingegneria	255.255.255.0
130.192.27.0	Ingegneria	255.255.255.0
130.192.28.0	Ingegneria	255.255.255.0
130.192.29.0	Ingegneria	255.255.255.0
130.192.30.0	Ingegneria	255.255.255.0
130.192.31.0	Ingegneria	255.255.255.0
130.192.32.0	Ingegneria	255.255.255.0
130.192.33.0	Mondovì	255.255.255.0

130.192.34.0	Ivrea	255.255.255.0
130.192.35.0	Aosta	255.255.255.0
130.192.36.0	Ingegneria	255.255.255.0
130.192.37.0	Ingegneria	255.255.255.0
130.192.38.0	Ingegneria	255.255.255.0
130.192.39.0	Ingegneria	255.255.255.0
130.192.41.0	Architettura	255.255.255.0
130.192.42.0	Architettura	255.255.255.0
130.192.43.0	Architettura	255.255.255.0
130.192.44.0	Architettura	255.255.255.0
130.192.47.0	Ingegneria	255.255.255.0
130.192.48.0	Ingegneria	255.255.255.0
130.192.50.0	Lingotto	255.255.255.0
130.192.51.0	Alessandria	255.255.255.0
130.192.55.0	Ingegneria	255.255.255.0
130.192.56.0	Ingegneria	255.255.255.0
130.192.57.0	V.C.Massaia	255.255.255.0
130.192.75.0	Vercelli	255.255.255.0
130.192.76.0	Vercelli	255.255.255.0
130.192.78.0	V.Cavalli	255.255.255.0
130.192.79.0	C.soFrancia	255.255.255.0

Bibliografia

[1] F. Douglis G. Glass M. Rabinovich A. Feldmann, R. Caceres. "Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments". *Proceedings of IEEE INFOCOM '99*, pages 107–116, 1999.

[2] N. Sharma N. Cardwell M. Brown T. Landray D. Pinnel A. Karlin H. Levy A. Wolman, G. Voelker. "Organization-Based Analysis of Web-Object Sharing an Caching". *Proceedings of the 2nd USENIX Conference on Internet Technologies and Systems*, October 1999.

[3] A.Feldman. "BLT: Bi-Layer Tracing of HTTP and TCP/IP". *Proceedingsof WWW-9*, May 2000.

[4] P. Almquist. RFC 1349: Type of service in the Internet Protocol suite, July 1992.

[5] M. J. Feeley B. M. Duska, D. Marwood. "The Measured Access Characteristics of World-Wide-Web Client Proxy Caches". *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pages 23–36, December 1997.

[6] M. E. Crovella C.R. Cunha, A. Bestavros. "Characteristics of WWW Client-based Traces". *Technical Report TR-95-010 Boston University Computer Science department*, June 1995.

[7] S. Seshan V. Padmanabhan R. H. Katz H. Balakrishnan, M. Stemm. "TCP Behavior of a Busy Internet Server: Analysis and Solutions". *Proceedings of IEEE INFOCOM '98*, pages 252–262, March 1998.

[8] V. Jacobson, R. Braden, and D. Borman. RFC 1323: TCP extensions for high performance, May 1992.

[9] M. St. Johns. RFC 1413: Identification protocolr, January 1993.

[10] K. Fall S. Floyd J. Heidermann A. Helmy P. Huang C. McCanne K. Varadhan Y. Xu H. Yu L. Brslau, D. Estrin. "Advances in Network Simulation". *IEEE Computer*, 33(5):59–67, May 2000.

[11] J. Almeida A. Broder L. Fan, P. Cao. "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol". *Proceedings of ACM SIGCOMM '98*, pages 254–265, 1998.

[12] W. Stevens M. Allman, V. Paxson. RFC 2581: TCP Congestion Control, 1999.

[13] A. Bestavros M. E. Crovella. "Self Similarity in World Wide Web Traffic: Evidence and Possible Causes". *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[14] B. Mah. "An Empirical Model of HTTP Network Traffic". *Proceedings of IEEE INFOCOM '97*, April 1997.

[15] D.Ott F.D.Smith M.Christiansen, K.Jeffay. "Tuning RED for Web Traffic". *Proceedings of ACM SIGCOMM 2000*, pages 130–150, September 2000.

[16] A. Bradley M. E. Crovella P. Barford, A. Bestavros. "Changes in Web Client Access Patterns: Characteristics and Caching Implications". *Wolrd Wide Web, Special Issue on Characterization and Performance Evaluation*, 2:15–28, 1999.

[17] M. E. Crovella P. Barford. "Generating Representative Web Workloads for Network and Server Performance Evaluation". *Proceedings of ACM SIGMETRICS '98*, pages 151–160, 1998.

[18] R. Caceres D. Mitzel D. Mestrin P. Danzig, S. Jamin. "An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations". *Internetworking: Research and Experience*, 3(1):1–26, 1992.

[19] S. Jamin P. Danzig. "tcplib: A library of TCP Internetwork Traffic Characteristics". *USC Technical report*, 1991.

[20] V. Paxons. "Empirically Derived Analytic Models of Wide-Area TCP Connections". *IEEE/ACM Transactions on Networking*, 2:316–336, August 1994.

[21] M.E.Crovella P.Barford. "A Performance Evaluation of Hypertext Transfer Protocol". *Proceedings of ACM SIGMETRICS '99*, (188-197), May 1999.

[22] J. Postel. RFC 791: Internet Protocol, September 1981.

[23] J. Postel. RFC 793: Transmission control protocol, September 1981.

[24] S. Jamin D. Mitzel R. Caceres, P. Danzig. "Characteristics of Wide-Area TCP/IP Conversations". *Proceedings of ACM SIGCOMM '91*, 1991.

[25] E. A. Brewer S. D. Gribble. "System Design Issues for Internet Middleware Services". *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December 1997.

[26] W. S. ClevelandD. LinD. X. Sun. "IP Packet Generation: Statistical Modelsfor TCP Start Times Based on Connection-Rate Superposition". *Proceedings of ACM SIGMETRICS 2000*, pages 166–177, June 2000.

[27] L. Wong T. Ott, T. Lakshman. "SRED: Stabilized RED". *Proceedings IEEE INFOCOM '99*, pages 1346–1355, March 1999.

[28] S. Floyd V. Paxons. "Wide-Area Traffic: The Failure of Poisson Modeling". *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[29] D. Saha K. Shin W. Feng, D. Kandlur. "Blue: A New Class of Active Queue Management Algorithms". *University of Michigan Technical Report CSE-TR-3387-99*, April 1999.

[30] A. Wolman. "On the Scale and Performance of Cooperative Web Proxy Caching". *Proceedings of ACM SOSP '99*, pages 16–31, December 1999.