Reducing Internet Latency

Matt Mathis, June 2013

As a community we have pretty good ideas of the sorts of things that we need to do to reduce Internet transaction latency (e.g. page load time, etc). The effects of "bufferbloat" are well known, although there is not a clear consensus on what path we should take to address this and other sources of latency.

I argue that (nearly) all approaches are needed, either during a transitional phase or in steady state. There are no shortcuts and nearly all solutions that are appropriate (or even optimal) in some context.

There is a fairly straightforward argument that the ultimate solution to "bufferbloat" has to be deployed in the network with some form of AQM, fair queueing or other strategies. The difficulty is that the deployment path for upgrading the entire network is very long and very expensive -- likely to require forklift upgrades at many points in the network.

Therefore, it is also desirable to pursue endsystems solutions, because they will be relatively quick, easy and inexpensive to deploy. These solutions all require augmenting existing congestion control with additional algorithms to detect the delay introduced by a standing queue at a bottleneck. It is easy to construct demonstrations where any of these solutions work well, especially for many extremely common scenarios with a single user on one access connection at a typical residence.

But it has been shown that delay sensing algorithms exhibit instabilities and other pathologies whenever multiple long running streams share a bottleneck, and that these problems are fundamental and cannot be solved in the end system alone.

Both are needed: the end system can fix many of the most common failure cases in the short term, but in the long term nearly all of the network will need to be upgraded eventually.

The most important requirement of the delay sensing algorithms in the end systems are that they not interfere with the long term deployment of proper network based solutions. The algorithm performance in the easy case (one large flow with moderate noise introduced by smaller flows) is not as important as making sure that it does not introduced barriers or disincentives to deploying correct solutions in the network. The delay sensing algorithms must do no harm, especially when they are not needed.

In the case of competing delay sensing flows is unlikely to be stable, whether or not the the controllers are "compatible". Trying to optimizing this part of the behavior is unlikely to be successful and should not be a priority.

The network AQM algorithms do not need to be standardized. Any debate about the merits of the various algorithms is counter productive. Any algorithm that does a reasonably good job of controlling the minimum queue occupancy (or the minimum sojourn time) over a wide range of conditions is fine. For a customer, what matters is if the algorithm covers all of the conditions that are relevant for your range of scale.

There is some risk that AQM without any mechanism to foster fairness will actually make the rate unfairness worse. The is because bloated buffers tend to partially mitigate the differences in RTT and thus partially mitigate the TCP rate unfairness. For this reason non-FQ Codel should not be recommended for aggregated traffic.

The Internet is also suffering from general problem that the public has no well calibrated expectation for performance. Without this knowledge the users are far less likely to complain when they should.

The most extreme example applies to the entire stack. I have first hand experience with multiple web administrators that had badly underpowered workstations and/or poor office networks. They had no comprehension of the actual performance as seen by others, and thought that 15s page load times were normal. They couldn't appreciate that on a properly connected modern workstation their own pages took 10 times longer to load than those of their competitor. This is especially problematic if management is using archaic resources to guide their resource allocation decisions.

Users also do not understand that it is a bug that bulk or streaming data interferes with other web applications. Their expectations are so low that they do not know they should be bothered by how poorly the network actually works.

In the end, these call for educational efforts, which are normally out of our scope.