Chapter 6

Related Work

A great deal of research effort has been made to enhance TCP performance for dynamic, large, leaky pipes. There are several other approaches to improving TCP scalability to large pipes that require only sender-side modification, including scalable TCP, high-speed TCP [92], and Vegas-based fast TCP [15]. In these schemes, as in traditional TCP, packet losses are exclusively treated as congestion signals. Compared with the previous schemes, MCCP is equipped with the ability to better handle random errors in high-speed heterogeneous networks. Explicit control protocol (XCP) [18] is a well-designed congestion control scheme for high-speed, long delay networks. However, it requires cooperation from routers and receivers, making it difficult to deploy.

With the increase of short-lived web traffic, researchers realize that startup performance is important, especially over large pipes. The fast probing scheme in MCCP provides a realistic means to figure out the right ssthresh on the fly. A variety of other methods have been recently suggested in the literature to avoid multiple losses and to achieve higher utilization during slow-start. A larger initial cwnd, roughly 4 KB, is proposed in [70]. This could greatly speed up transfers with only a few packets. However, the improvement is still inadequate when BDP is very large, and the file to transfer is bigger than just a few packets. Fast start [106] uses cwnd and ssthresh cached from recent connections to reduce the
transfer latency. The cached parameters may be too aggressive or too conservative when network conditions change. In [41], Hoe proposes to set the initial $ssthresh$ to the BDP estimated using packet pair measurements. This method can be too aggressive when the bottleneck buffer is not big enough, or many flows are coexisting. TCP Vegas detects congestion by comparing the achieved throughput over a cycle of length equal to RTT, to the expected throughput implied by $cwnd$ and $baseRTT$ (the minimum RTT) at the beginning of a cycle. This method is applied in both slow-start and congestion avoidance phases. During slow-start, a Vegas sender doubles its $cwnd$ only every other RTT, in contrast with Reno's doubling every RTT. A Vegas connection exits slow-start when the difference between achieved and expected throughput exceeds a certain threshold. However, Vegas is not able to achieve high utilization in large bandwidth delay networks, due to its overestimation of RTT.

To deal with dynamic bandwidth, TCP-EBN is proposed in [26]. In TCP-EBN, a TCP sender increases or decreases its $cwnd$ according to a bandwidth estimate that is sent to it from routers. Thus, this scheme relies on router cooperation and is, therefore, not an "end-to-end" approach to the problem at hand. Also, the router has to either keep per-flow state to get an accurate estimate, with the resultant scalability problem; or assumes that flows share bandwidth fairly, which is not true often. Comparing to this scheme, MCCP only requires sender-side modification, thus, much easier to deploy.