Characterizing Distributed Systems

- Def'n (CDK):
- A distsys is:
  - a set of autonomous computers
  - linked by a network

- with software designed to produce an integrated computing facility.

Criticisms:

- Absent: Enslow's ideas of
  - no shared memory
  - no shared clock, thus
    - time intedeterminacy owing to unpredictable (unbounded?) network delays

Criticisms:

- LeLann's (?) idea that we can *never* know the entire current state.
  - entire old state is knowable, OR
  - partial current state is knowable,
    but that's all. (cf. *Heisenberg Uncertainty Principle*)

Concepts mentioned only implicitly:

- multiple threads of control (Enslow)
- multiple resources, dynamically assignable (Enslow)

Concepts mentioned only implicitly:

- NO master-slave stuff! (Enslow)

  (unless we can elect a new master)

- transparency (Enslow)

Mentioned explicitly:

- high-level control (Enslow)
- co-operative autonomy (Enslow)

Key Characteristics:

- (more accurately:
  - good things which may be (?) more easily obtained in a dist than in a centralized world

Good things

- resource sharing:
  - amen!
- openness:
  - good for you, but not essential to ddp
- concurrency;
  - almost inevitable, unless we work at it (coroutines)

Good things

- scalability:
  - highly desirable, but not essential to a distsys

Good things

- fault tolerance:
  - arises naturally
    - (Lelann's observation that the limit case of delayed response is failed responder)

  and good for you, but not essential to a distsys (?)

Good things

- transparency:
  - amen once more.
  - Simplifies programming enormously and
  - facilitates scalability, fault tolerance , . . .

Examples of distsys:

- workstations & servers, connected by a LAN
    - (unix net)

Examples

- ATM network: ATMs, ATM mothers, account database machine & hot standby of same.
  - security,
  - reliability &
  - scalability   considered important

Resource Managers -> Objects

- **What's a resource manager?**
  just a process which mothers a *resource*
- **What’s a resource?**
  - hardware resource e.g. printer, or
  - data resource eg a flag or semaphore or file ;
- anything to which you might want to control access

Controlling access - “mothering”

- you access the resource by communicating with its mothering process
- the mother defines permitted operations
  - ("set the flag",
  - "reset the flag",
  - "test the flag state")

Mapping mother processes into objects

- the mother defines permitted operations
  - which may be called *methods*.
- mother is an instantiation of a definition
  - of code & data structures
  - which may be called a *class* IF carefully defined
- Hence the mothering process becomes a kind
  - of object. (but not yet with inheritance or scope).

My bias:

- objects are specialized processes
  - (active; independent threads of control)
  - not specialized data structures (passive, no threads
  - of control)
- For our next trick . . .

Resource Managers become Servers

- Story so far:
- Resource Mother is
  - just a process which mothers a resource
    - (hardware e.g. printer, data eg a flag or semaphore)
  - you access the resource by communicating with its mothering process