COSC441

Message Passing

Outline

- Conceptual model
- Rapid survey of message passing for POSIX
- Issues

Conceptual model

- Imagine a car
- You don't want a failure in the engine control system to disrupt the brakes
- You don't want a failure in the entertainment system to disrupt the engine control
- Put these on physically separate machines
- Communicating through wires
- Using small secure protocols

UNIX has...

- System V message queues
- POSIX message queues
- Pipes
- Named pipes
- Unix domain sockets
- TCP/IP sockets
- (possibly MPI)

System V

- msgget : create/open queue
- msgctl : query/change attributes, close
- msgsnd : send a message

  has type, size, data, wait/nowait
- msgrcv : receive a message

  has type wanted/received, size, data, wait/nowait

POSIX

- mq_open
- mq_close
- mq_send (data, size, priority)
- mq_timedsend (data, size, priority, timeout)
- mq_receive (data, size, priority got)
- mq_timedreceive (data, size, priority got, timeout)
- mq_getattr, mq_setattr

Pipes and named pipes

- Should already be familiar
- Like message queues but no message boundaries, types, or priorities

Issues: Naming

- How does a thread/process *find* a communication channel it wants to use?
- How can you *name* one you are creating?
- Are they even named at all (pipes weren't, but what about using /proc/${pid}/fd/${num}?
- Can you send a reference to a channel through a channel?

Issues: I/O rates

- latency: time between send and receive
- bandwidth: how many bytes or messages per second can be sent?
- capacity: how many bytes or messages can be buffered before the sender blocks?

Issues: flow control

- If the sender sends too fast, the receiver needs to be able to slow it down
- If the sender sends too slowly, the receive might need to give it a nudge
- Messages can be lost if they arrive too fast
- If messages arrive too slowly, timeouts and disconnections may occur.
- Flow control done by OS for message queues, pipes, UNIX domain sockets, by network stack for TCP
- Not done by system if you use UDP

Issues: encoding

- How are data encoded?
  - binary
  - ASN.1
  - XDR
  - *ad hoc* text
  - XML
  - JSON
  - S-expressions
  - UBF(A)

Issues: copying

- Putting a value on the wire involves *copying* it (perhaps in another form): marshalling, pickling, serialisation
- Receiving a value from the wire involves *copying* it (perhaps from another form): unmarshalling, unpickling, deserialisation.
- This is a cost that shared memory systems do not pay.
- Object identity may be lost
- Unless you use the PROXY pattern

Issues: monitoring and debugging

- Message capture is a great non-invasive tool
  - for record/replay debugging
  - for record/replay resilience
  - for performance monitoring
  - for checking conformance to protocols/contracts
  - for visualising system behaviour

Issues: system partitioning

- Processes that communicate by sending messages do not care whether they are on the same core, on the same chip, in the same box, or on a network
- *PROVIDED THAT NAMING WORKS.*
- Unix message queues and pipes do not work between hosts. TCP does.
- Relays (gateways) can bridge such gaps.

Issues: selective receive

- Can the sender prioritise messages?
- Can the receiver choose which messages to process first?
- Ada, Occam, Concurrent ML, Erlang all have some form of selective receive. It is a wonderful programming tool.

Outline

- Conceptual model
- Rapid survey of message passing for POSIX
- Issues