There is No Magic Transport Wand

by John Leslie, 23 June 2012

This paper is a bit of a rant. We're acting as if we believe congestion will magically be solved by a new transport algorithm. It won't. We need action at the Network layer, the Transport layer, and the Application layer, at a minimum.

I'm following several real-time-video Working-Groups, and I find none of them to be considering the congestion they contribute to and what steps they could take to alleviate congestion. They point to existing transport protocols and defer all such details to those.

But alas, those protocols often assume dedicated bandwidth, where a bandwidth is negotiated and assumed to be generally available. In today's Internet, glitches in that available bandwidth are increasingly common, and lead to increasingly nasty glitches in the user experience.

One aspect of this is obvious in the remote audio from IETF weeks: it starts the day with delay deemed sufficient to remove jitter; but as the day goes on, this jitter-clearing delay keeps increasing. Often it becomes unusable for remote participation.

In other cases, we experience major audio glitches which render comprehension doubtful. When these are rare enough, we ask the speaker to repeat; when they become frequent, we instead give up and guess what the speaker might have been saying. :^(

When both video and audio are involved, lip-sync problems are frequent. When this exceeds about half a second it becomes confusing, even to people who don't read lips. It would almost be better if their lips weren't perceived to be moving at all.

Increasingly, IMHO, we should be looking to provide both video and audio -- essentially all laptops sold today have the capability to process both simultaneously. Using these in combination points to a quite different approach to transport over the Internet.

1.
My first point applies to Application layer. When transporting both video and audio, cutting back on the data rate of either will alleviate congestion the same amount. And what the ideal tradeoff may be isn't the same for all users.

For the average user, we have lots of experience showing that audio glitches are more distracting than video glitches. (Of course, if the subject is a baseball game, a video glitch at the wrong time trumps any amount of audio glitches.)

But different users have different "glitch" thresholds. As we age, our hearing changes, typically losing any sensitivity to the highest frequencies. At any age, "hearing damage" may result in an underlying background "buzz" which interferes with hearing low decibel levels. And of course there are "hearing-impaired" folks that must depend on lip-reading. I don't believe there is any way to optimize an audio transport for all users.

So I believe that recognizing what a user will perceive as an audio glitch is for the application layer, not something we should even think about standardizing at transport layer. Similarly, I am disinclined to try to standardize what constitutes a video glitch.

If we agree these cannot be standardized at the transport layer, we must leave room for the application layer to optimize for them. Applications should be encouraged to send individual video frames (or partial frames) at increasing/decreasing rates to adjust to congestion, as well as using

differing audio bandwidths.

2.
My second point applies to Transport layer. Fast recognition of congestion is especially important in real-time audio and video. We should make use of something like ECN to immediately notify the receiver that congestion is being experienced along the path. There are things the receiver can do to mitigate glitching due to loss of packets during the next round-trip-time before the sender can adjust sending rate. In fact, just receiving a one-bit "congestion in path" indication _without_ the packet in transit would be helpful.

To actually signal the sender to reduce data rate, we need a dependable back-path, which means we cannot afford to wait for a timeout to determine a signal has been lost. Essentially this calls for a "heartbeat" at a pretty hefty rate, certainly no less than once in 50 milliseconds. In most conferencing cases, it's reasonable to expect the receiver will also be sending something (even an "away" signal) with that sort of frequency. What information to carry in such a "heartbeat" likely blurs the line between transport and application layers, though there's enough purely transport-layer information to justify it.

(To digress slightly, we should expect home-offices to be a major component of real-time traffic, using highly asymmetric links, so that incoming traffic greatly exceeds outgoing. Nonetheless, outgoing traffic has no reason to be zero, just one or more orders of magnitude less.)

There is an issue here about distinguishing downstream congestion from upstream congestion; but several approaches are obvious: with ECN it's trivial, but even without ECN there are obvious algorithms. We do have to agree that datarate won't go below "heartbeat" level without declaring connection failure, but there should be an obvious way to distinguish when heartbeat is no longer workable. (Can you say "backhoe"?)

So, bottom line, there is a good amount of work to be done at transport layer; but while it is well worth doing, it cannot solve enough of the problem.

3.
My third point concerns Network layer. Van Jacobson's CoDel paper deserves mention here. Basically it proposes packet drop (or ECN marking presumably) be based on how long the packet has been in a local queue as a solution to buffer-bloat problems. Buffer-bloat is very much our enemy in conferencing, driving up the RTT and thus the inevitable delay between speaker and hearer. CoDel should be encouraged.

At the network layer, we'd also do well to standardize a back-pressure signal at typical uplink queues (not ICMP source-quench, but perhaps a packet-delay metric). That's almost always where the worst bandwidth mismatch occurs: between ethernet speeds and uplink speeds; and this is where bufferbloat does its worst. It's also where the greatest benefit would flow from backpressure. A backpressure signal could typically reach the sender in about one millisecond, as opposed to 50 or more milliseconds typically found in the round-trip path of an ECN signal, least of all the hundreds of milliseconds needed to infer packet loss.

In real-time conferencing traffic, our delay budget is quite limited -- 150 milliseconds of one-way delay will noticeably impair communications. Codecs often eat 50 of those, and something has to be done about jitter, which can easily exceed 100 milliseconds. We have no delay budget left! Thus, a back-pressure signal in one millisecond instead of 100 would be a really significant win.

Backpressure has become a dirty word, largely because of insecurity of packets which might lead to backpressure signals. But I posit that the uplink case is easy enough to secure: a low Time-To-Live limits the danger, and the uplink should have a really good idea of which addresses are on the

customer side. Besides, the backpressure signal would be strictly advisory and safely ignored by any application not concerned about delay. (I do not mean that such applications couldn't cause problems to conferencing uses, but rather that controlling this damage is a strictly local problem not requiring standards work.)

4.
My fourth point belongs at Application layer, but would have been premature to mention before giving a background in transport and delay budget. I frankly don't believe we can keep lip-sync between audio and video without explicit frame timing data.

There is a tendency in WGs to repeat the mantra that all we need to do about lip-sync is to carry the audio and video in the same transport. This theory falls apart pretty badly as frame-rates of video change and possibly audio codecs also change -- both of which will occur as the application layer adjusts to congestion.

Thus, I claim we need somewhat-explicit timing signals, both for video frames as received and reconstituted and for audio samples as received and reconstituted. Then it should be up to the receiving application to match them "close enough". What will be "close enough" will also vary by individual; and I'm sure different vendors will want to use different approaches. There are audio tricks involving pitch-shifting which are very helpful in some situations but distracting in others. Omitting or repeating video frames definitely gets into "glitch" territory, but is often the least distracting alternative. Any such choices belong at the receivers' application.

5.
To summarize, work is needed at network layer to improve the timeliness of congestion signals and make ECN more attractive for vendors to enable. Work on "backpressure" from customer uplinks could be quite separate from other work. Both ECN and backpressure will involve some changes at transport layer to be fully effective; but work could proceed at network layer only.

Work is needed at transport layer to establish "heartbeat" feedback which can be "sufficiently reliable" without explicit retransmission. Proper use of ECN in non-TCP transports deserves some work, as well as work on what signals to application layer would be most helpful. Our real-time transports will have to mostly trust the application layer to reduce data-rate when asked (though they can still limit outgoing data-rate).

Work is needed at application layer to properly negotiate what tools for data-rate reduction may be called into use (without further negotiation, which would lead to unacceptable delays). A lot can be accomplished by simply varying video frame rate. For virtually any data-rate reduction, IMHO, there will need to be (derivable) explicit synchronizing timing signals for each video frame and each audio sample.

With this combination of work at different layers, I believe we can hope for fully acceptable conferencing solutions over the public Internet within five years. Without combined efforts, I am pessimistic we'll get there in twenty.

--
John Leslie <john@jlc.net>