6.033 Lecture 11
Networking III

Americo De Filippo

June 13, 2023

References

[1] Saltzer, Jerome H. and M. Frans Kaashoek. Principles of Computer System Design: An Introduction (2009): **Section: 7.6**

Today we are gonna continue with the "end to end" layer.

The charactersitcs about our network design It was Subject to losses, subject to reordering, delays and congestion (queueing).

1 E2E Layer

The end to end layer helps us deal with this limitation and not only. Provides the ability to multiplex different machine on the same network. The other thing that provides is fragmentation of the message (take a long message and fragement in fragments that will transmit into the network). When a message arrives to the machine it will be send to a specific process based on the **port number** of the message. This dispachement is done by the end to end layer, another thing that we have mentioned is fragmentation and recostruction. The end to end layer provide a stream that is just a flow of data from 2 machine, it will be loss-free and in-order (this stream abstraction is what the TCP protocol provides).

2 Dealing with Loss

It will done by providing "At least once" delivery. This tequinque is based on a protocol that will assure that the sender will know that the receiver may not have received the message. Futhermore, we have to avoid duplicates that will occurre.

2.1 At least once Protocol

This protocol will garantee that the message will be recevied by the receiver. When a message will arrive to the receiver, it will send an ACK message to the sender. When the sender will send a message is going to push it in a table where is going to store all the message that hasn't been already aknoledge. This is good 'cause in the case that a loss occurres the sender is gonna keep a timer for the element in the table, when the timer is up and the message is still in the table, the sender is going to resend the message. In the case however that the ACK message is lost the receiver has to be able to discard the duplicates messages. There is another thing with this protocol, is how we are gonna pick this time intervals?

How long to wait? However long is the round trip time or the RTT, this arroach has a problem, the RTT will not be constant. So we need some kind of slop value, adjustement factor. The idea is that we want to estimate the average RTT, one way is to keep a set of sample of n round trip times. But this is not efficient, instead we are gonna use a tequinque called: **EWMA**, what it does is that given a set of samples $S_1 ... S_{new}$ it increamentaly adjust the RTT according to the following formula: newRTT = $(1 - \alpha)S_{new} + \alpha * oldRTT$, with $0 <= \alpha <= 1$. What this does is to make the new RTT some weigthed combination of the old RTT. The slop is calculated as $\beta*$ the difference between the predicted S_{new} and the actual. In the TCP protocol α is set to 7/8, on the other hand β is set to 4.

2.2 At most once Protocol

The idea behind this is that we want to suppress duplicates. This works similary to how ACK message works, with a table. When the message is already in the receiver table it will resend only the ACK message and ingnore the duplicates. The problem here is that we are gonna have a really big table. The solution to this is done by letting know to the receiver which message has been completed, and eliminate it from the table. The message that are left over in the table are called **Tombstones** which are some kind of anomalies in the received messages. This practice is called "Lock step" because the sender will send a message and then it will wait the receiver for the ACK message. This thing is kind of a limitation for the network, so this is avoided by sending the next message without having already had the ACK of some previous message. So we have a lot of overlapping of old ACK and new sent messages in this way the tput is just limiteded from the speed of our network. The problem here could be that the recv is not able to handle the data at the maximum speed. So the recv will tell to the sender the rait at which he would receive the message in a way that are possible to be handled.

2.3 Flow Control

The flow control is necessary for having a peaceful conversation in a way the recv will be able to compute the information offred from the sender. This is implemented in different ways by 2 tequinque called: Fixed windows [1], Sliding windows.

2.3.1 Fix Windows

This is implemented by having some kind of agreement before the starting of the transmission. In this way we have period of times where the sender is just sit idle 'cause has to wait for the recv message of ending computing.

2.3.2 Sliding Windows

In this case we are gonna negotiate some WS and then every time that a message is processed is going to send the ACK message for the processed message. In this way every time that there is empty space in the buffer of the recv the sender will know and send others messages.

2.4 Pick WS

The problem that we now have is that small windows provided underutilization of the net. The problem now is this how big ought to be the window? We want that it's ¿= message processing rate × rtt. In this way we could use in a efficient way the networks, in a way that the WS is big enought to not continue processing data while the sender send others and of course receive ACKs.

[1]The number of data that the recv can accept at one time