# Shared Page Model

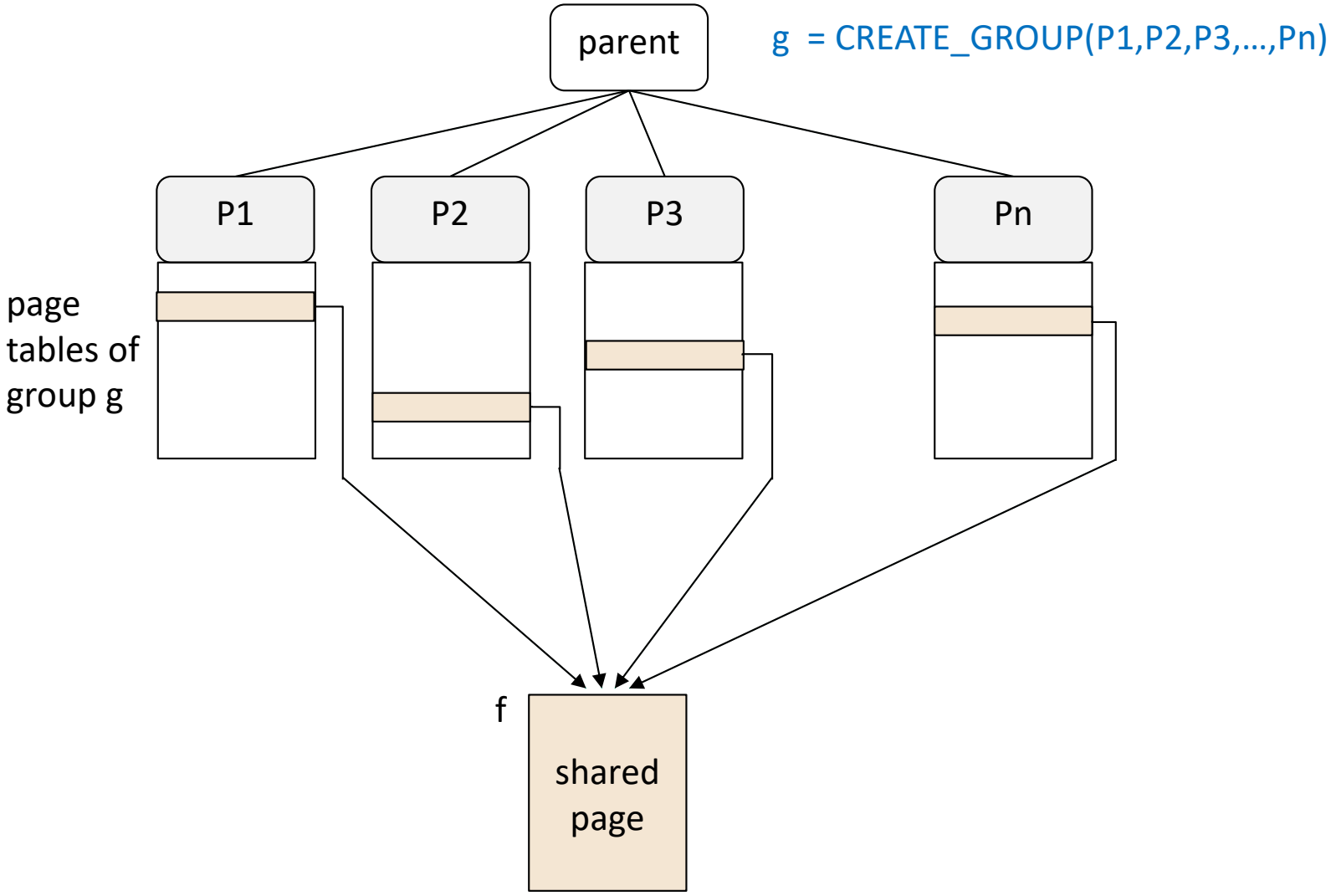
Peter J. Denning

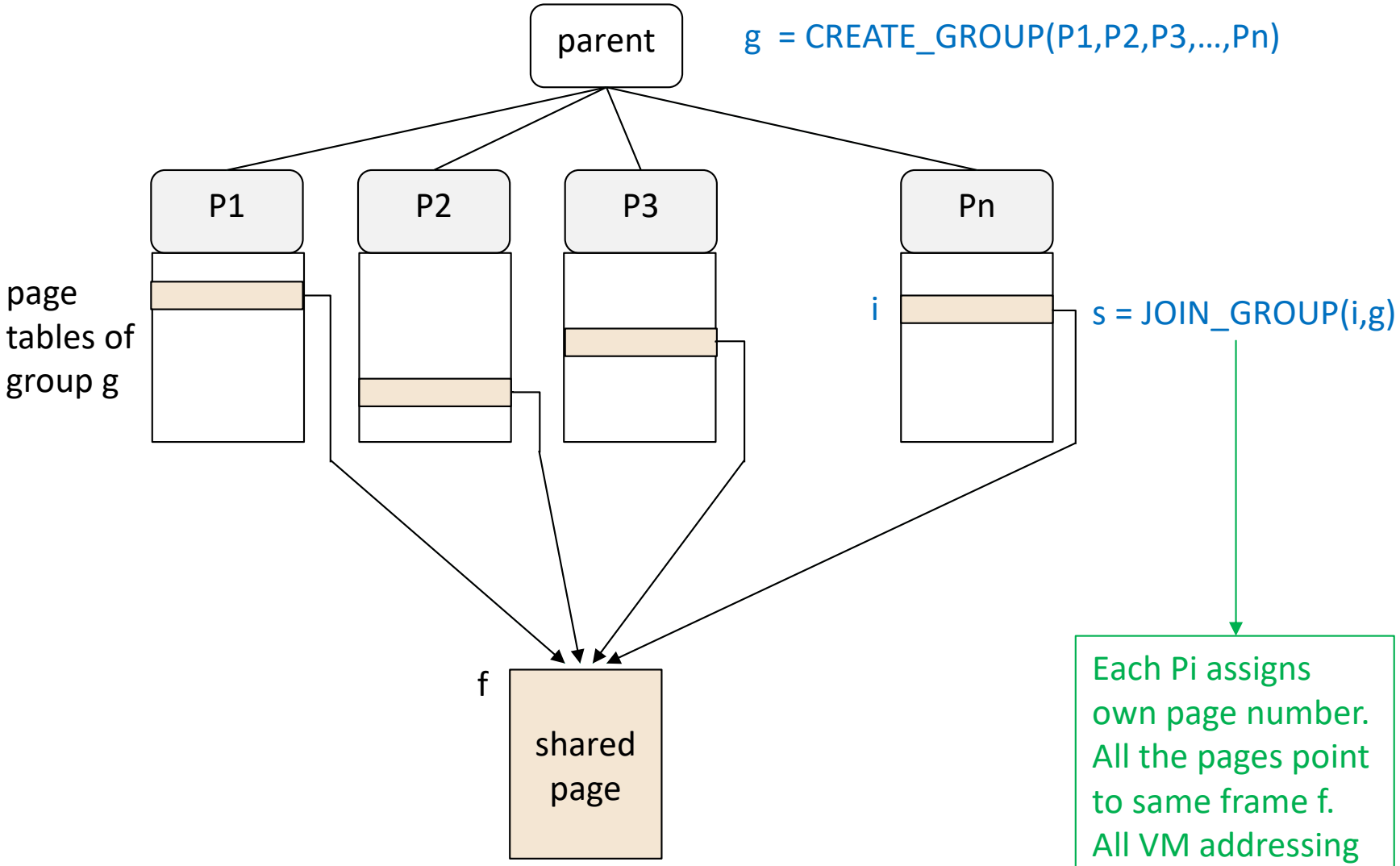
# Address space overlap

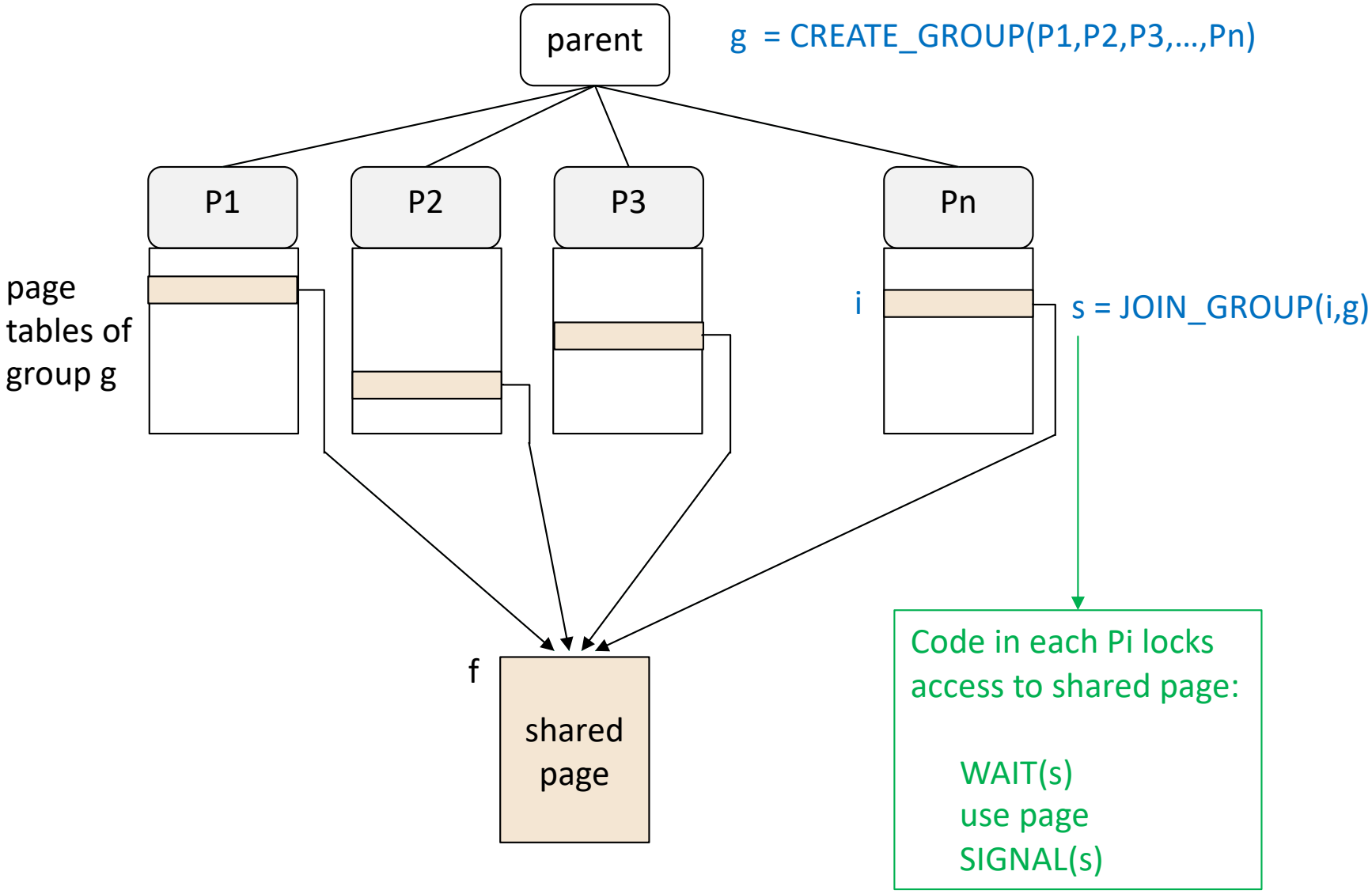
- Default isolation: processes have their own private address spaces
  - Impossible for one to access pages of another
- Easy channel between address spaces: share a page frame
- More clumsy and difficult than it sounds

# Specifications

- Process group (2 or more)
  - Typically set up by a parent of the processes in the group
- One shared page (frame) associated with the group
- Each process assigns a page to point to shared frame; need not be the same page in each process







# Undoing

- EXIT_GROUP(i,g)
  - By a process
  - Removes PT entry I
  - If last to exit, delete semaphore
- DISSOLVE_GROUP(g)
  - By creator
  - Only if all members have exited

# Undoing

Lots of questions if events are not “clean”, for example, a process quits without exiting the group.

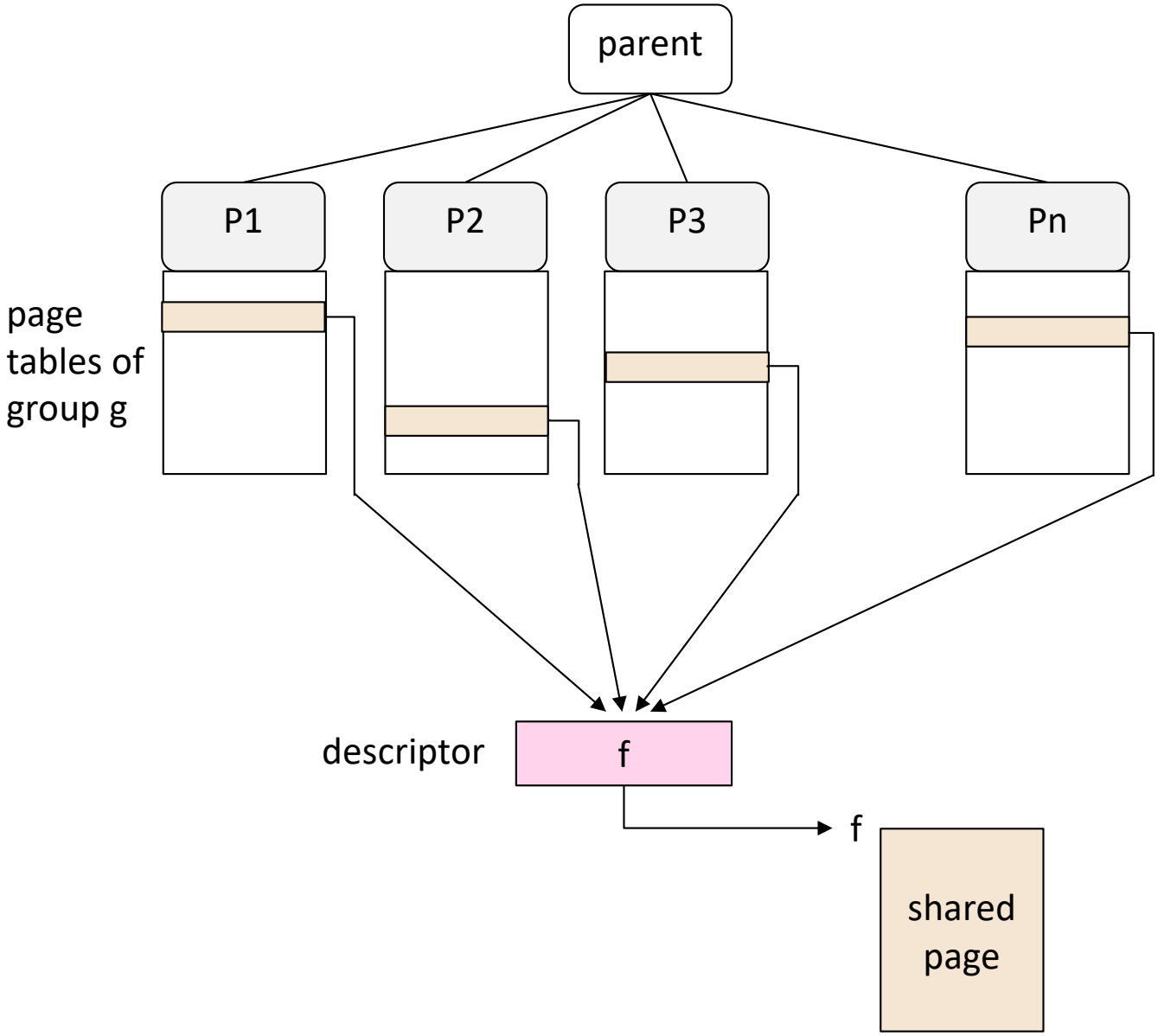
- EXIT_GROUP(i,g)
  - By a process
  - Removes PT entry I
  - If last to exit, delete semaphore
- DISSOLVE_GROUP(g)
  - By creator
  - Only if all members have exited

# Relocation problem

- What if OS decides to use a different frame for shared page?
  - Page not in use by any member of group: paged out. Reclaimed into different frame by a page fault from process in the group.
  - Entire group is suspended by the parent, resumed later.
- Relocation slow and expensive
  - Must maintain list of page numbers used by processes for their shared pages, so that their frame fields can be updated when the shared frame changes
  - Must suspend all processes in the group during relocation update

# Relocation problem - 2

- Could theoretically be solved by a level of indirection
  - Replace the shared page with a descriptor pointing to the shared frame
  - Update is fast because only the descriptor is updated
  - Must add bit to PT entries indicating that frame field points to descriptor not frame
  - Must modify the MMU hardware for additional level of indirection on some frame accesses
- Not attractive



# Overflow problem

- What if process attempts to write (or read) more bytes than the page size?
  - E.g., process requests write 600 bytes to 512-byte shared page
  - What happens to the 88 excess bytes?
- Virtual address (i,x) – page i line x
  - Next sequential is (i,x+1) if x<511 or (i+1,0) if x=511
  - Overflow bytes written into (or read from) next address space page, which is not shared

# Overflow problem - 2

- In some systems, the entire transfer is handed to DMA hardware, which then overflows into next sequential frame, which is not shared (a memory leak)
- No good solution

# What then?

- Conclusion: Shared page model not a good solution for interprocess communication
- Instead: Use simple message system to send messages between processes
  - Internet or RPC or both
- Messages can contain pointers to shared digital objects
  - Needs capability addressing – next OS level