Multimedia and streaming traffic analysis

Simone Arena*, Robert Birke†, Marco Mellia†, Michele Petracca†, Christian Racca*, Dario Rossi†
* TOPIX, Via Bogino 9, 10123 Torino, Italy, Email: {christian.racca,simone.arena}@topix.it
† Dipartimento di Elettronica, Politecnico di Torino, 10129 Torino, Italy, Email: {birke,mellia,petracca,rossi}@tlc.polito.it

***Abstract*— Network monitoring is becoming more and more appealing to Network Operators to keep track of the offered quality especially related to multimedia and streaming traffic. This paper presents an enhanced version of Tstat which is able to identify multimedia streaming flows. After a brief introduction on Tstat, section II shows a brief analysis of multimedia traffic to identify the streaming protocols mostly in use, while sections III and IV discuss on the heuristics implemented to identify these protocols. Section V presents the possible metrics. Finally the last sections present some real world measurements.**

I. Introduction

The monitoring of telecommunication networks is a problem becoming more and more interesting for network operators. Particularly, the evaluation of network performances offered to the users is used to quantify the QoS (Quality of Service) and to plan the development of the network. In IP networks the "best effort" policy, makes the monitoring of QoS based on passive traffic measurements particularly difficult.

The TNG (Telecommunication Network Group) of Electronic Department of Politecnico di Torino developed Tstat (TCP STatistic and Analysis Tool), a monitoring and performance analysis tool for IP telecommunication networks. Observing passively the traffic on a network link, Tstat is able to give a set of IP level performance indexes together with an analysis of the TCP [1] flows. Furthermore Tstat has been enhanced to be able to monitor RTP/RTCP [2] connections, carried both over UDP [3] or tunneled over TCP, giving statistics about the QoS seen by the user of these streaming or realtime flows.

The IP level measurements are performed observing the IP header field of each IP packet. All measurements allow to compute different distributions of traffic: e.g. the kind of transport layer protocol used, the length of the packets, the number of packets with the same destination IP address and so on. TCP level measurements are based on the hypothesis to observe data segments (from server to client) and confirmation segments (ACKs from client to server). The availability of both segment flows makes Tstat able to reconstruct the state of the TCP transceiver to extract performance indexes. Possible measurements are, for example, the data amount transferred during each connection and its duration, allowing to compute the mean throughput seen by the user. Tstat is also able to track special events like segment retransmissions, duplications and out of sequences, typical of the TCP behavior, allowing to calculate the probabilities of each such event. For multimedia flows Tstat is able to analyse their evolution and extract important measurements like delay, jitter and packet loss probability. These measures are extremely relevant for QoS since they impact heavily on the quality of the streaming or realtime flow.

II. Traffic of interest

The project focuses on the analysis of streaming protocols. The traffic over the output link of the Top-IX streaming farm is screened to evaluate the characteristics of traffic generated by streaming applications to find a set of metrics useful to describe the performances seen by the user. The first step is the characterisation of all kinds of protocols used by streaming servers, and particularly by Microsoft Windows Media Server, Apple QuickTime and RealNetworks Realserver. The second step is to expand Tstat to support analysis of such protocols if possible. For example, a proprietary protocol cannot be considered because the correct interpretation of the packet headers is not possible. The extension of Tstat consists also in the introduction of new metrics if necessary.

The focus is on the analysis of streaming protocols. In this context the first step was a preliminary multimedia traffic analysis in order to reveal what are the most used multimedia protocols and how they are combined with the transport layer protocols. This analysis has been performed on some sample traffic acquired on the central router of the Top-IX streaming farm running Microsoft Windows Media Server, Apple QuickTime and RealNetworks Realserver. Depending on the client application and its settings, as well as on other factors (e.g., such as the host network settings, presence of firewalls and NATs, etc.), the following combinations have been observed:

- RTP/RTCP over UDP
- RTP/RTCP tunneled over RTSP over TCP
- RTP/RTCP tunneled over HTTP/RTSP over TCP
- RDT (Real proprietary)
- MS-MMS (Microsoft proprietary)

Since the last two protocols are proprietary and not standard, we decided to focus the analysis on the RTP/RTCP standard protocols and on the several possible combinations with the lower layers. According to the standard, the transport of RTP streams can use both UDP and TCP transport protocols, with a strong preference for the datagram oriented support offered by UDP.

The UDP scenario is the simplest to analyse since the RTP segment is simply the payload of an UDP segment. Therefore it is possible to evaluate all the statistics analysing the RTP header.

The presence of firewall and NAT architectures in the network path however drove the applications to use the TCP connection oriented protocol, in order to circumvent connectivity problems. In this case, a tunnel technique is adopted. The RTP flow is encapsulated over TCP segments, possibly using the same connection that carries the RTSP [4] information exchange. The tunneling does not create a match between RTP segments and TCP segments, because TCP is a byte oriented transport protocol. Therefore, the boundary of a RTP segment can be found wherever into the payload of the TCP stream, since an RTP segment can be carried partially by a TCP segment and partially by the next one. RTP segments are identified by an RTSP header (4 bytes), containing a magic byte ($ as ASCII character) and the dimension of the following segment, allowing to possibly multiplex several RTP streams over the same TCP connection. To identify each RTP stream the header contains also a channel ID.

Considering the above scenario, it is easy to gather that monitoring a tunneled connection makes it difficult to reconstruct the original RTP stream, especially if we consider that:

i the whole packet payload should be exposed to the passive sniffer and analysed, possibly raising privacy issues;
ii sniffing the whole packets content rather than just the first bytes possibly entails scalability problems;
iii special cases, such as out of sequence packets in the TCP connection, have to be handled.

However, we point out that since the RTP flow uses the connection-oriented service offered by TCP, no packet loss nor out of sequence delivery will be perceived by the receiver application. Therefor, the underlying network dynamics would be hidden anyway to the client application (mainly in reason of buffering), which actually turns additional RTP measurements to useless. In other words, according to the latter observation, it is possible to consider a TCP connection carrying an RTSP session and to further identify whether this connection tunnels an RTP stream or not. If so, then all the statistics evaluated on the TCP connection can be inherited by the RTP layer; otherwise the connection will be considered as a common TCP flow.

Concerning the HTTP tunneling, the behavior is the same as discussed above. In this case the RTSP protocol is tunneled into an HTTP request connection and RTP is tunneled into RTSP: all using the same TCP connection. Similarly to the previous case, it is possible to discriminate whether the TCP connection carries an RTP over RTSP over HTTP stream, in which case all the application-layer statistics can be again inherited from the layer-4 statistics.

Finally, an additional consideration can be made for RTCP: the above preliminary analysis, carried over with different client applications and settings, revealed that the data carried by reports are generally unreliable or unknown (i.e., proprietary format). Therefore, since no clear understanding has been gathered on such feedback information, we believe that no significant statistics can be gathered under these conditions.

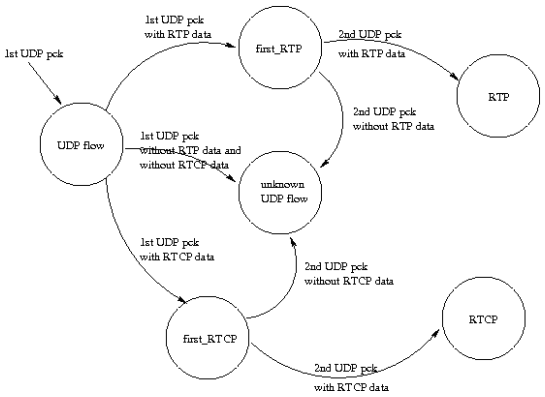

Fig. 1. Heuristic RTP over UDP identification

III. Identification of RTP over UDP flows

UDP transport layer is characterised by the lack of connection signaling. So, to identify an information exchange between two hosts the flow concept is introduced. An UDP flow is a sequence of UDP datagrams exchanged between the same hosts (identified by their IP addresses) and belonging to the same application (identified by the UDP ports). A new flow is identified when an UDP packet, that does not belong to the other monitored UDP flows, is detected. This packet is considered as the first one of the new flow. For each flow a timer, zeroed at each arrival of packets belonging to the considered UDP flow, controls the activity: when no packets are seen for a time greater than a given timeout threshold the flow is considered ended. An UDP flow, as seen in the previous section, can transport an RTP/RTCP flow. Being it impossible to detect RTP/RTCP flows by e.g. port numbers, an heuristic methodology has been defined and implemented, which is based on the FSM (Finite State Machine) shown in Figure 1. At the first UDP packet, the flow is labeled as ”unknown”. For each new UDP flow, Tstat double checks if the UDP payload may be an RTP/RTCP packet. This is done by double checking that:

- the version field is set to 2;
- the payload type field has an admissible value (for RTP or for RTCP);
- the UDP port is larger than 1024 and is even/odd for RTP/RTCP.

If all three conditions are satisfied the flow is marked as a possible RTP/RTCP flow. When the second UDP segment arrives, Tstat checks if, in the case of RTP:

- if the version is equal to 2;
- the same SSRC (source identifier) and the same payload type are present;
- the sequence number is the expected one.

or in the case of RTCP:

- the version must be equal to 2;
- the payload type is the correct one.

After that if the flow satisfies all conditions it is marked as RTP/RTCP and its analysis may start.

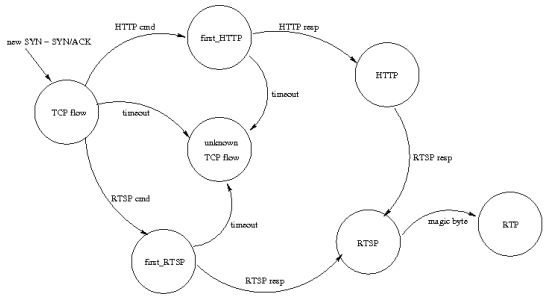

Fig. 2. Heuristic tunneled RTP identification

IV. Identification of TCP tunneled RTP flows

Identification of TCP tunneled RTP flows is more difficult. As seen in the previous section an RTP tunneled flow can be carried over different kind of protocols over a TCP connection. So the analysis of multimedia traffic is based on the identification of TCP connections which tunnel RTP. A generic TCP connection is identified by Tstat, monitoring the connection setup and tear down (e.g. SYN - SYN/ACK and FIN - RST segments). To understand if a TCP connection carries RTP an heuristic based on packet inspection is implemented using the FSM shown in Figure 2. If an HTTP command is found into the beginning part of a TCP flow payload (client to server), and an HTTP response is found into the first segment of the reverse direction (server to client), the connection is considered as HTTP. If an RTSP response follows the HTTP response, the connection can also be considered as RTSP, and it is labeled as an RTSP over HTTP over TCP connection. To catalogue a connection as simple RTSP over TCP, the condition is to find an RTSP command into the beginning of the TCP payload and a RTSP response into the first segment in the reverse direction. When a connection is catalogued as RTSP/TCP or RTSP/HTTP/TCP the algorithm starts to look for the RTP flow. This is done looking for the magic byte which signals that the connection carries a tunneled RTP stream.

The FSM is also able to identify the ICY protocol. This protocol is used by shoutcast server/clients (e.g. WinAmp) for Internet broadcasting.

V. Metrics of interest

Considering the classes of traffic that we want to analyse, we need to define specific performance metrics relevant to multimedia traffic. This metrics are shown in Table I. Some metrics are common, other metrics are specific to the underlying transport protocol. So for instance with TCP there will not be any lost packets but we will see retransmissions.

VI. Measurement setup

The analysis is performed over the multimedia traffic generated by the Top-IX streaming farm. Particularly, all the traffic otugoing and incoming from the Top-IX subnet is mirrored on a port of the edge router where a sniffer hosting Tstat is connected.

Common Metrics	
Metric	*Description*
Source	Source IP address and port.
Destination	Destination IP address and port.
Packets	Number of packets belonging to the flow that have been analysed.
IPG	Average time between two consecutive packets - Inter Packet Gap (IPG).
IPG jitter	Average jitter of the inter packet gap.
OoS	Number of packets Out of Sequence (OoS).
Duplicate	Number of duplicated packets.
TTL	Minimum, maximum and average of the Time To Live (TTL).
Duration	Flow length measured in time.
Data	Flow length measured in bytes.
Bitrate	Average bitrate of the flow.
UDP specific	
Late	Number of packets arrived later than the ideal arriving time plus a threshold.
Lost	Number of lost packets.
TCP specific	
RTT	Minimum, maximum and average Round Trip Time (RTT).
RTT stdev	Standard deviation of the RTT.

TABLE I
Multimedia metrics

	Multimedia		Data	
	Bytes [G]	Flows [K]	Bytes [T]	Flows [K]
IN	7,217	23,124	0,037	546,121
OUT	1,389	1,355	0,969	656,330
TOTAL	8,606	24,279	1,006	1285,911

TABLE II
Multimedia metrics

The results shown in the next section are obtained analysing offline traffic traces collected during the Berlinale Film Festival, February 11th-14th, 2006. Only a portion of traffic, relative to February 13th is not available for an unpredictable misbehavior. Only the first 128 bytes of each packet was collected during the acquisition process to save disk space.

VII. Preliminary results

The first target is the to determine the ratio of multimedia traffic with respect to data traffic shown in Table II. Note that the sum of the incoming and outgoing traffic is not necessarily equal to the total traffic because of local packets (traveling within the subnet) that are counted separately. Looking at the numbers we see that most of the flows are data traffic. The difference becomes even greater if look at the traffic volume in bytes. This means that the multimedia flows are generally shorter. This is not a typical situation. A further inspection showed that there was a lot of multicast traffic: 21491 flows against 2988 unicast flows. After inquiring we discovered that on the subnet is a host joining multicast groups to inspect multicast efficiency and terminates almost instantly the connection. The presence of this host explains the atypical traffic situation seen in Table II.

To avoid being biased by other hosts from here on the traffic has been filtered on the IP addresses of the multimedia

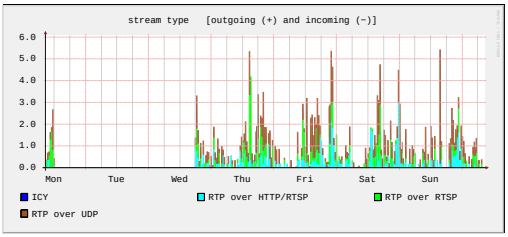

Fig. 3. Traffic load in terms of flows

servers only. Tstat is able to provide for each measure both its distribution and temporal evolution. It is important to note that there is no distinction between RTP over UDP and RTP over TCP flows.

Unfortunately, although the server farm is hosting the Berlinale, the load on the servers (Figure 3) is very low and therefor also the amount of collected data, affecting the statistical value of the measures. The traffic peaks barely reach 5-6 simultaneous connections. Although it is possible to observe a typical night/day behavior.

Looking at the distribution of the protocols used (Figure 4). We see that even if the UDP protocol is much lighter, TCP is used half of the times since it works around NATs and firewalls. The latter being especially the case when HTTP tunneling is used.

Figure 5 shows the distribution of the average IPG of the multimedia flows. We see that there are two peaks around 12 ms and 30 ms. This is probably due to two different constant bitrate codecs.

Figure 6 shows instead the distribution of out of sequence packets per each stream. We see that most streams (almost 70%) have no out of sequence packets. This is quite expected since we are very near to the server. The peak at 100 is due to the fact that every value exceeding the maximum is collected in the last bin.

Finally, as a last example, Figure 7 shows the flow length distribution in Kbytes. Again there are two peaks one near the origin and one around 3 Mbytes. The first one is probably due to people who started watching and then decided to interrupt the stream. The second peak is around the average file length (the videos are quite short: about 1 min).

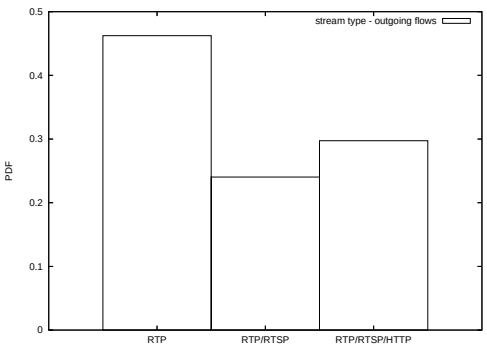

Fig. 4. Protocol distribution

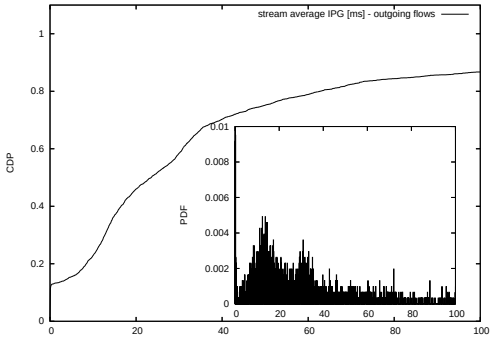

Fig. 5. IPG distribution

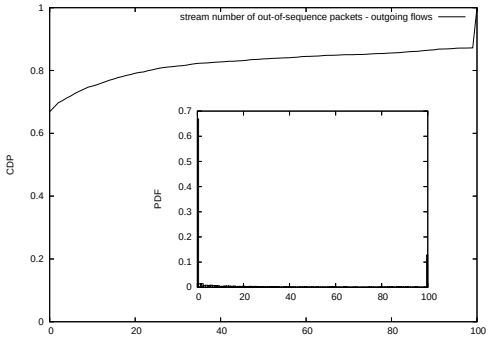

Fig. 6. OoS distribution

ACKNOWLEDGMENT

We would like to thank Top-IX for their collaboration and for letting us measure the traffic on their edge router.

REFERENCES

[1] J. Postel, "RFC 793, Transmission Control Protocol," URL: http://rfc.net/rfc793.html, September 1981.
[2] H. Schulzrinne and oth., "RFC 3550, RTP: A Transport Protocol for Real-Time Applications," URL: http://rfc.net/rfc3550.html, July 2003.
[3] J. Postel, "RFC 768, User Datagram Protocol," URL: http://rfc.net/rfc768.html, August 1980.
[4] H. Schulzrinne and oth., "RFC 2326, Real Time Streaming Protocol," URL: http://rfc.net/rfc2326.html, April 1998.

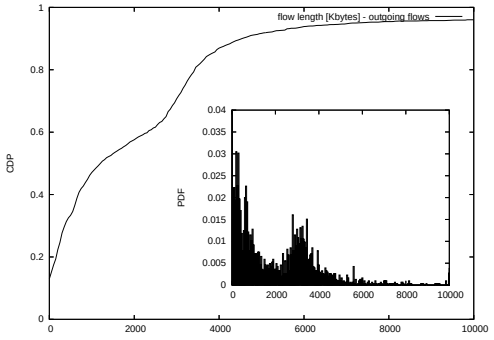

Fig. 7. Flow length distribution