Extensibility, Safety and Performance in the SPIN Operating System

Brian Bershad, Stefan Savage, Przemyslaw Pardyak,
Emin Gun Sirer, Marc E. Fiuczynski, David Becker,
Craig Chambers, Susan Eggers

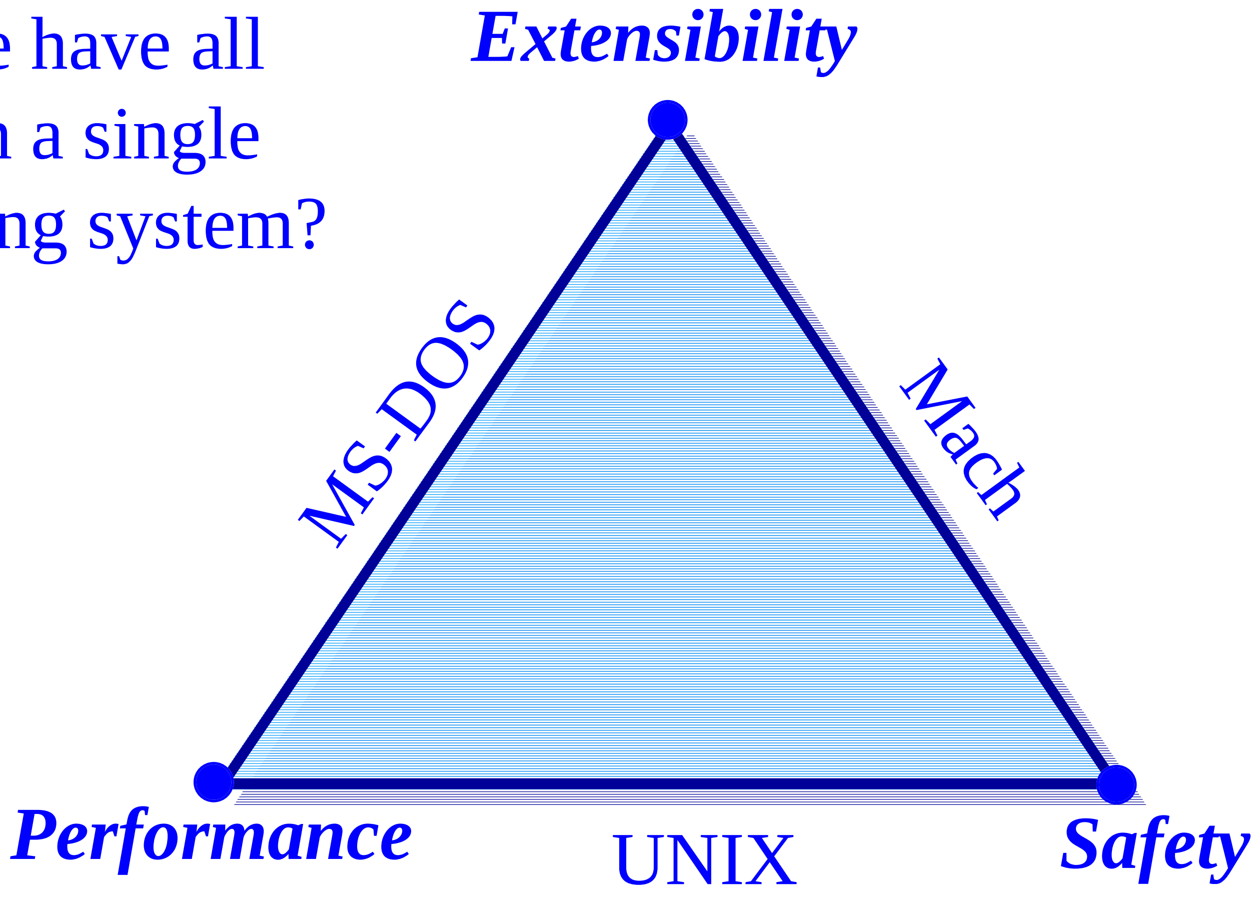
Department of Computer Science and Engineering
University of Washington

Goals

- Extensibility
  - Applications can dynamically extend system to provide specialized services
- Safety
  - Kernel is protected from actions of extensions
- Performance
  - Extensibility and safety have low cost

Why is this hard?

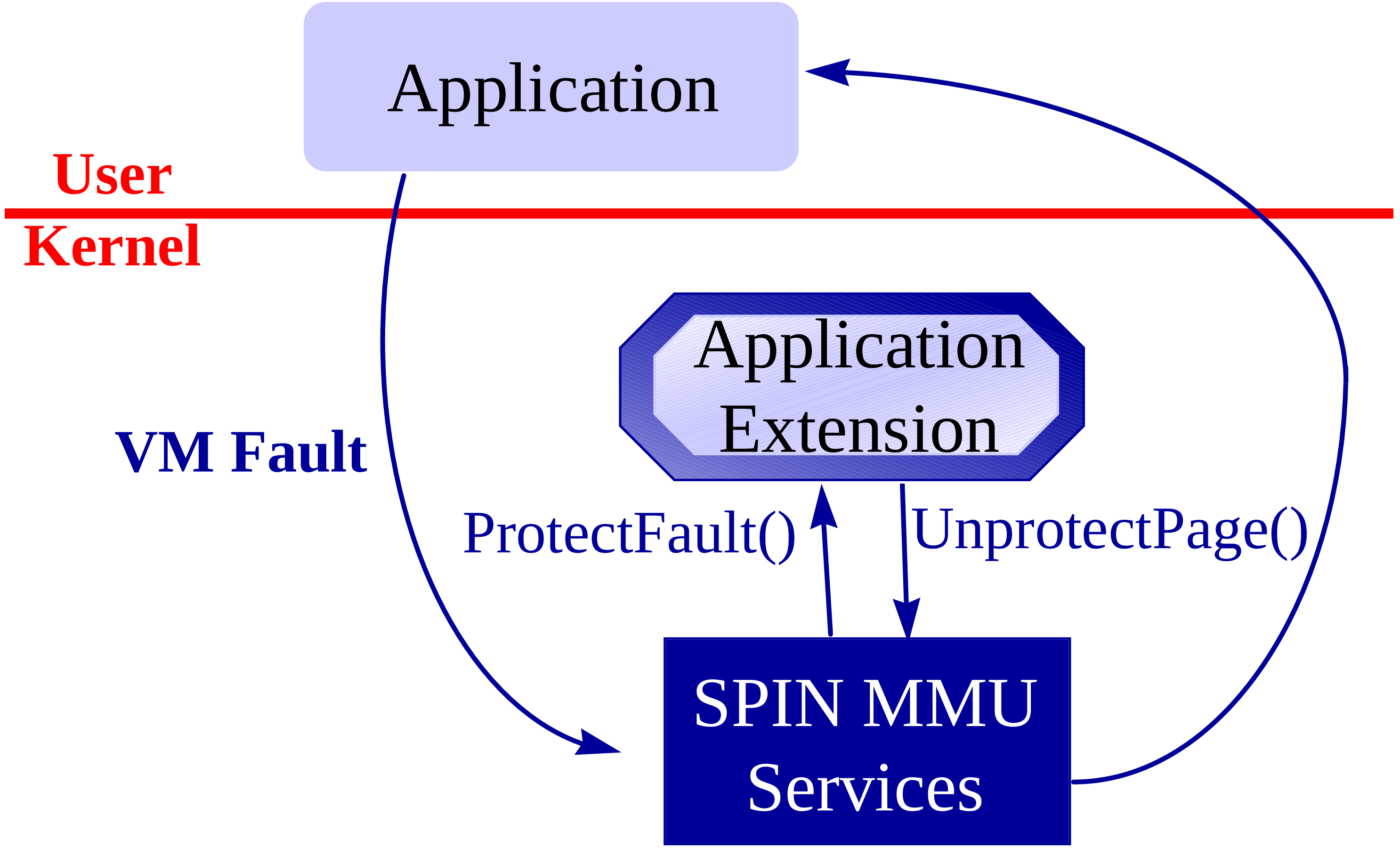
Can we have all three in a single operating system?



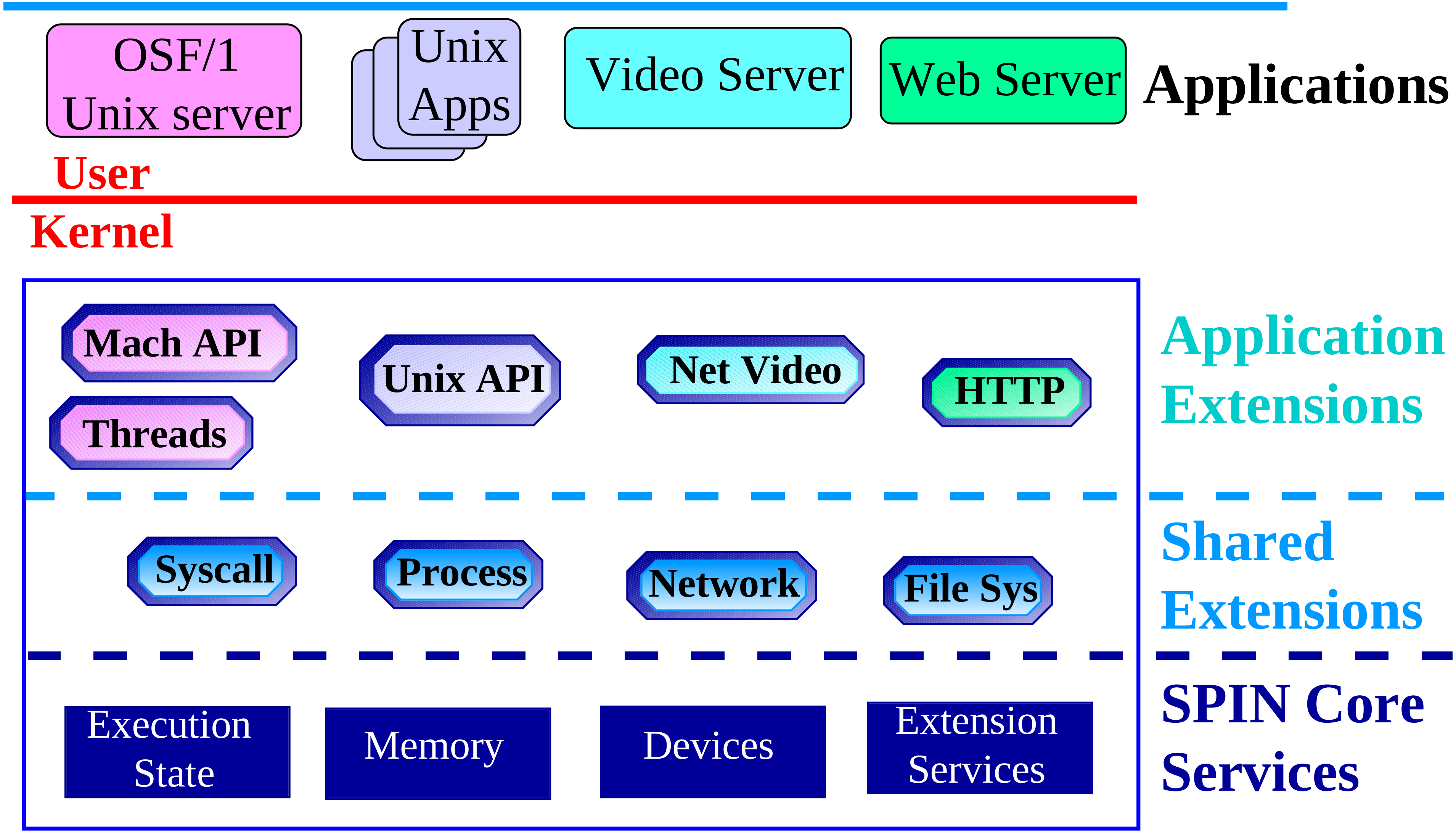
Approach

- Put extension code in the kernel
  - Cheap communication
- Use language protection features
  - Static safety
- Dynamically interpose on any service
  - Fine-grained extensibility

A SPIN extension



SPIN structure



Safety

Language-based protection

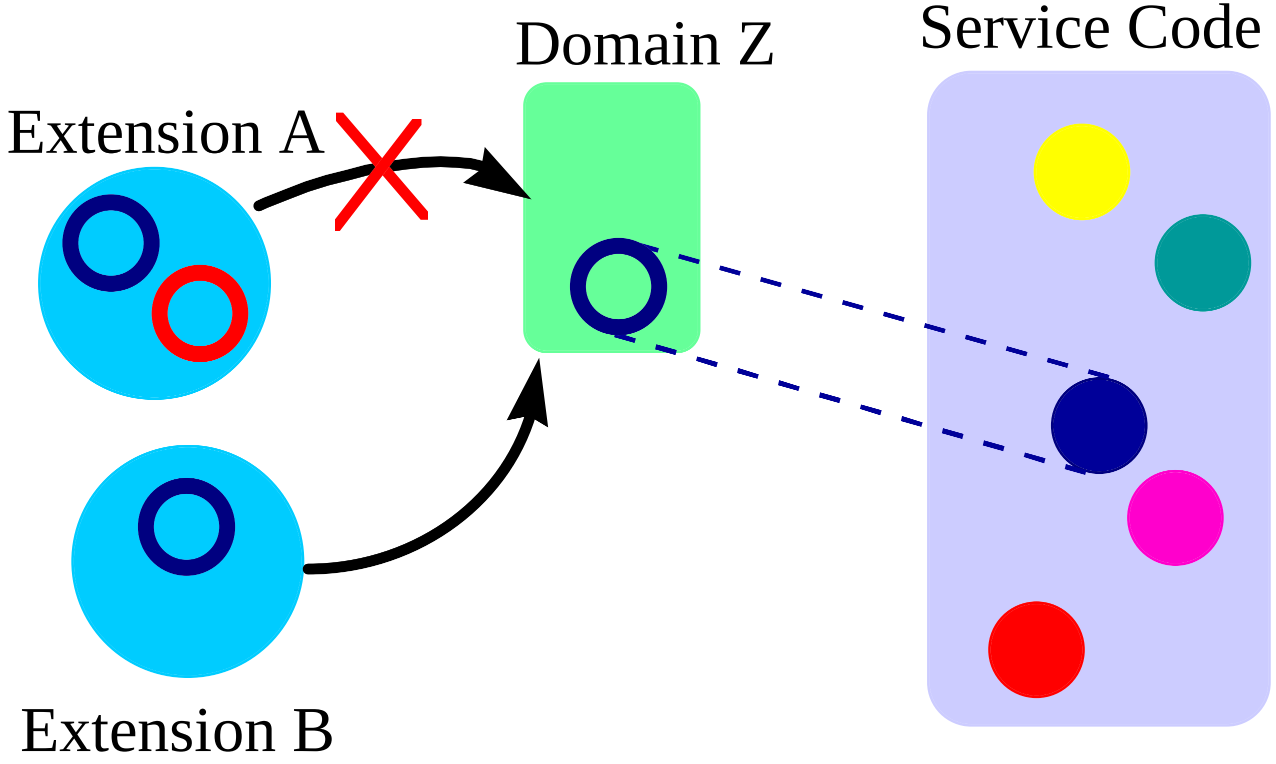
Modula-3

- Memory safe
- Interfaces for hiding resources
- Cheap capabilities

Restricted dynamic linking

Goal: control access to interfaces cheaply

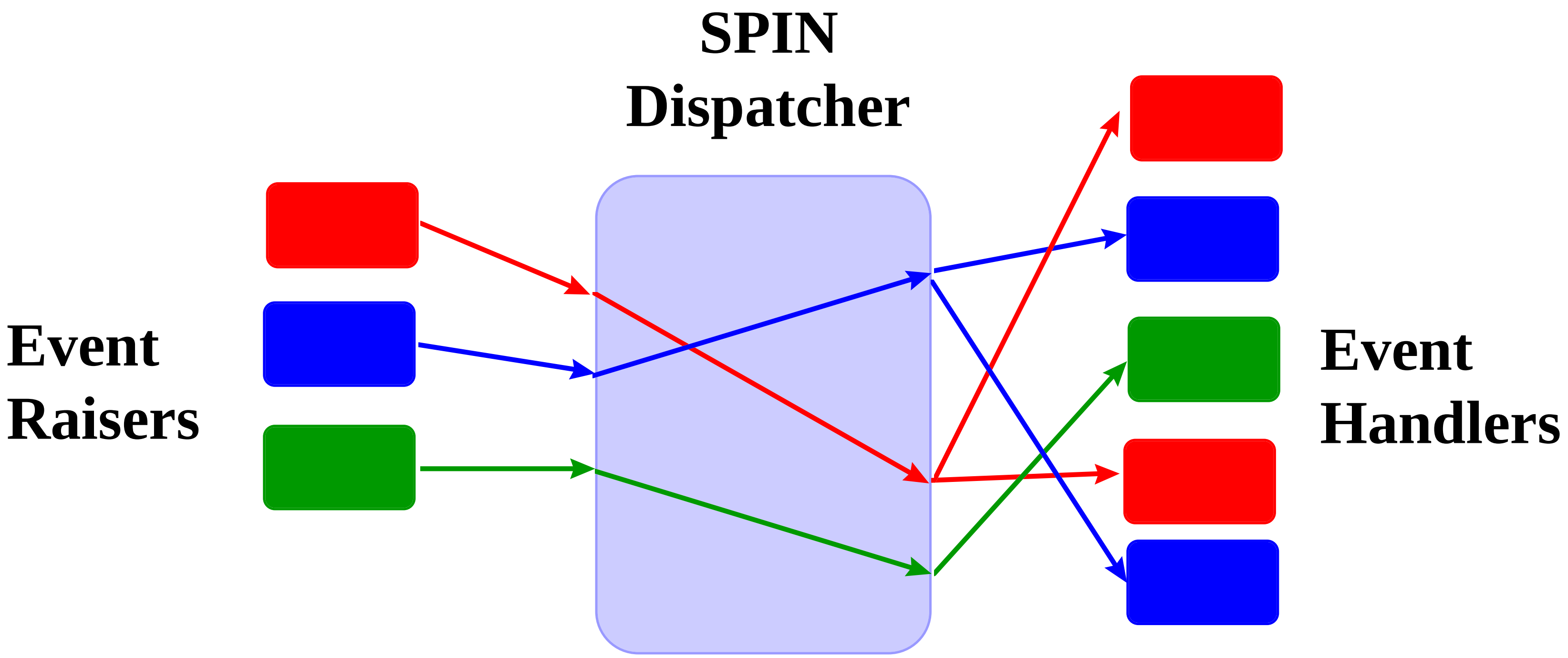
Strategy: restrict access at dynamic link-time



Extensibility

Dispatcher

Event-based communication model



Using Events

```
INTERFACE Network;
PROCEDURE PacketArrived(p:Pkt);

END Network.
```

Event definition

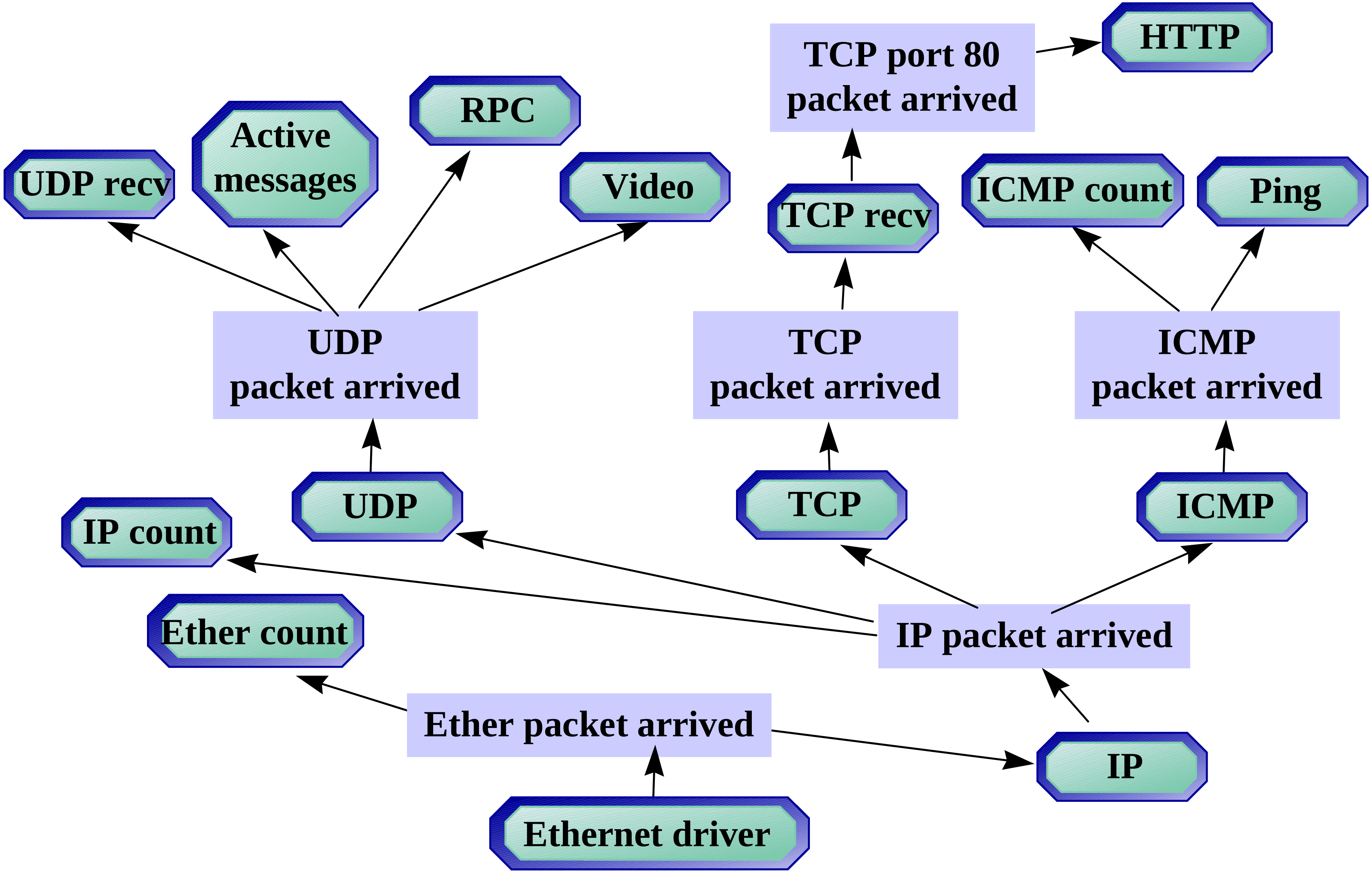
```
MODULE EthernetDriver;
PROCEDURE Interrupt(p: Pkt) =
  BEGIN
      Network.PacketArrived(p);
  END Interrupt;
```

Event raise

Other services

- Almost all “system” services are extensions
  - Network protocols
  - File systems
  - System call interface
- SPIN only implements services which cannot be safely implemented as extensions
  - Processor execution state
  - Basic interface to MMU and physical memory
  - Device IO/DMA
  - Dynamic linker and Dispatcher

A protocol graph in SPIN



Design summary

- Safety
  - Memory safe language for extensions
  - Link-time enforcement for access control
- Extensibility
  - Fast and safe centralized control transfer switch
- Result
  - Allows fast and safe fine-grained service extension

Performance

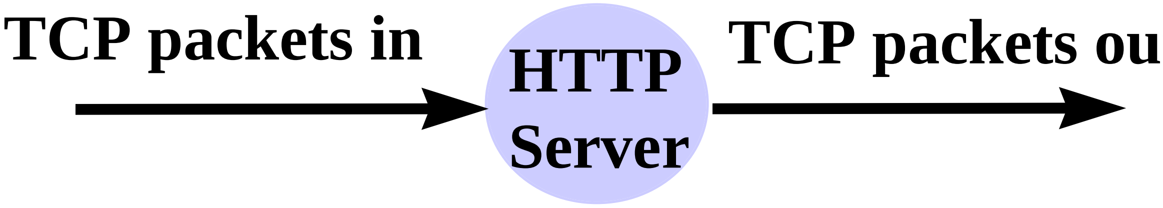
Platform

- SPIN runs on DEC Alpha platforms
- Measurements
  - DEC AXP 3000/400 @ 133Mhz
- Comparison systems
  - DEC OSF/1 V2.1
  - Mach 3.0

SPIN performance advantages

- Extensions provide specialized service
  - Don’t execute unnecessary code
- Extensions close to kernel services
  - Low latency response to faults/interrupts
  - Invoking services is cheap

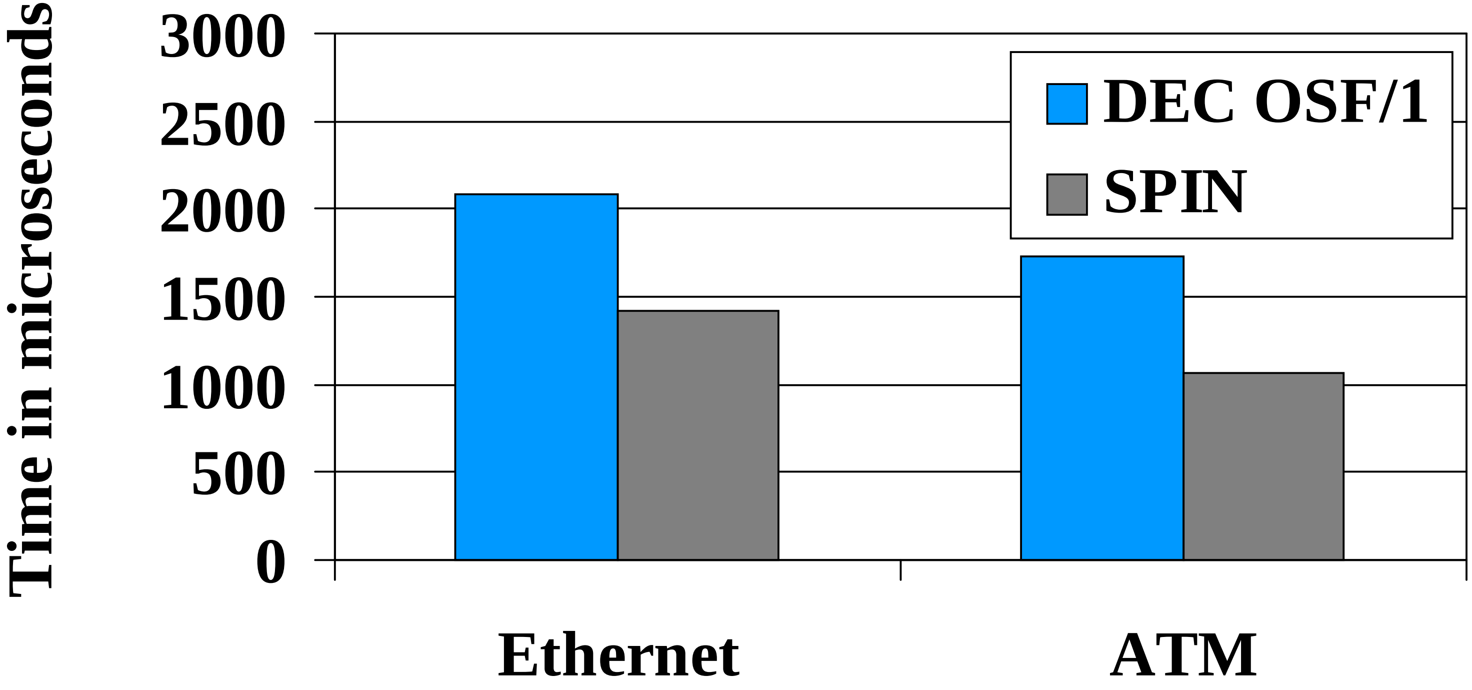
Per-port TCP packet forwarding



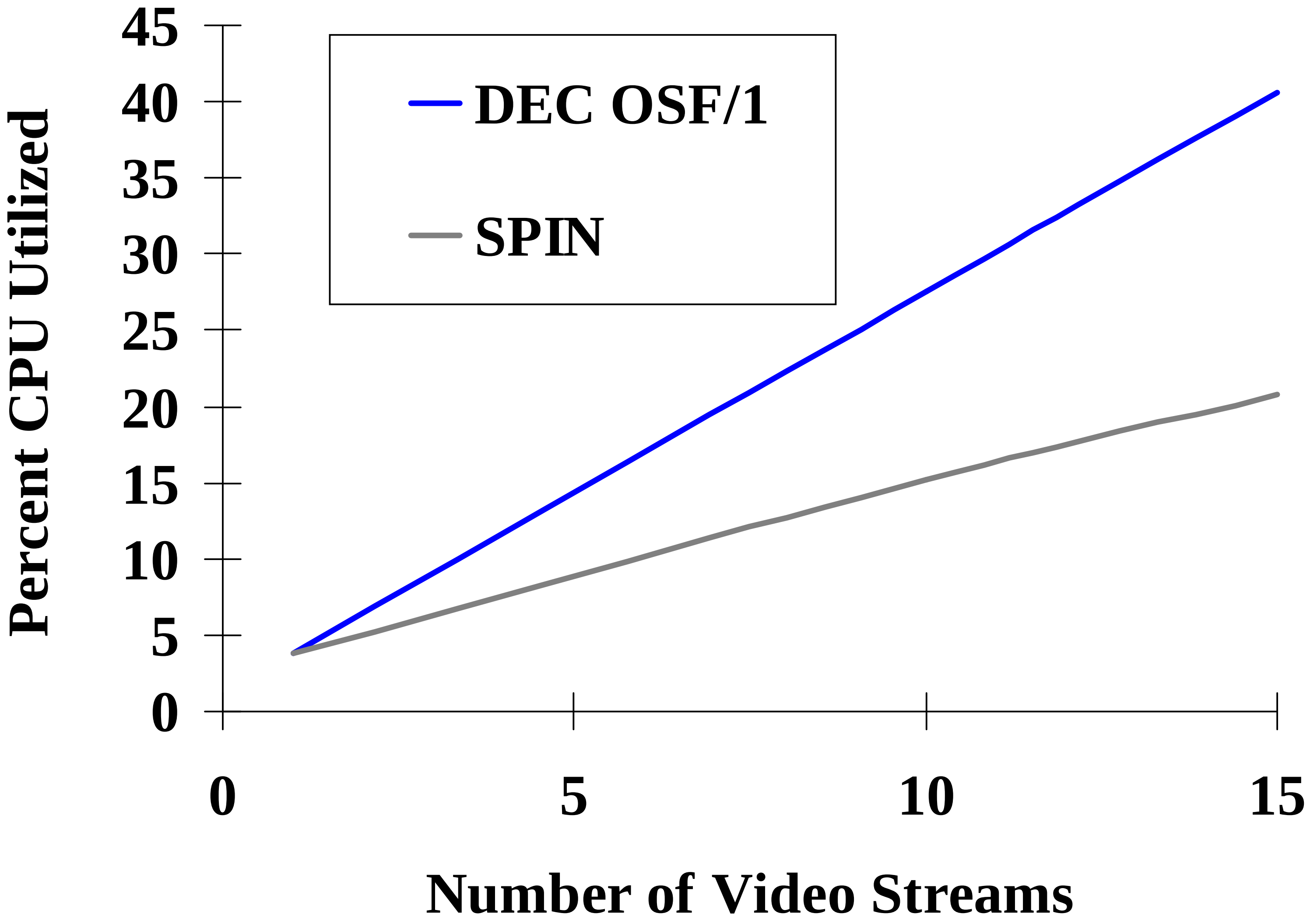
TCP packets in

HTTP Server

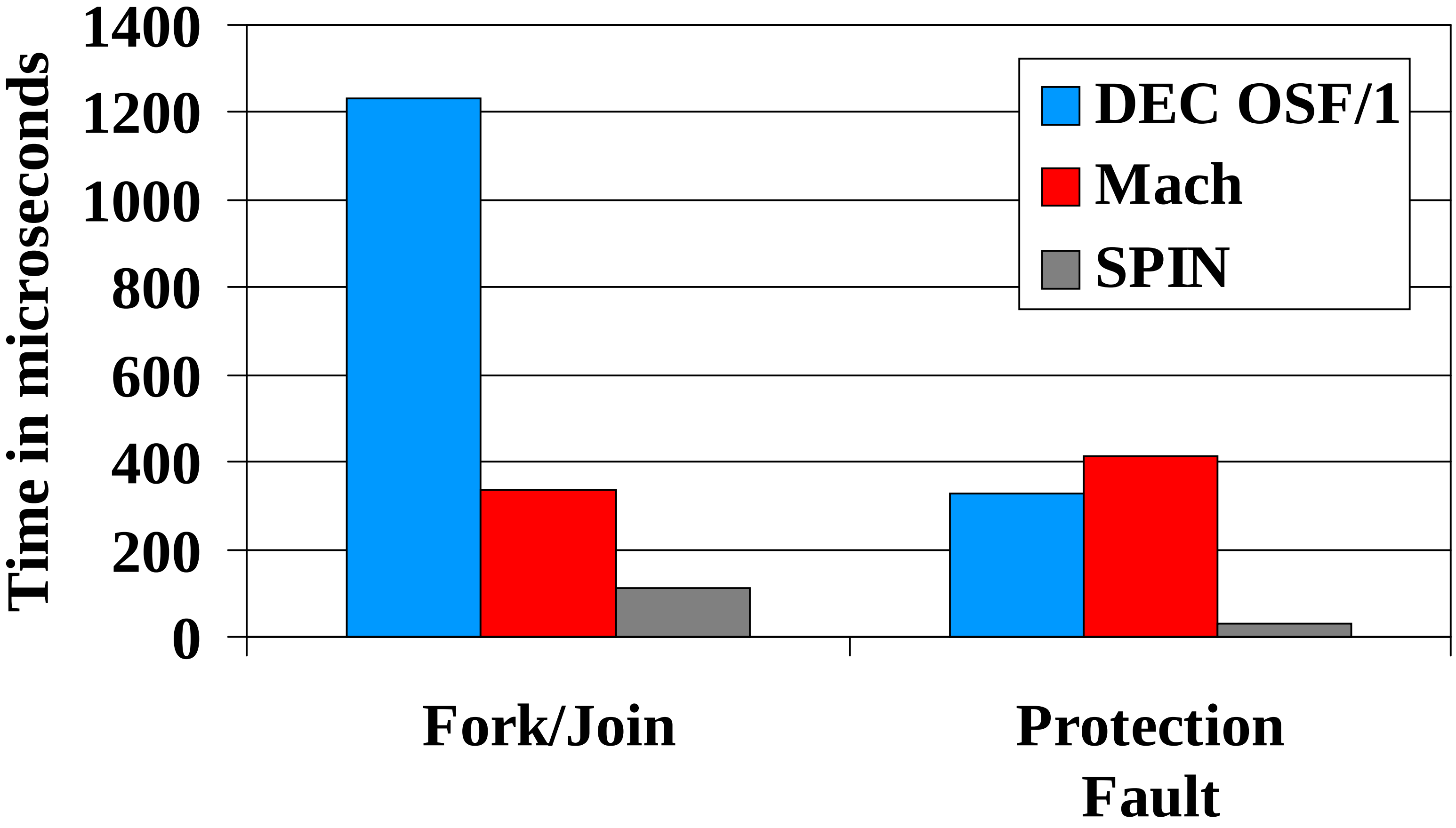
TCP packets out



Video service

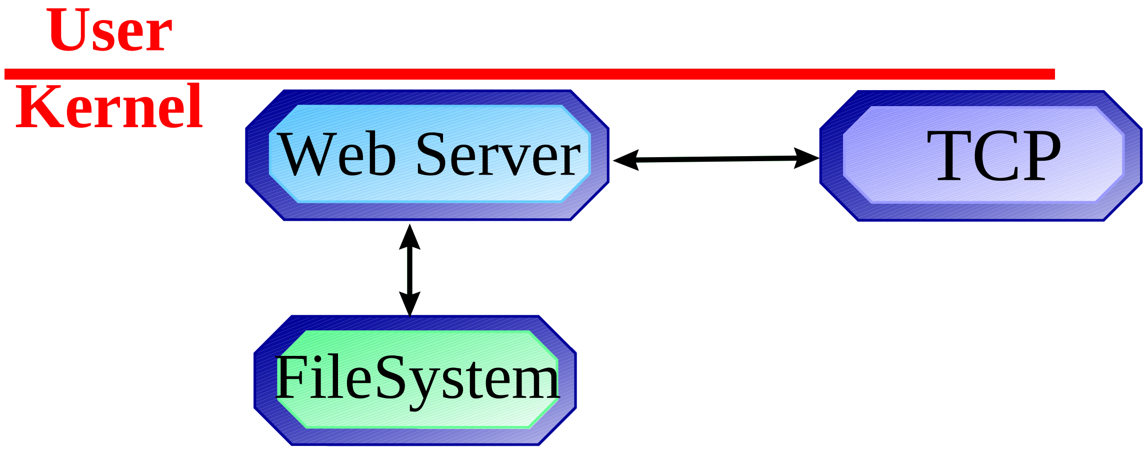


Other basic system services



Conclusions

- It is possible to combine extensibility, safety and performance in a single system
- Static mechanisms, implemented through the compiler, make this possible
- **http://www-spin.cs.washington.edu/**



Language-based capabilities

```
INTERFACE PageTable;
TYPE T <: REFANY;

PROCEDURE New(): T;
END PageTable.
```

```
INTERFACE PageTableInternal;
REVEAL PageTable.T =
          BRANDED REF RECORD
            PTBase: ADDRESS;
            …
          END;
END PageTableInternal.
```

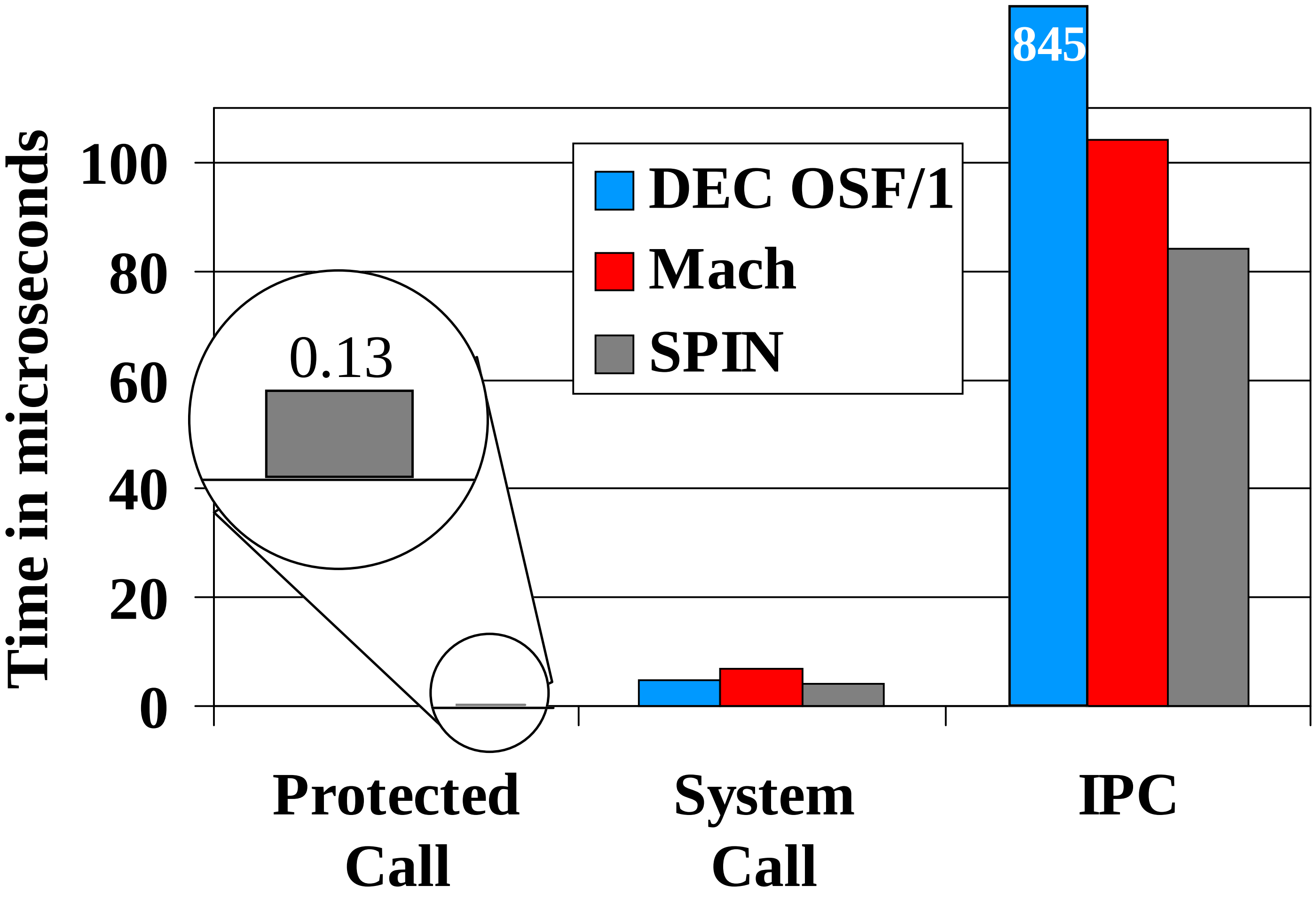
```
t := PageTable.New();
```

Event implementation

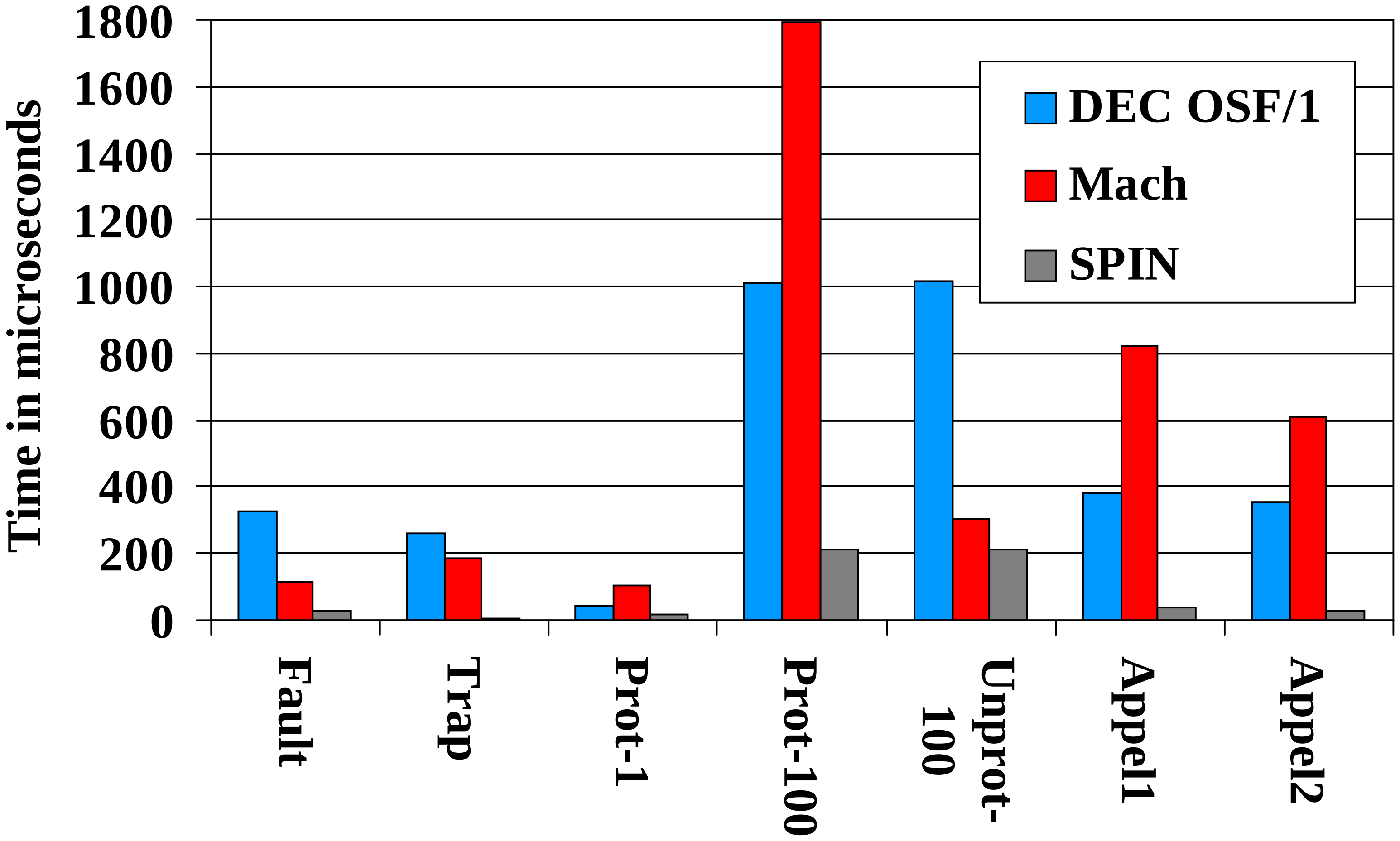
Use procedure call to define and invoke events

- Convenient syntax
- High performance implementation for common case
- Can protect events using domains
- Most procedures in the system can be extended

Protected communication



Memory management services



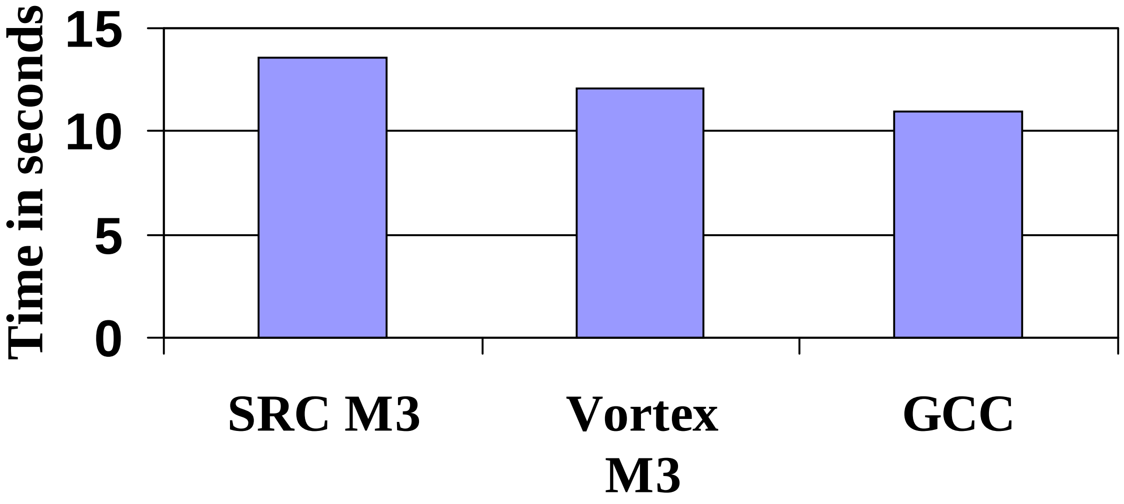
Modifications to Modula-3

- Memory safe cast
  - VIEW operator
- Procedures which may be terminated
  - EPHEMERAL procedure type
- Naming code
  - INTERFACE UNIT, MODULE UNIT
- Universal procedure type
  - PROCANY reference type

Performance of M3 vs C

- Most operations are compiled equivalently whether written in M3 or C
- M3 can sometimes introduce runtime checks to guarantee type safety

MD5 checksum benchmark


Time in seconds
SRC M3
Vortex M3
GCC