Mick Seaman Proposal for Enhancement to Raj's FECN
(Date: 01/24/2007)

Rate Reports

The basic idea is that, as part of a Congestion Avoidance algorithm comprising three sets of algorithms for Sources, Bridges, and Destinations, the Destination originates and transmits regular Rate Report (RR) frames to each active Source. Each RR traces the reverse path from the Destination to the Source and carries an advertised rate for use by the Source in transmitting to that Destination. The RR originally carries a rate set by the destination to be its receiving link speed. At each Bridge Port, if the rate that that port wishes to advertise for the S->D direction is lower than the RR rate, the latter is replaced in the RR frame by that lower rate.

The principle purpose of this mechanism is to improve control loop feedback, by ensuring that feedback is received regularly for all destinations, and to allow the feedback to be provided as a potential conversation starts, instead of relying on a statistical chance of sampling a forward going frame to trigger the feedback. The latter (Bridge sampling) naturally means that traffic has to be sent at a high rate simply to improve the chance of feedback, which essentially means increasing the chance of congestion and loss is necessary to gain the feedback to avoid it. This does not seem optimal. The overall goal is to reduce the overall control loop delay and provide early feedback to the point that no additional link level mechanism (especially PAUSE) is required to achieve acceptably low loss probability.

The Destination algorithm is to generate RRs every so many received bytes on an 'active' connection, and to generate an initial RR when a connection transitions from 'idle' to 'active'. By a connection in this sense I mean a particular {SA, DA} tuple, and such tuples are created a 'soft state' at Source and Destination in response to the frame flow. The overall RR generation algorithm is to be chosen to have an overhead of less than 1% bandwidth. As a first cut at the Destination algorithm, the 'idle' to 'active' transition occurs when two frames from the same Source are received within some number of mfts (maximum frame times - i.e. the time taken to transmit a maximum sized frame). The 'active' to 'idle' transition occurs after a time elapses with no reception , and the RR is sent about every 10 mf bytes (i.e. 15 Kbytes) when the connection is active. That's about a 0.5% overhead. I think it is likely that RRs could be sent less frequently but haven't tried lesser numbers yet.

The Source algorithm also treats connections as 'idle' or 'active', with an 'idle' connection being one for which no recent RR has been received. A low rate is associated with an 'idle' connection (perhaps 5 Mb/s = 1 max frame per 200 mfts on a 1 Gb/s link), and the rate is only updated when an RR is received. So a new or idle connection receives a low rate for the first few frames, which then stimulate the generation of an RR which reports the rate for the link. The Source rate is then increased towards that reported rate, with the RR rate diminishing as the new connection receives its share. The same RR rate is advertised to all sources, although any given source can behave as a number of (or as a fractional) virtual source.

I have only considered this sort of algorithm in terms of reporting rate feedback so far, though it is possible that the same idea of providing more predictable feedback per connection or conversation is applicable to feedback couched in other terms, and that the most important aspect - that of providing timely feedback on conversations that are just starting to be active so that they do not have to damage the network by injecting excess traffic to get congestion reports - may also be transferable.

The Bridge Port can calculate the rate to be placed in the RR packets (or other information as appropriate) periodically or upon some reasonably infrequent stimulus that does not require the

RR to be updated with information that has only become available just as the RR is received. This simplifies the RR update process to one for checking the (reserved) Ethertype for the RR, comparing the rate with that held by the Bridge Port, and overwriting if required. No new frames are injected into the stream of frames processed by this mechanism, and the precalculated rate held by the port can be that appropriate to the sum of the ports in a link aggregation, I believe that should simplify the Bridge/Bridge Port architecture as compared to an architecture that requires rapid injection of frames into the stream.