Extensible Kernels

Edgar Velázquez-Armendáriz

September 24th 2009

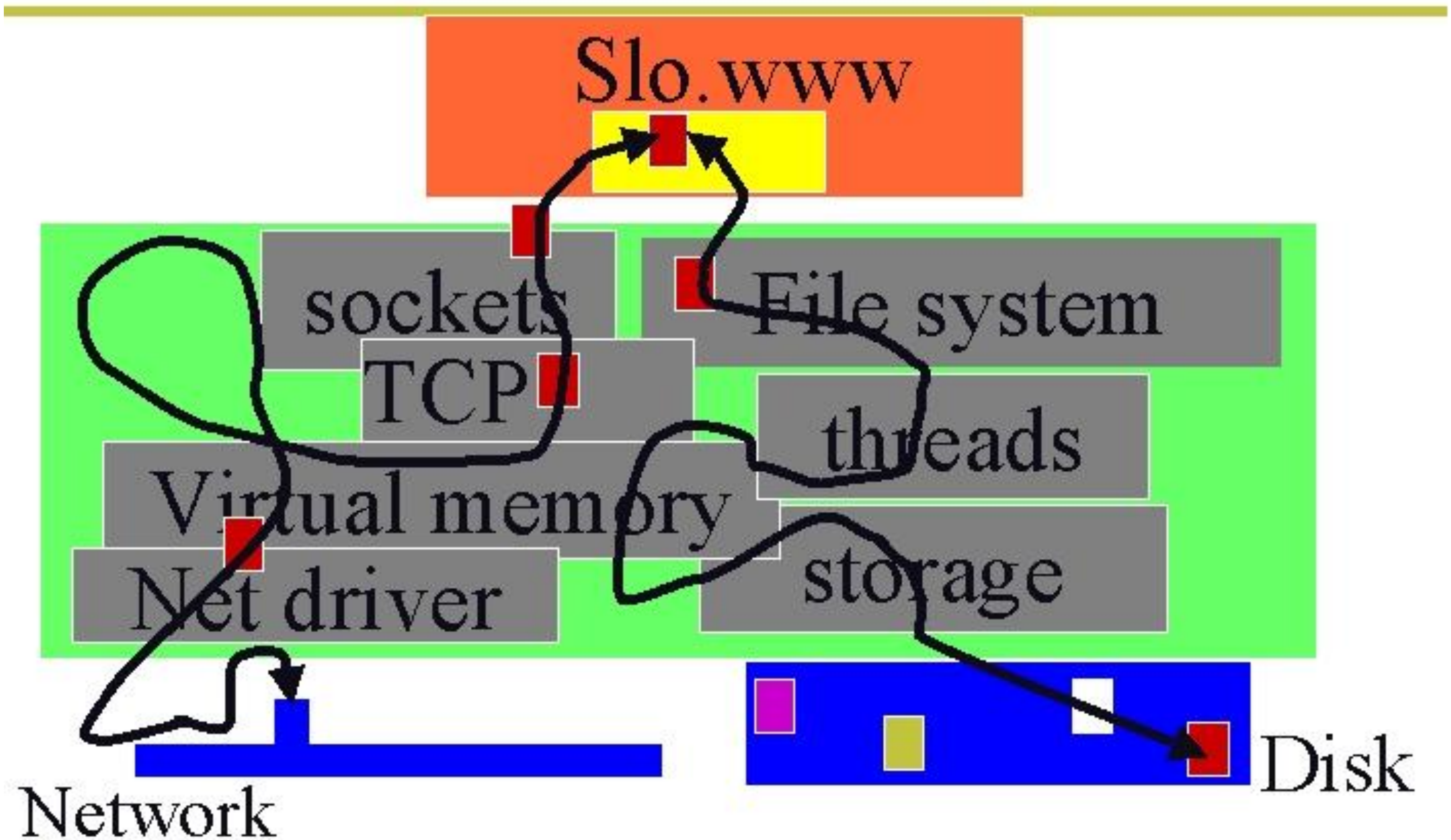
Agenda

- Exokernel: An Operating System architecture for Application-Level Resource Management
- Extensibility, Safety and Performance in the *SPIN* Operating System

Basic idea

- One size fits all… NOT!
  - Provide a better match between application and system capabilities.
- “Extreme” application of end-to-end argument.

Traditional OS Structure



Exokernels. MIT CSAIL, 1998

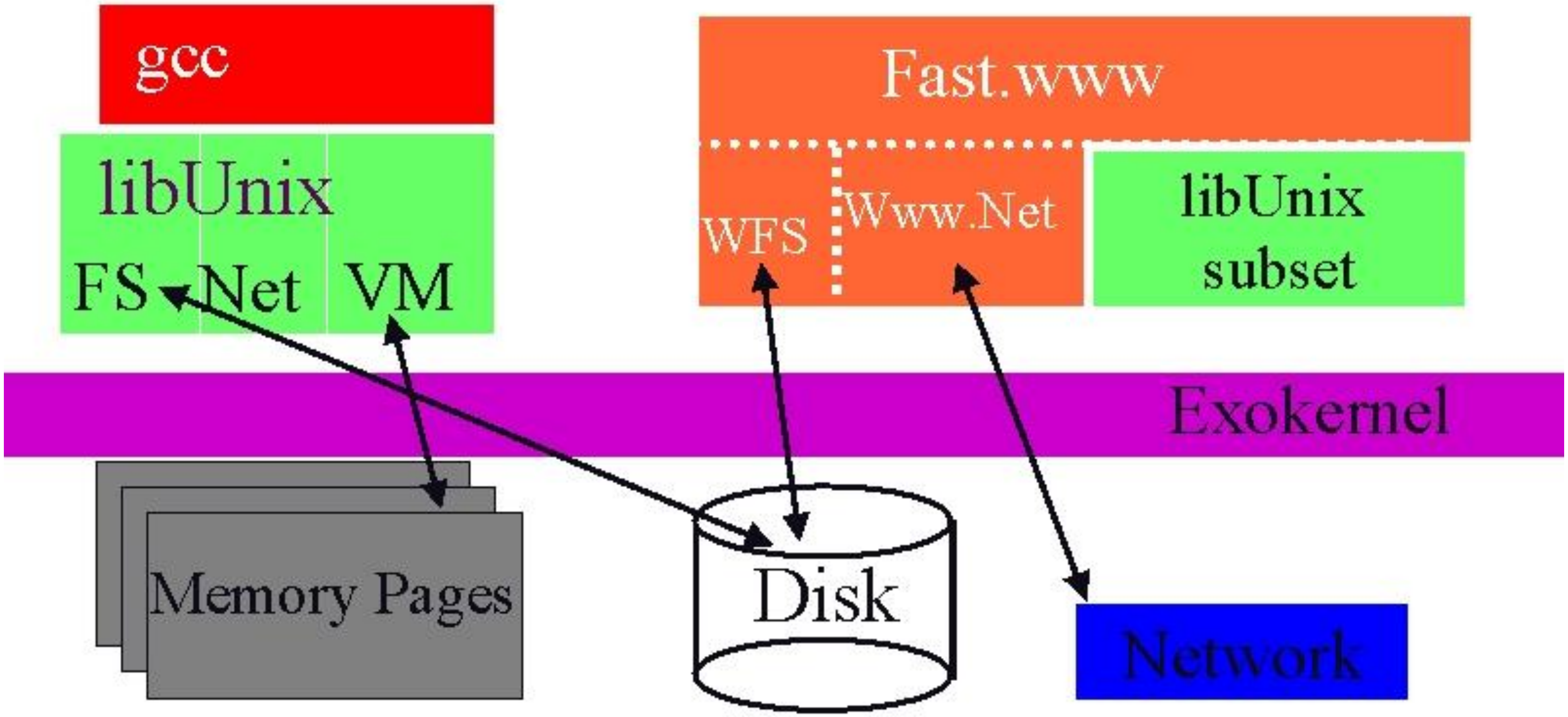
Exokernel

- 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.

Exokernel main ideas

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

Exokernel Architecture



Exokernels. MIT CSAIL, 1998

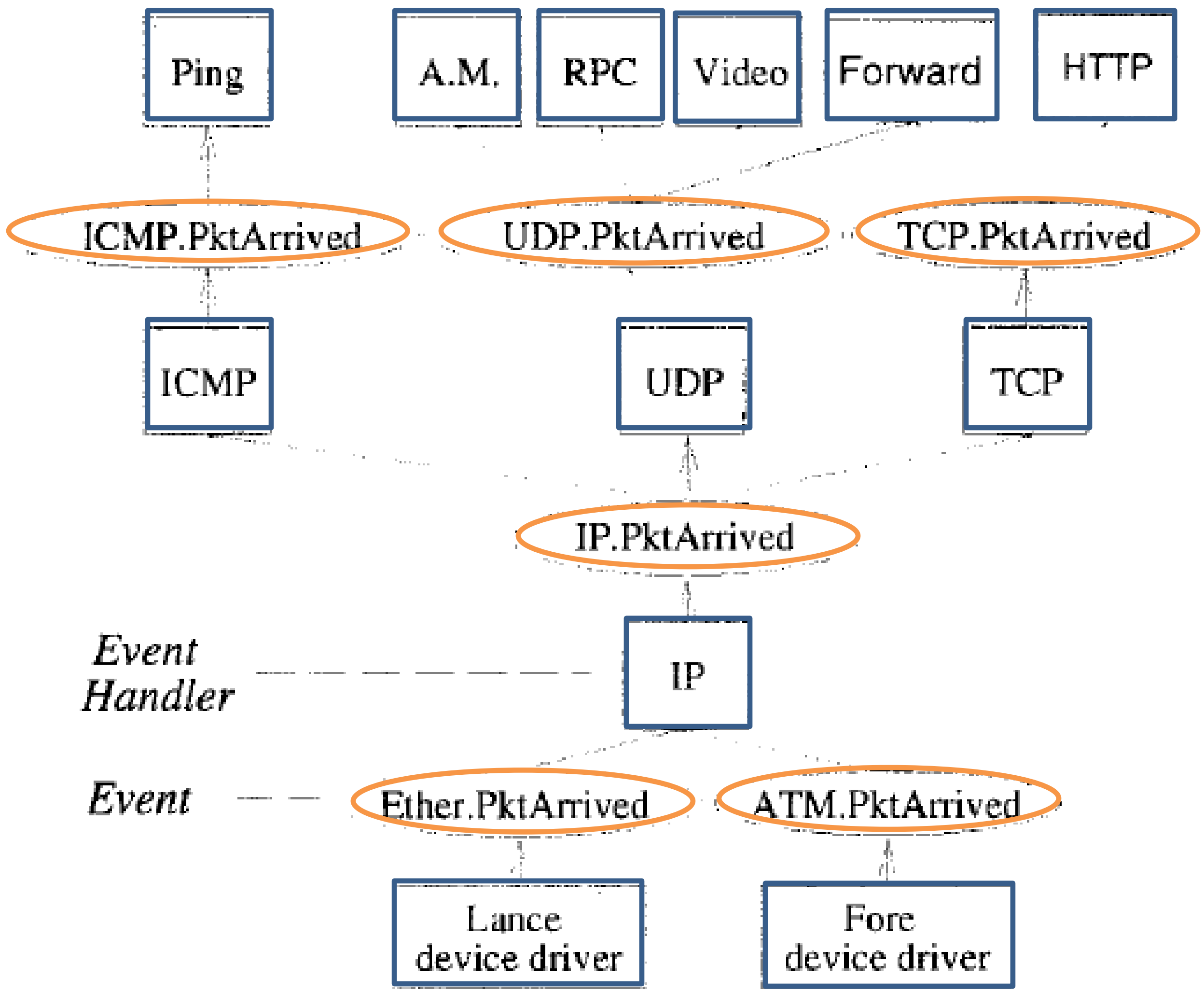
Exokernel vs Microkenels vs VM

- 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 et. al.
- Main ideas continue on Singularity, a C# system by MSR and U.W.

SPIN Architecture



SPIN main ideas

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

About Modula-3

- Interfaces
- Type safety
- Garbage collection
- Objects
- Generics
- 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
- Implementations

Exokernel design

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

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.
- 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

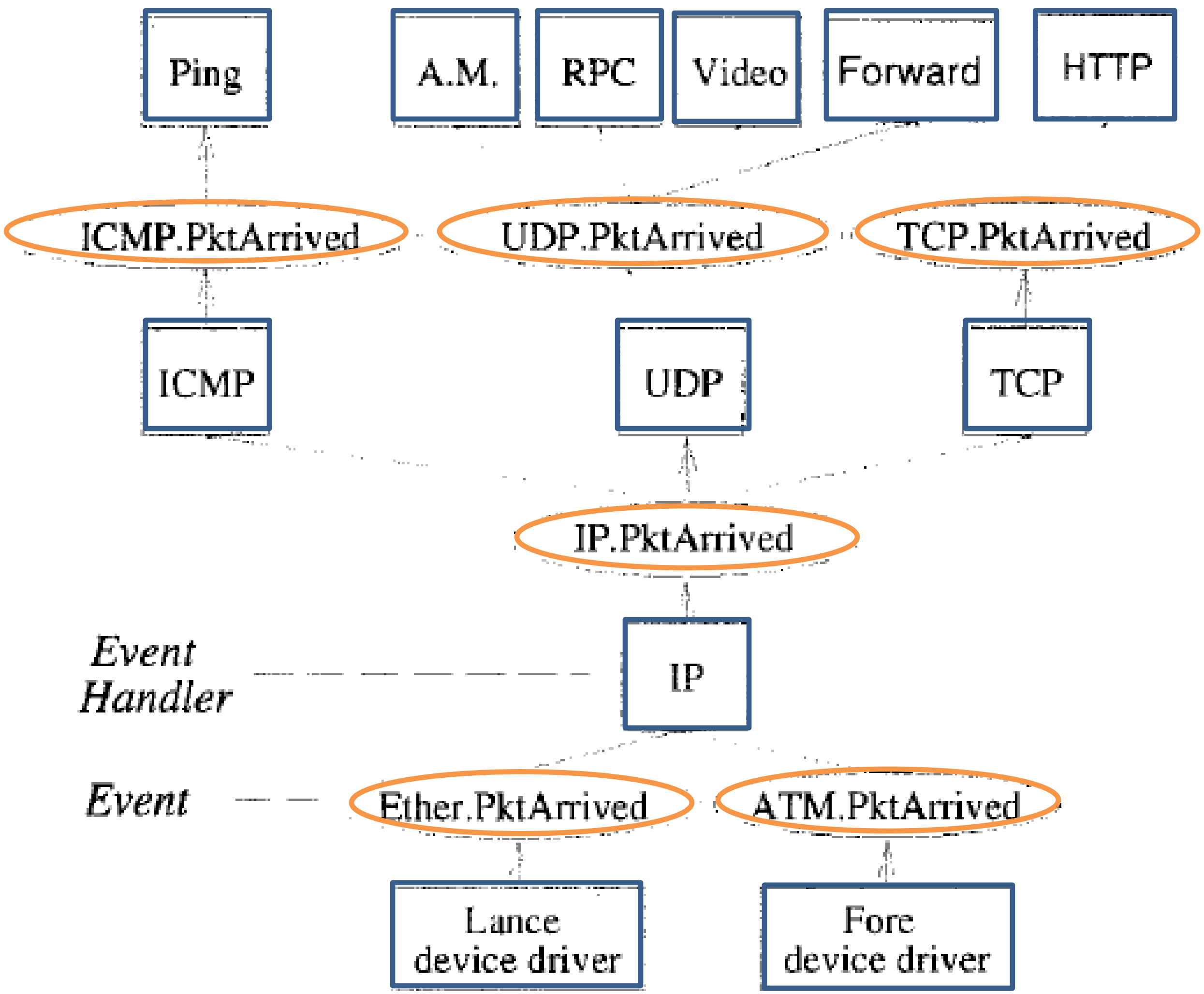
Exokernel Network

- 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, 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

SPIN

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

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

Catching up

- Extensible kernels are actually fast.
- End-to-end arguments.
- Efficient implementations.
- High level languages are not terrible!