# ELEC5471M

# Data Communications & Network Security

# 25/26

*Taught by Prof. Andrew Kemp*

*Notes by Jacqueline Walker*

---

## Contents

## Reading List

- Computer Networking, J. Kurose & K. Ross
- Protocols and Architectures for Wireless Sensor Networks, K. Holger & A. Willig
- Computer Networks, A. Tanenbaum

## Useful Documents

- Unit 1 PPT

# Unit 1

## Lecture 1: Introduction

Networking is the technology and architecture that allows for devices to connect with each other and communicate. The way in which a communication is facilitated is known as a channel: for example, twisted pair (Ethernet), radio links (Wi-Fi). End devices are the start and end of communication i.e. a computer, phone, modem or other routing device.

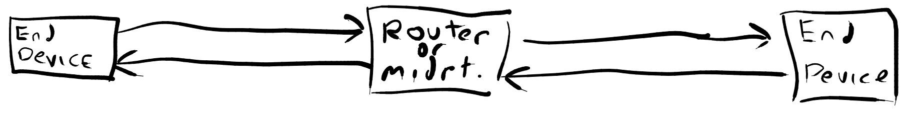

Figure 1: An example of a basic network connection, where each endpoint is connected with a router.

Inside each device there are protocol stacks, protocols, and specialised networking hardware to facilitate and structure networking communication (keeping data integrity, keeping messages secure).

### 1.1: Reasons for Networking

The motivation behind networking originally began in academic environments, when computers were usually very big and very expensive. A network would allow for the joint use of computational resources and the sharing of information between different academic institutes basically instantaneously. This would be done by creating a main connection with high bandwidth that each computer could connect to via a lower bandwidth connection.

Networking is also useful for corporations to allow for employees to work together even if they are in separate locations, and to allow remote viewing of the status of infrastructure (i.e. databases or other critical hardware).

For individuals, networking allows for easier streaming of entertainment (such as YouTube, IPTV, multiplayer games) and the ability for one-to-one communication with those around the world.

Of course, networking does have downsides. Having every computer connected allows for security issues (such as CHRISTMA EXEC, which spread through e-mail). There is also greater complexity & cost in connecting a computer to a network. In most cases though, the positives of networking far outweigh the negatives.

## 1.2: Scales of Networks

**PAN** Personal Area Network: Around the body, i.e. Bluetooth.

**LAN** Local Area Network: Around a home/office, i.e. Ethernet, Wi-Fi.

**MAN** Metropolitan Area Network: Around a city, i.e. core network for telephone systems.

**WAN** Wide Area Network: Around a very wide area/worldwide, i.e. the Internet

# Lecture 2: The Protocol Stack

## 2.1: Protocol Definition

A protocol defines the format and order of messages sent across the network, along with actions that are taken upon sending or receiving a message. Examples include TCP, HTML, DNS, etc. They govern all communication activity on the Internet

## 2.2: Protocol Layers

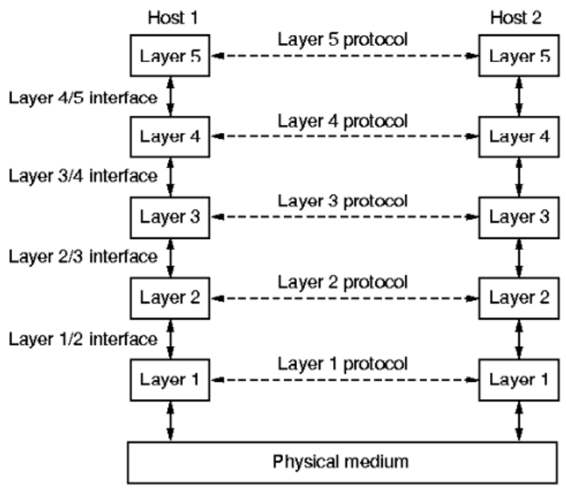

Figure 2: An example of a 5-layer protocol stack, where each layer only communicates directly with the layer below and above.

Network communication gets very complex. In order to send one human-readable message, you must first translate into bits that the computer can transmit, then a physical signal representation of those bits, then how that signal is transmitted through a channel/what channel is used along with error checking & recovery to make sure the message is sent successfully. Then, the signal must then go back up through each step to become human-readable at the other end. Along with this, you must allow for a variety of network layouts, such as wired/wireless, router make & feature set, whether there is a router at all, length of communication, etc.

Ideally, we'd like to break down this issue into multiple self-contained steps. For network communication, we call these 'layers'. Each layer in the stack is responsible for only part of the communication, i.e. the bottom layer is responsible for the transmission of bits over a medium, one layer is responsible for flow control, another for routing, etc. an explicit structure allows for easier identification & relationship of complex system pieces. It also allows for a modular structure, which allows for easier maintenance & updating of the system (IPv4 to IPv6). However, inefficiencies at each layer can add up, since the role of each layer may overlap.

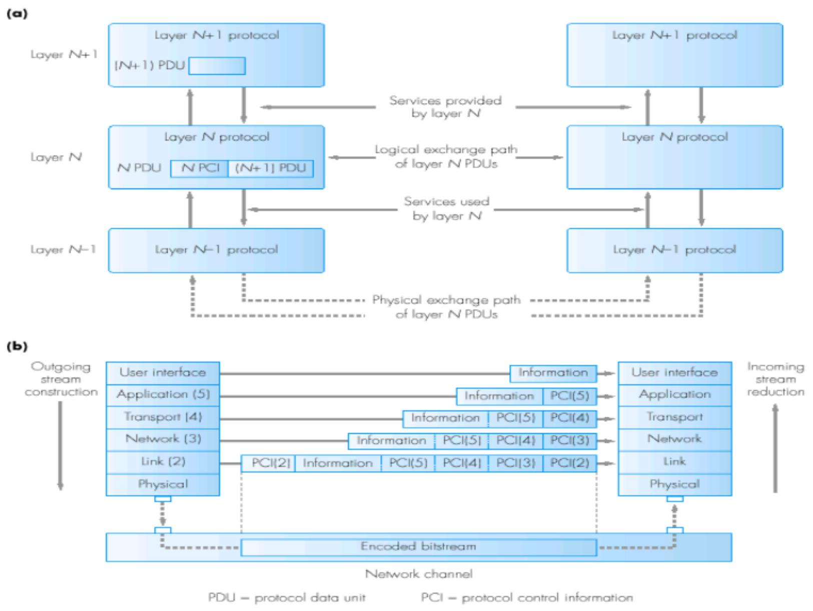

Figure 3: A more thorough example of layered communications.

Each layer implements a set of 'services' to the layer above. A service may be a request to do something, a requirement to be informed about something happening, a response to an event, or a confirmation of response.

In theory, each layer doesn't need to care about the implementation detail of lower layers. The message from layer N (Protocol Data Unit) is encapsulated with a header (Protocol Control Information) when passed to layer N-1. A PDU may be split into multiple PDUs for error checking purposes or size transmission limits. Each split will be given a header & tail.

## 2.3: Connection-Oriented & Connectionless Communication

Communication between two hosts may either be connection-oriented or connectionless. A connection-oriented communication requires that a connection must be established before communication, that the channel is held open for the duration of the communication, and that the connection is formally released. This is akin to a telephone call, where no-one else may use the channel while the phone call is active. There is the same delay for all packets transmitted & each packet is received in sequence.

A connection-less communication requires no establishment or release of intent to communicate. The channel is instead held for the duration of time of each message sent, where each message holds its own addressing details. A packet may arrive out-of-order, or that multiple packets in a row have varying delays.

# Lecture 3: The ISO/OSI Seven Layer Protocol Model

There are many issues that need to be considered when designing a layered communication model. There must be a way to identify the source and destination of the message, rules for the data transfer (simplex/duplex), identification of logical channel amounts, error control, flow control, sequencing, synchronisation, routing, and message splitting.

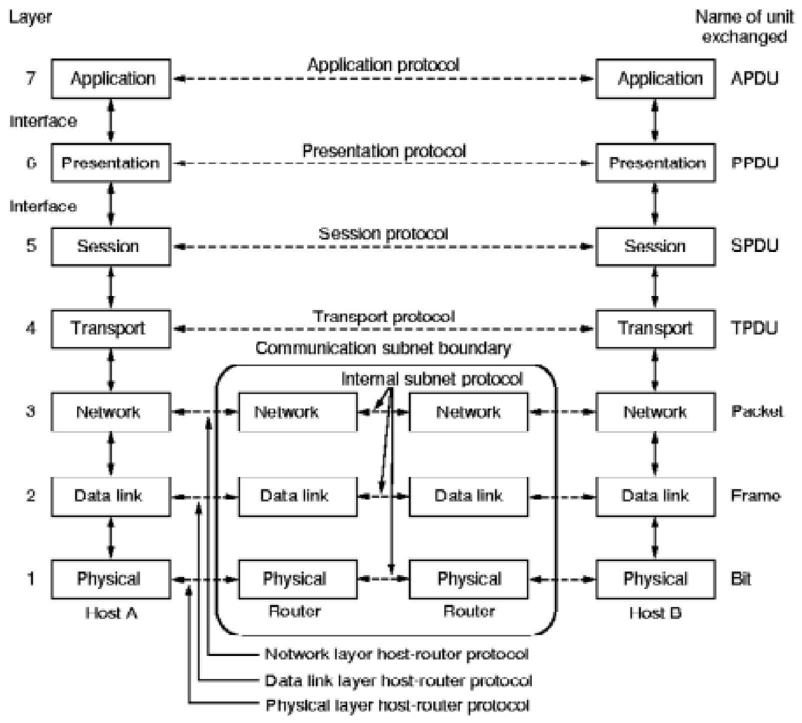

Figure 4: The ISO/OSI model diagram.

The ISO/OSI seven layer protocol model is a framework for standardising layered network communication into seven universally-recognised layers. Each layer is responsible for one major issue of communication.

The Physical Layer is responsible for the transmission of bits over a channel, whether that is a wired or wireless channel. It concerns itself with the representation of bits as a signal, polarity coding, whether the channel is simplex or duplex, the set-up & tear-down of the node-to-node channel if necessary, the pin-out of connectors, and channel nature. It is usually limited by the Nyquist limit (highest freq. component that can be reconstructed w/o distortion.) and the Shannon limit (the maximum amount of data that can be transmitted for a given BW and SNR).

The Data Link Layer is responsible for the reliable communication of frames across a physical link. A frame is usually around 100-1000 bytes. It ensures reliable communication via the implementation of acknowledgement (ACK) frames & error detection. If an error occurs, the frame wil be retransmitted in-order. The data stream may be split into multiple frames, with frame boundaries denoting the split. Each frame may also be 'stuffed' with bits/characters to pad out the size of a given frame. The DLL is also responsible for Medium Access Control (MAC), which allows for multiplexing of transmission.

The Network Layer is the only layer concerned with the routing of packets around the network. Routing is dynamically changed for each packet based on current network conditions. It is also concerned with internetworking between heterogenous networks (i.e. ones which use different addressing, protocols, packet sizes, etc.). The network layer also ensures that network owners get paid (accounting).

The Transport Layer is responsible for setting the Type of Service (TOS) offered to the Session Layer. The data from the session layer is split into separate datagrams/segments, and it allows for the multiplexing of data from multiple applications into one stream. In this case, the header for the transport layer will indicate which application should receive each message. The TL is also responsible for the set-up & tear-down of channels across the network, and the control of congestion so router buffers do not overflow.

The Session Layer is responsible for the maintenance & establishment of logical sessions (i.e. streaming a video from a web-server). It manages the synchronisation across the connection, recovery if the connection is halted for a period of time, and the control of who may speak at any given time (dialogue control). The data within the SL is called a PDU or SPDU.

The Presentation Layer provides general solutions to repeated problems. Its main focus is providing agreed-upon standards for PDU encoding & format for exchanging PDUs. It also is responsible for encryption.

The Application Layer is concerned with sending messages between applications. It provides user access to the OSI and defines a variety of common protocols for file transfer, mail, etc.

# Lecture 4: The TCP/IP Architecture

The TCP/IP architecture stems from the Advanced Research Packet Area Network (ARPANET), which was a military-funded research into packet-based networks. The ARPANET aimed to have seamless connection of multiple networks, a flexible architecture to allow a range of applications, along with the ability to survive the loss of subnet hardware.

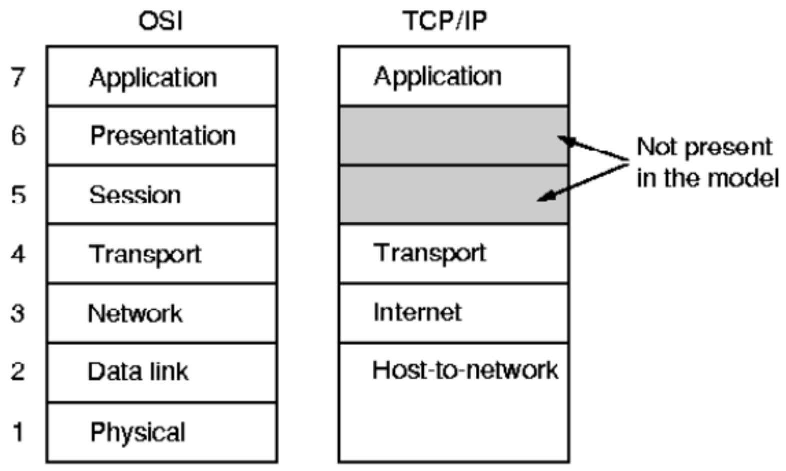

Figure 5: The TCP/IP model as it relates to the ISO/OSI model.

The TCP/IP model is a model based of the TCP/IP architecture as it is in use. It differs from the ISO/OSI model in a variety of ways, namely the lesser amount of layers. The functions present in the Presentation & Session layer are instead implemented in different protocols. The ISO/OSI model was designed to split into 3 areas: services, interfaces & protocols; the TCP/IP architecture was not designed with this in mind originally so, for example, protocols at each layer cannot be easily changed. The OSI model is very general, for use in describing a variety of different layered networking standards. In contrast, the TCP/IP model is only for use in describing the TCP/IP architecture and no other.

Each layer is distributed across the network, where each entity across the network implements the layers functions necessary for that entity (i.e. a router doesnt care about any layer functions above the Internet layer). Each entity performs actions & exchanges messages with its peers.

Each layer believes it is communicating directly with its peer (logical communication). For example, the Transport layer on the host entity believes it communicates directly with the Transport layer on the destination entity. In reality, the communication between goes down the layers on the host entity, across a link to a router, up the layers on the router for processing, down again across another link, before being received by the destination entity, where it is finally sent up the layers.

Similar to the ISO/OSI model, each layer takes data from above and prepends a header, before passing to the layer below.

## 4.1: Host-to-Network Layer

This is akin to the Physical & Data Link Layers from the ISO/OSI model. In the architecture, the only thing specified is that it requires that the host must connect to the network with a protocol that allows for IP packets to be sent. Actual implementation detail is left to specific protocols, i.e. Ethernet, point-to-point, etc.

## 4.2: The Internet Layer

This is akin to the Network Layer from the ISO/OSI model. It allows for packet-switched communication between nodes on a network. The protocol used is the IP protocol (IPv4 or IPv6). It is connectionless and allows for internetworking (insertion of packets onto heterogenous networks). Packets may arrive via any route and may arrive out-of-order.

## 4.3: The Transport Layer

This is akin to the Transport Layer from the ISO/OSI model. It provides source-to-dest peer entities to communicate. Two protocols are used, namely the Transmission Control Protocol (TCP), and the User Datagram Protocol (UDP). TCP is a connection-oriented reliable protocol that ensures in-order delivery of datagrams. It includes error checking & flow control. Messages may be fragmented for delivery. UDP is a connectionless unreliable protocol. It is very similar to IP, just in a higher layer. There is no guarantee of flow or message delivery.

## 4.4: The Application Layer

This is akin to the Application, Presentation, & Session Layers from the ISO/OSI model. It contains higher-level protocols, such as HTTP, SMTP, DNS, TELNET, FTP, etc.

# Unit 2

## Lecture 5: The Transport Layer

The Transport Layer takes the PDU from the Application Layer, and passes it down to the Internet Layer. The layer aims to provide efficient, reliable, and cost-effective services to the Application Layer (and therefore users).

The services that the Transport Layer implement can be found in:

- the Operating System kernel
- a separate user process
- a library package
- within the Network Interface Card

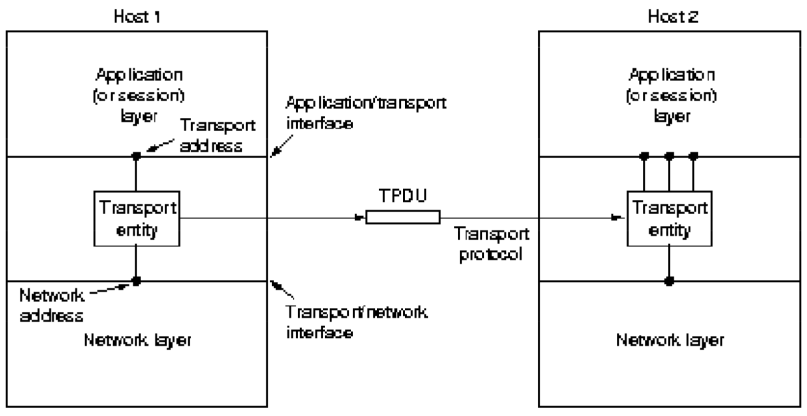

Figure 6: A diagram of logical communication between two Transport Layer entities.

The Transport Layer protocols provide logical communication between Application Layer processes. At the transmission side, Application messages are broken down into segments/datagrams for transmission. At the receiver side, the segments/datagrams are reassembled into messages to be passed back to the Application Layer. The layer aims to enhance the Network Layer reliability by allowing for detection &

compensation of lost/damaged packets. Since the service primitives are separate of Network Layer primitives, Application layer programs can be written with a standardised set of primitives.

To note is the difference between the Transport Layer & the Internet Layer: the Application Layer provides logical communication between PROCESSES, the Internet Layer provides logical communication between HOSTS.

The TCP/IP architecture has 2 main Transport Layer protocols: TCP, and UDP. TCP provides multiplexing, datagram integrity checking, flow control, sequencing, acknowledgements, timers & congestion control. UDP only provides multiplexing & integrity checking, but integrity checking is not made available as a service to higher layers. No delay or badwidth guarantees are given by either protocol.

For contrast, the ISO/OSI model has 5 separate Transport Layer protocols for different underlying services & connection types.

# Lecture 6: Reliable Data Transfer

The service implementation for the Transport layer is responsible for reliable data transmission. This is done by implementing several factors to mitigate the loss of data:

**Checksum** A method to convert a datagram into a small code that is transmitted along with the datagram. At the receiver side, the checksum is recomputed and compared against the transmitted one to see if errors occurred.

**Sequence Numbers** Each datagram has a unique number denoting at what point in transmission the datagram should be in. This allows messages to be reassembled in-order at the Rx side.

**ACK/NAK** Upon the receiving of a message with a given sequence number, the Rx side transmits an ACKnowledgement of that number. If the Rx side waits too long (usually $2t_{\text{prop}}$), then a NAK is sent instead. This alerts the Tx side to either send the next message or to resend the original message (assuming stop-and-wait transmission).

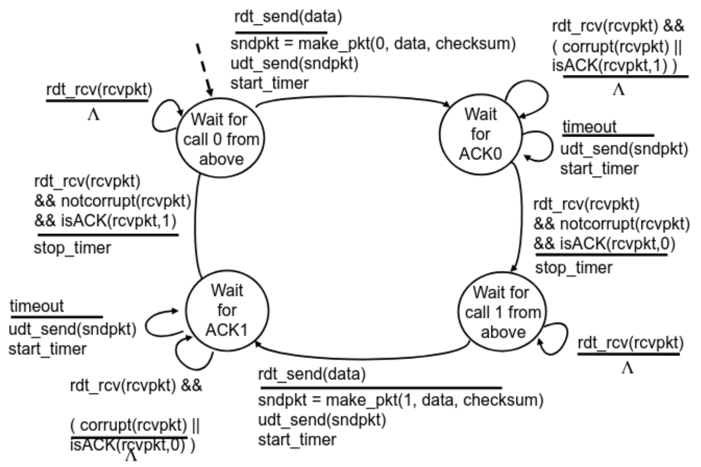
rdt_send(data)
sndpkt = make_pkt(0, data, checksum)
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt) && ( corrupt(rcvpkt) || isACK(rcvpkt,1) )
Λ
rdt_rcv(rcvpkt)
Λ
Wait for call 0 from above
Wait for ACK0
timeout
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt,1)
stop_timer
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt,0)
stop_timer
Wait for ACK1
Wait for call 1 from above
timeout
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt)
Λ
rdt_rcv(rcvpkt) &&
( corrupt(rcvpkt) || isACK(rcvpkt,0) )
Λ
rdt_send(data)
sndpkt = make_pkt(1, data, checksum)
udt_send(sndpkt)
start_timer

Figure 7: A finite state machine for reliable stop-and-wait transmission.

The figure shows a Finite State Machine of how a sender operates for a stop-and-wait reliable data transmission. When data from the Application layer is passed down to the Transport layer, it is converted to a datagram, with a sequence number & datagram included, before being transmitted. The sender then waits for an ACK for that datagram. If no ACK is received within reasonable time, then the datagram is retransmitted and the timeout resets. If an ACK is received, but for the wrong sequence number or the datagram is corrupted, the datagram is retransmitted and the timeout resets. If the ACK is correct, then the sender may move onto the next message to be sent.

Since stop-and-wait is used (where only one message is sent while waiting for an ACK.), only two sequence numbers are used (0 and 1). This is because only one packet will be in transmission at any time. The implementation of NAK is done by repeat transmission of the previous correct ACK.

## 6.1: Performance of Data Transmission.

Ideally, we'd like to max out our links (node-to-node channel) to be as busy as possible, so that there is as little time when nothing is being sent over a link. We can calculate the utilisation of a link by calculating the time to transmit a packet.

$$t_{\text{transmit}} = \frac{L}{R}$$

, where $L$ is the length of a packet in bits and $R$ is the transmission rate of the link in bits per second.

The utilisation of a link is then calculated as such:

$$U_{\text{sender}} = \frac{t_{\text{transmit}}}{2t_{\text{prop}}\ t_{\text{transmit}}}$$

Four our stop-and-wait RDT protocol, utilisation is very low as the sender must wait until the datagram has been ACK'd before sending a new datagram. This can be reduced by sending more packets before the ACK is received (Go-Back-N). Go-Back-N however increases the need for a large buffer at the sender, for if a packet early in transmission is lost/corrupted, the entire set of datagrams must be retransmitted. Selective Repeat is an alternative which only retransmits the datagram that didnt get received.

# Lecture 7: TCP & UDP

## 7.1: TCP

The TCP protocol is implemented via the Transport layer primitives. For TCP, this is usually done via Berkeley UNIX Sockets and made available to the Application layer through a network library (`netinet/tcp.h` for C programs in Linux). TCP provides reliable point-to-point connection-oriented data communication.

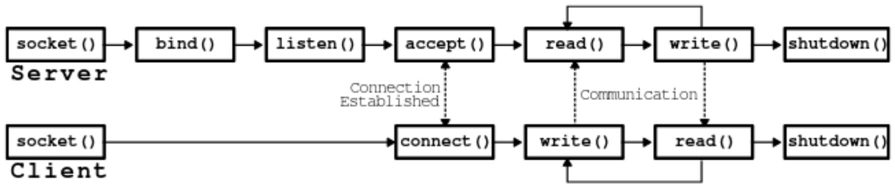

Figure 8: TCP socket communication via primitive calls.

The service primitives for TCP via Berkeley Sockets is as follows:

`socket()` Creates a new communication endpoint.

`bind()` Attaches a local address/port combination to a given socket.

`listen()` Announce willingness to accept connections.

`accept()` Wait until a connection attempt arrives, then establish connection.

`connect()` Send out a connection attempt.

`send()` Send data over established connection.

`receive()` Receive data from established connection.

`close()` Release the connection.

The header for TCP is at minimum 20 bytes and may grow with the inclusion of additional options. The Source Port & Destination Port are the ports which the message should be transmitted to/from (different port for different processes). The Sequence Number & Acknowledgement Number are counted in bytes and not segments. The Data offset is the header length in 32 bit words. A reserved section is included for future expansion. A section of control bits are included, with the following values:

**U** Urgent data

**A** ACK field is valid

**P** Push data now

**R** Reset connection

**S** Sync seq. numbers

**T** Teardown connection

The Window is the amount of bytes the receiver is able to accept. The Checksum is the Internet checksum, which is calculated as the ones-complement of all the 16 bit words of a shortened version of the header. The Urgent Pointer is used with the U control bit. The Options are a variable length section of additional options for TCP transmission, including alternate checksums, cryptography, and other alterations. The Options section is then padded to the nearest 32 bit word.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: The TCP datagram structure.

The initial sequence numberis randomly assigned to reduce the chance of collision with any other TCP communication on the network.

## 7.2: UDP

```
 0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                 |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|
|          data octets ...
+---------------- ...
```

Figure 10: The UDP datagram structure.

UDP is a connectionless unreliable Transport layer protocol. Its header is 8 bytes long, with only necessary information for sending & error checking. UDP only provides basic error checking & multiplexing of multiple Application layer message streams, via port numbers. UDP is suited for applications that send a lot of time-sensitive data, such as voice transmission or a video stream.

As there is no flow or congestion control, UDP transmission will use as much bandwidth it can. There is no sequencing, so packets may arrive out-of-place. The checksum is optional and is operated over a pseudo-header including the UDP header length, the IP addresses, and the IP protocol field.

# Lecture 8: TCP Features (1)

TCP has four algorithms to determine how the connection handles congestion. In addition to a 'receive window' for flow control, there is a 'congestion window' that governs the size of each packet sent within a RTT, so that the network is not overrun with packets all the maximum window size allowed.

The first algorithm dictates how to start transmission at a smooth rate, called Slow Start.

1. Upon the TCP handshake or timeout, one TCP packet is sent and an ACK is received.
2. Upon successful reception, the CongWin size is doubled until a threshold is reached.

The next algorithm, called Congestion Avoidance, dictates how to continue once the threshold is reached.

1. If no congestion occurs, the CongWin is increased by 1 TCP segment per RTT
2. The CongWin increases until TCP is finished transferring or CongWin is equal to the RecvWin.

If at any point there is no ACK received for a given datagram, the process starts over with Slow Start, with a threshold half that of its previous value. The threshold usually starts around 64kB.

A special case is Fast Retransmit, which will take over the Slow Start algorithm under the condition that three ACKs for a datagram have been received. In the case for Fast Recovery, we assume the next in-order datagram has been lost and we retransmit only that packet to reduce unnecessary retransmission & wasted bandwidth.

The final algorithm is Fast Recovery, which will take over after Fast Retransmit. The transfer of data is resumed with the Congestion Avoidance algorithm, with the modification that the CongWin size is halved.

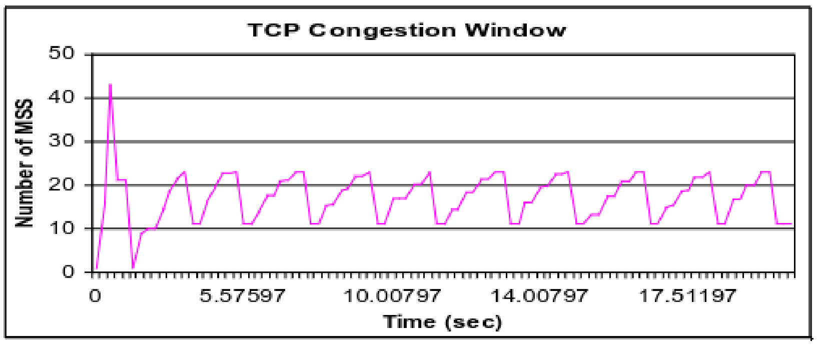

Figure 11: A diagram of a TCP connection with all 4 algorithms.

# Lecture 9: TCP Features (2)

## 9.1: TCP Retransmission Timer Calculation

The retransmission time decides how long the TCP connection should wait until retransmission. TCP estimates the round trip time (RTT) by sampling the RTT at each round trip. The timeout period shoud be greater than the measured RTT, so there is a lack of packet timeout and retransmission due to variations in the RTT.

At the start of a connection, the smoothed round trip time (SRTT) is set to a number greater than the the first RTT measurement (R). The round trip time variation (RTTVAR) is initially estimated to be half of R. The retransmission timer (RTO) is then set to be equal to SRTT and the maximum between the lowest change in time the system can detect and quadruple the RTTVAR.

$$\begin{aligned} SRTT &\leq R \\ RTTVAR &\leq R/2 \\ RTO &\leq SRTT + \max(G_{\text{clock}}, 4 * RTTVAR) \end{aligned}$$

For subsequent RTT measurements, the SRTT is updated according to the algorithm below. Smoothing factors, $\alpha$ and $\beta$, are applied to apply weight to previous SRTT values.

$$\begin{aligned} SRTT &\leq (1-\alpha) * SRTT + \alpha * R \\ RTTVAR &\leq (1-\beta) * RTTVAR + \beta + |SRTT - R| \\ RTO &\leq SRTT + \max(G_{\text{clock}}, 4 * RTTVAR) \\ \alpha &= 1/8; \quad \beta = 1/4 \end{aligned}$$

## 9.2: TCP Flow Control

The RecvWin field in the header maintains a count of how much space is left in the receiver buffer. TCP will not send more data than RecvWin says is free.

At the sender, an algorithm called Nagle's Algorithm is used to reduce the amount of small packets sent over a connection, by stopping the transmission of 1 byte packets, instead combining these small packets together. This is because the TCP header is 40 bytes or above in size, resulting in low utilisation of the connection.

At the receiver, an algorithm called Clark's Algorithm is used in a similar way. It tries to prevent the receiver from sending constant small increases of the RecvWin size, instead waiting until the size is greater than or equal to the maximum segment size, or the buffer is half empty.

## 9.3: TCP Fairness

When multiple TCP connections share a link on a network, TCP will aim to give each link an equal share of the available bandwidth so no link is blocked longer than others.

# Unit 3

## Lecture 10: IPv4

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: The IPv4 Packet Header

At the Network Layer of the TCP/IP architecture, communication is 'routed' through nodes on a network between the start and end of a route. Each node (host, router, gateway, etc.) in the route needs an 'address' to uniquely identify itself. The Internet Protocol version 4 (IPv4) utilised an addressing scheme where each address is 32 bits, with the address being broken down into a network id (net-id) and a host id (host-id). The address is usually depicted as four bytes in decimal demarked by a period. Multiple different schemes were used to better utilise the 32-bit addresses as the Internet grew.

### 10.1: Class-based Addressing

Originally, the address range was split into 5 separate classes from A-E, where each class has a different size of host-id and net-id or use case.

**Class A** 8-bit net-ID & 24-bit host-ID. From 0.0.0.0 to 127.255.255.255. For very large networks (i.e. very large companies)

**Class B** 16-bit net-ID & 16-bit host-ID. From 128.0.0.0 to 191.255.255.255. For large networks (i.e. large companies).

**Class C** 24-bit net-ID & 8-bit host-ID. From 192.0.0.0 to 223.255.255.255. For small networks.

**Class D** From 224.0.0.0 to 239.255.255.255. For multicast addressing.

**Class E** From 240.0.0.0 to 255.255.255.255. Reserved.

There is an issue that a network may be given a larger block than necessary. For example, if a network has 257 hosts, they would be assigned a class B address space, which gives them 65279 unused addresses.

## 10.2: Subnets

Subnetting is an improvement of class-based addressing, by allowing a network to be split into sub-networks (or subnets). The given host-ID range can now be split further by the use of a ‘subnet mask’, allowing a variable size of hosts depending on use case. The net-ID still is fixed by class-based addressing.

## 10.3: Classless Addressing

Classless Inter-Domain Routing (or CIDR) allows the net-ID to be any size via the use of a mask, akin to subnetting. The address is now specified as `a.b.c.d/n`, where n indicates the size of the net-ID in bits.

## 10.4: Network Address Translation

The previous methods are still limited by the size of the address range of IPv4, which is around 4.3 billion addresses. This method utilises the private address ranges, which are not used in the wider Internet, combined with the Transport Layer port numbers to allow for these addresses to be used. This requiredthat routers & gateways were capable of translation/de-translation of these addresses due to the way these span multiple network layers.

# Lecture 11: IPv6

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: The IPv6 Packet Header

IPv6 is a long-term solution to the dwindling amount of IPv4 addresses. The main difference is the address size increasing from 32 bits to 128 bits, allowing for 340 undecillion possible addresses. Other features include: a simplified header, greater security, autoconfiguration, quality of service guarantees and support of mobile computing.

As can be seen from the header format, IPv6 allows for a variable amount of headers, sent in-order. The Next Header field indicates the type of the next header. Extension headers may include routing, multicast, fragmentation or authentication options.

IPv6 addresses are depicted as eight 16-bit segments demarked by a colon. Each segment is usually depicted in hexadecimal. Leading zeros may be dropped and segements of all zeros may be omitted.

## 11.1: IPv6 Addressing Schemes

An IPv6 address type is denoted by a binary string at the start of the address. Common types include:

**0000 0000** Reserved. Usually used for embedding an IPv4 address within an IPv6 address.

**0000 001** Reserved. Usually used for NSAP addresses, an OSI addressing scheme for the Network layer.

**1111 1110 10** Link-local unicast address. Used for addressing within a subnet.

**1111 1110 11** Site-local unicast address. Used for addressing within a private network.

**1111 1111** Multicast address. Used to send messages to all nodes within a group, with some default addresses for link-local, site-local, or router multicasting.

Other addresses are usually unicast (to one device), or anycast (to nearest device in a group).

## 11.2: IPv6 Interoperability

Since it would be impractical to switch off IPv4 support and adopt IPv6 at once, we must instead allow for IPv6 to be able to communicate with IPv4. This causes 3 challenges:

1. IPv4 & IPv6 hosts communicating with one server.
2. IPv6 source & destination communicating over and IPv4 route
3. IPv6 host communicating with an IPv4 host.

For case 1, the server must have a ‘dual-stack’ to allow for both IPv4 & IPv6 support simultaneously.

For case 2, the hosts/gateway should have a dual-stack and allow for tunneling, where the IPv6 packet is encapsulated within an IPv4 packet.

For case 3, either the NAT must translate IPv6 packets to IPv4 or the IPv6 host must translate its own packets to IPv4.

# Lecture 12: Routing

Routing is an Internet layer function referring to selecting paths within a network along which to send data. This function is usually handled by routers, which contain specialised hardware to route packets efficiently. A gateway is a special type of router meant for bridging heterogenous networks, where translation between protocol types may be needed.

## 12.1: Router Architecture

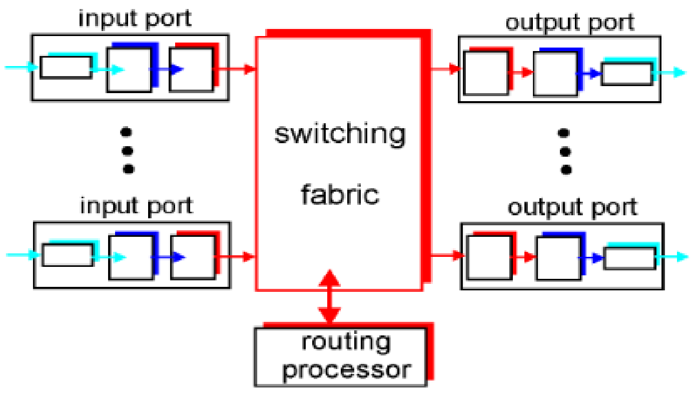

Figure 14: A generalised architecture of a router.

The architecture of a router is composed of four parts. The input ports handle the incoming data from a link. The output ports handle sending the outgoing data to a link. The switching fabric handles the routing from an input port to an output port. The routing processor is responsible for managing the switching fabric and handling router communication.

Both the input & output ports have buffers. The input port needs a buffer if the switching fabric is not able to handle the incoming data (due to output port contention), so that incoming data can be queued. The output port needs a buffer to store outgoing packets when the outgoing link speed is too slow to send each packet in time. Care must be taken to ensure that the buffers do not overflow via flow control & contention control.

## 12.2: Routing Algorithms

A routing algorithm needs to find the 'least-cost' path between two nodes. This cost may be time, inversely related to bandwidth or congestion, or equal to 1. Routing algorithms are either global, where every router has complete topology information, or decentralised, where every router only knows its connections & costs to its neighbours. An algorithm may also depend on whether the topology changes slowly (static) or frequently (dynamic).

A link-state routing algorithm is one where the entire topology is known (global). A distance-vector routing algorithm is one where only paths & costs between nodes is knowm.

# Network Security

## Lecture 13: Symmetric & Public Encryption Techniques

For a secure message, we expect:

- the message is confidential between sender and receiver
- the message received is authentic (the message is from the sender and not a third party)
- the message has not been tampered in transmission (the message integrity isnt comprimised)
- the message will be sent even under non-ideal conditions (denial of service)

For an unsecure channel between the transmitter (Alice) and the receiver (Bob), an eavesdropper (Eve) may intercept, delete or modify messages or deny service across the channel.

### 13.1: Symmetric Key Cryptography

To prevent eavesdropping over the channel, the messages may be encrypted. For a message $m$, the message may be encrypted by Alice's encryption key $K_A(m)$. Upon receiving, the message may be decrypted via Bob's key $K_B(K_A(m)) = m$.

The encryption key is a function to encrypt the message. A simple function would be the substitution cipher, which substitutes each character with another character in the alphabet. There are $10^{26}$ permutations for possible mappings, for which a brute force approach would take around 10 million years to check every permutation. However, this cipher is weak to distribution of letters in the English language, so the mapping may be guessed (statistical analysis).

Another approach is the one time pad, where a random sequence of bits (key $K_S$) is XOR'd with every single bit in the message $m$ by Alice. Bob must then XOR the message with the key $K_S$ to decrypt the message. This requires that Alice and Bob both have the key before messages are sent. This key must be sent securely, and the key should only be used once and that the key is as long as the message to prevent statistical analysis.

$$K_A(m) = m \bigoplus K_S$$
$$m = K_B(K_A(m)) = K_A(m) \bigoplus K_S$$

These are examples of symmetric encryption. Symmetric key encryption schemes are encryption schemes where the same key is used on encryption and decryption.

## 13.2: Public Key Cryptography

Public key cryptography (or asymmetric key encryption) is an encryption scheme where the encryption key and decryption key are different. The key to encrypt ($K_B^+$) is public to Alice, but the key to decrypt ($K_B^-$) is private. This scheme is more efficient than one time pad, since there is much less changing of keys.

The first widespread implementation of public key encryption is the RSA algorithm, which exploits 'one-way' functions, where a function is easy to go one way ($N = p * q$), but very computationally hard to go back. The security of RSA relies on this, and has a complexity of $O((\log N)^{(\log N)^{1/3}})$, assuming that N is large.