OMTP: COMBINING TCP AND UDP FOR MULTIMEDIA STREAMING

PANAYOTIS FOULIRAS, ATHANASIOS MANITSARIS, PANAGIOTIS FOTARIS

Abstract

In this paper we present the Overlay Media Transport Protocol (OMTP), a novel yet simple protocol, which combines the fairness and congestion control of TCP with the minimalist approach of UDP to deliver high performance for multimedia streaming in a flexible and efficient way. No modification is needed on either TCP or UDP. Furthermore, OMTP can be used with in any type of TCP/IP based network and can be applied to single or multiple levels of multimedia stream quality.

1. Introduction

During the last few years we have witnessed an explosive growth in the development and deployment of network applications capable of transmitting and receiving audio and video over the Internet. Despite these advances, providing Video on Demand (VoD) service over the Internet in a scalable way has remained a challenging problem [1]. This problem is even greater when video delivery must take place over mobile wireless networks, since there are important factors such as higher percentage of lost packets, smaller available bandwidth and variable addresses of clients.

Typically, VoD is composed of a video server, the client computer(s) and the intermediate network. Each video stream emanating from the server represents a *virtual channel*. Due to the nature of the Internet, it is not easy to avoid *jitters* by establishing isochronous virtual channels between the server and the client(s). Many researchers have proposed several interesting ideas, all of which try to minimize the duration of broadcast or the number of additional server channels for the same video [4]. In all cases, however, the principal Internet transport protocols are TCP and UDP. TCP tries to achieve reliability with retransmissions, while simultaneously avoiding/resolving network congestion through the use of the slow-start phase and other techniques. Since UDP does not implement any congestion control, it is not surprising that UDP is used instead of TCP.

The drawback is that UDP tends to “choke” TCP traffic out of the available network bandwidth. Although RTCP [4] (the associated control protocol of RTP) provides information on overall congestion, this is done indirectly and not from TCP’s point of view, since both RTP and RTCP are built on top of UDP. This has lead to a heated debate over the best means to achieve *Quality of Service* (QoS) for multimedia, as well as provide “fairness” to ordinary TCP traffic. Many efforts exist, with *DCCP (Datagram Congestion Control Protocol)* [2], intended as a replacement for UDP for streaming media, being the most recent effort by the IETF – currently at draft status.

In this paper we explore a way to combine the efficiency of UDP with the reliability and congestion control properties of TCP, so that normal TCP traffic is not choked. In section 2 we present our proposal (OMTP). In section 3 we analyze its performance. Our conclusions follow in section 4.

2. Proposed Solution

2.1 The Model

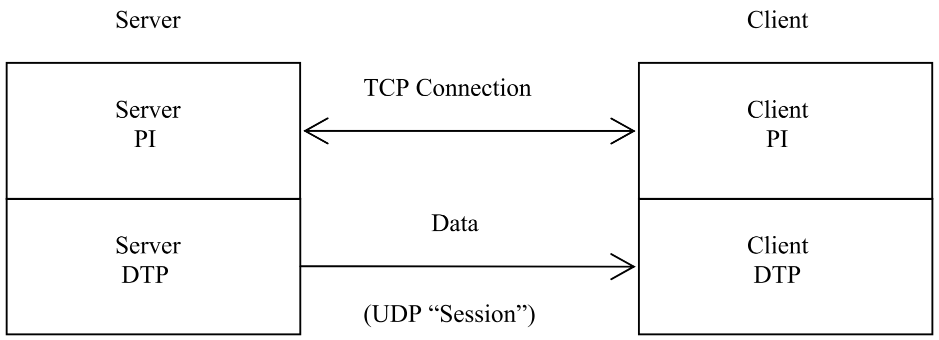

Fig. 1 OMTP Communication Model

In this paper we propose the *Overlay Media Transport Protocol* (*OMTP*). The main idea is to combine the light weight of UDP with the reliability and congestion control of TCP without any modification to either of them. The actual model of communication of OMTP is shown in Fig. 1. OMTP establishes two separate but interrelated sessions: The first is a typical UDP session for the actual transfer of data. The second is a TCP connection for out-of-band control information. The flow of data is unidirectional, namely from the server to the client.

2.2 OMTP Packet Types

Client	Server
Set-up Phase	
OMTP-Request	
	OMTP-Response
OMTP-Ack	
Data Transfer Phase	
	OMTP-Probe
OMTP-Ack	
	OMTP-Data
Termination Phase	
OMTP-CloseReq	
	OMTP-Close
OMTP-Reset	

(a)

1	2	3	4	5	6	7
Packet Type	Session Id	Detail Level	Sequence Number			

(b)

Fig. 2 OMTP (a) Packet Types, (b) Format in Bytes

OMTP uses nine packet types to implement various functions, mainly through its respective TCP connection. Each packet type contains a typical TCP header as well as seven bytes of payload with a few exceptions. There are three phases: *Set-up*, *Data Transfer* and *Termination* (see Fig. 2).

During the Set-up phase the application at the client uses OMTP to set-up a multimedia session with its peer at the server. This is accomplished by establishing a TCP connection and exchanging three OMTP packets: *OMTP-Request*, *OMTP-Response* and *OMTP-Ack*. These allow the application to establish a UDP "session" through which data may travel in a dynamic way, pass the necessary parameters and negotiate with its peer.

Each *OMTP-Data* packet is a UDP segment containing a 2-byte header for the Session-ID and Detail Level fields (see Fig.2b); the rest of the packet contains the application data. There is no need for additional fields, since UDP is used for just one packet type. All other OMTP packets are exchanged using the TCP connection.

The *OMTP-Request* packet has one byte for the packet type code, a second byte for the Session ID, a third byte for the requested Detail Level number and a 4-byte initial (random) sequence number. It also has an optional amount of fields in order to pass additional information to the server, such as the maximum and lower Detail Level values and their correspondence to actual transmission rate. The rest of the packet types have identical the first two fields and are as follows:

- The *OMTP-Response* packet has a third byte for the maximum available Detail Level number, that is less or equal to the earlier requested detail level value, and the sequence number received by the client, incremented by one, in order to acknowledge the previous request. It also attaches optional information respective to the request made earlier.
- The *OMTP-Ack* packet has a third byte for the initial *agreed* Detail Level and the initial sequence number incremented by one. This packet type is used by both client and server.
- The *OMTP-Probe* packet has a third byte for the current Detail Level number and the current sequence number incremented by one. The server periodically sends OMTP-Probe packets which are identified by their sequence number; this refers to the packet as a whole and not to some byte as in TCP.
- The *OMTP-CloseReq* packet has a third byte for the current Detail Level and the current sequence number plus one; the *OMTP-Close* packet has exactly the same format, with the difference that the Detail Level field is used to return a status code for the termination.
- Finally, the *OMTP-Reset* packet type is of the same format as the rest, but used to terminate an OMTP "session" in an "abrupt" way. It can be issued by both client and server.

2.3 Establishing and Managing Video Sessions

At first, the client sends an OMTP-Request packet, where the Session ID field is a random number different from the one used in its last OMTP session in order to avoid confusion; another (initially) random value is selected for the sequence number, and four additional fields are appended for the lowest and highest requested Detail Level field and their respective descriptions. A UDP port is also reserved and its port number sent to the server. If the server can provide for some or all client's requests, it replies with an OMTP-Response packet and reserves a UDP port whose number is appended at the end of the response. Finally, the client sends an OMTP-Ack packet and the server is ready to start video delivery.

OMTP-Probe packets are sent by the server to the client through the TCP connection. A special variable $r_{delivery}$ is maintained by the server. This is equal to the number of TCP segments sent by the server to the client that have been acknowledged within a specific time period d. This time interval can be set dynamically by some rule or be fixed at some constant value. In the latter case d is set to 0.2 seconds (similar to the one specified in [2]). In practice though, it is better to set it dynamically as explained in the following section. Another variable r_{max} is also maintained, so that the maximum value of $r_{delivery}$ ever reached is also measured. Initially, both variables are set to 0 and the data (through the UDP port) is sent at full speed. As TCP mechanisms of error, flow and congestion control are developed, the maximum value of r_{max} is reached and the transmission of OMTP-Data packets is determined by the ratio:

$$q = r_{delivery}/r_{max}$$

This is essentially used as a regulator value for the transmission of UDP data segments. The value of q can be used either by itself or in combination with the Detail Level field. In the simplest possible scenario, e.g. if $q = 1/5$, only one out of five queued UDP segments would be transmitted by the server. Alternatively, the server could try to inject packets with lower quality. In all cases note that no time-stamp is used in order to keep both the packet overhead low and the protocol simple to implement on top of TCP/UDP. Other relations such as multiple thresholds in step-like function scheme can also be adopted, after having been agreed upon between the client and the server during the Set-up phase. Finally, the Termination phase is pretty straightforward.

3. Performance and Mobile Multimedia

Under OMTP, the data is transmitted at maximum speed with indirect congestion control exercised when the TCP link dictates it, making OMTP fair compared to other traffic. The inherent slow-start of TCP can be addressed by having the client and the server agree on a (relative) threshold for q under which the OMTP-Data packet transmission rate will not be hindered. In this way, a minimal data transmission rate can be roughly guaranteed on behalf of the server.

The overhead of OMTP-Data packets is only two bytes. With 512 bytes typically for the application MSS (Maximum Segment Size), this represents less than 0.4% load on average.

As for the rest of the OMTP traffic, typical RTT times are 200 msec for international and 20 msec for national traffic; for LANs this value is even smaller (< 1 msec). Each of the OMTP-Probe packets represents an overhead of at least 47 bytes (20 for IP, 20 for TCP and 7 for OMTP headers), without considering retransmissions and data link layer overheads, but not much more than that. Hence, each OMTP packet does not contribute much in terms of overhead on the intermediate network; it is the *total amount* of OMTP-Probe packets that may cause alarm.

OMTP provides for an adjustable timer represented as the variable d, where $1 \text{ msec} \leq d \leq 200 \text{ msec}$. Also, there is a variable number $2 \leq v \leq 32$ of OMTP-Probe packets sent at the beginning of the time period d. These two parameters form the pair of transmission parameters (d, v).

At the beginning of the Data Transfer period, there is a sub-period for determining the appropriate values of (d, v). Initially $d = 1$ msec and $v = 2$. If at the end of this interval $q = 0$, it is assumed that (d, v) were too strict and another pair of values is tested until $0.1 < q < 0.9$ and either or both (d, v) have reached their maximum values.

The rule for increasing the values of these parameters is as follows:

```
Step 1:    while ($d < 200$ and $v < 32$) do
Step 2:    if ($q \leq 0.1$) then
               $d := 10*d$ and $v := 2$;
               goto Step 1
Step 3:    else if ($q \geq 0.9$) then
               $v := 4*v$;
               goto Step 1
```

This means that in the worst case (continental traffic or very slow link) the appropriate values will be reached in less than one second. The total overhead in the worst case (32 OMTP-Probe packets) is approximately 1,500 bytes per 200 msec or 7.5 KB/sec. Over a LAN this represents a tiny bandwidth fraction. Over national links with leased lines (typical RTT = 10 msec), this would mean less than 8% in the worst and 0.5% in the best case. For very slow links (or low bandwidth) the situation is also quite good: Assume a 32 kbps dial-up link and some audio transmission. There can be no more than 2 OMTP-Probe packets with < 100 bytes as communication overhead per 200 msec or < 1,000 bytes/sec, which is < 3.5% of overhead.

In the case of frequent retransmissions due to a high percentage of errors (typical for mobile clients), the overheads show similar behavior, since it is the number of acknowledged OMTP packets that are used for calculating actual throughput (the value of q). The fact that threshold values of q can be set for transmission of OMTP-Data packets makes OMTP more flexible and extendable in the face of new developments for TCP.

4. Conclusion and Future Work

We presented OMTP, a simple, novel, yet efficient and flexible overlay protocol for multimedia traffic, which introduces fairness to UDP when in the same environment with other traffic. It uses most of the functionality of TCP and can be employed in a variety of different networking environment with minor modifications at the application layer. The introduced overhead is minimal and communication bandwidth is used quite efficiently.

Work is under way to explore the possibility of having an OMTP session between a server and a *class* of computer clients of the same type, as well as having an OMTP session between a server and a group of clients with an additional client being their controller – something useful in strictly controlled environments as e-learning with an instructor having a computer with more privileges.

5. References

[1] CHAN G. S. H., Operation and Cost Optimization of a Distributed Servers Architecture for On-Demand Video Services, IEEE Communication Letters, Vol. 5, No. 9, 2001, pp. 384-386.

[2] KOHLER E., et al, Datagram Congestion Control Protocol (DCCP), draft-ietf-dccp-spec-06.txt, February 2004.

[3] KUROSE J. F., ROSS K. W, Computer Networking: A Top-Down Approach Featuring the Internet, Addison-Wesley, 2nd Edition, 2002.

[4] PERKINS C., RTP Audio and Video for the Internet, Pearson Educational, 2003.