COMP2221 Networks

David Head

University of Leeds

Lecture 3

Reminder of the last lecture

Last time we looked at **network layer** models, in particular the 5-layer TCP/IP model most often used for the internet:

- Application layer (at the 'top').
- Transport layer.
- Network layer.
- Link layer.
- Physical layer (at the 'bottom').

Data is sent from the application layer down through the **protocol stack**, and up the stack at the other side once it reaches the destination.

Today's lecture

Today we will learn about:

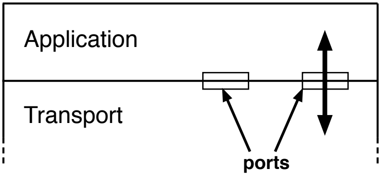
- **Ports**, including common ports.
- **Network communication**, as performed by all layers below the *Application* layer.
    - Transport layer.
    - Network layer.
    - Link layer.
    - Physical layer.

Ports are one of the concepts we need before we can start programming. The other is IP addressing, which we start looking at next lecture.

Ports

Applications are linked to the Transport layer *via* **network ports**.

- Different port numbers can be associated with different Application layer processes.
- This allows multiple applications to run simultaneously on the same host with the same IP address.
- Purely software, provided by the OS.
- Can be allocated to a particular service, *e.g.* email, HTTP *etc.*



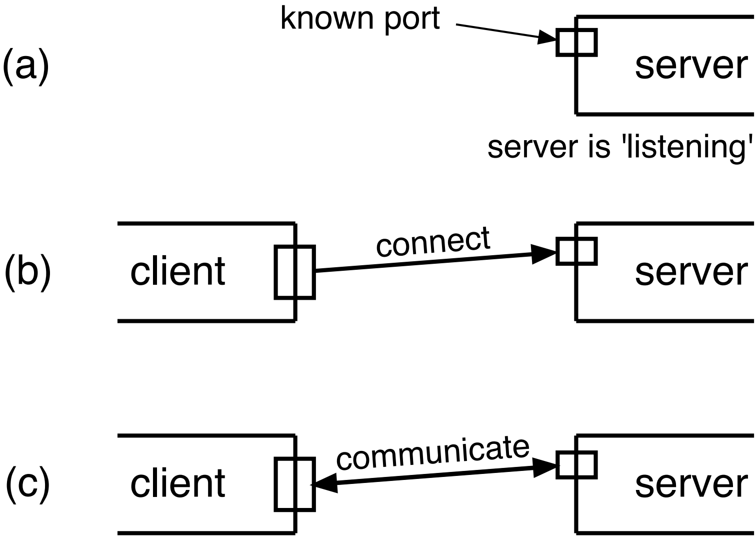
Sending and receiving

Converting host-to-host delivery to process-to-process delivery is the job of the Transport layer.

- Known as **multiplexing** (sending) and **demultiplexing** (receiving).

Applications typically publish the port they are **listening to** (receiving on).

- A sending process will attempt to establish a connection using the published port number.
- Will continue to communicate using the same port number, but the port can also be used to initiate new contacts.


(a)
known port
server
server is 'listening'
(b)
client
connect
server
(c)
client
communicate
server

Available ports

Ports 1-65535 are available on any given host.

- 16-bit unsigned int, with zero not allowed.
- TCP and UDP ports and **independent**.

Ports are characterised by:

- Ports 1-1023 are **reserved**; approved by IANA (Internet Assigned Numbers Authority).
- Commonly used ports lie outside this range, *i.e.* 1024-65535.
- We can use any freely available ports in this range.

To see ports on a UNIX machine, look at `/etc/services`

Remote login and file transfer

`telnet`

- Port 23.
- Short for teletype network.
- The original client-server communication protocol.
- Now rarely used as it is insecure.

`ssh`

- Port 22.
- Secure shell.

`ftp`

- Stands for file transfer protocol.
- Port 21 for commands (`dir`, `put`, `get`, *etc.*)
- Port 20 used for data (the original use; slow).

Email

`smtp`

- Port 25.
- Stands for simple mail transfer protocol.
- Used for sending mail.

`imap`

- Port 143.
- Used to access mail.
- Stands for internet message access protocol.

`pop3`

- Port 110.
- Also used to access mail.
- Stands for post office protocol.

Web

`http`

- Port 80.
- Stands for hypertext transfer protocol.
- Insecure.

`https`

- Port 443.
- Secure version of `http`, requiring authentication.
- We will look at security and authentication in Lecture 13.

Network communication

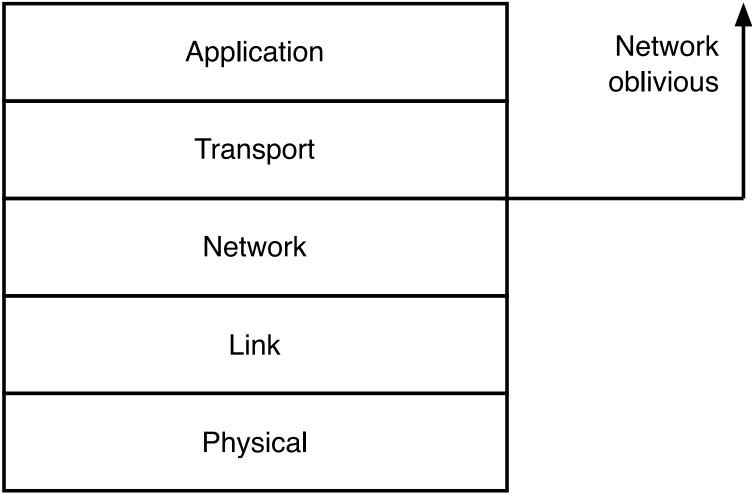
At the Application layer, the data to be sent to the destination host and port number is packaged for transport.

The Transport layer is oblivious to the subnet that enables the communication.

- Adds a header that includes information such as port numbers, checksum *etc.*

At the Network layer and below, the subnet **is** explicitly involved.

- *e.g.* The Network layer determines which network nodes the message will pass through.


Application
Transport
Network
Link
Physical
Network oblivious

There are two key protocols for the Transport layer:

- **UDP** : User Datagram Protocol.
- **TCP** : Transmission Control Protocol.

They differ in:

- The information provided in their headers.
- The reliability of service.
- Performance.

They both have a checksum for data integrity.

UDP

UDP is a **connectionless** protocol.

- Each message is independent.

Prepends an 8-byte header to the message including:

- Destination port number.
- Source port number.
- Message length.
- Checksum.

The ports identify the processes on both the source and destination hosts.

Has a checksum for data integrity, but simply discards data segments that have been corrupted.

UDP Header

Source port (2 bytes)	Destination port (2 bytes)
Length (2 bytes)	Checksum (2 bytes)
Message data . . .	

Note there are no host (IP) addresses in the header for UDP, and the same is true for TCP *(see below)*.

- This is the responsibility of the Network layer.

Uses of UDP

An important use of UDP is to access a DNS (Domain Name System) server, to map a readable internet address to an IP address.

- We will cover this next lecture.

Also useful for **real-time multi-media**, often wrapped into the **Real-time Transport Protocol** (RTP)

- Sits between Application and Transport layers.
- UDP packets are numbered such that receiver can determine if packets are missing.
- No correction or retransmission.
- Receiver can interpolate lost data.

TCP

Connection-oriented protocol, *i.e.* maintains a persistent connection between the two hosts.

- Ensures **reliable data transfer**, possibly by requesting re-transmission of lost or corrupt segments.
- Also provides a degree of **congestion control**.

20-byte header with:

- Sequence and acknowledgement numbers.
- Bits used for maintaining the connection.
- Some specialist fields.

The remaining fields are the same as UDP.

The sequence and acknowledgement numbers are used to guarantee ordering, and to check for missed packets.

TCP Header

<table>
<tr><td>Source port (2 bytes)</td><td>Destination port (2 bytes)</td></tr>
<tr><td colspan="2">Sequence number (4 bytes)</td></tr>
<tr><td colspan="2">Acknowledgement number (4 bytes)</td></tr>
<tr><td>Various flags (2 bytes)</td><td>Receive window (2 bytes)</td></tr>
<tr><td>Checksum (2 bytes)</td><td>Urgent data ptr (2 bytes)</td></tr>
<tr><td colspan="2">Options (may be empty)</td></tr>
<tr><td colspan="2">Message data ...</td></tr>
</table>

Most of these fields do not concern us; see *e.g.* Kurose and Ross, 7^{th}ed, §3.5.2, for more details.

Network layer - IP

Forwards Transport layer data with a 20-byte (IPv4) / 40-byte (IPv6) header *[see also Lecture 17]*.

- Source and destination address (IPv4 or IPv6).
- Protocol (TCP or UDP).
- Checksum for the header integrity (IPv4).
- Time-to-live.

The **time-to-live** decrements whenever the datagram (packet) passes through a network node. Once it reaches zero, the message is discarded.

- This avoids datagrams that circulate forever.

Link and physical layers

Frames of data are forwarded to the Physical layer. For example, **ethernet frames** have a 22-byte header containing:

- **Preamble**, used for synchronisation.
- MAC address (Media Access Control) for source and destination.
- A **type** field (*e.g.* for ARP; cf. Lecture 19).

Also a **checksum**, although this is often in a **footer**.

The MAC address enables the frame to 'hop' to the next network device.

The checksum is for frame integrity.

Altogether

Link layer header (*e.g.* 22 bytes for Ethernet)
Network layer header (20 or 40 bytes)
Transport layer header (8 or 20 bytes)
Message . . .

Note each header has a source and destination **‘address’**:

- **Ports** in between the Transport and Application layers.
- **IP Addresses** in the Network layer.
- **MAC Addresses** in the Link layer.

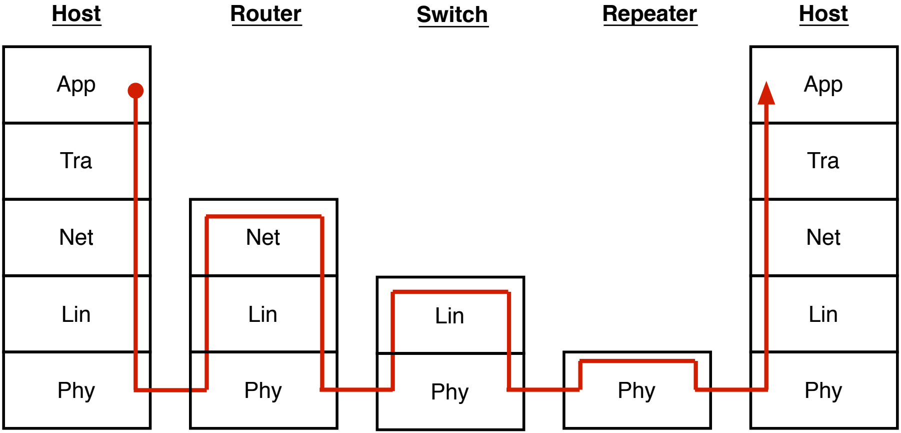
Network Transmission

The route from source to destination passes through multiple devices (typically).

Network-layer **routers** determine the path between hosts.

Link layer **bridges** and **switches** forward frames between network components.

Physical layer **repeaters** and **hubs** regenerate the signal, thereby enabling a greater range.


Host
Router
Switch
Repeater
Host
App
Tra
Net
Lin
Phy
Net
Lin
Phy
Lin
Phy
Phy
App
Tra
Net
Lin
Phy

Summary

- Headers are prepended to data at most levels as it is being sent, and removed at the same level as it is being received.
- Naming and addressing is key.
    - **MAC addressing** at the Link layer (not relevant to programming).
    - **IP addressing** at the Network layer, allows high-level specification of source and destination.
    - **Port numbering** at the Transport layer allows specific referencing to applications.

Next time we will look at DNS, the Domain Name System.