Chapter 2

Design Issues

Part 2.1

Naming

- . . . The Internet is running out of IP addresses
  - hence IP6 with 32-bit addresses
- . . . The Arpanet RAN out of hostnames in 1980
- . . . The DDD voice net is running out of phone addresses, so we need to use more area codes

The Lesson

- we need a name space which is
  - global
  - location-independent

Name-space lesson . . .

- expansible (indefinitely?)
- translatable to addresses (where) and routes
  - (how to get there from here)
- translatable efficiently to addresses

Communication

- **subnet plus**
  - **OS kernel for basic message passes plus**
    - **protocol stacks**

Communication

- need a high level programming model
  - (eg RPC, object-sends) BUT
- machinery ought to be modular BUT
- efficiency ought to be better than a few percent

AND

Communication

- all the rules change with fiber and radio !

THE LESSON

- We need a name space which is
  - global
  - location-independent
  - expansible (indefinitely?)
  - translatable to addresses (where) and routes (how to get there from here)
  - translatable efficiently to addresses

Fiber

- datarates of 10^9 bps approximate bus speeds
- data-in-flight (datarate*latency) is HUGE
- BER very small (negligible)

Radio (wireless)

- more or less the opposite, sigh.

Software Structure Paradigms

- data abstraction

Data abstraction

- processes as mothers who hide data
  (or hardware)
- processes which provide well-defined interfaces (methods!)
- processes who we invoke by messages

- add inheritance, and

- Voila! - - -
- Active Objects!

Active Objects

- PLUSSES:
- achieves mutual exclusion easily and naturally
- easily distributed especially if each object lives

  in its own virtual address space

  - minus:
- $$$

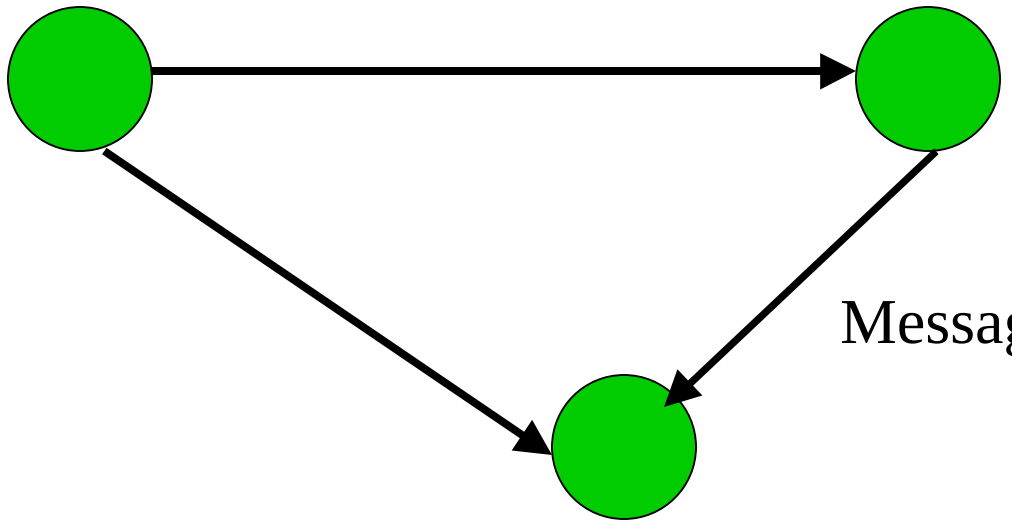
Active Objects require

- DOS to support objects

Active objects suggest

- DOS made of objects (Apertos)

the process-message graph abstraction



- works with active objects too