

CS134a: Modern OS (Week 8)

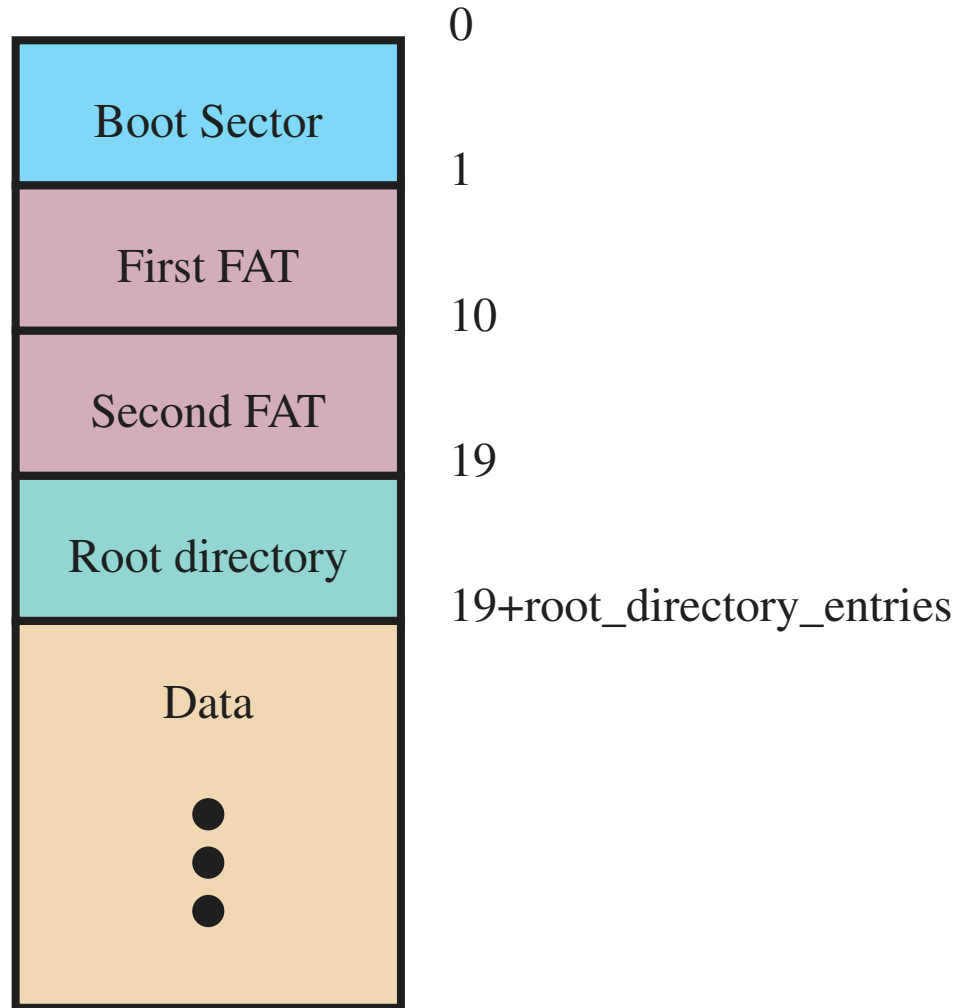
- Two rounds of research
- 1980's: modularity and micro-kernels
 - Mach (CMU; Rashid, ...)
 - Amoeba (Vrije; Tannenbaum, ...)
 - X-Kernel (U. Arizona; Peterson, ...)
 - V (Stanford; Cheriton, ...)
- 1990's: minimalism
 - ExoKernel (MIT; Kaashok, ...)
 - Vino (Harvard; Seltzer, ...)
 - **SPIN** (U. Washington; Bershad, ...)

But first lab 4!

- You will implement part of a FAT filesystem
 - *MSDOS *msdos_open(const char *devname)*
 - *void msdos_close(MSDOS *dosp)*
 - *int msdos_ls(MSDOS *dosp);*
 - *int msdos_cd(MSDOS *dosp, const char *name);*
 - *int msdos_rm(MSDOS *dosp, const char *name);*
 - *int msdos_save(MSDOS *dosp, const char *name);*



FAT filesystem structure



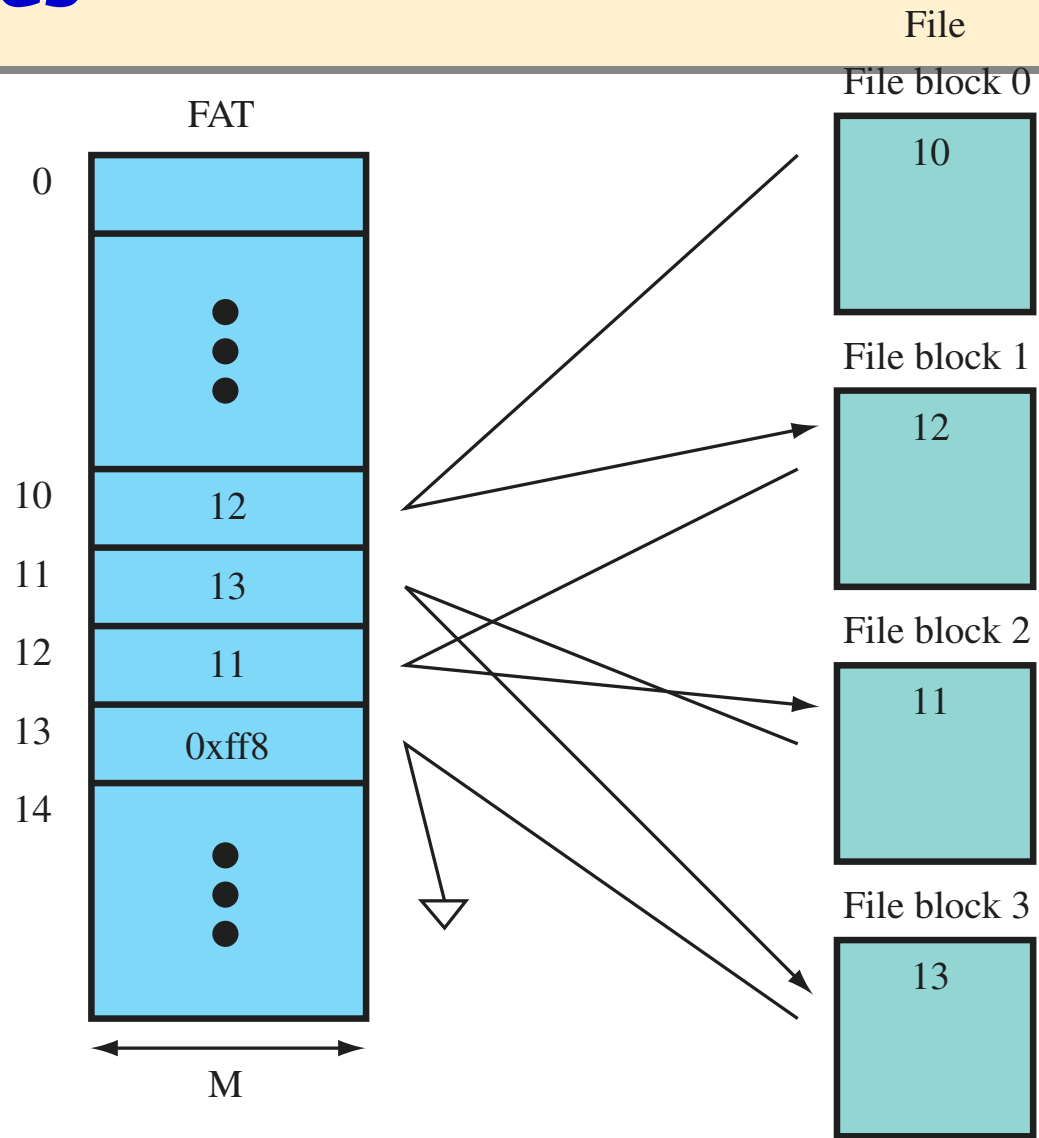
Boot sector

Boot Sector

Start	End	Description
0x00	0x02	Jump to 0x1e
0x03	0x0a	Manufacturer name
0x0b	0x0c	Sectors per cluster
0x0d	0x0f	Reserved sectors for the boot record
0x10	0x10	Number of FATs
0x11	0x12	Number of root directory entries
0x13	0x14	Number of logical sectors
0x15	0x15	Obsolete
0x16	0x17	Sectors per FAT
0x18	0x19	Sectors per track
0x1a	0x1b	Number of heads
0x1c	0x1d	Number of hidden sectors
0x1e	...	Boot program



FAT tables



First FAT entry

- First entry is reserved

Descriptor	Heads	Sectors/Track	Cylinders	Capacity
0x0fe	1	8	40	160K
0x0ff	2	8	40	320K
0x0fc	1	9	40	180K
0x0fd	2	9	40	360K
0x0f9	2	9	80	720K
0x0f0	2	18	80	1440K

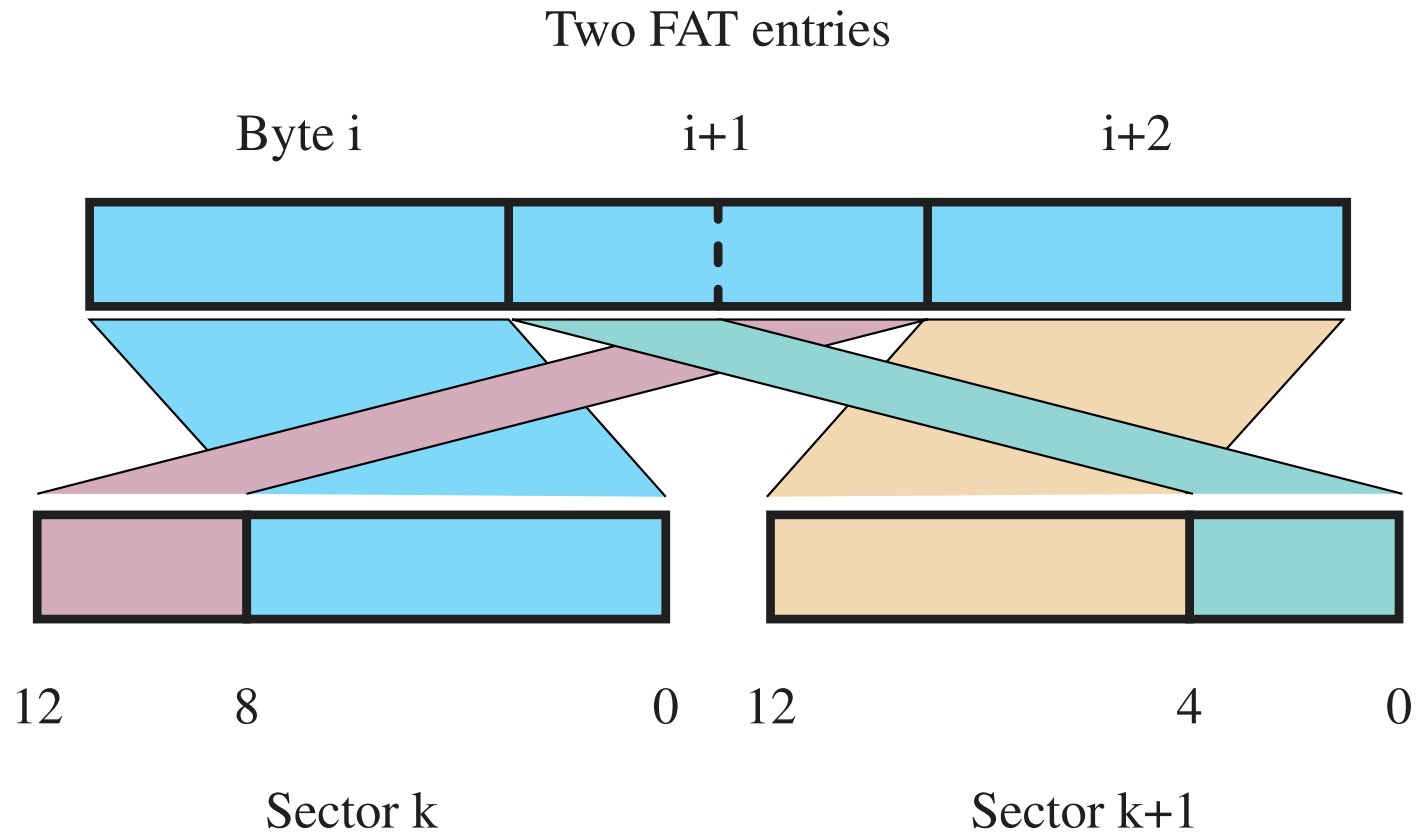


Other FAT entries

Value	Meaning
0x000	Unused
0xff0-0xff6	Reserved
0xff7	Bad cluster
0xff8-0xffff	Last cluster of a file
anything else	Number of the next cluster in a file



FAT12 packing



Directory Entry

Offset	Length	Description
0x00	8	Filename
0x08	3	Extension
0x0b	1	Attributes
0x0c	10	Reserved
0x16	2	Time (coded as $\text{hour} * 2048 + \text{min} * 32 + \text{sec} / 2$)
0x18	2	Date (coded as $(\text{year} - 1980) * 512 + \text{month} * 32 + \text{day}$)
0x1a	2	First cluster
0x1c	4	File size (in bytes)



Entry Attributes

Bit	Mask	Attribute
0	0x01	Read-only
1	0x02	Hidden
2	0x04	System
3	0x08	Volume label
4	0x10	Subdirectory
5	0x20	Archive



SPIN: Extensible OS

- Extensions like VINO
- Safety of extensions are guaranteed by compiler
- Modula-3
 - Like a modular Pascal
 - Static/run-time type checking guarantee safety
 - SFI is run-time checking

SPIN

- u Efficient application-specific specialization and extension of system services
- u Provide safe access to physical and virtual resources
- u Support multiple users simultaneously
- u Predictable resource allocation
- u Core set of useful services (threads, VM, etc)
- u Real OS on top

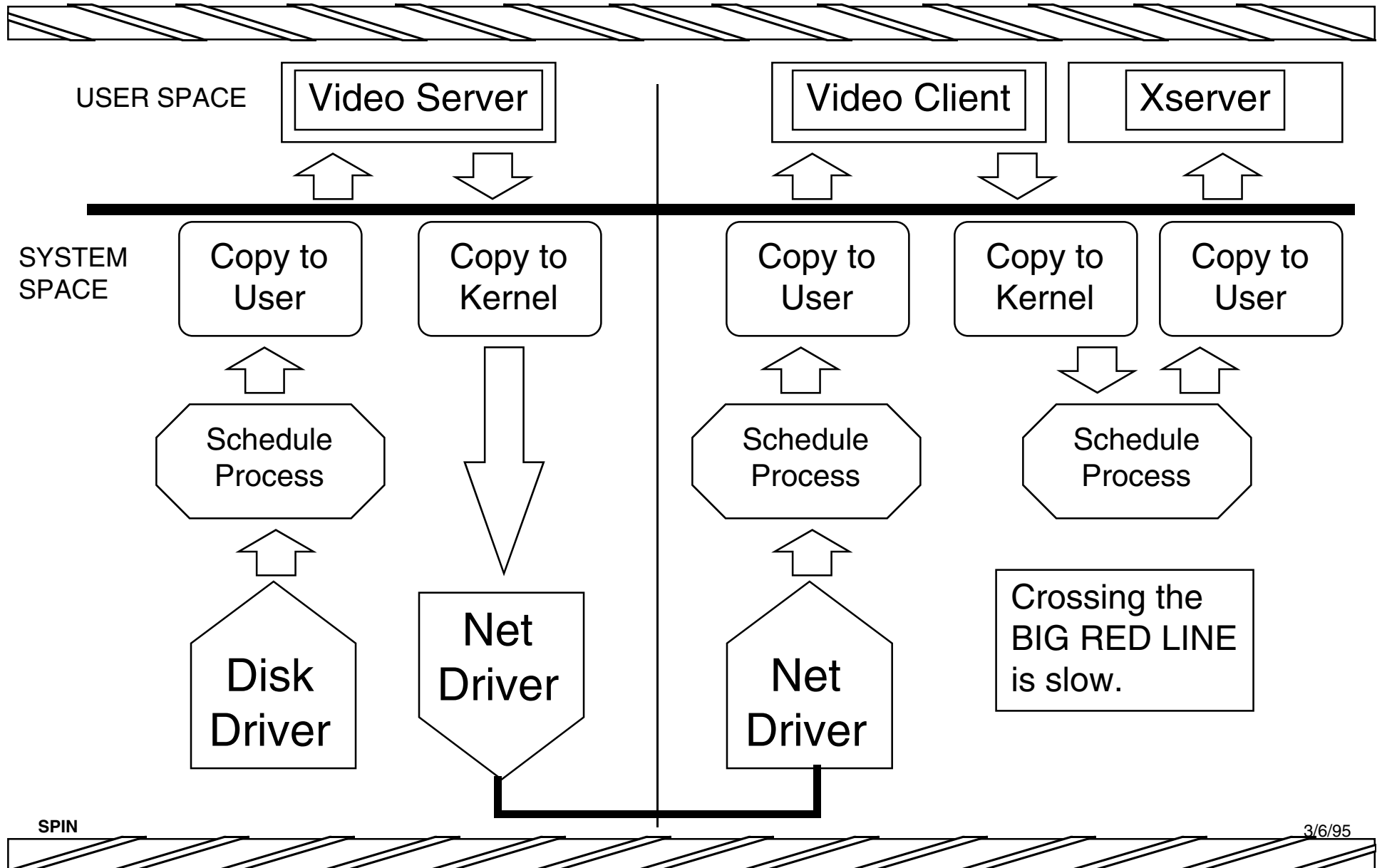
Some Leading Questions

- u 100MIPs workstation but only 10 HTTP connections/second?
- u 5 TPS in a Dist transaction system?
- u Jerky video?
- u 5 seconds to context switch on a PowerPC Mac?
- u How come DOS/MAC programs do all those tricks with the file system, but your workstation doesn't?
- u How come your workstation stays up but DOS doesn't seem to.

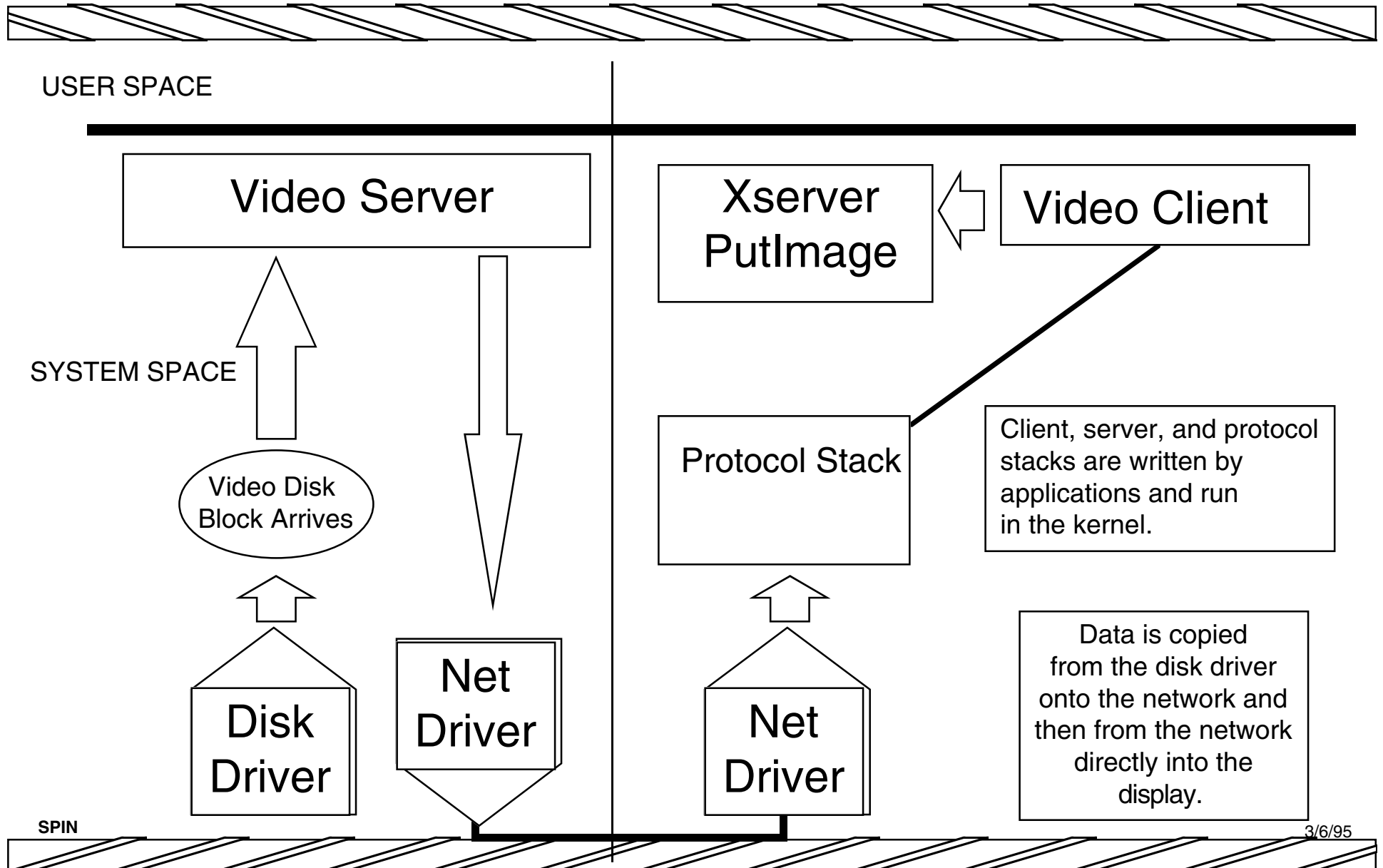
The Problem

- u Diverse application requirements.
 - Database and TPS
 - Parallel and scientific
 - Multimedia/Realtime
 - Secure HTTP Gateways and Redirectors
 - GroupWare
 - Emulators
- u Limited operating system services
 - files
 - network
 - scheduler
 - V M

How Networked Video Works Now



The Way It Ought to Be



Specializing Operating Systems is Hard Work

- u **Complexity**

- Most people don't know how.

- u **Prohibition**

- Most environments don't permit it.

- u **Barriers**

- Applications aren't allowed (or shouldn't).

- u **Fear**

- correctness, safety, performance

- u **Economics**

- if it's so painful, it ought to be general, but...

Some examples of application-specific services

u Filing and buffer cache management

- multimedia, mail, secure anything

u Protocol processing

- Network splice
- Active messages
- Integrated DSM

u Scheduling and thread management

- Optimistic synchronization
- Asynch I/O, Multi-level schedulers

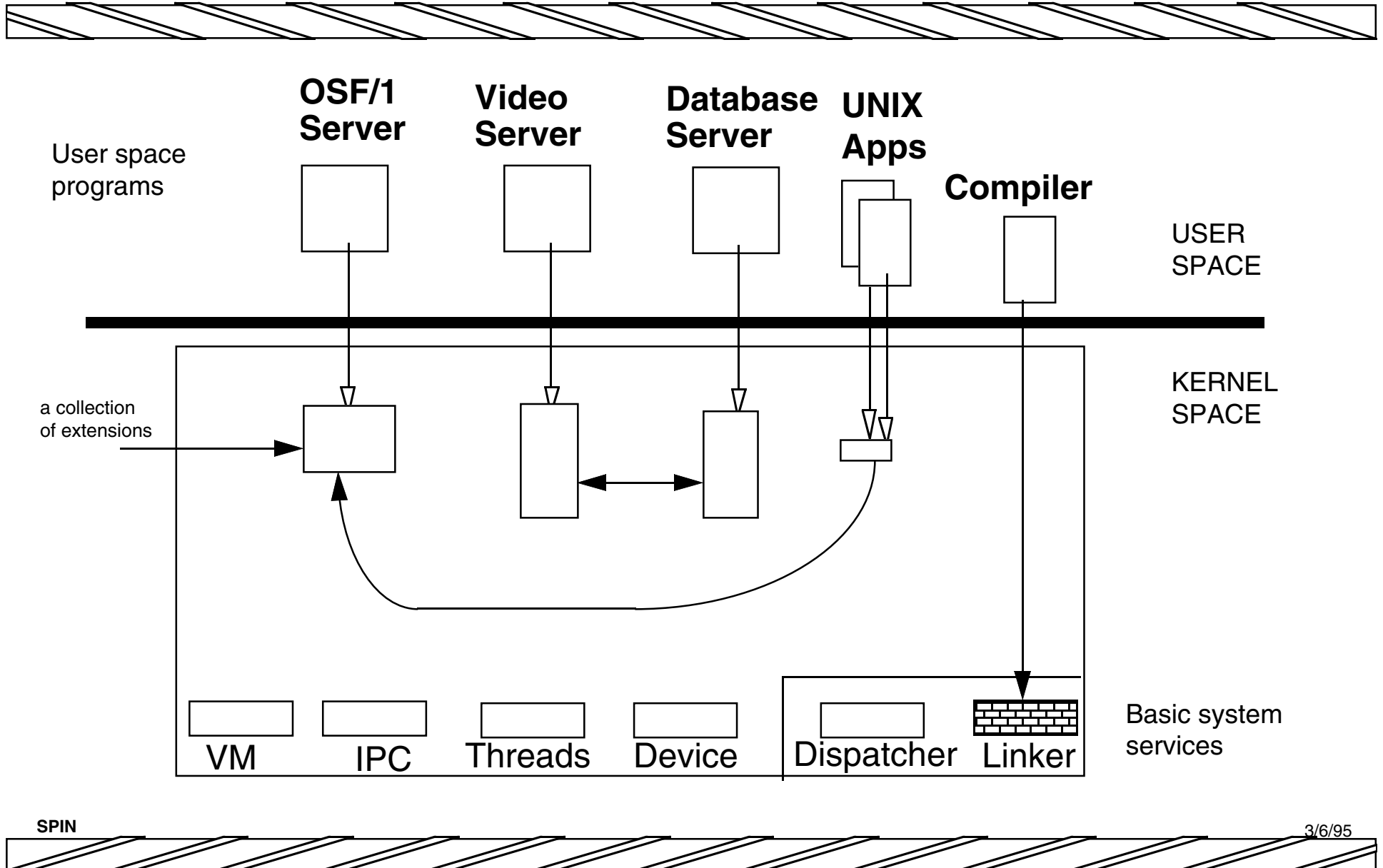
u Virtual memory

- Mach, Database, Cao&Li, Hipec, V++, TLB-prefetch, Network resource reservation, real-time, cache management

Application-specific services with SPIN

- u Extensions are linked into a running kernel.
 - inexpensive sharing
- u Modular, pointer-safe extension language
 - no bad access
- u Logical protection domains within the kernel isolate “semantic” failures.
 - inexpensive context switch
- u Exposed kernel interfaces implemented defensively.
 - cannot rely on something you cannot trust.

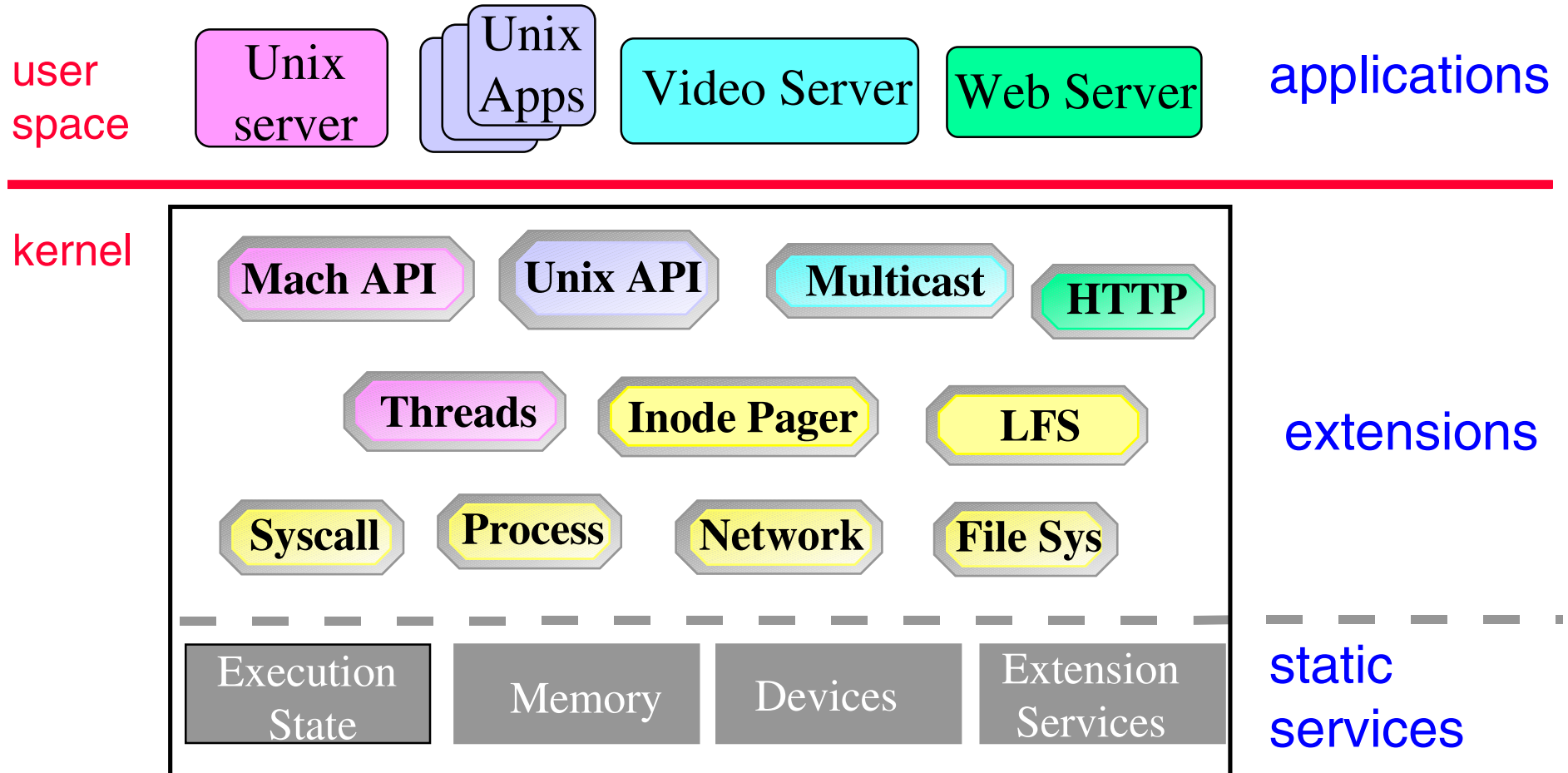
SPIN System Components



What can you extend?

- System calls (install handlers)
- Publish/subscribe events
 - Publish a function by *name*
 - Subscribe by name

The *SPIN* Operating System



Requirements

- Flexible
- Simple
- Fast
- Safe

Existing Approaches

- Dynamic linking
- Service-specific interfaces
- Object-orientation
- Publish-subscribe communications
(events)

The *SPIN* Approach

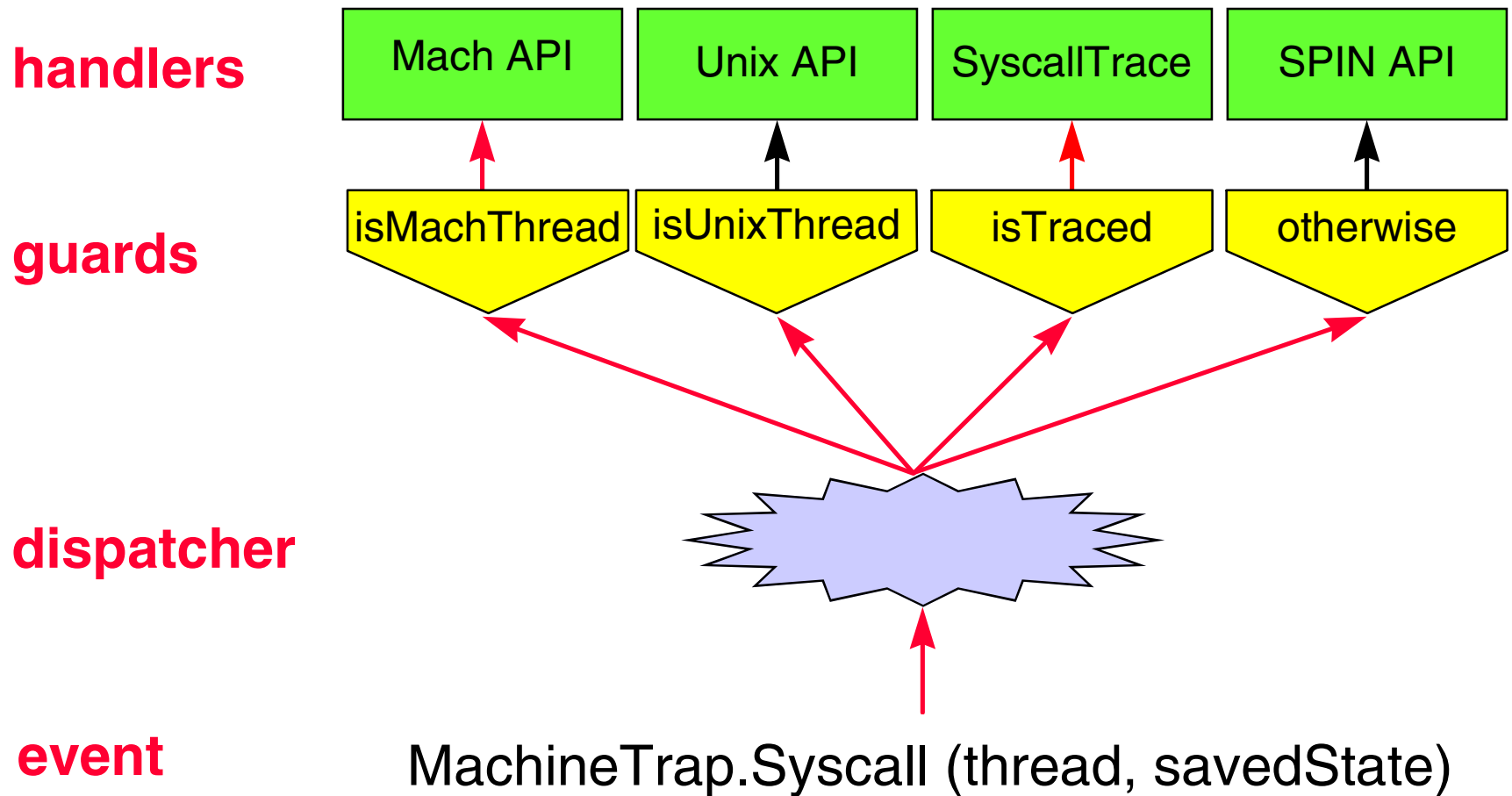
- Combine the best of all worlds
 - Flexibility of publish-subscribe communication
 - Language support for uniformity and simplicity
- Plus
 - Access control for safety
 - Runtime code generation for performance

SPIN Events

EVENTS = PROCEDURES + FLEXIBILITY

- Procedure syntax
- Extended procedure semantics
 - Multicast, conditional execution, asynchrony
- Every procedure is an event

The life of a system call



Programmer's View

Event definition

```
INTERFACE MachineTrap;  
PROCEDURE Syscall(  
    thread : Strand.T;  
    state  : SavedState);  
  
END MachineTrap.
```

Event invocation

```
MODULE MachineTrapPrivate;  
  
PROCEDURE Syscall(thread, state) =  
    BEGIN  
        ...  
        MachineTrap.Syscall(thread, state)  
        ...  
    END Syscall;  
END MachineTrapPrivate.
```

Extension

```
MODULE MachEmulation;  
  
PROCEDURE Handler (thread, state) =  
    CASE state.v0 OF  
        ...  
        | -65 => VM.vm_allocate();  
        ...  
    END Handler;  
  
PROCEDURE Guard (thread, state)  
    : BOOLEAN =  
    RETURN IsMachThread(thread);  
END Guard;  
  
BEGIN  
    Dispatcher.InstallHandler(  
        MachineTrap.Syscall,  
        Guard, Handler);  
END MachEmulation.
```

Applicability

- Available in the whole system
- Used to implement
 - System calls
 - Threads
 - Networking
 - VM
- How can an untrusted application use them?

Safety

- Events enable communication with untrusted extensions
- Protect
 - Visibility of events
 - Handler installation
 - Event delivery
- Prevent denial of service

Handler installation

- Event's authorizer verifies handler installations



Event delivery

- Imposed guards dynamically check the right to receive the event occurrence

