Advanced Network Programming VII
UDP

Juliusz Chroboczek

20 April 2025

Up to now, we have been basing our protocols on HTTP(S). Using HTTP(S) as the convergence layer has a number of advantages:

- HTTP crosses firewalls, NATs and other middleboxes;
- proxying and caching technologies for HTTP are well understood, and they make the protocol scale very well[1].

However, HTTP(S) imposes a number of constraints:

- it is client-server only, especially when secured by TLS (server certificates require a domain name);
- it obeys a strict request-response discipline;
- there is a fairly large per-request overhead;
- requests are not ordered (if a client sends two requests in a row, they may arrive in a different order than the one they went sent in).

In addition, since HTTP (up to HTTP/2) is based on TCP, it inherits the limitations of TCP:

- it is reliable and ordered, and therefore unsuitable for real-time communication.

This last point is often misunderstood. UDP is not faster than TCP: an IP packet containing a TCP segment is usually routed just as fast as an IP packet containing a UDP datagram. However, TCP is ordered: packets that arrive out of order are delayed until they can be delivered in order. And TCP is reliable: a packet can be delayed by arbitrary amounts of time while it is being resent. In consequence, if a packet is lost, all packets following the lost packet are delayed until the missing packet has been resent.

1 Digression: real-time communication

A *real-time* system is a system that makes guarantees about the time after which a given event happens. For example, a real-time operating system is able to give guarantees about the delay

1. Recall the example of mobile phone operating system upgrades.

after which a given process is scheduled on the CPU, and a real-time communication system gives guarantees about the time for a packet to arrive.

There are two kinds of real-time systems

Hard real time A hard real-time system makes guarantees about the maximum time after which an event happens. For example, an industrial robot might be able to guarantee that the system is shutdown at most 100 ms after a human limb has been detected.

Hard real time is extremely costly, and impossible to achieve over the Internet, which is a stochastic system. It is therefore of little concern to us.

Soft real time A soft real-time system makes statistical guarantees about the time after which an event happens. For example, an industrial robot might be able to guarantee that the system reacts to excessive pressure on a component within 100 ms in 99% of cases.

Soft real-time systems are an open research area, and they can be described in many ways. Network links are often described in terms of average delay (the average time taken by a packet to cross the network), jitter (the variability of delay between packets) and packet loss (the fraction of packets that are lost in the network, or arrive too late to be taken into account in the average delay and jitter measurements).

2 UDP

UDP is a minimal protocol over IP: it essentially augments IP with port numbers, which are a hard requirement to make it practical to make a protocol usable by user-space code. UDP should not be seen as a transport layer protocol, but merely as a very slight wrapping of IP. Thus, it is reasonable to build transport layer protocols over IP.

UDP has the following characteristics:

- UDP source and destination addresses are socket addresses (pairs of an IP address and a port number);
- UDP transports limited-size *datagrams* (UDP packets);
- UDP is unreliable: a datagram may be lost with no error indication;
- UDP is not ordered: datagrams are delivered to the application as soon as they arrive, and therefore not necessarily in the order they were sent in;
- UDP has neither flow nor congestion control: the UDP sender is allowed to send datagrams as fast as it can, even if the data will be discarded by the network (lack of congestion control) or by the receiver (lack of flow control).

2.1 Datagram size

Different networks are able to carry different sizes of IP packets. Thus, the maximum size of a datagram that can be expected to arrive to the destination depends on the network.

The simplest way to work around this problem is to define a *minimum maximum packet size*: a packet size that is guaranteed to work in all networks. Unfortunately, there is no commonly agreed minimum maximum packet size:

- IPv4 guarantees a minimum maximum packet size of 576 bytes; however, in practice, all IPv4 networks are able to carry packets of 1500 bytes, possibly by fragmenting them;
- IPv6 guarantees a minimum maximum packet size of 1280 bytes.

As a result, it is usually safe to send datagrams of 1272 bytes (8 bytes for the UDP header), which leaves enough space to send a user payload of 1024 bytes plus some overhead (transport layer data, cryptographic signatures, etc.).

Path MTU discovery In order to send larger datagrams, it is necessary to dynamically probe the maximum allowable datagram size. This is the problem of *Path MTU discovery* (PMTUD).

2.2 Reliability

UDP is unreliable: a packet may be lost with no error indication. Therefore, a protocol based on UDP must either implement its own reliability (by sending acknowledgements), or else be designed to work well even in the presence of packet loss.

Consider for example a video game where the position of a player's opponents is broadcast periodically. If a packet is lost, the old position would be obsolete by the time it would be resent; it is better to send the new position instead.

2.3 Flow and congestion control

In order to achieve decent performance, it is necessary to send packets slowly enough to avoid overwhelming buffers both at the receiver and inside the network. The simplest way to achive that is to use a request-response algorithm, and to make sure that there is always at most one packet in flight (sent but not yet replied to). In practice, this leads to protocols that are unacceptably slow[2].

Flow control is the algorithm that avoids overwhelming buffers at the receiver, and it is fairly easy to achieve: there are well understood receiver-driven sliding window algorithms.

Congestion control is the algorithm that avoids overwheliming the network, and it is an area of open research. Traditional congestion control algorithms take packet loss as an indicator of congestion, which only works well when the routers implement modern packet drop algorithms (Active Queue Management policies, AQMs). More modern algorithms take both packet loss and an increase in end-to-end delay as an indication of congestion, and work reasonably well even in the absence of AQMs.

2. Because the speed of light is glacially slow.