Session 5: Networking

Dr John Fawcett

What’s the problem?

We want computers to be able to communicate with each other.

Key challenges:

1. Long distance
2. Scale – there are lots of computers in the world
3. Heterogeneity – computers are not all the same
4. Handling failures and disconnections
5. Security

7 layers

The OSI Reference Model for digital communications systems has 7 layers:

1. Physical Layer
2. Datalink Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

Layer 1 is lowest level; layer 7 is the highest layer.

Simplicity and Complexity

We need our networks to operate at extremely high data rates.

We can only achieve this if the hardware and software involved is extremely simple.

We also want very sophisticated networks, which suggests complexity!

The solution is modularity. We have many implementations at each layer, each kept very simple and only offering one option. Any single network is a combination of layers so there are very many different networks that we can build but any single network can only work in one way.

Layer 1: Physical Layer

Pick a property of the physical world that you can manipulate and measure.

- Sound waves
- Radio waves
- Flashes of light
- Electrical potential between the earth and a single cable
- Difference in electrical potential between two cables
- Water pressure
- Streams of particles
- …

Physical layers might be 1-to-1 but usually *shared-medium networks* (multi-drop).

Layer 1: Physical Layer

The physical layer conveys *symbols* from one place or time to another. The *alphabet* is the set of supported symbols.

- A binary alphabet has two symbols: 0 and 1, or high and low.
- Digital television commonly uses a 64-symbol alphabet.

More symbols ⇒ faster, because more information gets through per symbol sent.

BUT we need a tradeoff because…

More symbols ⇒ harder to tell them apart (they’re closer together) so takes more time to identify a symbol.

Layer 1: Physical Layer

We also need to worry about background noise: nature, and other people, using the same physical property as us.

- Cosmic background radiation
- Lightning strikes
- …

The physical layer needs to represent its symbols with some noise immunity, i.e. so that the levels of noise we expect to encounter do not cause the receiver to think they have seen the wrong symbols.

Result from theoretical computer science: it is impossible to eliminate all bit errors!

Layer 1: Physical Layer

We also need to worry about *clock recovery*: if I send a very long sequence of the same symbol, can the receiver really distinguish 999 from 1000 of them? This is *clock recovery*: they need to recreate the sender's clock so they can count symbols.

An easy method is to send the clock as a binary-encoded (high, low, high, low, …) signal on an additional wire. This uses a lot of energy, and another wire, so is not always practical.

Another method is to encode the user's data into symbols in such a way that we can guarantee long sequences of the same symbol cannot occur. Doing this without decreasing the bit rate becomes a difficult mathematical exercise.

Layer 2: Datalink Layer

We divide layer 2 into two sublayers:

1. Medium Access Control
2. Logical Link Control

When we have lots of computers ‘connected’ to the same medium, we need medium access control to decide who gets to transmit next.

When there are symbols flying about all the time, logical link control tells us which are intended for us, which are for other computers, and which are broadcast (for everyone).

Layer 2: Datalink, Medium Access Control

Before you can say that “it’s X’s turn”, you have to a way to identify who ‘X’ is! Could be their socket number, their radio frequency, etc.

Then you need some rules to determine the schedule of who speaks next.

If every device needs to use the same bandwidth (telephones), you can let each device transmit in a rotating sequence (round robin).

It’s more difficult when the devices have different or variable demands. One option is to tell each device to listen before trying to transmit and “go for it” if no-one else is currently speaking. If two devices collide, then try again a random amount of time later. There are various ways to pick random delays that have a low chance of causing further collisions.

Layer 2: Datalink, Logical Link Control

We distinguish the *physical link*, which is shared by all the devices connected to it, from *logical links*, which temporarily use the physical link to connect one sender to one or more receivers.

To achieve this we need…

1. To give transmittable names to each device
2. Framing: a way to tell which symbols belong together to form a single message.

Framing could be implicit (e.g. all messages are 20 bytes long), or explicit (e.g. messages begin with a value describing what follows, including its length).

Framing has to work even if you start listening part-way through a message! This causes most ‘obvious’ solutions to fail! Framing is harder than you think!

Layer 3: Network Layer

Layer 1 allowed us to encode our ideas and transfer them to another place or time. Layer 2 built a local network on top, allowing point-to-point and broadcast communications over a shared medium.

Layer 3 scales this up to global, and now into space.

The challenges tackled in layer 3 are:

- Heterogeneity – a global network needs to connect different types of Datalink together, which create an encapsulation problem: presenting a common API over different L2s.
- Scale – we cannot use the small-scale solutions that worked at layer 2
    - Global unique addressing?
    - Global message routing?

Layer 3: Network, Heterogeneity

A Layer-3 message is address to a Layer-3 endpoint address (of course), but how do we translate that to a Layer-2 address for the single (one) next hop, and how do we translate ("*resolve*") the same Layer-3 address to a different Layer-2 address at the next hop – what's changed?

Different Layer-2s have different maximum frame sizes. Layer-3 protocols need to *fragment* and *reassemble* messages differently to encapsulate them inside each Layer-2 protocol. As a message makes its way through the full network, it might need to be encapsulated and re-encapsulated several times to complete the full journey. How do we make this work at all? How do we make this efficient?

Layer 3: Network, Global scale

We needed unique datalink addresses for each device at Layer 2 but our solution does not scale: we cannot have a network administrator employed to run the global (universal?) network! We also knew that we could send data directly to any other device because everything was connected to a shared medium. Both become problems.

We need to impose some kind of structure on addresses to make it easier to allocate them without duplicates. One consequence is that devices change their Layer-3 address as they move around even though they often do not need to change Layer-2 address.

We also need global routing, and we need to allocate addresses in ways that minimise the size of the routing databases so our routers do not run out of memory as the network becomes larger and larger.

Layer 4: Transport Layer

Notice that layers 1, 2 and 3 required a presence to implement their protocols at every node in the network.

Layer 4 and above are end-to-end only: their implementations only require a presence at the endpoints.

Layer 4 abstracts the common functionality that lots of programs require so application developers do not have to keep re-inventing solutions to the same problems.

Let’s look at what these features are…

Layer 4: Transport Layer

Ordering: ensuring receivers process data in the intended order, even if messages take different paths through the network and arrive out of order. Removal of duplicates.

Error correction.

Loss recovery: detect lost (or very delayed) messages and re-send them.

Flow control: don’t send too fast for the recipient.

Congestion control: don’t send too fast for the network.

Stream abstraction: application developers are freed from dividing their data into packets.

Multiplexing: sharing the network connection between applications.

Layer 5: Session Layer

Extending the idea of a conversation across multiple connections, either at the same time or over an extended period.

Similar to seeing your friends every morning in school: you haven't forgotten the *context* of your conversations yesterday and can refer back to that shared knowledge for compression and privacy.

Multiple connections in use at once allows different types of traffic to use different networks with different characteristics, e.g. audio and video streams, file transfer and audio, etc.

Layer 6: Presentation Layer

This is where we harmonise the different ways to say the same thing.

Some differences are caused by hardware choices, e.g. big-endian or little-endian.

Some differences are caused by software choices, e.g. how to encode strings, how many bytes in an integer, are integers signed or unsigned, how to represent floating point numbers, …

Some differences are caused by algorithms containing arbitrary decisions, e.g. we saw many ways to encode the same graph as bytes.

Presentation layer protocols “pick one” way of doing it so everyone can understand each other’s data formats.

Layer 7: Application Layer

The application layer is where we build protocols that are specific to the program we want to create. Examples:

- HTTP: required for the application of web serving / browsing
- SMTP: required for the application of sending / receiving email
- FTP: required for the application of file transfer

Now it’s time for you to have a go

Exercise sheet!