



Active Oberon on Intel SMP

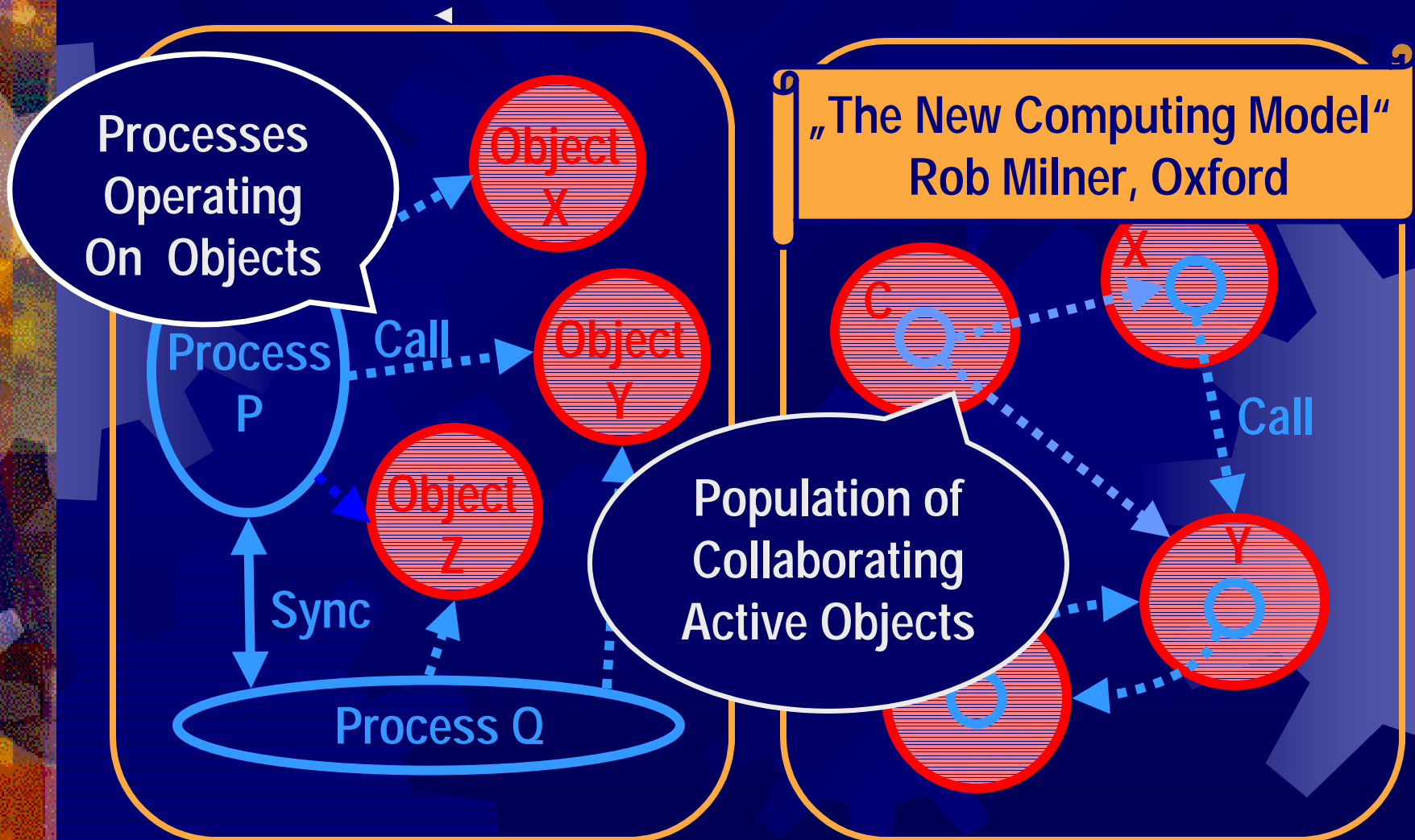
Implementation of Active Objects on Intel SMP Platforms

Jürg Gutknecht, ETH Zürich

Seminar on Effective Implementation
of OO Languages, Schloss Dagstuhl,
November 2000

Active Objects

A Uniform Computing Model



ETH Zürich

Programming Languages

- **1960 Algol**

- Procedures, Locality

- **1970 Pascal**

- Data Types

P-Code

Programming
In-the-Small

- **1980 Modula**

- Modules

- **1990 Oberon**

- Type Extension

Native on
Intel PC

Programming
In-the-Large

- **2000 Active Oberon**

- Active Objects

Native on
Intel SMP

- **2000 Lightning Oberon**

- Language Interoperability

On .NET
with MSR

Active Oberon

Basic Principles

```
Z = OBJECT
```

State

```
VAR t: T;
```

```
PROCEDURE p (u: U; VAR v: V);
```

```
BEGIN { EXCLUSIVE } ...
```

```
END p;
```

Methods

```
PROCEDURE q;
```

```
BEGIN assertion := TRUE
```

```
END q;
```

```
BEGIN { ACTIVE }
```

```
BEGIN { EXCLUSIVE } ...
```

```
PASSIVATE assertion; ...
```

```
END
```

```
END Z;
```

Call

Object Acts
as Monitor

Exclusive
Access

Separate
Thread

Await
Assertion

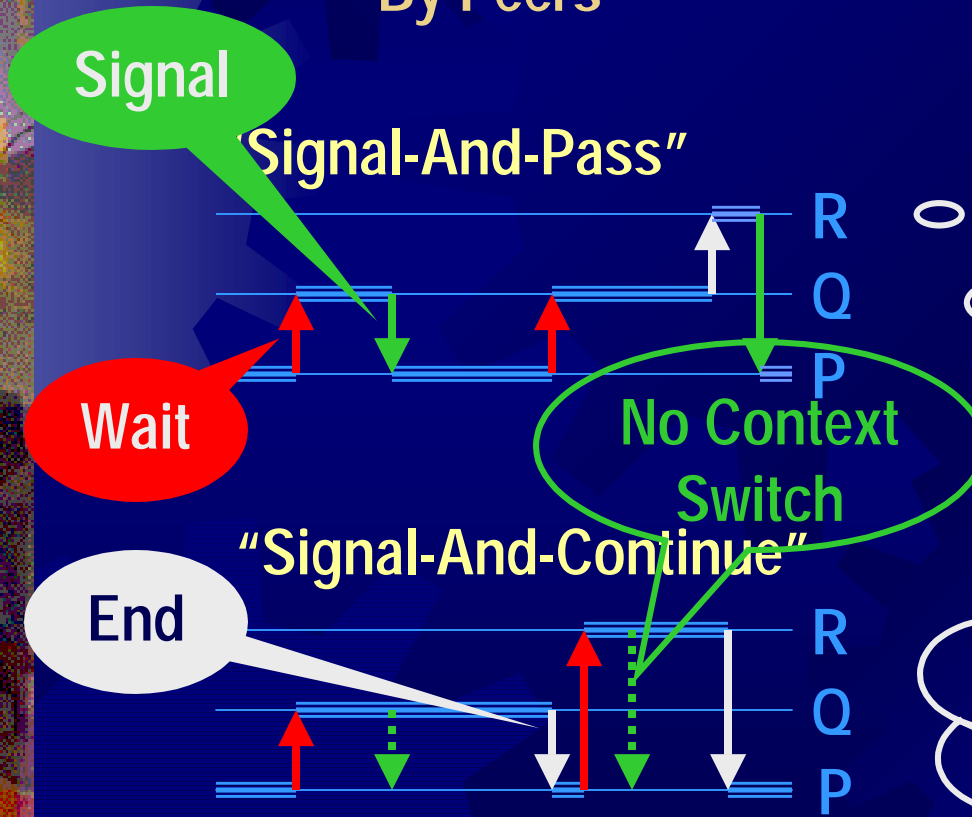
X

Y

Assertions

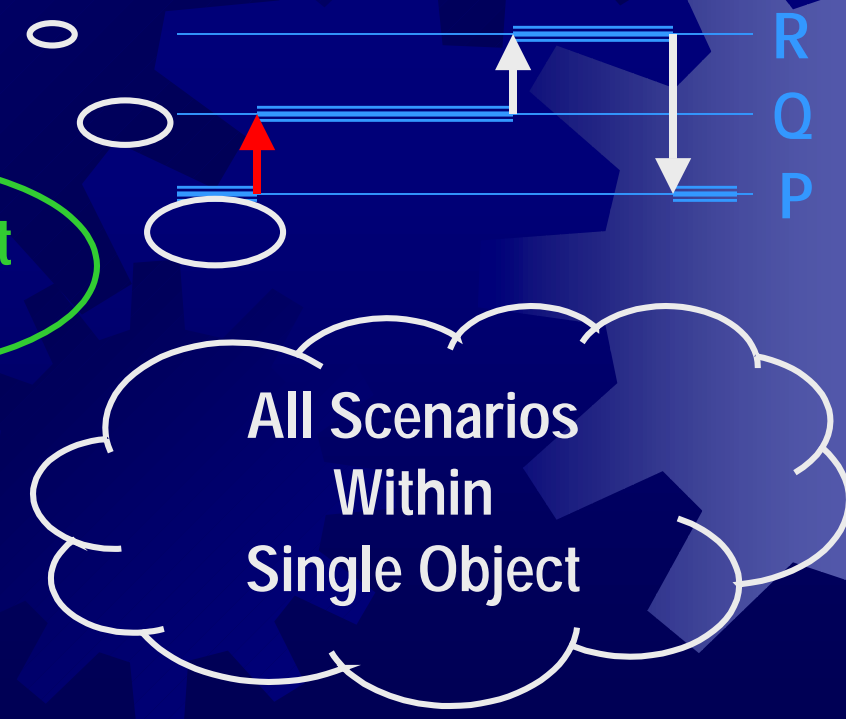
A Synchronization Tool

Based on Signaling
By Peers



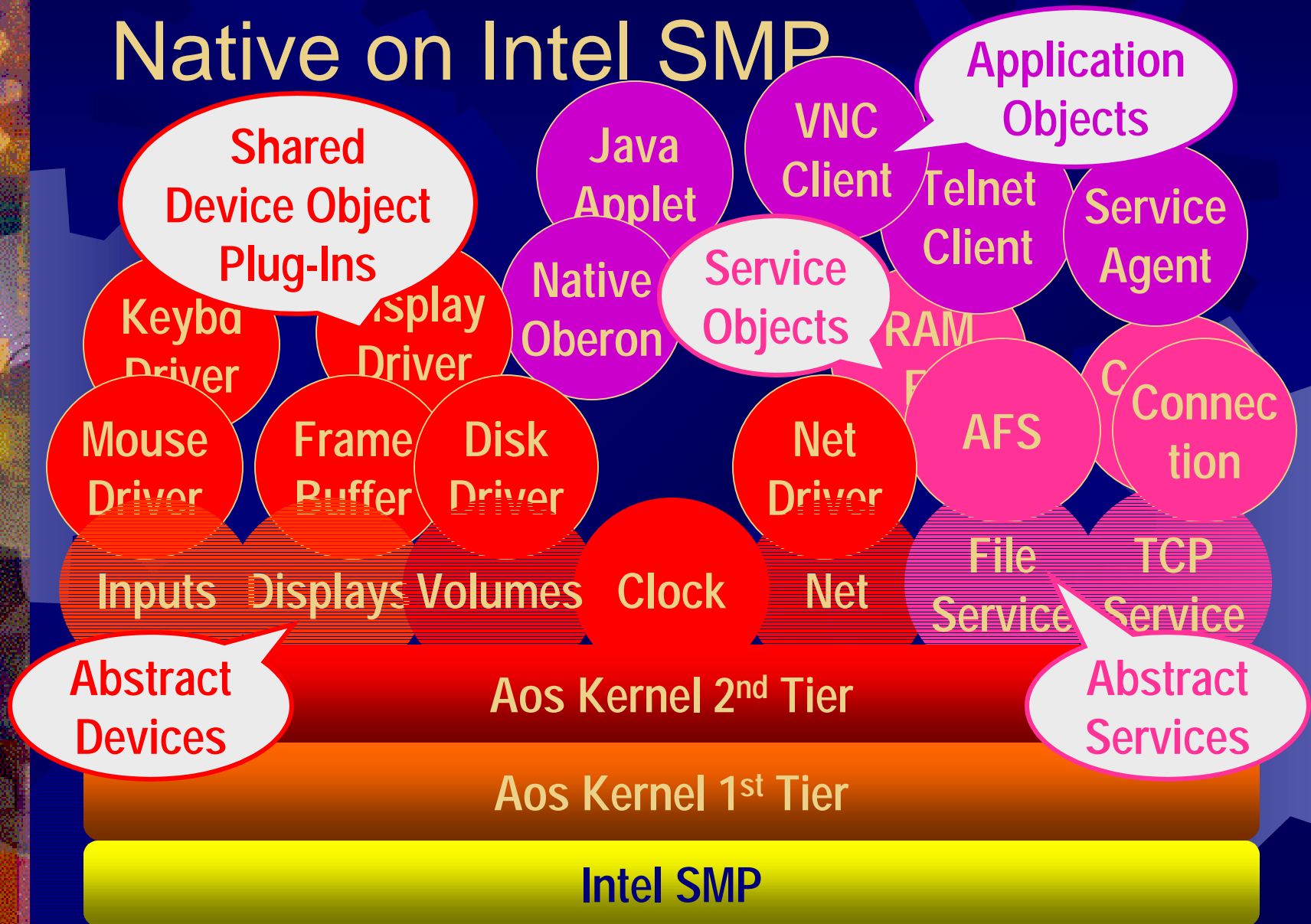
Based on System
Managed Assertions

Fully Scaling



Aos Kernel (by Pieter Muller)

Native on Intel SMP



Aos Kernel

Functionality & Size

{ SharedDevices & Services

- Input, Display, Volumes, Files, Network

*Total Size
125 KB*

} Process Management

- Conditions & Object Locks

CPU Management

- Scheduler & Interrupt Handlers

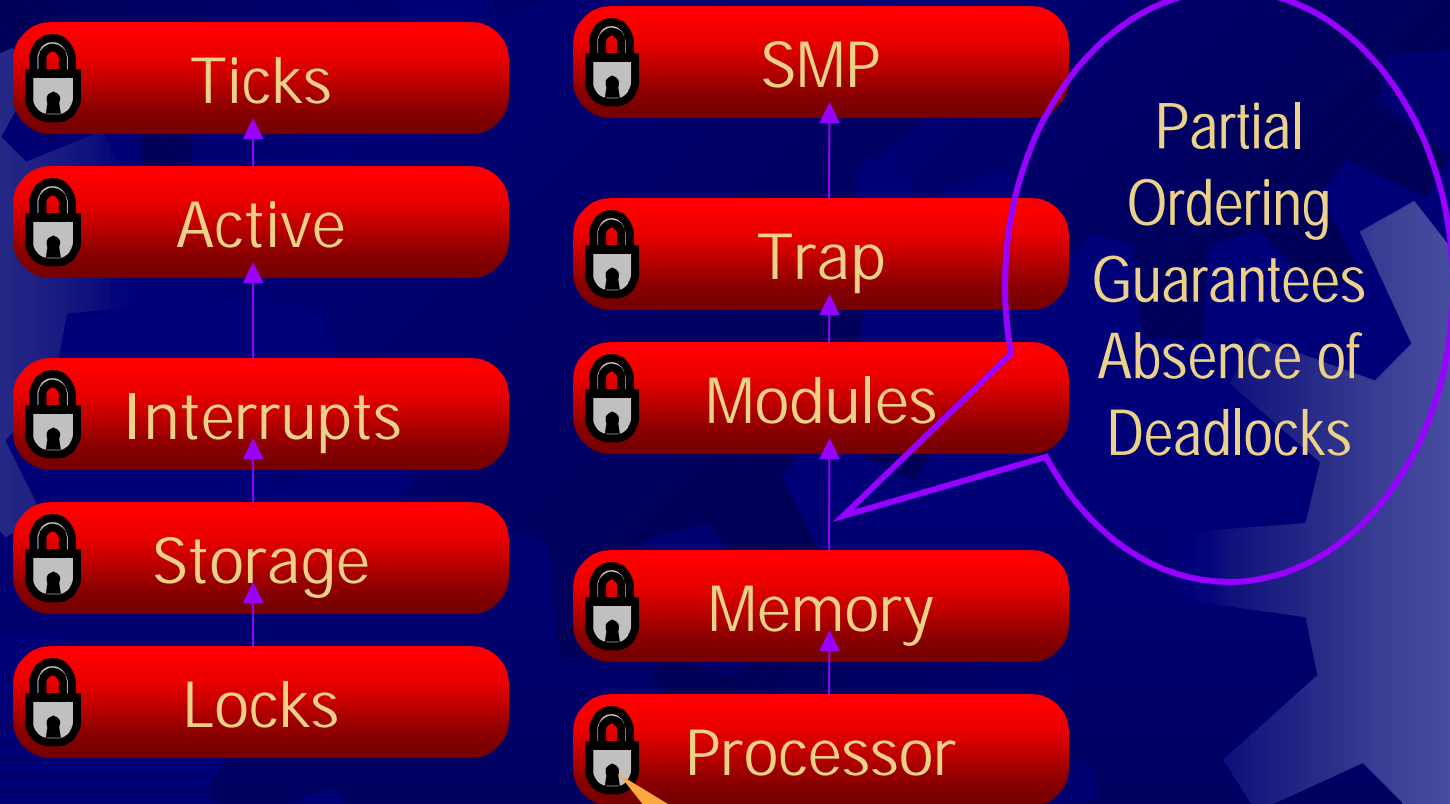
Memory Management

- Stacks (Based on Memory Mapping)
- Heap

*Total Size
50 KB*



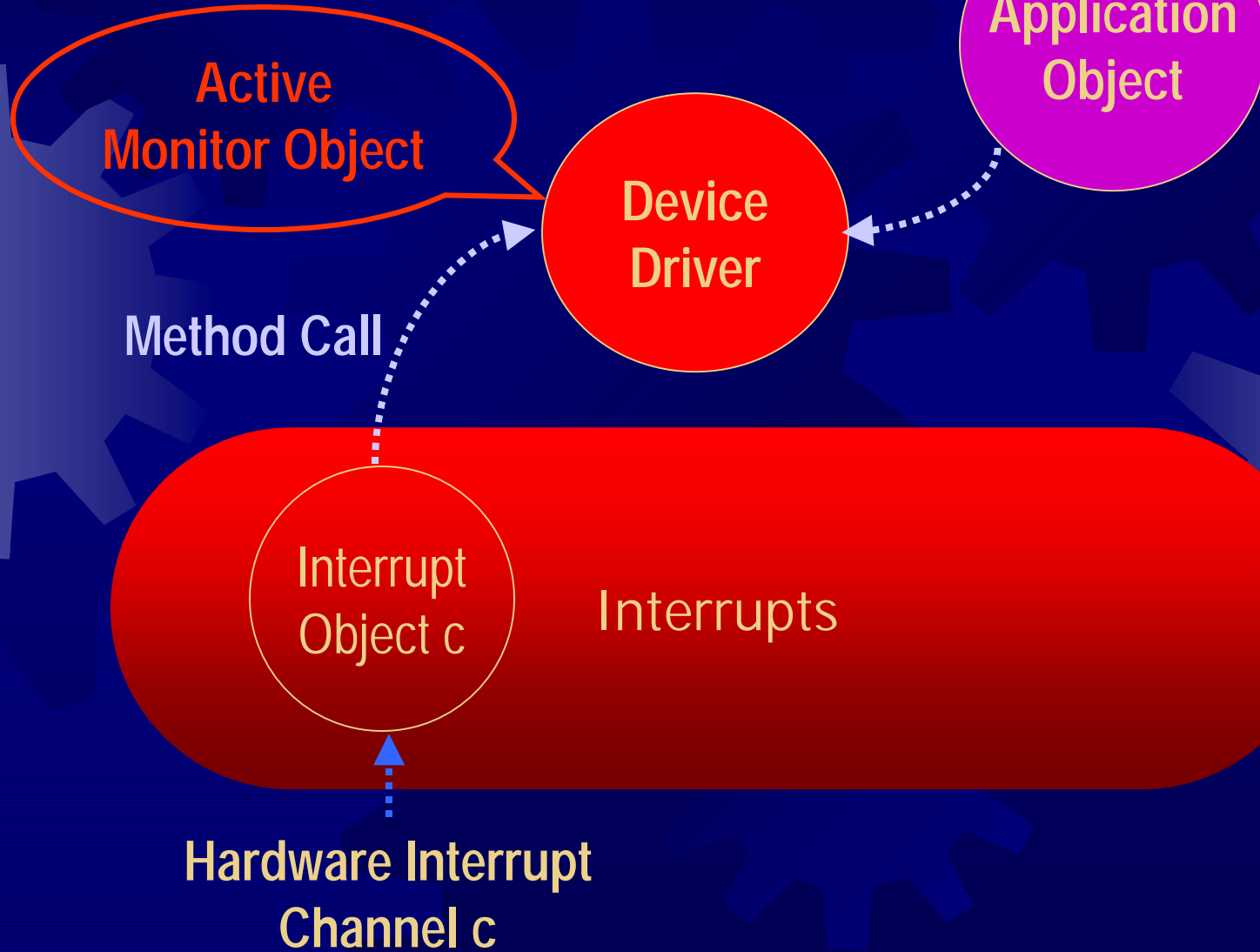
Aos Kernel Modular Structure



1st Tier Zoomed-In

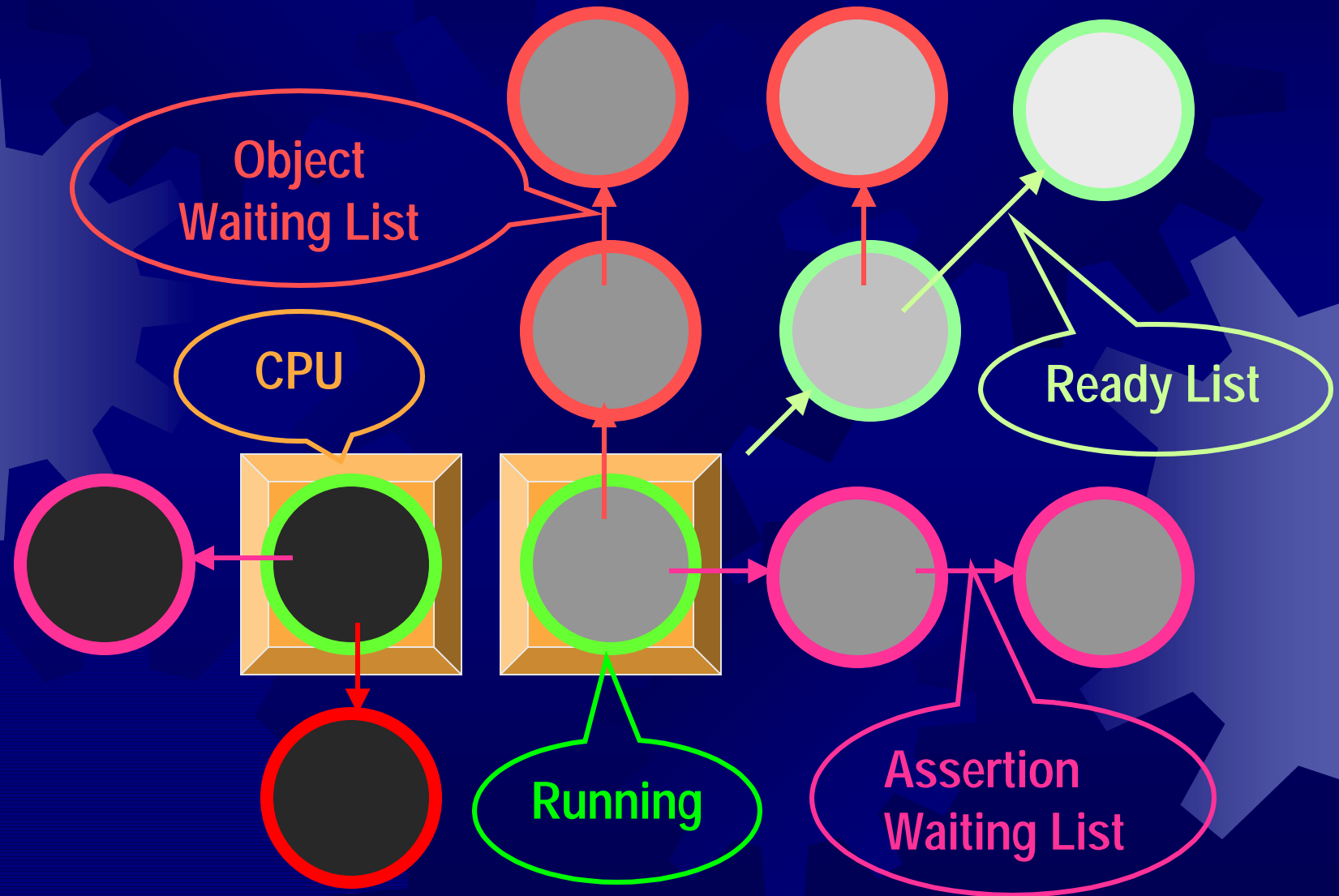
Module Lock

Aos Kernel Device Drivers



Aos Kernel

Runtime Data Structure



Aos Kernel Object Scheduling



NEW
Create object;
Create process
Set to ready 0

Enter Monitor
IF monitor lock set THEN
Put me in monitor obj wait list;
Run next ready 1
ELSE set monitor lock
END 2

without context switch

Exit Monitor
Find first asserted x in assn wait list;
IF x found THEN set x to ready 5
ELSE Find first x in obj wait list:
IF x found THEN set x to ready 4
ELSE clear monitor lock
END
END
Run next ready 1

Preempt
Set to ready 6
Run next ready 1

PASSIVATE
Put me in monitor assn wait
Exit monitor 3 4 5

END
Run next ready 7 1