Tesina TIR

Analisi di traffico reale su una rete domestica

ver. 1.0

SOMMARIO

1. INTRODUZIONE

Lo scopo del presente elaborato è di raccogliere ed in seguito analizzare i dati di una rete domestica per rispondere al seguente quesito della tesina:

Analisi di traffico reale in una rete domestica

Ai candidati viene proposto di raccogliere ed in seguito analizzare i dati di una rete domestica, proponendo come tesina finale una analisi dei dati ottenuti sotto diversi punti di vista, riferiti ai protocolli di trasporto (es. percentuale dati UDP e TCP, durata connessioni TCP, valori RTT e congestion window stimati, ecc.)

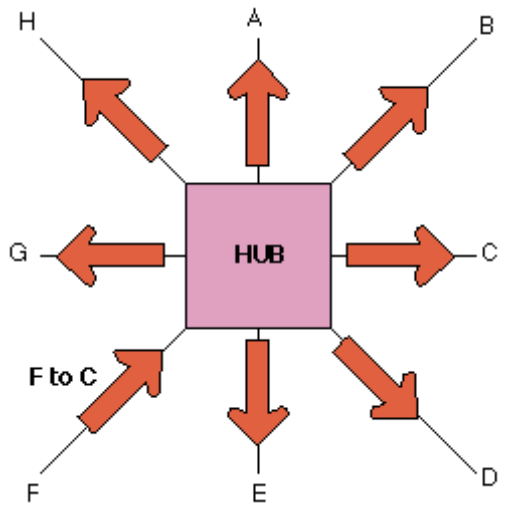
Workflow proposto:

1. Raccolta dati in formato PCAP di una rete domestica secondo diversi orario di utilizzo
2. Valutazione di software di analisi esistenti, o preparazione software di analisi che fornisca i dati richiesti sul dataset
3. Presentazione dei risultati tramite tabelle e analisi degli stessi in un elaborato

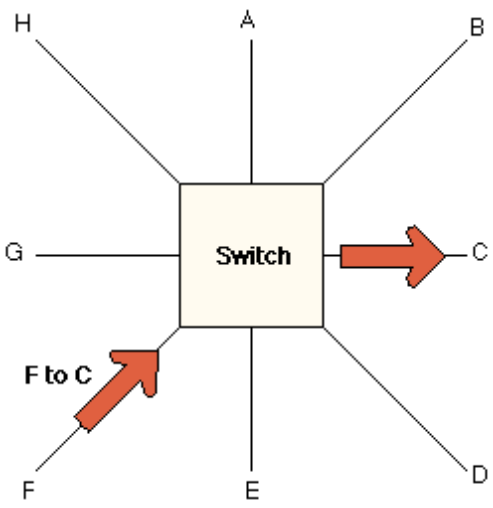
L'importanza dell'access point per l'esecuzione dell'esperimento

La comunicazione fra due dispositivi appartenenti alla stessa rete locale, quale ad es. una rete domestica, avviene mediante un dispositivo di rete che funge da nodo di smistamento; tale dispositivo è posizionato nel centro stella ed è dotato di più porte: i dati in arrivo da una qualsiasi porta a seconda che sia uno switch o un hub vengono inoltrati su una o su tutte le porte.

La differenza sostanziale fra hub e switch è che il primo, lavorando a livello 1 (fisico) del modello ISO/OSI, replica incondizionatamente il traffico proveniente da una qualsiasi porta su tutte le altre (fig. 1a), con ovvi problemi di congestione della banda disponibile ed il conseguente aumento delle collisioni, mentre il secondo, lavorando a livello 2 (data link) del modello ISO/OSI, inoltra i dati mediante una corrispondenza MAC address destinazione – porta (fig. 1b), ottimizzando l'uso della banda.



(fig. 1a - hub)



(fig. 1b - switch)

L'utilizzo degli switch è subentrato anche nelle reti domestiche migliorando le prestazioni generali della comunicazione, ma qualora si voglia raccogliere il traffico dell'intera rete LAN aggiunge alcune complicazioni in quanto il dispositivo attaccante, in virtù del principio di funzionamento dello switch, potrà apprezzare esclusivamente il proprio traffico.

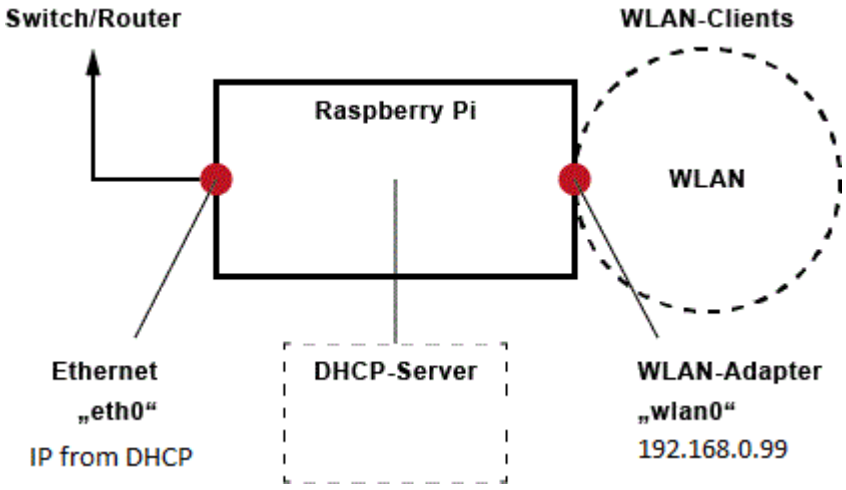
Essendo uno dei nostri scopi quello di acquisire l'intero traffico di una rete LAN domestica possiamo mettere in campo diverse metodologie, alcune lecite ed altre illecite: ad es. un metodo illecito consiste nello sniffing del traffico mediante un attacco "ARP Spoofing" mediante il quale tutti i pacchetti in transito passano attraverso il dispositivo attaccante per poi essere re-inoltrati alla/e vittima/e; mentre un metodo lecito consiste nella selezione di candidati volontari ad utilizzare un particolare access point che consentirà di intercettare e memorizzare tutti i pacchetti in transito dall'interfaccia wireless all'interfaccia ethernet.

Abbiamo preferito questo secondo metodo in primo luogo perché non ci costringeva a commettere un reato, ed in secondo luogo perché operavamo su un mezzo trasmissivo wireless che tipicamente è quello che viene utilizzato dalla maggior parte dei dispositivi presenti in un'abitazione.

La presente tesina è organizzata come segue: nel capitolo 2 viene spiegato come configurare adeguatamente il dispositivo per creare una rete wireless ed intercettarne il suo traffico, nel capitolo 3 vengono valutati alcuni software utili a questo elaborato, nel capitolo 4 viene spiegato come installare il software selezionato, nel capito 5 vengono studiate alcune metriche per l'analisi del traffico, nel capito 6 viene spiegato come sono state implementate le metriche del capitolo precedente, nel capitolo 7 vengono utilizzati gli script implementati e viene fatta l'analisi del traffico acquisito ed infine nel capitolo 8 vengono fatte le conclusioni su questo studio.

2. PREDISPOSIZIONE APPARATO DI TEST

Concordando sull'utilizzo di una metodologia lecita e sul coinvolgimento volontario e consapevole degli utenti all'esperimento, si è scelto di utilizzare un dispositivo che consentisse di attivare una nuova rete wifi e che contemporaneamente alla funzionalità di bridge fra rete wireless e rete cablata consentisse l'acquisizione del traffico. Inoltre, essendo un'attività che potenzialmente andava eseguita in luoghi diversi si è considerata la trasportabilità e l'assenza di particolari configurazioni che rendessero il dispositivo installabile in diversi ambienti.
Per queste ragioni abbiamo scelto di utilizzare un dispositivo Raspberry pi model B+, che date le dimensioni ridotte, la presenza di una porta ethernet e diverse porte usb e l'utilizzo mediante sistema operativo linux, si prestava bene allo scopo del presente elaborato.
L'idea è stata di configurare tale dispositivo per connettersi alla rete lan per mezzo di un comune cavo rj45, ottenendo un ip dinamico dall'eventuale DHCP presente nella LAN, di creare una rete wireless per mezzo di un adattatore wifi e di inoltrare tutto il traffico wifi verso la porta ethernet, potendo così intercettare e memorizzare il traffico in transito sull'interfaccia wireless (fig. 2).
Sicuramente questa soluzione è accettabile per un contesto domestico caratterizzato da un traffico limitato generato da pochi dispositivi ma impensabile per un contesto, come ad esempio un albergo, con tanti client perché causerebbe senza dubbi perdita di pacchetti.



(fig. 2 – schema apparato di test)

Setup Raspberry - base

Il setup del raspberry ed il primo avvio effettuato seguendo i seguenti passi:

1) download della iso con Raspbian Jessie (https://www.raspberrypi.org/downloads/raspbian);
2) formattazione SD con "sd formatter" avendo cura di settare su ON l'opzione "Format Size Adjustment";
3) scrittura della iso sull'sd, mediante il tool per Windows Win32DiskImager;
4) Inserimento SD nel raspberry e primo avvio mediante GUI;
5) Avviare Terminal e lanciare "Sudo raspi-config" per i vari settaggi, tra cui:
    a. expand file system (per utilizzare tutto lo spazio a disposizione della scheda SD);
    b. change user password (utente pi) -> "TesinaTIR2016";
    c. boot options -> selezionare "Console" (per avviare il dispositivo senza GUI);
    d. Internationalization Options -> Change Locale -> It.UTF8;
    e. Internationalization Options -> TimeZone -> Europe -> Rome;

   f. Internationalization Options -> Keyboard -> Generic 105 -> Other -> Italian -> Italian WinKeys -> Right Alt (AltGr) -> Right Alt (AltGr) ;
   g. Internationalization Options -> WiFi Country -> IT;
   h. Advanced -> HostName -> piTIR;
   i. Advanced -> SSH -> Yes (per attivare SSH);
6) Riavviare ed accedere con l'utente "pi"
7) Modificare /etc/apt/sources.list, decommentando deb-src per accedere ai repository dei codici sorgenti;
8) Sudo passwd per modificare la password dell'utente root -> "TesinaTIR2016";
9) Sudo apt-get update e poi Sudo apt-get upgrade per aggiornare il sistema

Setup Raspberry – AP

Seguire le seguenti istruzioni per configurare il raspberry come access point:

1) Verificare che la scheda di rete wireless si stata riconosciuta dal sistema ed impostare un ip statico per l'interfaccia wlan0:

```
ifconfig
nano /etc/network/interfaces

auto wlan0
iface wlan0 inet static
address 192.168.0.99
netmask 255.255.255.0
```

2) Verificare se la scheda preveda la modalità AP (deve esserci AP in "Supported Interface Mode")

```
iw list
```

3) Installare e configurare modulo per AP. Potrebbe essere necessario utilizzare una versione di hostapd compilata appositamente per la scheda di rete wireless in uso.

```
apt-get install hostapd
nano /etc/hostapd/hostapd.conf

interface=wlan0
driver=rtl871xdrv
ssid=pi_wifi
hw_mode=g
channel=8
macaddr_acl=0
auth_algs=1
wmm_enabled=0
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=prova2016
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

```
nano /etc/dafault/hostapd
specificare "/etc/hostapd/hostapd.conf" in DAEMON_CONF
```

Setup Raspberry – DHCP

Seguire le seguenti istruzioni per attivare il DHCP sull'interfaccia wlan0:

1) Installare e configurare DHCP:

```
apt-get install udhcpd
nano /etc/udhcpd.conf

impostare start-end range 192.168.0.20 -> 50
interface wlan0
remaining yes
opt dns 8.8.8.8 4.4.4.4
opt router 192.168.0.99

nano /etc/default/udhcpd
remmare #DHCPD_ENABLED="no"
```

Setup Raspberry – IP forward

Seguire le seguenti istruzioni per configurare il forward wlan0/eth0:

1) Attivazione IP forward e creazione regole iptables

```
nano /proc/sys/net/ipv4/ip_forward
indicare 1

/etc/sysctl.conf
Aggiungere net.ipv4.ip_forward=1

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT

per salvare
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"

per avvio automatico
/etc/network/interfaces
up iptables-restore < /etc/iptables.ipv4.nat
```

Setup Raspberry – Wireshark

Seguire le seguenti istruzioni per installare Wireshark, un network protocol analyzer:

1) Installazione Wireshark

```
Sudo apt-get install wireshark
nano /usr/share/wireshark/init.lua
disable_lua = true
```

2) Per avviare la cattura del traffico in transito dall'interfaccia wlan0

```
Su
Touch test.pcap
Chmod o=rw test.pcap
sudo tskark -i wlan0 -w /test.pcap
Chmod pi /test.pcap
```

3. VALUTAZIONE SOFTWARE DI ANALISI

Al fine di individuare un software adeguato per l'elaborazione della presente tesina sono stati analizzati e valutati alcuni software soffermandosi sulle caratteristiche elencate di seguito:

- open source/closed source, free/pagamento
- ambiente windows/unix-linux
- linea di comando/ide
- programmazione mediante linguaggio di scripting
- estendibile mediante package e disponibilità di package statistici

La nostra attenzione si è focalizzata su tre prodotti: Matlab, Octave, R e Wireshark.

Matlab

Matlab è senza dubbio il prodotto leader per questa tipologia di applicazioni, infatti le caratteristiche del prodotto e dei package presenti di default e acquistabili da terze parti coprono qualsiasi esigenza, superando di gran lunga le altre piattaforme, ma purtroppo è a pagamento.

Octave

Octave è concettualmente paragonabile a Matlab, open source e free, per tutte le piattaforme, supporta quasi tutte le istruzioni base di matlab. Purtroppo i package sviluppati per MatLab spesso non sono compatibili.

R

R è una delle migliori piattaforme per l'analisi di dati e possiede circa 2000 package per l'analisi statistica, che è sostanzialmente il campo d'interesse per l'elaborato. E' disponibile per tutte le piattaforme ed è free.

Wireshark

Wireshark è un software per analisi di protocollo e packet sniffer utilizzato per l'analisi del traffico di rete ed utilizzato da network engineers e comunità scientifiche.

La tabella seguente (tab. 1) riepiloga le caratteristiche di ogni software, il cui confronto ci ha fatto propendere per l'accoppiata R e Wireshark, perché quest'ultimo dispone di grafici e statistiche di base che vengono elaborate in brevissimo tempo, mentre R seppur più lento fornisce un grado di liberà superiore circa l'analisi dei dati.

Software	Open source	Unix/linux	Ide	Scripting	packages	free
Matlab	No	Si	Si	Si	Si	No
Octave	Si	Si	No	Si	Si	Si
R	Si	Si	Si	Si	Si	Si
Wireshark	Si	Si	Si	Si, LUA	No	Si

(tab. 1)

4. INSTALLAZIONE DI WIRESHARK ED R SU UBUNTU

Data la sua semplicità si ommette la descrizione dell'installazione di wireshark, mentre merita un approfondimento l'installazione di R.
Per installare R mediante apt è necessario aggiungere il repository del CRAN che contiene i sorgenti per il sistema operativo ubuntu ed aggiungere la chiave pubblica per verificare l'integrità e l'autenticità dei pacchetti scaricati.

```
sudo sh -c 'echo "deb http://cran.rstudio.com/bin/linux/ubuntu trusty/" >> etc/apt/sources.list'
gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9
gpg -a --export E084DAB9 | sudo apt-key add -
```

Lanciare i seguenti comandi che consentono dapprima per aggiornare l'elenco dei package e quindi di installare R.

```
sudo apt-get update
sudo apt-get -y install r-base
```

Per testare l'installazione digitare il comando "R" che dovrebbe produrre la visualizzazione dei dati relativi alla versione corrente, poi q()+invio per uscire:

```
R

R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

…
```

Per lo svolgimento del progetto è stata selezionata un'ide compatibile con R chiamata Rstudio; per la sua installazione è necessario scaricare il package da https://www.rstudio.com/ ed installarlo da shell con il seguente comando:

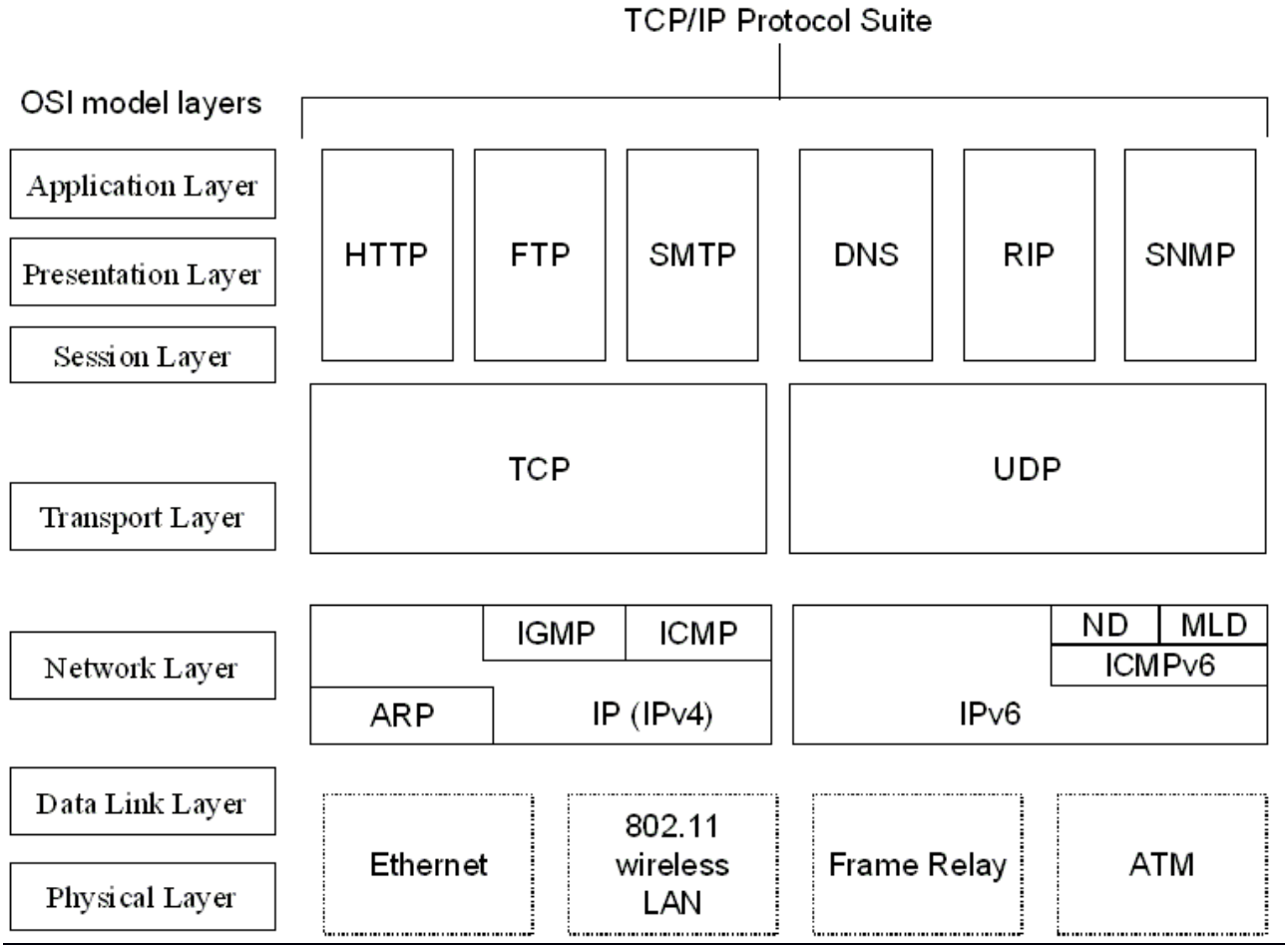
```
sudo dpkg -i rstudio-<version>.deb
rstudio
```

Per lanciare gli script sviluppati per il presente elaborato è necessario installare dal prompt di R alcuni package aggiuntivi:

```
Install.packages("stringr")
Install.packages("argparser")
install.packages("roxygen2")
install.packages("devtools")
```

5. METRICHE

In considerazione della struttura a strati del protocollo TCP/IP (fig. 3) e della finalità di valutare il traffico da diversi punti di vista abbiamo stabilito alcune metriche che ci avrebbero guidato sia nella scelta delle statistiche e grafici da elaborare per con Wireshark sia nello sviluppo degli script che nell'interpretazione dei dati.



(fig. 3 – stack TCP)

Per flusso tcp si intende un flusso identificato dalla quintupla TCP + srcIP + srcPorta + dstIP + dstPorta, inizializzato dal three way handshake e terminato da un segmento FIN/RST.
Per flusso udp si intende un flusso identificato dal primo all'ultimo pachetto avente quintupla UDP + srcIP + srcPorta + dstIP + dstPorta.

Dati generali dell'acquisizione (da wireshark):

- Ora inizio/Ora fine
- Totale pacchetti / totale bytes
- Totale pacchetti persi
- Tipologia di utenza sottoposta al test
- Tipologia di traffico generato
- Eventuali anomalie rilevate

Traffico IN/OUT, traffico intranet (da wireshark):

- N.° segmenti IN/OUT

- Bytes IN/OUT
- N.° segmenti intranet

TCP vs UDP (R script 1):

- totale flussi tcp/udp
- totale segmenti/dati tcp
- totale segmenti /dati udp
- campionamento numero flussi tcp e udp (ogni secondo)
- campionamento segmenti tcp/udp (ogni secondo)
- campionamento dati tcp/udp (ogni secondo)

Lost Analysis (da wireshark):

- andamento ack duplicati, (tcp.analysis.duplicate_ack, Packets)
- andamento lost OUT/IN

Andamento RTT nel tempo (da wireshark):

- min, max, media

Distribuzione durata dei flussi (R script 3):

- TCP vs UDP

Distribuzione trasmissioni flussi (R script 4):

- TCP vs UDP

Gerarchia protocolli (wireshark):

- schema gerarchia protocolli

6. SOFTWARE SVILUPPATO

Questa sezione illustra i passi necessari per il download, la configurazione e l'uso dei vari script sviluppati per il presente elaborato da applicare ad un file in formato PCAP che consentano di dare una risposta alle metriche stabilite nel precedente paragrafo.

Git

L'intero progetto e la documentazione è scaricabile da Git:

```
git clone https://github.com/DanieleCampagnoli/TesinaTIR
```

Struttura del software e contenuto dei file

Il software è contenuto in una cartella chiamata TesinaTIR la quale contiene due sottocartelle:

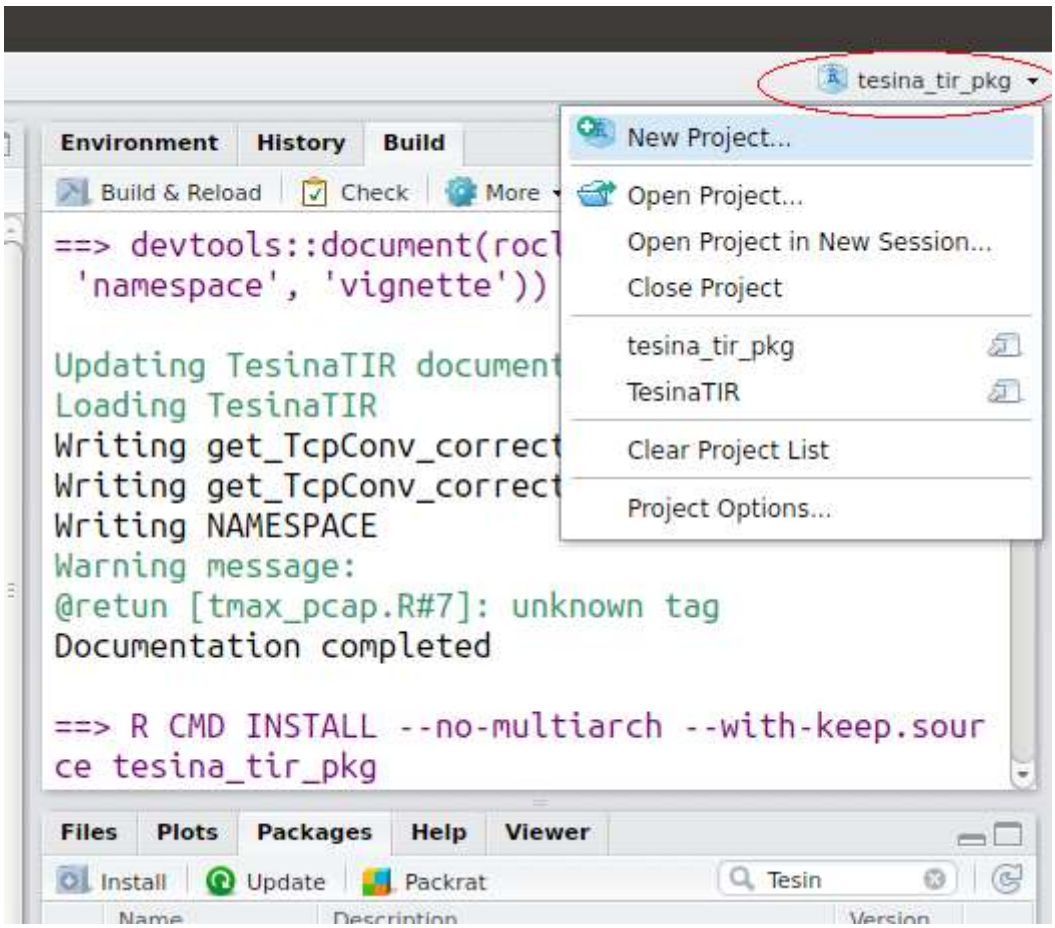
- scripts: contiene i vari script che partendo da un file PCAP estraggono metriche sopra indicate producendo i relativi grafici;
  Per lanciare un qualsiasi script presente in tale cartella è sufficiente sostituire la x con il numero di script desiderato digitando il seguente comando:

  ```
  Rscript script_x.R -h
  ```

  Al suo interno è inoltre presente una sottocartella pcap per contenere i flussi acquisiti e da elaborare;
- tesina_tir_pkg: package di R che contiene le funzioni utilizzate da dagli script inclusi nella cartella precedente ed è strutturata come segue:
  - R: contiene il codice sorgente delle funzioni
  - man: contiene il manuale in formato .Rd
  - inst/extdata: contiene un pcap di esempio associato al package
  - altri file necessari alla gestione del progetto

Installazione TesinaTIR

Di seguito verranno elencate le operazioni per utilizzare il software sviluppato. I vari sottoparagrafi devono essere eseguiti in sequenza e la procedura di installazione è stata testata solo su ubuntu.
Importare il package in rstudio cliccando su “Open Project”



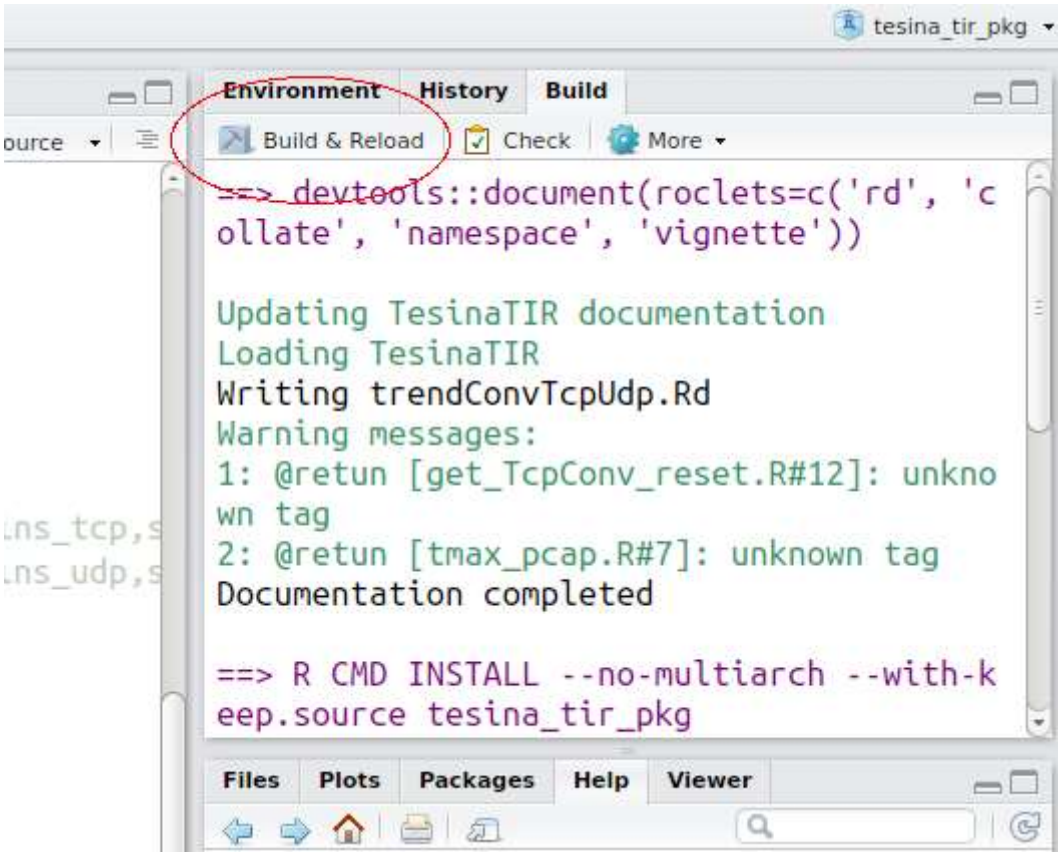
(fig. 4 – Open Project)

Impostare le seguenti opzioni:

```
Menù Build -> Configure Build Tools -> Build Tools
  -> (baffetto su Generate Documentation with Roxygen)
```

```
Menù Build -> Configure Build Tools -> Build Tools->Configure
  -> (baffetto su tutte le opzioni)
```

cliccare su “Build & Reload” per installare e caricare la libreria in r



(fig. 5 – build & reload)

Lanciare I seguenti comandi per testare l'installazione del package

```
library(TesinaTIR)
pcapName<-system.file("extdata", "dump.pcap", package = "TesinaTIR")
tmax<-tmaxPcap(pcapName)
```

Documentazione

La documentazione del software sviluppato è suddivisa in due parti:

- documentazione degli script
- documentazione del package tesina_tir_pkg

La documentazione degli script è ottenibile lanciando lo script da terminale con il seguente comando, dove x indica il numero dello script.

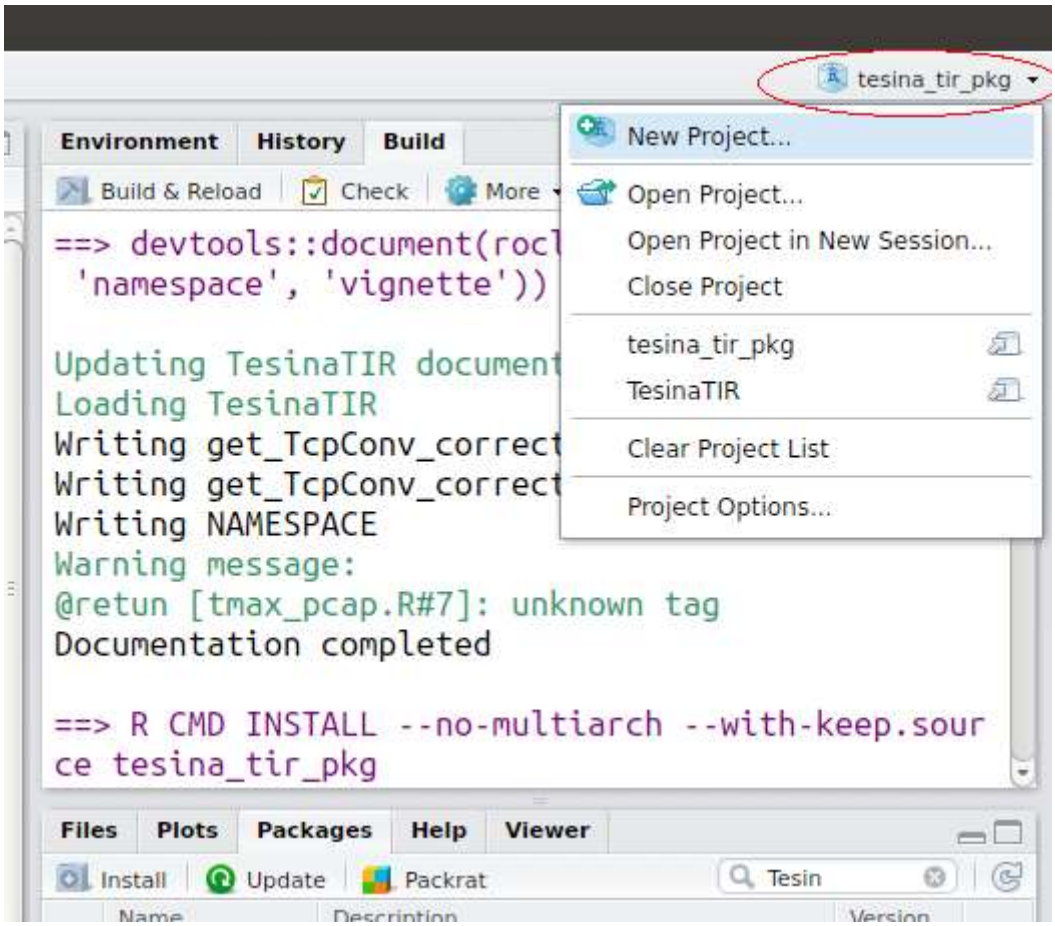
```
Rscript script_x.R -h
```

Di seguito è descritto un esempio di esecuzione di uno script.

```
Rscript script_1.R -i ./pcap/GazzettaDiModena_20161120.pcap -o ./out_x
```

La documentazione del package TesinaTIR è ottenibile nel seguente modo:

- importare il package in rstudio cliccando su “Open Project”



(fig. 6 – import package)

- ricercare il package nel menu packages inserendo nella barra la stringa “TesinaTIR” e cliccare nei risultati su TesinaTIR.

(fig. 7 – ricerca package)

8. ELABORAZIONE TRAFFICO E RISULTATI

Stabilite le metriche, l'hardware necessario e la selezione dei soggetti partecipanti al test, si è proceduto all'acquisizione del traffico che volutamente è avvenuto su due fasce orarie distinte perché si è voluto eseguire l'analisi su due tipologie di utenti: la prima riguardava dei minori (primo pomeriggio), la seconda degli adulti (tardo pomeriggio).
L'estrazione dei dati è avvenuta mediante la generazione di grafici del software wireshark (Statistiche -> Grafici I/O), impostando opportunamente il filtro ed il campo da rappresentare sull'asse delle Y (di seguito: filtro, asse Y, campo Y)

Primo pomeriggio

Dati generali dell'acquisizione (da wireshark):

- Ora inizio/Ora fine, tipologia di utenza sottoposta al test, tipologia di traffico generato, totale pacchetti persi, eventuali anomalie rilevate:
  Il traffico è stato acquisito dalla navigazione Internet mediante un tablet, perlopiù su un sito di streaming video, e da un notebook collegato via RDP ad un pc esterno alla rete;
  Da Wireshark si constata che l'acquisizione è stata avviata alle 13.55 del 8/12/2016 alle 16.15 del 8/12/2016 (2h 20m), sono transitati 418.215 pacchetti per un totale di 249.893.822 byte; I pacchetti persi ammontano al 4% (18.598 pacchetti). Nell'ultima mezz'ora gli utenti hanno rilevato errori di risoluzione DNS.

Traffico IN/OUT, traffico intranet (da wireshark):

- N.° segmenti IN/OUT (fig. 8a/8b), (ip.src != 192.168.0.0/24, Packets e ip.dst != 192.168.0.0/24, Packets)
- Bytes IN/OUT (fig. 9a/9b), (ip.src != 192.168.0.0/24, Bytes e ip.dst != 192.168.0.0/24, Bytes)
- N.° segmenti intranet (fig. 10) (ip.src == 192.168.0.0/24 && ip.dst == 192.168.0.0/24, Packets)

I grafici dei segmenti e dei dati IN/OUT evidenziano che in generale, seppur con impegno di banda di un ordine di grandezza superiore fra IN e OUT, a traffico in uscita corrisponde traffico in entrata, a momenti di assenza di traffico in uscita corrispondono momenti di assenza di traffico in entrata che come studiato nella teoria del protocollo TCP ci aspettavamo.
La spiegazione del perché il traffico in uscita sia inferiore di un ordine grandezza rispetto al traffico in entrata è dovuto proprio al fatto che la tipologia del traffico domestico è sostanzialmente in download per la fruizione di pagine web o video in streaming limitando il traffico in uscita alle richieste dei contenuti e ai relativi ACK.
Questa analisi globale del traffico IN/OUT ci indica inoltre che il livello medio di utilizzo della banda in download (IN) è nettamente inferiore a quanto disponibile come da specifiche del provider essendo di 7Mbit/s (917.504 byte/s), anche il livello medio di utilizzo della banda in upload è nettamente inferiore a quanto disponibile e dichiarata a 200Kbit/s (25.600 byte/s). Si evidenzia che alcuni picchi in uscita occupano totalmente la banda.
Il grafico del traffico intranet ci dice che c'è un costante "rumore di fondo" dovuto al continuo invio di pacchetti NBNS (NetBIOS).

TCP vs UDP (script 1):

- totale flussi tcp/udp: 462/882 (34% / 66%)
- totale segmenti/dati tcp 385.851/224.403.823 (92% / 89%)
- totale segmenti/dati udp 31.138/25.426.187 (8% / 11%)
- campionamento numero flussi tcp e udp (ogni secondo, fig. 11)
- campionamento numero segmenti tcp/udp (ogni secondo, figg. 12a e 12b) (tcp e udp, Packets)

- campionamento dati tcp/udp (ogni secondo, figg. 13a e 13b) (tcp e udp, Bytes)

Il traffico UDP è per la maggior parte dovuto a richieste DNS.

Lost Analysis:

- andamento ack duplicati (fig. 14) (tcp.analysis.duplicate_ack, Packets)
- andamento lost OUT/IN (fig. 15a e 15b)
  (tcp.analysis.lost_segment && ip.src == 192.168.0.0/24, COUNT FIELDS(Y), tcp.analysis.lost_segment)
  (tcp.analysis.lost_segment && ip.src != 192.168.0.0/24, COUNT FIELDS(Y), tcp.analysis.lost_segment)

La maggior parte dei pacchetti sono stati persi in ricezione (il fondo scala è 14 pacchetti), e ciò è avvenuto nella parte finale dell'acquisizione proprio quanto sono state rilevate alcune anomalie dagli stessi utenti.

Andamento RTT nel tempo (script 2):

- min, max, media (fig. 16), (tcp, MIN/MAX/AVG(Y), tcp.analysis.ack_rtt)

La rilevazione dei RTT, paragonata ad un ping verso 8.8.8.8 di 56 ms (il triplo rispetto ad una linea aziendale giustificato evidentemente dalla tipologia di linea residenziale), in assenza di un'analisi approfondita delle cause, indica solo che se tolleriamo un RTT medio di 150 ms, tutto sommato la prestazione generale rilevata è accettabile, considerando ovviamente che in momenti critici (es. la parte finale dell'acquisizione) i valori RTT diventano esagerati.

Distribuzione durata dei flussi (R script 3):

- TCP vs UDP (figg. 17a e 17b)

Distribuzione trasmissioni flussi (R script 4):

- TCP vs UDP (figg. 18a e 18b)

La transmission_normDuration_tcp è una normalizzazione della durata del flusso e evidenzia che la probabilità che un flusso trasmetta è proprio concentrata nella parte inziale e finale del flusso.
Inoltre si rileva che i flussi di breve durata sono molto più probabili, e quindi più numerosi, di quelli di lunga durata quindi la distribuzione senza questa normalizzazione viene sbilanciata verso gli istanti iniziali del flusso.

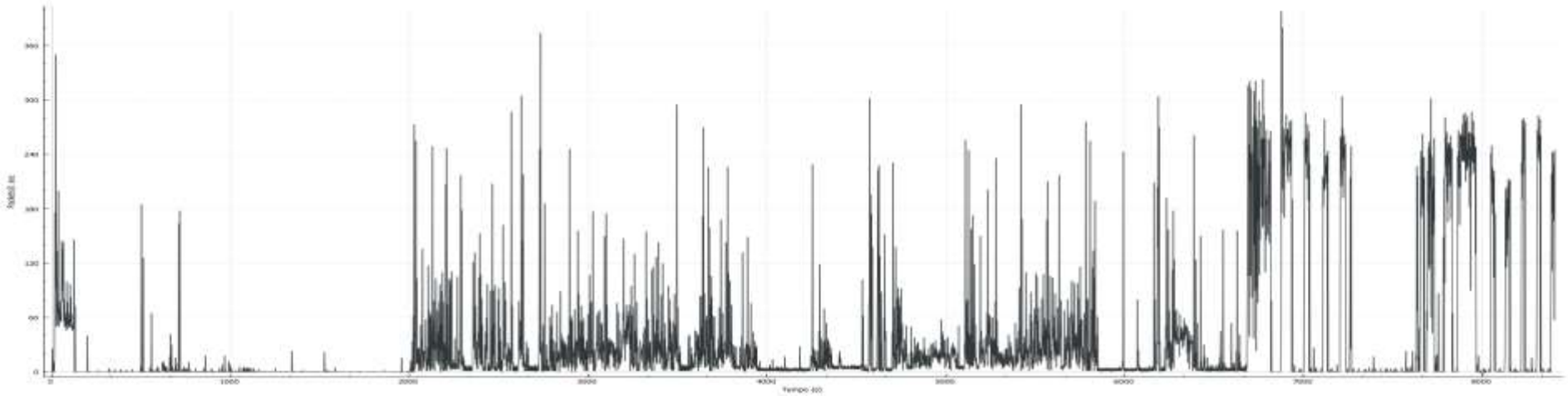
Procedimento della normalizzazione:
1) creiamo n intervalli chiamati parti
2) per ogni flussi andiamo a suddividere la sua durata in n parti uguali
3) calcoliamo la probabilità che un flusso trasmetta all'interno della parte

In questo modo si nota che la prima e l'ultima parte ha probabilità 1, questo è realistico perchè nella prima parte abbiamo il three way handshake e nell'ultima abbiamo il four way handshake con i fin oppure un reset.

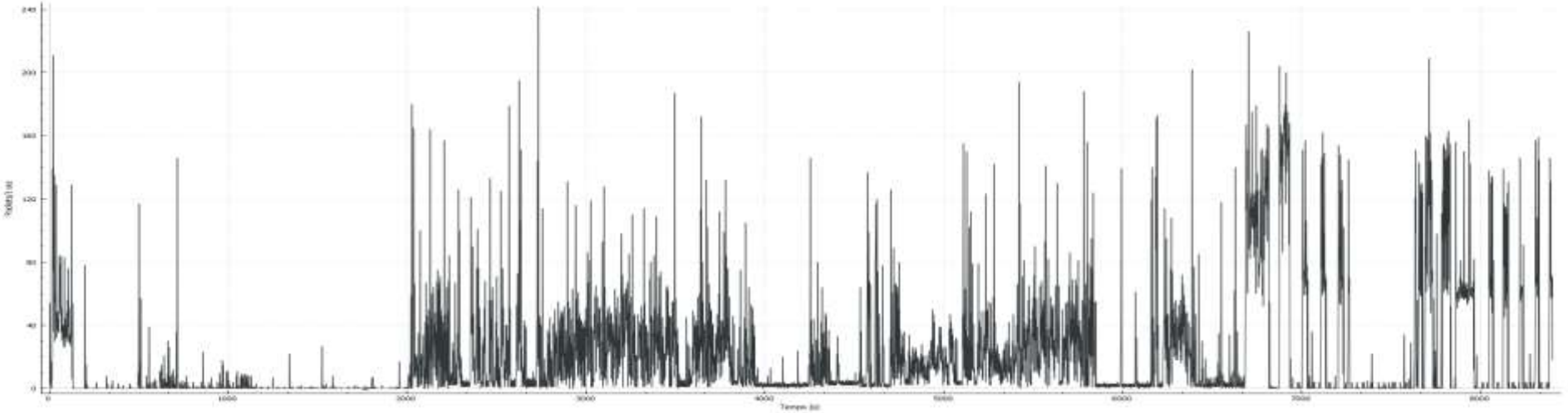
Gerarchia protocolli (wireshark):

- Gerarchia protocolli (fig. 19), (Statistiche -> Gerarchie dei protocolli)

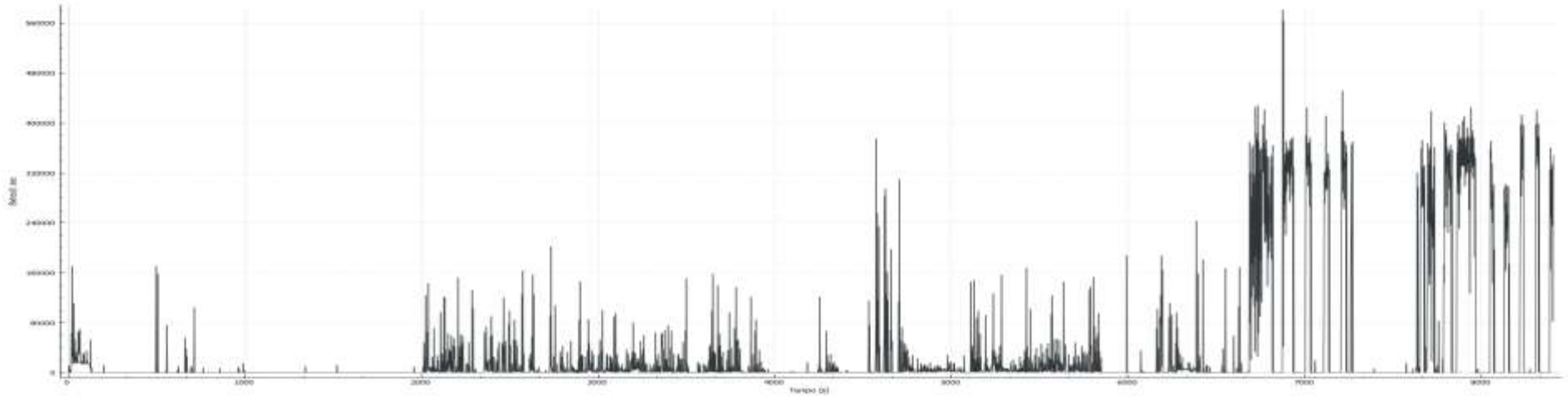
Infine il grafico della gerarchia dei protocolli conferma quello che immaginavamo rispetto all'utilizzo della rete domestica e cioè che la maggior parte di pacchetti sono tcp per la fruizione di contenuti multimediali e/o streaming video (SSL), escludendo l'anomalia del traffico "Data" dovuto all'RDP, mentre l'udp è caratterizzato da richieste DNS.



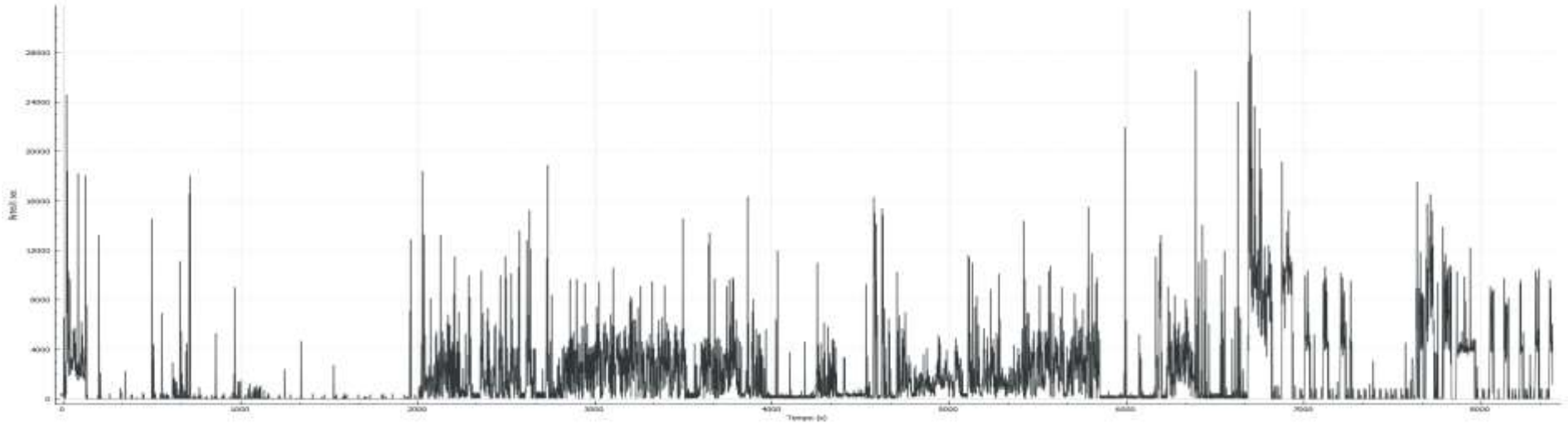
(fig. 8a – segmenti IN pomeriggio - 360)



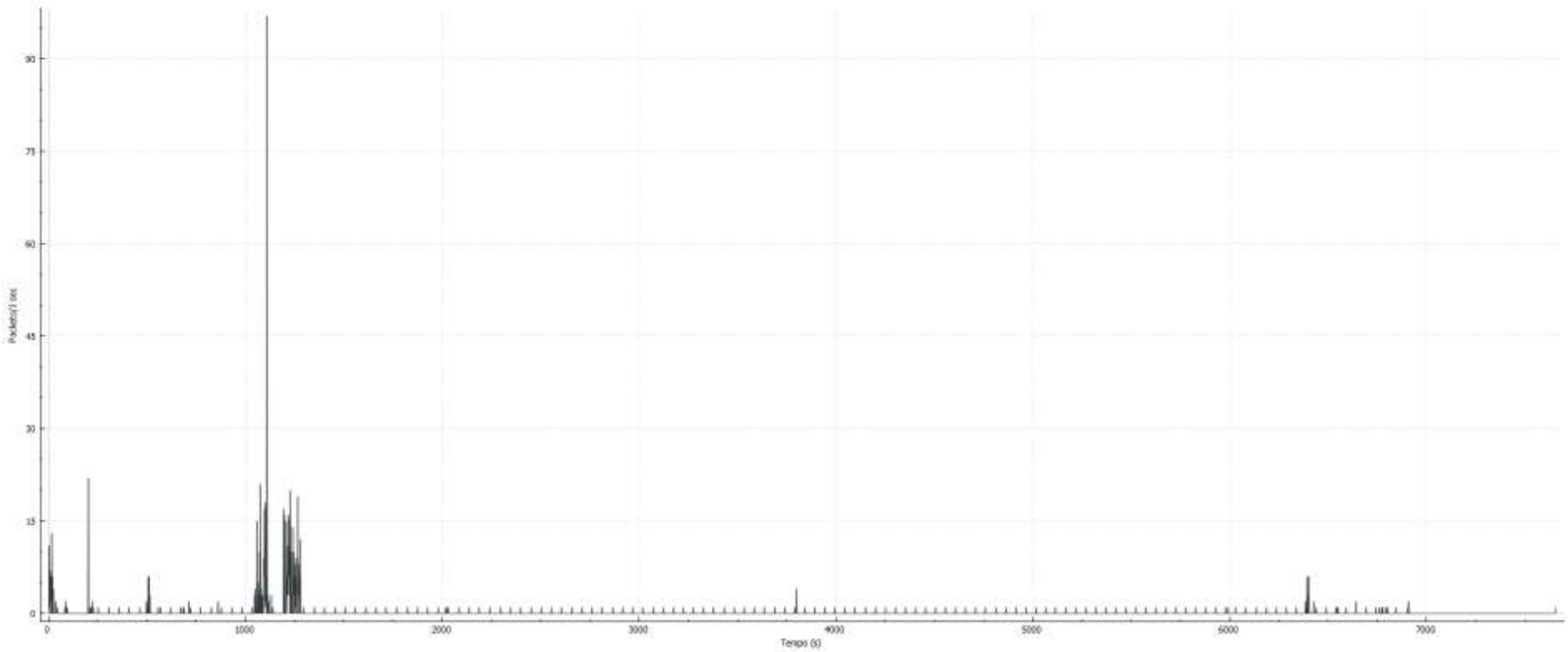
(fig. 8b – segmenti OUT pomeriggio - 240)



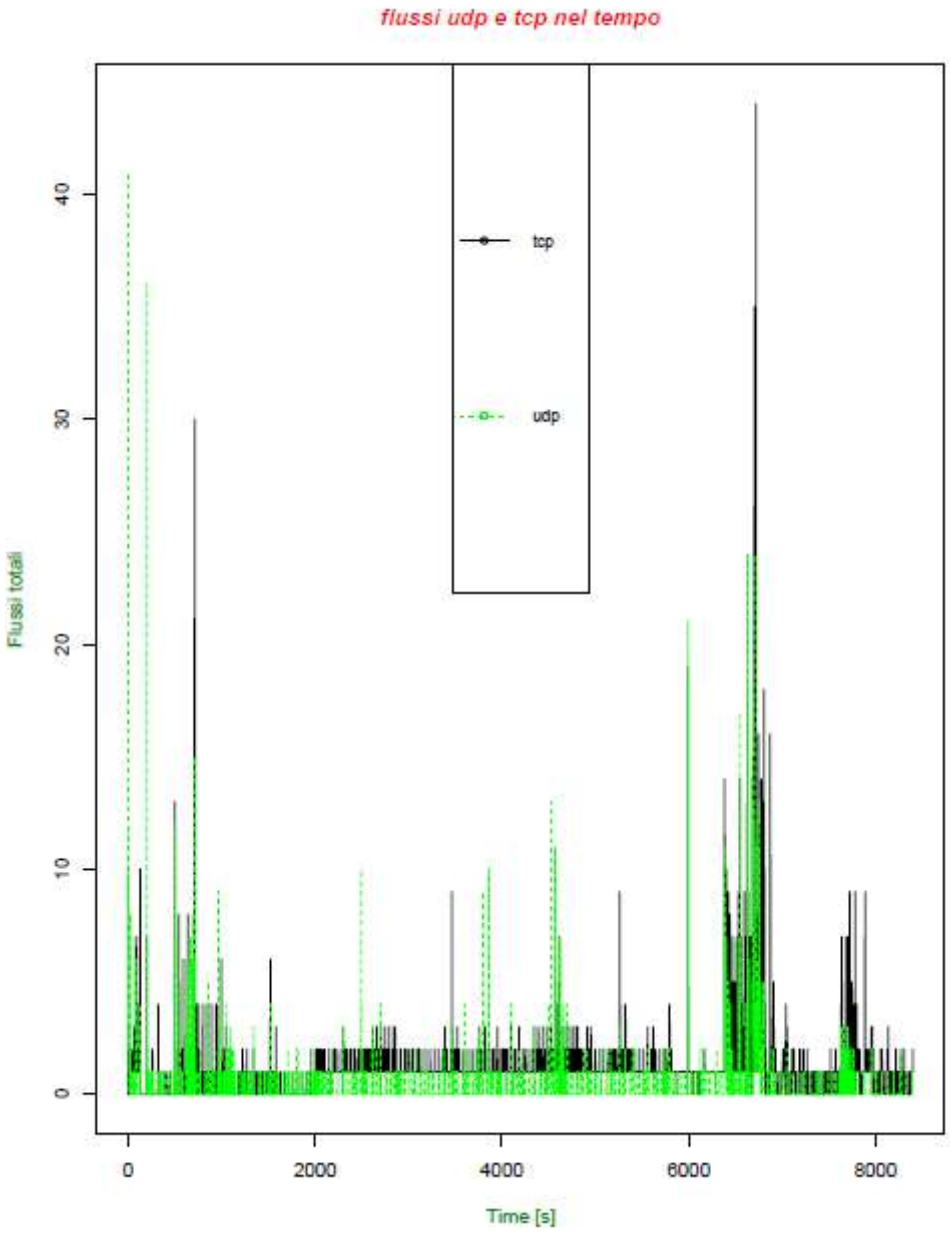
(fig. 9a – bytes IN pomeriggio – 560.000)



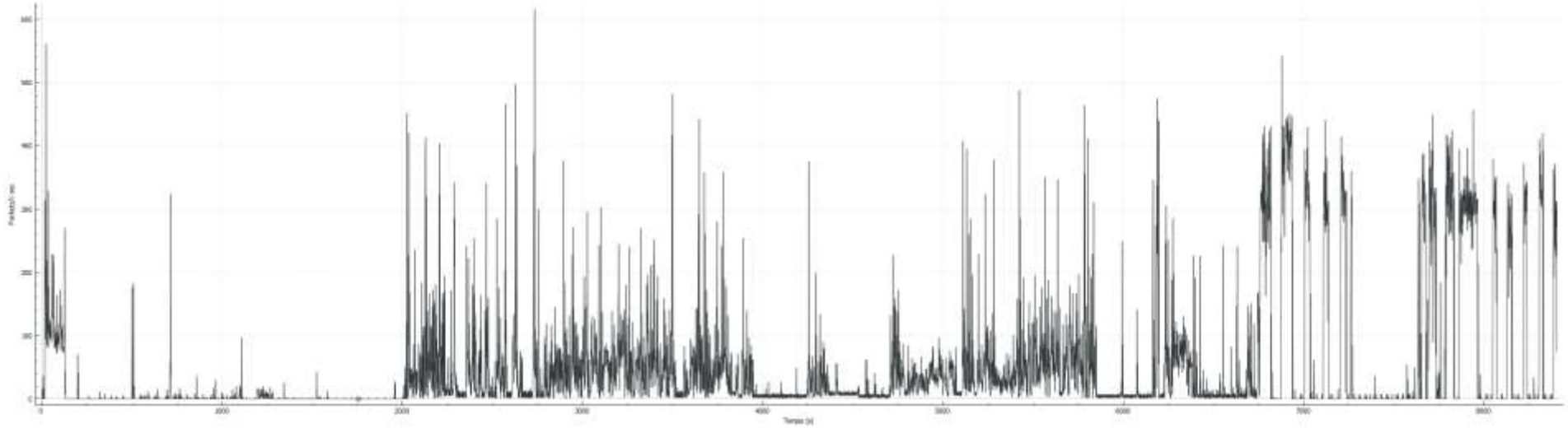
(fig. 8b – bytes OUT pomeriggio – 28.000)



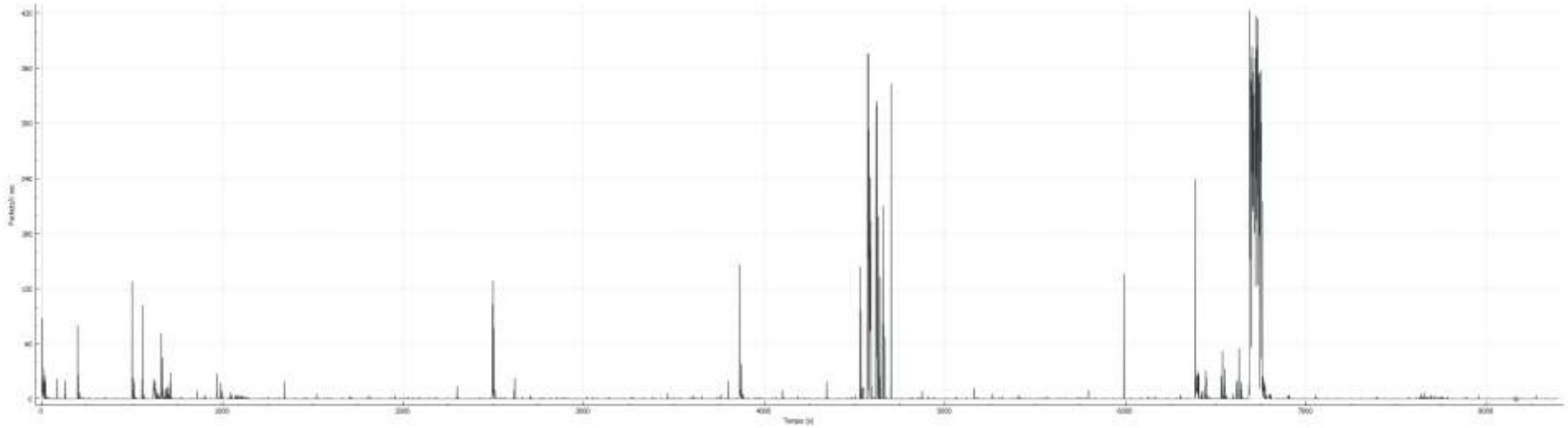
(fig. 10 – pacchetti traffico intranet - 90)



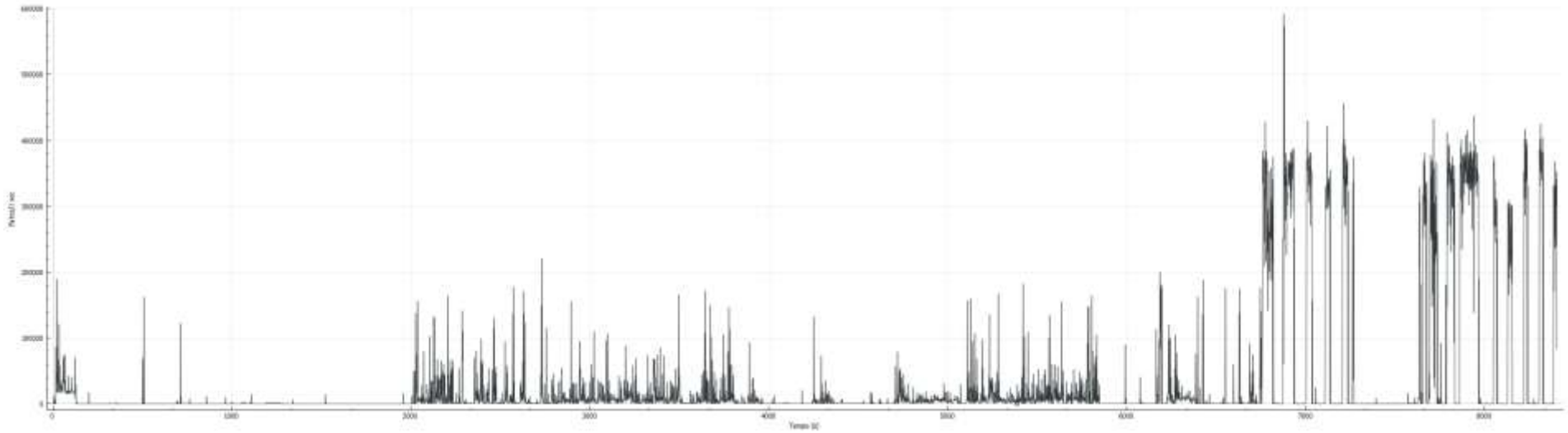
(fig. 11 – numero flussi tcp/udp pomeriggio)



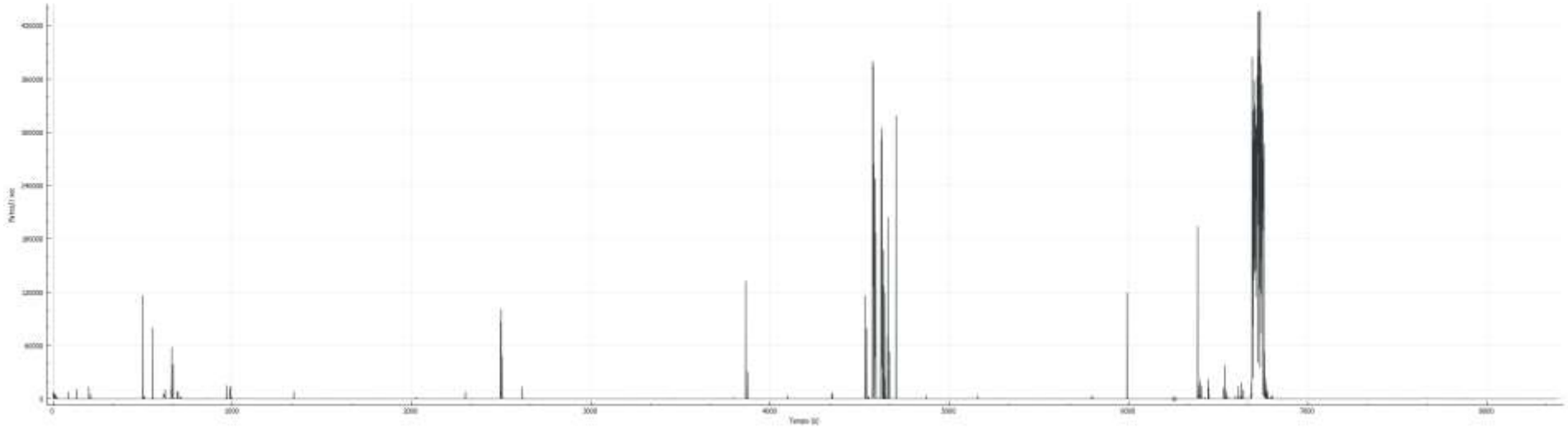
(fig. 12a – numero segmenti TCP pomeriggio - 600)



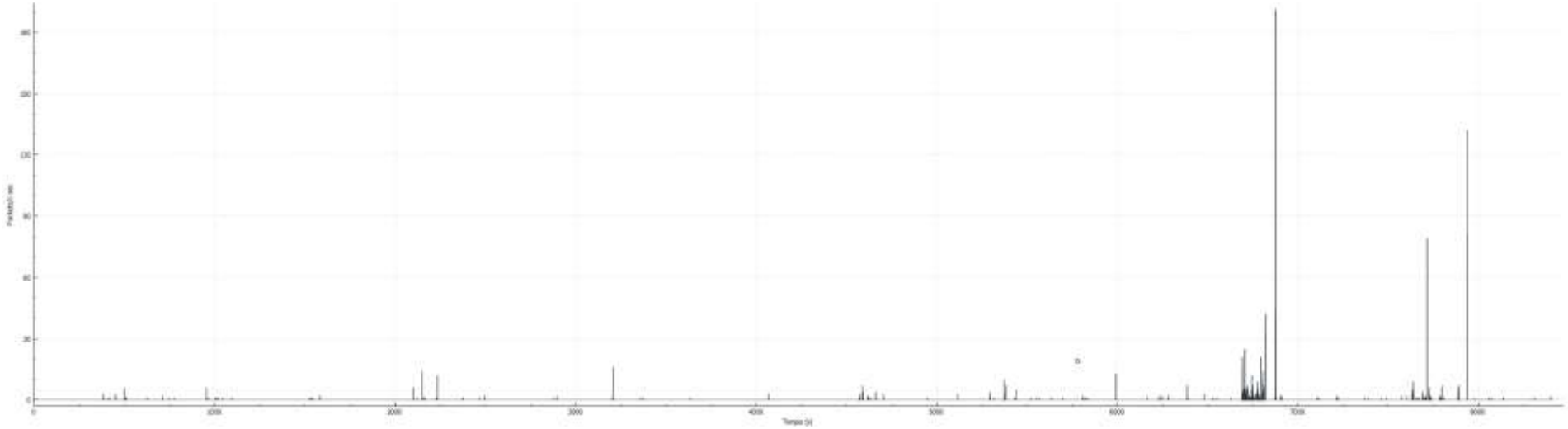
(fig. 12b – numero segmenti UDP pomeriggio - 420)



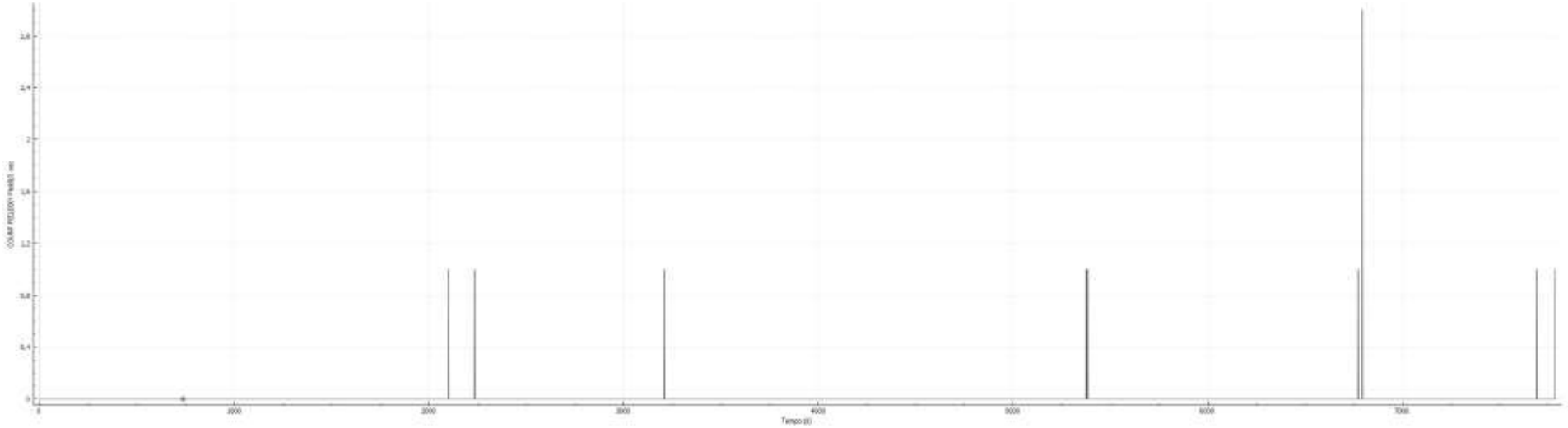
(fig. 13a – bytes TCP pomeriggio 600.000)



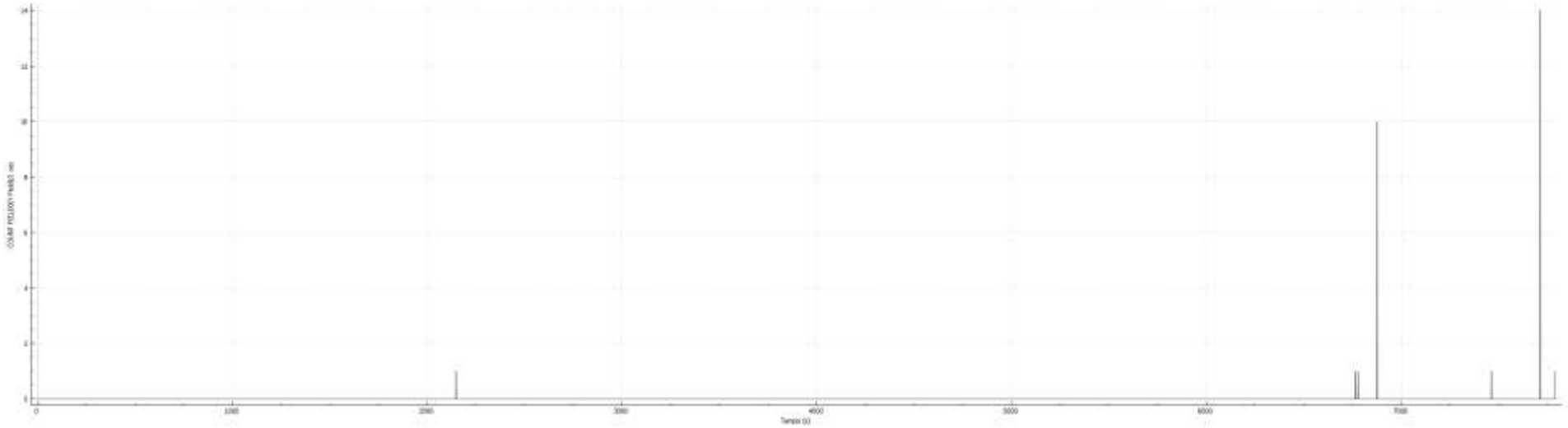
(fig. 13b – bytes UDP pomeriggio – 420.000)



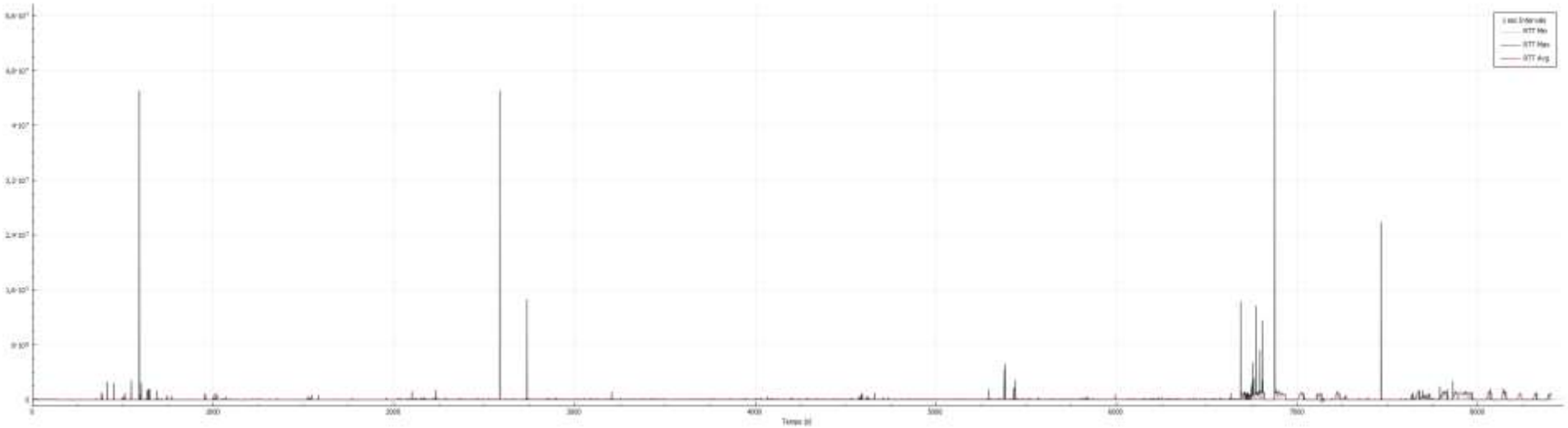
(fig. 14 – duplicate ACK pomeriggio - 180)



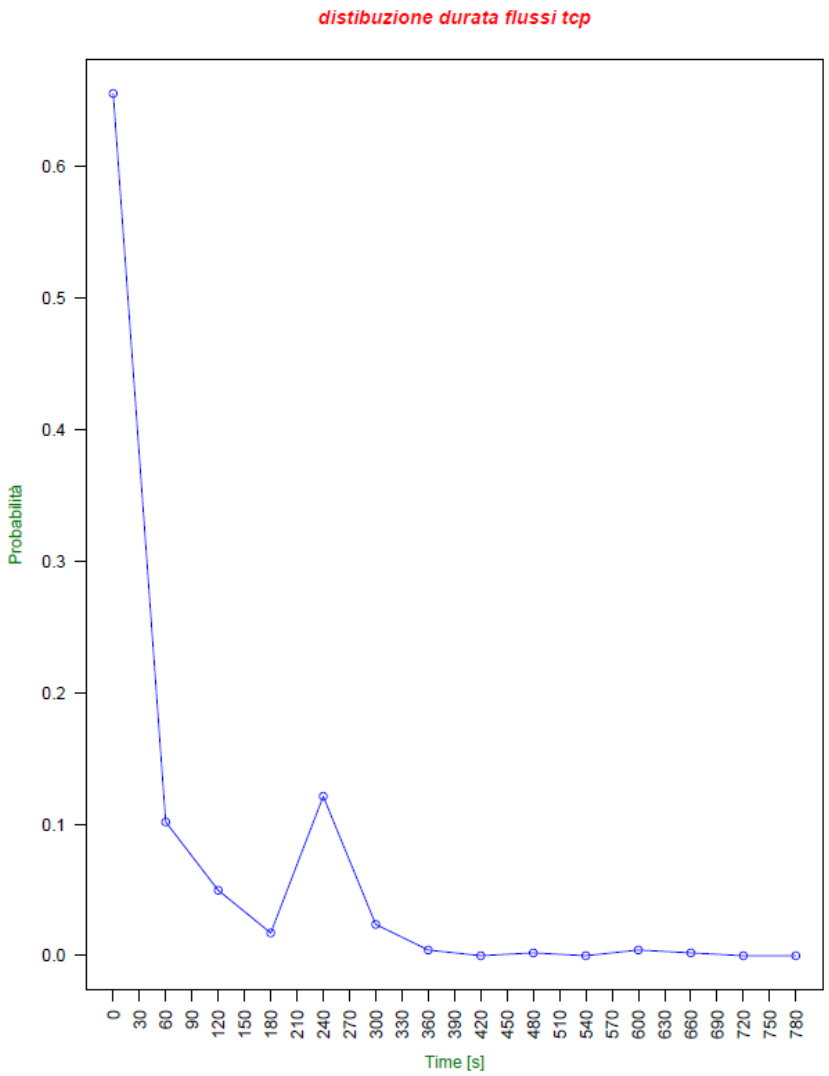
(fig. 15a – lost OUT pomeriggio – 2,8)



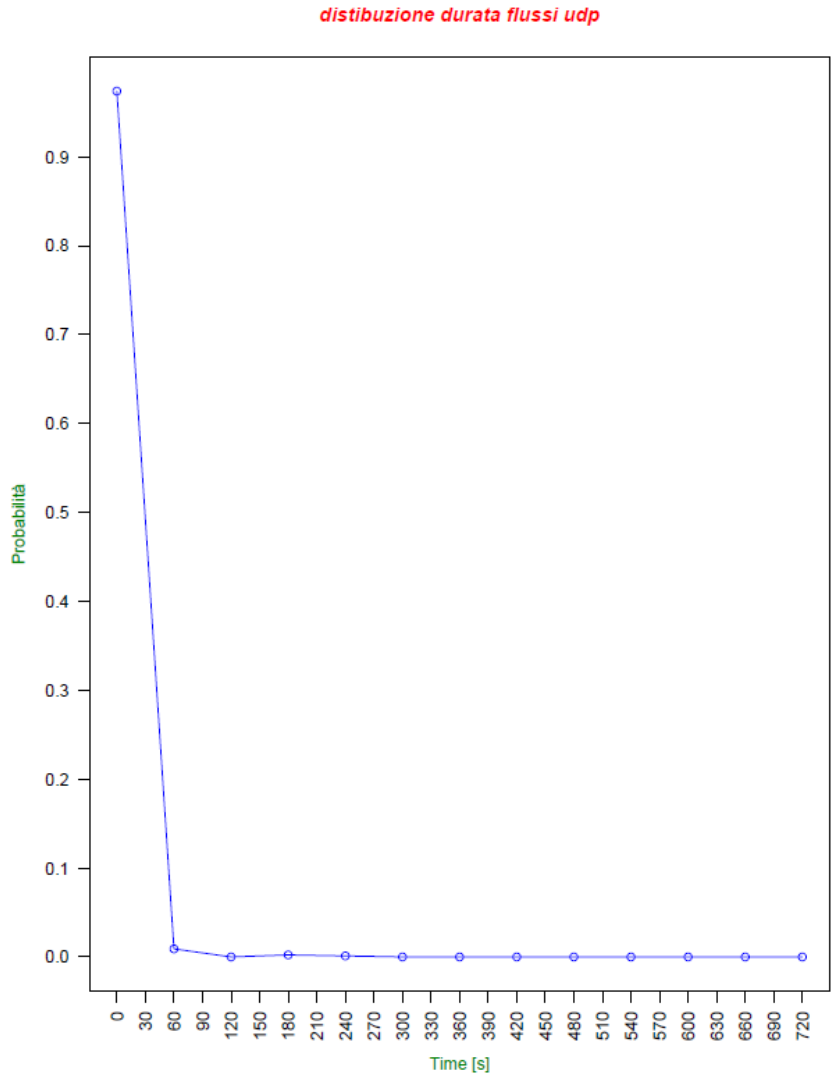
(fig. 15b – lost IN pomeriggio - 14)



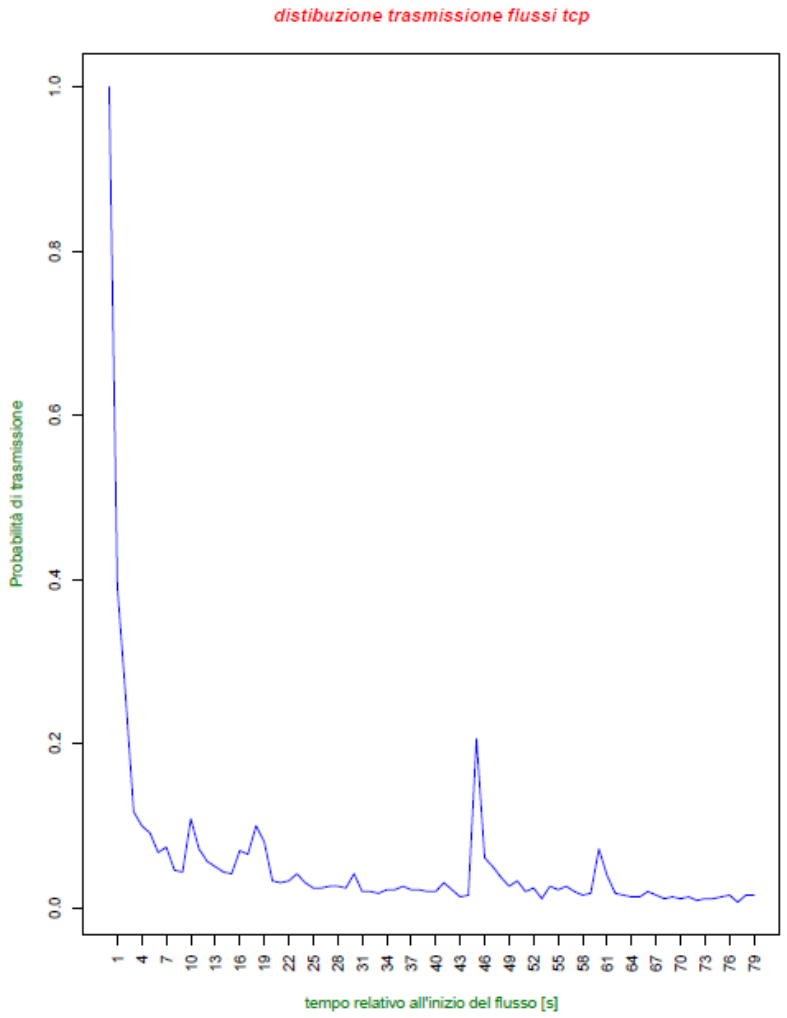
(fig. 16 – RTT Min, RTT Max, RTT Avg pomeriggio – 5,6 10^7)



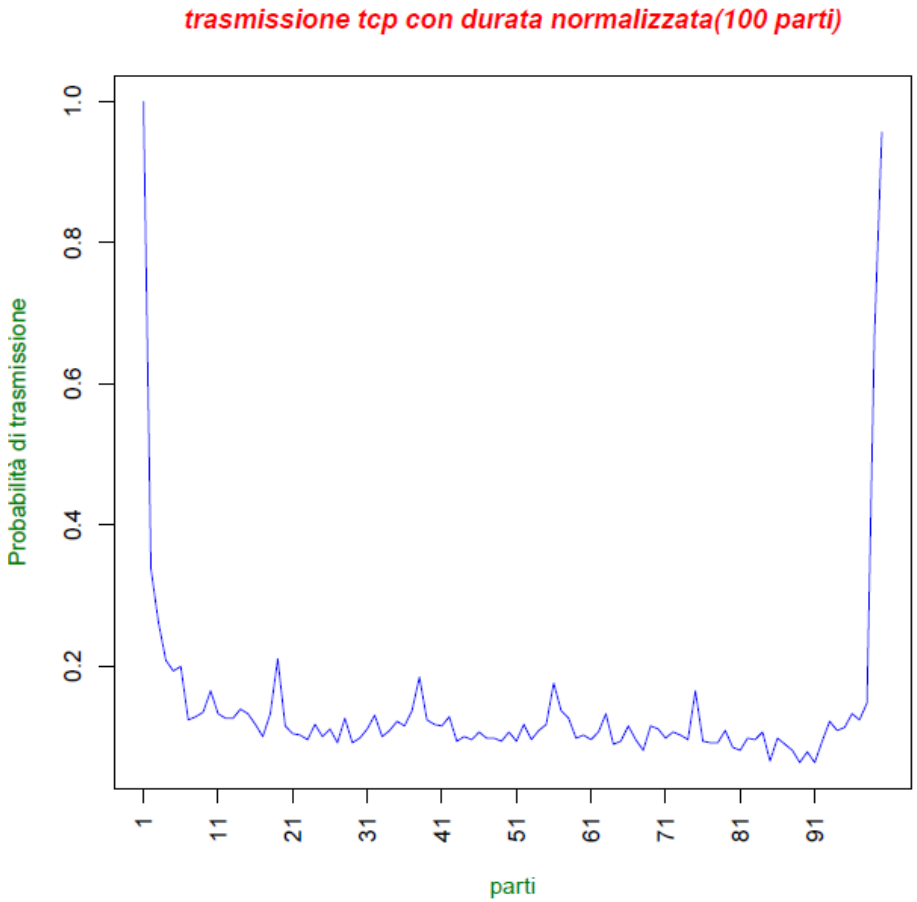
(fig. 17a - distribuzione durata flusso tcp pomeriggio)



(fig. 17b – distribuzione durata flusso udp pomeriggio)



(fig. 18a - distribuzione trasmissioni flusso tcp pomeriggio)



(fig. 18b – distribuzione trasmissioni flusso tcp normalizzata pomeriggio)

Gerarchia protocolli

Wireshark · Statistiche gerarchia di protocolli · 20161208_pomeriggio

Protocollo	Percentuale pacchetti	Pacchetti	Percentuale byte	Byte	Bit/s	Pacchetti finali	Byte finali	Bit/s finali
Frame	100.0	418215	100.0	249893822	237 k	0	0	0
Ethernet	100.0	418215	2.3	5855010	5573	0	0	0
Internet Protocol Version 6	0.1	376	0.0	43529	41	0	0	0
User Datagram Protocol	0.1	264	0.0	2112	2	0	0	0
Multicast Domain Name System	0.0	38	0.0	1879	1	38	1879	1
Link-local Multicast Name Resolution	0.0	196	0.0	5224	4	196	5224	4
DHCPv6	0.0	14	0.0	1330	1	14	1330	1
Data	0.0	16	0.0	14000	13	16	14000	13
Internet Control Message Protocol v6	0.0	112	0.0	3160	3	112	3160	3
Internet Protocol Version 4	99.7	416910	3.3	8338572	7937	0	0	0
User Datagram Protocol	7.4	30850	0.1	246800	234	0	0	0
Simple Service Discovery Protocol	0.1	211	0.0	36503	34	211	36503	34
QUIC (Quick UDP Internet Connections)	6.9	28931	9.6	23907947	22 k	28931	23907947	22 k
Network Time Protocol	0.0	12	0.0	576	0	12	576	0
NetBIOS Name Service	0.1	249	0.0	12882	12	249	12882	12
Multicast Domain Name System	0.0	35	0.0	1663	1	35	1663	1
Link-local Multicast Name Resolution	0.0	182	0.0	4888	4	182	4888	4
Domain Name System	0.2	934	0.0	55136	52	934	55136	52
Data	0.0	151	0.0	22280	21	151	22280	21
Bootstrap Protocol	0.0	145	0.0	47671	45	145	47671	45
Transmission Control Protocol	92.3	385851	84.5	211284889	201 k	224237	146960061	139 k
SSH Protocol	0.1	444	0.0	42733	40	443	42721	40
Secure Sockets Layer	5.6	23422	63.9	159596930	151 k	23032	153489270	146 k
Malformed Packet	0.0	46	0.0	0	0	46	0	0
Hypertext Transfer Protocol	0.1	315	0.4	960358	914	170	81841	77
Portable Network Graphics	0.0	29	0.0	64779	61	29	69207	65
Media Type	0.0	30	0.2	505160	480	30	284395	270
Line-based text data	0.0	43	0.3	751217	715	43	303929	289
JPEG File Interchange Format	0.0	18	0.1	151378	144	18	160625	152
eXtensible Markup Language	0.0	22	0.0	115114	109	22	47134	44
Compuserve GIF	0.0	3	0.0	1318	1	3	1708	1
Data	32.9	137778	13.6	33888568	32 k	137778	33888568	32 k
Internet Group Management Protocol	0.0	93	0.0	1400	1	93	1400	1
Internet Control Message Protocol	0.0	116	0.0	8028	7	116	8028	7
Address Resolution Protocol	0.2	929	0.0	26012	24	929	26012	24

Nessun filtro di visualizzazione.

Chiudi | Copia | Aiuto

(fig. 19 – gerarchie protocolli pomeriggio)

Tardo pomeriggio

Dati generali dell'acquisizione (da wireshark):

- Ora inizio/Ora fine, tipologia di utenza sottoposta al test, tipologia di traffico generato, totale pacchetti persi, eventuali anomalie rilevate:
  Il traffico è stato acquisito dalla navigazione Internet mediante un tablet, perlopiù su siti comuni, e da un notebook collegato via RDP ad un pc esterno alla rete e da un cellulare;
  Da Wireshark si constata che l'acquisizione è stata avviata alle 17.16 del 8/12/2016 alle 23.08 del 8/12/2016 (5h 52m), sono transitati 565.792 pacchetti per un totale di 245.770.159 byte; Non sono stati rilevati pacchetti persi.

Traffico IN/OUT, traffico intranet (da wireshark):

- N.° segmenti IN/OUT (fig. 20a/20b) , (ip.src != 192.168.0.0/24, Packets e ip.dst != 192.168.0.0/24, Packets)
- Bytes IN/OUT (fig. 21a/21b), (ip.src != 192.168.0.0/24, Bytes e ip.dst != 192.168.0.0/24, Bytes)
- N.° segmenti intranet (fig. 22), (ip.src == 192.168.0.0/24 && ip.dst == 192.168.0.0/24, Packets)

Anche in questo caso i grafici dei segmenti e dei dati IN/OUT evidenziano che in generale, l'impegno di banda di un ordine di grandezza superiore fra IN e OUT, a traffico in uscita corrisponde traffico in entrata, a momenti di assenza di traffico in uscita corrispondono momenti di assenza di traffico in entrata che come studiato nella teoria del protocollo TCP ci aspettavamo.
La spiegazione del perché il traffico in uscita sia inferiore di un ordine grandezza rispetto al traffico in entrata è dovuto proprio al fatto che la tipologia del traffico domestico è sostanzialmente in download per la fruizione di pagine web o video in streaming limitando il traffico in uscita alle richieste dei contenuti e ai relativi ACK.
Questa analisi globale del traffico IN/OUT ci indica inoltre che il livello medio di utilizzo della banda in download (IN) è in media inferiore a quanto disponibile come da specifiche del provider essendo di 7Mbit/s (917.504 byte/s), mentre il livello medio di utilizzo della banda in upload è al limite rispetto a quanto disponibile e dichiarata a 200Kbit/s (25.600 byte/s).
Il grafico del traffico intranet ci dice che c'è un costante "rumore di fondo" dovuto al continuo invio di pacchetti NBNS (NetBIOS).

TCP vs UDP (script 1):

- totale flussi tcp/udp: 1827/1676 (52% / 48%)
- totale segmenti/dati tcp 464.331/157.815.655 (82% / 64%)
- totale segmenti/dati udp 99.500/87.864.754 (18% / 36%)
- campionamento numero flussi tcp e udp (ogni secondo, fig. 23)
- campionamento numero segmenti tcp/udp (ogni secondo, figg. 24a e 24b)
- campionamento dati tcp/udp (ogni secondo, figg. 25a e 25b)

Il traffico UDP è per la maggior parte dovuto a richieste DNS.

Lost Analysis:

- andamento ack duplicati (fig. 26), (tcp.analysis.duplicate_ack, Packets)
- andamento lost OUT/IN (fig. 27a e 27b)
  (tcp.analysis.lost_segment && ip.src == 192.168.0.0/24, COUNT FIELDS(Y), tcp.analysis.lost_segment)
  (tcp.analysis.lost_segment && ip.src != 192.168.0.0/24, COUNT FIELDS(Y), tcp.analysis.lost_segment)

La maggior parte dei pacchetti sono stati persi in ricezione (il fondo scala è 42 pacchetti), quattro volte superiore alla rilevazione del pomeriggio, e ciò è avvenuto a macchia di leopardo all'interno di tutta la rilevazione.

Andamento RTT nel tempo (script 2):

- min, max, media (fig. 16), (tcp, MIN/MAX/AVG(Y), tcp.analysis.ack_rtt)

Stesse considerazioni come sull'acquisizione del pomeriggio, infatti la rilevazione dei RTT, paragonata ad un ping verso 8.8.8.8 di 56 ms (il triplo rispetto ad una linea aziendale giustificato evidentemente dalla tipologia di linea residenziale), in assenza di un'analisi approfondita delle cause, indica solo che se tolleriamo un RTT medio di 150 ms, tutto sommato la prestazione generale rilevata è accettabile, considerando ovviamente che in momenti critici (es. la parte finale dell'acquisizione) i valori RTT diventano esagerati.

Distribuzione durata dei flussi (R script 3):

- TCP vs UDP (figg. 17a e 17b)

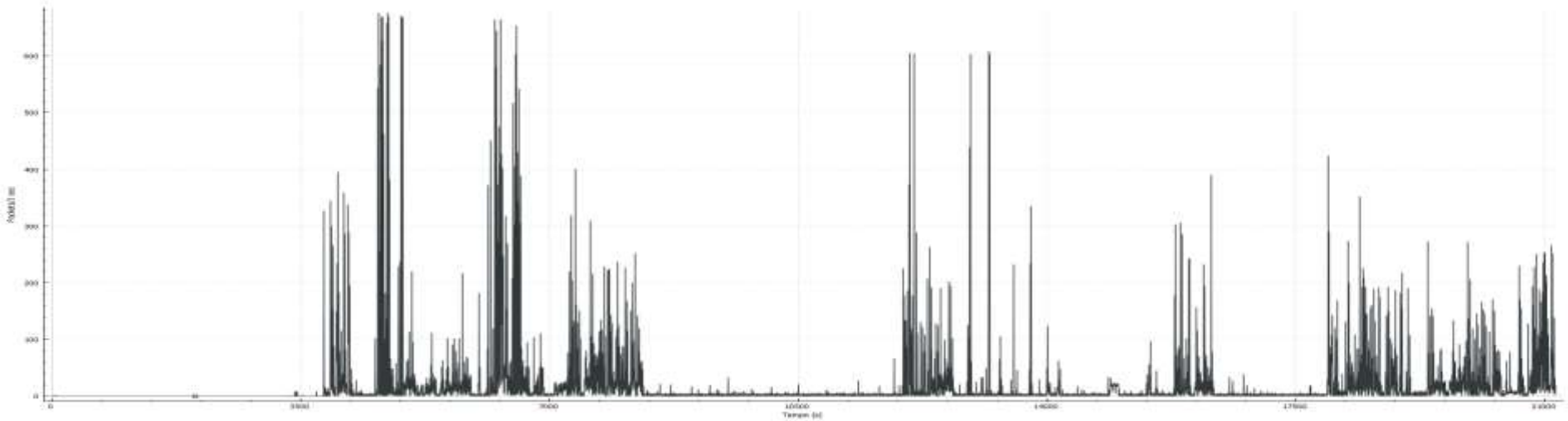
Distribuzione trasmissioni flussi (R script 4):

- TCP vs UDP (figg. 18a e 18b)

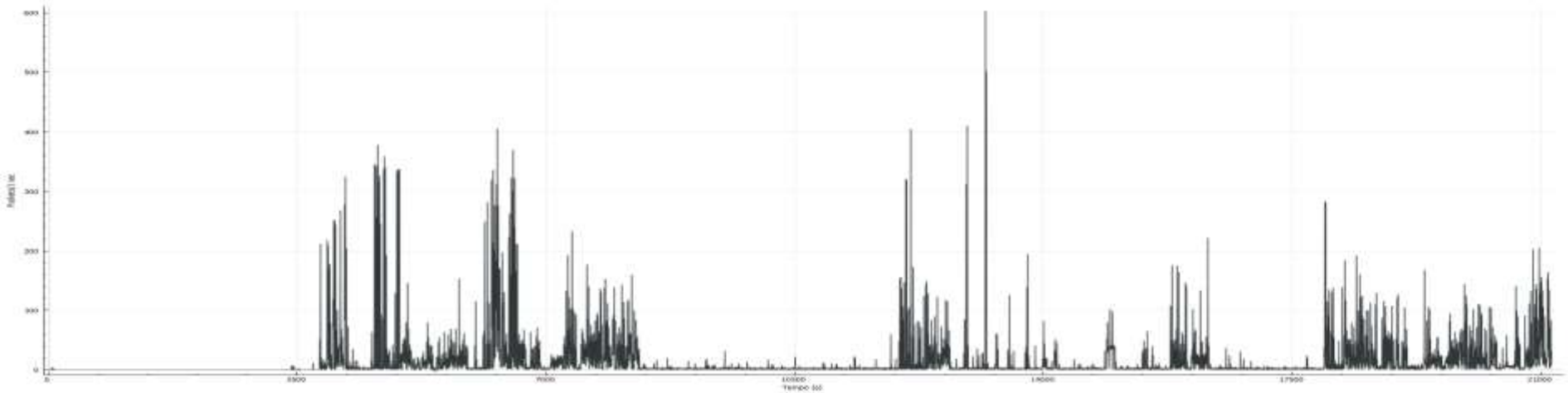
Gerarchia protocolli (wireshark):

- Gerarchia protocolli (fig. 31), (Statistiche -> Gerarchie dei protocolli)

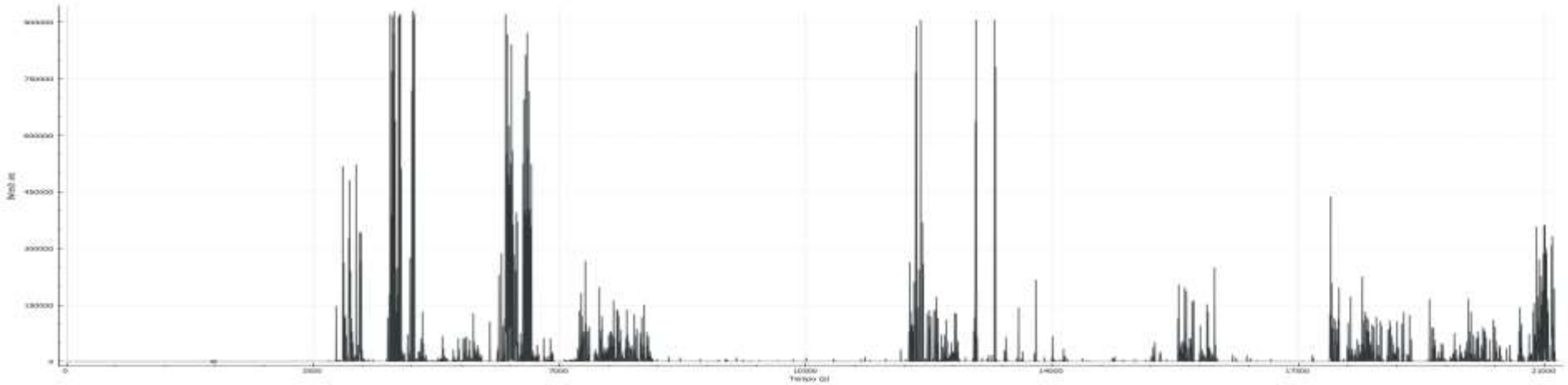
Le considerazioni delle ultime tre statistiche sono identiche a quelle fatte per il traffico del pomeriggio. Infine abbiamo preso un flusso a caso tra quelli con lost segment ed abbiamo visualizzato in un grafico l'andamento dei sequence number e della dimensione della finestra, trovando un'applicazione pratica di quanto studiato nella parte teorica (figg. 32a e 32b).



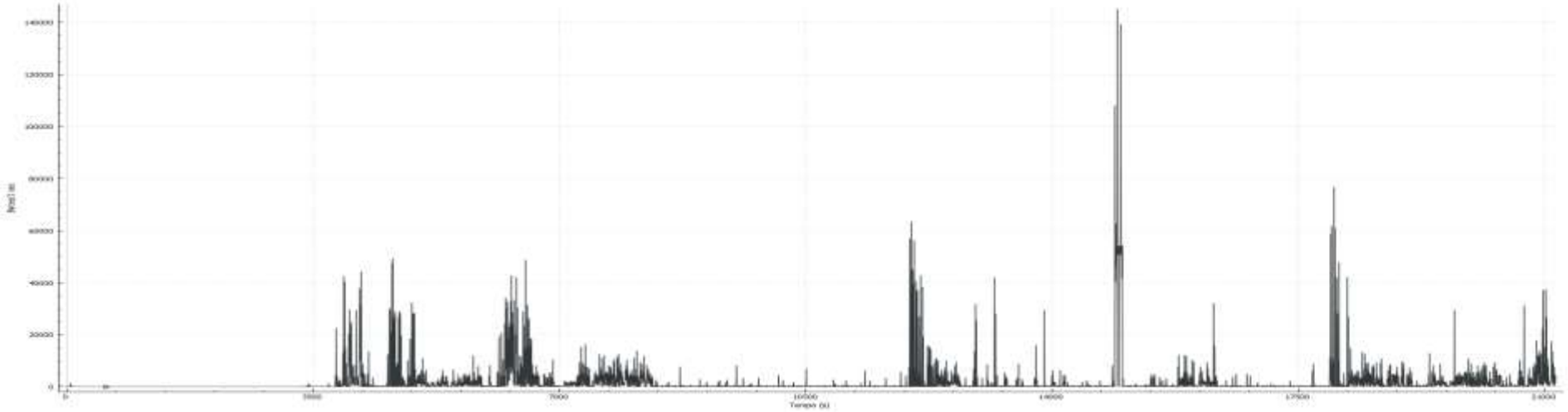
(fig. 20a – segmenti IN tardo pomeriggio - 700)



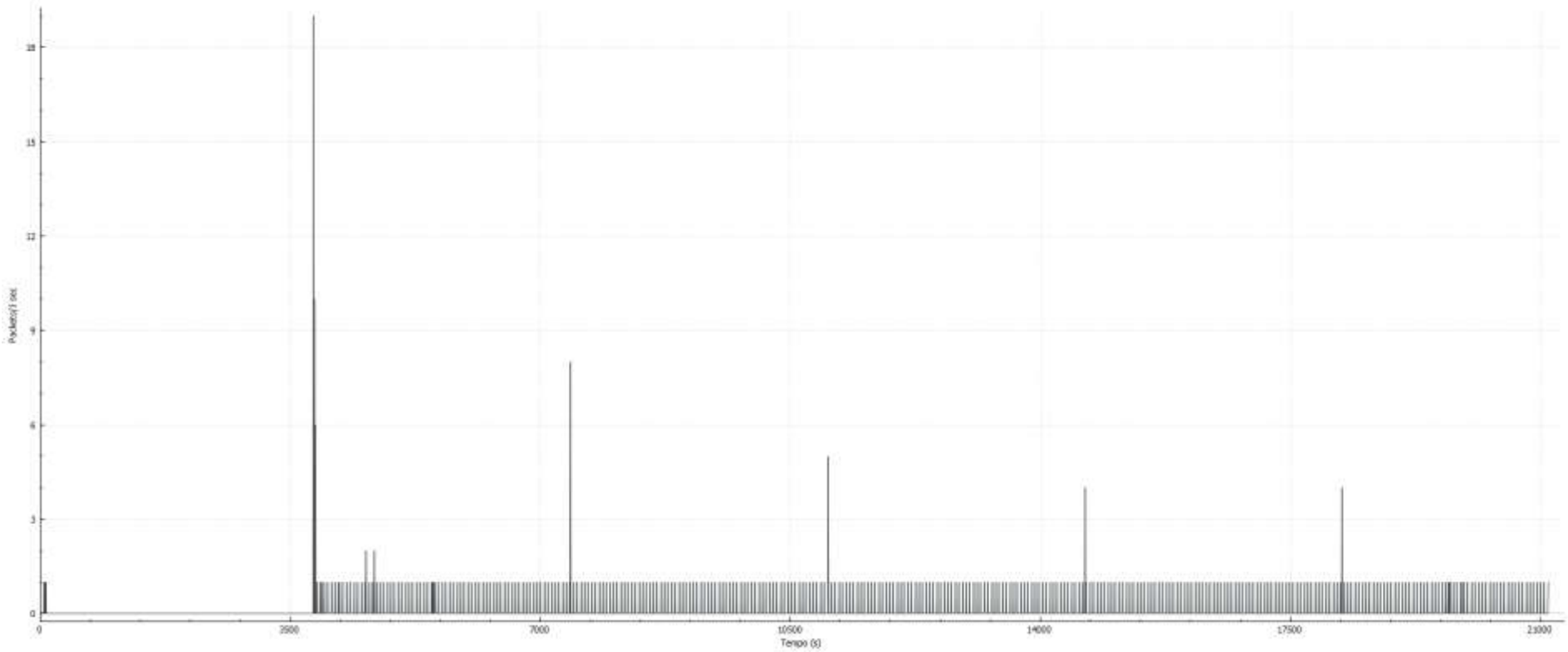
(fig. 20b – segmenti OUT tardo pomeriggio - 600)



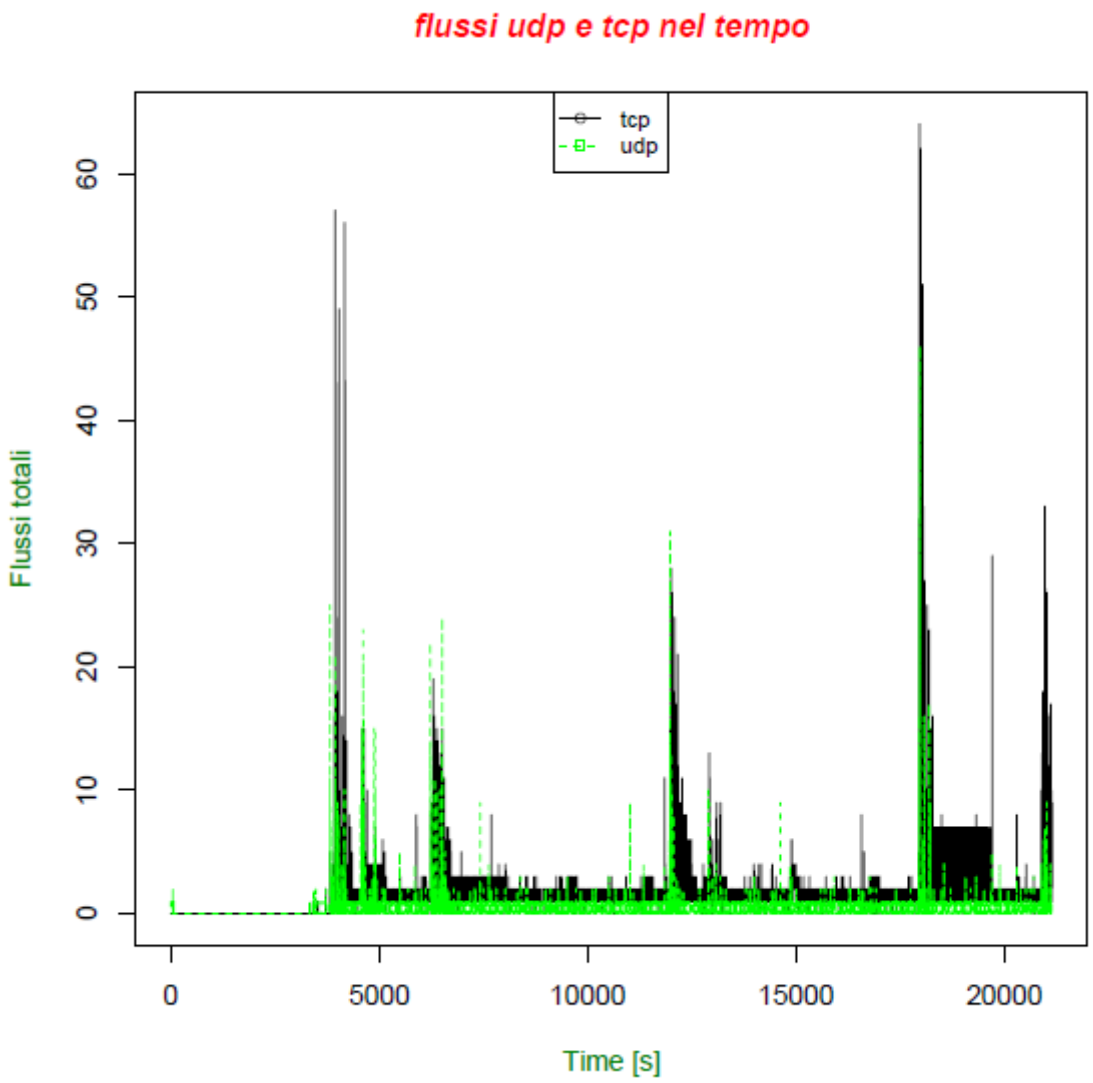
(fig. 21a – bytes IN tardo pomeriggio – 900.000)



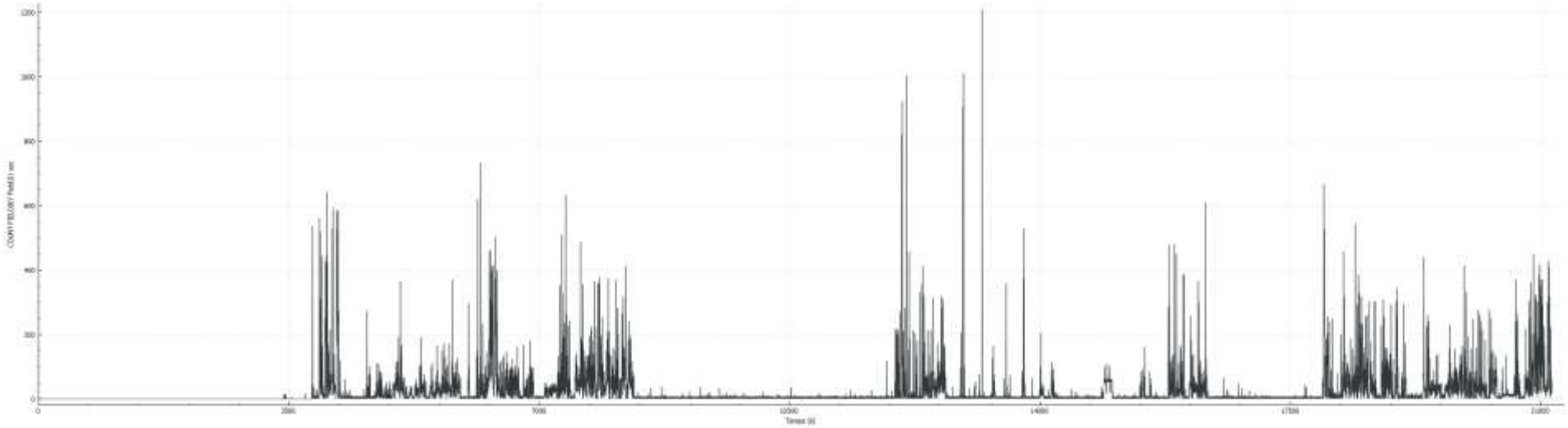
(fig. 21b – bytes OUT tardo pomeriggio – 140.000)



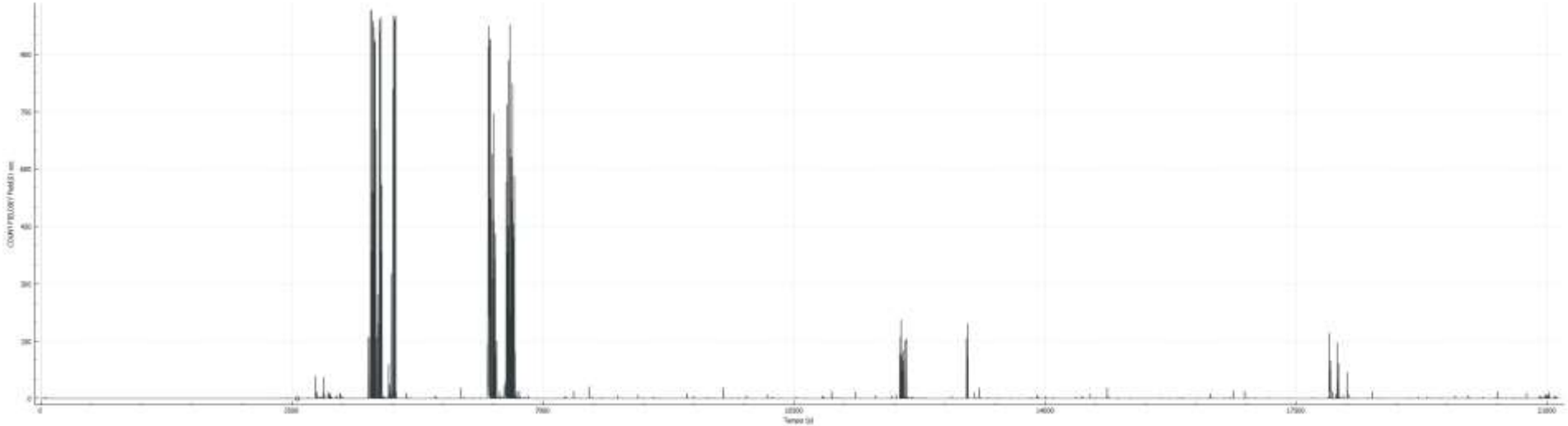
(fig. 22 – pacchetti traffico intranet tardo pomeriggio - 18)



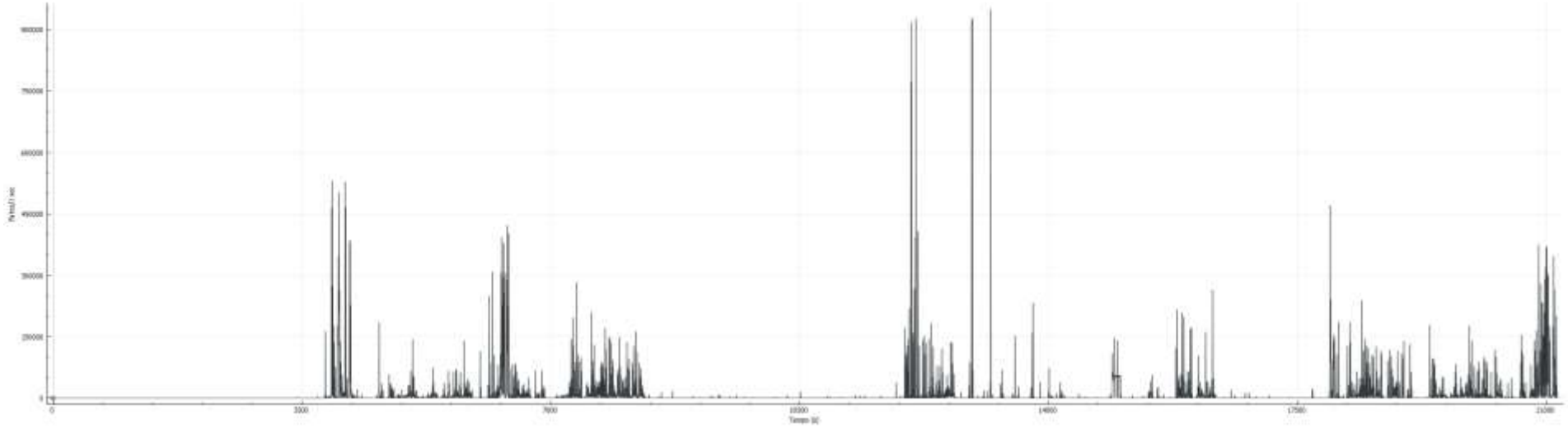
(fig. 23 – numero flussi tcp/udp tardo pomeriggio)



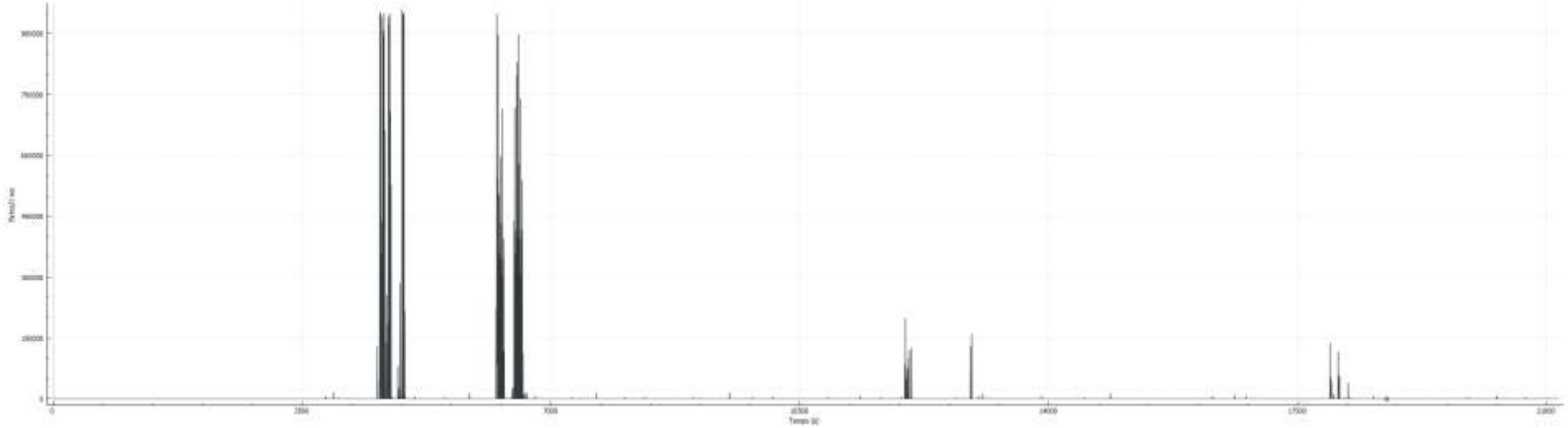
(fig. 24a – numero segmenti TCP tardo pomeriggio - 1200)



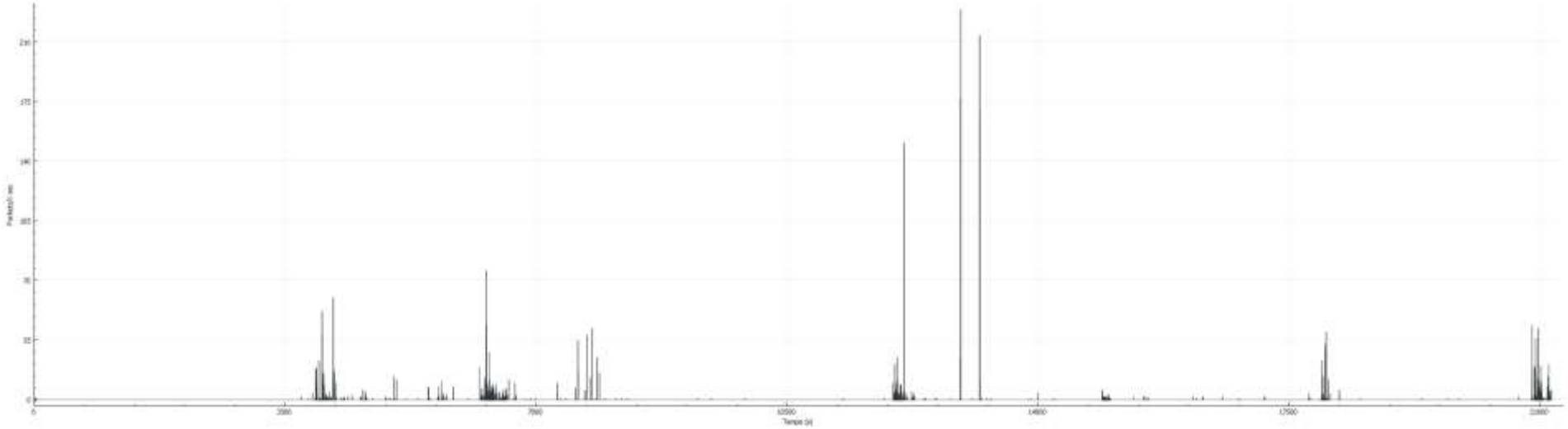
(fig. 24b – numero segmenti UDP tardo pomeriggio - 900)



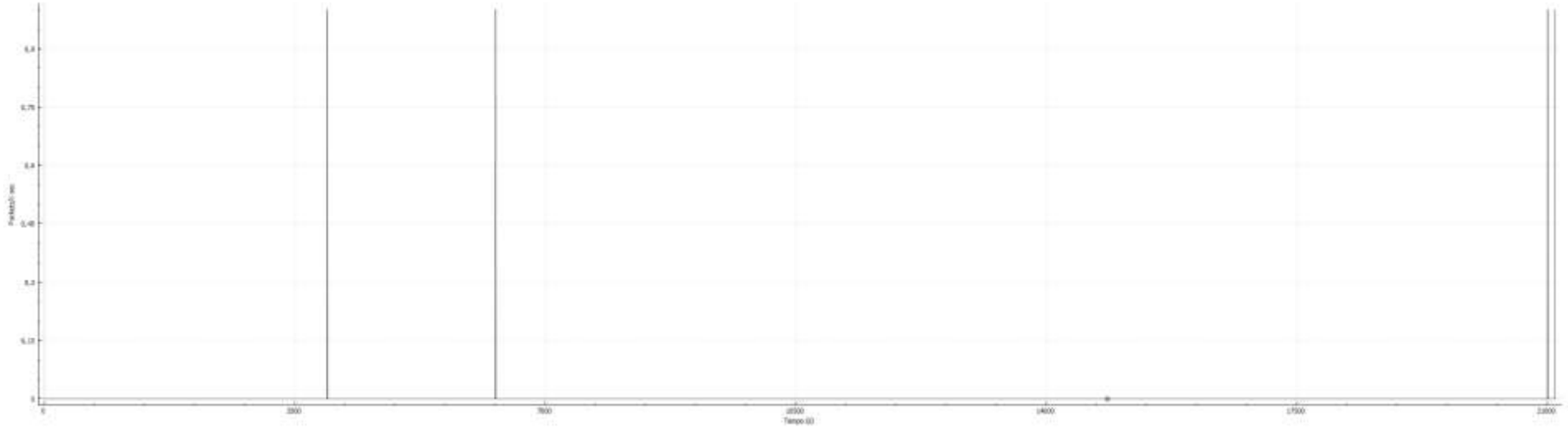
(fig. 25a – bytes TCP tardo pomeriggio 900.000)



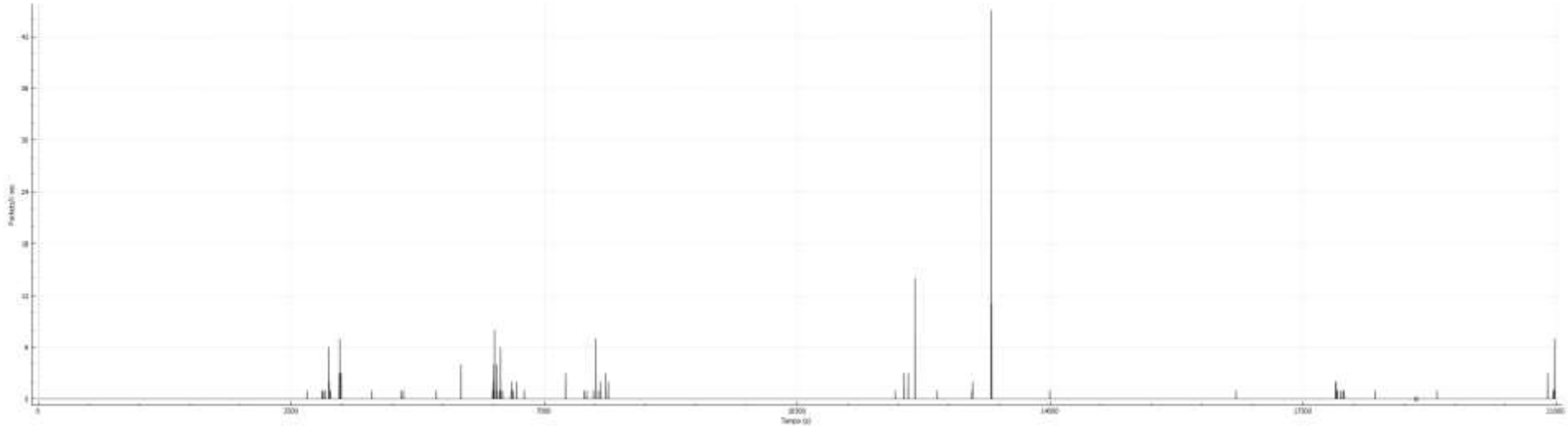
(fig. 25b – bytes UDP tardo pomeriggio 900.000)



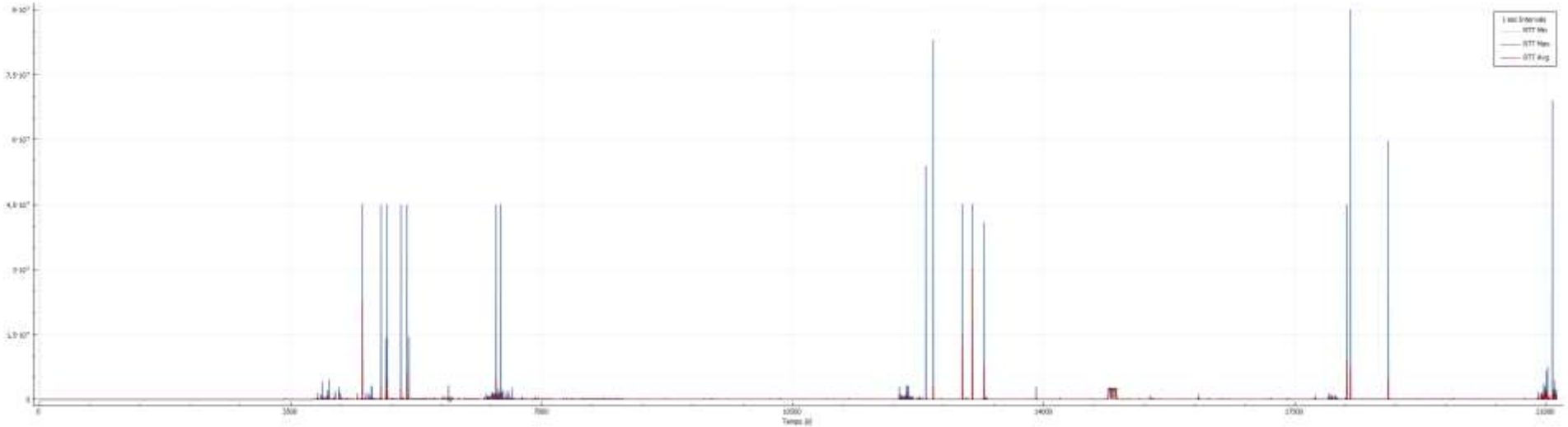
(fig. 26 – duplicate ACK tardo pomeriggio - 210)



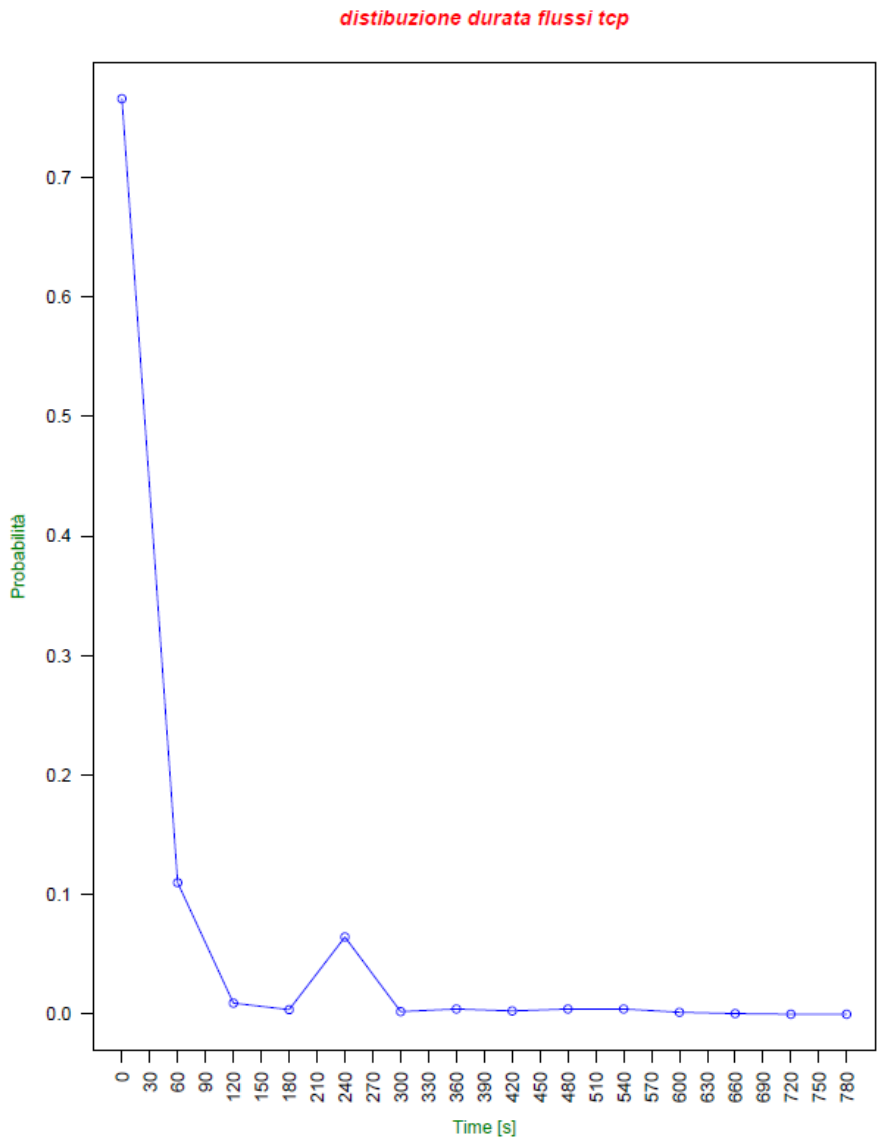
(fig. 27a – lost OUT tardo pomeriggio – 0,9)



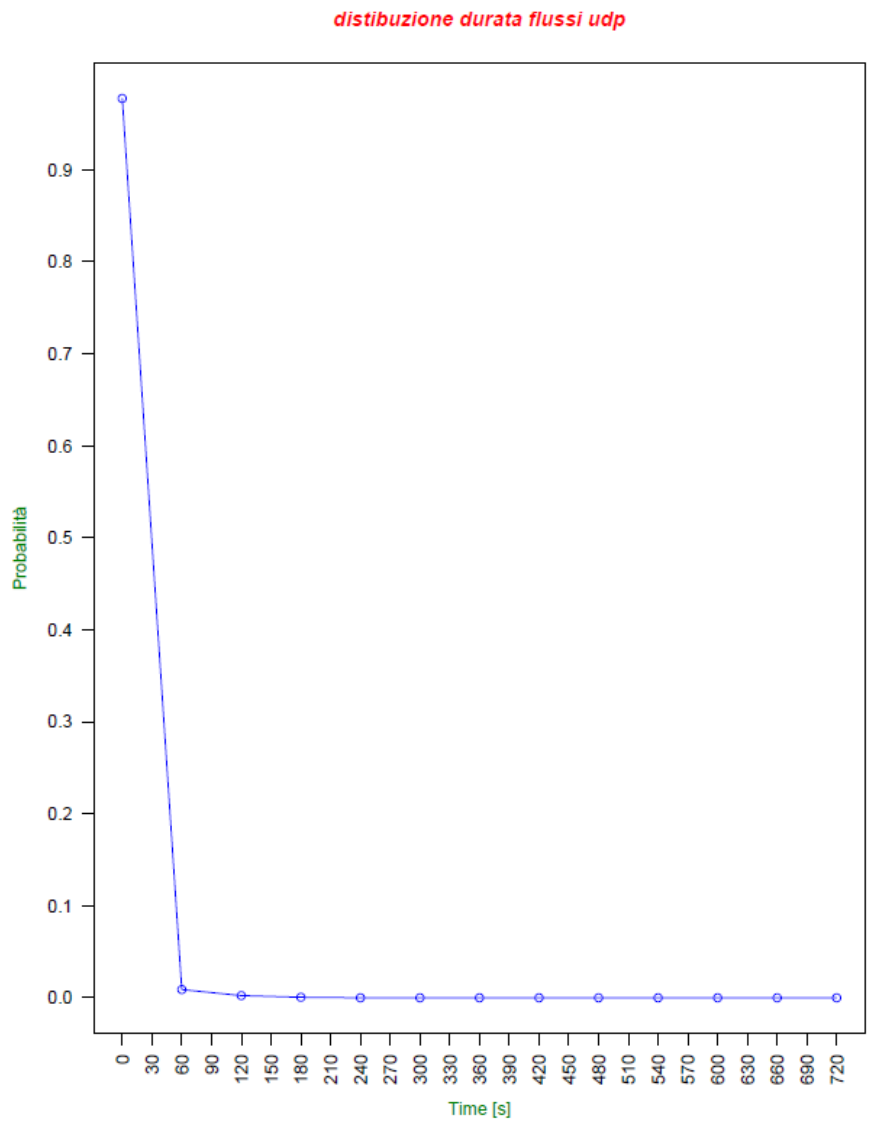
(fig. 27b – lost IN tardo pomeriggio - 42)



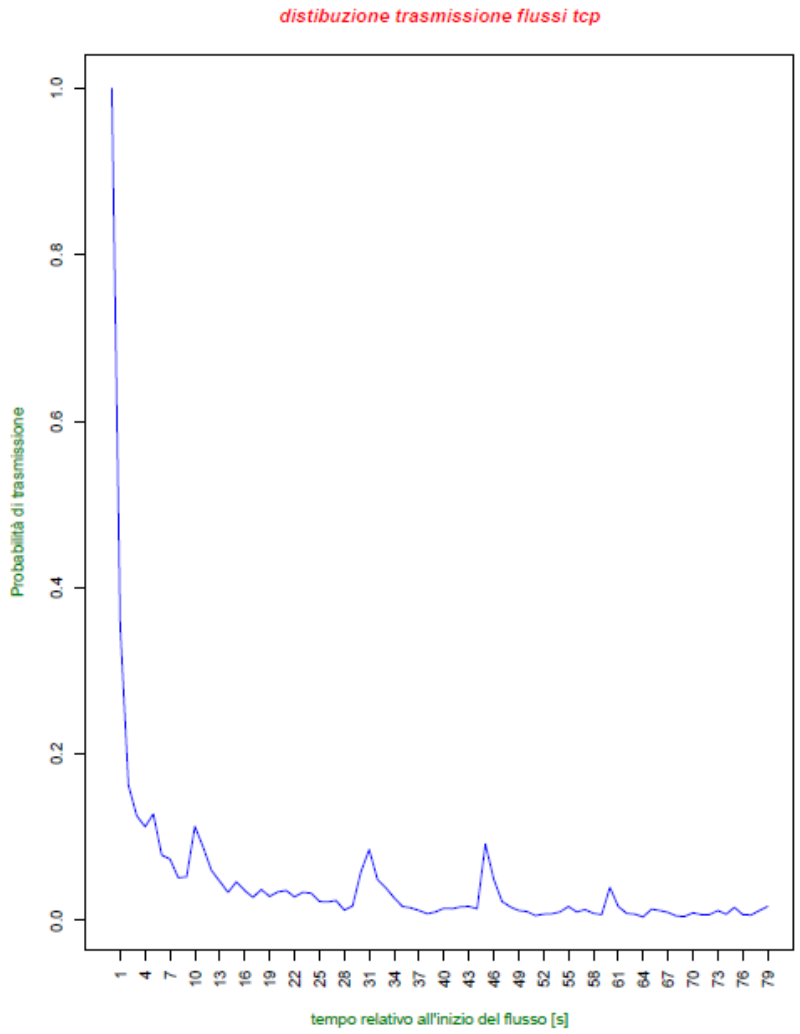
(fig. 28 – RTT Min, RTT Max, RTT Avg tardo pomeriggio – 9 * 10^9)



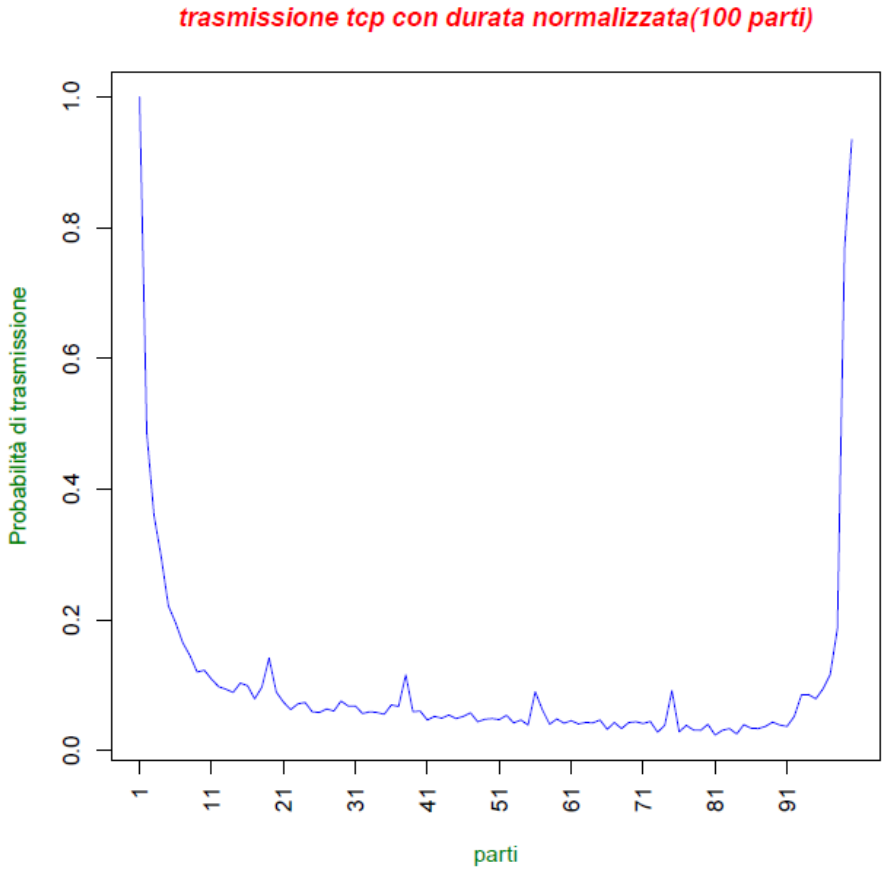
(fig. 29a - distribuzione durata flusso tcp tardo pomeriggio)



(fig. 29b – distribuzione durata flusso udp tardo pomeriggio)

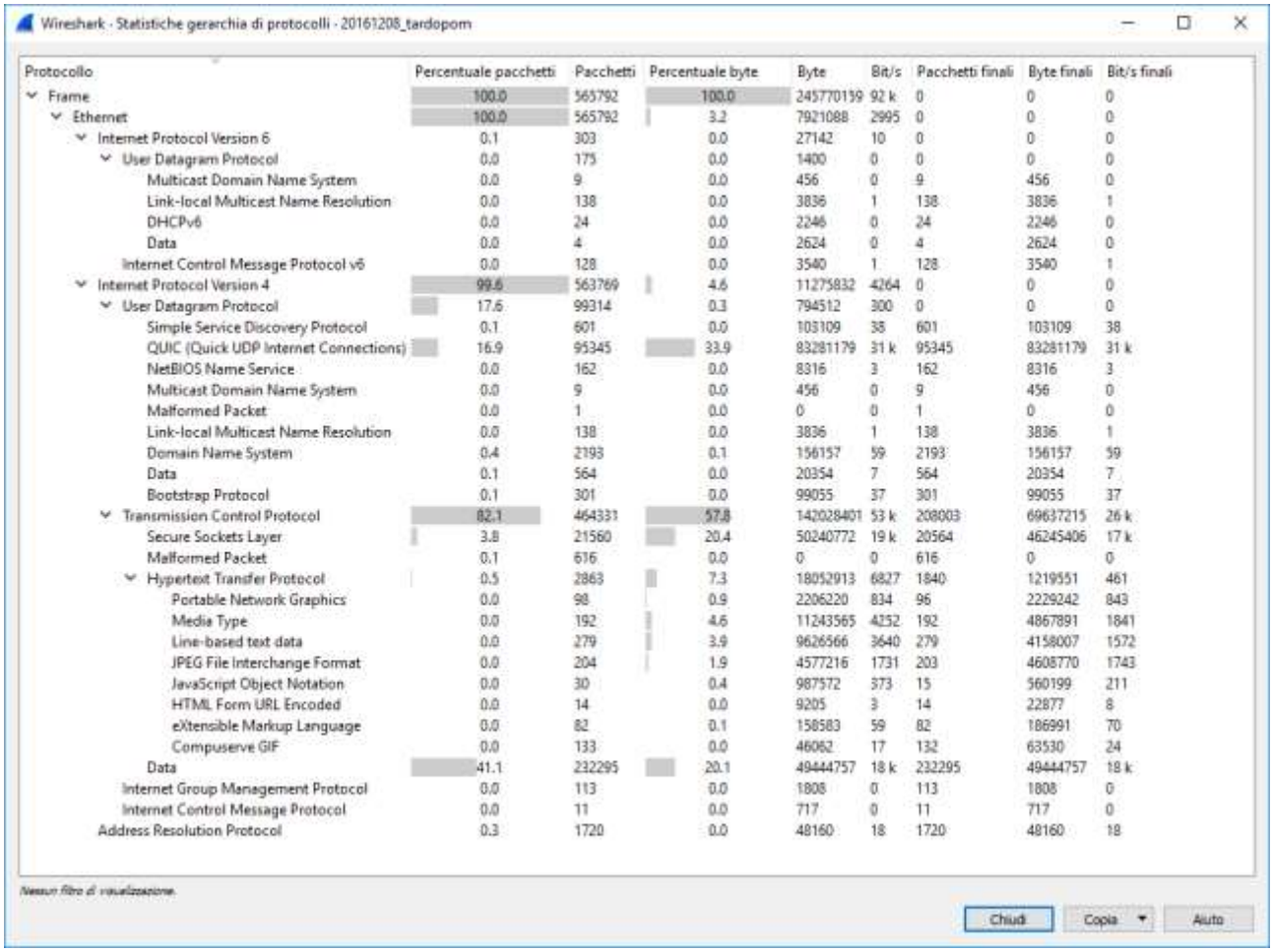


(fig. 30a - distribuzione trasmissioni flusso tcp tardo pomeriggio)



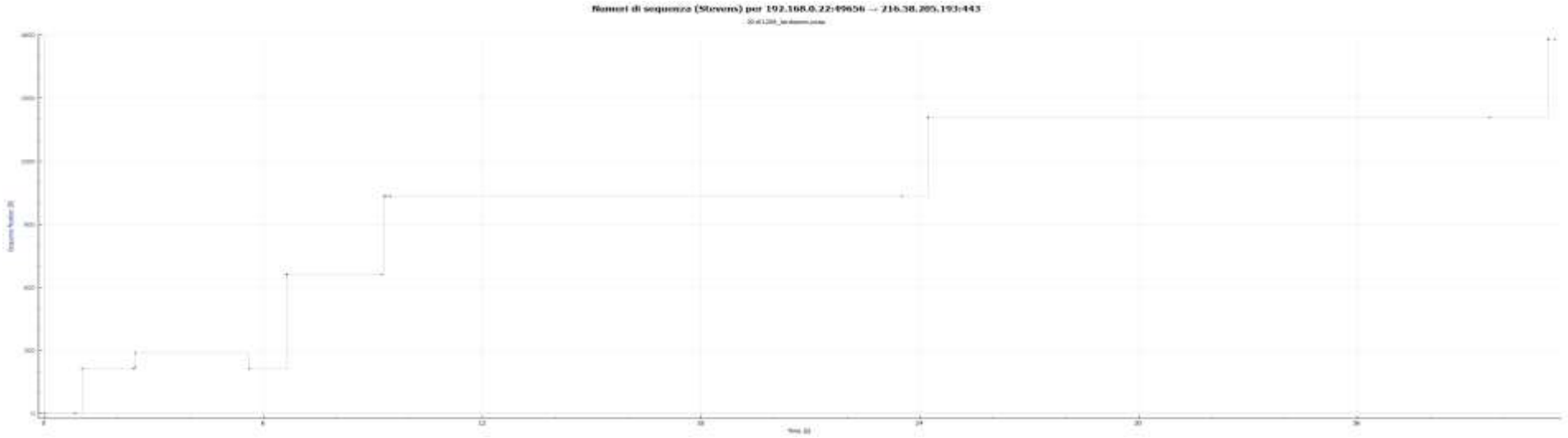
(fig. 30b – distribuzione trasmissioni flusso tcp normalizzata tardo pomeriggio)

Gerarchie protocolli

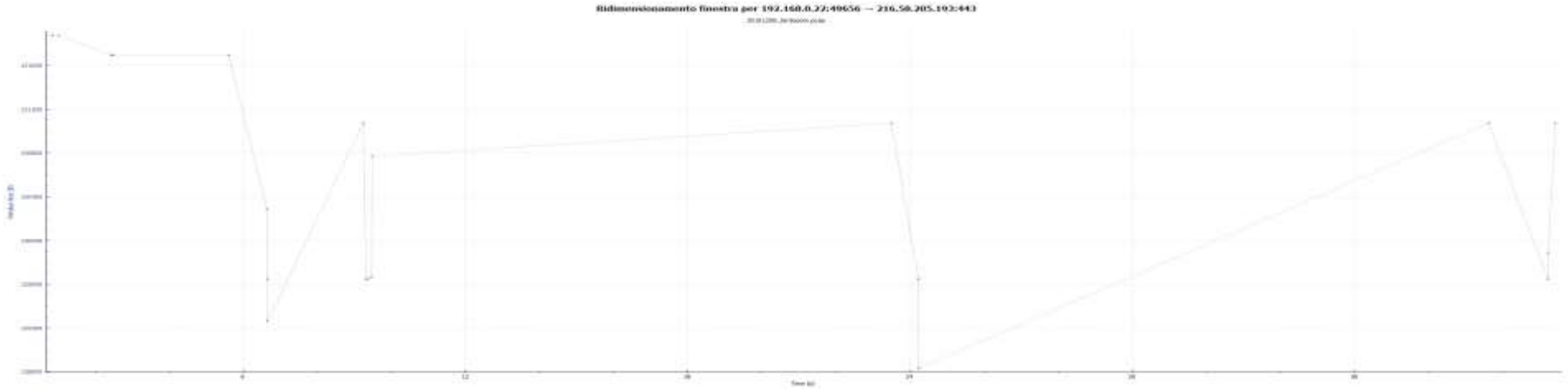


Protocollo	Percentuale pacchetti	Pacchetti	Percentuale byte	Byte	Bit/s	Pacchetti finali	Byte finali	Bit/s finali
Frame	100.0	565792	100.0	245770139	92 k	0	0	0
Ethernet	100.0	565792	3.2	7921088	2995	0	0	0
Internet Protocol Version 6	0.1	303	0.0	27142	10	0	0	0
User Datagram Protocol	0.0	175	0.0	1400	0	0	0	0
Multicast Domain Name System	0.0	9	0.0	456	0	9	456	0
Link-local Multicast Name Resolution	0.0	138	0.0	3836	1	138	3836	1
DHCPv6	0.0	24	0.0	2246	0	24	2246	0
Data	0.0	4	0.0	2624	0	4	2624	0
Internet Control Message Protocol v6	0.0	128	0.0	3540	1	128	3540	1
Internet Protocol Version 4	99.6	563769	4.6	11275832	4264	0	0	0
User Datagram Protocol	17.6	99314	0.3	794512	300	0	0	0
Simple Service Discovery Protocol	0.1	601	0.0	103109	38	601	103109	38
QUIC (Quick UDP Internet Connections)	16.9	95345	33.9	83281179	31 k	95345	83281179	31 k
NetBIOS Name Service	0.0	162	0.0	8316	3	162	8316	3
Multicast Domain Name System	0.0	9	0.0	456	0	9	456	0
Malformed Packet	0.0	1	0.0	0	0	1	0	0
Link-local Multicast Name Resolution	0.0	138	0.0	3836	1	138	3836	1
Domain Name System	0.4	2193	0.1	156157	59	2193	156157	59
Data	0.1	564	0.0	20354	7	564	20354	7
Bootstrap Protocol	0.1	301	0.0	99055	37	301	99055	37
Transmission Control Protocol	82.1	464331	57.8	142028401	53 k	208003	69637215	26 k
Secure Sockets Layer	3.8	21560	20.4	50240772	19 k	20564	46245406	17 k
Malformed Packet	0.1	616	0.0	0	0	616	0	0
Hypertext Transfer Protocol	0.5	2863	7.3	18052913	6827	1840	1219551	461
Portable Network Graphics	0.0	98	0.9	2206220	834	96	2229242	843
Media Type	0.0	192	4.6	11243565	4252	192	4867891	1841
Line-based text data	0.0	279	3.9	9626566	3640	279	4158007	1572
JPEG File Interchange Format	0.0	204	1.9	4577216	1731	203	4608770	1743
JavaScript Object Notation	0.0	30	0.4	987572	373	15	560199	211
HTML Form URL Encoded	0.0	14	0.0	9205	3	14	22877	8
eXtensible Markup Language	0.0	82	0.1	158583	59	82	186991	70
Compuserve GIF	0.0	133	0.0	46062	17	132	63530	24
Data	41.1	232295	20.1	49444757	18 k	232295	49444757	18 k
Internet Group Management Protocol	0.0	113	0.0	1808	0	113	1808	0
Internet Control Message Protocol	0.0	11	0.0	717	0	11	717	0
Address Resolution Protocol	0.3	1720	0.0	48160	18	1720	48160	18

(fig. 31 – gerarchia protocolli)



(fig. 32a – numeri sequenza flusso 1824)



(fig. 32b – window resize flusso 1824)

8. CONCLUSIONI E SVILUPPI FUTURI

Seguendo scrupolosamente la traccia fornita abbiamo tentato di dare una risposta ai quesiti indicati:

1. Raccolta dati in formato PCAP di una rete domestica secondo diversi orario di utilizzo: predisponendo un raspberry configurato come access point abbiamo acquisito e memorizzato il traffico in due fasce orarie;
2. Valutazione di software di analisi esistenti, o preparazione software di analisi che fornisca i dati richiesti sul dataset: studiando le caratteristiche di alcuni software e selezionando quelli che a nostro parere potevo contribuire all'elaborazione delle statistiche e dei grafici; wireshark ed R;
3. Presentazione dei risultati tramite tabelle e analisi degli stessi in un elaborato: predisponendo svariati grafici e commentando gli stessi;

Gli script e le statistiche utilizzate si prestano bene per un traffico di 12/24 ore ma purtroppo su file di dimensioni maggiori il tempo di elaborazione aumenta esponenzialmente e in tal caso andrebbe ottimizzato.
Tenendo da parte il traffico RDP rilevato in entrambi i range temporali, causato dalla presenza di un componente della famiglia connesso in telelavoro, l'analisi del traffico ha evidenziato che in un contesto composto da pochi dispositivi non si evidenziano particolari contese della banda disponibile concentrando la maggior parte del traffico su protocolli HTML/SSL quali la navigazione e la fruizione di video in streaming.
Inoltre si è notato che il traffico pomeridiano generato da minori, ovviamente sottoposti a stretto controllo, era dovuto prevalentemente a video in streaming, mentre quello del tardo pomeriggio generato da adulti era prevalentemente HTML.
In generale, seppur con impegno di banda differenti, l'entità del traffico in uscita è sovrapponibile al traffico in ingresso, ovvero c'è un costante scambio di pacchetti fra client e server.

Come sviluppi futuri potrebbero essere interessanti:

- Analizzare PCAP di dimensioni molto maggiori, come ad es. di un mese, a patto di rivedere pesantemente gli script al fine di ottimizzarne le prestazioni; ipotizziamo comunque che in considerazione delle abitudini di navigazione, le statistiche relative all'uso dei protocolli si avvicineranno con elevata probabilità al campione analizzato in questa tesina dovuto alle abitudini di navigazione degli utilizzatori;
- Analizzare l'utilizzo degli IP e delle porte di destinazione in base alle fasce di orario;
- Analizzare la connessione/disconnessione di dispositivi ad una rete domestica;
- Analizzare la tipologia dei dispositivi connessi ad una rete domestica ed il traffico generato da ognuno di essi;
- Analizzare le cause degli RTT elevati: se dovuti ad una reale congestione della rete o se dovuti alla configurazione dell'accesso dell'utenza domestica da parte del provider;

RIFERIMENTI

[1] Analysis of WLAN Traffic in the Wild (Caleb Phillips, Suresh Singh)
[2] Characterizing Home Network Traffic: An Inside View (Kuai Xu, Feng Wang, Lin Gu, Jianhua Gao, and Yaohui Jin)
[3] An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance (Junxian Huang,Subhabrata Sen)
[4] How to monitor packet transfers on global network (Subbarao V. Wunnava, Manoj Monga)
[5] Wireshark network analysis second edition (Laura Chappel)