Transport Layer Responsibility

- Congestion occurs at routers, so it is detected at the network layer. However, congestion is ultimately caused by traffic sent into the network by the transport layer. The **only effective way** to control congestion is for the transport protocols to send packets into the network more slowly.
- The Internet relies heavily on the transport layer for congestion control, and specific algorithms are built into TCP and other protocols.

18

The way that a transport protocol should regulate the sending rate depends on the form of the feedback returned by the network. Different network layers may return different kinds of feedback. The feedback may be explicit or implicit, and it may be precise or imprecise.

19

- An example of an explicit, precise design is when routers tell the sources the rate at which they may send. Designs in the literature such as **XCP (eXplicit Congestion Protocol)** operate in this manner (Katabi et al., 2002).
- An explicit, imprecise design is the use of **ECN (Explicit Congestion Notification)** with TCP**.** In this design, routers set bits on packets that experience congestion to warn the senders to slow down, but they do not tell them how much to slow down.

20

- In other designs, there is no explicit signal. **FAST TCP** measures the round trip delay and uses that metric as a signal to avoid congestion (Wei et al., 2006).
- Finally, **in the form of congestion control most prevalent in the Internet today, TCP with drop-tail or RED routers**, packet loss is inferred and used to signal that the network has become congested.

21

TCP Congestion Control

History: Starting in 1986, the growing popularity of the early Internet led to the first occurrence of what became known as a **congestion collapse**, a prolonged period during which goodput dropped precipitously due to congestion in the network. Jacobson (and many others) set out to understand what was happening and remedy the situation.

22

TCP Congestion Control: Detection

All the Internet **TCP algorithms assume that lost packets are caused by congestion** and monitor timeouts and look for signs of trouble the way miners watch their canaries.

- A good retransmission timer is needed to detect packet loss signals accurately and in a timely manner.

23

TCP Congestion Control: retransmission timer

For each connection, TCP maintains a variable, ***SRTT* (Smoothed Round-Trip Time),** that is the best current estimate of the round-trip time to the destination in question. When a segment is sent, a timer is started, both to see how long the acknowledgement takes and also to trigger a retransmission if it takes too long. TCP measures how long the acknowledgement took, say, *R*. It then updates *SRTT* according to the formula

$$SRTT = \alpha\, SRTT + (1 - \alpha)\, R$$

Typically, α = 7/8.

24

TCP Congestion Control: retransmission timer

Even given a good value of *SRTT*, choosing a suitable retransmission timeout is a nontrivial matter. Initial implementations of TCP used 2x*SRTT*, but experience showed that a constant value was too inflexible because it failed to respond when the variance went up.

25

TCP Congestion Control:
retransmission timer

To fix this problem, Jacobson proposed making the timeout value sensitive to the variance in round-trip times as well as the smoothed round-trip time. This change requires keeping track of another smoothed variable, *RTTVAR* (Round-Trip Time VARiation) that is updated using the formula

$$RTTVAR = \beta\ RTTVAR + (1 - \beta)\ |SRTT - R|$$

retransmission timeout, *RTO*, is set to be:

$$RTO = SRTT + 4 \times RTTVAR$$

typically $\beta = 3/4$

26

TCP Congestion Control
Congestion Window

TCP maintains a **congestion window** whose size is the number of bytes the sender may have in the network at any time. The corresponding rate is the window size divided by the round-trip time of the connection.

27

TCP Congestion Control:
TCP Tahoe (Slow Start)

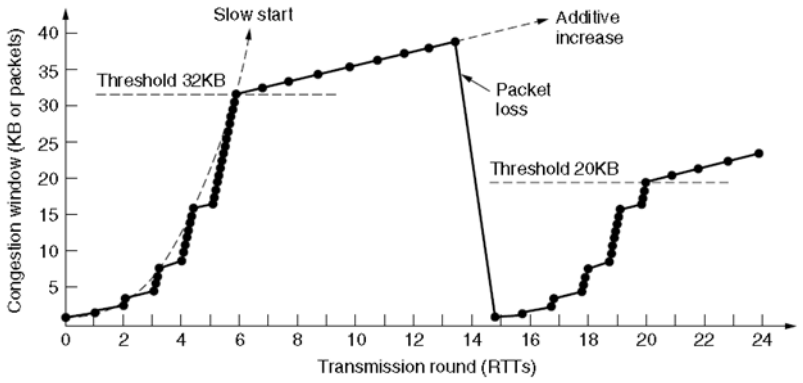

Figure 6-46. Slow start followed by additive increase in TCP Tahoe.

TCP Tahoe (which included good retransmission timers) provided a working congestion control algorithm that solved the problem of congestion collapse.

28

TCP Congestion Control:
TCP Reno (Sawtooth behavior)

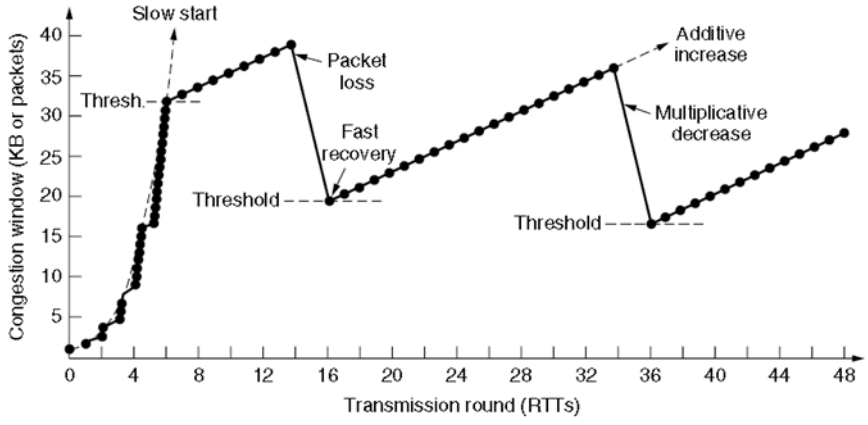

Figure 6-47. Fast recovery and the sawtooth pattern of TCP Reno.

29

Backup Slides

30