Lecture 12

Gidon Rosalki

2026-01-11

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_networking`

1 TCP Reno

This avoids slow start, to prevent the network links from emptying after encountering packet loss. The basic idea is to treat timeout, and 3 duplicate ACKs differently, since every duplicate ACK indicates a successful packet transmission. It uses AIMD (Additive Increase Multiplicative Decrease). It probes the available bandwidth, and has a fast recovery to avoid a slow start. Duplicate ACKs result in a fast recovery, with a fast retransmission, but a timeout results in a retransmission, and a reset to slow start.

The basic algorithm is as follows:

Reno 1

1: **for** ACK in ACKs **do**
2: $\quad W += \frac{1}{W}$
3: **end for**
4: **for** Every 3 duplicate ACKs **do**
5: $\quad$ ssthreshold $= \frac{W}{2}$
6: $\quad W = \frac{W}{2}$
7: **end for**
8: **for** Every timeout **do**
9: $\quad$ ssthreshold $= \frac{W}{2}$
10: $\quad$ Enter slow start (with $W = 1$)
11: **end for**

The first for loop is the additive increase stage, and the second multiplicative decrease.

Let us consdier the *throughput* of TCP Reno. We will calculate it on average as a function of window size, and RTT, ignoring slow start, and other buffers. Let W be the window size when a loss occurs. When the window is W, then the throughput is $\text{MSS} \times \frac{W}{RTT}$. Just after the loww, the window size drops to $\frac{W}{2}$, and the throughput to $\text{MSS} \cdot \frac{W}{2RTT}$. Thus, the average is

$$\frac{3}{4} \cdot \text{MSS} \cdot \frac{W}{RTT}$$

Let us define a *fair* protocol. If k TCP sessions share the same bottleneck link of bandwidth R, then each should have an average rate of $\frac{R}{k}$. We can see that TCP Reno is fair from the following example. Consider 2 competing sessions, and a graph where on x we have the throughput of connection 1, and on y the throughput of connection 2. There is a line connecting the R value on both axes, representing the maximum throughput. Fairness is at the point where this line meets $y = x$. We will note (from testing) that no matter where they start, both will converge on that point.

This does not happen on the internet, since many applications (such as multimedia) do not use TCP, since they do not want their rate throttled by congestion control, so handle packet loss, but transmit at a constant rate. Additionally, for all TCP may be fair, our browser can open many connections, and thus bypass each connection being "fair", since now it is taking up 10 times the bandwidth proportion.

So, we have two variants, Reno, and Vegas. They differ mostly in their congestion avoidance. Vegas is delay based, and Reno is loss based. Vegas ran into issues, where despite its model working excellently in the laboratory, when it worked with *other* TCP protocols, its transmission rate went to almost nothing, since non of the other protocols were reducing the throughput in a similar manner.

2 Queuing at routers

Congestion control is a distributed asynchronous algorithm to share bandwidth. TCP adapts the sending rate to congestion, and router queuing adjusts, and feeds back congestion information. There are 2 main ways, the naïve method is *drop tail*, which is a FIFO queue, that drops packets that arrive at a full buffer. There are also implicit feedback models, where the queueing process implicitly computes and feeds back the congestion measure. Active queue management gives **explicit** feedback, which provides congestion information by probabilistically marking packets. There are 2 ECN bits in the IP header that are allocated for active queue management. This is supported by routers, but is usually turned off.

3 Interdomain routing and BGP

We have already seen and discussed many routing methods, which all tried to find the shortest route from point A, to point B, within a single network. However, the wider internet is comprised of *many* different networks, which can range from small businesses / schools, to large multinational corporations. Every such network is called an Autonomous System (AS). For this routing, we will create the topology where each node is an AS, destinations are prefixes (such as 12.0.0.0/8) , and edges are links / business relationships. Autonomous System Numbers (ASNs) are 16 (or sometimes 32) bit values. For example, MIT has 3, Harvad 11, HUJI 378, AT&T have 7018, 6341, 5074, and more. Verizon have 701, 702, 284, 12199, and more.
In the commercial internet, ASes sign bilateral long term contracts about how much traffic to carry between them, which destinations they reach, and how much money they pair for this server. Neighbouring pairs of ASes typically have a *customer provider* relationship, or a *peering* relationship.

For the customer provider relationship, the customer needs to be reachable from everyone, so the provider ensures that all its neighbours can reach the customer
For the peering relationship, peers exchange traffic between customers. Often this relationship is settlement free (no money exchanged).

At the top of the internet hierarchy, we have the tier 1 ISPs. These have no upstream providers, and typically have a large (inter)national backbone. There are around 10-12 ASes, comprised of AT&T, Sprint, and others.
Tier 2 providers provide transit service to the downstream customers, but need at least one provider of their own. They typically have national, or regional scope, and there are a few thousand such ASes. There are also stub ASes, which do not provide transit service, and connect to upstream providers. These stubs comprise of most (about 85%) of the ASes (like HUJI).

3.1 Interdomain routing

Different ASes are connected to each other through border routers, which communicate over the protocol BGP (Border Gateway Protocol. This uses interdomain routing, performed between ASes, across different entities. This contrasts to intradomain routing, which we have discussed until now, which is within a single AS, where all the network devices belong to the same entity.
BGP is **not** shortest path routing. Different requirements may be placed on the route. For example, Google might require the cheapest route, Verizon might want the cheapest, but Comcast and AT&T might want routes that avoid each others networks.
BGP is critical for routing in the internet. For example, almost 50% of Skype disruptions are BGP related, and every year or so, a serious BGP related Internet outage makes the news. It is notoriously vulnerable to attacks.

3.1.1 Routing overview

Every node has an AS number, and each AS has an IP prefix. Each AS announces itself to its neighbours, including the AS number, and its prefix. These neighbours can then choose whether to make use of this route to the IP prefix, and distribute it to their neighbours, or make use of a different route. Routes to the destination are built hop by hop as reachability information propagates through the network, and route selection is based on local routing policies.
As we can see, this means that BGP is a distributed protocol, where for each destination IP prefix every node repeatedly executes the following process, upon receiving BGP routes from its neighbours:

- Apply *import policy*: (e.g. filter unwanted routes from neighbours)
- Select "best" route: Choose the best route according to the local preferences (distance, cost, speed, avoiding certain people)
- Apply *export policy*: Filter routes that you do not want to announce to neighbours
- Transmit the chosen BGP route to the neighbours

3.1.2 Key points

1. Independent computation for **each** IP prefix: This means that routes to **every** IP prefix are computed *independently*

2. Entire path announced: Under BGP, the **entire** AS level path to the destination is announced when a link is published (so, when AS 2 builds a link to AS 1, AS 2 will announce the path (2, 1), as opposed to AS 1 which only announced path (1)). This was originally intended for loop detection purposes

3. Local routing policies: BGP allows ASes to express complex local routing policies, such as node 2 preferring the path (2, 3, 1) over (2, 1), and node 1 may not let node 3 hear the path (1, 2).