



- J. K. Ousterhout, D. A. Scelza, and P. S. Sindhu, [Medusa: An Experiment in Distributed Operating Systems Structure](#), Communications of the ACM, Vol. 23, No. 2, February 1980, pp. 92-105.

*Q: What are the three distributed OS structures outlined in the paper, which structure does Medusa use, and why?*

- D. D. Redell, Y. K. Dalal, T. R. Horsley, H. C. Lauer, W. C. Lynch, P. R. McJones, H. G. Murray, and S. C. Purcell, [Pilot: An Operating System for a Personal Computer](#), Communications of the ACM, Vol. 23, No. 2, February 1980, pp. 81-92.

*Q: How do the requirements of the Pilot operating system differ from the systems we have read about so far, and how does the design of Pilot reflect those differences?*

## Preamble

- Design vs. implementation
  - systems papers often present the design and implementation of system
  - it is frequently the case that not all of the design is implemented
    - "vapor research"?
    - it is much better to present a coherent design than just what is implemented
- Performance evaluation
  - it is difficult to compare systems across hardware
  - can compare on the same hardware, but without applications it is difficult to understand importance
  - can evaluate system micro-benchmarks to ensure that primitive perform reasonably (Medusa does this)

CSE221 - Operating Systems  
Yuanxuan

10/12/21

Main  
Point

The component is distributed.

OS itself more others are replicated.

Approach

① central node handle

1) bottleneck

2) SPF

② replicate OS

1) No Enough memory.

③ Components on different CPU/Node.

Terms

Utility?

Activities

Pipes

### Terminology

- What is a utility?
  - a single OS module, abstraction
  - communication via messages
  - distributed among Cm nodes for parallelism
  - multiple instances for load balancing, reliability
- What are activities?
  - "process" on a given node for a given utility
- What are pipes?
  - communication channels among activities

Descriptors : kernel managed objects

### Descriptors

- What are all these descriptors for?
  - think of them as references to kernel-managed objects
- What are UDL descriptors?
  - utility "entry points", think of them as system calls
  - they provide a level of indirection to help with reliability (more below)
- What are PDL descriptors?
  - reference private pages, pipes
- What are SDL descriptors?
  - reference shared pages
- What are XDL descriptors?
  - "external" descriptors
  - for mapping a local descriptor onto a remote PDL/SDL descriptor
  - level of indirection
  - local references to XDL actually refer to what remote PDL/SDL descriptors
  - supports "unsealing"

CSE221 - Operating Systems

Unsealing

### Unsealing

- What is unsealing? What does it have to do with reliability?
  - this point is rather subtle and the paper does not draw much attention to it, but it's key to reliability for the system
  - we have multiple activities for the same utility for reliability.
- Example:
  - let's use the file system, the idea is that an application will be using one file system activity at a time.
  - let's say that FS activity crashes.
  - Medusa will then automatically change the UDLs the application was using for FS activity X and now have them refer to FS activity Y so that the application can continue to use the file system utility without having to restart.
  - Now we usually refer to it as "fail-over".

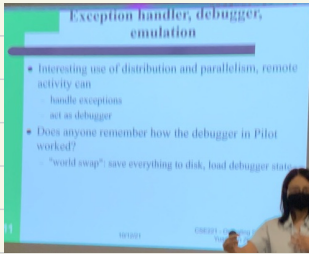
7 10/12/21 CSE221 - Operating Systems, Yuanyuan Zhou

### Unsealing (2)

- How to support fail-over?
  - A file system needs to maintain a "file object" keeping track of the state of an open file. in a standard OS, the file object is stored in the kernel's address space. So should we keep the file object in the activity's address space?
- Problem: To fail over when an activity fails, we need to move the file object from activity X to activity Y?
- Solution: store it in the application's address space.
  - Kernel objects stored in the application's address space are sealed to the application; it cannot directly manipulate it.
  - When an activity in a utility needs to manipulate the kernel object, the object is remote, so it needs to map it as an external object (XDL) and thereby unseal it; then manipulate it; then seal it again (remove from its XDL).

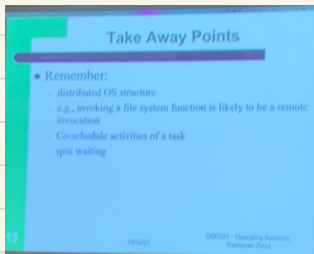
8

## Exception Debugger



## Spin-wait Comm

Take a chance that other thread will return faster than Ctx



co-schedule: (aka gang schedule)

just like Tele-meeting  
vs  
email back and forth

??

CPU schedule all together?

for uni-processor

Don't! No one will release.

Android - Don't spin  
taking too much battery

# Pilot

## Main Goal

- What is the main goal of Pilot?
  - design issues for a personal computer operating system
- How is the personal computer environment different from a time-sharing environment, and how is the difference reflected in the Pilot features?
  - single-user
  - resources do not have to be shared among users
  - "full exploitation of a resource rich environment"
  - processes cooperative more than competitive
  - defensive more than absolute protection
  - operating system != big brother
  - fairness not as much of a goal

## Breakout Discussion

- Group of 4-5 students.
- What were the design decisions of Pilot based on the fact that it was for a PC?
- If you are asked to design a new PC OS now, what do you want to change from Pilot?
- Consider
  - protection, language support, UI/UX, scheduling, resource sharing, files, storage, etc

4 10/12/21 CSE221 - Operating Systems, Yueshan Zhou

- Bad Design
- ① Bind to Mesa
  - ② Ignore Security
  - ③ Single memory space
  - ④ Virtual memory mapping file system

## Design

## Design

- What are the interesting design aspects of Pilot?
  - single address space
  - single language support
    - everything implemented in Mesa language
    - close coupling: Pilot implemented in Mesa, Mesa depends upon Pilot features
  - limited features for protection and resource allocation
    - language-based protection (cannot express violations)
    - does not secure against malicious processes
    - protection more to guard against errors than maliciousness
  - accept hints from applications (Question: when hints are acceptable to a time sharing system?)
    - e.g., when to page out data
  - integrated support for network communications
  - Pilot is again a kernel on which additional features can be built
    - e.g., no file system per se

5 10/12/21 CSE221 - Operating Systems, Yueshan Zhou

why paper not talking about scheduling?

Not foreseeing the Internet

# Files Attributes

**Files Attributes**

- Files have just four attributes, e.g., size and type.
- Files can also be made "permanent" and/or "immutable".
- What does "permanent" imply about a file?
  - If a file is not permanent, what is it used for?
  - How does this compare with OSes today?
- What is an "immutable" file?
  - Why is it useful? (hint: cached files, CDN) *new version everytime*

Integration of networks into Pilot have influenced how files are named. How is this?

Note that files and volumes do not implement a file system per se, but are the "kernel" of a file system. What is missing?

10/12/21 CSE221 - Operating Systems, Yuanquan Zhou

Using name/tag with special  
format instead.

## Virtual Memory

**Virtual Memory**

- What is unique about the Pilot virtual memory system?
- Linear virtual address space is partitioned into hierarchical memory "Spaces". Spaces serve three purposes:
  - Allocation: data can only be accessed in allocated regions of virtual memory
  - Mapping: maps data to backing store
  - Swapping: unit of swapping between memory and backing store (note: not pages)

10/12/21 CSE221 - Operating Systems, Yuanquan Zhou

**Implementation**

- What is an example of the circular dependency problem?
  - circularity between files and VM: files can only be accessed via virtual memory, and virtual memory requires files for backing store
- How does kernel/manager break circularity?
  - manager maintains full "database" (e.g., complete file data structures).
  - kernel does low-level operations with a cache of data
  - low-level disk I/O with already open files

## Communications

- Communications tightly integrated into Pilot.
- Many of same features and issues of networking support in TCP/Unix.
  - e.g., sockets, listen, datagrams, connections, etc.
- Not going to go into details.
- Note: clients on same computer can use network comm to do IPC
  - doesn't matter whether clients on same machine or not

CSE221 - Operating Systems, Yulanwen Zhou

Conclude

Pilob is seikh OS  
JVM is runtime library

## General Questions

- Why do you think they named the system Pilot?
- Why do we run timesharing systems (Unix) on our PCs/laptops/tablets today?
- How does Mesa/Pilot compare to Java/JVM?
- The designers of Pilot made certain assumptions about the need for protection on a personal computer system. In light of today's computing environments, do you think those assumptions would work well today?
  - In what systems today some of the assumptions are still true?

→ security missing