Networks

The OSI Model

The seven OSI layers are

1. Physical
2. Data Link
3. Network
4. Transport
5. Session
6. Presentation
7. Application

Networks

The OSI Model: Physical Layer

The *physical layer* (PHY) or *layer 1* is the hardware layer and deals with the transmission of bits over a channel

Networks

The OSI Model: Physical Layer

The *physical layer* (PHY) or *layer 1* is the hardware layer and deals with the transmission of bits over a channel

For example:

Networks

The OSI Model: Physical Layer

The *physical layer* (PHY) or *layer 1* is the hardware layer and deals with the transmission of bits over a channel

For example:

- what voltages to use or colours of light pulses or radio wavelengths to use

Networks

The OSI Model: Physical Layer

The *physical layer* (PHY) or *layer 1* is the hardware layer and deals with the transmission of bits over a channel

For example:

- what voltages to use or colours of light pulses or radio wavelengths to use
- what encoding for bits; how long (in time) a bit should be

Networks

The OSI Model: Physical Layer

The *physical layer* (PHY) or *layer 1* is the hardware layer and deals with the transmission of bits over a channel

For example:

- what voltages to use or colours of light pulses or radio wavelengths to use
- what encoding for bits; how long (in time) a bit should be
- how many wires to use in a cable

Networks

The OSI Model: Physical Layer

The *physical layer* (PHY) or *layer 1* is the hardware layer and deals with the transmission of bits over a channel

For example:

- what voltages to use or colours of light pulses or radio wavelengths to use
- what encoding for bits; how long (in time) a bit should be
- how many wires to use in a cable
- what plugs and sockets to use on the cable

Networks

The OSI Model: Physical Layer

The *physical layer* (PHY) or *layer 1* is the hardware layer and deals with the transmission of bits over a channel

For example:

- what voltages to use or colours of light pulses or radio wavelengths to use
- what encoding for bits; how long (in time) a bit should be
- how many wires to use in a cable
- what plugs and sockets to use on the cable
- and many more

Networks

The OSI Model: Physical Layer

The *physical layer* (PHY) or *layer 1* is the hardware layer and deals with the transmission of bits over a channel

For example:

- what voltages to use or colours of light pulses or radio wavelengths to use
- what encoding for bits; how long (in time) a bit should be
- how many wires to use in a cable
- what plugs and sockets to use on the cable
- and many more

Generally, anything to do with choices regarding hardware

Networks

The OSI Model: Data Link Layer

The *data link layer*, also called the *media access layer* (MAC) or *layer 2*, takes the physical layer and tries to create a channel where there are no undetected errors of transmission

Networks

The OSI Model: Data Link Layer

The *data link layer*, also called the *media access layer* (MAC) or *layer 2*, takes the physical layer and tries to create a channel where there are no undetected errors of transmission

Note “undetected”: we know networks are not 100% reliable (e.g., wireless networks in particular) so we presumably want to take into account possible errors and deal with them: the ISO standard recommends you think about that here

Networks

The OSI Model: Data Link Layer

The *data link layer*, also called the *media access layer* (MAC) or *layer 2*, takes the physical layer and tries to create a channel where there are no undetected errors of transmission

Note “undetected”: we know networks are not 100% reliable (e.g., wireless networks in particular) so we presumably want to take into account possible errors and deal with them: the ISO standard recommends you think about that here

A typical MAC layer sends the data as a sequence of *frames* (recall the packet nature of the Internet). A frame is a chunk of bytes, maybe tens or thousands of bytes long

Networks

The OSI Model: Data Link Layer

If a frame is corrupted, maybe the MAC layer can resend it; or send a message to the next layer indicating a problem

Networks

The OSI Model: Data Link Layer

If a frame is corrupted, maybe the MAC layer can resend it; or send a message to the next layer indicating a problem

A popular choice in real standards is to do nothing at all: let a higher layer figure out what's gone wrong and choose a remedy

Networks

The OSI Model: Data Link Layer

If a frame is corrupted, maybe the MAC layer can resend it; or send a message to the next layer indicating a problem

A popular choice in real standards is to do nothing at all: let a higher layer figure out what's gone wrong and choose a remedy

Again: it is up to the standard we are designing as to what actually happens. The layering model just says it is a good idea to consider this kind of thing here

Networks

The OSI Model: Data Link Layer

If a frame is corrupted, maybe the MAC layer can resend it; or send a message to the next layer indicating a problem

A popular choice in real standards is to do nothing at all: let a higher layer figure out what's gone wrong and choose a remedy

Again: it is up to the standard we are designing as to what actually happens. The layering model just says it is a good idea to consider this kind of thing here

In real implementations, this layer is often strongly intertwined with the physical layer and we tend to talk about both of them together

Networks

The OSI Model: Network Layer

The *network layer*, *layer 3*, controls the operation of the network, particularly the issue of routing data from source to destination

Networks

The OSI Model: Network Layer

The *network layer*, *layer 3*, controls the operation of the network, particularly the issue of routing data from source to destination

Also, it can deal with *congestion*: where there is too much data for a particular link it might route some data via another link, or use *flow control* to slow down the rate of transmission

Networks

The OSI Model: Network Layer

The *network layer*, *layer 3*, controls the operation of the network, particularly the issue of routing data from source to destination

Also, it can deal with *congestion*: where there is too much data for a particular link it might route some data via another link, or use *flow control* to slow down the rate of transmission

Or speed up the rate if things are going well

Networks

The OSI Model: Network Layer

The *network layer*, *layer 3*, controls the operation of the network, particularly the issue of routing data from source to destination

Also, it can deal with *congestion*: where there is too much data for a particular link it might route some data via another link, or use *flow control* to slow down the rate of transmission

Or speed up the rate if things are going well

Accounting might be managed in this layer: counting the number of bits so we can bill the user

Networks

The OSI Model: Network Layer

The *network layer*, *layer 3*, controls the operation of the network, particularly the issue of routing data from source to destination

Also, it can deal with *congestion*: where there is too much data for a particular link it might route some data via another link, or use *flow control* to slow down the rate of transmission

Or speed up the rate if things are going well

Accounting might be managed in this layer: counting the number of bits so we can bill the user

And *quality of service*: e.g., ensuring there is always enough bandwidth to stream a video

Networks

The OSI Model: Transport Layer

The *transport layer*, *layer 4*, accepts data from the session layer (layer 5) and arranges it into packets suitable for the network layer: *packetisation*

Networks

The OSI Model: Transport Layer

The *transport layer*, *layer 4*, accepts data from the session layer (layer 5) and arranges it into packets suitable for the network layer: *packetisation*

Similarly, it takes packets from the network layer (layer 3) and reassembles them into the original data stream: *depacketisation*. This might need to deal with packets arriving out of order

Networks

The OSI Model: Transport Layer

The *transport layer*, *layer 4*, accepts data from the session layer (layer 5) and arranges it into packets suitable for the network layer: *packetisation*

Similarly, it takes packets from the network layer (layer 3) and reassembles them into the original data stream: *depacketisation*. This might need to deal with packets arriving out of order

You might want to think about reliability in this layer: ensuring the data received is the same as the data sent. No corruption or loss in the data

Networks

The OSI Model: Transport Layer

The *transport layer*, *layer 4*, accepts data from the session layer (layer 5) and arranges it into packets suitable for the network layer: *packetisation*

Similarly, it takes packets from the network layer (layer 3) and reassembles them into the original data stream: *depacketisation*. This might need to deal with packets arriving out of order

You might want to think about reliability in this layer: ensuring the data received is the same as the data sent. No corruption or loss in the data

Curiously, reliability is not always a requirement of a network!

Networks

The OSI Model: Session Layer

The *session layer*, *layer 5*, manages *sessions* between source and destination.

Networks

The OSI Model: Session Layer

The *session layer*, *layer 5*, manages *sessions* between source and destination.

- Establishing and terminating connections; e.g., a remote login session

Networks

The OSI Model: Session Layer

The *session layer*, *layer 5*, manages *sessions* between source and destination.

- Establishing and terminating connections; e.g., a remote login session
- Restarting interrupted connections

Networks

The OSI Model: Session Layer

The *session layer*, *layer 5*, manages *sessions* between source and destination.

- Establishing and terminating connections; e.g., a remote login session
- Restarting interrupted connections

Sessions can be quite short, e.g., just long enough for an email or Web page to be transmitted; or arbitrarily long

Networks

The OSI Model: Session Layer

The *session layer*, *layer 5*, manages *sessions* between source and destination.

- Establishing and terminating connections; e.g., a remote login session
- Restarting interrupted connections

Sessions can be quite short, e.g., just long enough for an email or Web page to be transmitted; or arbitrarily long

In general, a session is just some logically connected set of exchanges that have some unified identity

Networks

The OSI Model: Session Layer

For example, if the network crashes and reboots halfway through a big data transfer, the session can be picked up from where it left off, rather than starting again

Networks

The OSI Model: Session Layer

For example, if the network crashes and reboots halfway through a big data transfer, the session can be picked up from where it left off, rather than starting again

You may already know that protocols like HTTP *don't* automatically pick up from where they left off

Networks

The OSI Model: Session Layer

For example, if the network crashes and reboots halfway through a big data transfer, the session can be picked up from where it left off, rather than starting again

You may already know that protocols like HTTP *don't* automatically pick up from where they left off

This tells us there is possibly a gap or omission somewhere in the relevant protocols: something they didn't address in the design

Networks

The OSI Model: Session Layer

For example, if the network crashes and reboots halfway through a big data transfer, the session can be picked up from where it left off, rather than starting again

You may already know that protocols like HTTP *don't* automatically pick up from where they left off

This tells us there is possibly a gap or omission somewhere in the relevant protocols: something they didn't address in the design

This may have been through deliberate choice; but it's equally likely they just didn't think about it

Networks

The OSI Model: Presentation Layer

The *presentation layer*, *layer 6* provides some things to help us retain the *meaning* of data

Networks

The OSI Model: Presentation Layer

The *presentation layer*, *layer 6* provides some things to help us retain the *meaning* of data

In particular, it decides on representations of data, such as characters, integers and floating point values, colours, sounds and so on so that the source and destination can agree on the data communicated

Networks

The OSI Model: Presentation Layer

So if the source wants to send the number 42, the presentation layer deals with encoding this in a suitable way as (say) some bits, which are then transmitted (passed to layer 5)

Networks

The OSI Model: Presentation Layer

So if the source wants to send the number 42, the presentation layer deals with encoding this in a suitable way as (say) some bits, which are then transmitted (passed to layer 5)

And the destination presentation layer can determine that this particular sequence of bits it has just received (from layer 4) represents the number 42

Networks

The OSI Model: Presentation Layer

So if the source wants to send the number 42, the presentation layer deals with encoding this in a suitable way as (say) some bits, which are then transmitted (passed to layer 5)

And the destination presentation layer can determine that this particular sequence of bits it has just received (from layer 4) represents the number 42

They can agree on “42” regardless of how each host chooses to represent integers internally

Networks

The OSI Model: Application Layer

The *application layer*, *layer 7*, is the layer application programmers use: ideally programmers would not have to worry about lower layers in their application

Networks

The OSI Model: Application Layer

The *application layer*, *layer 7*, is the layer application programmers use: ideally programmers would not have to worry about lower layers in their application

It contains protocols like HTTP for the Web, SMTP for email, and so on

Networks

The OSI Model: Application Layer

The *application layer*, *layer 7*, is the layer application programmers use: ideally programmers would not have to worry about lower layers in their application

It contains protocols like HTTP for the Web, SMTP for email, and so on

Built on top of these protocols are the applications that the users see, e.g., Firefox or Chrome for the Web, Outlook or Thunderbird for email

Networks

Layering Models

Conceptually, data from an application is passed down through the layers until it reaches the hardware: i.e., through a sequence of pieces of software that perform the functions of each layer

Networks

Layering Models

Conceptually, data from an application is passed down through the layers until it reaches the hardware: i.e., through a sequence of pieces of software that perform the functions of each layer

As it passes from later to layer it is *encapsulated*: a transformation of the data in such a way that the layer below can cope with it transparently

Networks

Layering Models

Conceptually, data from an application is passed down through the layers until it reaches the hardware: i.e., through a sequence of pieces of software that perform the functions of each layer

As it passes from later to layer it is *encapsulated*: a transformation of the data in such a way that the layer below can cope with it transparently

And in a way that it can be untransformed back again

Networks

Layering Models

At each layer, the transformation might

Networks

Layering Models

At each layer, the transformation might

- add an identifying header or trailer or both that is needed for the functionality of the layer

Networks

Layering Models

At each layer, the transformation might

- add an identifying header or trailer or both that is needed for the functionality of the layer
- encode any bit patterns that might be misinterpreted or mis-transmitted by the next layer

Networks

Layering Models

At each layer, the transformation might

- add an identifying header or trailer or both that is needed for the functionality of the layer
- encode any bit patterns that might be misinterpreted or mis-transmitted by the next layer
- put items in a standard form, e.g., integers into a well-known format

Networks

Layering Models

At each layer, the transformation might

- add an identifying header or trailer or both that is needed for the functionality of the layer
- encode any bit patterns that might be misinterpreted or mis-transmitted by the next layer
- put items in a standard form, e.g., integers into a well-known format
- do some arbitrarily complicated manipulation

Networks

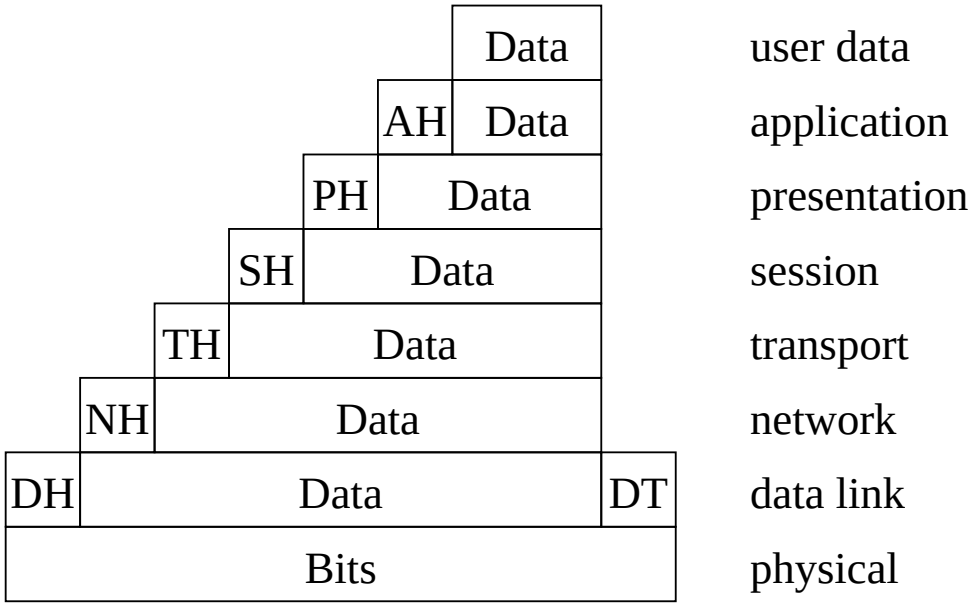
Layering Models

At each layer, the transformation might

- add an identifying header or trailer or both that is needed for the functionality of the layer
- encode any bit patterns that might be misinterpreted or mis-transmitted by the next layer
- put items in a standard form, e.g., integers into a well-known format
- do some arbitrarily complicated manipulation
- do nothing at all!

Networks

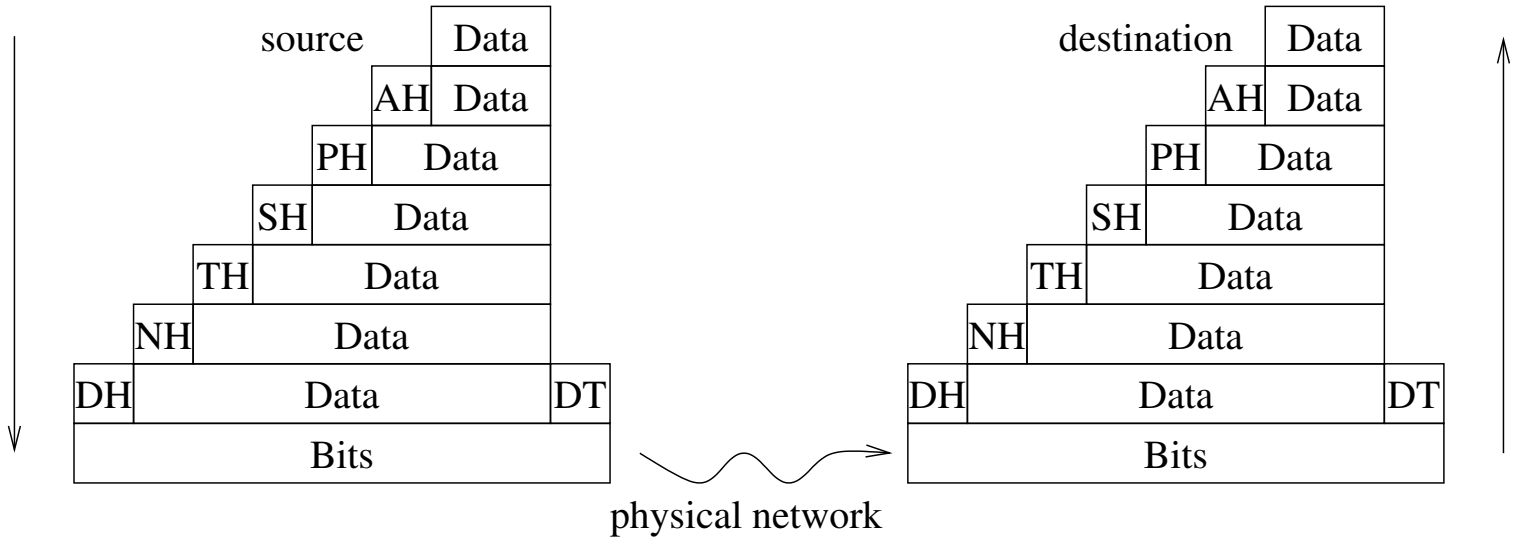
Layering Models



A possible (but unlikely) OSI encapsulation

Networks

Layering Models



Data is encoded and decoded

Networks

Encapsulation

An example. Some early modems treated byte values less than 32 as commands to the modem, not data to be transmitted

Networks

Encapsulation

An example. Some early modems treated byte values less than 32 as commands to the modem, not data to be transmitted

E.g., value 4 might mean “end of transmission” and the modem should drop the connection

Networks

Encapsulation

An example. Some early modems treated byte values less than 32 as commands to the modem, not data to be transmitted

E.g., value 4 might mean “end of transmission” and the modem should drop the connection

What do you do if your data happens to contain the value 4?

Networks

Encapsulation

An example. Some early modems treated byte values less than 32 as commands to the modem, not data to be transmitted

E.g., value 4 might mean “end of transmission” and the modem should drop the connection

What do you do if your data happens to contain the value 4?

You can’t just send it, as the modem would interpret the data as a command and end the connection

Networks

Encapsulation

So you need to transform the data somehow so that “4” is never seen by the modem in the datastream

Networks

Encapsulation

So you need to transform the data somehow so that “4” is never seen by the modem in the datastream

And the transformation must be reversible, so the other end can reconstruct the 4

Networks

Encapsulation

So you need to transform the data somehow so that “4” is never seen by the modem in the datastream

And the transformation must be reversible, so the other end can reconstruct the 4

This is why encapsulation is necessary: so data can be transmitted accurately, even if you are using weird hardware

Networks

Byte Stuffing

In this case, the transformation often used was *byte stuffing*: the link layer could replace byte value “04” by, say, a pair of bytes “DB D4” (hexadecimal)

Networks

Byte Stuffing

In this case, the transformation often used was *byte stuffing*: the link layer could replace byte value “04” by, say, a pair of bytes “DB D4” (hexadecimal)

Both bytes will be transmitted unmolested by the modem

Networks

Byte Stuffing

In this case, the transformation often used was *byte stuffing*: the link layer could replace byte value “04” by, say, a pair of bytes “DB D4” (hexadecimal)

Both bytes will be transmitted unmolested by the modem

The link layer at the other end could recognise this pair and replace it by the single byte “04”

Networks

Byte Stuffing

In this case, the transformation often used was *byte stuffing*: the link layer could replace byte value “04” by, say, a pair of bytes “DB D4” (hexadecimal)

Both bytes will be transmitted unmolested by the modem

The link layer at the other end could recognise this pair and replace it by the single byte “04”

The “DB” is called an *escape character*, and its presence in the datastream means the next character is encoded, so special action must be taken

Networks

Byte Stuffing

Take a while to think of the issues this raises: what happens if our original data contained the pair of values “DB D4”?

Networks

Byte Stuffing

Take a while to think of the issues this raises: what happens if our original data contained the pair of values “DB D4”?

We can’t just send “DB D4” as the other end will replace them by “04”

Networks

Byte Stuffing

Take a while to think of the issues this raises: what happens if our original data contained the pair of values “DB D4”?

We can’t just send “DB D4” as the other end will replace them by “04”

Not only do the bytes under 32 need to be stuffed, so does the escape character

Networks

Byte Stuffing

Take a while to think of the issues this raises: what happens if our original data contained the pair of values “DB D4”?

We can’t just send “DB D4” as the other end will replace them by “04”

Not only do the bytes under 32 need to be stuffed, so does the escape character

For example, “DB” in the original data could be stuffed as “DB FF”

Networks

Byte Stuffing

Take a while to think of the issues this raises: what happens if our original data contained the pair of values “DB D4”?

We can’t just send “DB D4” as the other end will replace them by “04”

Not only do the bytes under 32 need to be stuffed, so does the escape character

For example, “DB” in the original data could be stuffed as “DB FF”

The datastream “DB D4” becomes “DB FF D4”

Networks

Byte Stuffing

Take a while to think of the issues this raises: what happens if our original data contained the pair of values “DB D4”?

We can’t just send “DB D4” as the other end will replace them by “04”

Not only do the bytes under 32 need to be stuffed, so does the escape character

For example, “DB” in the original data could be stuffed as “DB FF”

The datastream “DB D4” becomes “DB FF D4”

With byte stuffing, we exchange some expansion of the data for the correct transmission of that data

Networks

This kind of situation is why encapsulation exists

Networks

This kind of situation is why encapsulation exists

Of course, modern hardware doesn't act like early modems, but the principle remains

Networks

Layering Models

Say you want to send an email. In a strict implementation adhering to the layers the following might happen

Networks

Layering Models

Say you want to send an email. In a strict implementation adhering to the layers the following might happen

- The email application might add a standard email header (`From`, `To`, etc.)

Networks

Layering Models

Say you want to send an email. In a strict implementation adhering to the layers the following might happen

- The email application might add a standard email header (`From`, `To`, etc.)
- This is passed to the presentation layer. As far as this layer is concerned it gets a chunk of text from the application layer

Networks

Layering Models

Say you want to send an email. In a strict implementation adhering to the layers the following might happen

- The email application might add a standard email header (`From`, `To`, etc.)
- This is passed to the presentation layer. As far as this layer is concerned it gets a chunk of text from the application layer
- It doesn't (or shouldn't) know that the first few characters are an email header

Networks

Layering Models

Say you want to send an email. In a strict implementation adhering to the layers the following might happen

- The email application might add a standard email header (`From`, `To`, etc.)
- This is passed to the presentation layer. As far as this layer is concerned it gets a chunk of text from the application layer
- It doesn't (or shouldn't) know that the first few characters are an email header
- It may transform the characters in some way, e.g., converting video into a transmissible format; it might prepend its own header to indicate what it has done

Networks

Layering Models

- This is passed to the session layer. As far as this layer is concerned it gets a bunch of bits from the previous layer

Networks

Layering Models

- This is passed to the session layer. As far as this layer is concerned it gets a bunch of bits from the previous layer
- It doesn't (or shouldn't) know that the first few bits are a layer header

Networks

Layering Models

- This is passed to the session layer. As far as this layer is concerned it gets a bunch of bits from the previous layer
- It doesn't (or shouldn't) know that the first few bits are a layer header
- It may transform the bits in some way; it might prepend a header to help it manage sessions

Networks

Layering Models

- This is passed to the session layer. As far as this layer is concerned it gets a bunch of bits from the previous layer
- It doesn't (or shouldn't) know that the first few bits are a layer header
- It may transform the bits in some way; it might prepend a header to help it manage sessions
- And so on down through the layers

Networks

Layering Models

- This is passed to the session layer. As far as this layer is concerned it gets a bunch of bits from the previous layer
- It doesn't (or shouldn't) know that the first few bits are a layer header
- It may transform the bits in some way; it might prepend a header to help it manage sessions
- And so on down through the layers

Eventually, the physical layer transmits some bits

Networks

Layering Models

At the destination a bunch of bits is received by the hardware

Networks

Layering Models

At the destination a bunch of bits is received by the hardware

We now proceed up the layers, unwrapping and untransforming as we go

Networks

Layering Models

At the destination a bunch of bits is received by the hardware

We now proceed up the layers, unwrapping and untransforming as we go

And, eventually, we get the original data arriving at the application (we hope)

Networks

Layering Models

At the destination a bunch of bits is received by the hardware

We now proceed up the layers, unwrapping and untransforming as we go

And, eventually, we get the original data arriving at the application (we hope)

So why do this as it seems so wasteful?

Networks

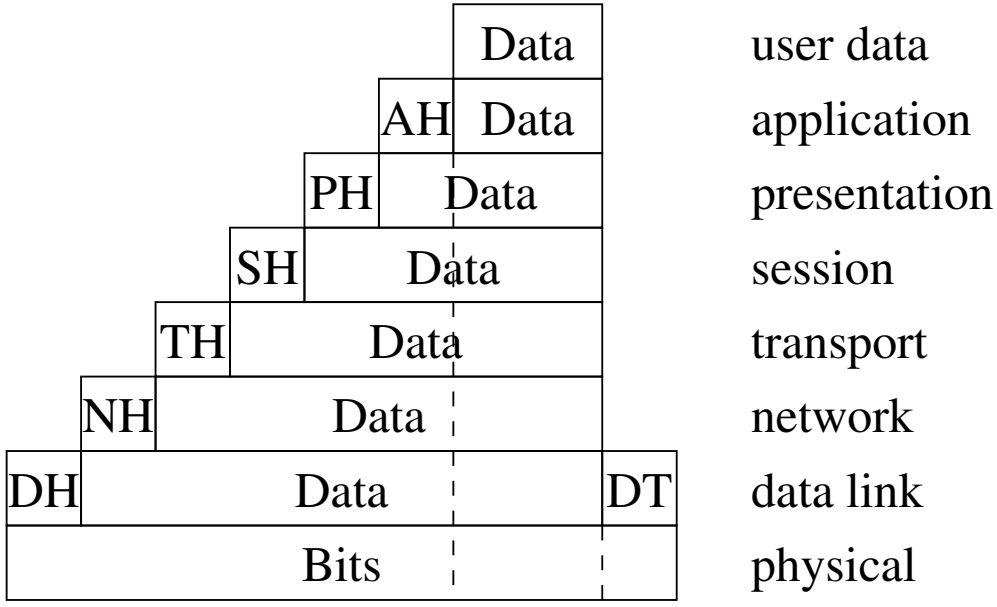
Layering Models

If the original data are small the data transmitted on the wire can be mostly headers from the various layers

Networks

Layering Models

If the original data are small the data transmitted on the wire can be mostly headers from the various layers



Encapsulation overhead

Networks

Layering Models

Surely it is easier just to put the original data on the wire?

Networks

Layering Models

Surely it is easier just to put the original data on the wire?

- Encapsulation adds complexity to the implementation

Networks

Layering Models

Surely it is easier just to put the original data on the wire?

- Encapsulation adds complexity to the implementation
- It adds overhead (both space and time)

Networks

Layering Models

Surely it is easier just to put the original data on the wire?

- Encapsulation adds complexity to the implementation
- It adds overhead (both space and time)
- thereby reducing effective throughput

Networks

Layering Models

Surely it is easier just to put the original data on the wire?

- Encapsulation adds complexity to the implementation
- It adds overhead (both space and time)
- thereby reducing effective throughput

But it turns out layering and encapsulation actually *reduces* overall complexity, just like breaking a large program into functions/objects/whatever does for programming

Networks

Layering Models

Surely it is easier just to put the original data on the wire?

- Encapsulation adds complexity to the implementation
- It adds overhead (both space and time)
- thereby reducing effective throughput

But it turns out layering and encapsulation actually *reduces* overall complexity, just like breaking a large program into functions/objects/whatever does for programming

It also gives flexiblity

Networks

Layering Models

Suppose we have a 1Gb network card in our machine and someone comes along with a 10Gb card

Networks

Layering Models

Suppose we have a 1Gb network card in our machine and someone comes along with a 10Gb card

Because the physical layer is (mostly) separate from the data link layer we can just write a new standard for the 10Gb physical layer and slot it in where the old 1Gb standard was

Networks

Layering Models

Suppose we have a 1Gb network card in our machine and someone comes along with a 10Gb card

Because the physical layer is (mostly) separate from the data link layer we can just write a new standard for the 10Gb physical layer and slot it in where the old 1Gb standard was

The upper layers needn't know anything has changed

Networks

Layering Models

Suppose we have a 1Gb network card in our machine and someone comes along with a 10Gb card

Because the physical layer is (mostly) separate from the data link layer we can just write a new standard for the 10Gb physical layer and slot it in where the old 1Gb standard was

The upper layers needn't know anything has changed

And we can slot in the implementation for the new hardware in exactly the same way

Networks

Layering Models

Suppose we have a 1Gb network card in our machine and someone comes along with a 10Gb card

Because the physical layer is (mostly) separate from the data link layer we can just write a new standard for the 10Gb physical layer and slot it in where the old 1Gb standard was

The upper layers needn't know anything has changed

And we can slot in the implementation for the new hardware in exactly the same way

We don't have to rewrite our email application (and Web browser, and all our other applications) because of the upgrade

Networks

Layering Models

Similarly for all the other layers: we can replace specifications in a layer and implementations of those specifications without affecting the rest of the stack

Networks

Layering Models

Similarly for all the other layers: we can replace specifications in a layer and implementations of those specifications without affecting the rest of the stack

In principle, you could use carrier pigeons as the physical layer and your browser should work unchanged

Networks

Layering Models

Similarly for all the other layers: we can replace specifications in a layer and implementations of those specifications without affecting the rest of the stack

In principle, you could use carrier pigeons as the physical layer and your browser should work unchanged

Apart, perhaps, from speed

Networks

Layering Models

Similarly for all the other layers: we can replace specifications in a layer and implementations of those specifications without affecting the rest of the stack

In principle, you could use carrier pigeons as the physical layer and your browser should work unchanged

Apart, perhaps, from speed

Someone really did this once!

Networks

Layering Models

Similarly for all the other layers: we can replace specifications in a layer and implementations of those specifications without affecting the rest of the stack

In principle, you could use carrier pigeons as the physical layer and your browser should work unchanged

Apart, perhaps, from speed

Someone really did this once!

Exercise Read RFC1149

Networks

Layering Models

And as each layer simply hands over to the next, it doesn't actually matter what the next layer "really" does

Networks

Layering Models

And as each layer simply hands over to the next, it doesn't actually matter what the next layer "really" does

As long as it has the right behaviour, it doesn't matter how it is actually implemented

Networks

Layering Models

And as each layer simply hands over to the next, it doesn't actually matter what the next layer "really" does

As long as it has the right behaviour, it doesn't matter how it is actually implemented

This enables useful tricks like *tunnelling*, which we shall look at later