Tstat generates three different types of measurement collections:

- **Log files**, storing flow-level measurements.
- **Histograms**, storing the distribution of a given quantity during a time interval.
- **RRD**, storing histograms in a RRD database.

LOG Files

Tstat creates a set of `TXT` files where each row corresponds to a different flow and each column is associated to a specific measure. When it is useful, the columns are grouped according to **C2S** - Client-to-Server and **S2C** - Server-to-Client traffic directions.
For most logs, the first row contains a summary with the description of all columns.
The generated logs are:

log_tcp_complete, **log_tcp_nocomplete**:
report every *TCP connection* that has been tracked by Tstat. A TCP connection is identified when the first `SYN` segment is observed, and is ended when either:

- the `FIN/ACK` or `RST` segments are observed;
- no data packet has been observed (from both sides) for a default timeout of 10s after the opening SYN segment (`TCP_Singleton_Time`), or 5min after the last data packet (`TCP_Idle_Time`); the actual values of the timeout timers is controlled by the corresponding global constants provided via the `-G` command line option;

Tstat discards all the connections for which the three way handshake is not properly seen. Then, in case a connection is correctly closed it is stored in `log_tcp_complete`, otherwise in `log_tcp_nocomplete`.

log_udp_complete:
reports every tracked UDP flow pair. An UDP flow pair is identified when the first UDP segment is observed for a UDP socket pair, and is ended when no packet has been observed (from both sides) for 10s after the first packet (`UDP_Singleton_Time`), or 3min 20s after the last data packet (`UDP_Idle_Time`); the actual values of the timeout timers is controlled by the corresponding global constants provided via the `-G` command line option.
By default, Skype and chat protocols running over UDP are reported only in a separate file (see `LOG_ALL_UDP` in `param.h`).

log_video_complete:
Tstat can produce a `log_video_complete` file which logs every TCP Video connection that has been tracked. Currently are classified as Video the RTMP connections, TLS connections associated to YouTube, and HTTP connections based on 2 distinguished approaches:

1. HTTP request URL matching (Tstat HTTP engine)
2. HTTP response DPI (Tstat Streaming Classifier engine)

The Tstat HTTP engine (1) classifies relevant video-related HTTP connections (YouTube, Vimeo, generic FLV/MP4, VOD and FlashVideo) mostly based on simple URL matching. The Tstat Streaming Classifier engine (2) classifies HTTP connections based on 2 distinguished approaches:

1. Value of Content-Type information in the HTTP's header
2. Signature matching in the video payload, to identify the video container.

log_http_complete:
Tstat can produce a `log_http_complete` file which logs information from every HTTP request and response. It is **NOT** generated by default, so it must be explicitly activated with the corresponding directive in `runtime.conf`.

log_mm_complete:
reports statistics for the RTP and RTCP flows. The classification process exploits a finite state machine that perform checks of version field, sequence numbers and payload types. In particupar, if in the *first UDP* packet

- the version field is set to 2
- the payload type field has an admissible values (for RTP or for RTCP)
- the UDP ports are larger than 1024 and are even/odd for RTP/RTCP

the flow is marked as possible RTP/RTCP flow (`FIRST_RTP/FIRST_RTCP`).
When the *second UDP* segment of this UDP flow (same IP/ports) is observed, then Tstat double checks if it still be interpreted as RTP/RTCP payload controlling if

- the version is equal to 2
- the same ssrc is present
- the seqno is the expected one

Navigation Shortcuts
Main
Overview
Web Interface
Gallery
Download
Archives
SVN
Available Traces
Skype
Instant Messaging
Multicast IP-TV
WeBrowse
Documentation
Measurement
Histograms
Logs
RRD interface
Publications
HOWTO
Useful Links
Contacts
Tstat Mailing List
M. Mellia
M. Munafò

- the payload type is the same as before.

If checks succedes, then the flows is marked as `RTP` and its analysis may start.

For RTCP flows, a simpler heuristic is used:

- the version must be equal to 2
- the payload type must be a correct one
- the UDP ports are larger than 1024 and are even/odd for RTP/RTCP.

If so, the flow is considered a `RTCP` flow and its analysis may start.

log_skype_complete:
reports statistics for each SKYPE flow identified using the methodology described into "Revealing skype traffic: when randomness plays with you". Note that records change according to the trasport layer (UDP or TCP) used by Skype.

log_chat_complete, **log_chat_messages**:
Tstat is able to classify MSN Messenger, Yahoo! Messenger and Chat based on XMPP Protocol such as Jabber or Google Talk. In `log_chat_complete` are reported statistics for each chat flow while in `log_chat_messages` for each chat message.

[Logs]

log_tcp_complete, log_tcp_nocomplete

Both files have similar format with values separated by spaces. Columns are grouped according to C2S - Client-to-Server and S2C - Server-to-Client traffic directions.
The log is made by the composition of different measurements sets, whose presence is controlled by variables in the `[options]` section of `runtime.conf`
The order of the sets is hardcoded, so the structure of `log_tcp_complete` is always:

- Core TCP set
- TCP End to End set (optional)
- TCP P2P set (optional)
- TCP Options set (optional)
- TCP Advanced set (optional)
- TCP Layer7 set (optional)

The structure of `log_tcp_nocomplete` is fixed and it will always only include the Core TCP set.
Here it follows a brief description of the columns for each set, considering that often the actual column number will depend on the mix of sets, and you should refer to the header line in the file to identify the current log content.
Note for advanced users: the current log composition is encoded as a 'magic number' in the first characters of the header line, that starts with "`#nn#`", '`nn`' being a bitmask indicating which set were active at the time of the logging. Interested users should refer to tstat.h for the bitmask values.

Core/Basic TCP Set

This set contains the basic information for all TCP flows, it is always included and cannot be disactivated in `runtime.conf`

C2S	S2C	Short description	Unit	Long description
1	15	Client/Server IP addr	-	IP addresses of the client/server
2	16	Client/Server TCP port	-	TCP port addresses for the client/server
3	17	packets	-	total number of packets observed form the client/server
4	18	RST sent	0/1	0 = no RST segment has been sent by the client/server
5	19	ACK sent	-	number of segments with the ACK field set to 1
6	20	PURE ACK sent	-	number of segments with ACK field set to 1 and no data
7	21	unique bytes	bytes	number of bytes sent in the payload
8	22	data pkts	-	number of segments with payload
9	23	data bytes	bytes	number of bytes transmitted in the payload, including retransmissions
10	24	rexmit pkts	-	number of retransmitted segments
11	25	rexmit bytes	bytes	number of retransmitted bytes
12	26	out seq pkts	-	number of segments observed out of sequence
13	27	SYN count	-	number of SYN segments observed (including rtx)
14	28	FIN count	-	number of FIN segments observed (including rtx)
29		First time abs	ms	Flow first packet absolute time (epoch)
30		Last time abs	ms	Flow last segment absolute time (epoch)
31		Completion time	ms	Flow duration since first packet to last packet
32		C first payload	ms	Client first segment with payload since the first flow segment
33		S first payload	ms	Server first segment with payload since the first flow segment
34		C last payload	ms	Client last segment with payload since the first flow segment

35	S last payload	ms	Server last segment with payload since the first flow segment
36	C first ack	ms	Client first ACK segment (without SYN) since the first flow segment
37	S first ack	ms	Server first ACK segment (without SYN) since the first flow segment
38	C Internal	0/1	1 = client has internal IP, 0 = client has external IP
39	S Internal	0/1	1 = server has internal IP, 0 = server has external IP
40	C anonymized	0/1	1 = client IP is CryptoPAn anonymized
41	S anonymized	0/1	1 = server IP is CryptoPAn anonymized
42	Connection type	-	Bitmap stating the connection type as identified by TCPL7 inspection engine (see protocol.h)
43	P2P type	-	Type of P2P protocol, as identified by the IPP2P engine (see ipp2p_tstat.h)
44	HTTP type	-	For HTTP flows, the identified Web2.0 content (see the http_content enum in struct.h)

TCP End to End Set

This set includes measures about RTT and TTL for TCP connections. It is enabled in `runtime.conf` setting "`tcplog_end_to_end = 1`".

C2S	S2C	Short description	Unit	Long description
45	52	Average rtt	ms	Average RTT computed measuring the time elapsed between the data segment and the corresponding ACK
46	53	rtt min	ms	Minimum RTT observed during connection lifetime
47	54	rtt max	ms	Maximum RTT observed during connection lifetime
48	55	Stdev rtt	ms	Standard deviation of the RTT
49	56	rtt count	-	Number of valid RTT observation
50	57	ttl_min	-	Minimum Time To Live
51	58	ttl_max	-	Maximum Time To Live

TCP P2P Set

This set includes P2P specific information (the general P2P type is included in the Core set). It is enabled in `runtime.conf` setting "`tcplog_p2p = 1`".
The P2P set will start at column Y (Y being 45 or 59, depending on the status of the E2E stat set). All the other columns will be relative to Y

C2S	S2C	Short description	Unit	Long description
Y		P2P subtype	-	P2P protocol message type, as identified by the IPP2P engine (see ipp2p_tstat.c)
Y+1		ED2K Data	-	For P2P ED2K flows, the number of data messages
Y+2		ED2K Signaling	-	For P2P ED2K flows, the number of signaling (not data) messages
Y+3		ED2K C2S	-	For P2P ED2K flows, the number of client<->server messages
Y+4		ED2K C2C	-	For P2P ED2K flows, the number of client<->client messages
Y+5		ED2K Chat	-	For P2P ED2K flows, the number of chat messages

TCP Options Set

This set includes specific TCP protocol statistics. It is enabled in `runtime.conf` setting "`tcplog_options = 1`".
The TCP options set will start at column X (X depending on the status of the E2E and P2P stat sets). All the other columns will be relative to X

C2S	S2C	Short description	Unit	Long description
X	X+23	RFC1323 ws	0/1	Window scale option sent
X+1	X+24	RFC1323 ts	0/1	Timestamp option sent
X+2	X+25	window scale	-	Scaling values negotiated [scale factor]
X+3	X+26	SACK req	0/1	SACK option set
X+4	X+27	SACK sent	-	number of SACK messages sent
X+5	X+28	MSS	bytes	MSS declared
X+6	X+29	max seg size	bytes	Maximum segment size observed
X+7	X+30	min seg size	bytes	Minimum segment size observed
X+8	X+31	win max	bytes	Maximum receiver window announced (already scale by the window scale factor)
X+9	X+32	win min	bytes	Maximum receiver windows announced (already scale by the window scale factor)

X+10	X+33	win zero	-	Total number of segments declaring zero as receiver window
X+11	X+34	cwin max	bytes	Maximum in-flight-size computed as the difference between the largest sequence number so far, and the corresponding last ACK message on the reverse path. It is an estimate of the congestion window
X+12	X+35	cwin min	bytes	Minimum in-flight-size
X+13	X+36	initial cwin	bytes	First in-flight size, or total number of unack-ed bytes sent before receiving the first ACK segment
X+14	X+37	rtx RTO	-	Number of retransmitted segments due to timeout expiration
X+15	X+38	rtx FR	-	Number of retransmitted segments due to Fast Retransmit (three dup-ack)
X+16	X+39	reordering	-	Number of packet reordering observed
X+17	X+40	net dup	-	Number of network duplicates observed
X+18	X+41	unknown	-	Number of segments not in sequence or duplicate which are not classified as specific events
X+19	X+42	flow control	-	Number of retransmitted segments to probe the receiver window
X+20	X+43	unnece rtx RTO	-	Number of unnecessary transmissions following a timeout expiration
X+21	X+44	unnece rtx FR	-	Number of unnecessary transmissions following a fast retransmit
X+22	X+45	!= SYN seqno	0/1	1 = retransmitted SYN segments have different initial seqno

TCP Layer 7 Set

This set includes Layer 7 and Application specific information (HTTP, TLS, DNS). It is enabled in `runtime.conf` setting "`tcplog_layer7 = 1`".
The Layer 7 set will start at column K (K depending on the status of the other stat sets). All the other columns will be relative to K. The Layer 7 set will always be printed as the last information in each row (also after the possible Advanced set).

C2S	S2C	Short description	Unit	Long description
K		HTTP Request count	-	Number of HTTP Requests (GET/POST/HEAD) seen in the C2S direction (for HTTP connections)
K+1		HTTP Response count	-	Number of HTTP Responses (HTTP) seen in the S2C direction (for HTTP connections)
K+2		First HTTP Response	-	First HTTP Response code seen in the server->client communication (for HTTP connections)
K+3		PSH-separated C2S	-	number of push separated messages C2S
K+4		PSH-separated S2C	-	number of push separated messages S2C
K+5		TLS Client Hello SNI	-	For TLS flows, the server name indicated by the client in the Hello message extensions
K+6		TLS Server Hello SCN	-	For TLS flows, the subject CN name indicated by the server in its certificate
K+7		TLS Client NPN/ALPN	-	For TLS flows, a bitmap representing the usage of NPN/ALPN for HTTP2/SPDY negotiation
K+8		TLS Server NPN/ALPN	-	For TLS flows, a bitmap representing the usage of NPN/ALPN for HTTP2/SPDY negotiation
K+9		TLS Client ID reuse	-	For TLS flows, indicates that the Client Hello carries an old Session ID
K+10		TLS Client Last Handshake	ms	For TLS flows, time of Client last packet seen before first Application Data (relative)
K+11		TLS Server Last Handshake	ms	For TLS flows, time of Server last packet seen before first Application Data (relative)
K+12		TLS Client App Data Time	ms	For TLS flows, time between the Client first Application Data message and the first flow segment
K+13		TLS Server App Data Time	ms	For TLS flows, time between the Server first Application Data message and the first flow segment
K+14		TLS Client App Data Bytes	bytes	For TLS flows, relative sequence number for the Client first Application Data message
K+15		TLS Server App Data Bytes	bytes	For TLS flows, relative sequence number for the Client first Application Data message
K+16		FQDN	-	Fully Qualified Domain Name recovered using DNHunter
K+17		IP of DNS resolver	-	IP address of the contacted DNS resolver
K+18		DNS request time	ms	unixtime (in ms) of the DNS request
K+19		DNS response time	ms	unixtime (in ms) of the DNS response

NPN (Next Protocol Negotiation) is the old TLS mechanism to negotiate the application layer to be used over TLS, identified by the code 0x3374 followed, possibly, by a list of supported protocols. It has been replaced by ALPN (Application-Layer Protocol Negotiation), RFC 7301, identified by the code 0x0010 followed by the explicit list of supported protocols. Both options can be used by the browser (client), while the server usually supports only one of them.
The NPN/ALPN bitmap possible values are the following:

```
0000 0000
^^^^ ^^^^
```

```
|||| ||||_____ NPN used, no explicit value negotiated (implicitely SPDY)
|||| |||______ NPN used, HTTP/1.x negotiated
|||| ||_______ NPN used, SPDY negotiated
|||| |________ NPN used, HTTP/2 negotiated
||||
||||__________ ALPN used, no explicit value negotiated
|||___________ ALPN used, HTTP/1.x negotiated
||____________ ALPN used, SPDY negotiated
|_____________ ALPN used, HTTP/2 negotiated
```

TCP Advanced Set

This set includes advanced information for the TCP flows, seldom used and that must be activated at compile time. It is enabled in `runtime.conf` setting "`tcplog_advanced = 1`".
Due to the dependency on compile time options, refer to the header row and the source code for the meaning of the additional columns.

Additional Tables

The values that can appear in columns 42, 43 and 44 of the TCP Core Set are reported below.

Connection Type (col. 42)
Since the connection type is a bitmask, it's possible to have combinated values. Common combinations are 257 (HTML and P2P), or 16640 (P2P and OBF).
P2P Type (col. 43)
Unless explicitely enabled at compilation time by `P2P_OLDPROTO`, the protocols marked by [*] are now considered obsolete and are ignored by Tstat 3.0.
HTTP Web2.0 Type (col. 44)
Due to the evolution of the Web, and the shift of traffic to TLS connections, some specific classifications have nowadays mostly a historical reason only.

Connection type - col.42 (see protocol.h)

Bitmask Value	Protocol
0	Unknown protocol
1	HTTP protocol
2	RTSP protocol
4	RTP protocol
8	ICY protocol
16	RTCP protocol
32	MSN protocol
64	YMSG protocol
128	XMPP protocol
256	P2P protocol
512	SKYPE protocol
1024	SMTP protocol
2048	POP3 protocol
4096	IMAP4 protocol
8192	TLS/TLS protocol
16384	ED2K protocol (obfuscated)
32768	SSH 2.0/1.99 protocol
65536	RTMP protocol
131072	Bittorrent MSE/PE protocol

P2P type - col. 43 (see ipp2p_tstat.h)

Bitmask n-th bit	Internal	Protocol
1	IPP2P_ED2K	eMule
2	IPP2P_DATA_KAZAA	Kazaa Data [*]
3	IPP2P_DATA_ED2K	Ed2k Data
4	IPP2P_DATA_DC	DirectConnect++ Data [*]
5	IPP2P_DC	DirectConnect++ [*]
6	IPP2P_DATA_GNU	Gnutella Data [*]
7	IPP2P_GNU	Gnutella [*]
8	IPP2P_KAZAA	Kazaa [*]
9	IPP2P_BIT	BitTorrent
10	IPP2P_APPLE	Apple [*]
11	IPP2P_SOUL	SoulSeek [*]
12	IPP2P_WINMX	WinMX [*]
13	IPP2P_ARES	Ares [*]
14	IPP2P_MUTE	Mute [*]
15	IPP2P_WASTE	Waste [*]
16	IPP2P_XDCC	XDCC [*]
17	IPP2P_KAD	eMule KAD
18	IPP2P_KADU	Adunanza (eMule mod)

HTTP Web2.0 type - col. 44 (see struct.h)

Type	Internal	Description
1	HTTP_GET	Unclassified GET command
2	HTTP_POST	Unclassified POST command
3	HTTP_MSN	MSN Chat command tunneled over HTTP (POST)
4	HTTP_RTMPT	RTMPT - RTMP over HTTP Tunnel (POST)
5	HTTP_YOUTUBE_VIDEO	YouTube video content download (GET)
6	HTTP_VIDEO_CONTENT	Generic FLV or MP4 video download (GET)
7	HTTP_VIMEO	Vimeo video content download (GET)
8	HTTP_WIKI	Wikipedia (GET)
9	HTTP_RAPIDSHARE	RapidShare file

These values are different from 0 only for identified HTTP connections (column no. 42). There constants are also used in the RRD data and in histograms (decreased by one so that HTTP_GET is 0 and HTTP_GMAPS is 13).

1) HTTP_VOD connection identification is experimental and not valid for usage outside Politecnico di Torino.

2) HTTP_SOCIAL is a set of matchings tailored for Nasza-Klasa (PL) and IWIW (HU). Since IWIW seems to be based on OpenSocial, it should match also generic OpenSocial traffic. Probably not useful outside Poland or Hungary.

3) HTTP_FLASHVIDEO identify traffic from a few popular flash-based video distribution sites.

		download (GET)
10	HTTP_MEGAUPLOAD	MegaUpload file download (GET)
11	HTTP_FACEBOOK	Facebook-related connections (GET/POST)
12	HTTP_ADV	Site advertisement (GET)
13	HTTP_FLICKR	Flickr photo download (GET)
14	HTTP_GMAPS	GoogleMaps images (GET)
15	HTTP_VOD	Video-on-Demand download (GET) [1]
16	HTTP_YOUTUBE_SITE	YouTube site content download (GET)
17	HTTP_SOCIAL	Localized social-networking (GET/POST) [2]
18	HTTP_FLASHVIDEO	Generic FLV video download (GET) [3]
19	HTTP_MEDIAFIRE	MediaFire file download (GET)
20	HTTP_HOTFILE	Hotfile.com file download (GET)
21	HTTP_STORAGE	Storage.to file download (GET)
22	HTTP_YOUTUBE_204	YouTube "pre-loading" (GET) [4]
23	HTTP_YOUTUBE_VIDEO204	YouTube "pre-loading" and video (GET) [4]
24	HTTP_YOUTUBE_SITE_DIRECT	YouTube: video request on YouTube site (GET) [5]
25	HTTP_YOUTUBE_SITE_EMBED	YouTube: embedded video request (GET) [5]
26	HTTP_TWITTER	Twitter unencrypted traffic (GET/POST) [6]
27	HTTP_DROPBOX	Dropbox presence traffic (GET) [7]

4) HTTP_YOUTUBE_204 and HTTP_YOUTUBE_VIDEO204 are counted as HTTP_YOUTUBE_VIDEO in RRDs and histograms (i.e. they are classified in idx4).

5) HTTP_YOUTUBE_SITE_DIRECT and HTTP_YOUTUBE_SITE_EMBED are counted as HTTP_YOUTUBE_SITE in RRDs and histograms (i.e. they are classified in idx15).

6) HTTP_TWITTER refers just to Twitter unencrypted connections, mostly related to the Twitter widgets in web pages. HTTP_TWITTER is counted as HTTP_SOCIAL and WEB_SOCIAL in RRDs and histograms (i.e. it is classified in idx16).

7) HTTP_DROPBOX refers to the presence/keep-alive connections maintained by the Dropbox client. Experimental. It is counted as HTTP_GET and WEB_OTHER in RRDs and histograms.

[Logs]

log_udp_complete

Columns are grouped according to C2S - Client-to-Server and S2C - Server-to-Client traffic directions and values are separated by spaces.

C2S	S2C	Short description	Unit	Long description
1	10	Client/Server IP addr	-	IP addresses of client/server
2	11	Client/Server UDP port	-	UDP port addresses of client/server
3	12	First time	ms	client/server first packet in absolute time (epoch)
4	13	Completion time	s	Time between the first and the last packet from the 'client'
5	14	Data bytes	bytes	Number of bytes transmitted in the payload
6	15	Packets	-	Total number of packets observed from the client/server
7	16	Internal	0/1	1 = IP address is internal
8	17	Anonymized	0/1	1 = IP address is CryptoPAn anonymized
9	18	UDP Type	-	Protocol type (see also the udp_type enum in struct.h)
19		FQDN	-	Fully Qualified Domain Name recovered using DNHunter

The values that can appear in columns 9 and 18 are reported below in the separate table. Unless explicitely enabled at compilation time by `P2P_OLDPROTO`, the protocols marked by [*] are now considered obsolete and are ignored by Tstat 3.0.

UDP type - col. 9, 18 (see struct.h)

Value	Internal	Description
0	UDP UNKNOWN	Unknown (unclassified)
1	FIRST_RTP	Unknown (possible unclassified RTP flow)
2	FIRST_RTCP	Unknown (possible unclassified RTCP flow)
3	RTP	RTP protocol

4	RTCP	RTCP protocol
5	SKYPE_E2E	Skype End-to-End
6	SKYPE_E2O	SkypeOut
7	SKYPE_SIG	Skype signalling
8	P2P_ED2K	eMule ED2K protocol
9	P2P_KAD	eMule KAD (Kamdelia) protocol
10	P2P_KADU	Adunanza (eMule mod) KAD (Kamdelia) protocol
11	P2P_GNU	Gnutella protocol [*]
12	P2P_BT	BitTorrent DHT protocol (only)
13	P2P_DC	DirectConnect protocol [*]
14	P2P_KAZAA	KaZaa protocol [*]
15	P2P_PPLIVE	PPLive IP-TV protocol
16	P2P_SOPCAST	SopCast IP-TV protocol
17	P2P_TVANTS	TV-Ants IPTV protocol
18	P2P_OKAD	eMule obfuscated KAD protocol
19	DNS	DNS protocol
20	P2P_UTP	BitTorrent uTP protocol (only)
21	P2P_UTPBT	BitTorrent DHT and uTP protocols (mixed)
22	UDP_VOD	MPEG2 PES Streaming over UDP
23	P2P_PPSTREAM	PPStream IP-TV protocol
24	TEREDO	Teredo IPv6 tunneling over UDP (mostly BitTorrent)
25	UDP_SIP	SIP over UDP messages
26	UDP_DTLS	DTLS protocol
27	UDP_QUIC	QUIC protocol

[Logs]

log_video_complete

The log contains a composition of measurement sets already included in the TCP `log_tcp_complete` log, and other sets specific to video streams. The presence of the various sets is controlled by variables in the `[options]` section of `runtime.conf`
The order of the sets is hardcoded, so the structure of `log_video_complete` is always:

- Core TCP set
- TCP End to End set (optional)
- Video Core set
- Video Information set (optional)
- YouTube Information set (optional)
- Video Advanced set (optional)
- TCP Options set (optional)
- TCP Layer7 set (optional)

Since YouTube now transfers a large part of traffic over TLS, we include in `log_video_complete` also those TLS connection for which the client requested (in the TLS handshake) a site name matching "`*.youtube.com`", "`*.youtube-nocookie.com`", and "`*.googlevideo.com`". For these connections, obviously, no video specific information can be reported. The requested site name is included in the log only if the TCP Layer 7 set is enabled (but the connections are always included).

Here it follows a brief description of the columns for each set, considering that often the actual column number will depend on the mix of sets, and you should refer to the header line in the file to identify the current log content. Moreover, for the description of TCP specific sets you can refer to the description for the `log_tcp_complete` file.
Note for advanced users: the current log composition is encoded as a 'magic number' in the first characters of the header line, that starts with "`#nn#`", '`nn`' being a bitmask indicating which set were active at the time of the logging. Interested users should refer to tstat.h for the bitmask values.

Core/Basic TCP/Video Set

This set contains the basic information for all TCP flows, it is always included and cannot be disactivated in `runtime.conf`. It is composed by the Core TCP set (the same reported in `log_tcp_complete`) and the Core Video information.
For the sake of information we replicate here the columns in the Core TCP set, pointing to the `log_tcp_complete` file description for further details.

C2S	S2C	Short description	Unit	Long description
1	15	Client/Server IP addr	-	IP addresses of the client/server
2	16	Client/Server TCP port	-	TCP port addresses for the client/server
3	17	packets	-	total number of packets observed form the client/server
4	18	RST sent	0/1	0 = no RST segment has been sent by the client/server
5	19	ACK sent	-	number of segments with the ACK field set to 1
6	20	PURE ACK sent	-	number of segments with ACK field set to 1 and no data

7	21	unique bytes	bytes	number of bytes sent in the payload
8	22	data pkts	-	number of segments with payload
9	23	data bytes	bytes	number of bytes transmitted in the payload, including retransmissions
10	24	rexmit pkts	-	number of retransmitted segments
11	25	rexmit bytes	bytes	number of retransmitted bytes
12	26	out seq pkts	-	number of segments observed out of sequence
13	27	SYN count	-	number of SYN segments observed (including rtx)
14	28	FIN count	-	number of FIN segments observed (including rtx)
29		First time abs	ms	Flow first packet absolute time (epoch)
30		Last time abs	ms	Flow last segment absolute time (epoch)
31		Completion time	ms	Flow duration since first packet to last packet
32		C first payload	ms	Client first segment with payload since the first flow segment
33		S first payload	ms	Server first segment with payload since the first flow segment
34		C last payload	ms	Client last segment with payload since the first flow segment
35		S last payload	ms	Server last segment with payload since the first flow segment
36		C first ack	ms	Client first ACK segment (without SYN) since the first flow segment
37		S first ack	ms	Server first ACK segment (without SYN) since the first flow segment
38		C Internal	0/1	1 = client has internal IP, 0 = client has external IP
39		S Internal	0/1	1 = server has internal IP, 0 = server has external IP
40		C anonymized	0/1	1 = client IP is CryptoPAn anonymized
41		S anonymized	0/1	1 = server IP is CryptoPAn anonymized
42		Connection type	-	Bitmap stating the connection type as identified by TCPL7 inspection engine (see protocol.h)
43		P2P type	-	Type of P2P protocol, as identified by the IPP2P engine (see ipp2p_tstat.h)
44		HTTP type	-	For HTTP flows, the identified Web2.0 content (see the http_content enum in struct.h)

For (partial) compatibility with the `log_tcp_complete` file structure, the position of the columns for Core Video measurements, will depend on the presence of other TCP measurements sets, even if they will always appear in the file.

The Core Video set will start at column X (X depending on the status of the TCP End to End stat set). All the other columns will be relative to X

C2S	S2C	Short desc.	Unit	Long description
X		Video Content-Type	-	The identified video format, based on the HTTP Content-Type information
X+1		Video Payload	-	The identified video format, based on the video payload information
X+2		Video ID16/46	-	16-char or 46-char YouTube video identifier, '--' otherwise
X+3		Video Format	-	YouTube Video Format code, '--' otherwise.

The YouTube Video Format is the 'fmt/itag' value as reported on Wikipedia.
The Video Content-Type and Video Payload, as identified by the Tstat Streaming Classifier engine, are summarized in the table below.

Video Formats

Value	VIDEO FORMAT	Description
0	NOT_DEFINED	Unclassified or not video
1	FLV	Adobe Flash Video container
2	MP4	MPEG-4 video, including F4V format and fragmented MP4[1]
3	AVI	AVI video format and DivX media format
4	WMV	Microsoft Media Video File (WMV) and ASF content
5	MPEG	MPEG-1, MPEG-2 and VOB video[2]
6	WEBM	Video format based on VP8 codec
7	3GPP	3rd Generation Partnership Project (3GPP). The releases 5 and 6 are classified as MP4
8	OGG	Ogg Vorbis Codec compressed Multimedia file
9	QUICKTIME	Video exported with QuickTime Apple Inc software[3]
10	ASF	ASF control packets (ASF video are generally classified as WMV)
12	HLS	HTTP Live Streaming
13	NFF	NDS File Format (Cisco Videoscape), used by Sky+ boxes for VOD as 'video/nff'
11	UNKNOWN	Other videos formats or Content-Type values like 'video/*'

These values are different from 0 only for identified HTTP connections (column no. 42). There constants are also used in the RRD data and in histograms.

1) F4V and FLV differences are summarized here.

2) The signatures for MPEG encoded videos are based on the rules described here.

3) The classification relays only on the Content-Type value announce by the server. Currently the payload matching is not supported for this video format.

Video TCP End to End Set

This set includes measures about RTT and TTL for TCP connections and it's identical to the End to End set in `log_tcp_complete`. It is enabled in `runtime.conf` setting "`videolog_end_to_end = 1`".
Refer to the `log_tcp_complete` description for details.

Video Information Set

This set includes information about video duration and resolution, as extracted by the video header metadata by the Video Streaming engine. It is enabled in `runtime.conf` setting "`videolog_videoinfo = 1`".
The Video Information set will start at column X (X depending on the status of other video stat set). All the other columns will be relative to X

C2S	S2C	Short desc.	Unit	Long description
X		Video duration	s	Video duration as indicated in the payload
X+1		Video total datarate	kbps	Total data rate as indicated in payload
X+2		Video width	pixel	Video width as indicated in the payload
X+3		Video height	pixel	Video heigth as indicated in the payload

Note: These values can be reported only if the information is contained in the frame header, and are mostly available only for FLV and MP4 formats. Video Total Datarate is not reported for AVI format.

YouTube Information Set

This set includes additional information about YouTube video and connections, as extracted by the URLs by the Tstat HTTP engine. It is enabled in `runtime.conf` setting "`videolog_youtube = 1`".
Basic YouTube info (video ID and video format) is included in the Core Video set.
The YouTube Information set will start at column X (X depending on the status of other video stat set). All the other columns will be relative to X

C2S	S2C	Short desc.	Unit	Long description
X		Video ID11	-	11-char YouTube video request ID if YOUTUBE_REQUEST_ID is defined, '--' otherwise
X+1		Begin Offset	ms	Playback offset for the Youtube video, 0 otherwise
X+2		Redirect	-	Request has been redirected
X+3		Redir Count	-	Redirection counter
X+4		Mobile Device	-	Type of mobile device 0=Desktop 1=Other 2=Apple iOS 3=Android
X+5		Streaming	-	Video streaming classification

Redirect and Redir Count are based on the parameters `redirect_count` provided in the videodownload URL.
The Video Streaming Classification refers to the different streaming techniques used by YouTube, identified by the different URL formats, and summarized below.

ID	Comment
0	No streaming. Progressive or adaptive download (DASH)
1	HLS streaming of recorded content
2	HLS streaming of live content
3	Advertisement
4	HLS streaming of live content (different from 2)
5	Live FLV streaming

Video TCP Options Set

This set includes specific TCP protocol statistics and it's identical to the TCP Options set in `log_tcp_complete`. It is enabled in `runtime.conf` setting "`videolog_options = 1`".
Refer to the `log_tcp_complete` description for details.

Video TCP Layer 7 Set

This set includes TCP Layer7 and Application specific information (HTTP, TLS) and it's identical to the TCP Layer7 set in `log_tcp_complete`. It is enabled in `runtime.conf` setting "`videolog_layer7 = 1`".
TLS information is relevant only for YouTube-related TCP flows. Moreover, as in `log_tcp_complete`, the Video TCP Layer 7 set will always be printed as the last information in each row.
Refer to the `log_tcp_complete` description for details.

Video Advanced Set

This set includes advance information for the Video flows, mostly related to the video transfer bitrate. It is enabled in `runtime.conf` setting "`videolog_advanced = 1`".
The Video Advanced set will start at column X (X depending on the status of other video stat set). All the other columns

will be relative to X

C2S	S2C	Short desc.	Unit	Long description
X	X+7	Rate Samples	-	Number of samples C2S/S2C in the rate measurement
X+1	X+8	Zero Samples	-	Number of empty samples C2S/S2C in the rate measurement
X+2	X+9	Zero Streak	-	Maximum number of consecutive C2S/S2C empty samples
X+3	X+10	Average rate	kbps	Average rate in the C2S/S2C direction
X+4	X+11	Stdev rate	kbps	Standard deviation rate in the C2S/S2C direction
X+5	X+12	min rate	-	Minimum (non zero) rate sample
X+6	X+13	max rate	-	Maximum rate sample
X+14 - X+23		Rate samples	bytes	Bytes in the first 10 rate sampling slots C2S
X+24 - X+33		Rate samples	bytes	Bytes in the first 10 rate sampling slots S2C
X+34		PSH-Messages	-	Number of PSH-separated 'messages' in the S2C flow
X+35 - X+44		PSH-Messages size	bytes	Bytes in the first 10 PSH-separated 'messages in the S2C flow

Columns X+(14/33) refer to the estimation of the initial flow bitrate obtained by counting the amount of bytes transfered in both directions in the first 10 measurements intervals (10 seconds) (part of the information summarized in previous columns X/X+13). Columns X+(35/44) estimate the structure of the S2C streaming flow, counting the blocks of data (as PSH-separated messages) and the dimension of the first 10 measured blocks.

[Logs]

log_http_complete

The `log_http_complete` file differs from the other Tstat `log_*` logs for two key features:

- fields in each row are **TAB separated** and not space separated (this because some fields will contain spaces)
- C2S and S2C rows are alternated, have different information, and a different number of fields.

Here it follows a brief description of the columns.

C2S	S2C	Short description	Unit	Long description
1	1	Client IP addr	-	IP addresses of the client (sending the request/receiving the response)
2	2	Client TCP port	-	TCP port addresses for the client
3	3	Server IP addr	-	IP addresses of the server (receiving the request/sending the response)
4	4	Client TCP port	-	TCP port addresses for the server
5	5	Segment time abs	s	Absolute time [s] (epoch) of the request/response
6		Request method	-	Request method (GET/POST/HEAD) [*]
7		Hostname	-	Value of the "Host:" HTTP request field
8		FQDN	-	DN-Hunter cached DNS name [^]
9		URL Path	-	URL request path
10		Referer	-	Value of the "Referer:" HTTP request field
11		User agent	-	Value of the "User-Agent:" HTTP request field
	6	Response string	-	Response identifier (always "HTTP") [*]
	7	Response code	-	HTTP response code (2xx/3xx/4xx/5xx)
	8	Content len	bytes	Value of the "Content-Length:" HTTP response field
	9	Content type	-	Value of the "Content-Type:" HTTP response field
	10	Server	-	Value of the "Server:" HTTP response field
	11	Range	-	Value of the "Content-Range:" HTTP response field for partial content (Code 206)
	12	Location	-	Value of the "Location:" HTTP response field for redirected content (Code 302)

[*] Column 6 can be used to distinguish between S2C Responses (for which there is always the string "HTTP"), and C2S Requests (identified by method GET/POST/HEAD)
[^] The DN-Hunter FQDN is included only if DN-Hunter has been enabled at compilation time

Privacy Concerns
To reduce possible privacy issues, the fields "URL Path" (col. 9 C2S), "Referer" (col. 10 C2S), and "Location" (col. 12 S2C) are subject to the `http_full_url` parameter in `runtime.conf`.
By default `"http_full_url = 0"`, the "URL Path" is always empty/invalid (`"-"`), and only the base HTTP and HTTPS URLs (scheme+path) are registered in the "Referer" and "Location" fields. URLs with different schemes are always removed.
Using `"http_full_url = 1"` the URL fields are truncated to the actual path part of the URL, removing any parameter after the `'?'` character.
To record the complete fields, the directive `"http_full_url = 2"` must be used in `runtime.conf`. When Tstat is run using the `-0` command line option ("strict(er) privacy"), `"http_full_url = 0"` is enforced and the value cannot be overridden by the value in `runtime.conf`.

[Logs]

log_mm_complete

C2S	S2C	Short Description	Unit	Long Description	Protocol
1		L4 Proto	1/2	1 = TCP, 2 = UDP	All
2	38	Protocol	3/4	3 = RTP, 4 = RTCP	All
3	39	IP address	-	Client/Server IP addresses	All
4	40	L4 port	-	TCP/UDP port addresses for the Client/Server	All
5	41	Internal	0/1	1 = internal ip	All
6	42	Packets	-	Number of packets Tstat has seen belonging to the flow	All
7	43	IPG	ms	Inter Packet Gap (IPG)	All
8	44	Jitter AVG	ms/ts	Jitter (average): - if RTP, computed by Tstat as in RFC3550 [ms] - if RTCP, extracted from the RTCP header [codec timestamps units]; - if TCP, computed using only data packets [ms]	All
9	45	Jitter Max	ms/ts	Jitter (max) - if RTP, computed by Tstat as in RFC3550 [ms] - if RTCP, extracted from the RTCP header [codec timestamps units] - if TCP, computed using only data packets [ms]	All
10	46	Jitter Min	ms/ts	Jitter (min) - if RTP, computed by Tstat as in RFC3550 [ms] - if RTCP, extracted from the RTCP header [codec timestamps units] - if TCP, computed using only data packets [ms]	All
11	47	TTL AVG	-	Time to live (TTL) (average)	All
12	48	TTL Max	-	Time to live (TTL) (max)	All
13	49	TTL Min	-	Time to live (TTL) (min)	All
14	50	Start	s	Start time	All
15	51	Duration	s	Duration	All
16	52	Data	bytes	Data transfered	All
17	53	Bitrate	bit/s	Average speed [bit/s]	All
18	54	SSRC	-	SSRC	RTP, RTCP
19	55	Lost pkts	-	Lost packets, computed by Tstat using a window based algorithm	RTP
20	56	Out of seq. pkts	-	Out of sequence packets computed by Tstat computed by Tstat using a window based algorithm	TCP,RTP
21	57	Dup pkts	-	Duplicate packets computed by Tstat - if RTP, computed by Tstat using a window based algorithm - if TCP, computed as retrasmissions	TCP,RTP
22	58	Late pkts	-	Late packets computed by Tstat computed by Tstat using a window based algorithm	RTP
23	59	RTP type	-	RTP payload type	RTP
24	60	Reset	-	Bogus reset	RTP
25	61	Cum lost pkts	-	Cumulative packet loss: - each lost packets increments this counter, - each duplicated packets decremnets it from RTCP	RTCP
26	62	Frac lost pkts	-	Extracted from the RTCP header [%]	RTCP
27	63	Flow length	-	Associated RTP flow length	RTCP
28	64	Flow length	bytes	Associated RTP flow length	RTCP
29	65	RTT AVG	ms	Round Trip Time (RTT) (average)	TCP, RTCP
30	66	RTT Max	ms	Round Trip Time (RTT) (max)	TCP, RTCP
31	67	RTT Min	ms	Round Trip Time (RTT) (min)	TCP, RTCP
32	68	RTT	ms	Round Trip Time (RTT) (samples)	TCP, RTCP
33	69	Truncated RTCP header	-	Truncated RTCP header	RTCP
34	70	First HTTP	s	First HTTP packet	TCP
35	71	First RTSP	s	First RTSP packet	TCP
36	72	FIRST RTP	s	First RTP packet	TCP
37	73	FIRST ICY	s	First ICY packet	TCP

[Logs]

log_skype_complete (TCP)

C2S	S2C	Short Description	Unit	Long Description
		Client/Server		Client IP

log_skype_complete (UDP)

C2S	S2C	Short description	Unit	Long description
1	24	Client/Server	-	IP address of the

1	17	IP address	-	address
2	18	Client/Server TCP Port	-	Client TCP port
3	19	Internal	0/1	1 = internal IP address
4	20	Flow Size	bytes	Flow Size
5	21	Total packets	-	No. of Total flow packets
6	22	Audio/video pkts	-	No. of audio or audio+video packets
7	23	Video only pkts	-	No. of video only packets
8	24	Avg Pktsize	-	Average Packet size
9	25	Avg Pktsize: MMB	-	Average Packet Size: Max Mean Belief
10	26	Avg IPG	-	Average Inter-packet Gap
11	27	Avg IPG: MMB	-	Average IPG: Max Mean Belief
12	28	CHI HDR max	-	Chi-square on Header: max value
13	29	CHI PAY max	-	Chi-square on Payload: max value
14	30	BFT	-	Bayesian Flow Type
15	31	CSFT	-	Chi-square Flow Type
16	32	Video present	0/1	1 = Video is present
33		Start Time	s	Flow Start Time
34		Elapsed Time	s	Flow Elapsed Time
35		L4 proto	'U'	Label to state a UDP flow

		IP addr		'client'
2	25	Client/Server port	-	TCP/UDP port address for the 'client'
3	26	Internal	0/1	1 = internal IP address
4	27	Flow Size	bytes	Flow Size
5	28	Total packets	-	No. of Total flow packets
6	29	E2E packets	-	No. of End-to-End packets
7	30	E2O packets	-	No. of SkypeOut packets
8	31	SIG packets	-	No. of Signaling packets
9	32	UNK packets	-	No. of Unknown packets
10	33	Audio/Video pkts	-	No. of audio or audio+video packets
11	34	Video only pkts	-	No. of video only packets
12	35	Avg Pktsize	-	Average Packet size
13	36	Avg Pktsize: MMB	-	Average Packet Size: Max Mean Belief
14	37	Avg IPG	ms	Average Inter-packet Gap
15	38	Avg IPG: MMB	-	Average IPG: Max Mean Belief
16	39	CHI HDR min	-	Chi-square on Header: min value
17	40	CHI HDR max	-	Chi-square on Header: max value of {1-4} & {7,8} blocks
18	41	CHI HDR min 5,6	-	Chi-square on Header: min value of {5,6} blocks
19	42	CHI PAY max	-	Chi-square on Payload: max value
20	43	DFT	-	Deterministic Flow Type
21	44	BFT	-	Bayesian Flow Type
22	45	CSFT	-	Chi-square Flow Type
23	46	Video present	0/1	1 = Video is present
47		Start Time	s	Flow Start Time (epoch)
48		Elapsed Time	s	Flow Elapsed Time
49		L4 proto	'T'	Label to state a TCP flow

[Logs]

log_chat_complete

C2S	S2C	Short description	Unit	Long description
1	11	Client/Server IP addr	-	IP address of client/server
2	12	Client/Server port	-	TCP port address of client/server
3	13	Flow Size	bytes	Flow Size [Bytes]
4	14	Total packets	-	No. of Total flow packets
				No. of Total

Chat Flow Type - col. 23

Value	Description	IM Protocols
0	Unknown	All
1	Login	All
2	Presence	All
3	Chat	All
4	Presence+Chat	Yahoo only
5	Http Tunneling	MSN only
6	Peer-to-Peer Chat (i.e. direct connection	Yahoo only

5	15	Total messages	-	messages sent by client
6	16	MSG_A	-	No. of MSG_A sent by client [for MSN only, 0 for the others]
7	17	MSG_D	-	No. of MSG_D sent by client [for MSN only, 0 for the others]
8	18	MSG_N	-	No. of MSG_N sent by client [for MSN only, 0 for the others]
9	19	MSG_U	-	No. of MSG_U sent by client [for MSN only, 0 for the others]
10	20	MSG_Y	-	No. of MSG_Y sent by client [for MSN only, 0 for the others]
21		Start Time	s	Flow Start Time
22		End Time	s	Flow End Time
23		Chat Flow Type	-	Chat Flow Type
24		Chat Version	-	Version of the protocol used by the Instant Messaging application
25		Internal	0/1	1 = internal IP address
26		TCP Flow No.	-	TCP Flow ID Number
27		'T'	-	Label to state a TCP Flow
28		Chat Protocol	32=MSN 64=Yahoo 128=Jabber/GTalk	Type of Upper Level Protocol
29		C anonymized	0/1	1 = client IP is CryptoPAn anonymized
30		S anonymized	0/1	1 = server IP is CryptoPAn anonymized

	between clients)	
7	Unclassified Yahoo Messenger flow	Yahoo only

log_chat_messages

Col.no.	Short descr	Long description
1	TCP Flow No.	TCP Flow ID Number
2	Message type	Type of Message (? if not available)
3	Dir	TCP Flow Direction (1=C2S, -1=S2C)
4	Message size	Message Payload Size [Bytes] (? if not available)
5	Payload size	TCP Payload Size [Bytes]
6	Start Time	Flow Start Time [in Unix Epoch Time]
7	Arrival Time	Message Arrival Time [s]

Histogram description

An Histogram represents the empirical distribution of a specific index considering a fixed measurement period. For each measured index, Tstat creates and updates an histogram that collects the hit number of that quantity. For examples, considering the IP packet length, Tstat updates, for each observed IP packet, the counter of the number of observed packets with a particular length. At the end of the measurement period, Tstat saves each

histogram in a separate `TXT` file, reset all the values and then restarts to collect samples. The duration of a measurement period is defined by the **`MAX_TIME_STEP`** parameter, which is defined in the file `param.h`, and by default, it is set to **5 minutes**.

Recalling that (see HOWTO) Tstat is able to distinguish between **IN**-coming, **OUT**-going and **LOC**-al traffic and among **C2S** - Client-to-Server and **S2C** - Server-to-Client, it follows that, when applicable, it generates histograms according to traffic directions. Histograms names are strictly related both to the direction and the type of measure and as to have a quick remainder of the supported indexes it can be used:

```
bash> tstat -H ?
#name              min  bin_size     max  description
profile_flows      0    1            5    flows handled
profile_cpu        0    1            4    cpu load [clock/time]
chat_flow_num      0    1            7    Number of tracked IM flow
web_bitrate_loc    0    1            7    Web 2.0 content bitrate [bit/s] - local segments
web_bitrate_out    0    1            7    Web 2.0 content bitrate [bit/s] - outgoing segments
web_bitrate_in     0    1            7    Web 2.0 content bitrate [bit/s] - incoming segments
L7_WEB_num_loc     0    1            7    Number of tracked Web 2.0 flows - local flows
...
```

The following tables report a verbose description of all the supported histograms grouped as:

- IP Layer: statistics related to ip addresses and IP protocol;
- **TCP Segments**: statistics related to individual TCP segments;
- **TCP Flows**: statistics related to TCP flows;
- **UDP Layer**: statistics related to UDP flows;
- **Streaming Flows**: statistics related to streaming flows;
- **RTCP Flows**: statistics related to RTCP protocol;
- **HTTP Flows**: statistics related to HTTP protocol;
- **Profile**: profiling of the machine running Tstat;

[Histograms]

IP Layer

Name	Direction	Min	Bin Size	Max	Unit	Description
ip_tos	loc,out,in	0	1	255	-	IP TOS field
ip_ttl	loc,out,in	0	1	255	-	IP TTL field
ip_len	loc,out,in	0	4	1500	byte	IP packet length
ip_bitrate	loc,out,in	0	1	4	kb/s	IP bitrate
ip_protocol	loc,out,in	0	1	255	-	IP protocol
addresses	-	-	-	-	-	This file collects the number of packets originated/destined to a particular IP subnet. By default, Tstat considers /24 subnets, and counts how many packets have been sent/received having a particular IP subnet source/destination address. The format of this histogram is different from the others, has it stores in the first column the subnet address, in the second column the number of packets whose IP source in the subnet, and in the third column the number of packets whose IP destination is in the subnet. No particular order is applied when saving the histogram, so that sorting is left to the user.

[Histograms]

TCP Segments

Name	Direction	Min	Bin Size	Max	Unit	Description
tcp_mss_used	-	0	4	1600	-	Negotiated TCP MSS: minimum between MSS declared by the server and the client
tcp_mss_b	-	0	4	1600	-	Server TCP MSS declared
tcp_mss_a	-	0	4	1600	-	Client TCP MSS declared
tcp_opts_TS	-	1	1	4	-	TCP option: Timestamp. 1 = ok, 2 = only client offered, 3 = only server offered, 4 = none offered
tcp_opts_WS	-	1	1	4	-	TCP option: WindowScale. 1 = ok, 2 = only client offered, 3 = only server offered, 4 = none offered
tcp_opts_SACK	-	1	1	4	-	TCP option: SACK. 1 = ok, 2 = only client offered, 3 = only server offered, 4 = none offered
tcp_bitrate	loc,out,in	0	1	29	bit/s	TCP application bitrate
tcp_port_syndst	loc,out,in	0	1	65536	-	TCP destination port of SYN segments only
tcp_port_synsrc	loc,out,in	0	1	65536	-	TCP source port of SYN segments only
tcp_port_dst	loc,out,in	0	1	65536	-	TCP destination port
tcp_port_src	loc,out,in	0	1	65536	-	TCP source port

[Histograms]

TCP Flows

Name	Direction	Min	Bin Size	Max	Unit	Description
tcp_interrupted	-	0	1	1	-	TCP Early interrupted flows. A flow is considered early interrupted according to the rules identified in: Rossi D., Casetti C. and Mellia M., *"User Patience and the Web: a hands-on investigation"*, IEEE Globecom 2003, San Francisco, CA, USA, December 1-5, 2003.
tcp_thru	c2s,s2c	0	1	1000	kb/s	TCP application throughput. The throughput is defined as the ratio between the data sent by the server/client over the time since the first SYN segment up to the last segment carrying

						data from the server/client, i.e., no TCP tear-down latency is included.
tcp_tot_time	-	0	50	720000	ms	TCP flow lifetime, i.e., the time since the first ever seen SYN segment up to the very last segment of this flow.
tcp_anomalies	s2c,c2s,loc,out,in	0	1	64	-	TCP total number of anomalies per each flow. TCP anomalies are identified according to the algorithm described in Mellia M., Meo M. and Muscariello L., *"TCP Anomalies: identification and analysis"*, Tyrrhenian International Workshop on Digital Communications Sorrento, July 4-6.
tcp_rtx_RTO	s2c,c2s,loc,out,in	0	1	100	-	TCP anomaly: Number of RTO Retransmission
tcp_rtx_FR	s2c,c2s,loc,out,in	0	1	100	-	TCP anomaly: number of FR Retransmission
tcp_flow_ctrl	s2c,c2s,loc,out]	0	1	100	-	TCP anomaly: number of Flow Control
tcp_flow_control_in	-	0	1	100	-	TCP anomaly: number of Flow Control - incoming flows
tcp_net_dup	s2c,c2s,loc,out,in	0	1	100	-	TCP anomaly: number of Network duplicates
tcp_reordering	s2c,c2s,loc,out,in	0	1	100	-	TCP anomaly: number of packet reordering
tcp_unnrtx_FR	s2c,c2s,loc,out	0	1	100	-	TCP anomaly: number of Unneeded FR retransmission
tcp_unnecessary_rtx_FR_in	-	0	1	100	-	TCP anomaly: number of Unneeded FR retransmission - incoming flows
tcp_unnrtx_RTO	s2c,c2s,loc,out	0	1	100	-	TCP anomaly: number of Unneeded RTO retransmission
tcp_unnecessary_rtx_RTO_in	-	0	1	100	-	TCP anomaly: number of Unneeded RTO retransmission - incoming flows
tcp_unknown	s2c,c2s,loc,out,in	0	1	100	-	TCP anomaly: number of unknown anomalies
tcp_rtt_cnt	s2c,c2s,loc,out,in	0	1	200	-	TCP flow RTT: number of valid valid samples
tcp_rtt_stdev	s2c,c2s,loc,out,in	0	10	3500	ms	TCP flow RTT: standard deviation
tcp_rtt_max	s2c,c2s,loc,out,in	0	10	3500	ms	TCP flow RTT: maximum RTT

tcp_rtt_avg	s2c,c2s,loc,out,in	0	10	3500	ms	TCP flow RTT: average RTT
tcp_rtt_min	s2c,c2s,loc,out,in	0	10	3500	ms	TCP flow RTT: minimum RTT
tcp_cl_b_l	s2c,c2s,loc,out,in	0	50000	50000000	byte	TCP flow length - coarse granularity histogram
tcp_cl_b_s	s2c,c2s,loc,out,in	0	50	50000	byte	TCP flow length - fine granularity histogram
tcp_cl_p	s2c,c2s,loc,out,in	0	1	1000	packet	TCP flow length
tcp_cwnd	-	0	256	65536	byte	TCP in-flight-size: the difference among the highest sequence number and the highest acknowledgment number on the reverse path seen when a new ACK is received.
tcp_win_max	-	0	256	65536	byte	TCP max RWND: the maximum RWND (eventually scaled by the WS option observed during flow lifetime. Only RWND values sent by the client are considered.
tcp_win_avg	-	0	256	65536	byte	TCP average RWND: the average RWND (eventually scaled by the WS option observed during flow lifetime. Only RWND values sent by the client are considered.
tcp_win_ini	-	0	256	65536	byte	TCP initial RWND: the first ever observed value of the RWND (eventually scaled by the WS option)

[Histograms]

UDP Layer

Name	Direction	Min	Bin Size	Max	Unit	Description
udp_port_flow_dst	-	0	1	65536	-	UDP destination port per flow
udp_port_dst	loc,in,out	0	1	65536	-	UDP destination port per segment
udp_tot_time	-	0	50	720000	ms	UDP flow lifetime: time since the first segment ever observed to the last observed segment
udp_cl_b_l	loc,in,out	0	50000	50000000	byte	UDP flow length - coarse granularity histogram
udp_cl_b_s	loc,in,out	0	50	50000	byte	UDP flow length - fine granularity histogram
udp_cl_p	loc,in,out	0	1	1000	packet	UDP flow length
udp_bitrate	loc,in,out	0	1	50	bit/s	UDP application bitrate

[Histograms]

Streaming Flows

Name	Direction	Min	Bin Size	Max	Unit	Description
						Stream burst length of lost packets: number of missing packets with continuous sequence number
mm_burst_loss	loc,out,in	0	1	20	packet	
mm_p_late	loc,out,in	0	1	1000	-	Stream prob of late packets per flow: ratio between the number of packet arrived with a delay larger than 20 sequence number (i.e., packet 32 arrived when expecting packet 55) and the total number of flow packets.
mm_p_lost	loc,out,in	0	1	1000	-	Stream prob of lost packets per flow: ratio between the number of missing segments over the flow total number of segments
mm_p_dup	loc,out,in	0	1	1000	-	Stream prob of duplicate packets per flow: ratio between the number of duplicated segments over the total flow number of segments
mm_p_oos	loc,out,in	0	1	1000	-	Stream prob of out-of-sequence packets per flow: ratio between the number of out-of-sequence segments over the flow total number of segments
mm_n_oos	loc,out,in	0	1	100	-	Stream number of out-of-sequence packets per flow: total number of out-of-sequence segments (any segment whose seqno is not the largest ever seen plus 1) observed in the whole flow life
mm_oos_p	loc,out,in	0	1	0	-	Total stream number of out of sequence packets
mm_reord_p_n	loc,out,in	0	1	0	-	Total stream number of reordered packets observed in during the time intervals
mm_reord_delay	loc,out,in	0	1	100	-	Stream delay of reordered packets: time elapsed since the reception of the out-of-sequence packet and its immediate predecessor
mm_avg_jitter	loc,out,in	0	1	5000	0.1m	Stream average jitter per flow
mm_avg_ipg	loc,out,in	0	1	5000	0.1m	Stream average IPG per flow
mm_avg_bitrate	loc,out,in	0	10	10000	kb/s	Stream bitrate

mm_cl_b	loc,out,in	0	50000	100000000	byte	Long stream flow length
mm_cl_p	loc,out,in	0	10	50000	packet	Long stream flow length
mm_cl_b_s	loc,out,in	0	100	100000	byte	Short stream flow length
mm_cl_p_s	loc,out,in	0	1	1000	packet	Short stream flow length
mm_tot_time_s	loc,out,in	0	1	5000	ms	Short stream flow lifetime
mm_tot_time	loc,out,in	0	1	5400	s	Stream flow lifetime
mm_rtp_pt	loc,out,in	0	1	128	-	RTP payload type
mm_uni_multi	loc,out,in	0	1	1	-	Unicast/multicast flows
mm_type	loc,out,in	0	1	8	-	Stream type

[Histograms]

RTCP Flows

Name	Direction	Min	Bin Size	Max	Unit	Description
rtcp_bt	loc,out,in	0	10	10000	bit/s	RTCP average bitrate
rtcp_mm_bt	loc,out,in	0	1	5000	kb/s	RTCP associated MM flow average bitrate during interval
rtcp_mm_cl_b	loc,out,in	0	50000	100000000	byte	RTCP associated MM flow length
rtcp_mm_cl_p	loc,out,in	0	10	50000	packets	RTCP associated MM flow length
rtcp_t_lost	loc,out,in	0	10	10000	-	RTCP lost packets per flow
rtcp_f_lost	loc,out,in	0	1	1000	-	RTCP fraction of lost packets during interval
rtcp_dup	loc,out,in	0	1	1000	-	RTCP duplicated packets during interval
rtcp_lost	loc,out,in	0	1	1000	-	RTCP lost packets during interval
rtcp_jitter	loc,out,in	0	1	1000	-	RTCP jitter during interval
rtcp_rtt	loc,out,in	0	1	3000	ms	RTCP round trip time
rtcp_avg_inter	loc,out,in	0	1	5000	-	RTCP interarrival delay
rtcp_cl_b	loc,out,in	0	1	3000	byte	RTCP flow length
rtcp_cl_p	loc,out,in	0	1	3000	packet	RTCP flow length

[Histograms]

HTTP Flows

Name	Direction	Min	Bin Size	Max	Unit	Description
http_bitrate	loc,in,out	0	1	21	bit/s	HTTP content bitrate
web_bitrate	loc,in,out	0	1	7	bit/s	Web2.0 content bitrate
L7_HTTP_num	loc,in,out	0	1	21	-	Number of tracked HTTP flows
L7_WEB_num	loc,in,out	0	1	7	-	Number of tracked Web2.0 flows

[Histograms]

Profile

Name	Direction	Min	Bin Size	Max	Unit	Description
profile_flow	-	0	1	5	-	Flows handled

profile_cpu	-	0	1	4	-	CPU load (clock/time)

-->