Extensibility, safety and Performance in the SPIN Operating System

Brian N. Bershad *et al*.
Department of Computer Science and Engineering
University of Washington

Presenters: Meiyuan Zhao & Song Ye (Group 4)
Dartmouth College
CS108 Winter 2002

SPIN

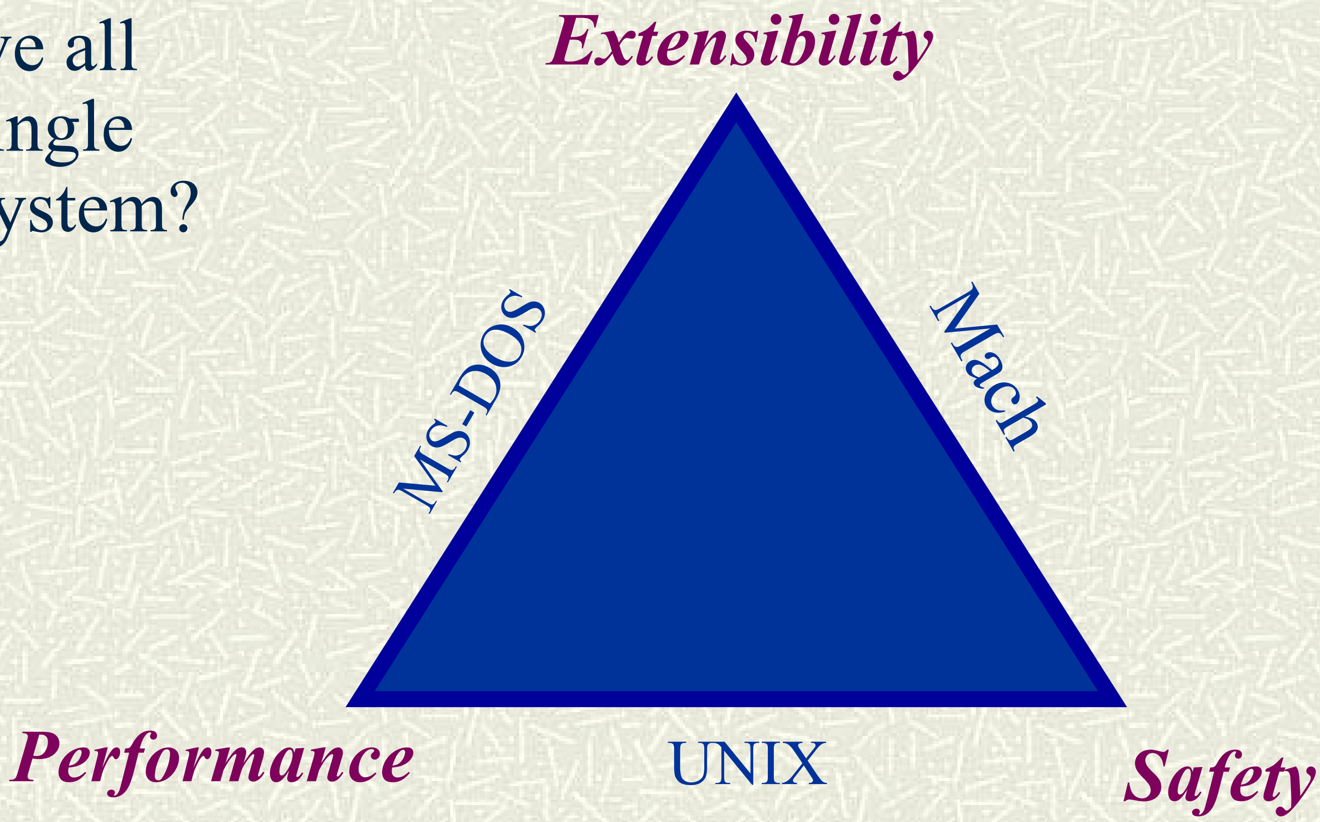
- What is SPIN?
- Goals
  - Extensibility
  - Safety
  - Performance

Outlines

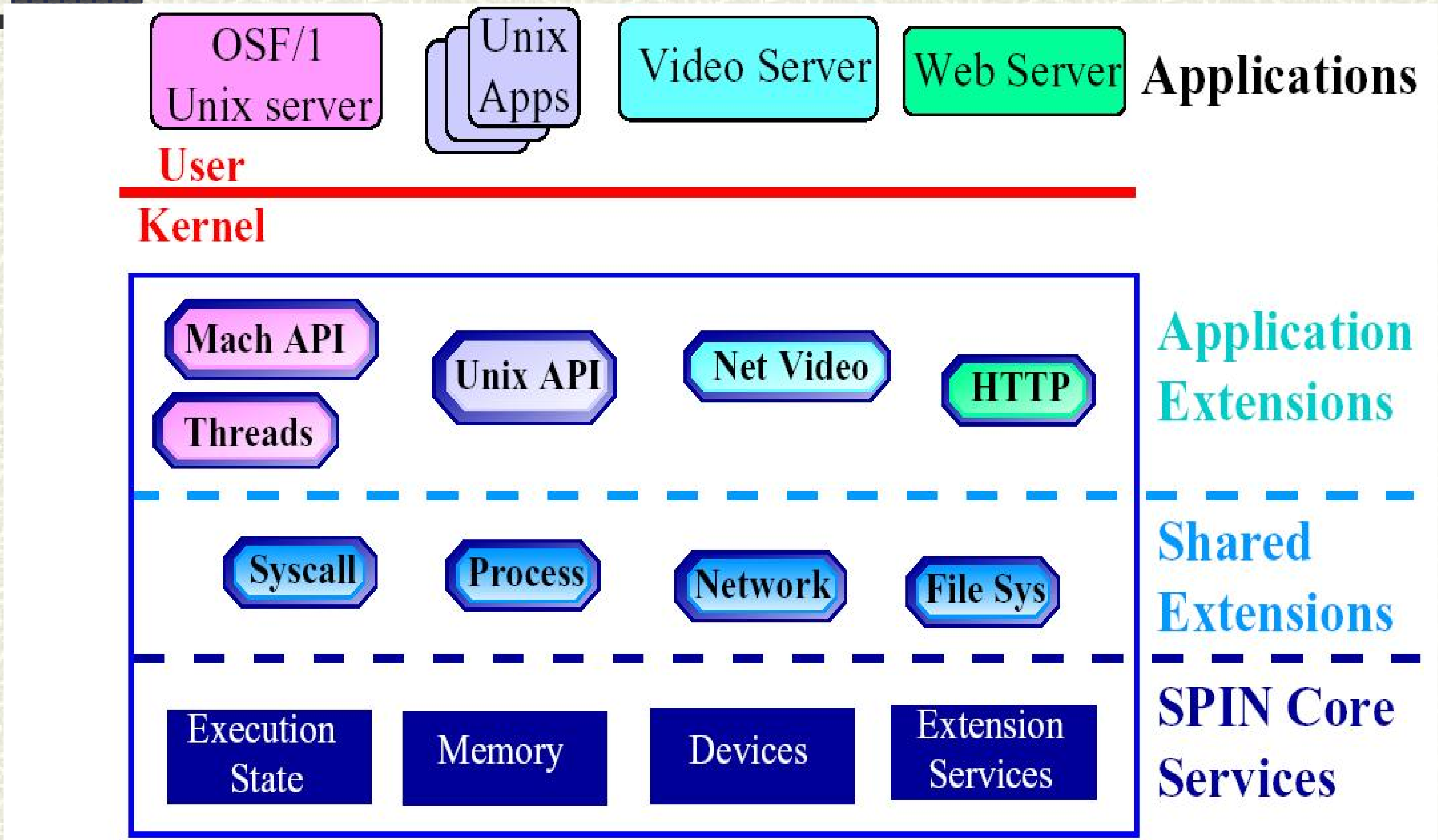
- SPIN Structure
- Safety
- Extensibility
- Design Summary
- Performance
- Conclusion

Why is this hard?

Can we have all three in a single operating system?



SPIN Structure



Approach

- Co-location
  - *Put extension code in the kernel*
- Enforce modularity
  - *Language protection*
- Logical protection domains
  - *Language protection*
- Dynamic call binding
  - *Dynamically interpose on any service*

Safety

- Modula-3
  - Memory safe
  - Interfaces for hiding resources
  - Cheap capabilities

Safety -- Memory Safe

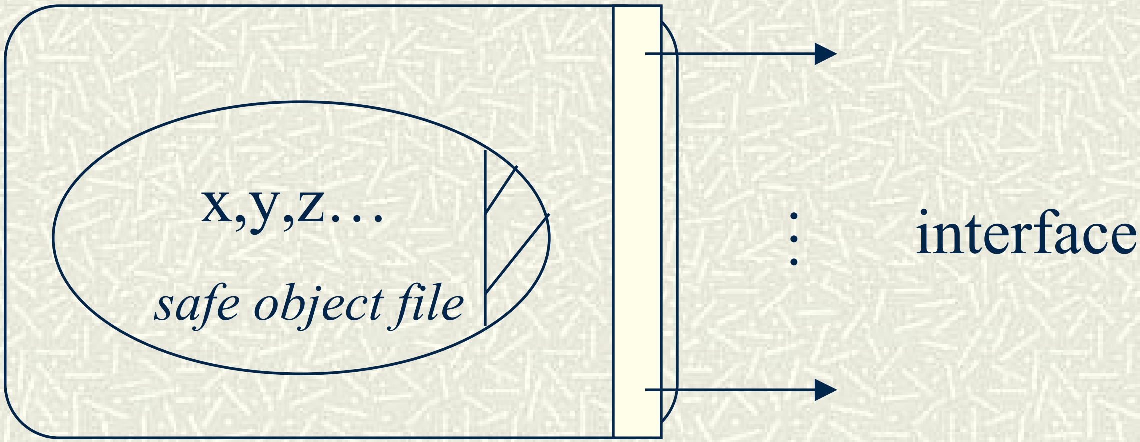
- A pointer may only refer to objects of its referent's type
- Array indexing operations must be checked for bounds violation

Safety -- Capabilities

- All kernel resources in SPIN are referenced by capabilities
- Implemented using pointers, by support of language

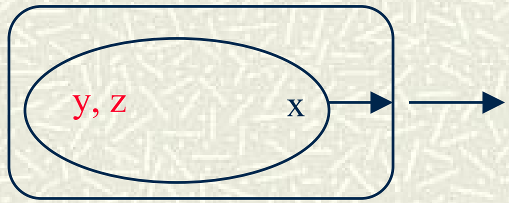
Safety -- Protection Domains

- Protection domain: A set of names
- Used to control dynamic linking

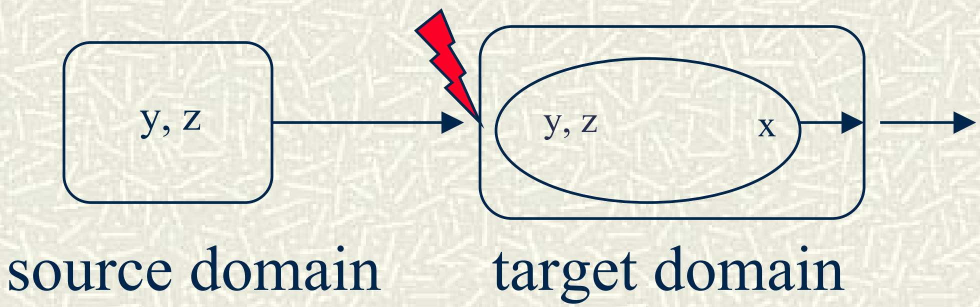


Domain Operations

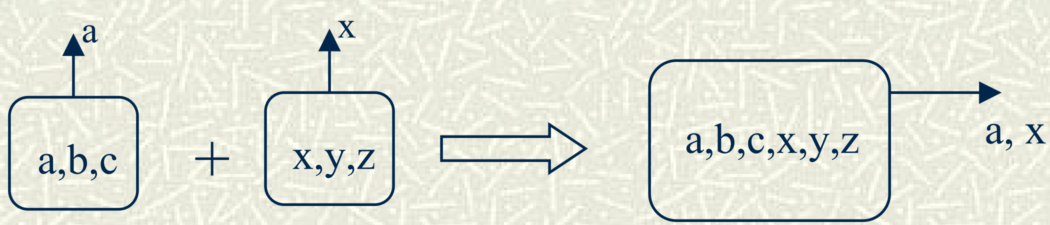
- Create



- Resolve

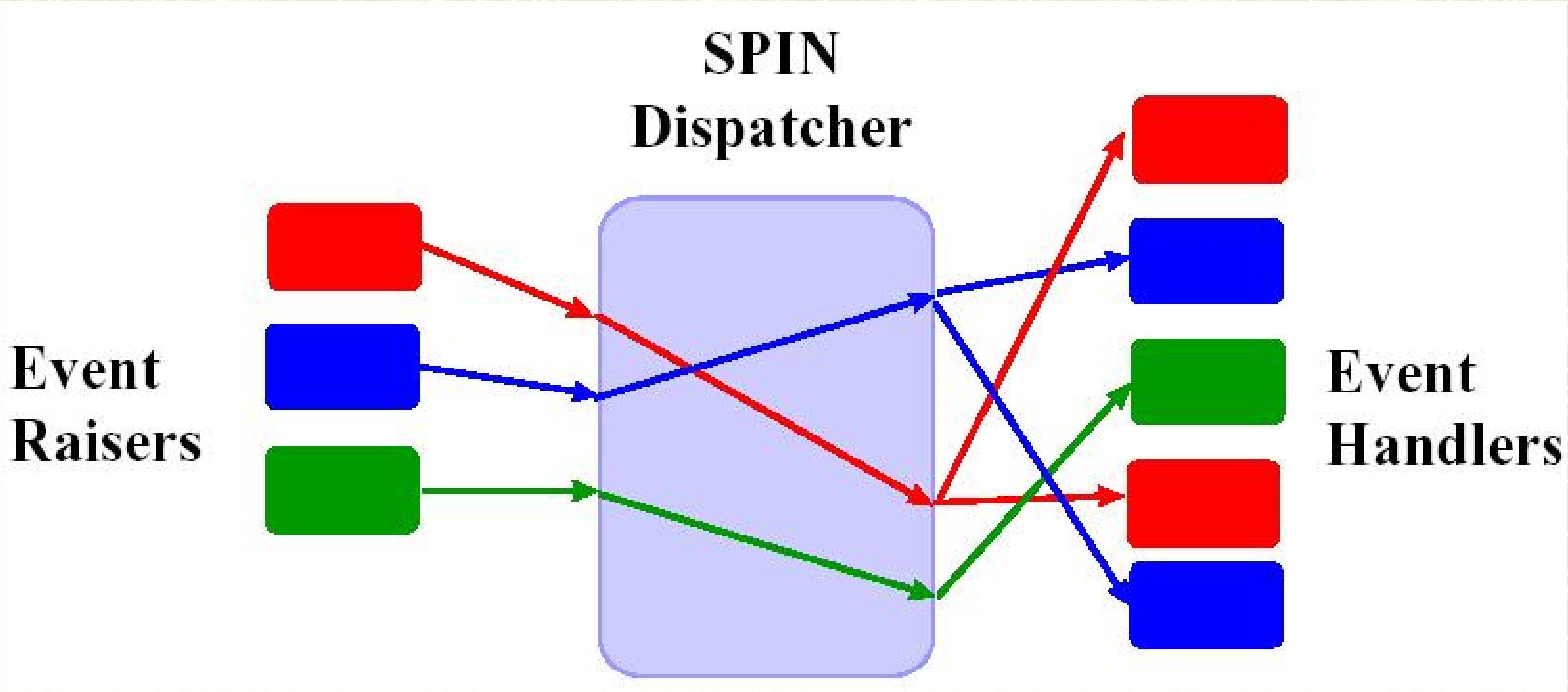


- Combine



Extensibility

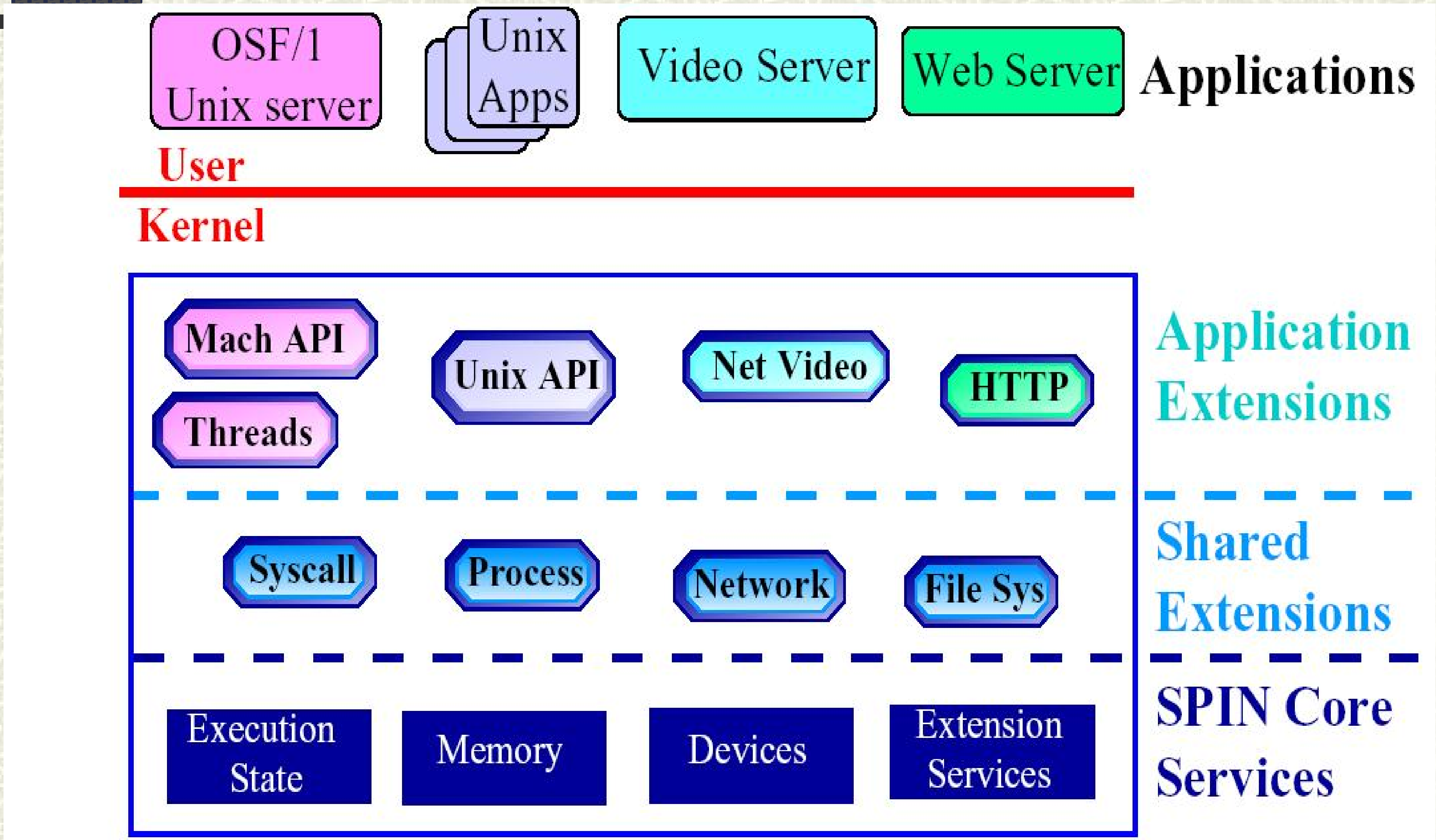
- Extension = event + event handler(s)



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

SPIN Structure



Extensible memory management

- Components
  - Physical storage -- physical address service
  - Naming -- virtual address service
  - Translation -- translation service
- Do not define an address space model directly

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 extensions

System performance

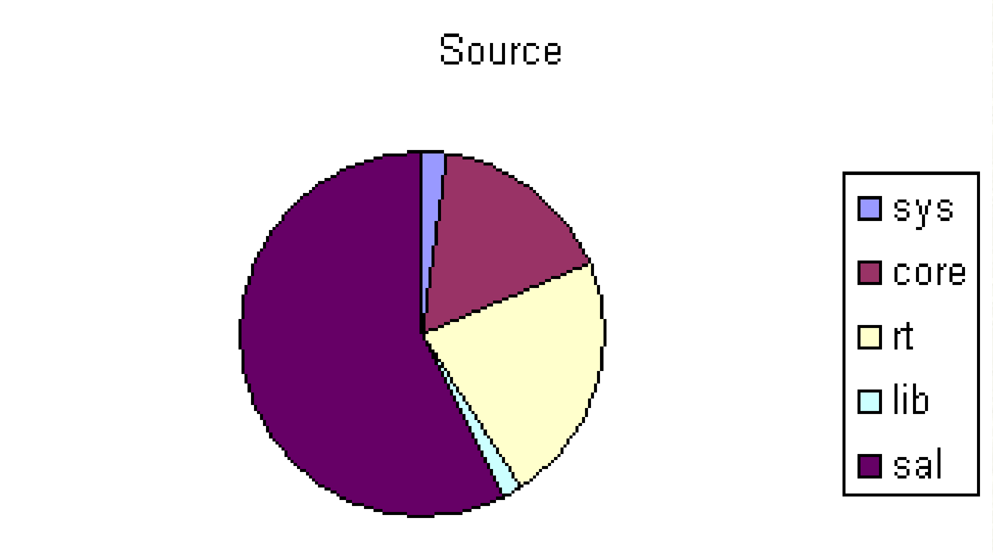
- System Size
  - The size of the system in terms of lines of code and object size
- Microbenchmarks
  - Measurements of low-level system services
- Networking
  - Measurements of a suit of networking protocols
- End-to-end performance

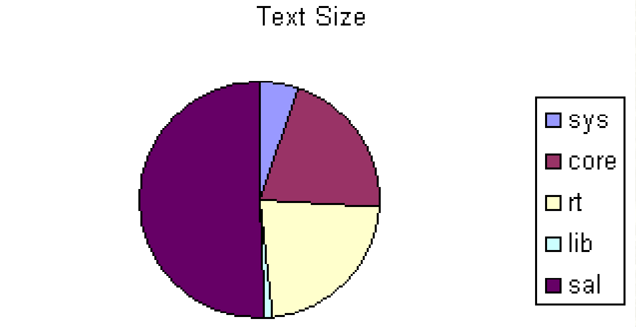
System Components

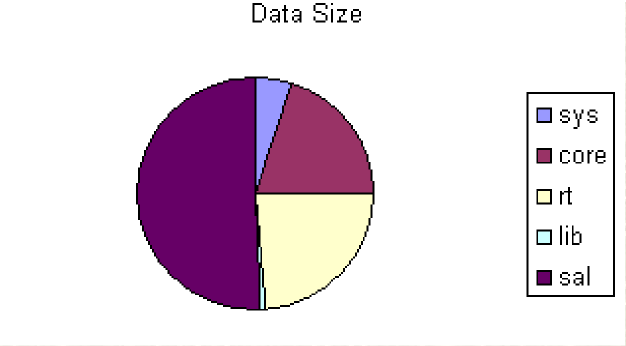
- Five main components of SPIN
  - Sys
  - Core
  - Rt
  - Lib
  - Sal

Component	Source size		Text size		Data size	
	lines	%	bytes	%	bytes	%
sys	1646	2.5	42182	5.2	22397	5.0
core	10866	16.5	170380	21.0	89586	20.0
rt	14216	21.7	176171	21.8	104738	23.4
lib	1234	1.9	10752	1.3	3294	.8
sal	37690	57.4	411065	50.7	227259	50.8
Total kernel	65652	100	810550	100	447274	100

Table 1: *This table shows the size of different components of the system. The sys, core and rt components contain the interfaces visible to extensions. The column labeled "lines" does not include comments. We use the DEC SRC Modula-3 compiler, release 3.5.*


Source
sys
core
rt
lib
sal


Text Size
sys
core
rt
lib
sal


Data Size
sys
core
rt
lib
sal

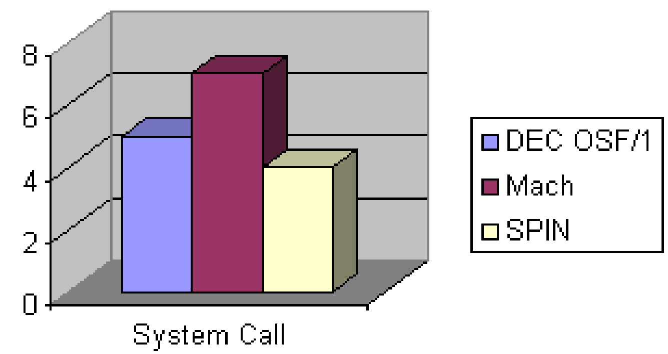
Microbenchmarks

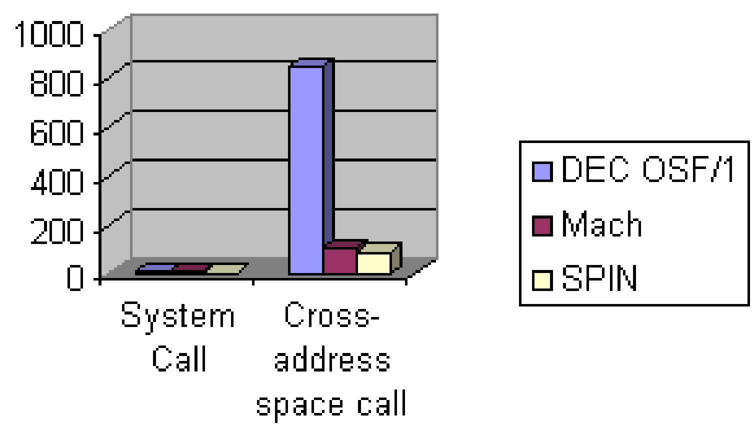
- Protected Communication
  - System calls
  - Cross-address space calls
- Thread Management
  - Fork-Join
  - Ping-Pong
- Virtual memory
  - The overhead of handling page faults in applications

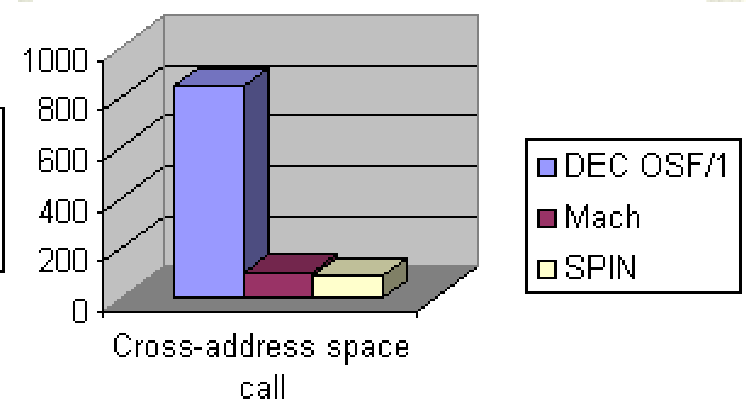
Protected Communications

Operation	DEC OSF/1	Mach	*SPIN*
Protected in-kernel call	n/a	n/a	.13
System call	5	7	4
Cross-address space call	845	104	89

Table 2: *Protected communication overhead in microseconds. Neither DEC OSF/1 nor Mach support protected in-kernel communication.*



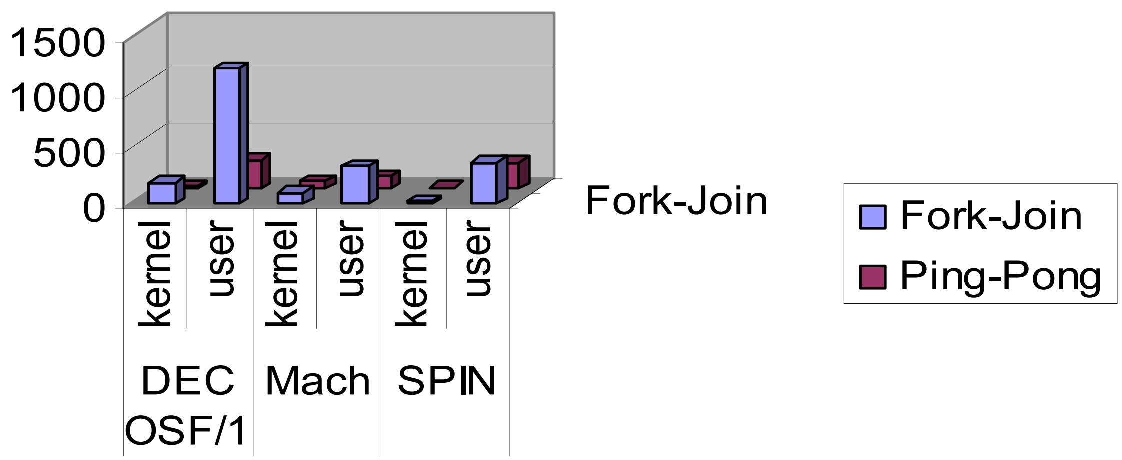




Thread Management

Operation	DEC OSF/1 kernel	DEC OSF/1 user	Mach kernel	Mach user	SPIN kernel	SPIN user layered	SPIN user integrated
Fork-Join	198	1230	101	338	22	262	111
Ping-Pong	21	264	71	115	17	159	85

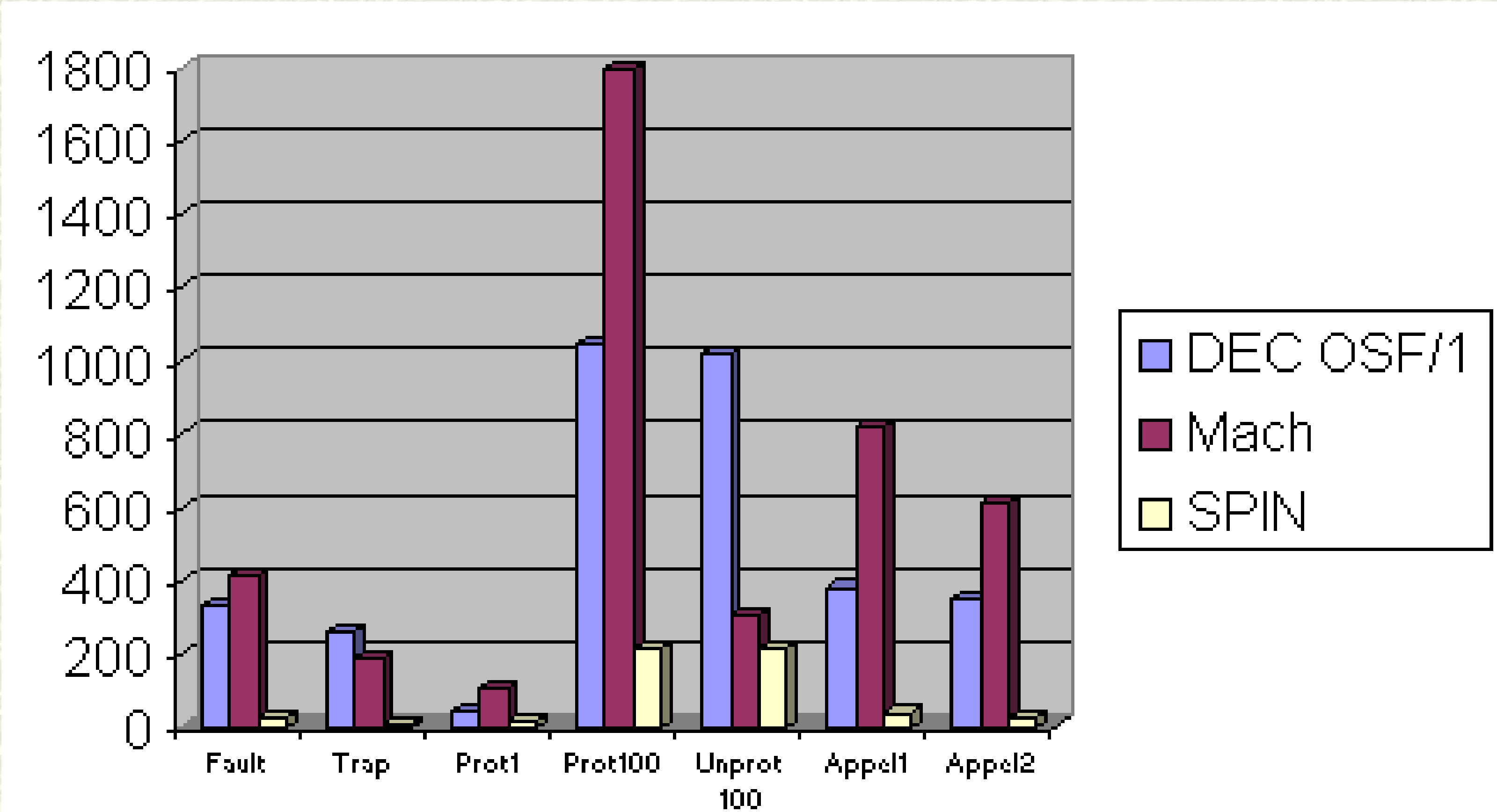
Table 3: *Thread management overhead in microseconds.*



Virtual Memory

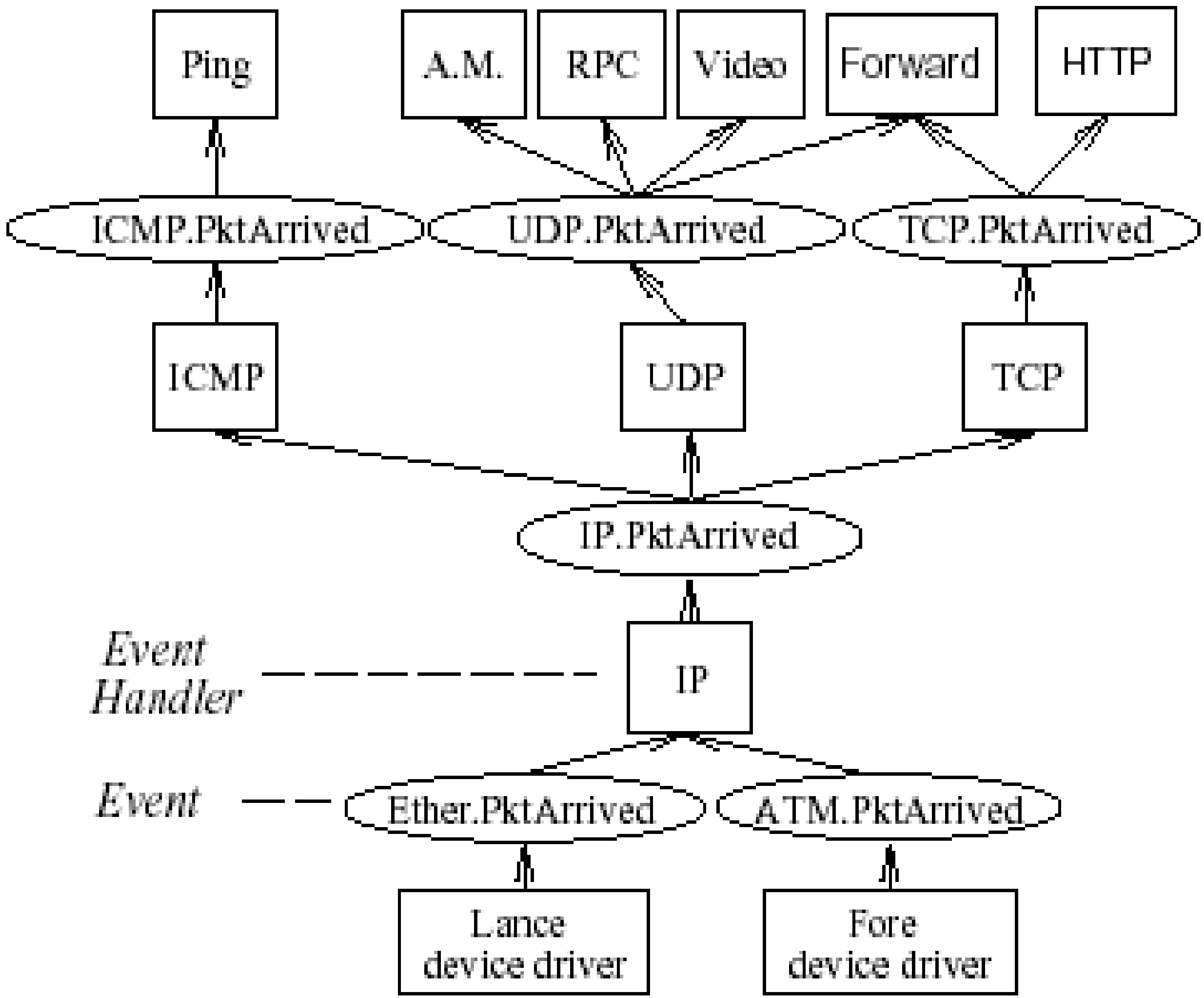
Operation	DEC OSF/1	Mach	*SPIN*
Dirty	n/a	n/a	2
Fault	329	415	29
Trap	260	185	7
Prot1	45	106	16
Prot100	1041	1792	213
Unprot100	1016	302	214
Appel1	382	819	39
Appel2	351	608	29

Table 4: *Virtual memory operation overheads in microseconds. Neither DEC OSF/1 nor Mach provide an interface for querying the internal state of a page frame.*


1800
1600
1400
1200
1000
800
600
400
200
0
Fault
Trap
Prot1
Prot100
Unprot 100
Appel1
Appel2
DEC OSF/1
Mach
SPIN

Networking

- A set of network protocol stacks are implemented using SPIN's extension architecture
  - Ethernet
  - ATM
- SPIN permits user code to be dynamically placed within the stack


Ping
A.M.
RPC
Video
Forward
HTTP
ICMP.PktArrived
UDP.PktArrived
TCP.PktArrived
ICMP
UDP
TCP
IP.PktArrived
Event Handler
IP
Event
Ether.PktArrived
ATM.PktArrived
Lance device driver
Fore device driver

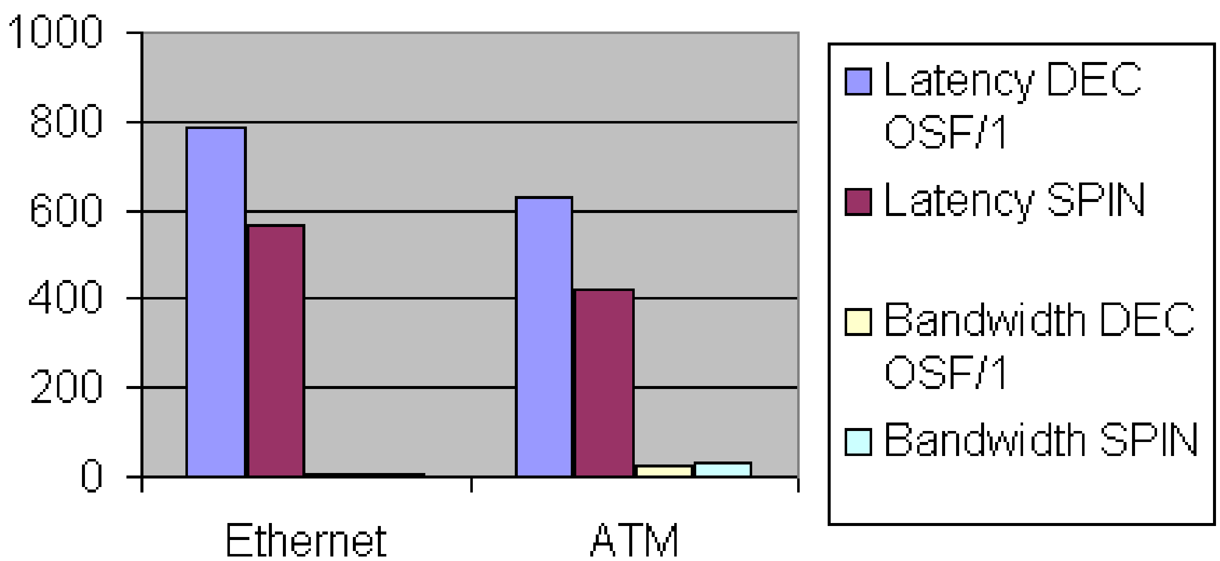
Latency and Bandwidth

- For DEC OSF/1
  - Application code executes at user level
  - Each packet processed involves a trap and several copy operations
- For SPIN
  - Application code executes as an extension in the kernel
  - Low-latency access to both the device and data

Latency and Bandwidth

	Latency		Bandwidth	
	DEC OSF/1	*SPIN*	DEC OSF/1	*SPIN*
Ethernet	789	565	8.9	8.9
ATM	631	421	27.9	33

Table 5: *Network protocol latency in microseconds and receive bandwidth in Mb/sec. We measure latency using small packets (16 bytes), and bandwidth using large packets (1500 for Ethernet and 8132 for ATM).*



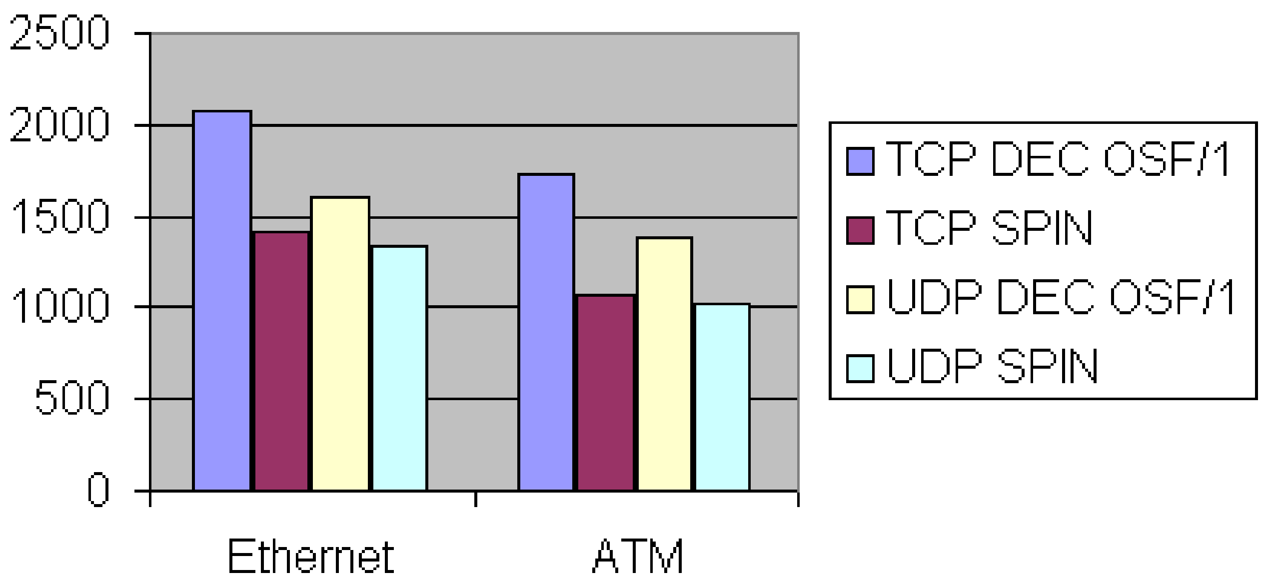
Protocol Forwarding

- SPIN's extension architecture can be used to provide protocol functionality not generally available in conventional systems
  - TCP redirection protocols can be defined by an application as a SPIN extension
  - It will be more difficult to implement the similar protocol on traditional operating systems using user-level service

Protocol Forwarding

	TCP		UDP	
	DEC OSF/1	*SPIN*	DEC OSF/1	*SPIN*
Ethernet	2080	1420	1607	1344
ATM	1730	1067	1389	1024

Table 6: *Round trip latency in microseconds to route 16 byte packets through a protocol forwarder.*



End-to-end performance

- The SPIN's extension architecture provides satisfying end-to-end performance
  - In a networked video system, SPIN's server can support a larger number of clients.
  - SPIN web server implements its own hybrid caching policy based on file type:
    - LRU for small files
    - No-cache for large files

End-to-end performance

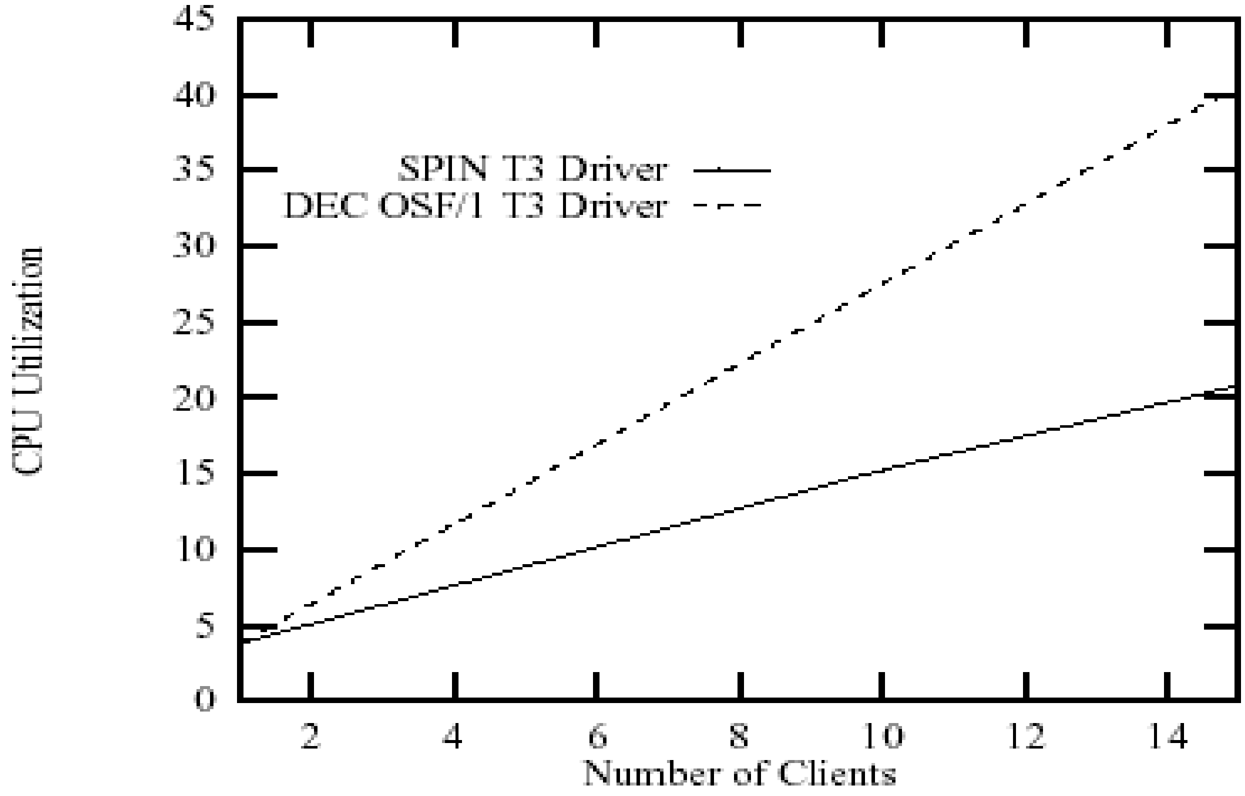


Figure 6: *Server utilization as a function of the number of client video streams. Each stream requires approximately 3 Mb/sec.*

Some Other Issues (1)

- The latency of the dispatcher is critical
  - For each guard/handler pair installed on an event, the dispatcher evaluates the guard and invoke the handler if needed
  - The overhead of an event dispatcher is linear with the number of guards and handlers installed
  - Guard-specific optimizations can be made for SPIN

Some Other Issues (2)

- Impact of automatic storage management
  - SPIN uses a trace-based, mostly-copying, garbage collector to safely reclaim memory resources.

Conclusion

- It is possible to achieve good performance in an extensible system without compromising safety
- Extensions can be dynamically defined and accessed at the granularity of a procedure call

Questions

- Does SPIN support extensibility, safety, and performance, three key features, together?
- Any other features important to extensible operating systems?