

Esercitazione sullo Streaming

Anno Accademico 2009-2010

Darwin Streaming Server

Docente: prof. Davide Quaglia

Studente: Giovanni Antino

Matricola: vr092700

Studente: Iris Dimni

Matricola: vr089538

Indice

1	Darwin Streaming Server	1
1.1	Cos'è Darwin	1
1.2	Installazione	1
1.3	Avvio del Server	1
1.3.1	Primo Avvio e configurazione base	1
1.3.2	Accesso e verifica dello stato del server	2
1.4	Streaming Unicast	4
1.5	Cattura dello stream unicast	6
1.6	Playlist	6
1.6.1	Cerazione	6
2	Casi d'uso	9
2.1	Sessione Unicast	9
2.2	Nat e TCP	11
2.3	Multicast VS Unicast	12
3	Firewall	14
4	TCP/UDP	15
4.1	Streaming HTTP con VLC	16
5	Analisi degli Stream	19
5.1	Server Load	19
5.2	Analisi di una sessione Unicast	21
5.2.1	Handshake TCP	21
5.2.2	RTSP method invocation, OPTIONS	22
5.2.3	RTSP method invocation, DESCRIBE	22
5.2.4	RTSP method invocation, SETUP	24
5.2.5	RTSP method invocation, PLAY	27
5.2.6	RTSP method invocation, TEARDOWN	28
6	Appendice	29
6.1	Path a log e configurazioni	29
6.2	TCP analysis filter	30

1 Darwin Streaming Server

1.1 Cos'è Darwin

Darwin è la versione open source e multiplatforma di Apple Quick Time Server. Permette di distribuire contenuti multimediali su internet utilizzando i protocolli RTP e RTSP . Per visualizzare i contenuti in streaming si possono utilizzare

- Totem
- Vlc
- QuickTime

1.2 Installazione

Il sorgente e i pacchetti deb della versione 6.0.3 possono essere scaricati tramite i seguenti link:

Sorgente <http://static.macosforge.org/dss/downloads/DarwinStreamingSrvr6.0.3-Source.tar>

Deb x86 <http://cloud.github.com/downloads/lstoll/dss/DarwinStreamingSrvr6.0.3-Linux.deb.gz>

Deb x86_64 <https://github.com/downloads/lstoll/dss/DarwinStreamingSrvr6.0.3-L2-Linux-x64.deb.gz>

Una volta scaricato il pacchetto adatto alla propria architettura (sul desktop nella VM 1) lo si può installare con ubuntu software center.

nota l'installazione di DSS porta alla creazione (automatica) di un nuovo utente (qtss).

1.3 Avvio del Server

L'interfaccia di amministrazione, di default, si avvia sulla porta 1220 ([http://\[ServerIP\]:1220](http://[ServerIP]:1220)) Al primo avvio è richiesta la configurazione dell'utente amministratore. Una volta eseguita la procedura si verrà rediretti alla schermata di gestione del server. Qui, in alto a sinistra, è possibile vedere lo stato del server, avviarlo, arrestarlo, consultare i log ed eseguire tutte le procedure di configurazione avanzate.

1.3.1 Primo Avvio e configurazione base

Al primo avvio accedere a [http://\[ServerIP\]:1220](http://[ServerIP]:1220) e inserire come utente *admin* e password *admin* (figura 6). Al primo avvio è richiesta la configurazione delle impostazioni base del server.

- Impostare come password per l'mp3 broadcast *mp3play* (figura 2)



Figura 1: Avviare l'installazione

- Selezionare il login plain, per utilizzare SSL si dovrebbero generare i certificati (figura 3)
- Impostare il path per la directory dei contenuti. In questo caso lasciamo quella di default, il materiale video, per essere elaborato da Darwin, va convertito in mp4 o mov. Nel path di default sono già presenti dei file demo correttamente codificati (figura 4).
- In fine lasciarase deselezionata l'opzione per lo streaming attraverso la porta 80. (figura 5).

1.3.2 Accesso e verifica dello stato del server

Per verificare il corretto avvio del server e l'eventuale presenza di errori è possibile consultare l'Error Log (in basso nel frame di sinistra). In caso di avvio senza errori questo si presenta come in figura 7. Nel frame superiore si trova il pulsante per avviare/arrestare il server, alla sua destra è indicato lo stato attuale. Un paio di annotazioni sul log.

- è in ordine cronologico, la prima entry è la più vecchia, per vedere quella corrente è necessario scorrerlo fino al termine
- in caso di avvio corretto la scritta **Streaming Startup** compare in verde



Figura 2: MP3 broadcast password



Figura 3: Amministrazione SSL

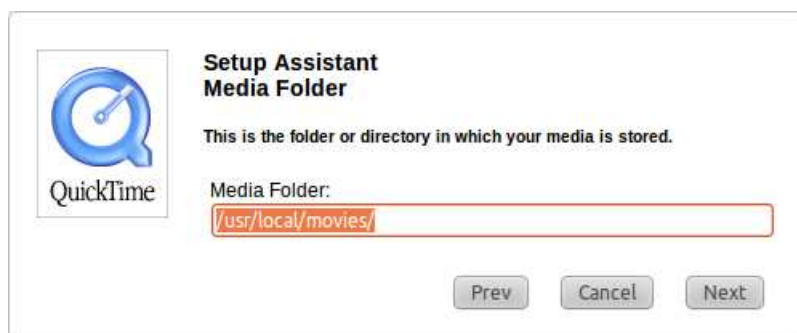


Figura 4: Media Folder

- in calce al log è presente un comando per resettarlo (figura 8)
- l'Error Log contiene solo una minima parte dei messaggi, l'elenco completo si trova in una cartella che dipende dal sistema operativo. La posizione di log, file di configurazione e playlist è definita nel file `defaultPaths.h` presente nel sorgente del server. Per comodità la lista dei path di default per Mac OS X, Windows e Linux è riportata in calce a questo pdf (Appendice 29).



Figura 5: Streaming su porta 80

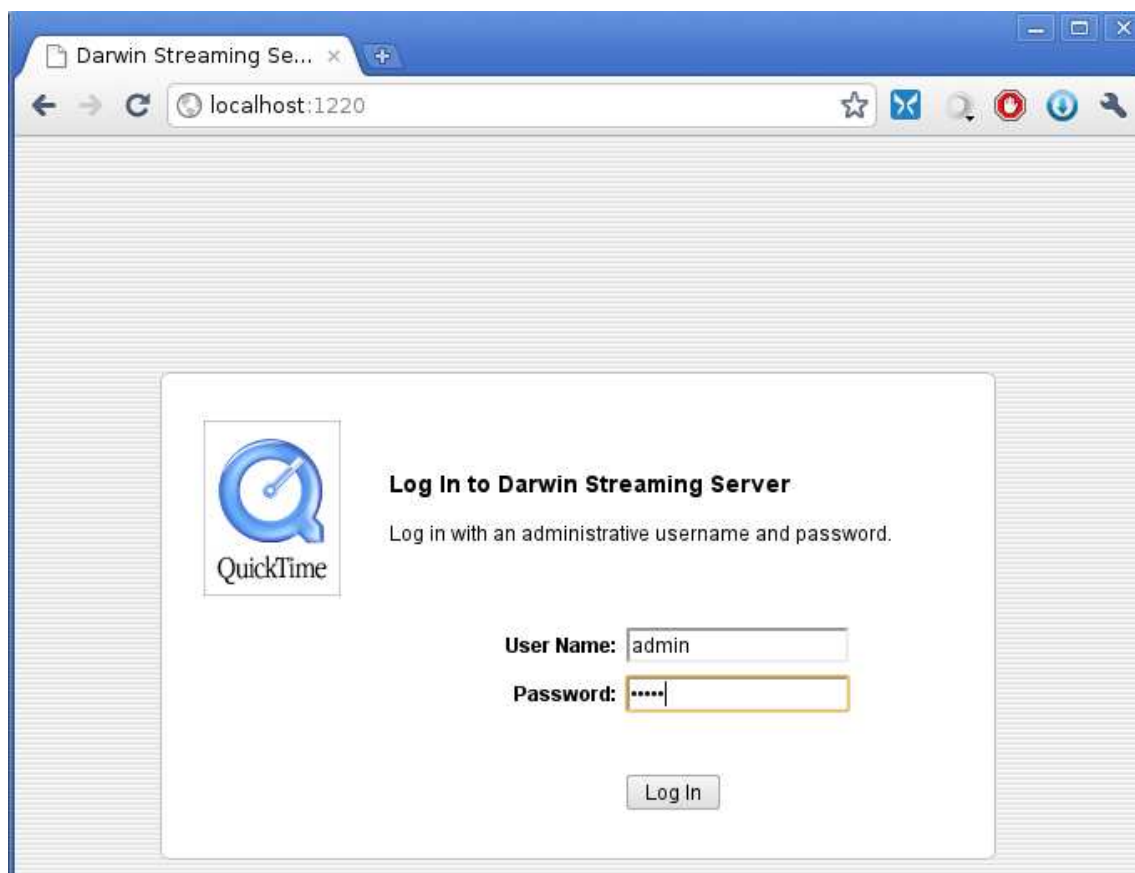


Figura 6: Schermata di Login

1.4 Streaming Unicast

Per avviare lo streaming unicast basta richiedere uno dei video demo:
rtsp : //[ServerIP]/sample_300kbit.mp4 Quando si fa una richiesta al server è possibile che questo non risponda istantaneamente o che fallisca anche query legittime. Di solito accade alla prima connessione. Questo perché DSS esegue una calibrazione della banda (solo alla prima connessione del client) e può mandare in timeout la

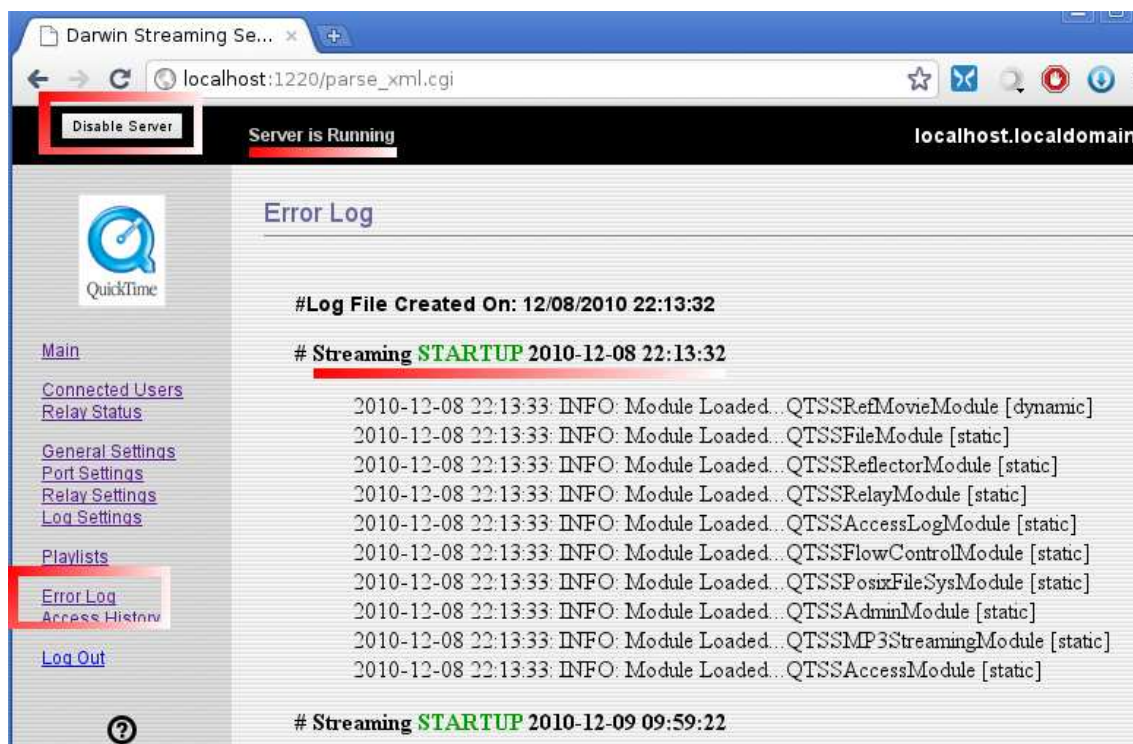


Figura 7: Error Log



Figura 8: Error Log Reset

connessione. Per vedere se le richieste arrivano ma non sono servite si può utilizzare l'Access History (figura 9). Qui sono raccolte tutte le pagine richieste al server, anche quelle illegali o malformattate. In questo modo è possibile verificare se l'eventuale mancata risposta è dovuta ad un errore di comunicazione o configurazione.

nota le pagine dei log, delle history e delle playlist non sono attive (o hanno tempi di aggiornamento lunghi), per aggiornarle è necessario ricliccare il relativo link nel frame di sinistra.

nota 2 nella sezione playlist l'uso del comando di refresh può portare a comportamenti incoerenti (doppi inserimenti...) perché ritrasmette anche le query fatte con l'operazione precedente (aggiunte di brani, eliminazioni...).

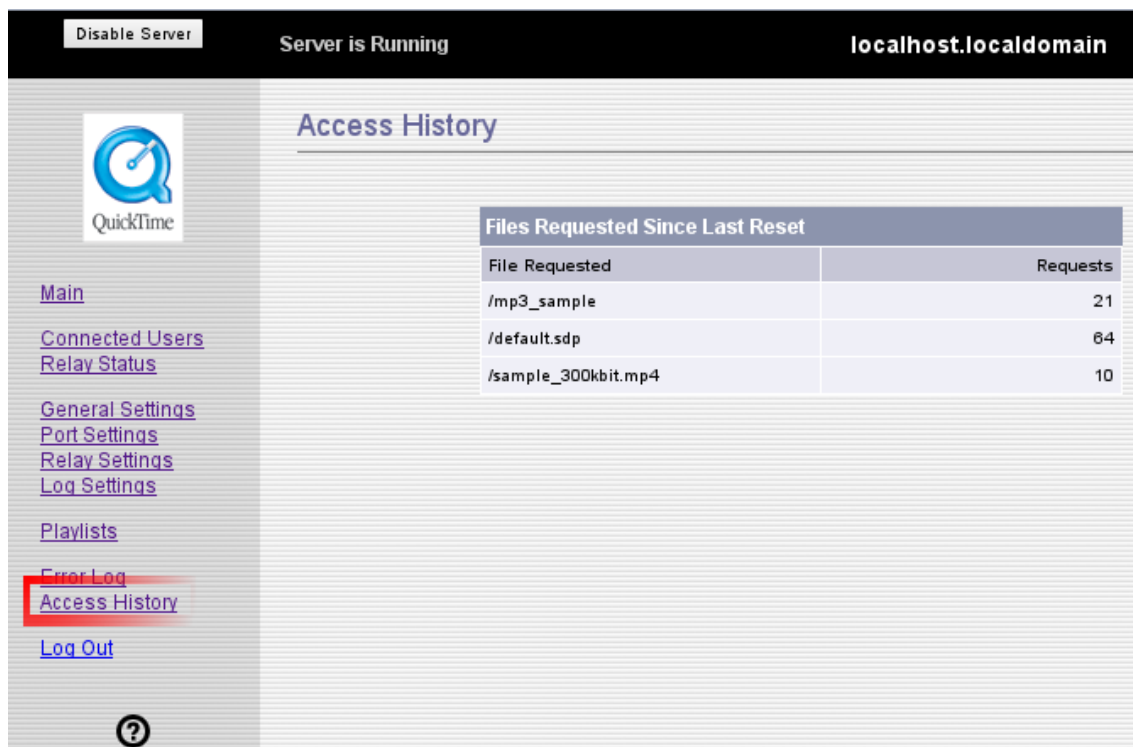


Figura 9: Access History

1.5 Cattura dello stream unicast

La configurazione si riferisce ad una macchina virtuale Ubuntu su VirtualBox. Per eseguire una cattura pulita è possibile impostare la scheda di rete della VM in modalità *Solo Host*. Questo creerà una nuova interfaccia di rete (di default *vboxnet0*) dalla quale sarà possibile catturare i soli pacchetti relativi alle sessioni di Streaming. Per farlo **non** serve impostare l'interfaccia in modalità promiscua. Una volta avviato wireshark e selezionato come target l'interfaccia *vboxnet0* possiamo procedere con la cattura dei pacchetti.

1. avviamo la cattura su *vboxnet0*
2. lanciamo la demo *vlc "rtsp : //192.168.56.101/sample_300kbit.mp4"*

1.6 Playlist

1.6.1 Cerazione

Aprire l'interfaccia di amministrazione (figura 6)

`http://[ServerIP]:1220`

Aprire la voce **Playlist** nella colonna di sinistra (figura 10), sulla destra **New Playlist**.

A questo punto configuriamo la nuova playlist (figura 11)



Figura 10: List of Playlist

- 1 il campo *Name* contiene il nome logico del file playlist
- 2 il campo *Mount Point* conterrà l'url relativo della playlist. Questo è indipendente da *Name*. Per richiamare i contenuti dovremo inserire nell'URL il contenuto di *Mount Point*
- 3 *Play Mode* permette di scegliere la modalità di riproduzione (impostiamo Sequential Looped per la prova)
- 4 metainformazioni sulla playlist
- 5 serve a risalire nell'albero delle directory del server
- 6 per aggiungere un elemento lo si trascina da *Available Content* in *Items in This Playlist*. Per gli elementi in *Items in This Playlist* è possibile impostare un parametro *weight* che determina con quale probabilità devono essere riprodotti in modalità Weighted Random
- 7 rimuove i brani selezionati dall'elenco
- 8 al termine della pagina è presente una checkbox (da selezionare) per abilitare il log sulla playlist (vivamente consigliato attivarlo, per sapere dove sono salvati Appendice 29)
- 9 per concludere **Save Changes** in basso a destra

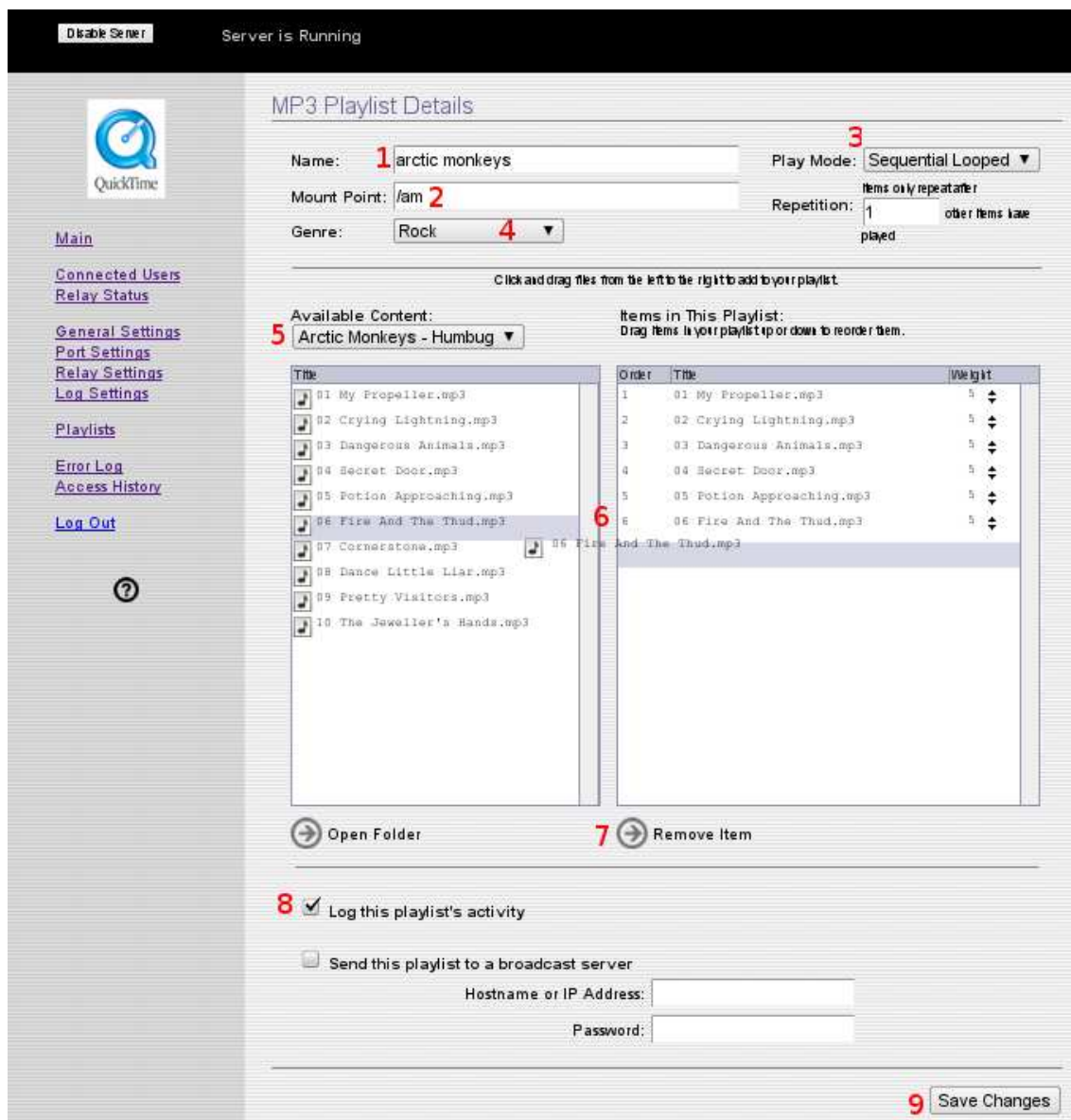


Figura 11: Creazione di una playlist

Dopo la configurazione e il **Save Changes** la nuova playlist dovrebbe comparire nell'elenco di quelle disponibili (figura 12) ma disabilitate. Per avviarla premiamo play a destra del nome. In caso di errori (formato dei file non valido, doppio binding...) non riusciremo ad avviarla e verrà marcata come la playlist test dell'esempio. In questo caso si dovrà consultare il log sulla VM, nell'**Error Log** accessibile dalla gui di DSS non compaiono gli errori delle playlist. Se tutto va bene lo stato dovrebbe cambiare in **Playing** (figura 13). Per ascoltare la playlist `http://[ServerIP]:554/[PuntoDiMountPlaylist]`

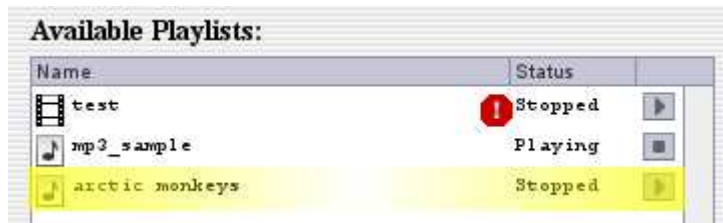


Figura 12: Playlist disabilitata

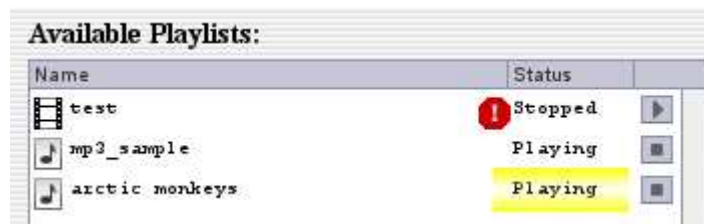


Figura 13: Playlist abilitata

2 Casi d'uso

2.1 Sessione Unicast

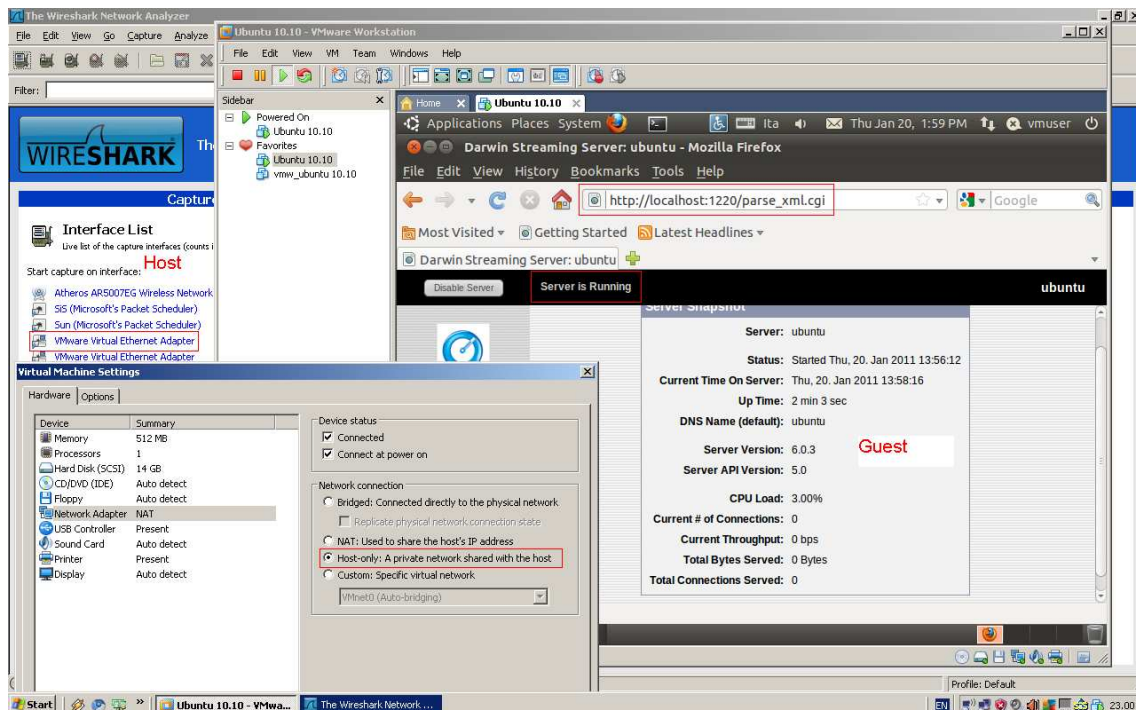


Figura 14: Primo caso d'uso

Nel primo caso d'uso impiegheremo la macchina virtuale in modalità *Solo Host*. Nella macchina virtuale DSS, nel sistema *Host* Wireshark in ascolto sull'interfaccia del sistema *Guest*.

Cambiamo le impostazioni di rete della macchina virtuale in *Host Only*. A questo punto apriamo un terminale ed eseguiamo il comando *ifconfig* per scoprire l'ip della macchina virtuale e proviamo a pingare il nostro *Host*. L'ip della macchina *Host* sarà uguale all'ip della macchina *Guest* tranne l'ultimo gruppo di cifre, da impostare a 1 (*Guest*=192.168.214.128, *Host*=192.168.214.1).

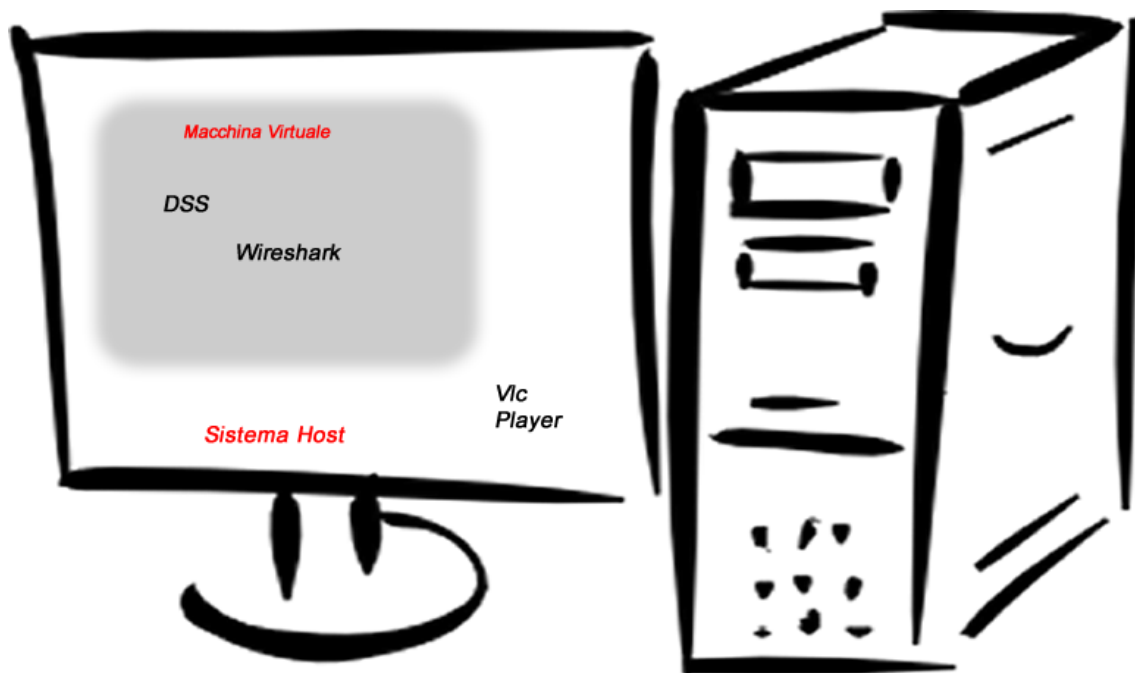


Figura 15: Schema del primo caso d'uso

Una volta fatto questo eseguiamo *wireshark*, dalla macchina *Host*, in ascolto sull'interfaccia di rete della macchina virtuale e solo dopo ci conatteremo al *DSS* (figura 15). Con *vlc player* utilizzando l'indirizzo *rtsp://VM_ip_adress/sample_300kbit.mp4*. Possiamo notare su *wireshark*, durante le fasi che precedono la connessione e l'invio dei dati al player *vlc*, l'invocazione dei diversi metodi del protocollo *RTSP* che cominciano sempre con il **Client** che fa una richiesta di *OPTIONS* al **Server**. È possibile vedere una cattura di prova del unicast nel file `sessione_unicast.libcap`

- Quale metodo *RTSP* invocano per primo ogni volta i **Client** ?
- Quali metodi hanno un corrispondente comando nel player *vlc* ?
- Trovare su *wireshark* i punti in cui vengono invocati i metodi *RTSP* quando si esegue un seek.

- Cosa succede al profilo di banda se proviamo a collegare più client contemporaneamente?
- Come distinguiamo le diverse connessioni unicast dirette alla stessa macchina ma a player differenti?
- Che differenza c'è fra uno stream (anche registrabile) e un download? (nell'esercitazione verrà messo a disposizione il file sample per il download da un server condiviso)

nota In questa sessione 5.2 è possibile vedere l'analisi della sessione demo `sessione_unicast.libcap`

2.2 Nat e TCP

Nel caso in cui sulla rete sia presente un firewall o un NAT i pacchetti UDP non possono passare dal server al client. In queste situazioni DSS opera in automatico il passaggio dallo stream dati UDP allo stream TCP .

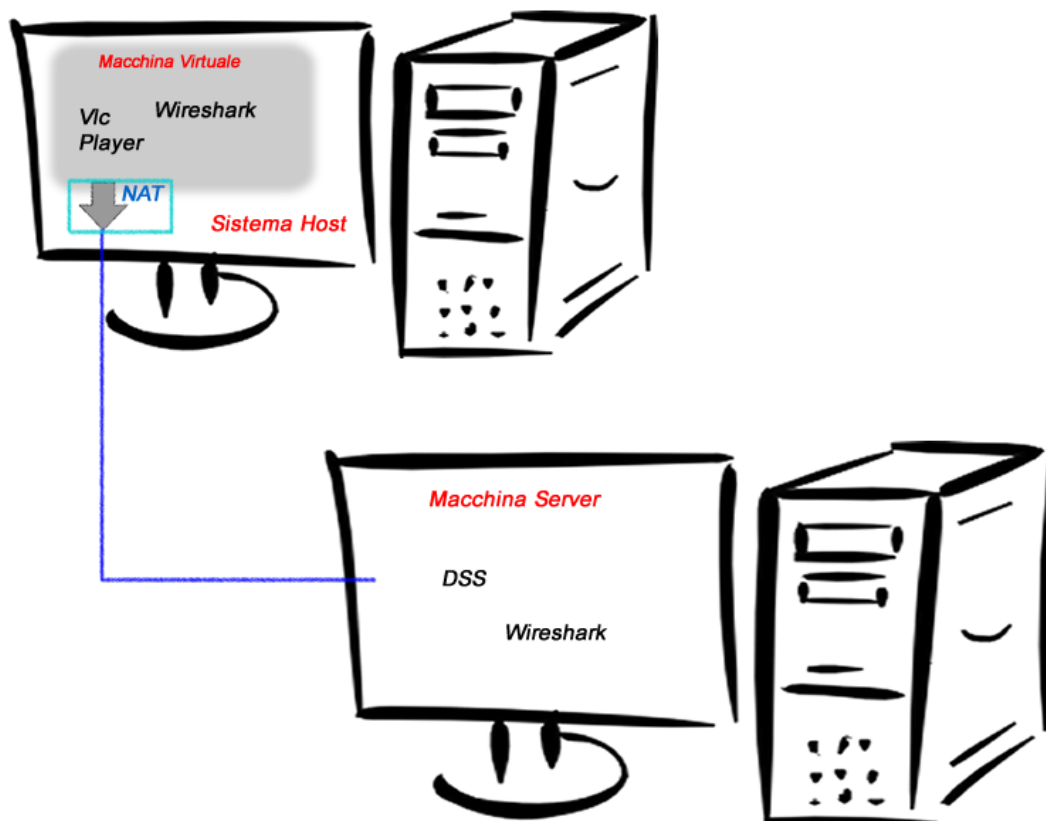


Figura 16: Schema NAT

Per simulare questa situazione abbiamo varie opzioni:

- utilizzare due macchine, una come server, l'altra come client, e sul client utilizzare lo script firewall per impostare la modalità workstation (anche 2 VM sullo stesso computer possono servire allo scopo)
- utilizzare un server nativo e tentare la connessione dall'interno della VM con interfaccia di rete NAT (figura 16)
- utilizzare un'installazione di DSS sulla macchina host e vlc all'interno della vm guest con script firewall (sempre in modalità workstation)

Per questa esercitazione faremo riferimento al secondo caso. Verrà messa a disposizione una macchina server su cui gira nativamente DSS senza firewall. A questo punto si potrà provare a connettersi dall'interno delle vm in modalità NAT. È comunque disponibile un file di cattura demo generato (sul server) in queste condizioni per l'analisi.

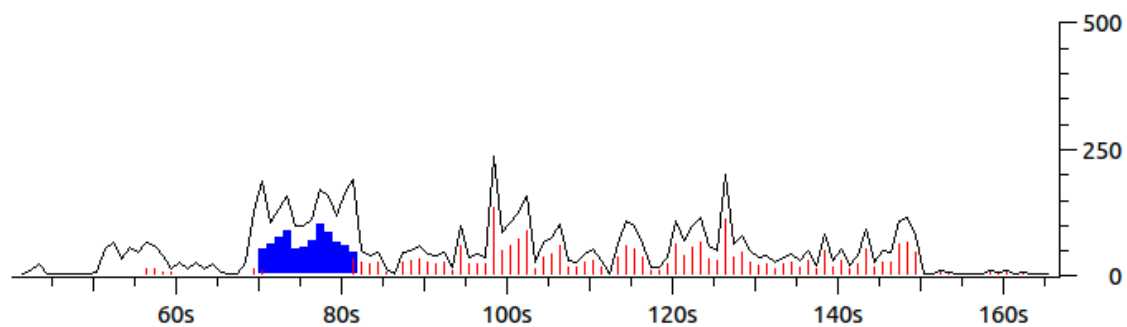


Figura 17: Profilo di una connessione nattata lato server

In figura 17 è mostrata la tipologia del traffico per queste connessioni in blu il traffico UDP in rosso il TCP dati. È possibile ricostruire questo grafico a partire dal file demo `natCap.libcap`

- perchè passa il traffico TCP ma non l'UDP ?
- come appare il corrispondente profilo lato client?
- come si comporta il client durante la trasmissione UDP ? (provate a lanciare vlc con l'opzione `-vv`)

2.3 Multicast VS Unicast

In questa sezione analizzeremo uno stream di tipo Multicast creato con l'applicazione `mp4live` (figura 18) (fa parte del pacchetto `mpeg4ip` reperibile nel repository main di Ubuntu 10.10).

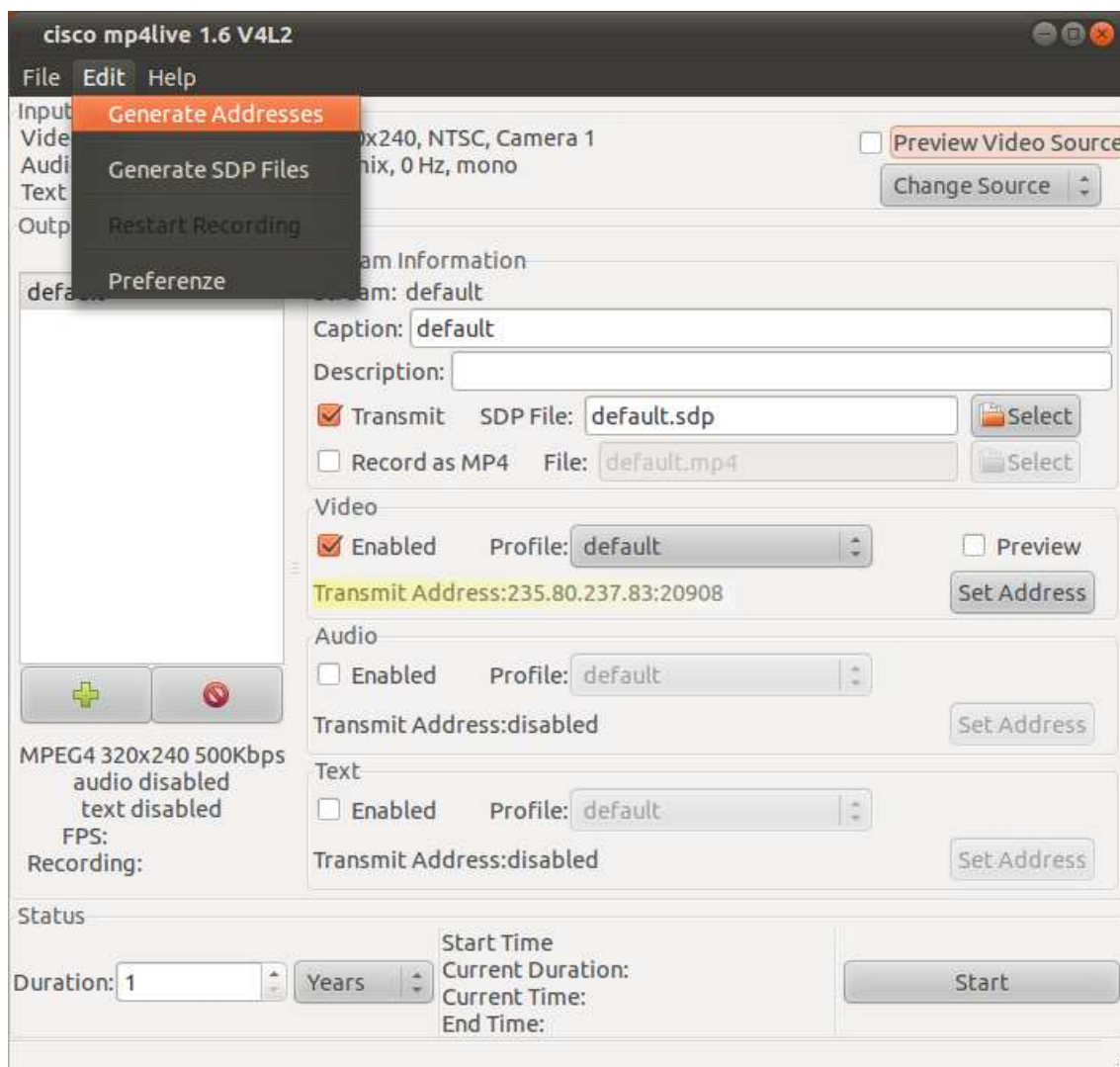


Figura 18: Interfaccia di amministrazione di mp4live

La prova consisterà nel configurare e poi analizzare uno stream di tipo Multicast.

1. per prima cosa genereremo un indirizzo di tipo multicast tramite il comando *Edit* – > *Generate Addresses*
2. poi si procederà con la creazione del file `default.sdp` necessario ai client per connettersi e comprendere le caratteristiche dello stream multicast (dall'indirizzo di bind alla tipologia dello stream incapsulato)
3. in fine avvieremo lo stream (*Start*) e l'ascolto sull'interfaccia di rete specificata nel file `.sdp`

Il problema principale di questa tipologia di streaming è che la maggior parte delle reti impedisce l'invio di pacchetti di tipo multicast. In questi casi il client

si connette, è possibile vedere viaggiare i pacchetti **IGMP** che eseguono la join ed, eventualmente, la leave, ma non i pacchetti **udp** dello stream effettivo. Il client (es. *vlc*) si avvia, non dà errori, ma resta in attesa dei pacchetti **UDP** che non arrivano. Per eseguire il test è possibile connettersi sull'interfaccia loopback (127.0.0.1). Il vantaggio maggiore degli stream di tipo Multicast è che non aumentano il carico di rete all'aumentare dei client connessi.

- È possibile vedere una cattura di prova del multicast nel file `multicast_cattura.libcap`
- per visualizzare solo i pacchetti relativi allo stream utilizzare il filtro `udp||rtsp||igmp`

Sul file di demo è possibile eseguire i seguenti test:

1. trovare i frame relativi all'uscita dal gruppo di due client
2. nell'esempio i client connessi erano 3, due hanno lasciato il gruppo, creare un grafico in cui si evidenzia, oltre all'uso di banda, il momento in cui due dei client escono dal gruppo
3. come cambia il profilo del traffico **UDP** ?
4. creare, come mostrato nella prima parte dell'esercitazione, una situazione in cui 5 client sono collegati in unicast. Catturare il traffico relativo e, tramite un grafico, confrontare il profilo di utilizzo di banda fra il caso unicast e quello multicast.
5. cosa succede se al traffico se chiudiamo 3 dei 5 client?
6. quale è la particolarità del mac dei pacchetti multicast?

Oltre al file di cattura è disponibile il `.sdp` utilizzato per configurare la connessione.

3 Firewall

Con lo script `firewall.sh` (Figura 19) è possibile simulare un ambiente in cui la maggior parte delle porte di un sistema risultino chiuse. Nel nostro caso l'uso dell'opzione 4 (modo workstation) simula il comportamento di un client standard, configurato per il solo uso del web browser e per accettare solamente connessioni "richieste". In questo tipo di ambiente è possibile vedere come i comandi **RTSP** su **TCP** siano in grado di passare mentre, i pacchetti **UDP**, in cui è incapsulato lo stream **RTP**, vengano sistematicamente bloccati. Questo non accade per gli stream su **TCP** dove è il client stesso ad iniziare la connessione. Per poter acquisire un video basato su **UDP** il client deve poter ascoltare su un determinato range di porte (per **DSS** e **QSS** 6970~6999). L'opzione 19 dello script abilita il client all'ascolto su quelle porte.


```
root@ubuntu: ~
File Modifica Visualizza Cerca Terminale Aiuto
1. HELP
2. stato di iptables
3. reset delle catene
4. chiudi il firewall (modo workstation)
5. apri porta in input
6. apri porta in output
7. default policy, tutto aperto
8. path attuale iptables (/sbin/iptables)
9. definisci un nuovo path per iptables
10. chiudi il firewall (COMPLETO)
11. abilita logging su INPUT-OUTPUT-FORWARD
12. echo INPUT chain
13. echo OUTPUT chain
14. echo FORWARD chain
15. permetti loopback
16. permetti ping
17. salva configurazione
18. carica configurazione salvata
19. apri porte udp (6970:6999) DARWIN
20. apri porte udp (definisci range)
21. apri porte tcp (definisci range)
99. exit
(1. HELP)?:> █
```

Figura 19: Firewall Script

4 TCP/UDP

In questa sezione vedremo come generare in modo immediato uno stream RTSP basato su TCP. Questo genere di Stream è generalmente utilizzato per aggirare le limitazioni imposte dai firewall. A differenza della versione classica su UDP richiede la creazione di un canale. La prima cosa che si nota di diverso è la massiva presenza degli ack di TCP. Per ogni segmento RTSP – TCP trasmesso viene reinviato un ack. Un'altra differenza sono:

- la presenza delle ritrasmissioni (per ritardo, perdita, ...)
- le rinegoziazioni della dimensione della finestra di invio
- la presenza delle informazioni temporali di RTT negli ack a livello di TCP
- la presenza di pacchetti duplicati

Per ricavare queste informazioni è possibile utilizzare i filtri tcp.analysis di wireshark (appendice 30). È possibile testare questi strumenti sul file `http_session.libcap`. La selezione del protocollo di trasmissione con DSS avviene in automatico in base alle capacità del client e della connessione. Per forzare l'uso di HTTP utilizzerem un altro server, VLC streaming server. Il server di streaming è installato insieme al client VLC.

4.1 Streaming HTTP con VLC

- dal menù Media (figura 20) selezionare **trasmissione**



Figura 20: Trasmissione

- per la prova selezioniamo un video qualsiasi dalla macchina locale cliccando **aggiungi**. È possibile utilizzare lo stesso di DSS
/usr/local/movies/sample_300kbit.mp4
Questo dovrebbe comparire nell'elenco a sx del pulsante una volta aggiunto. (figura 21). Premiamo poi **Flusso** in basso a dx nella finestra.

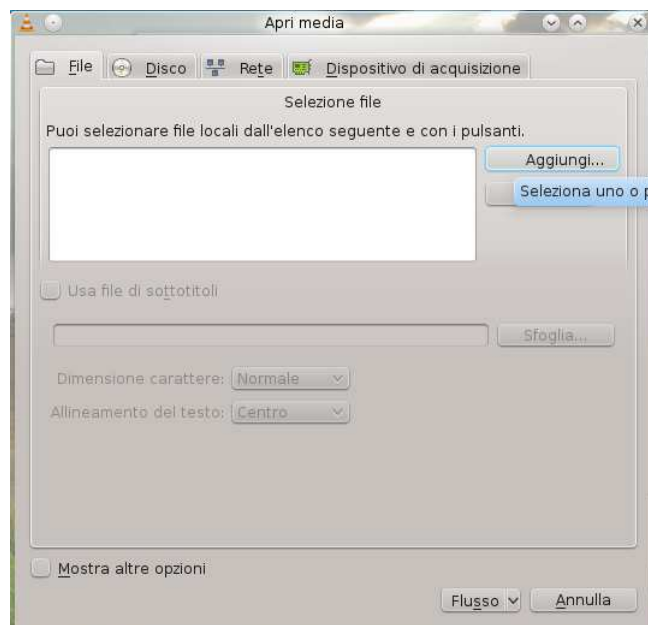


Figura 21: Selezione del media

- La schermata successiva (figura 22) serve per configurare le opzioni di acquisizione nel caso si utilizzi una sorgente differente da un media locale. Passiamo alla prossima con il bottone **Successivo** in basso a dx.

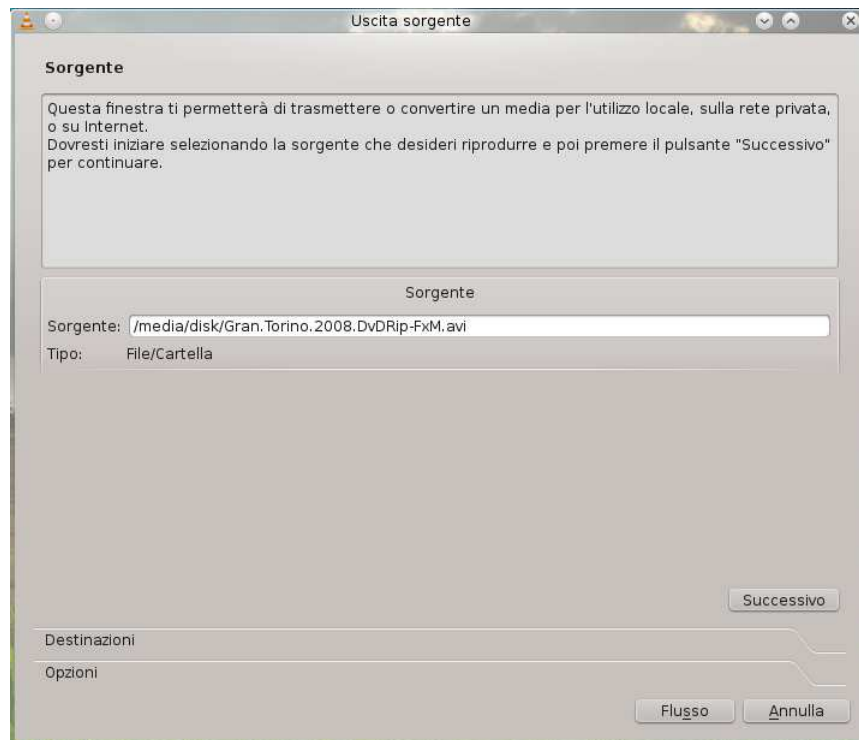


Figura 22: Configurazione dello Stream

- In questa sezione (figura 23) si configurano le opzioni di streaming. Dalla tendina in alto selezioniamo HTTP (figura 24). La finestra cambierà presentando le impostazioni per lo streaming su HTTP (figura 25). Impostiamo la porta a 8080 e il punto di mount a `\httpstream`. Impostiamo la transcodifica (la trasformazione in tempo reale del formato del file) a **Video -- h264 + AAC (MP4)**. Quest'ultimo accorgimento permette di utilizzare un qualsiasi formato in input senza preoccuparsi della sua correttezza (su DSS questa cosa non è possibile) e clicchiamo su **Successivo**.

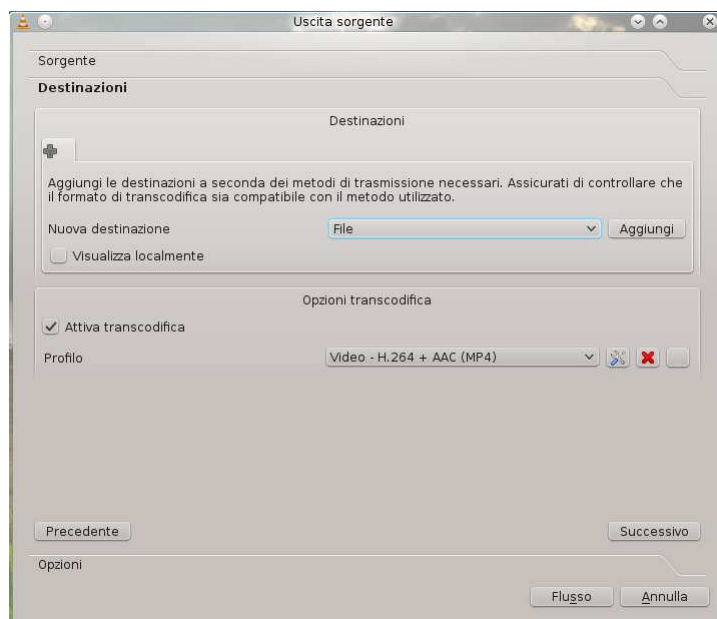


Figura 23: Configurazione del tipo di Stream

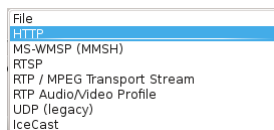


Figura 24: Configurazione del tipo di Stream - http

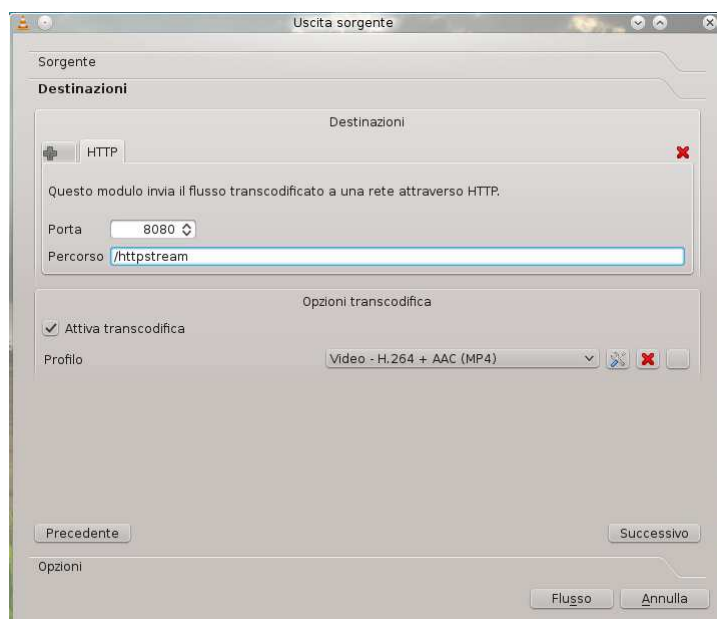


Figura 25: Configurazione del tipo di Stream - porta e mountpoint

- ora lo stream è pronto e avviato. È possibile verificarne il funzionamento aprendo l'url `http://localhost:8080/httpstream`. Per farlo utilizziamo `mplayer` da terminale. Nella figura 26 è possibile vedere come si presenta VLC se tutto è andato a buon fine.

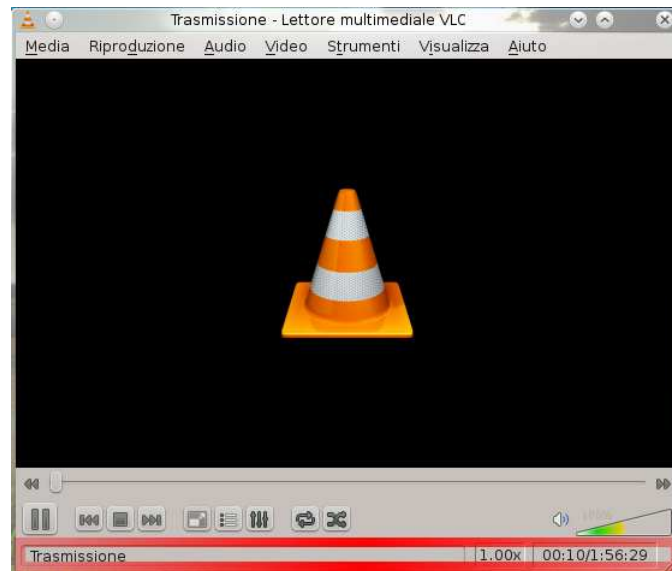


Figura 26: Verifica dell'avvio del sever

nota la transcodifica in tempo reale richiede una macchina molto potente e potrebbe mettere in crisi le VM. È consigliabile disattivarla per risparmiare risorse.

nota la transcodifica viene fatta in real-time, per questa ragione, se la macchina non è sufficientemente veloce si potrebbero verificare delle perdite di frame dal lato del server.

nota in alcuni casi l'utilizzo di questa opzione peggiora notevolmente la qualità del video (soprattutto se associata a scalature o ricampionamenti). Per queste ragioni, generalmente, è preferibile processare il video offline e far fare al server solo la trasmissione.

5 Analisi degli Stream

5.1 Server Load

In questa sezione è riportata un'analisi dimostrativa di come varia il carico del server nel caso di streaming multicast. Per avere un'idea della capacità (in termini di ampiezza di banda) richiesta a un server di streaming è possibile fare riferimento alla tabella 27. In questa tabella sono riportati i carichi di lavoro nel caso di stream Unicast.

Unicast packets on typical IP networks have a single source and destination. Most traffic on IP networks between clients and servers on today's networks is unicast. Multicast packets have a single source and multiple destinations. Multicast packets can save network bandwidth when multiple clients need to view the same streaming media simultaneously. Instead of sending out individual unicast packets to each client, a single stream of multicast packets can be viewed by multiple clients. *Today's commodity Internet does not support multicast (Internet 2 does).*

dalla documentazione di DSS

Connection	Bandwidth ($\frac{kbits}{second}$)	Unicast Streams
T1 or DS-1	1,544,000	3
T2 or DS-2	6,000,300	12
T3 or DS-3	44,736,000	89
OC-3	155,000,000	310
T4 or DS-4	274,000,000	548
OC-12	600,000,000	1,200
OC-48	2,400,000,000	4,800
OC-192	10,000,000,000	20,000

Figura 27: 500 kilobit Unicast streams

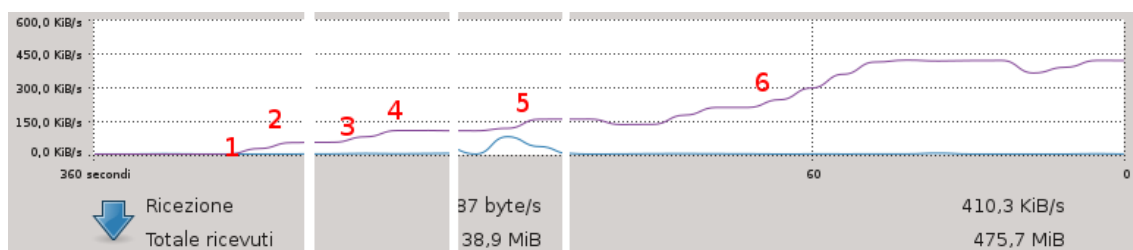


Figura 28: Avvio e connessione di client su streaming Unicast

1 qui il server non ha ancora iniziato lo stream

1-2 all'inizio della trasmissione il carico passa a 40 kbps

2 connessione del primo client

3 e 4 connessione del secondo e del terzo client

5 connessione del quarto client

6 connessione dei restanti 5 client.

5.2 Analisi di una sessione Unicast

I partecipanti alla sessione sono:

Server	192.168.56.101
Client	192.168.56.1

La sessione è salvata nel file *sessione_unicast.libcap*, il formato è quello standard delle libcap leggibile con wireshark. La sessione inizia con l'handshake al frame 24.

5.2.1 Handshake TCP

24	3.712615	192.168.56.1	192.168.56.101	TCP	59268 > rtsp	[SYN]	Seq=0 Win=5840 Len=0 MSS=1460 TSV=1276907 TSER=0 WS=7
25	3.712903	192.168.56.101	192.168.56.1	TCP	rtsp > 59268	[SYN, ACK]	Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=4294922909 TSER=1276907 WS=6
26	3.712938	192.168.56.1	192.168.56.101	TCP	59268 > rtsp	[ACK]	Seq=1 Ack=1 Win=5888 Len=0 TSV=1276907 TSER=4294922909

pagina 27 RFC 793

I pacchetti dal 24 al 26 si occupano del handshake. Al 24 il client imposta a 1 il flag di richiesta di connessione (SYN). Il server risponde al pacchetto 25 impostando a 1 i flag ACK e SYN, confermando la connessione. Il client, a sua volta, risponde col pacchetto 26, impostando a 1 il flag ACK e concludendo l'handshake.

5.2.2 RTSP method invocation, OPTIONS

Frame 27 Pacchetto RTSP , dal **Client** al **Server** , con il flag PUSH impostato, in cui vengono richieste le opzioni per il video sample_300kbit.mp4

C ▷ S: OPTIONS rtsp://192.168.56.101/sample_300kbit.mp4
RTSP/1.0
CSeq: 1
User-Agent: LibVLC/1.1.4 (LIVE555 Streaming Media
v2010.04.09)

Frame 28-29 Alla richiesta il **Server** risponde con un ACK(28), in un pacchetto TCP e prosegue inviando le opzioni al **Client** (29)

S ▷ C: RTSP/1.0 200 OK
Server: DSS/6.0.3 (Build/526.3; Platform/Linux;
Release/Darwin Streaming Server; State/Development;)
Cseq: 1
Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE,
OPTIONS, ANNOUNCE, RECORD

pagina 30 RFC 2326

5.2.3 RTSP method invocation, DESCRIBE

Il **Client** manda un pacchetto TCP di ACK(30) e prosegue con pacchetto RTSP , facendo una richiesta DESCRIBE(31).

The DESCRIBE method retrieves the description of a presentation or media object identified by the request URL from a server. It may use the Accept header to specify the description formats that the client understands. The server responds with a description of the requested resource. The DESCRIBE reply-response pair constitutes the media initialization phase of RTSP.

pagina 31 RFC 2326

Frame 30-31 Describe

C ▷ S: DESCRIBE rtsp://192.168.56.101/sample_300kbit.mp4
RTSP/1.0
CSeq: 2
Accept: application/sdp
User-Agent: LibVLC/1.1.4 (LIVE555 Streaming Media
v2010.04.09)

Frame 32-33 Il **Server** manda un ACK(32) e un pacchetto RTSP/SDP, con la risposta alla richiesta del **Client**

S ▷ C: RTSP/1.0 200 OK
Server: DSS/6.0.3 (Build/526.3; Platform/Linux;
Release/Darwin Streaming Server; State/Development;)
Cseq: 2
Last-Modified: Mon, 06 Oct 2008 07:14:43 GMT
Cache-Control: must-revalidate
Content-length: 1269
Date: Sun, 28 Nov 2010 18:02:13 GMT
Expires: Sun, 28 Nov 2010 18:02:13 GMT
Content-Type: application/sdp
x-Accept-Retransmit: our-retransmit
x-Accept-Dynamic-Rate: 1
Content-Base: rtsp://192.168.56.101/sample_300kbit.mp4/

Il **Server** risponde con un pacchetto RTSP/SDP (33)

```
v=0
o=StreamingServer 3499956133 1223277283000 IN IP4 192.168.56.101
s=/sample_300kbit.mp4
u=http:///
e=admin@
c=IN IP4 0.0.0.0
b=AS:258
t=0 0
a=control:*
a=mpeg4-iod:data:application/mpeg4-iod;base64,AoJrAE...
a=isma-compliance:1,1.0,1
a=range:npt=0- 70.00000
m=video 0 RTP/AVP 96
b=AS:209
a=3GPP-Adaptation-Support:1
a=rtpmap:96 MP4V-ES/90000
a=control:trackID=3
a=cliprect:0,0,480,380
a=framesize:96 380-480
a=fmtp:96 profile-level-id=1;config=000001B0F3000001...
a=mpeg4-esid:201
m=audio 0 RTP/AVP 97
b=AS:48
a=3GPP-Adaptation-Support:1
a=rtpmap:97 mpeg4-generic/22050/2
a=control:trackID=4
a=fmtp:97 profile-level-id=15; mode=AAC-hbr; sizelength=13;
indexlength=3; indexdeltalength=3; config=1390
a=mpeg4-esid:101
```

5.2.4 RTSP method invocation, SETUP

Il client manda un ACK(34) e prosegue facendo al server una richiesta di SETUP:

The SETUP request for a URI specifies the transport mechanism to be used for the streamed media. A client can issue a SETUP request for a stream that is already playing to change transport parameters, which a server MAY allow. If it does not allow this, it MUST respond with error 455 Method Not Valid In This State. For the benefit of any intervening firewalls, a client must indicate the transport parameters even if it has no influence over these parameters, for example, where the server advertises a fixed multicast address. Since SETUP includes all transport initialization information, firewalls and other intermediate network devices (which need this information) are spared the more arduous task of parsing the DESCRIBE response, which has been reserved for media initialization. The Transport header specifies the transport parameters acceptable to the client for data transmission; the response will contain the transport parameters selected by the server.

pagina 33 RFC 2326

Frame 34-35 SETUP

C ▷ S: SETUP rtsp:// 192.168.56.101/sample_300kbit.mp4/trackID=3
RTSP/1.0
CSeq: 3
Transport: RTP/AVP;unicast;client_port=50710-50711
User-Agent: LibVLC/1.1.4 (LIVE555 Streaming Media
v2010.04.09)

Frame 36

S ▷ C: RTSP/1.0 200 OK
Server: DSS/6.0.3 (Build/526.3; Platform/Linux;
Release/Darwin Streaming Server; State/Development;
)
Cseq: 3
Last-Modified: Mon, 06 Oct 2008 07:14:43 GMT
Cache-Control: must-revalidate
Session: 8517205348755364675
Date: Sun, 28 Nov 2010 18:02:13 GMT
Expires: Sun, 28 Nov 2010 18:02:13 GMT
Transport: RTP/
AVP; unicast; source=192.168.56.101;
client_port=50710-50711; server_port=6970-6971;
ssrc=4FF0E421

Frame 37-38 **Client** e **Server** si scambiano ancora il pacchetto di setup e relativa risposta come conseguenza del fatto che il **Server** a seguito di una richiesta di setup genera un campo session tramite il quale il **Client** si identifica:

Frame 37

C > S: SETUP rtsp:// 192.168.56.101/sample_300kbit.mp4/trackID=3
 RTSP/1.0
 CSeq: 3
 Session: 8517205348755364675
 Transport: RTP/AVP; unicast; client_port=50710-50711
 User-Agent: LibVLC/1.1.4 (LIVE555 Streaming Media
 v2010.04.09)

Frame 38

S > C: RTSP/1.0 200 OK
 Server: DSS/6.0.3 (Build/526.3; Platform/Linux;
 Release/Darwin Streaming Server; State/Development;
)
 Cseq: 3
 Last-Modified: Mon, 06 Oct 2008 07:14:43 GMT
 Cache-Control: must-revalidate
 Session: 8517205348755364675
 Date: Sun, 28 Nov 2010 18:02:13 GMT
 Expires: Sun, 28 Nov 2010 18:02:13 GMT
 Transport: RTP/
 AVP;unicast;source=192.168.56.101;
 client_port=50710-50711; server_port=6970-6971;
 ssrc=4FF0E421

5.2.5 RTSP method invocation, PLAY

Il client a questo punto manda al server un pacchetto rtsp con richiesta di PLAY:

The PLAY method tells the server to start sending data via the mechanism specified in SETUP. A client MUST NOT issue a PLAY request until any outstanding SETUP requests have been acknowledged as

pagina 33 RFC 2326

Frame 39 PLAY

C > S: PLAY rtsp:// 192.168.56.101/sample_300kbit.mp4/
 RTSP/1.0
 CSeq: 5
 Session: 8517205348755364675
 Range: npt=0.000-
 User-Agent: LibVLC/1.1.4 (LIVE555 Streaming Media
 v2010.04.09)

Frame 40-46 Sono pacchetti UDP mandati dal **Server** che contengono varie informazioni riguardo lo streaming del nostro contenuto.

Frame 47

```
S > C:      RTSP/1.0 200 OK
           Server:  DSS/6.0.3 (Build/526.3; Platform/Linux;
           Release/Darwin Streaming Server; State/Development;
           )
           Cseq:   5
           Session: 8517205348755364675
           Range:  npt=0.00000-70.00000
           RTP-Info:
           url=rtsp:// 192.168.56.101/sample_300kbit.mp4/trackID=3;
           seq=29193; rtptime=2027080437, url=rtsp://192.168.56.101
           /sample_300kbit.mp4/trackID=4; seq=29818;
           rtptime=1915190729
```

5.2.6 RTSP method invocation, TEARDOWN

Il resto, sono pacchetti UDP con dati di streaming fino alla richiesta da parte del **Client** di TEARDOWN(3676)

The TEARDOWN request stops the stream delivery for the given URI, freeing the resources associated with it. If the URI is the presentation URI for this presentation, any RTSP session identifier associated with the session is no longer valid. Unless all transport parameters are defined by the session description, a SETUP request has to be issued before the session can be played again.

pagina 36 RFC 2326

```

C > S:  TEARDOWN rtsp:// 192.168.56.101/sample_300kbit.mp4/
RTSP/1.0
CSeq:  6
Session:  8517205348755364675
User-Agent:  LibVLC/1.1.4 (LIVE555 Streaming Media
v2010.04.09)
Richiesta alla quale il server risponde chiudendo la
connessione ed il flusso dei dati.
S->C: RTSP/1.0 200 OK
Server:  DSS/6.0.3 (Build/526.3; Platform/Linux;
Release/Darwin Streaming Server; State/Development;
)
Cseq:  6
Session:  8517205348755364675
Connection:  Close

```

6 Appendice

6.1 Path a log e configurazioni

In questa sezione sono riportati i percorsi standard a log e configurazioni del server. Il file originale si trova nel pacchetto sorgente (scaricabile da 1.2).

```

1 #ifdef __Win32__
2
3 # define DEFAULTPATHS_DIRECTORY_SEPARATOR      "\\"
4
5 # define DEFAULTPATHS_ETC_DIR                  "c:\\Program Files\\Darwin Streaming Server\\"
6 # define DEFAULTPATHS_ETC_DIR_OLD              "c:\\Program Files\\Darwin Streaming Server\\"
7 # define DEFAULTPATHS_SSM_DIR                  "c:\\Program Files\\Darwin Streaming Server\\"
8 # define DEFAULTPATHS_SSM_MODULES              "c:\\Program Files\\Darwin Streaming Server\\"
9 # define DEFAULTPATHS_LOG_DIR                   "c:\\Program Files\\Darwin Streaming Server\\Logs"
10 # define DEFAULTPATHS_MOVIES_DIR                "c:\\Program Files\\Darwin Streaming Server\\Movies\\"
11 # define DEFAULTPATHS_PID_FILE                  ""
12 # define DEFAULTPATHS_PID_DIR                   ""
13 #elif __MacOSX__
14 # define DEFAULTPATHS_DIRECTORY_SEPARATOR      "/"
15
16 # define DEFAULTPATHS_ETC_DIR                   "/Library/QuickTimeStreaming/Config/"
17 # define DEFAULTPATHS_ETC_DIR_OLD               "/etc/"
18 # define DEFAULTPATHS_SSM_DIR                   "/Library/QuickTimeStreaming/Modules/"
19 # define DEFAULTPATHS_LOG_DIR                   "/Library/QuickTimeStreaming/Logs/"
20 # define DEFAULTPATHS_MOVIES_DIR                 "/Library/QuickTimeStreaming/Movies/"
21 # define DEFAULTPATHS_PID_DIR                   "/var/run/"
22
23 #else
24
25 # define DEFAULTPATHS_DIRECTORY_SEPARATOR      "/"
26
27 # define DEFAULTPATHS_ETC_DIR                   "/etc/streaming/"
28 # define DEFAULTPATHS_ETC_DIR_OLD               "/etc/"
29 # define DEFAULTPATHS_SSM_DIR                   "/usr/local/sbin/StreamingServerModules/"
30 # define DEFAULTPATHS_LOG_DIR                   "/var/streaming/logs/"
31 # define DEFAULTPATHS_MOVIES_DIR                 "/usr/local/movies/"
32 # define DEFAULTPATHS_PID_DIR                   "/var/run/"
33
34 #endif

```

Figura 29: Path ai file di configurazione

6.2 TCP analysis filter

```
1 tcp.analysis.ack-lost-segment ACKed Lost Packet
2   No value
3   This frame ACKs a lost segment
4 tcp.analysis.ack-rtt The RTT to ACK the segment was
5   Time duration
6   How long time it took to ACK the segment (RTT)
7 tcp.analysis.acks-frame This is an ACK to the segment in frame
8   Frame number
9   Which previous segment is this an ACK for
10 tcp.analysis.bytes-in-flight Number of bytes in flight
11   Unsigned 32-bit integer
12   How many bytes are now in flight for this connection
13 tcp.analysis.duplicate-ack Duplicate ACK
14   No value
15   This is a duplicate ACK
16 tcp.analysis.duplicate-ack-frame Duplicate to the ACK in frame
17   Frame number
18   This is a duplicate to the ACK in frame #
19 tcp.analysis.duplicate-ack-num Duplicate ACK #
20   Unsigned 32-bit integer
21   This is duplicate ACK number #
22 tcp.analysis.fast-retransmission Fast Retransmission
23   No value
24   This frame is a suspected TCP fast retransmission
25 tcp.analysis.flags TCP Analysis Flags
26   No value
27   This frame has some of the TCP analysis flags set
28 tcp.analysis.keep-alive Keep Alive
29   No value
30   This is a keep-alive segment
31 tcp.analysis.keep-alive-ack Keep Alive ACK
32   No value
33   This is an ACK to a keep-alive segment
34 tcp.analysis.lost-segment Previous Segment Lost
35   No value
36   A segment before this one was lost from the capture
37 tcp.analysis.out-of-order Out Of Order
38   No value
39   This frame is a suspected Out-Of-Order segment
40 tcp.analysis.retransmission Retransmission
41   No value
42   This frame is a suspected TCP retransmission
43 tcp.analysis.reused-ports TCP Port numbers reused
44   No value
45   A new tcp session has started with previously used port numbers
46 tcp.analysis.rto The RTO for this segment was
47   Time duration
48   How long transmission was delayed before this segment was retransmitted (RTO)
49 tcp.analysis.rto-frame RTO based on delta from frame
50   Frame number
51   This is the frame we measure the RTO from
52 tcp.analysis.window-full Window full
53   No value
54   This segment has caused the allowed window to become 100% full
55 tcp.analysis.window-update Window update
56   No value
57   This frame is a tcp window update
58 tcp.analysis.zero-window Zero Window
59   No value
60   This is a zero-window
61 tcp.analysis.zero-window-probe Zero Window Probe
62   No value
63   This is a zero-window-probe
64 tcp.analysis.zero-window-probe-ack Zero Window Probe Ack
65   No value
66   This is an ACK to a zero-window-probe
```

Figura 30: tcp analysis filter