Network : Congestion Control

Jae Hyeon Kim

Reference

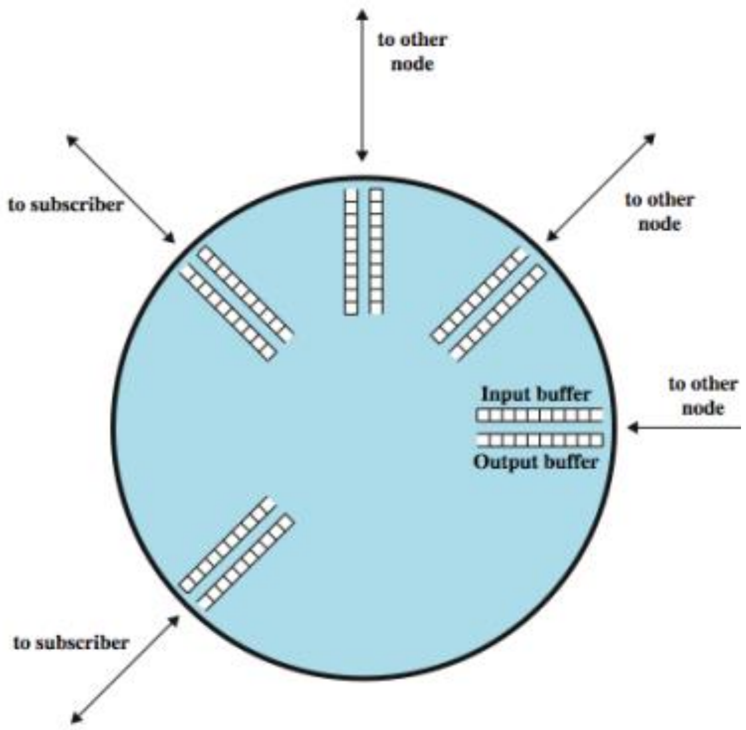
William Stalling, Data and Computer Communications 10/E, Prentice Hall

Congestion Control

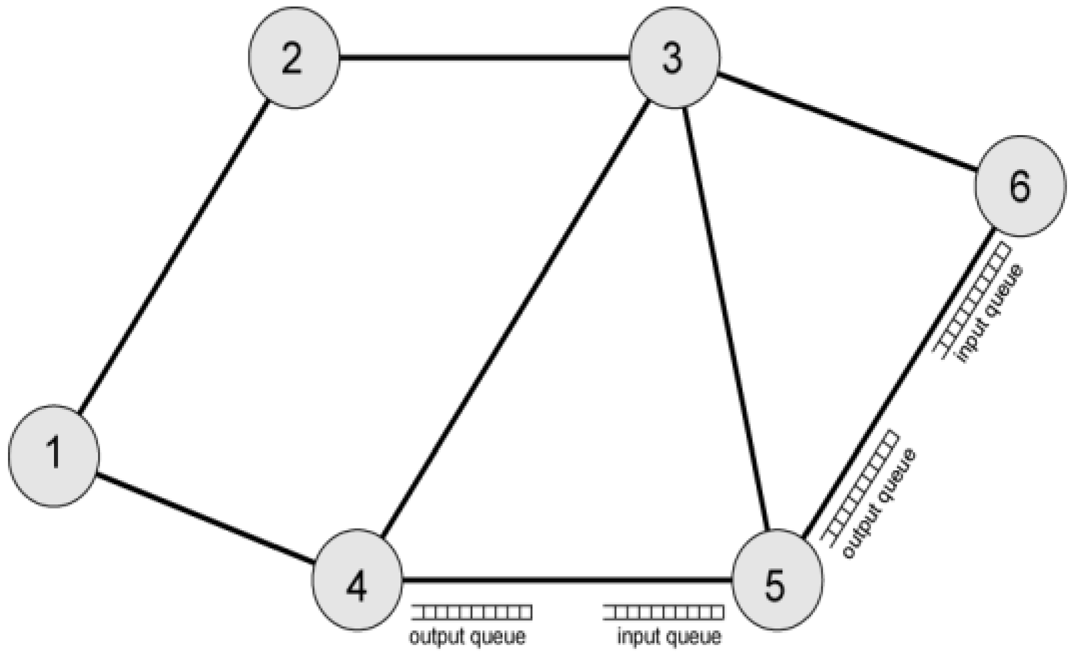
- Congestion occurs when the number of packets being transmitted through the network approaches the packet handling capacity of the network
  - Congestion control aims to keep number of packets below level at which performance falls of dramatically
- Data network is a network of queues
  - Generally 80% utilization is critical
- Finite queues mean data may be lost

Finite Queues

- Queue at a node



- Interaction of queues

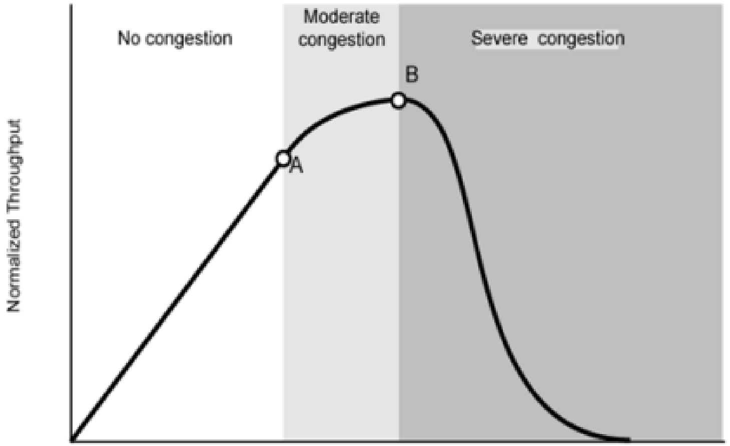


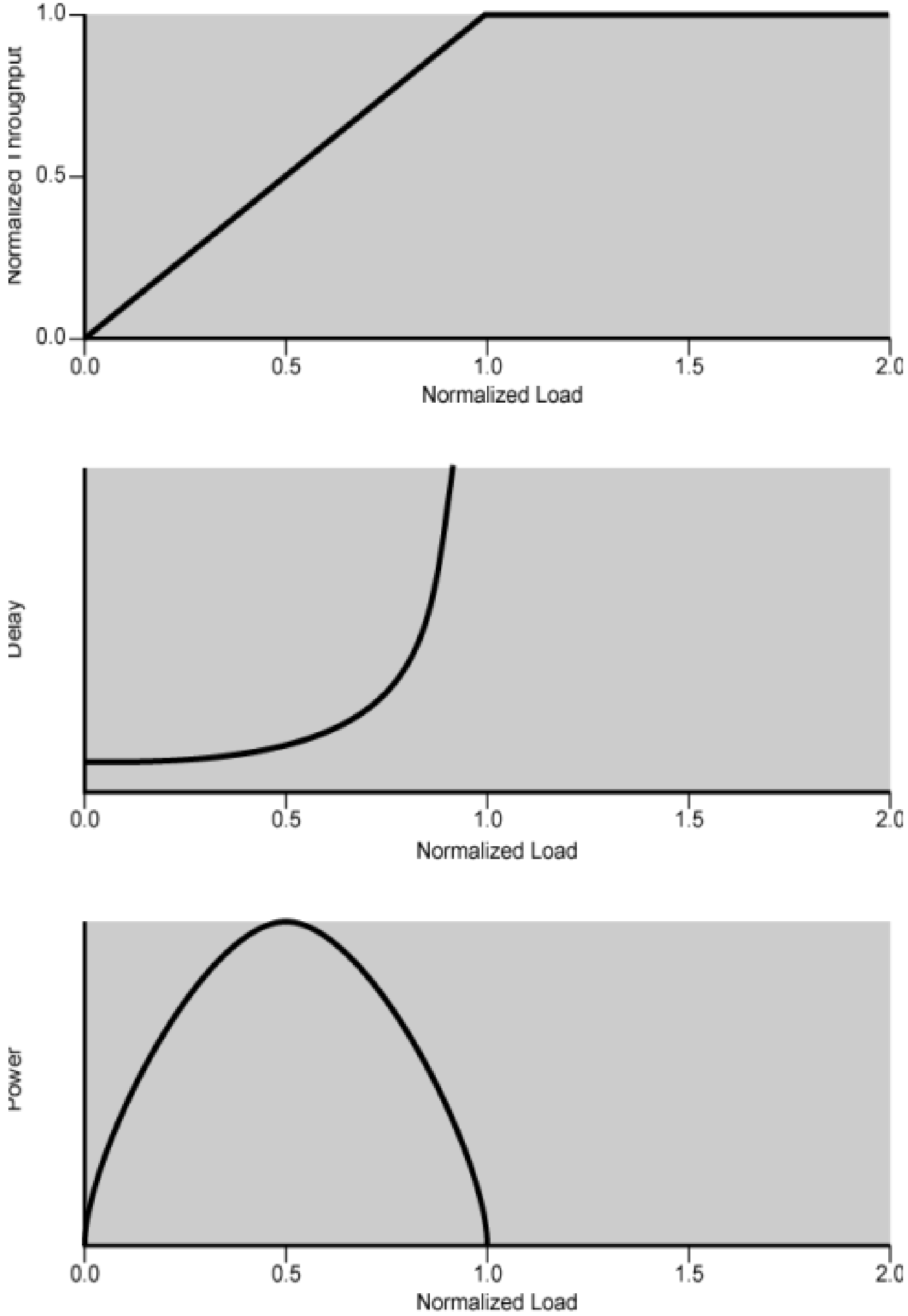
Effects of Congestion

- What typical works on a router
  - Packets arriving are stored at input buffers
  - Routing decision made
  - Packet move to output buffer
- Packets queued for output transmitted as fast as possible
  - Statistical time division multiplexing
- If packets arrive to fast to be routed, or to be output, buffers will fill
  - Can discard packets
  - Can use flow control : propagate congestion though network

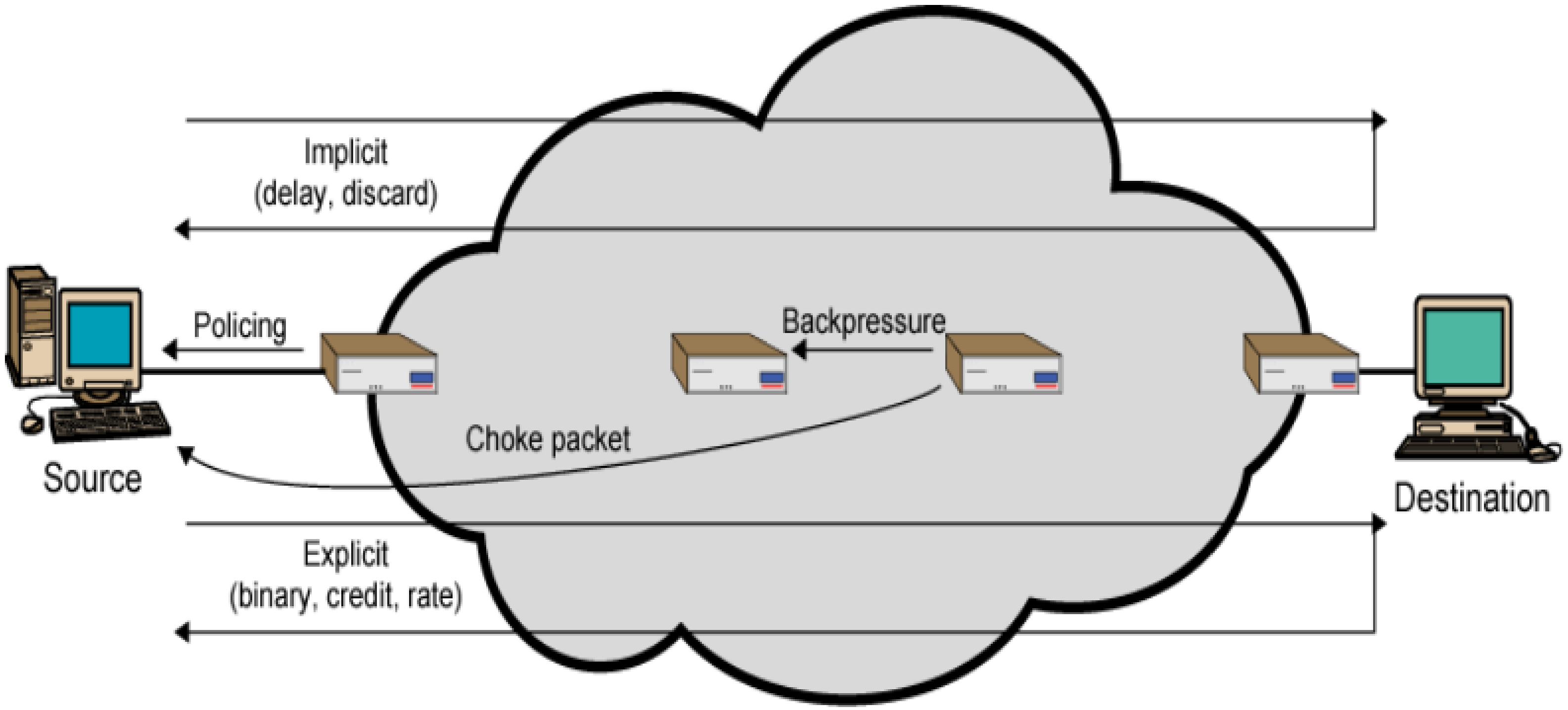
Idel vs. Practical Performance

- Practical performance
  - Ideal assumes infinite buffers and no overhead
  - But, buffers are finite
  - Overheads occur in exchanging control msgs
- Effects of congestion
  - With no control





– Mechanisms for Congestion Control



Backpressure

- If a node becomes congested it can slow down or halt flow of packets from other nodes
  - Then, other nodes have to apply control on incoming packet rates
  - Flow restriction propagates backward to sources, which are restricted in the flow of new packets into the network
- Can be exerted on the basis on links or logical connections
  - Can be selectively applied to logical connections so that the flow from one node to the next is only restricted on some connections
- Only recently developed for IP

Choke Packet

- Control packet
  - Generated at congested node
  - Sent to source node
- An example is Internet Control Message Protocol (ICMP) source quench packet
  - From router or destination end system
  - Source cuts back until it no longer receives quench messages
  - Message is issued for every discarded packet
  - Message may also be issued for anticipated congestion
- Crude control technique

Congestion Signaling

- Implicit congestion signaling
  - With network congestion, transmission delay increases, and/or packets may be discarded
  - Source can detect congestion and reduce flow
  - Effective on connectionless (datagram) networks
- Explicit congestion signaling
  - Network alerts end systems of increasing congestion
  - End systems take steps to reduce offered load
  - Backwards : congestion avoidance in opposite direction
  - Forwards : congestion avoidance in same direction

Explicit Signaling Categories

- Binary
    - A bit set in a packet indicates congestion
- Credit based
    - Indicates how many packets source may send
    - Common for end to end flow control
- Rate based
    - Supply explicit data rate limit
    - Nodes along path may request rate reduction

Traffic Management

- Fairness
  - Provide equal treatment of various flows
- Quality of Service (QoS)
  - Different treatment for different connections
- Reservations
  - Traffic contract between user and network
  - Excess traffic discarded or handled on a best-effort basis

Congestion Control in Packet Switched Networks

- Send control packet to some or all source nodes
  - Requires additional traffic during congestion
- Rely on routing information
  - May react too quickly
- End to end probe packets
  - Adds to overhead
- Add congestion info. To packets as they cross nodes
  - Either backwards or forwards
- So, two categories of TCP congestion control
  - Retransmission timer management
  - Window management

Retransmission Timer Management (1)

- As network conditions change, a static retransmission timer is likely to be either too long or too short
  - All TCP attempt to estimate the current round-trip time and then set the timer to a value somewhat greater than the estimated time
  - Simple average

    ARTT(k+1) = k*ARTT(k)/(k+1) + RTT(k+1)/(k+1) ARTT : Average RTT

  - Exponential average : RFC 793

    give greater weight to more recent instances because they are more likely to reflect future behavior

    SRTT(k+1) = α * SRTT(k) + (1- α) * RTT(k+1) SRTT : Smoothed RTT

    RTO (k+1) = Min(UB, Max(LB, β * SRTT(k+1))) RTO : retransmission timeOut, UB : Upper Bounds, LB: Lower Bound, typically 0.8<α<0.9, so, 0.875, 1.3<β<2.0

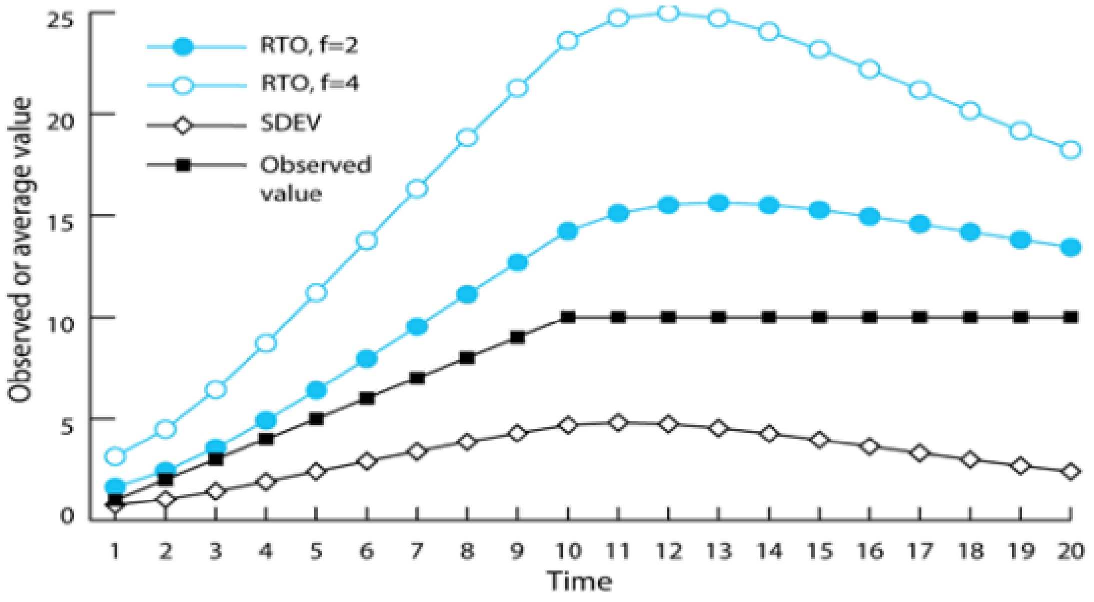
Retransmission Timer Management (2)

- Jacobson’s algorithm
  - RTT exhibits a relatively high variance
  - With low variance of RTT, RTO is too high, whilst in an unstable environment, β = 2 may be inadequate with unnecessary retrans.
  - Again, give greater weight to more recent instances because they are more likely to reflect future behavior

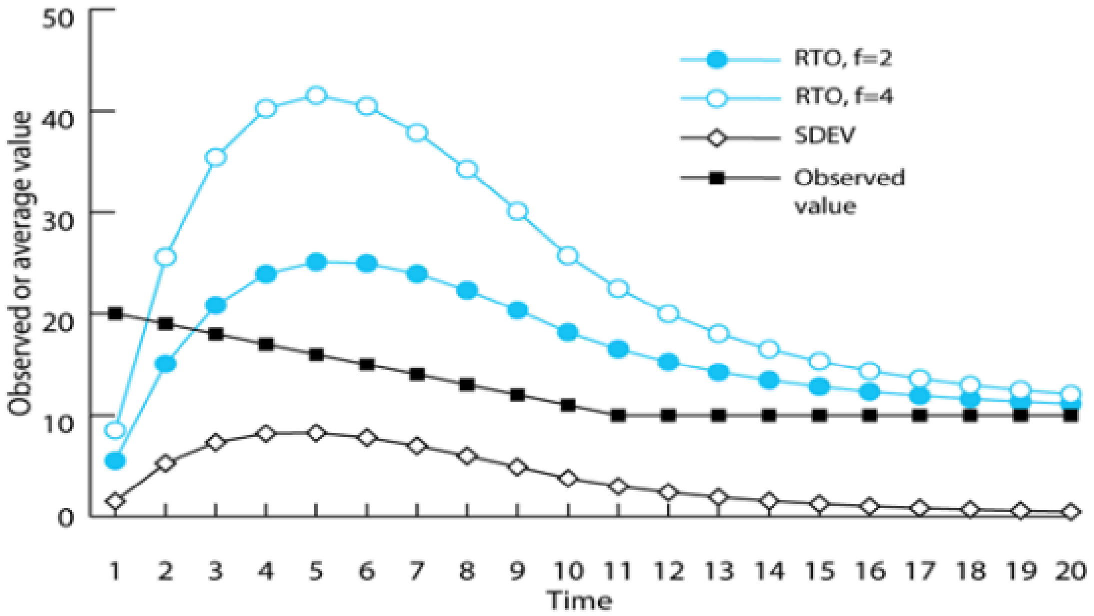
$$SRTT(k+1) = (1-g) * SRTT(k) + g * RTT(k+1)$$
$$SERR(k+1) = RTT(k+1) - SRTT(k)$$
$$SDEV(k+1) = (1-h) * SDEV(k) + h * |SERR(k+1)|$$
$$RTO(k+1) = SRTT(k+1) + f * SDEV(k+1)$$

typically $g = 0.125$, $h = 0.25$, $f = 4$

Jacobson's RTO Calculation



(a) Increasing function



(b) Decreasing function

Exponential RTO Backoff

- When a TCP sender times out on a segment, it must retransmit that segment
  - Timeout probably due to congestion, such as dropped packet or long round trip time
- Hence, maintaining the same RTO is not good idea
- A more sensible policy dictates that a sending TCP entity increase its RTO each time a segment is re-transmitted

RTO = q*RTO

commonly q=2 (binary exponential backoff)

as in Ethernet CSMA/CD

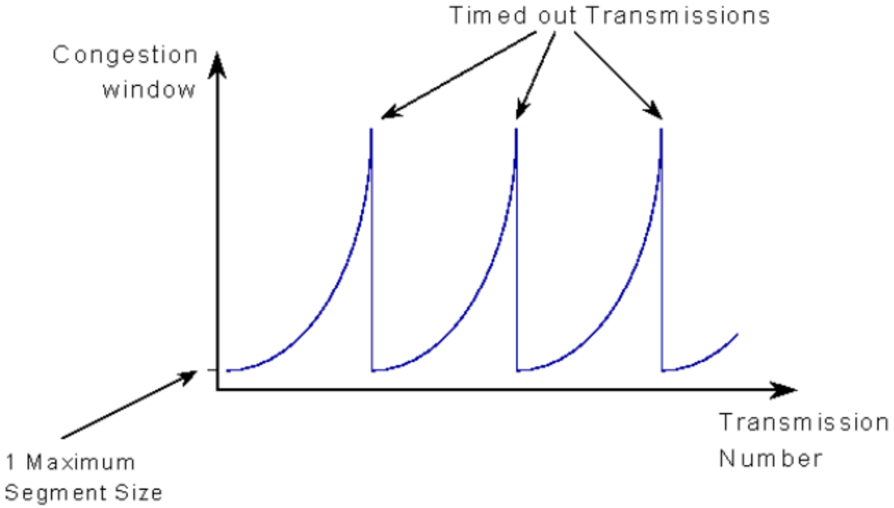
Karn's Algorithm

- If a segment is re-transmitted, the ACK arriving may be:
  - For the first copy of the segment (RTT longer than expected)
  - For second copy
- No way to tell for which one
- Do not measure RTT for re-transmitted segments
- But calculate backoff when re-transmission occurs
- Use the backoff RTO until ACK arrives for segment that has not been re-transmitted

Window Management (1)

- The size of sending window can have a critical effect on TCP performance in front of a congestion
- Slow start : gradually expanding the window
  - awnd = MIN[credit, cwnd]
  - Start connection with cwnd = 1
  - Increment cwnd by 1 (actually 2) at each ACK, to some max

Where credit = initially negotiated window size
cwnd = congestion window size
awnd = allowed window size



Window Management (2)

- Dynamic windows sizing on congestion
  - Jacobson points out that “it is easy to drive a network into saturation but hard for the net to recover”
  - With the slow start, cwnd keeps growing exponential until it becomes equal to receiver window (credit)
  - However, for the congestion, the exponential growth of cwnd may be too aggressive and may worsen the congestion

whenever a timeout occurs

set slow start threshold to half current congestion window

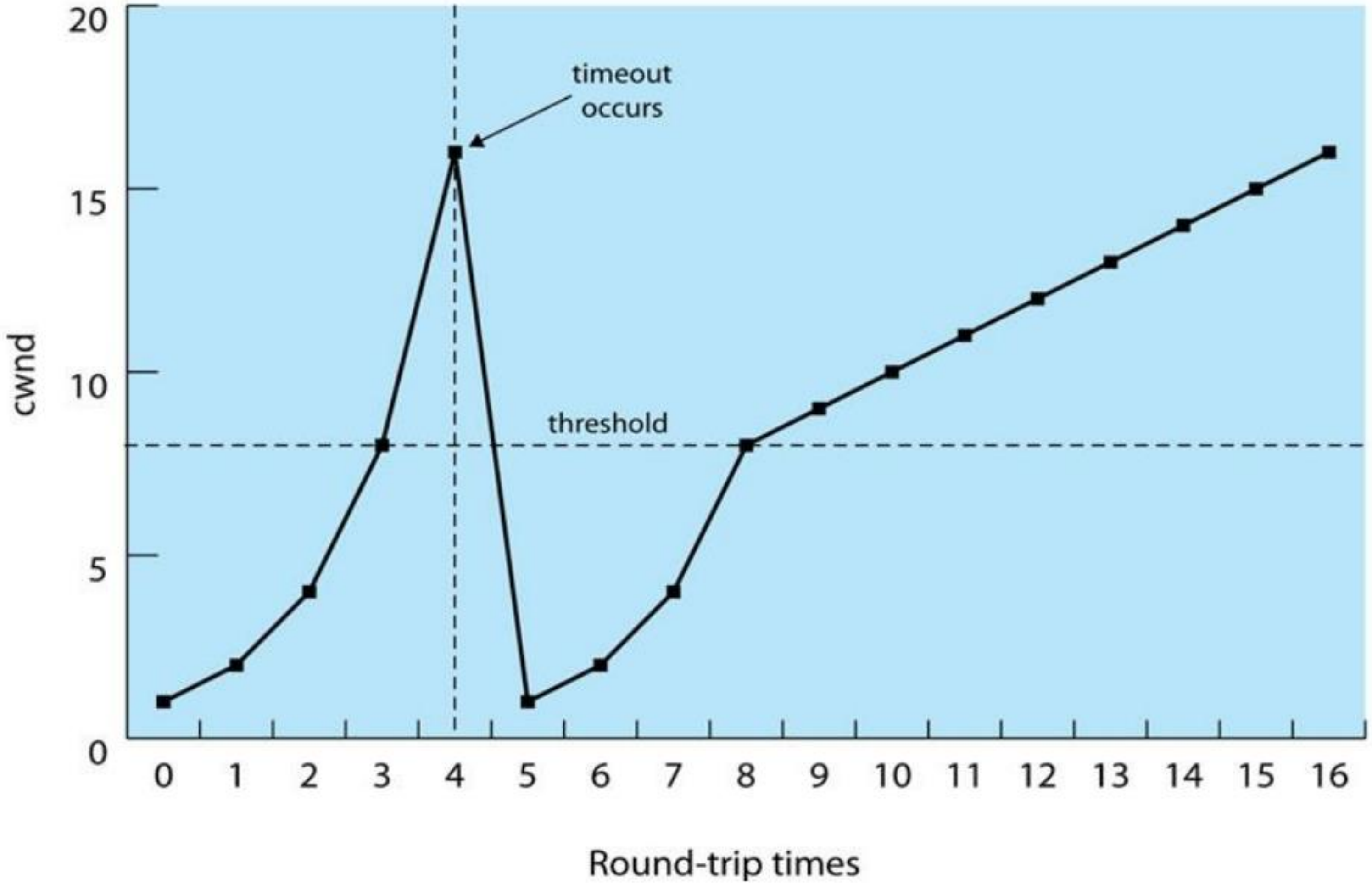
- ssthresh=cwnd/2

set cwnd = 1 and slow start until cwnd=ssthresh

- increasing cwnd by 1 for every ACK

for cwnd >=ssthresh, increase cwnd by 1 for each RTT

Illustration of Slow Start and Congestion Avoidance



Implementation of TCP Congestion Control Measures

Measure	RFC 1122	TCP Tahoe	TCP Reno	NewReno
RTT Variance Estimation	✓	✓	✓	✓
Exponential RTO Backoff	✓	✓	✓	✓
Karn's Algorithm	✓	✓	✓	✓
Slow Start	✓	✓	✓	✓
Dynamic Window Sizing on Congestion	✓	✓	✓	✓
Fast Retransmit		✓	✓	✓
Fast Recovery			✓	✓
Modified Fast Recovery				✓