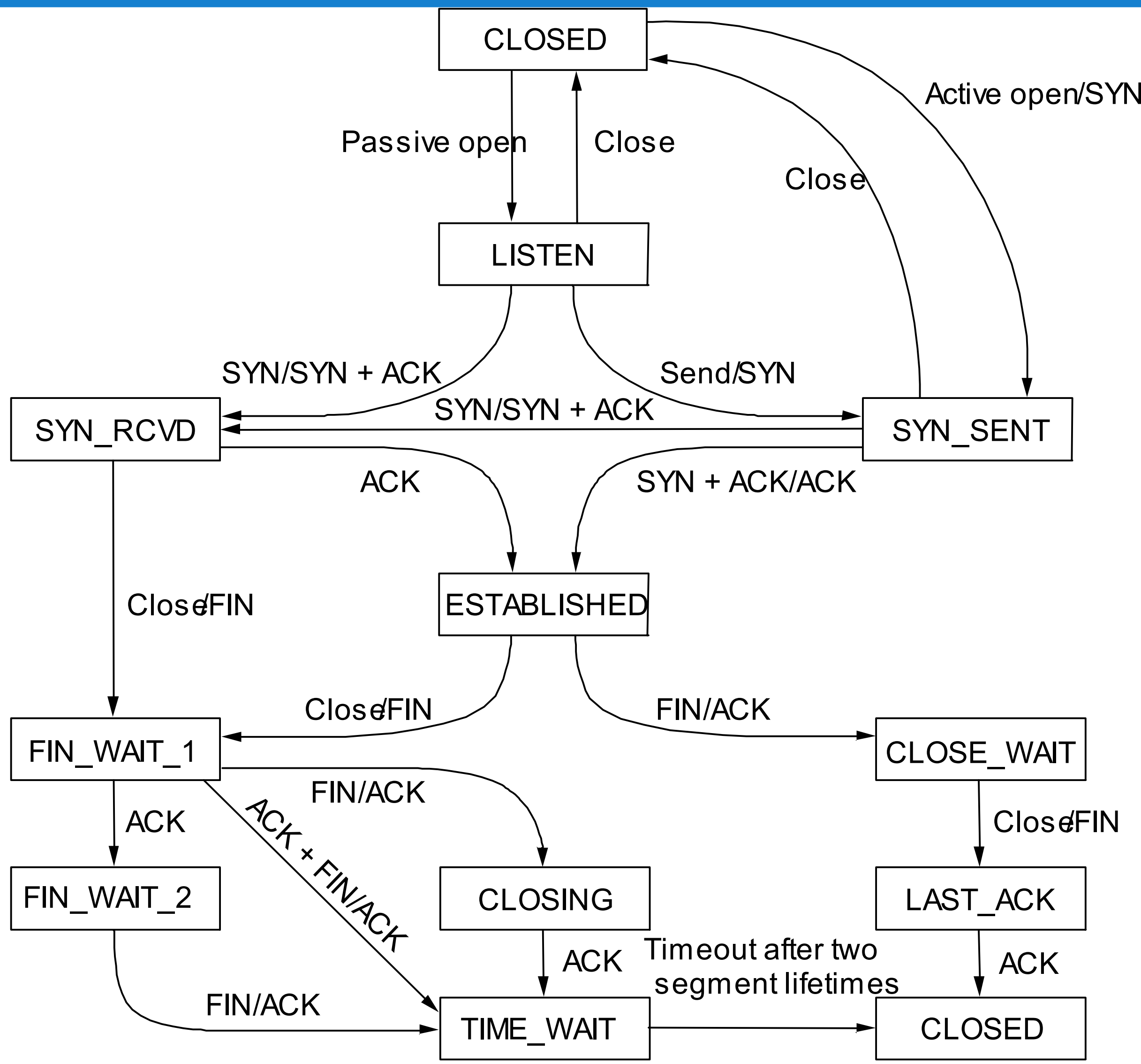
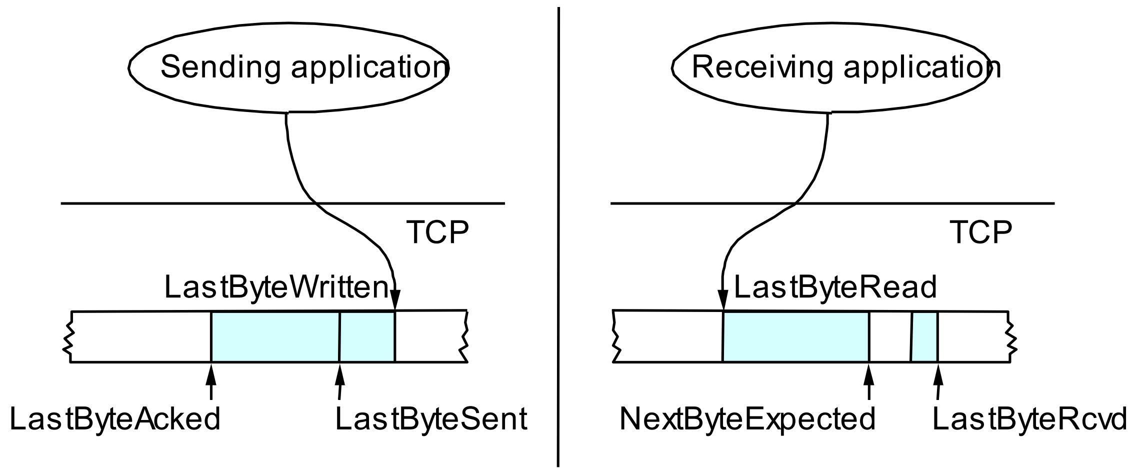
State Transition Diagram



Netstat -n in darwin.cc.nd.edu

```
127.0.0.1.60340      127.0.0.1.6019      32768     0 32768     0 CLOSE_WAIT
127.0.0.1.6019       127.0.0.1.60340     32768     0 32768     0 FIN_WAIT_2
127.0.0.1.60343      127.0.0.1.6019      32768     0 32768     0 CLOSE_WAIT
127.0.0.1.6019       127.0.0.1.60343     32768     0 32768     0 FIN_WAIT_2
127.0.0.1.60344      127.0.0.1.6019      32768     0 32768     0 CLOSE_WAIT
127.0.0.1.6019       127.0.0.1.60344     32768     0 32768     0 FIN_WAIT_2
129.74.250.114.22    129.74.98.159.62351  65535    47 25488     0 ESTABLISHED
129.74.250.114.22    67.176.34.217.3977   63148     0 24820     0 ESTABLISHED
129.74.250.114.60349 129.74.250.221.993   24820     0 24820     0 ESTABLISHED
129.74.250.114.22    66.254.224.43.3246   64500     0 24752     0 ESTABLISHED
129.74.250.114.60350 129.74.250.114.32775 32768     0 32768     0 TIME_WAIT
129.74.250.114.22    67.176.34.217.3993   63544     0 24820     0 ESTABLISHED
```

Sliding Window Revisited



- Sending side
  - **`LastByteAcked <= LastByteSent`**
  - **`LastByteSent <= LastByteWritten`**
  - buffer bytes between **`LastByteAcked`** and **`LastByteWritten`**
- Receiving side
  - **`LastByteRead < NextByteExpected`**
  - **`NextByteExpected <= LastByteRcvd +1`**
  - buffer bytes between **`NextByteRead`** and **`LastByteRcvd`**

Flow Control

- Fast sender can overrun receiver:
  - Packet loss, unnecessary retransmissions
- Possible solutions:
  - Sender transmits at pre-negotiated rate
  - Sender limited to a window’s worth of unacknowledged data
- Flow control different from congestion control

Flow Control

- Send buffer size: MaxSendBuffer
- Receive buffer size: MaxRcvBuffer
- Receiving side
  - LastByteRcvd - LastByteRead < = MaxRcvBuffer
  - AdvertisedWindow = MaxRcvBuffer - (NextByteExpected - NextByteRead)
- Sending side
  - LastByteSent - LastByteAcked < = AdvertisedWindow
  - EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAcked)
  - LastByteWritten - LastByteAcked < = MaxSendBuffer
  - block sender if (LastByteWritten - LastByteAcked) + y > MaxSenderBuffer
- Always send ACK in response to arriving data segment
- Persist when AdvertisedWindow = 0

Round-trip Time Estimation

- Wait at least one RTT before retransmitting
- Importance of accurate RTT estimators:
  - Low RTT -> unneeded retransmissions
  - High RTT -> poor throughput
- RTT estimator must adapt to change in RTT
  - But not too fast, or too slow!

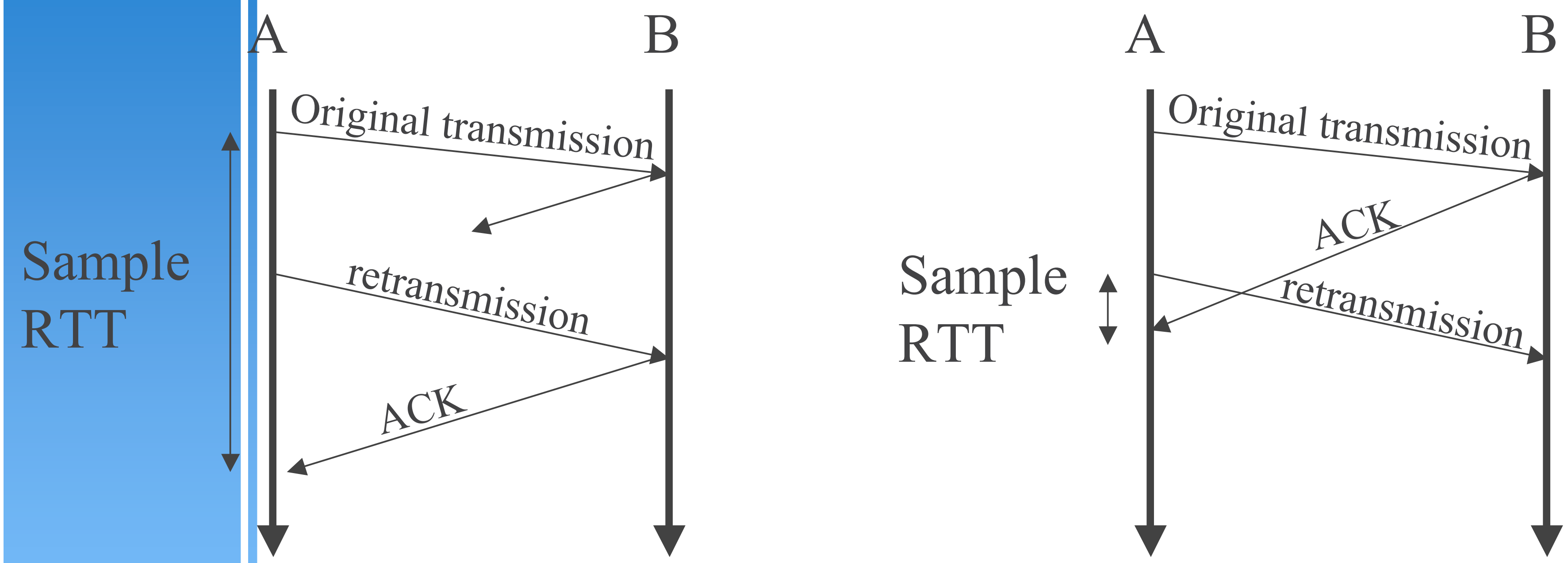
- Problem: If the instantaneously calculated RTT is 10, 20, 5, 12, 3 , 5, 6; what RTT should we use for calculations?

Initial Round-trip Estimator

Round trip times exponentially averaged:

- New RTT = α (old RTT) + (1 - α) (new sample)
- Recommended value for α: 0.8 - 0.9
- Retransmit timer set to β RTT, where β = 2
- Every time timer expires, RTO exponentially backed-off

Retransmission Ambiguity

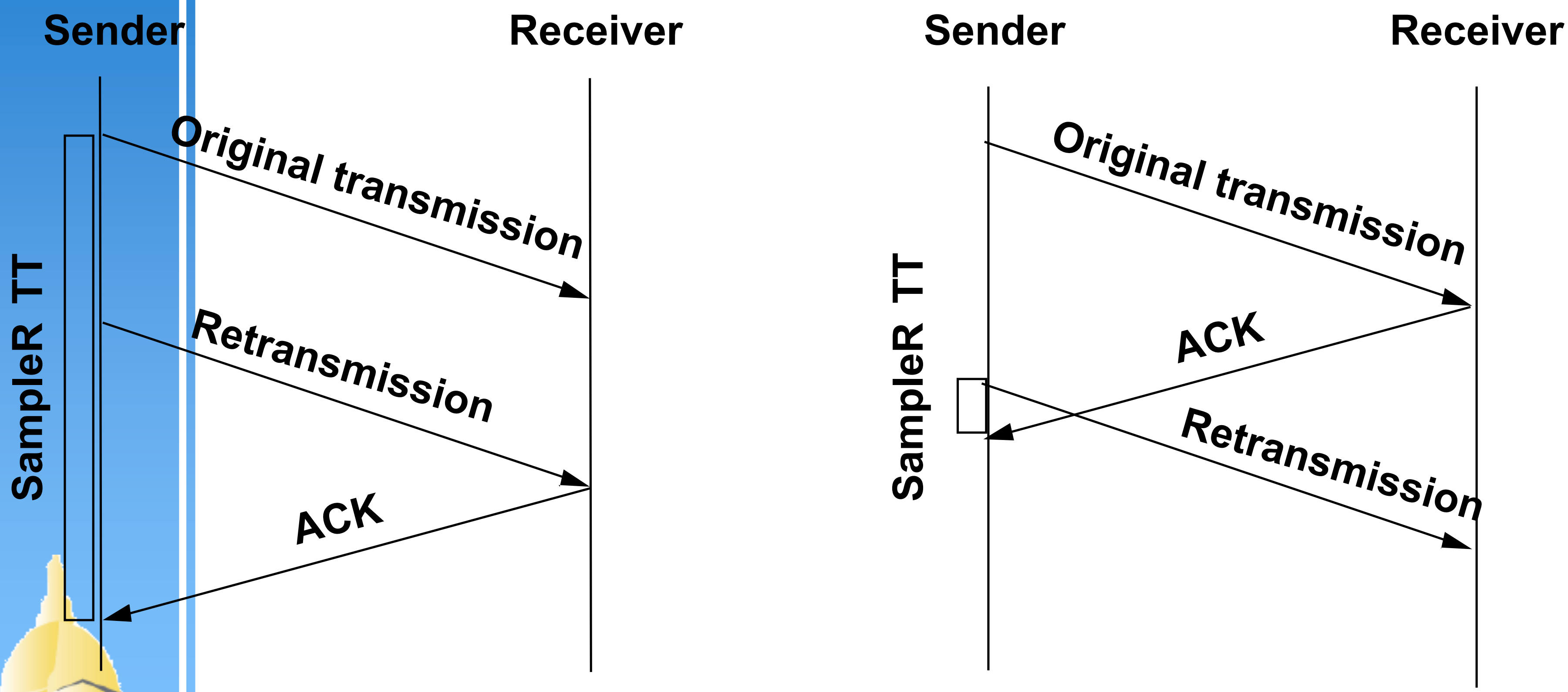


Karn's Retransmission Timeout Estimator

- Accounts for retransmission ambiguity
- If a segment has been retransmitted:
  - Don't count RTT sample on ACKs for this segment
  - Keep backed off time-out for next packet
  - Reuse RTT estimate only after one successful transmission

Karn/Partridge Algorithm

- Do not sample RTT when retransmitting
- Double timeout after each retransmission



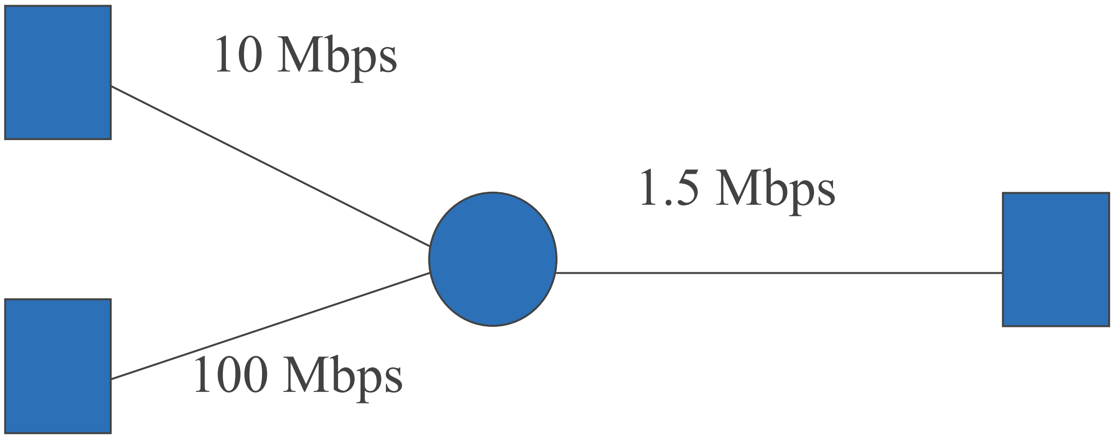
Jacobson's Retransmission Timeout Estimator

- Key observation:
  - Using β RTT for timeout doesn't work
  - At high loads round trip variance is high
- Solution:
  - If D denotes mean variation
  - Timeout = RTT + 4D

Jacobson/ Karels Algorithm

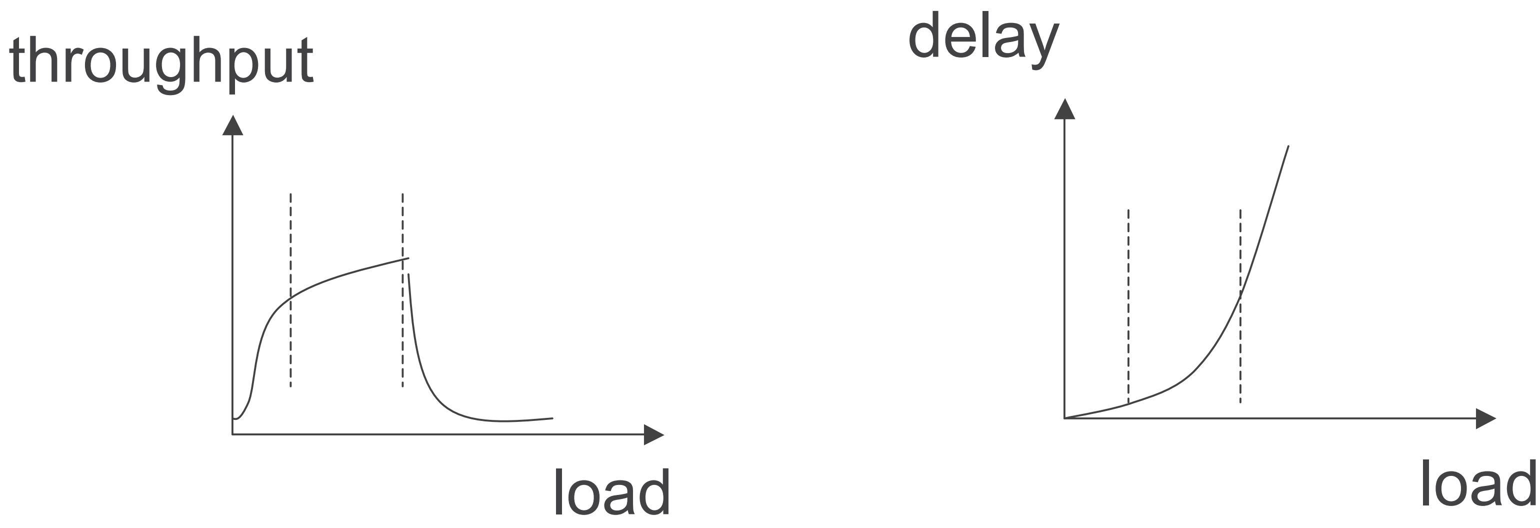
- New Calculations for average RTT
- Diff = SampleRTT - EstRTT
- EstRTT = EstRTT + (d x Diff)
- Dev = Dev + d(|Diff| - Dev)
  - where d is a factor between 0 and 1
- Consider variance when setting timeout value
- TimeOut = m x EstRTT + f x Dev
  - where m = 1 and f = 4
- Notes
  - algorithm only as good as granularity of clock (500ms on Unix)
  - accurate timeout mechanism important to congestion control (later)

Congestion



- If both sources send full windows, we may get congestion collapse
- Other forms of congestion collapse:
  - Retransmissions of large packets after loss of a single fragment
  - Non-feedback controlled sources

Congestion Response



Avoidance keeps the system performing at the *knee*

Control kicks in once the system has reached a congested state

Separation of Functionality

- Sending host must adjust amount of data it puts in the network based on detected congestion
- Routers can help by:
  - Sending accurate congestion signals
  - Isolating well-behaved from ill-behaved sources

6.3 TCP Congestion Control

- Idea
  - assumes best-effort network (FIFO or FQ routers)each source determines network capacity for itself
  - uses implicit feedback
  - ACKs pace transmission (*self-clocking*)
- Challenge
  - determining the available capacity in the first place
  - adjusting to changes in the available capacity

TCP Congestion Control

- A collection of interrelated mechanisms:
  - Slow start
  - Congestion avoidance
  - Accurate retransmission timeout estimation
  - Fast retransmit
  - Fast recovery

Congestion Control

- Underlying design principle: packet conservation
  - At equilibrium, inject packet into network only when one is removed
  - Basis for stability of physical systems
- A mechanism which:
  - Uses network resources efficiently
  - Preserves fair network resource allocation
  - Prevents or avoids collapse
- Congestion collapse is not just a theory
  - Has been frequently observed in many networks

TCP Congestion Control Basics

- Keep a congestion window, cwnd
  - Denotes how much network is able to absorb
- Sender's maximum window:
  - Min (advertised window, cwnd)
- Sender's actual window:
  - Max window - unacknowledged segments

Congestion Under Infinite Buffering

- Nagle (RFC 970) showed that congestion will not go away even with infinite buffers
- Basic argument
  - A datagram network must have TTL
  - With infinite buffering queuing delays increase
  - Even if buffers are not dropped for lack of buffering, they will be dropped because TTL expires