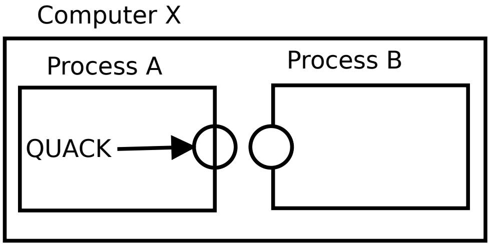
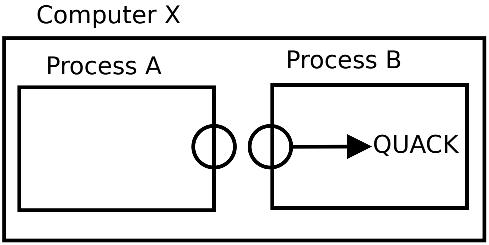
CSSE2310 — 9.2

Networks — Ogres

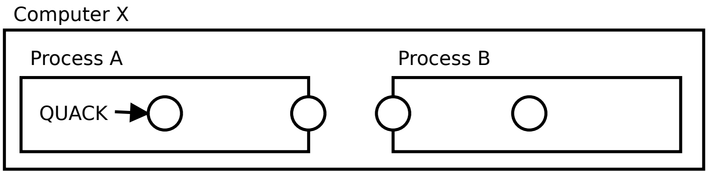
Where are we going?



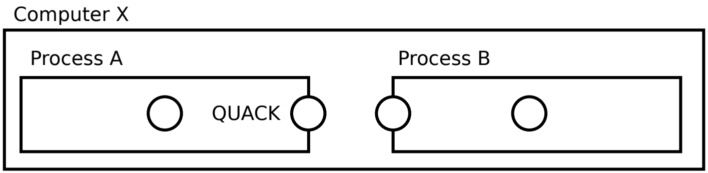
Where are we going?



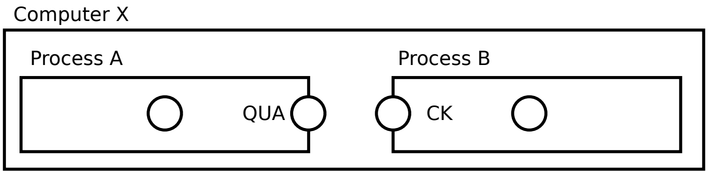
Where are we going?



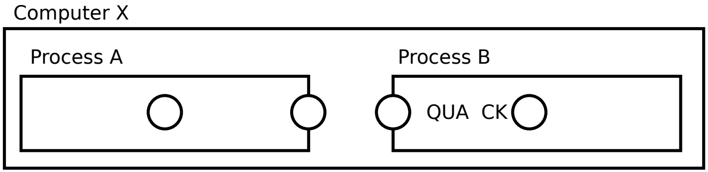
Where are we going?



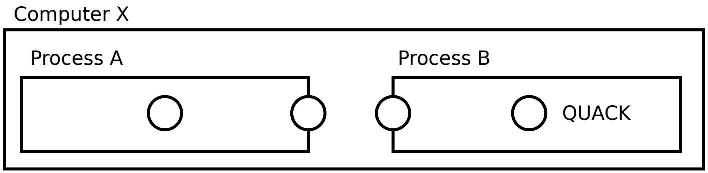
Where are we going?



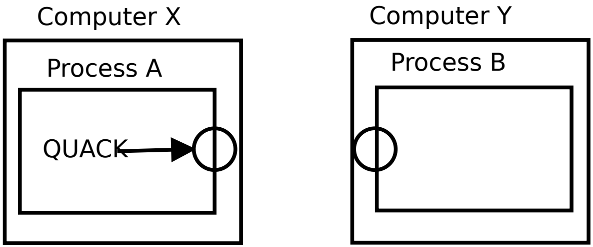
Where are we going?



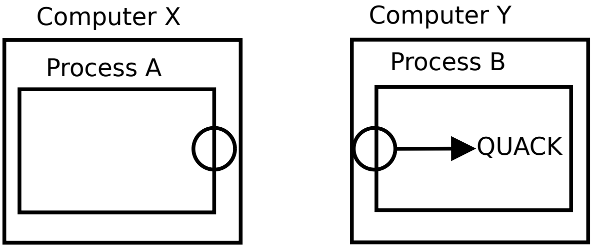
Where are we going?



Where are we going?



Where are we going?



Software layers

Rather than developing software in one lump, it can be easier (although not always more efficient) to write in layers.

- Make each part of the problem more managable.
- Add capabilities or modes of operation the lower level doesn't allow.
- Hide procedural aspects.
- Make one system act like another system.
- Allow lower levels to be replaced without disrupting the whole system.

A connected set of layers is a “stack”.

C I/O

Standard C I/O is via “streams” (FILE*).

- act as an unstructured sequence of bytes.
- can just keep calling `fgetc()`
  - You might need to wait longer sometimes
- Can ask for numbers or “words” from the stream. (`fscanf()`)

Streams

But we know that bytes from outside need to come from the kernel.

- The kernel doesn't provide that interface.
- Only `read()`/`write()` fixed sized arrays.
  - Which might not process the amount you asked for.
  - Which might be interrupted by signals.
- C standard I/O simulates streams using a hidden buffer and lower level calls.

Devices

Behind the scenes in the kernel. What if the underlying device:

- takes a byte at a time?
- a fixed size block (and nothing smaller) [disk?]
- variable sized chunks [networks?]

Substitution?

Another system suports C standard I/O.
Does it support `read()`?

- Does it matter?

Simple communication stack

1. Communicate with exactly one other entity.
2. Choose which other peer to send a message to get closer to destination.
3. Break a message into parts for commication and reassemble them.

Headers/envelopes

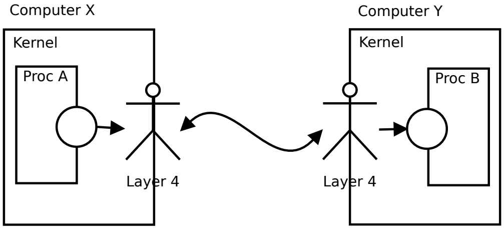
In order to work with lower layers, the "message" may need:

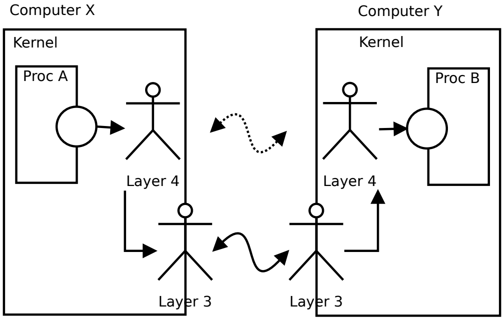
- to be encoded (eg bytes sent via an optical network $\rightarrow$ light pulses)
- Have extra information added
  - Headers (who is message to? from? ...)
  - Footers (ethernet uses both)

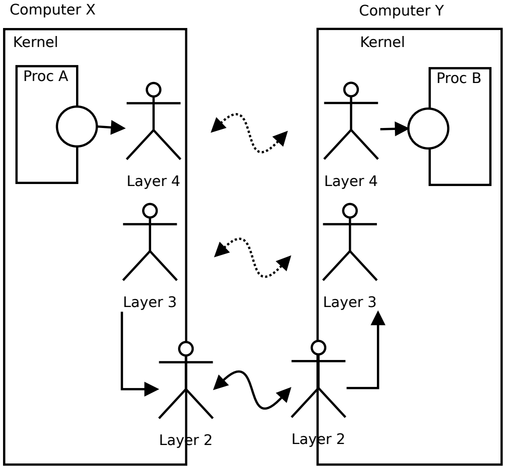
That extra information

- will (usually) be removed at the other end.
- might be the only part of the message that level understands.
  - So we use an envelope as an analogy (you can't see into the envelope)

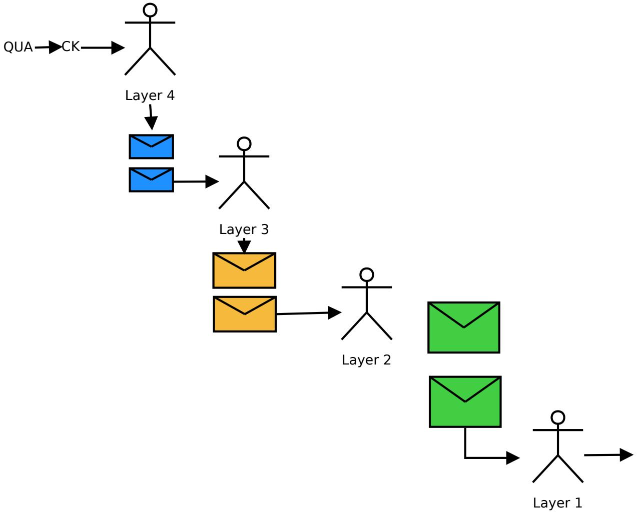
Layers — illustrations

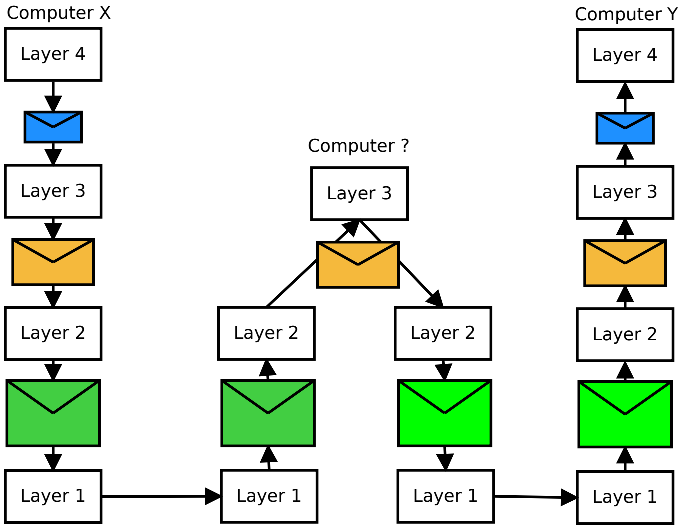

Computer X
Kernel
Proc A
Layer 4
Computer Y
Kernel
Proc B
Layer 4


Computer X
Kernel
Proc A
Layer 4
Layer 3
Computer Y
Kernel
Proc B
Layer 4
Layer 3

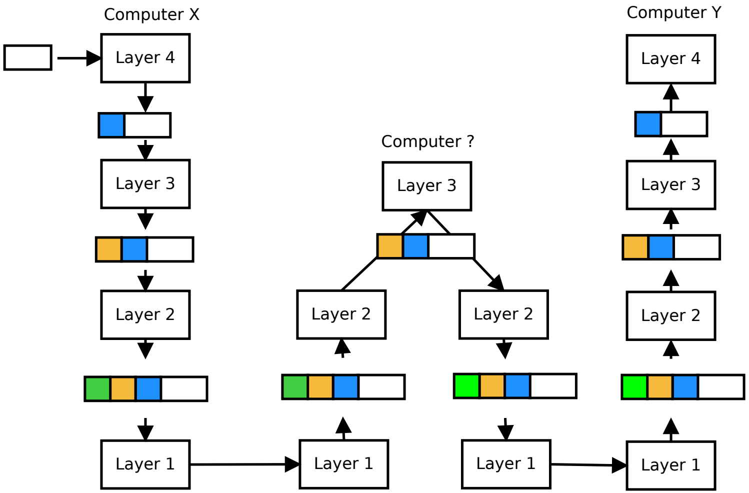

Computer X
Computer Y
Kernel
Kernel
Proc A
Proc B
Layer 4
Layer 4
Layer 3
Layer 3
Layer 2
Layer 2

View 1 — envelopes


QUA
CK
Layer 4
Layer 3
Layer 2
Layer 1


Computer X
Layer 4
Layer 3
Layer 2
Layer 1
Computer ?
Layer 3
Layer 2
Layer 1
Layer 2
Layer 1
Computer Y
Layer 4
Layer 3
Layer 2
Layer 1

View 2 — headers


Computer X
Layer 4
Layer 3
Layer 2
Layer 1
Computer ?
Layer 3
Layer 2
Layer 2
Layer 1
Layer 1
Computer Y
Layer 4
Layer 3
Layer 2
Layer 1