CS 372 Lecture #21

Reliable data transfer with TCP

- network congestion
  - causes
  - costs
  - control

Note: Many of the lecture slides are based on presentations that accompany *Computer Networking: A Top Down Approach,* 6th edition, by Jim Kurose & Keith Ross, Addison-Wesley, 2013.

Network Congestion

- As utilization approaches maximum, *delay* increases
  - Excessive traffic is called ***congestion***
- Cause: end systems are sending too much data too fast for network/routers to handle
  - Why do we blame the routers?
- Costs: delayed/dropped packets (buffer overflow at routers)
  - long delays from queuing in router buffers
    - causing queues to fill up … which causes incoming packets to be dropped
  - dropped packets
    - everything used for those packets … wasted
  - slow ACKs cause unnecessary retransmission
    - waste of resources … and retransmission compounds the problem
  - eventual *congestion collapse*

- Affected by, but different from flow control
- On top-10 list of important networking research topics

Relationship: delay ↔ utilization

- D is the *effective delay*
  - Also called *expected delay*
- U is the *network utilization*
  - range is [0 … 1]
- D_0 is the delay when the network has no other traffic (throughput is at its maximum)
  - i.e., the delay when U is zero.

$$D = \frac{D_0}{(1 - U)}$$

Relationship: delay ↔ utilization

- Example: Suppose that "no-traffic delay" $\mathbf{D_0}$ is 2 ms, and the network is used at 65% of capacity. (utilization $\mathbf{U}$ = 0.65)
- The *effective delay* is approximately

$$D = \frac{D_0}{(1-U)} \qquad \frac{2ms}{(1-0.65)} = \frac{2ms}{(0.35)} \approx 5.7ms$$

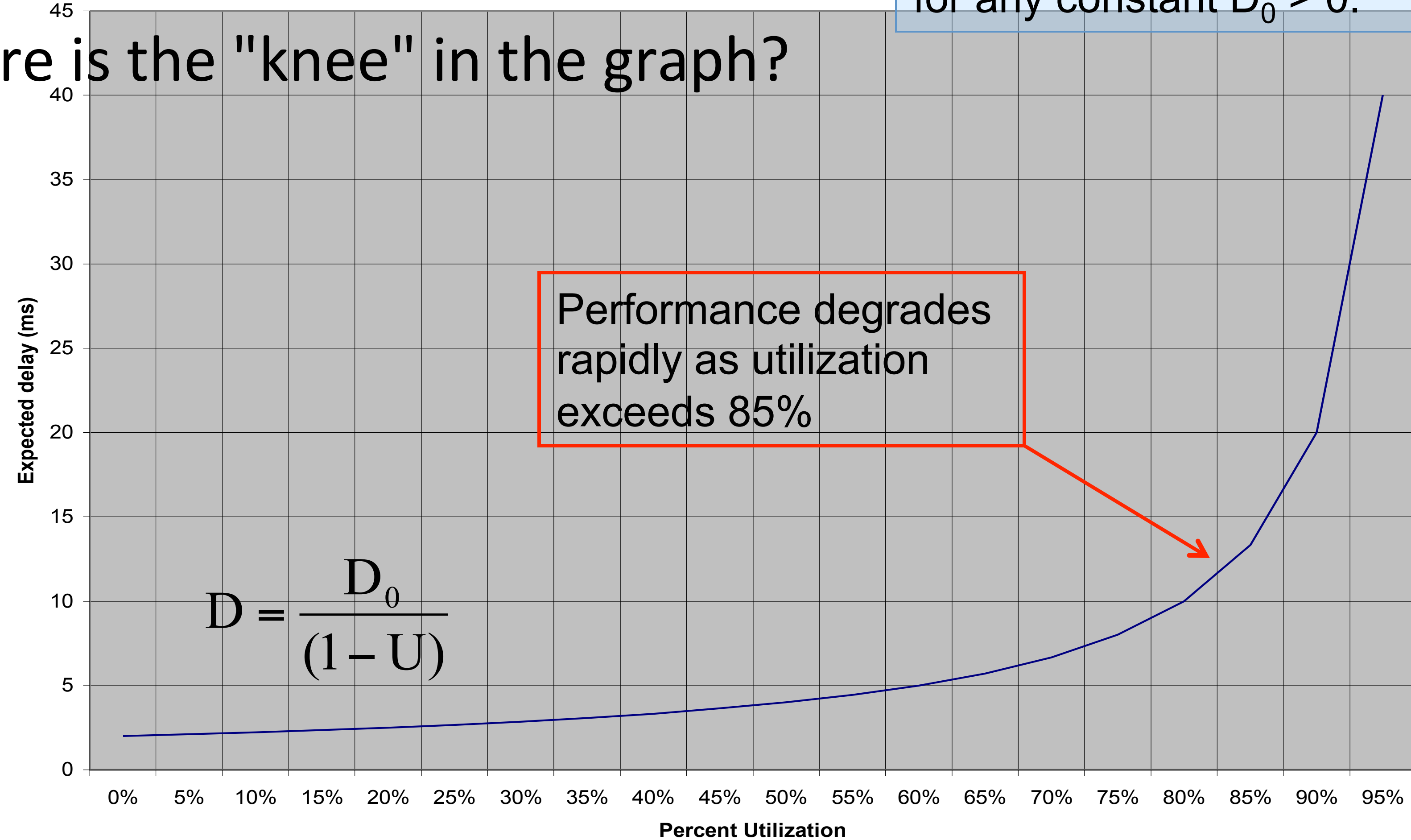
- When utilization is 0 (no traffic), $D = D_0$.
- When utilization is 1 (100% of capacity), *congestion collapse* is guaranteed.

Graphing the problem

Plot the *effective delay* for network utilization between 0% and 95%.

Where is the "knee" in the graph?

Note: Let $D_0 = 2$. The graph will look the same for any constant $D_0 > 0$.



$$D = \frac{D_0}{(1 - U)}$$

Congestion Control

- Goals:
  - Optimize utilization
  - Handle causes of network congestion
    - Retransmission handles only the symptoms
  - Detect congestion
  - Avoid congestion (when possible)
  - React to congestion (when necessary)

Optimizing utilization

- Smarter timeouts
- Better sliding-window size management
- More efficient acknowledgements

- Others?
  - Discussion topic

Smarter timeouts: TCP Round Trip Time (RTT) and Timeout

TCP count-down timer

- RTT:
  - can only be estimated
  - varies from one packet to another
- Timer too short:
  - premature timeout
  - unnecessary retransmissions
- Timer too long:
  - slow reaction to segment loss
- "just right" timer can improve efficiency

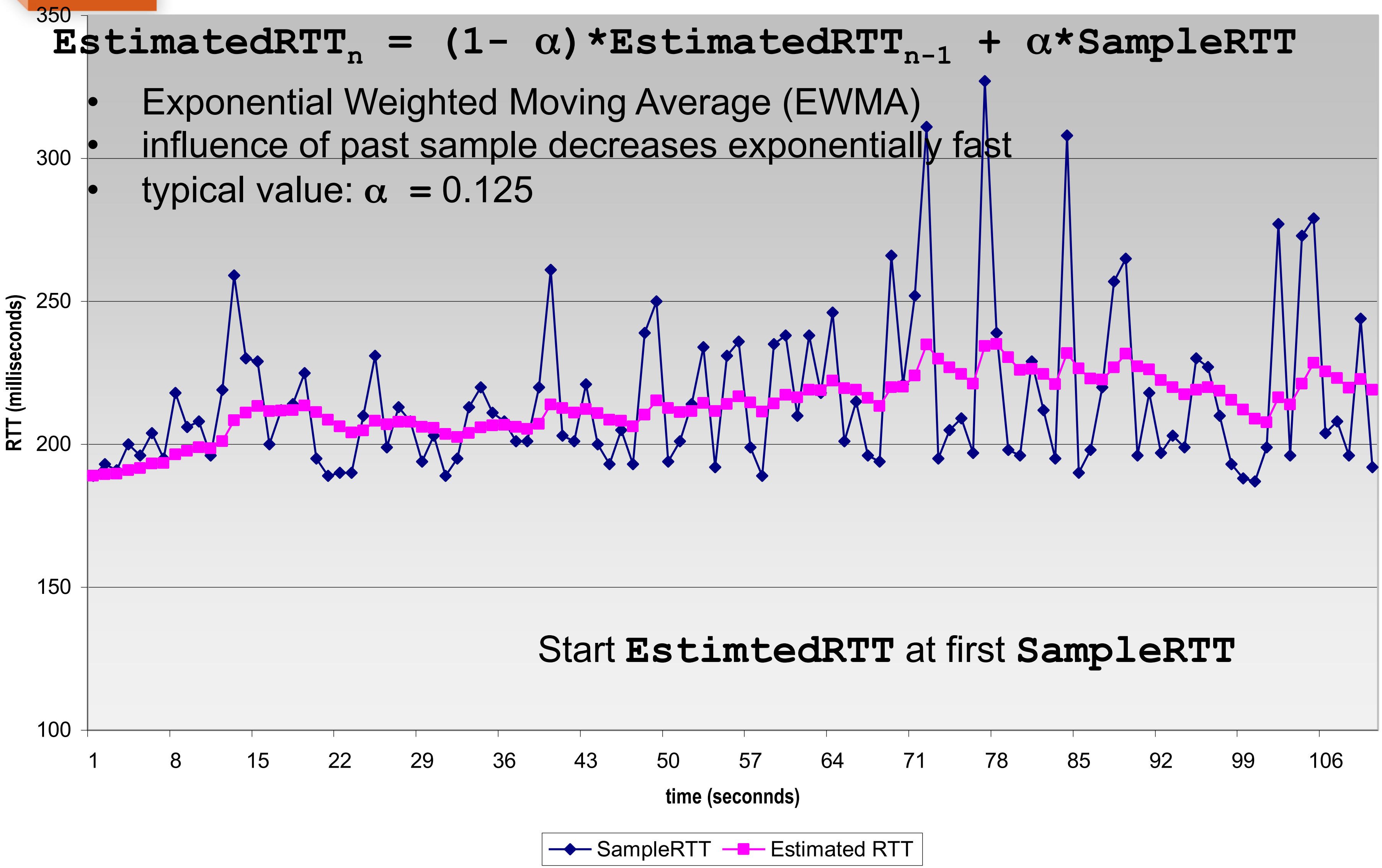
Estimating the RTT:

- `SampleRTT`: measured time from end of segment transmission until receipt of ACK
  - ignore retransmissions
  - can use handshake timing (plus)
  - however, `SampleRTT` can vary widely
- To get "smoother" estimated RTT:
  - average several recent measurements, not just current `SampleRTT`

TCP Round Trip Time and Timeout

$$\texttt{EstimatedRTT}_n = (1-\alpha)*\texttt{EstimatedRTT}_{n-1} + \alpha*\texttt{SampleRTT}$$

- Exponential Weighted Moving Average (EWMA)
- influence of past sample decreases exponentially fast
- typical value: $\alpha = 0.125$



Start **EstimtedRTT** at first **SampleRTT**

TCP Round Trip Time and Timeout

Setting the timeout

- Best to add a "safety margin" to **EstimtedRTT**
  - large variation in **EstimatedRTT** **->** larger safety margin

- see how much **SampleRTT** deviates from **EstimatedRTT**:

$$\mathbf{DevRTT_n = (1-\beta)*DevRTT_{n-1} + \beta*(SampleRTT-EstimatedRTT)}$$

(typically, $\beta = 0.25$)

Then set timeout interval:

TimeoutInterval = EstimatedRTT + 4*DevRTT

Summary Lecture #21

- Definitions
  - utilization
  - congestion, congestion collapse
  - effective delay
- Setting timeouts
  - `SampleRTT`, `EstimatedRTT`, `DevRTT`