# TNE30019/TNE80014 – Unix for Telecommunications

## Network Programming Basics

Dr. Jason But

Swinburne University

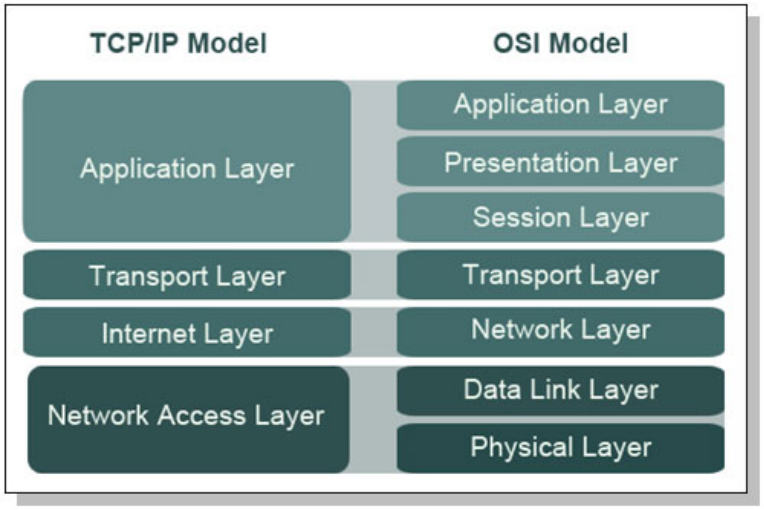


## Outline

- Layers in the TCP/IP protocol stack
  - Each layers relevance to programmers
  - What can be achieved at each layer
- What network services are available to programmer



## TCP/IP Stack – Layers




http://electronics.stackexchange.com/questions/31171/internet-vs-serial-communication



## Physical Layer

- Physical communication between hosts sharing medium
  - Standardised by IEEE etc.
  - How is information encoded into signal on medium
  - Amplitudes, frequencies, wave lengths etc.
- No general access for applications or main kernel
- Implemented mostly in hardware
- Device driver may have some control and provide configuration options

## Link Layer

- Packet communication between hosts in same IP subnet ("directly" connected)
  - Shared medium access
  - Synchronise sender and receiver
  - Addressing (MAC addresses)
  - Reliability (checksums, retransmissions)
- Tightly coupled to physical layer
- Implemented in hardware/driver
- No application access, e.g. MAC addresses managed in kernel
- Possible access by modifying kernel or configuration options provided



## Network Layer – Internet Protocol (IP)

- Packet communication between Internet hosts
  - End-to-end addressing and routing (IP addresses)
  - Packetisation (packet length, packet fragmentation)
  - Quality of service (Type of Service byte)
  - Reliability (time-to-live, checksum)
  - Unicast, multicast, broadcast
- Implemented within kernel
- `root` may construct IP packets (**raw** sockets)
- All other users have limited access to IP layer (socket options allow some configuration)



## Transport Layer – Transmission Control Protocol (TCP)

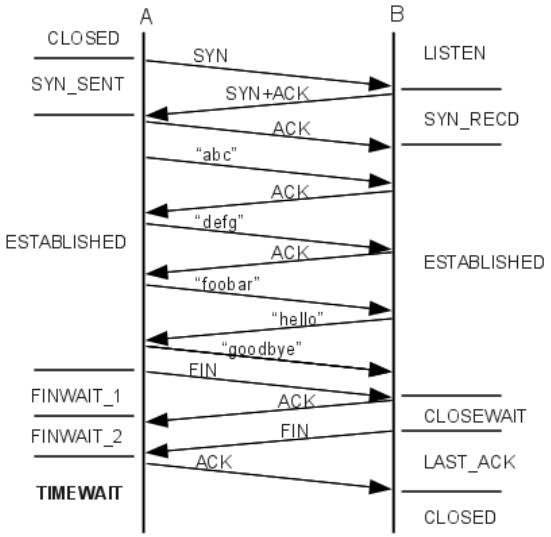
**Properties**

- Packets delivered in order to application
- Guaranteed error-free delivery of data
- Application addressing (ports)
- Manages throughput based on available network bandwidth and receiver buffer space (flow control)

- Implemented within kernel
- Kernel manages connection establishment and teardown, throughput, receiver buffering, sender retransmissions
- `root` may construct TCP packets (**raw** sockets)
- Other users have limited access (socket options)
- Application requests TCP connection to be created:
  - Sending data to TCP socket is similar to writing to file
  - Reading data from TCP socket is similar to reading from file



## TCP Protocol Message Exchange




http://intronetworks.cs.luc.edu/current/html/tcp.html

## Transport Layer – User Datagram Protocol (UDP)

### Properties

- Datagram, best-effort based service
- Application addressing (ports)
- Error detection

- Implemented within kernel
- `root` may construct UDP packets (**raw** sockets)
- Other users have limited access (socket options)
- Application sends "messages" which arrive (or not) at destination
- OS doesn't buffer packets – sent as soon as possible



## TCP vs UDP

### When to use TCP

- Need to transmit stream of data reliably to destination
- Don't need timely sending
- Need congestion avoidance (variable transmission rate)

### When to use UDP

- Need to send data timely
- Don't want overheads of guaranteed transmission
- Don't want to back-off the transmission rate

### Typical applications for TCP

- Non time-critical applications, reliable client-server applications
- WWW, database protocols, file transfer, email transfer

### Typical applications for UDP

- Single packet request-response
- Real-time services
- Online games, Voice-over-IP (VoIP), streaming video/audio, NTP



## New Transport Protocols

- Multi-Path TCP (MP-TCP)
  - Multi-homing support – use multiple concurrent sub-flows
  - Shared path detection – *sort of*
  - Doesn't need specialised middle-box support
- Stream Control Transmission Protocol (SCTP)
  - For signalling protocols (e.g. SS7)
  - Independent streams over same connection
  - Multi-homing support
  - Reliable in-order or out-of-order message delivery
  - Improved error detection
- Datagram Congestion Control Protocol (DCCP)
  - Unreliable transport like UDP
  - But flow/congestion control like TCP
- Slow adoption



## Upper Layers

### Application Layer

- Protocols that applications use to communicate on top of TCP/IP etc.
  - For example, HTTP, SMTP, SSH, FTP, ...
- Implemented within application
- Communicate with Transport Layer via standard APIs

### APIs

- API must exist to allow applications (user space) to interact with Transport/IP Layer (kernel space)
- Name resolution
  - Resolves names into IP addresses (and vice-versa)
  - DNS resolver (DNS client) implemented in kernel
- Data communications: **Sockets API**