Congestion Control

Simon Foucher 260 223 197

Amine

Ivan

Congestion control focuses on proper network resources allocation such that when the demand is greater than the available resources, the network operates in an optimal way. These resources include bandwidth of links, buffers and node processing capacity. There are many myths and speculations about new technologies believed to solve congestion issues, and here are the main ones.

 ‘Infinitely’ large buffer space, possible with cheaper memory, has been found to produce more harm than good. Even though they solve the issue of packet drops, larger queues can get so big that packets timeout while in the queue, in which case, upper layer protocols unnecessarily resend them. Introducing high speed links into a network is also not guaranteed to help performance; as long as the low speed links remain, the congestion point will simply be relocated to just before the low speed link. This is further supported by Amhdal’s law. Similarly, introducing high speed processors will, by the same argument, not help a congestion issue.

 Based on these arguments, one could stipulate that congestion can be avoided by balancing the speeds of participants on a network, but this is not the case since congestion may result in a bottleneck network configuration.

 Congestion can be classified as a single or distributed resource problem. Single resource congestion may involve dumb resources, such as wire availability in LANs, in which case a solution has to be provided by end users. It can also involve intelligent resource, like a name server, which can allocate itself appropriately. The problem is much harder to solve in distributed resources, for example the links between users.

We can also divide congestion control in 2 classes: those who decrease the demand, which involve the user, and those which increase the available resources which can be implemented dynamically by the network under heavy load without user involvement. Demand reductions are divided in 3 classes: service denial, service degradation, and scheduling, which is a special case of service degradation where the user is asked to resend a demand at a later time when the network is less busy.

 All these schemes require the network to measure load, where feedback information from a congested point is sent to an upstream control point that takes remedial action. There are many ways to gather feedback: send a choke feedback packet, but this might cause more congestion, feedback about congestion added to routing message, but this causes too many messages to be sent because of fast congestion changes, rejecting further traffic, source sending probe packet to measure congestion, and adding a feedback field into packets.

 There have also been proposal locations for control to be implemented, which are: transport layer, since the traffic is generated by the end systems, network access, like traffic lights on networks that are only accessible if they are not congested, network layer, where routers when overloaded can reduce service to incoming links and spend more time flushing buffers, and data link layer, using backpressure on buffer exhaustion.

 There are lots of requirements for congestion control which make it difficult for a solution to be satisfactory. It must have low overhead, it must be fair and avoid starvation, it must be responsive such that it matches the demand dynamically with the available resources, it must work in bad environments and it must allow total network performance to be maximized.

 Any network design decision affecting the load or the resources allocation is part of congestion control. There are 2 types of networks: connection oriented where intermediate nodes reserve resources for the session and connectionless which offer more flexibility since they can easily be reconfigured. Maintaining queue fairness amongst links of a gate does not provide fairness for users going to different destinations. Fairness might be implemented using a round robin through the queues, but this way, larger packets get greater bandwidth. With regards to packet dropping, it might be preferable to drop packets at the head of the queue if they are part of a file transfer and the newly arrived packet is part of some real time application. In path splitting, only paths with equivalent bandwidth as the original one will typically be selected, such that there can be low bandwidth path that are virtually unoccupied while high bandwidth paths are really congested.

 Finding a good algorithm for estimating round trip delays have been the first step towards congestion control, since it helps adjusting packet lifetime policies which might lead to unnecessary retransmission of packets which are assumed to have been dropped but are just stuck in the congested network.

 There are 2 types of packet flow control: window based, where the destination decides the number of packets a source can send, and rate based when the destination specifies a rate of packets per seconds that can be transmitted. The choice is usually made to match the bottlenecked resource flow control on the network such that it is not overrun. All these policies can also be applied at the data link layer on a hop per hob basis.

 It is important to design solutions with control frequencies equal to feedback frequencies. For shorter congestion period, it is better to implement data link and network control systems. For longer congestion, session control should be used, and finally for indefinite period congestion, adding more resources to the network is the key. The author proposes 3 congestion control schemes.

 Timeout-Based Congestion Control exploits the idea that packet loss is a good indicator of congestion. When this happens, load on the network should be reduced. After the congestion, if the user receives W packets without a drop, it should increase its window size to W+1.

 The second idea observes response time and throughput. Under small loads, throughput increases with load increases. When the load reaches network capacity, throughput stops increasing, and this is called ‘the knee’. Further load increase builds up queues and packets start to drop. When the load reaches ‘the cliff’, throughput suddenly has a sharp decrease with further load increases, due to buffers being filled up. DECbit Scheme for Congestion Avoidance allows a network to operate in the knee, whereas conventional congestion control stops a network from falling in the cliff.

 Because networks architecture may differ from one AS to the other, feedback messages might not be compatible. Also, more passive devices such as bridges and hubs don’t have the capacity to build packets. The author proposes to use implicit information rather than explicit feedback messages in his Delay Based Scheme for Congestion Avoidance.

 There is still a lot of research to be done in the congestion control fields. For example, in current networks, packets traveling to the same destination take the same path, or traffic is split equally between 2 optimal paths, which can cause congestion on those paths while other sub-optimal paths are left unused. Also, the fairness issue of giving priority to real time VoIP over regular traffic as telecom networks move towards ATM sharing internet resources is generating more and more debates. Finally, fine tuning of a packet’s time to live is yet to be solved; how long should a host wait before it assumes the packet has been dropped before issuing a new one. This time needs to be minimized to speed up communication, while times that are too short will cause the generation of new excess traffic by resending packets that are simply stuck in traffic. So overall, there is still a lot to be done in the field of congestion avoidance.