Why TCP Over TCP Is A Bad Idea

A frequently occurring idea for IP tunneling applications is to run a protocol like PPP, which encapsulates IP packets in a format suited for a stream transport (like a modem line), over a TCP-based connection. This would be an easy solution for encrypting tunnels by running *PPP over SSH*, for which several recommendations already exist (one in the Linux HOWTO base, one on my own website, and surely several others). It would also be an easy way to compress arbitrary IP traffic, while datagram based compression has hard to overcome efficiency limits.

Unfortunately, it doesn't work well. Long delays and frequent connection aborts are to be expected. Here is why.

TCP's retransmission algorithm

TCP divides the data stream into *segments* which are sent as individual IP datagrams. The segments carry a *sequence number* which numbers the bytes in the stream, and an *acknowledge number* which tells the other side the last received sequence number. [RFC793]

Since IP datagrams may be lost, duplicated or reordered, the sequence numbers are used to reassemble the stream. The acknowledge number tells the sender, indirectly, if a segment was lost: when an acknowledge for a recently sent segment does not arrive in a certain amount of time, the sender assumes a lost packet and re-sends that segment.

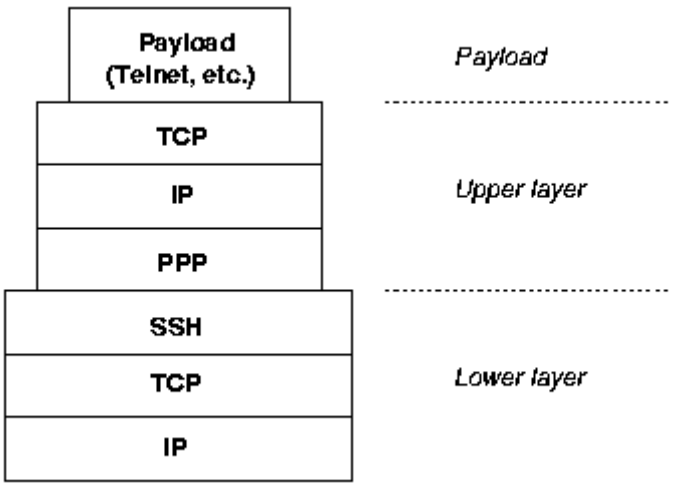
Many other protocols using a similar approach, designed mostly for use over lines with relatively fixed bandwidth, have the "certain amount of time" fixed or configurable. In the Internet however, parameters like bandwidth, delay and loss rate are vastly different from one connection to another and even changing over time on a single connection. A fixed timeout in the seconds range would be inappropriate on a fast LAN and likewise inappropriate on a congested international link. In fact, it would increase the congestion and lead to an effect known as "meltdown".

For this reason, TCP uses adaptive timeouts for all timing-related parameters. They start at conservative estimates and change dynamically with every received segment. The actual algorithms used are described in [RFC2001]. The details are not important here but one critical property: *when a segment timeouts, the following timeout is increased* (exponentially, in fact, because that has been shown to avoid the meltdown effect).

Stacking TCPs

The TCP timeout policy works fine in the Internet over a vast range of different connection characteristics. Because TCP tries very hard not to break connections, the timeout can increase up to the range of several minutes. This is just what is sensible for unattended bulk data transfer. (For interactive applications, such slow connections are of course undesirable and likely the user will terminate them.)

This optimization for reliability breaks when stacking one TCP connection on top of another, which was never anticipated by the TCP designers. But it happens when running PPP over SSH or another TCP-based protocol, because the PPP-encapsulated IP datagrams likely carry TCP-based payload, like this:



Note that the upper and the lower layer TCP have different timers. When an upper layer connection starts fast, its timers are fast too. Now it can happen that the lower

connection has slower timers, perhaps as a leftover from a period with a slow or unreliable base connection.

Imagine what happens when, in this situation, the base connection starts losing packets. The lower layer TCP queues up a retransmission and increases its timeouts. Since the connection is blocked for this amount of time, the upper layer (i.e. payload) TCP won't get a timely ACK, and will also queue a retransmission. Because the timeout is still less than the lower layer timeout, *the upper layer will queue up more retransmissions faster than the lower layer can process them*. This makes the upper layer connection stall very quickly and every retransmission just adds to the problem - an internal meltdown effect.

TCPs reliability provisions backfire here. The upper layer retransmissions are completely unnecessary, since the carrier guarantees delivery - but the upper layer TCP can't know this, because TCP always assumes an unreliable carrier.

Practical experience

The whole problem was the original incentive to start the CIPE project, because I used a PPP over SSH solution for some time and it proved to be fairly unusable. At that time it had to run over an optical link which suffered frequent packet loss, sometimes 10-20% over an extended period of time. With plain TCP, this was just bearable (because the link was not congested), but with the stacked protocols, connections would get really slow and then break very frequently.

This is the detailed reason why CIPE uses a datagram carrier. (The choice for UDP, instead of another IP-level protocol like IPsec does, is for several reasons: this allows to distinguish tunnels by their port number, and it adds the ability to run over SOCKS.) The datagram carrier has exactly the same characteristics as plain IP, for which TCP was designed to run over.

Olaf Titz
Last modified: Mon Apr 23 11:50:59 CEST 2001

Olafs Home Page (archive.org)