Extensible Kernels: Exokernel and SPIN

Presented by Hakim Weatherspoon

(Based on slides from Edgar Velázquez-Armendáriz and Ken Birman)

Traditional OS services – Management and Protection

- Provides a set of abstractions
  - Processes, Threads, Virtual Memory, Files, IPC
  - Sys calls and APIs (eg: Win32, POSIX)
- Resource Allocation and Management
- Accounting
- Protection and Security
  - Concurrent execution

Context for these papers

- Researchers (mostly) were doing special purpose O/S hacks
- Commercial market complaining that O/S imposed big overheads on them
- O/S research community began to ask what the best way to facilitate customization might be. In the spirit of the Flux OS toolkit…

Problems (examples coming-up)

- Extensibility
  - Abstractions overly general
  - Apps cannot dictate management
  - Implementations are fixed
- Performance
  - Crossing over into the kernel is expensive
  - Generalizations and hiding information affect performance
- Protection and Management offered with loss in Extensibility and Performance

Need for Application controlled management (examples)

- Buffer Pool Management In DBs (*)
  - LRU, prefetch (locality Vs suggestion), flush (commit)

- Shared Virtual Memory (+)
  - use a page fault to retrieve page from disk / another processor

Examples (cont.)

- Concurrent Checkpointing (+)
  - Overlap checkpointing and program being checkpointed
  - Change rights to R-only on dirty pages
  - Copy each page and reset rights
  - Allow reads; Use write faults to {copy, reset rights, restart}

* OS Support for Database Management (Stonebraker)

+ Virtual Memory Primitives for User Programs (Andrew W. Appel and Kai Li)

Examples (cont.)

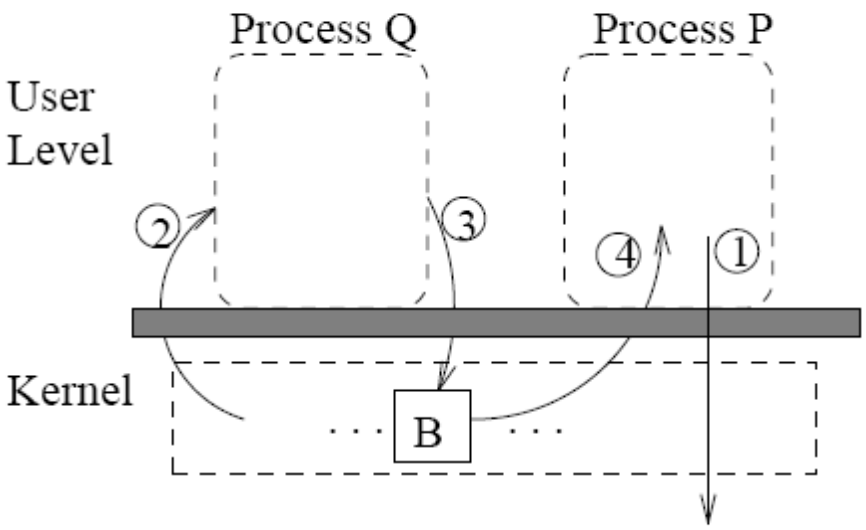


Figure 1: Interaction between kernel and user processes in two-level replacement: (1) P misses; (2) kernel consults Q for replacement; (3) Q decides to give up page B; (4) kernel reallocates B to P.

Feedback for file cache block replacement

[Implementation and Performance of Application-Controlled File Caching – Pei Cao, et al.]

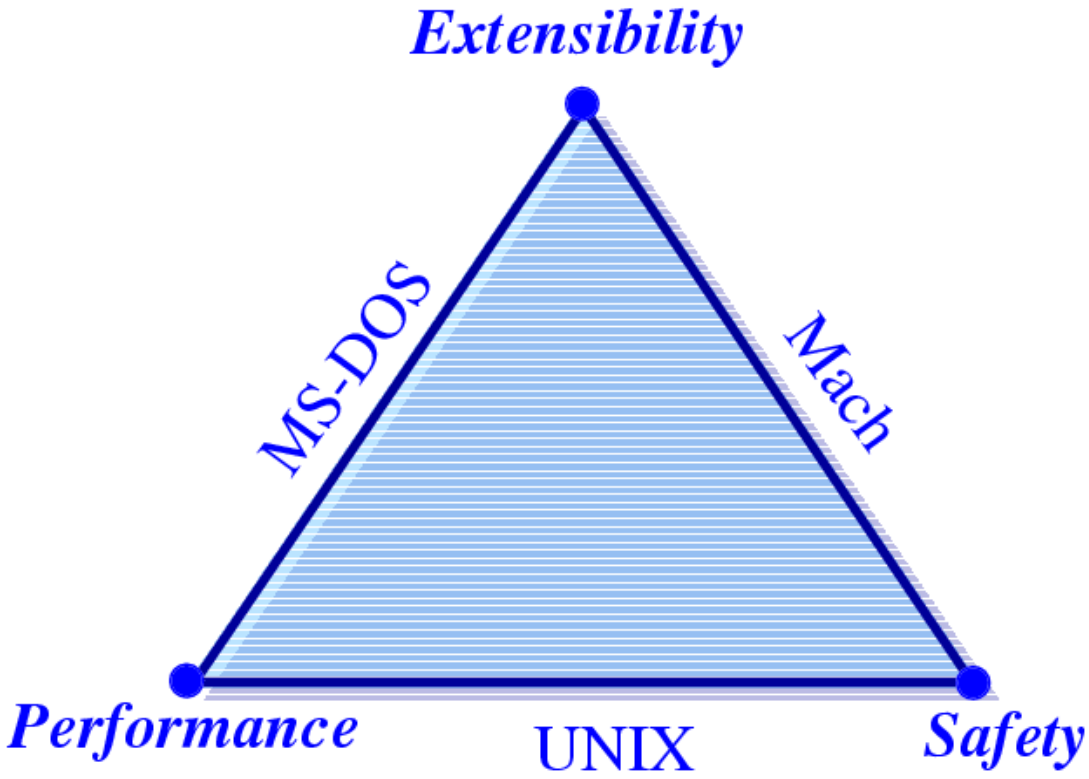
Down with monarchy!

French Revolution - Execution of Louis XVI

Challenges

- Extensibility
- Security
- Performance

Can we have all 3
in a single OS?



From Stefan Savage's SOSP 95 presentation

Extensible Kernels

- Exokernel (SOSP 1995): safely exports machine resources
  - Kernel only multiplexes hardware resources (Aegis)
  - Higher-level abstractions in Library OS (ExOS)
  - Secure binding, Visible resource revocation, Abort
  - Apps link with the LibOS of their choice
- SPIN (SOSP 1995): kernel extensions (imported) safely specialize OS services
  - Extensions dynamically linked into OS kernel
  - Safety ensured by Programming Language facilities

Notice difference in pt. of view

- Exokernel assumes that very significant extensions to the kernel are needed in many settings and that home-brew kernels may remain common for long into the future
  - Goal is to enable this sort of work while reducing risk that developer will trash the file system, debugging tools, etc
- SPIN is more focused on protecting standard OS against a device driver run amok.
  - Sees this as the more common need…

Exokernel

- Dawson R. Engler, M. Frans Kaashoek and James O'Toole Jr.
- Engler's Master's Thesis.
- Follow-up publications on 1997 and 2002.
- Kaashoek later worked on Corey.

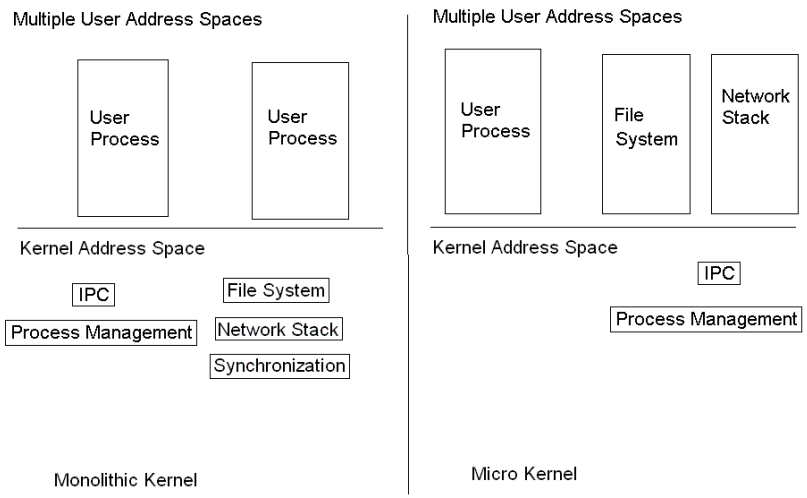
Exokernels - Motivation

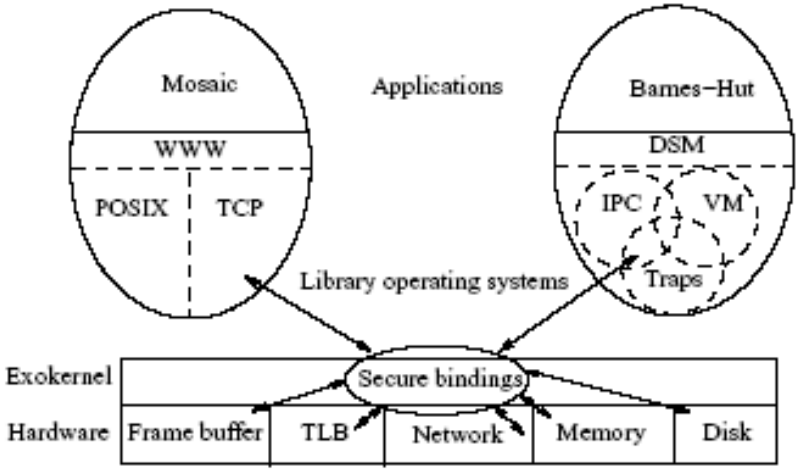
- Existing Systems offer fixed high-level abstractions which is bad
  - Hurt app performance (generalization – eg: LRU)
  - Hide information (eg: page fault)
  - Limit functionality (infrequent changes – cool ideas don’t make it through)

Motivation (cont.)

- Separate protection from management, mgmt in user space
- Apps should use domain specific knowledge to influence OS services
- Small and simple kernel – adaptable and maintainable

OS Component Layout





Exokernel

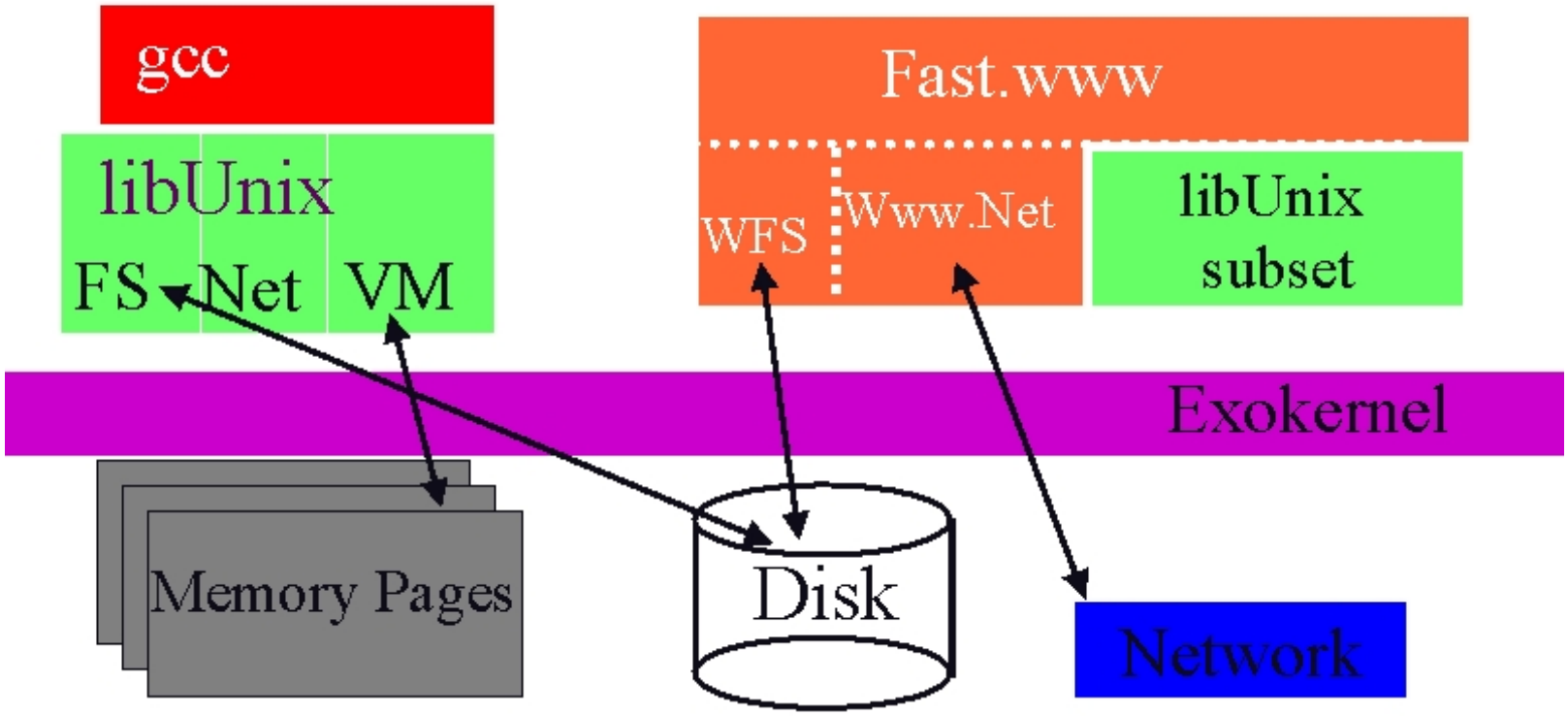
Exokernel main ideas

- Kernel
  - Resource sharing, not policies
- Library Operating System
  - Responsible for the abstractions
    - IPC
    - VM
    - Scheduling
    - Networking

Lib OS and the Exokernel

- Lib OS (untrusted) can implement traditional OS abstractions (compatibility)
- Efficient (Lib OS in user space)
- Apps link with Lib OS of their choice
- Kernel allows LibOS to manage resources, protects LibOss

Exokernel Architecture



Exokernels. MIT CSAIL, 1998

Exokernel vs Microkenels vs VM

- Exokernel defines only a low-level interface.
- A microkernel also runs almost everything on user-level, but has fixed abstractions.
- A VM emulates the whole machine, doesn’t provide direct access.

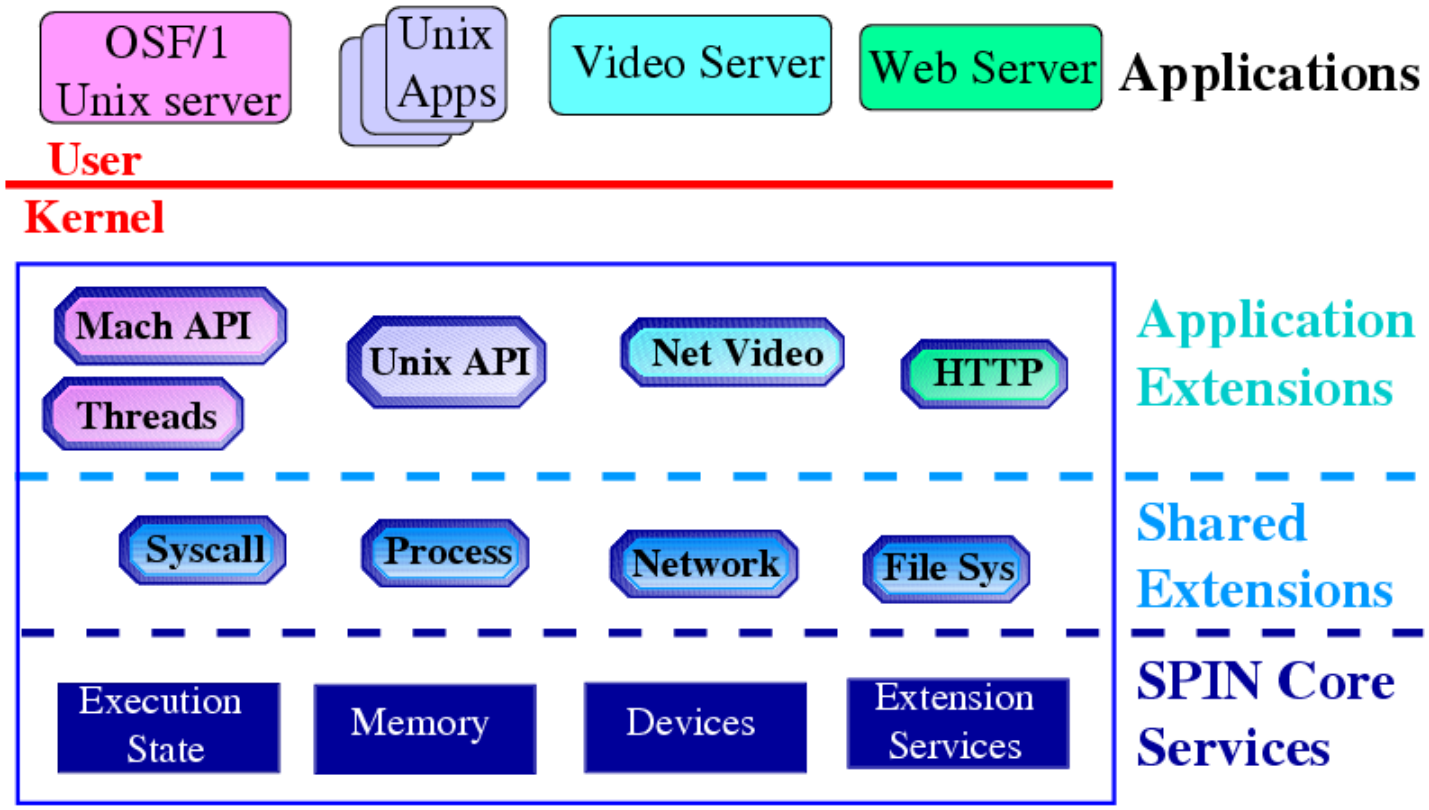
SPIN

- University of Washington.
- Brian N. Bershad, Stefan Savage, Emin Gun Sirer, Marc E. Fiuczynski, David Becker, Craig Chambers, Susan Eggers
- Main ideas continue on Singularity, a C# system by MSR and U.W.

SPIN

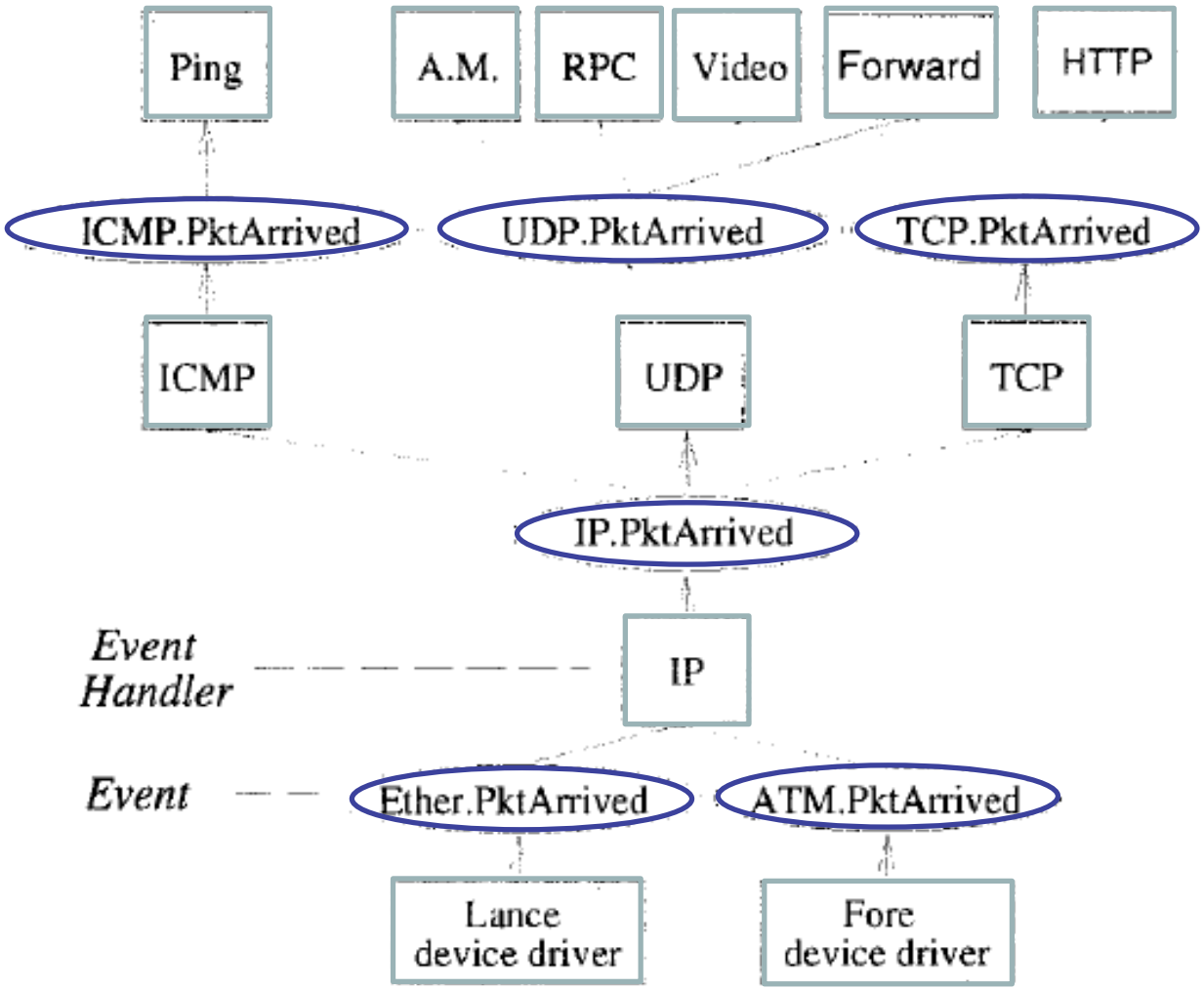
- Use of language features for Extensions
  - Extensibility
    - Dynamic linking and binding of extensions
  - Safety
    - Interfaces. Type safety. Extensions verified by compiler
  - Performance
    - Extensions not interpreted; Run in kernel space

SPIN structure



From Stefan Savage's SOSP 95 presentation

SPIN Architecture



SPIN main ideas

- Extend the kernel at runtime through statically-checked extensions.
- System and extensions written in Modula-3.
- Event/handler abstraction

Language: Modula 3

- Interfaces
- Type safety
- Array bounds checking
- Storage Management

- Threads
- Exceptions

SPIN vs Exokernel

- SPIN uses programming language facilities and communicates through procedure calls.
- Uses hardware specific calls to protect without further specification.

Agenda

- Overview
- Design
  - Protection
  - Memory
  - Sharing
  - Scheduling
  - network
- Implementations

Exokernel design

- Securely expose hardware
  - Decouple authorization from usage
- Expose allocation
- Expose names
  - Raw access to hardware features
- Expose revocation
  - “Polite” and then forcibly abort
  - Repossession

SPIN design

- Co-location
  - Same memory-space as kernel
- Enforces modularity
- Local protection domains
  - Resolves at link time
- Dynamic call binding
  - Event handler pattern.

Protection model

- Capabilities
  - Immutable references to resources
- Protection domains
  - Names accessible at an execution context
  - Provided by the language
  - Linking through Resolve and Combine

Exokernel Memory

- Guard TLB loads and DMA
- Large Software TLB
- Library Operating System handles page faults if it’s allowed to.

SPIN Memory

- The kernel controls allocation of physical and virtual addresses capabilities.
- Extension react to page faults and error through handlers.

Exokernel processor sharing

- Round robin allocation of slices.
- Library operating system responsible for context switching.
  - Can implement own scheduling policy: donate timeslice
- It the time a process takes is excessive, it is killed.

SPIN processor sharing

- Based on Modula-3 threads.
- Organized in *strands.*
- Communicates through Block, Unblock, Checkpoint and Resume events.
- Preemptive round-robin schedule of strands

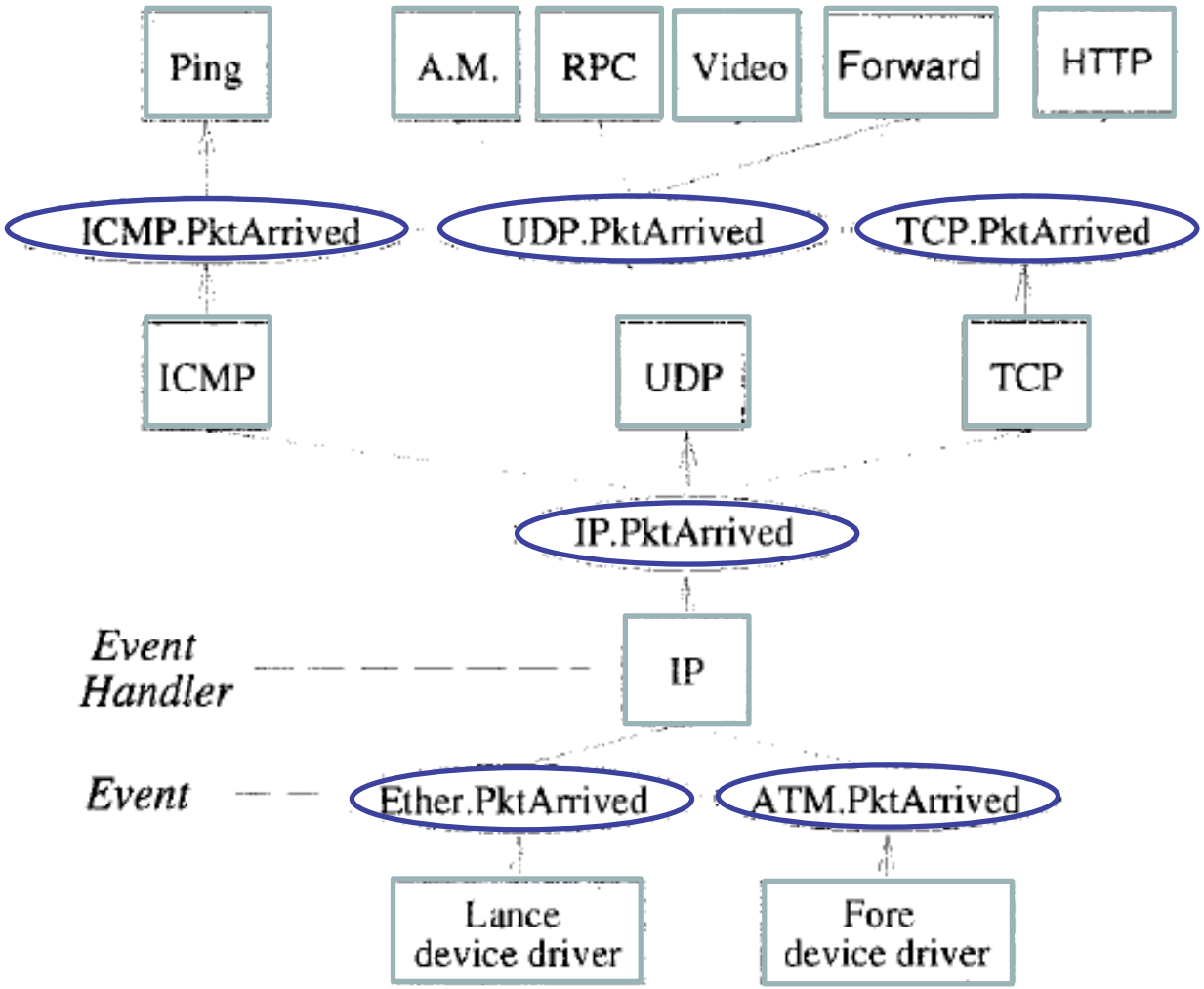
Exokernel Network

- Downloadable filters
- Application-specific Safe Handlers
- Respond directly to traffic

SPIN Network

- Protocol stack.
- Packet pulled by handlers.

SPIN Network



Agenda

- Overview
- Design
- Implementations

Exokernel

- DEC MIPS
- Aegis: actual exokernel
  - Processor
  - Physical memory
  - TLB
  - Exceptions, Interrupts
- ExOS: library operating system
  - Processes, IPC, Virtual Memory, Network protocols

Microbenchmark results

Machine	OS	pipe	pipe'	shm	lrpc
DEC2100	Ultrix	326.0	n/a	187.0	n/a
DEC2100	ExOS	30.9	24.8	12.4	13.9
DEC3100	Ultrix	243.0	n/a	139.0	n/a
DEC3100	ExOS	22.6	18.6	9.3	10.4
DEC5000	Ultrix	199.0	n/a	118.0	n/a
DEC5000	ExOS	14.2	10.7	5.7	6.3

Machine	OS	Roundtrip latency
DEC5000/125	ExOS/ASH	259
DEC5000/125	ExOS	320
DEC5000/125	Ultrix	3400
DEC5000/200	Ultrix/FRPC	340

ExOS Virtual Memory

Machine	OS	dirty	prot1	prot100	unprot100	trap	appel1	appel2
DEC2100	Ultrix	n/a	51.6	175.0	175.0	240.0	383.0	335.0
DEC2100	ExOS	17.5	32.5	213.0	275.0	13.9	74.4	45.9
DEC3100	Ultrix	n/a	39.0	133.0	133.0	185.0	302.0	267.0
DEC3100	ExOS	13.1	24.4	156.0	206.0	10.1	55.0	34.0
DEC5000	Ultrix	n/a	32.0	102.0	102.0	161.0	262.0	232.0
DEC5000	ExOS	9.8	16.9	109.0	143.0	4.8	34.0	22.0

+ Fast Sys call.
- Half the time in look-up (vector).

Repeated access to Aegis STLB and ExOS PageTable

SPIN

- DEC Alpha
- System components
  - Sys
  - Core
  - Rt
  - Lib
  - Sal (device drivers)

SPIN Microbenchmarks Results

IPC

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

Sockets, SUN RPC

Mesgs.

In-kernel Call

Thread Mgmt

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

All numbers are in microseconds

SPIN Microbenchmark Results

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

In-Kernel calls are more efficient than traps or messages

All numbers are in microseconds

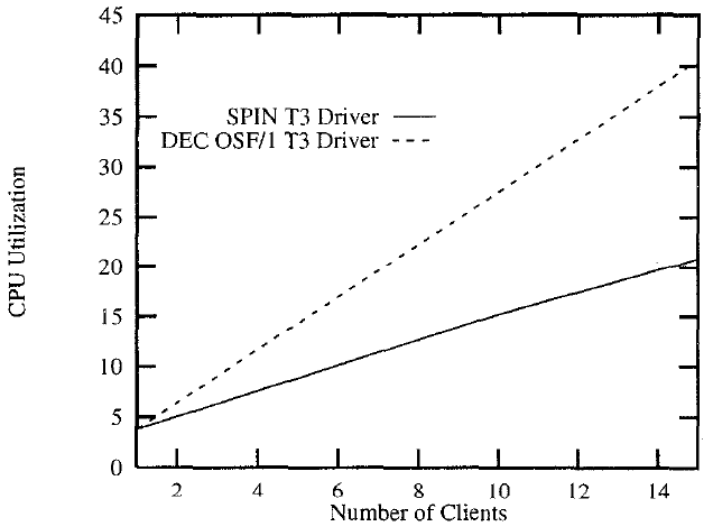
Performance: Networking

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

Lower RTT because of in-kernel extension

time in microseconds, Bandwidth in Mbps

End-to-End Performance



Networked Video

Server CPU utilization
(network interface supports DMA)

Issues

- Dispatcher scalability
- Handler scheduling
- Garbage collection

Perspective

- Extensible kernels are actually fast.
- End-to-end arguments.
- Efficient implementations.
- High level languages are not terrible!
- Extensibility without loss of security or performance
  - Exokernels
    - Safely export machine resources
    - Decouple protection from management
  - SPIN
    - kernel extensions (imported) safely specialize OS services
    - Safety ensured by Programming Language facilities

Next Time

- Read and write review:
  - *Disco: Running Commodity Operating Systems on Scalable Multiprocessors*, Bugnion et al. 16th SOSP, 1997
  - *Tornado: maximizing locality and concurrency in a shared memory multiprocessor operating system*, Gamsa et al. 3rd OSDI, Feb 1999.

Next Time

- Read and write review:
- Project Proposal
  - Return comments later today
- Project Survey Paper due next Thursday
- Check website for updated schedule