Networks and Operating Systems

Sophie Boyle - Glasgow University Computer Science

Part A

1 Protocols and Layers

1.1 Network Protocols

[**Network Protocol**] Agreed language for encoding messages. Defines message meaning and how often they can be sent.

[**Protocol Data Unit**] Textual/binary messages which construct a protocol. Have syntax/grammar defining legal messages.

Protocol semantics define: who by and when PDUs can be sent, the response required, machine roles (client-server, p2p), error handling.

1.2 Protocol Layering

[**Layering**] Separation of communication systems.
Example: Web browser-server five layers: HTML, HTTP, TCP, IP Ethernet

1.3 OSI Model

[**Systems**] Application, Presentation, Session, Transport, Network, Data Link, Physical.
[**Router**] Network, Data Link.

[**Physical**] Cables/optical fibre: size/shape, length, type.
For Wireless: Radio freq, transmission power, antenna, etc.

[**Data Link**] *Structure/frame physical layer bit stream*, by splitting it into messages and detecting errors. *Performs media access control*, by assigning addresses to hosts, decide when they are allowed to send messages, and ensuring fair access to avoid overwhelming the host.

[**Network Layer**] Connects links, forming wide area network. Controls data delivery, routing, naming/addressing, and admission/flow-control.
E.g. IP, ICMP

[**Transport Layer**] Transer of data from source to destination; i.e. between session level service at source, and the corresponding service at destination.
E.g. TCP, UDP

[**Session Layer**] Manages multiple transport layer connections.
E.g. Opening several TCP/IP connections whilst downloading a web page over HTTP

[**Presentation Layer**] Manages representation/conversion of data.
E.g. Character set, markup languages, format conversion

[**Application Layer**] User application protocols.
E.g REST APIs, WebRTC

1.4 Protocol Standards

Formally describes a network protocol, and ensures interoperability.
E.g. Open/closed development, intellectual property rights, individuals/corporate/national membership.

2 Physical Layer

The task of the physical layer is to transmit bits over an analogue channel.

Transmission of **bytes**.
Decides on which cables/wireless link, encoding, and channel capacity.

Involves physical characteristics of **cables/optical fibre**.
Size/shape/length, cable type (voltage, current, modulation), fibre type (single/multi mode, colour, laser modulation)

- *Unshielded Twisted Pair: Unidirectional pair, twists reduce interference, noise increases with cable length.*
- *Optical Fibre: Glass core in plastic jacket, unidirectional (laser emitter to photodetector), low noise due to usage of light, expensive lasers required to operate.*

2.1 Baseband Data Encoding

Encode signal as change in voltage (cable) / intensity of light (optical fibre).

[**NRZ**] 1 = High signal, 0 = Low signal.
Problems arise when there are runs of the same value, causing 'clock skew' and 'baseline wander'

[**NRZ-Inverter**] 1 = Change in signal, 0 = Constant signal.
Solves constant runs of 1s, but not 0s.

[**Manchester**] 1 = High-low transition, 0 = Low-high transition.
Doubles required bandwidth, but solves both problems.

2.2 Carrier Modulation

Shifting signal from baseband to higher carrier frequency, allowing for multiple signals on a single channel.

2.3 Bandwidth, Channel Capacity, Noise

[**Bandwidth**] Determines transportable frequency-range. *Limitations are physical.*

[**SNR**] (Signal to Noise Ratio) used to measure how good the signal is. *Max transmission rate of channel grows logarithmically to SNR*

3 Data Link Layer

Arbritrates access to physical layer by addressing devices, structuring/framing raw bitstream, detecting/correcting errors, and media access control.
Interface with Physical layer: raw bit stream. Interface with Network layer: addressing, packets.

3.1 Addressing

Physical links may be *point-to-point* or *multi-access*.
Host addresses may be *link-local* or *global scope*.

3.2 Framing and Synchronisation

Bits provided by the physical layer are unreliable; they may be **corrupt**, or the timing may be disrupted.
Data Link layer may correct these problems by:

- Breaking raw bit stream into **'frames'**.
- **Transmitting/repairing** individual frames.
- **Limiting scope** of transmission errors.

3.3 Error Detection/Correction

Error detecting codes may be added to each packet.

[**Parity Codes**] Calculate data parity.
Count 1 bits. Odd amount = parity 1, even amount = parity 0. (XOR of data bits.)
Data packets are transmitted with the parity, to be checked by the receiver.
Detects single bit errors.

[**Internet Checksum**] Sum data values, send checksum per frame.
Receiver calculates checksum, a mismatch indicates an error.
Detects multiple bit errors.

Error detecting codes can be used to correct errors.
Error correcting code transmitted with each frame.
E.g. Hamming code

Other means of repair involve **requesting retransmission** of data.

3.4 Synchronisation

Methods of detecting a message's beginning include:

- Leave **gaps** between frames *Physical Layer is unreliable for this.*
- Each frame preceded with **length field** *Corrupt length causes issues.*
- Each frame preceded with **special start code**.

3.5 Media Access Control

Links may be point-to-point/multi-access.

[**Point-to-point**] 2 unidirectional links
Individual cables for each.
Framing in each direction.

[**Multi-Access**] Bidirectional link
Single physical cable - nodes fight for access to link.
Single radio frequency.

[**Link Contention**] Simultaneous transmission from two hosts causes collision.
Signals overlap, resulting in fuck all received.

3.6 Contention-Based MAC

A system is **contention-based** when multiple hosts share a channel, in a way which leaves possibility for collision.

[**Two-Stage Access to Channel**] Detects a collision is occuring/will occur by listening to the channel while/before sending.
Back-off delay is randomised, preventing repeated collisions, however this can be arranged in terms of priority to specific traffic.

[**ALOHA Network**] Transmits whenever data is available.
Waits a random period of time before retransmitting.
Repeats until success.
Simple, poor performance: low channel utilisation and long delays.

[**Carrier Sense Multiple Access**] When propagation delay is low, listen before sending.
Upon another active transmission, back-off.
When link is idle, send data.
Improves channel utilisation.

CSMA may be updated with **collision detection**

4 Network Layer

Responsible for delivery of data (end-to-end) and building an internet.

[**The Internet**] Globally interconnected networks running *IP*.
IP provides an abstraction layer involving connectionless packet delivery, addressing, routing, fragmentation, and reassembly.

[**IP Service Model**] Network attempts to deliver packet.
Packet may be discarded if it cannot be delivered, or if it is lost, reordered, duplicated or corrupt.
Can be run over any type of link layer. A universal standard.

4.1 Addressing

Each network interface is intended to have a unique address.
This may change for privacy. *Dynamic IP*
Firewalls make it so that an addressable IP is not always reachable.

[**IPv4**] 32 bit address.
[**IPv6**] 128 bit address.

4.2 Packet Properties

[**Fragmentation**] Solution to the link layer having a maximum packet size *(MTU)*.
IPv4 will fragment packets larger than the MTU.
IPv6 does not support fragmentation.

[**Loop Protection**] Packets have a forwarding limit.
Set a non-zero value when a packet is sent, each router forwarding the packet reduces the value by 1. Upon reaching zero, the packet is discarded.

[**Header Checksum**] Similar concept to Link Layer Checksum (uses different algorithm).
Only protects IP header.
Supported by *IPv4* only.

[**Transport Layer Protocol Identifier**] Network packets include transport layer meta data.
Identifies the transport layer protocol to be used, ensuring data is passed to the correct protocol in the upper layer.

4.3 Identity and Location

Constant addresses denoting host identity, independent of where/when they attach to network.
Simple upper layer protocols, but adds complexity to network layer.

Location-based addresses indicating where a host attaches to the network.
Pushes complexity to upper layers.

4.4 IP Addresses

Specifies location of network interface.
Allocated hierarchically (in order).
Fixed length binary values. *(IPv4 32-bit, IPv6 128-bit)*

[**Netmask**] Indicates the number of bits in the network section.
[**Network Address**] Host section (equal to 0).
[**Broadcast Address**] All bits of host section equal to 1.

[**IP Classes**] Allocated to give the netmask a multiple of 8 bits.
Class A → /8 Network *16 million addresses*
Class B → /16 network *65536 addresses*
Class C → /24 network *256 addresses*

4.5 IP Address Management

IANA allocates addresses to RIRS, which allocate to ISPs, which allocate to customers.// The last available allocations were made in 2011.// IPv6 would solve the address shortage.

Problems with deploying IPv6 is that host changes would be required, and routers/firewalls would all need to be updated.

5 Routing

Network layer is responsible for routing data from source to destination, using multiple *hops*.
Nodes learn some of the network topology, and run a **routing algorithm**, deciding where to forward packets.
Routing algorithms include: intra-domain, and inter-domain

5.1 Inter-Domain Routing

Finds the best "Autonomous-System level" path from the source AS to destination AS.
AS treated as nodes, connections between treated as edges.

Edge Networks use default route to core.
Core Networks use full routing table.

5.2 Routing in the Default Free Zone

Core networks are well-connected and know about every other network.
Route is by **policy**, *not by shortest path*
Complete AS-level topology required.

5.3 Intra-Domain Routing

Operates within network.
Any network is considered an AS.
Single routing protocol.
Running on IP routers within an AS.

5.4 Distance Vector Protocols

Each node has a vector, containing the distance to every other node in the network.
Packets are forwarded according to the least distance to destination.

A routing table is updated in rounds.
Nodes are initialised with vector distances to their neighbours.
Spreads by one hop: nodes will get distance vectors to neighbours of neighbours.
etc

5.4.1 Limitations

[**Count to infinity**] Hops can be seen to loop.
Therefore we define infinity as 16 hops.

[**Split Horizon**] Upon routing update, a route learned from a neighbour cannot be sent back to the neighbour.
This prevents looping involving two nodes.

5.4.2 Pros

Simple implementation.
Routers only need to store distance to each other node. $O(n)$ *complexity*
Slow convergence.

5.5 Link State Routing

[**Link State Information**] Nodes know links to neighbours, and costs of link usage.
Flood this info, giving all nodes complete network map.
Each node directly calculates shortest path to every other node **locally**.
The local calculation is used as the *routing table*.

Packets are forwarded based on shortest path, which is recalculated upon each routing update.

5.5.1 Pros

More complex (than Distance Vector Protocol)
Each router stores complete network path. $O(n^2)$ *complexity*
Faster convergence.

6 Transport Layer

Isolates upper layers from network layer.
Hides network complexity.
Provides usable service model and programming API.

[**Addressing and Multiplexing**] Transport layer identifies a **user process** running on a host
e.g. A service/program

[**Reliability**] Enhances network service to match application requirements.
Different applications may have different levels of reliability.
e.g. a voice/streaming video can tolerate data loss for quick delivery, whereas an email must suffer from no data loss.

[**Framing**] Applications desire to send structured data.
Transport layer maintains boundaries by framing the data.

[**Congestion and Flow Control**] Controls application sending rate to match..
How fast the *Network Layer delivers* data (congestion).
How fast the *receiver* can *process data* (flow).

[**End-to-End Principle**] Trust the Network Layer with nothing besides necessary functions.
Leave reliability checking to the transport layer, as the application will have to check data anyway.

6.1 Internet Transport Protocols

Internet Protocol provides common base for:

- **UDP** User Datagram Protocol
- **TCP** Transmission Control Protocol
- **DCCP** Datagram Congestion Control Protocol
- **SCTP** Stream Control Transmission Protocol

[**(UDP) User Datagram Protocol**] Raw IP exposed to applications.
Connectionless: Framed but unreliable packet delivery.
No congestion control.

[**(TCP) Transmission Control Protocol**] Reliable byte stream protocol over IP.
Has reliability and congestion control.
No framing: ordered byte-stream only.

Both use: 16 bit port number as service identifier.

6.2 TCP: Transmission Control Protocol

6.2.1 Connection Setup

3-way Handshake

- SYN/ACK flags in TCP header.
- SYN bit set in initial packet (randomly chosen initial sequence number).
- Reply also has SYN bit set, and ACKnowledges the initial packet.
- Handshake is completed upon acknowledgement.

Randomly chosen initial numbers: ensures robustness to delayed packets and restarted hosts.
Acknowledgement: ensures reliability.

6.2.2 Reliability

[**SYN**] **Sequence Number** Counts the bytes that are sent.
[**ACK**] **Acknowledgement Number** Specifies the next byte expected to be received.
Retransmission of lost packets.

Lost Detection performed as follows...

[**Triple duplicate ACK**] 4 identical ACKs in a row.
Some packets lost, however later packets still arrive.

[**Timeout**] Data is sent, but acknowledgements stop.

Packet Reordering
Duplicate ACKs may be caused by packet delay, resulting in reordering.
Gives illusion of loss, when data was actually delayed.
Triple duplicate ACK may therefore be used as indication of packet loss, resulting in request of re-transmission.

6.2.3 Congestion Control

Transmission speed is adapted to match network capacity.
Prevents the congestion collapse of a network.

Can either be implemented at Network or Transport layer.
Network Layer: Safe, all transport protocols are congestion controlled, however requires a consistent congestion control scheme across all applications.

Transport Layer: Flexible, protocols may be opitimised for different applications, however a misbehaving transport may congest the network.

[**Conservation of Packets**] Transmission rate matches the network capacity.
At which, a packet is sent for each acknowledgement recieved.
Total number of packets in transit is always constant.
Transmission rate automatically adjusted for congestion.

[**(AIMD) Additive Increase Multiplicative Decrease Algorithm**] defines adjustment of transmission rate.
Slow start, gradually increasing to equilibrium by adding a small amount to transmission speed at each interval without loss.
Respond to congestion rapidly by multplying the sending factor by a float $x < 1$ during each interval of loss.

6.2.4 TCP Sockets

Sockets initially unbound and can accept or make connections.
Commonly found in client-server structure.
Host makes socket bind(), listen(), and accept() connections on an established port.
Other host makes socket connect().
Either can send() and recv().

6.2.5 Communication

[**accept()/connect()**] 3-way handshake during.
[**send()**] transmits data: blocks until data written and returns number of bytes sent.
[**recv(x)**] reads up-to X bytes of data, and blocks until data is available/connection closed. Also returns byte object with data received.
[**exceptions**] used for error handling.

Send() queues data, which is split into segments, for transmission.
Each segment packaged in a TCP packet.
Packet sent when allowed to by Congestion Control Algorithm, *such as AIMD.*
If data in send() is too large, it will be split into several segments.
If data in send() is "too small", many requests may be aggregated into a single TCP segment.
Data returned by recv() may correspond to a single or multiple send() call(s).
Applications must be written to dealt with data returned in varying size.

6.3 UDP: User Datagram Protocol

UDP identify applications via 16-bit port number.
Used peer-to-peer: both IPs must bind to known port.

$socket(type = SOCK_DGRAM)$
No connections in UDP: i.e. no connect()/accept().

[**sendto()**] Sends a single datagram, and can be used to send to multiple addresses.
[**recvfrom()**] Reads a single datagram and additionally returns sender's address.
Each recvfrom() corresponds to a single sendto().

Transmission is unreliable: packet loss, delay, reordered, and duplicated, is not handled.
Application is responsible for handling the above with sequencing, reliability, and timing, alongside congestion control. (i.e. everything that is handled already by TCP.)

7 Network Address Translation

NAT hides a private network behind one public IP address.
Multiple hosts with private addresses under one public IP.

[TCP NATS]
Outgoing connection creates state in NAT.
Data must be sent periodically to avoid timeout. (Usually 2 hours)
Incoming connections have no state.
NAT cannot know where to forward incoming connections without port forwarding.

[UDP NATS]
NAT has shorter timeout.
P2P (Peer-to-peer) connections are easier.

Many applications fail with NAT.
Servers behind NAT need port-forwarding.
P2P applications fail due to the complex algorithm requiring them to connect.
There is also no particular security benefit.

Why use NAT?
ISPs don't have enough addresses to give each customer their own prefix, and the customers don't want to pay for more addresses.
Avoids re-numbering a network when changing ISP, as people can hardcode IPs (even when they shouldn't).

8 Higher Layer Protocols

8.1 Session Layer

Determines the connections required by the application.

IP addresses encode location, therefore moving around breaks transport layer connections.
Session layer must find new location and establish new connections.
May be redirected by old location (HTTP redirect).
Network complexities are being pushed to the upper layers.

A single session may span multiple transport connections, which must be co-ordinated by the session layer.

Protocols may rely on middleboxes/caches.
Web cache - optimises performance by moving popular content closer to hosts.
Email server - supports disconnected operation by holding emails until user connects.
Instant messaging servers - Locate users and respond for offline users.

8.2 Presentation Layer

Manages presentation/representation/conversion of data.

[**Media Types**] Media types identify the format of data and are included in protocol headers.
Formats are categorised into 8 top-level types, and have subtypes and parameters.
e.g. text/plain; charset=iso-8859-1 → Plain is the subtype, and charset=iso.. is the parameter.

[**Content Negotiation**] Protocols may negotiate the media formats used, ensuring sender and reciever have a common format.
Consists of an offer-answer exchange:

- Offer lists supported formats ordered by preference.
- Answer contains the most preferred format that the receiver understands.
- Negotiates common format in one round-trip time.

Channel Encoding Data is *encoded* to fit the used character-set, and encoding must be *signalled*.
Must be backwards compatible with text-only systems (ones that only support 7bit ASCII and enforce maximum line length.)
Must survive translation between character sets (Boomer legacy systems using ASCII)
Must use non-printing characters.
Must avoid escape characters.

8.3 Application Layer

Protocol functions specific to application logic. *e.g. Deliver email/Retrieve webpage.* Consider: message types needed, how interactions occur, and how errors are reported.

8.3.1 Domain Name Service (DNS)

Application layer protocol running over a network.
Most widely used UDP-based application.
Network operates entirely on IP addresses.
Translates user-friendly names to IP.
www.dcs.gla.ac.uk $\rightarrow$ *130.209.240.1*

Heiarchy of DNS zones.
One logical server per zone.
Hop-by-hop name look-up.
.com $\rightarrow$ *.google* $\rightarrow$ *maps or www*

8.3.2 HyperText Transfer Protocol (HTTP)

Application layer request-response (pull) protocol running on a reliable Transport layer protocol.
Stateless: each request "agnostic" of previous requests.
HTTP/1.0 single request/response per connection $\rightarrow$ HTTP/1.1 multiple request/response pairs per connection.
Typing/negotiation of data representation, meaning systems can be built independently of data being transferred.
Text-based requests, where binary data is encoded.

Endpoints defined by **Uniform Resource Identifier (URI)**
Consists of: Protocol (http(s)), Hostanme (IP address/DNS domain name), Port number, Path/-filename (of requested resource).
Special characters are not allowed.

Client parses URI, extracting protocol/hostname/port-number/path, and construct a request message.
Client connects to server using TCP/IP socket, and sends the request message.

Server parses client request, mapping path to local resource, and construct response.
Client reads response from socket until end or connection-closed.
Client parses HTML and extracts URIs.

Subsequent requests may start from beginning or reuse the socket if "keep-alive" is on.

9 Privacy and Security

It is possible to intercept traffic on a network.
Countries do it for legal reasons, organisations do it for business reasons, and malicious users monitor it on a link.
Encrypting data helps achieve confidentiality.

[**Symmetric Cryptography**] Converts plain-text → cipher-text.
Fast, and therefore good for bulk-encryption.
Conversation protected by secret key, used to encrypt and decrypt.

[**Public Key Cryptography**] Key split: Public/Private key.
Public key is widely distributed and used to encrypt.
Private key is kept secret and used to decrypt.
Slow encryption/decryption.

[**Hybrid Cryptography**] Combination of public-key and symmetric cryptography.
Generate random session key which is used with symmetric cryptography.
Public key used to distribute session key.
Encrypt data using symmetric cryptography, keyed by session key.

[**Authentication**] Cryptographic hash and public key cryptography can be used to create a **digital signature**, which identifies whether a message has been tampered with or not.
Clarifies that there is no man in the middle, and can be used to trace data origin.

[**Cryptographic Hash Functions**] Generate a fixed length hash code of arbitrary length input value.
Cannot derive input value from hash.
Cannot duplicate hash.

[**Digital Signature Algorithms**]
Production: Generate a cryptographic hash, and encrypt the hash with your private key.
Verification: Re-calculate cryptographic hash, and decrypt signature using public key. Results of both actions should match.

[**Validating Input Data**] Networked applications must be prepared to deal with data supplied by untrusted third-parties.
This data should be validated.

e.g. **Buffer Overflow Attack**
Memory-safe programming languages check array bounds, whereas unsafe languages don't.
This is easy to take advantage of.