Internet Engineering

241-461

Robert Elz

kre@munnari.OZ.AU
kre@coe.psu.ac.th

http://fivedots.coe.psu.ac.th/~kre

TCP Notes

- TCP connections are bi-directional
  - Every packet has sequence & acknowledgement numbers
    - Sequence number indicates which data are in this packet (if any)
    - Or where in sequence next data will go
    - Acknowledgement indicates which data are expected to be received next
  - Can acknowledge data received,
    - and send answer (any reply)
    - in the same packet
- Window size sent in every packet
  - Can be varied as buffer space at receiver varies

TCP Connections

- Application needs to communicate with peer
  - establish data path between applications
  - TCP connection
- Must identify remote application
    - Transport Protocol Address
  - System address
    - To come later
  - Application Identifier
    - Port Number
- Sender also needs address
    - For returning packets
  - Another Port Number

TCP Connections (2)

- Connection Identity
  - (1) Network Address (system 1)
  - (2) Network Address (system 2)
  - (3) Port Number (system 1)
  - (4) Port Number (system 2)
- Port Number
  - 16 bit identifier
    - 0 .. 65535
- Each different set of identifiers
  - identifies a different connection

TCP Connections (3)

- TCP Model
  - All connections created when TCP created
    - 1980 ???
    - For IPv4:
      - $2^{32} * 2^{32} * 2^{16} * 2^{16}$
  - Each system (with 1 network address)
    - $2^{32} * 2^{16} * 2^{16}$
  - Remain forever
    - mostly dormant

TCP Connections (4)

- Each connection has a state
  - Closed
  - Opening
  - Data Transfer
  - Closing
    - Simplified model
- TCP uses Finite State Machine
  - Defines how states are manipulated

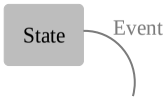
Finite State Machine

- More formal method of speificiation
  - Often depicted using drawing
- Some number of states defined
  - drawn as circles or rectangles
- In each state
  - specific events can occur
    - inputs
    - timeouts
  - cause transition to another state
  - cause output action to occur

FSM Basics

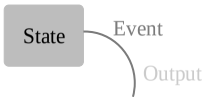
- The FSM is in some state

FSM Basics



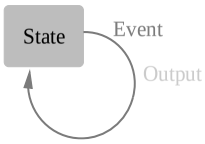
- An event occurs
  - drawn beside a line
  - shows transition to a new state

FSM Basics



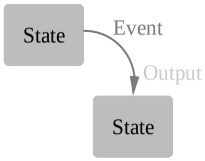
- Some output may accompany transition

FSM Basics



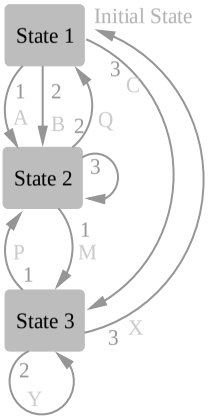
- Transition may return to the same state

FSM Basics



- Or may transition to another state

FSM (example)



FSM (example)

Three States 1 2 and 3

Three events that occur 1 2 and 3

Several output actions

State 1 is the initial state

For input events

1 1 1 2 3 2 3 1 2 2 3

What states does FSM pass through?

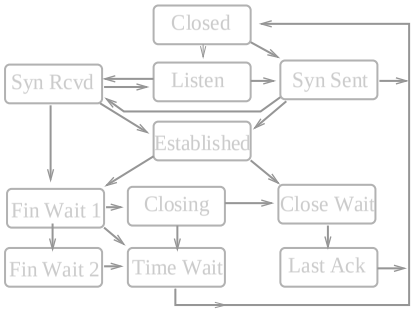
What output actions are performed?

1 2 3 2 1 3 3 1 2 1 2

2

A M P O C Y X A O B

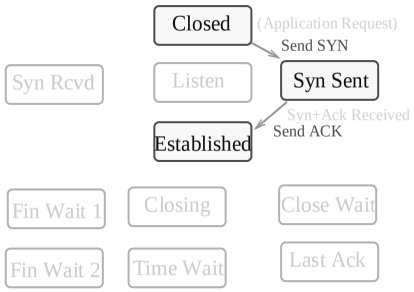
TCP Connections (States)



FSM of TCP connection machinery

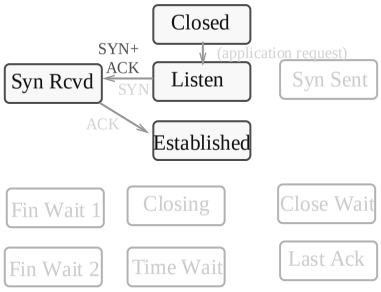
- Inputs & Outputs to come later

TCP Client Open



- 3-way Handshake
- SYN SYN+ACK ACK

TCP Server Open



- Same 3-way handshake
  - SYN SYN+ACK ACK

TCP Opens

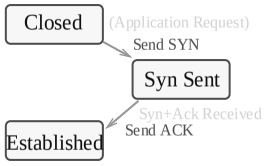
- Start in CLOSED state
  - Neither system knows anything about other
- Send SYNchronise
  - Inform other end we want to communicate
  - Tell it what sequence number we use
- SYN is a kind of data
  - Uses one sequence number itself
  - Always the first sequence number of this connection
- Sender picks sequence number
  - Not just Start at zero
  - To meet some objectives
    - No old packets from previous connection
    - Not easy for anyone to guess

TCP Opens (2)

- SYN is data
  - So must be acknowledged
- Receiver of SYN server
  - Must send ACKnowledge
    - So sender knows it was received
- TCP is full duplex
    - data goes both directions
  - Need sequence numbers both directions
  - So must SYNchronise in reverse
    - Send SYN back to client
- Data and ACK can travel in same packet
  - Data here is SYN
    - Therefore SYN+ACK

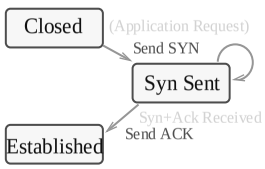
TCP Opens (3)

- Data in SYN+ACK packet must be acknowledged
  - That is the SYN
- Must ACKnowledge that SYN
  - But just ACK this time
  - SYN already sent
    - Other than retransmit
    - Just one SYN per connection
      - In each direction



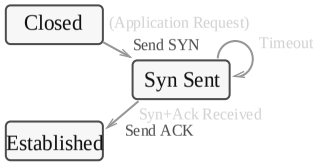
TCP Opens (3)

- Data in SYN+ACK packet must be acknowledged
  - That is the SYN
- Must ACKnowledge that SYN
  - But just ACK this time
  - SYN already sent
    - Other than retransmit
    - Just one SYN per connection
      - In each direction



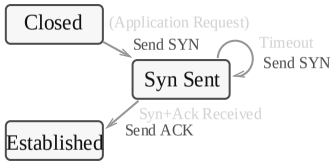
TCP Opens (3)

- Data in SYN+ACK packet must be acknowledged
  - That is the SYN
- Must ACKnowledge that SYN
  - But just ACK this time
  - SYN already sent
    - Other than retransmit
    - Just one SYN per connection
      - In each direction



TCP Opens (3)

- Data in SYN+ACK packet must be acknowledged
  - That is the SYN
- Must ACKnowledge that SYN
  - But just ACK this time
  - SYN already sent
    - Other than retransmit
    - Just one SYN per connection
      - In each direction



- Investigate
  - 2 systems talking to each other
    - Both acting as clients