MINIX

James
Why Micro-kernel

- Kernel bugs are usually fatal
- User-mode bugs are not
History

- **MINIX 1.0** (1987)
  - FOR IBM PC/AT microcomputers
- **MINIX 2.0** (1997)
  - Available for x86 and SPARC architecture
  - Follow POSIX
  - Support 32-bit mode
  - TCP/IP stack
  - In kernel drivers
- **MINIX 3.0** (2005)
  - Latest version
  - High reliability
  - Many software: X11, emacs, gcc, ftp, etc.
MINIX 3

- Only has a tiny and trusted micro-kernel
- Constantly monitor the status of OS
- Automatically replace or shut down malfunctioning component
Self-repairing OS

- Major faults are properly isolated
- Defects are detected
- Failing components can be replaced on the fly
Architecture
What’re in micro-kernel

- Interrupt handling
- Memory management unit (MMU)
- Scheduling
- Interprocess communication (IPC)
- Kernel calls
- Clock driver
- System tasks
What’re in user mode

- Drivers
  - Send messages to other processes
  - Make kernel calls (read/write IO ports)
- Servers
  - Network service
    - Has complete TCP/IP stack
    - BSD socket
  - File system service
    - Process user requests
    - Talk to the disk driver for data
    - Call system calls to move data from FS server to user
- Process management
  - Process creation
  - Memory allocation
  - Handle signals
  - Implement policies
  - Call kernel calls for low-level processing
• More servers
  • Reincarnation
    • Parent process of all drivers and servers
    • Monitor system well-being by periodically inquires
  • Assumptions
    • No malicious processes
    • Problems can be detected
    • Restarting component could repair a defect

• Data store
  • Tiny database server
  • Backup states
  • Publish system configuration information
Self-repairing property
MINIX3 features

1. Small kernel size
   - 4k lines
2. Reduce the damage a bug can do
3. Drivers can no longer use privileged calls
   - e.g. disabling interrupt
4. Self-repairing
5. Soft interrupt and messages
6. Need MMU to protect private address space
7. Instruction and data space are split
8. Scheduling with dynamic priority