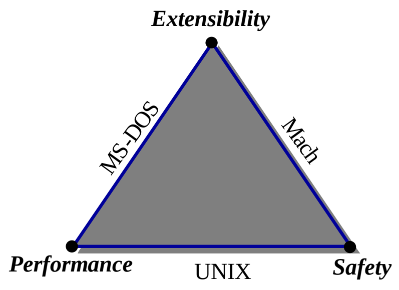
Extensibility, Safety and Performance in the SPIN Operating System

Emin Gun Sirer

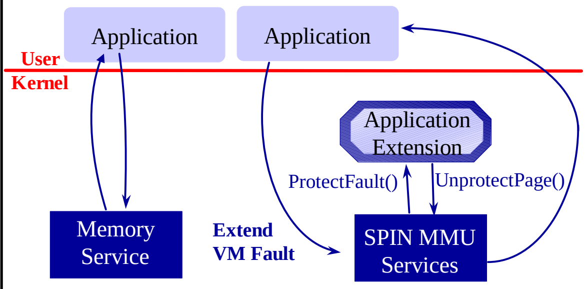
Motivation



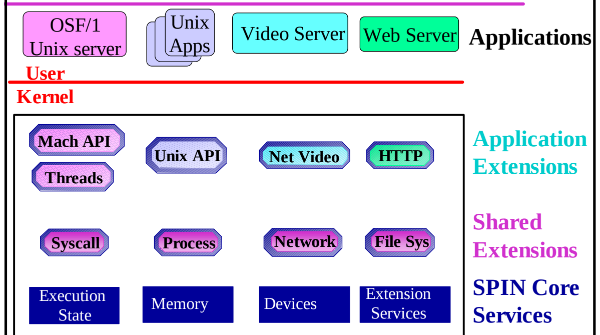
Approach

- Extensibility
  - » Allow applications to extend any service
- Performance
  - » Dynamically inject application code into the kernel
- Safety
  - » Rely on language protection for memory-safety
  - » Rely on interface design for component-safety

A SPIN extension



Structure



Safety

Language-based protection

Modula-3

- Type-safe & system-safe
- Interfaces for hiding resources
- Cheap capabilities

Typesafety vs. System safety

- Typesafety (a la Mesa, Java, et al.)
  - » Objects of type X can only be treated as X or one of its supertypes
    - Pointers are cast-checked, arrays are bound-checked, stack references are size-checked, and garbage collection is used to pick up free objects

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();
```

* Unforgeable

* Optionally opaque

* Cheap

Shortcomings of typesafety

- Typesafety is not strong enough!!
  - » Need to be able to make statements about program, not type, invariants.
    - Your module will not be left in an inconsistent state with respect to locks, updates, data values.
- Sometimes, typesafety is too restrictive!
  - » Need to be able to "bend" typesafety rules in order to avoid copying.
    - A network packet is both a bag of bytes and an object of type IP.

System safety

- Additions to M3 for system safety
  - » Abortable upcalls
    - Procedures marked EPHEMERAL can be terminated at any time. Compiler ensures that the system is left intact.
  - » Interaction with the collector
    - Objects can be pinned down when communicating with the outside world, e.g. device drivers.
  - » Unforgeable objects
    - An object may only be created by the module that defines it; rogue extensions cannot forge objects.
  - » System-safe (but not typesafe) casts
    - An object of type A can be VIEWed as an object of type B as long as the conversion would not cause program faults.

SPIN Protection Domains

- Kernel provided abstraction:
  - »*<u>Logical Protection Domains</u>*
- Handles for code management and linking
- Provide isolation within a single address space
- Named by capabilities

Operations on Domains

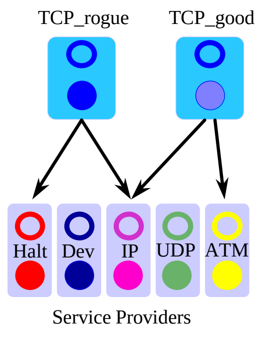
- Create
- Name
- Resolve
  - » Exercise access
- Export
  - » Share interfaces

MODULE TCP	INTERFACE IP;
… IP.Send(data);	PROCEDURE Send();

```
DIp := Domain.Create(INTERFACE(IP));
Nameserver.Register(“ip”,Dip,Auth);
```

```
DTcp := Domain.Create(ObjectFile);
DIp := Nameserver.Query(“ip”,Cred);
Domain.Resolve(DTcp, DIp);
Domain.Initialize(Dtcp);
```

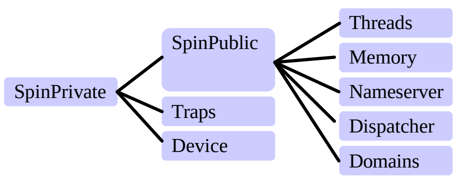
Using Domains



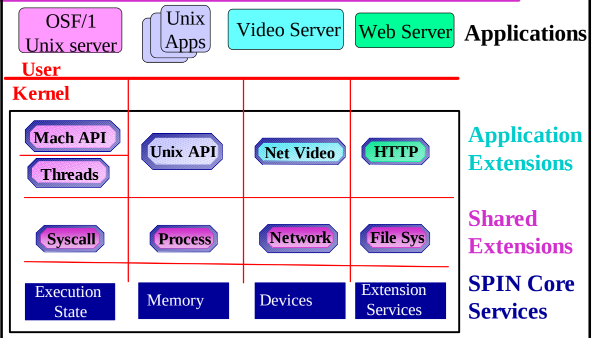
- *Resolve* symbol references to symbol definitions
- The types of the imported and exported symbol must match

Domains as Capabilities

- Domains nest to simplify capability management
- Binding code generated automatically
- Domain lookup through a nameserver



Domains & protection



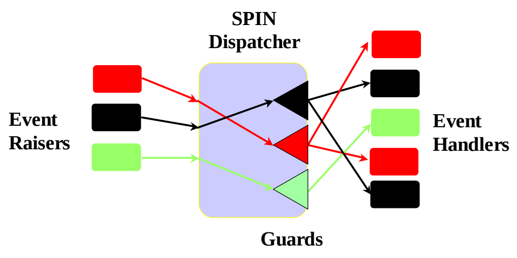
Domain Summary

- Logical protection domains within a single address space
- Complements type-safety to achieve system safety
- Sharing is cheap
  - » Share code by jumping directly
  - » Share data by passing pointers
- No runtime overhead

Extensibility

Dispatcher

Event-based communication model



Event implementation

Use procedure call to define and invoke events

- Convenient syntax
- High performance implementation for common case
- Most procedures in the system can be extended

Using Events - Defining/Raising

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

END Ethernet.
```

Event definition

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

Event raise

Using Events - Handling

```
PROCEDURE IPPacketArrivedGuard(p: Pkt)
                                   : BOOLEAN =
BEGIN
       RETURN p.ethertype = IPPacket;
END IPPacketArrivedGuard;
```

Guard

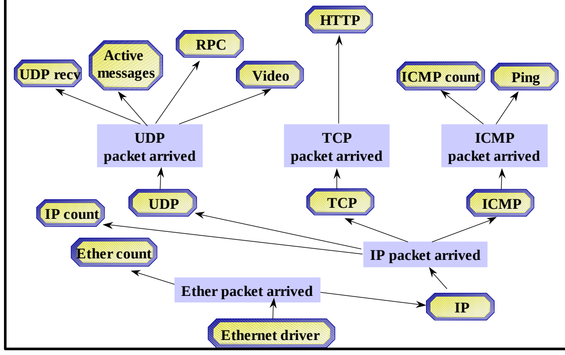
```
PROCEDURE IPPacketArrivedHandler(p: Pkt) =
  BEGIN
       (* Perform IP fragment assembly *)
  END IPPacketArrivedHandler;
```

Event handler

```
Dispatcher.Install(Ethernet.PacketArrived,
                     IPPacketArrivedGuard,
                     IPPacketArrivedHandler,
                     Credentials);
```

Installation

A protocol graph in SPIN



Design summary

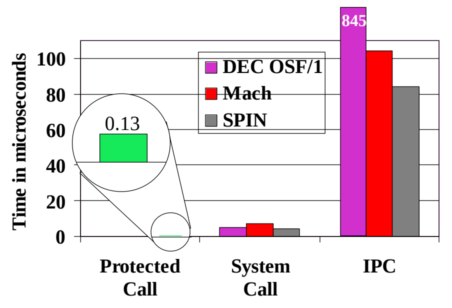
- Safety
  - Memory safe language for extensions
  - Link-time enforcement for access control
- Extensibility
  - Fast and safe centralized control transfer switch
- Result
  - Allows fast and safe fine-grained service extension

Performance

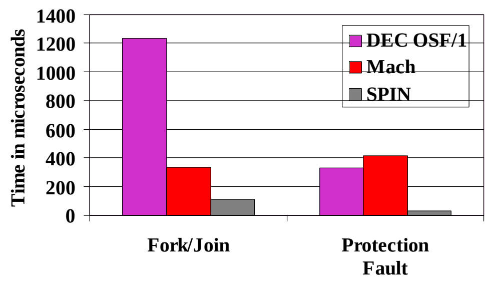
SPIN performance advantages

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

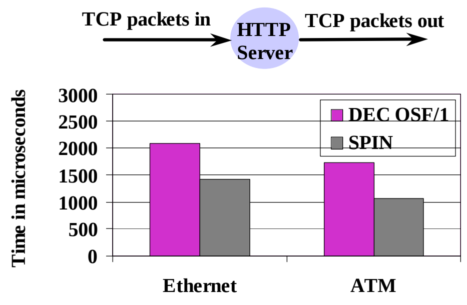
Protected communications



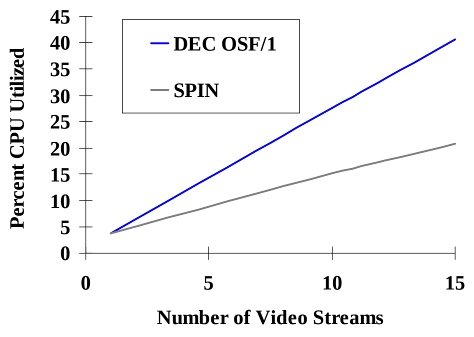
Other basic system services



Per-port TCP packet forwarding

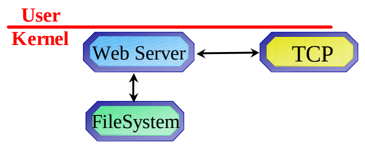


Video service



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/**



Modifications to Modula-3

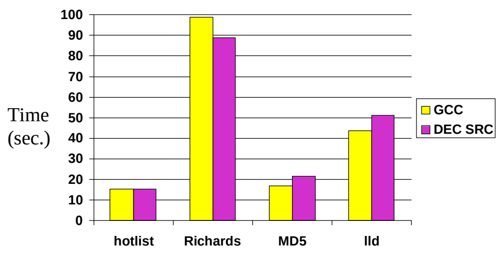
- Memory safe cast
  - VIEW operator
- Procedures which may be terminated
  - EPHEMERAL procedure type
- Naming code
  - INTERFACE UNIT, MODULE UNIT
- Universal procedure type
  - PROCANY reference type

How big are these extensions?

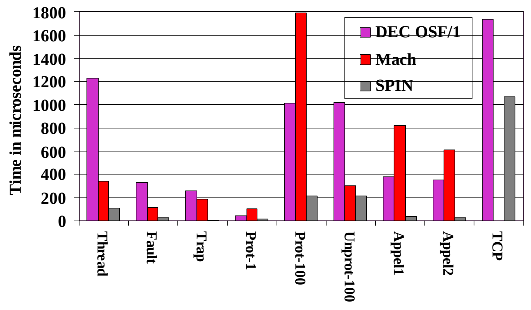
Component	source size in lines	text size in bytes	data size in bytes
NULL syscall	19	96	656
IPC	127	1344	1568
Cthreads	219	2480	1792
DEC OSF/1 threads	305	2304	3488
VM workload	263	5712	1472
IP	744	19008	13088
UDP	1046	23968	16704
TCP	5077	69040	9840
HTTP	392	5712	4176
TCP Forward	187	4592	2080
UDP Forward	138	4592	2144
Video Client	95	2736	1952
Video Server	304	9228	3312

Execution speed

- Performance is comparable to that of C.



System Performance



Language Extensions

- Run-time handles for interfaces and modules.
- Isolation of trust.
- Pointer-safe casting

Isolating Callers

- Execute untrusted code from interrupts
  - » Active messages
- Untrusted clients may not terminate
  - » Forceful termination may violate system state
- **`EPHEMERAL`** procedures can be terminated at any time
  - » Can only call other **`EPHEMERAL`** procedures.

```
EPHEMERAL PROCEDURE ActiveMsgHandler(m: Mbuf.T) =
  BEGIN
    time := time + VIEW(m.data, TimeDelta.T);
  END;
```

Safe Casts

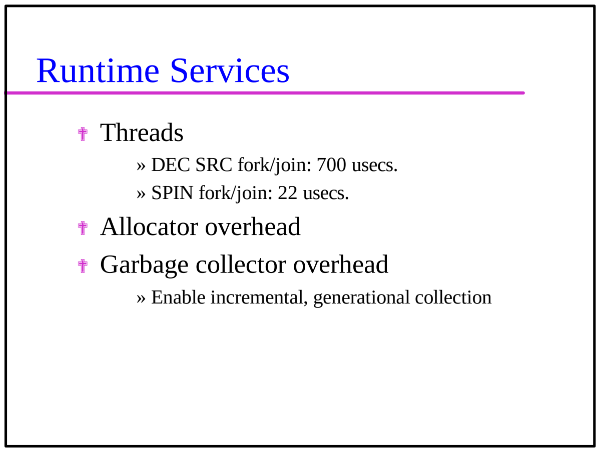
- View raw data as typed data
  - » OSes require viewing bits as typed objects
  - » Copying is expensive and violates sharing
- `WITH NewView = VIEW(var, T) DO … END;`
  - » Cannot forge pointers or create illegal values

Modula-3 Concerns

- Execution speed
- Threads, allocation, GC
- Memory usage
- Mixed-language environment

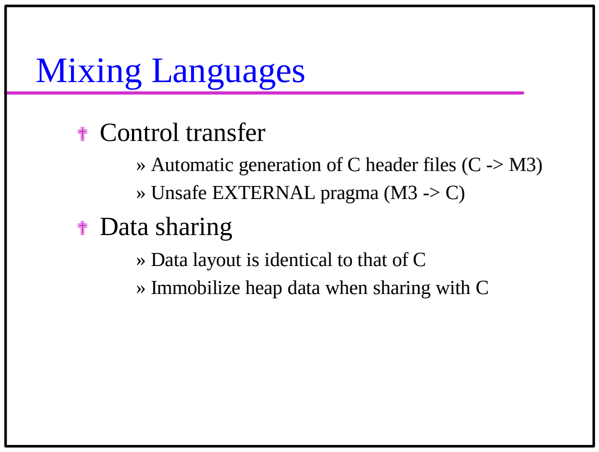
Memory usage

- Code and data size is small
- Sharing reduces memory requirements
- Typical examples:
  - » Web server extension: 9K
  - » Cthreads Package: 4K
  - » TCP forwarder: 6K



Runtime Services

- Threads
  - » DEC SRC fork/join: 700 usecs.
  - » SPIN fork/join: 22 usecs.
- Allocator overhead
- Garbage collector overhead
  - » Enable incremental, generational collection



Mixing Languages

- Control transfer
  - » Automatic generation of C header files (C -> M3)
  - » Unsafe EXTERNAL pragma (M3 -> C)
- Data sharing
  - » Data layout is identical to that of C
  - » Immobilize heap data when sharing with C