The JX Operating System: An Overview

Michael Golm

University of Erlangen
Department of Computer Science
(Operating Systems and Distributed Systems)
Martensstrasse 1
91058 Erlangen, Germany
golm@cs.fau.de

Outline

- Objectives
- Architecture
- Protection
- Flexibility
- Performance

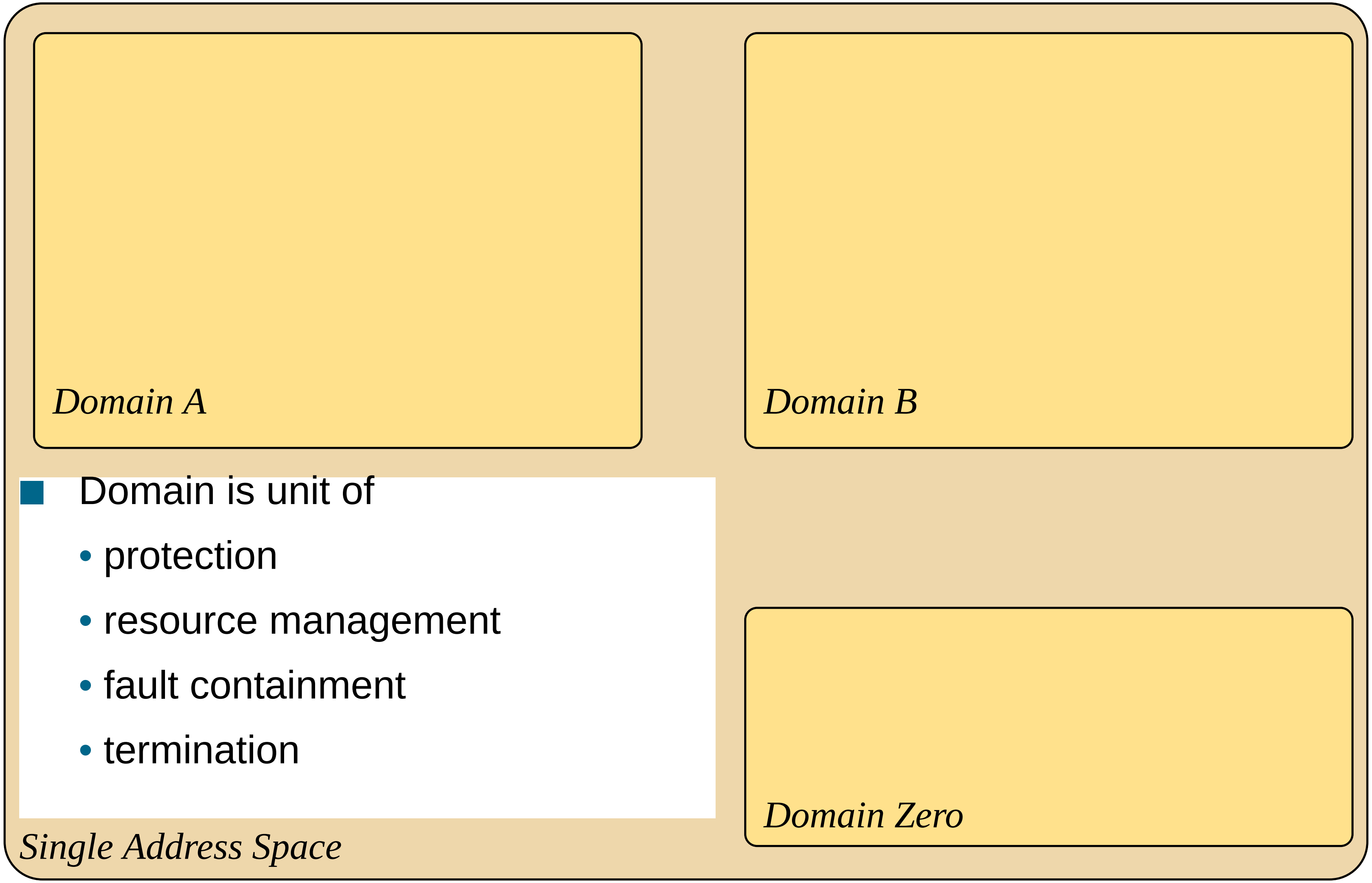
Objectives of the JX system

- Make writing an OS as easy as writing applications
  - → simple and robust architecture
- Dynamic OS extension with untrusted components
  - → exact resource accounting and customizable management
- Code reuse
  - tailored OS configurations; dedicated systems
- Protection
  - flexibility
  - performance
  - robustness

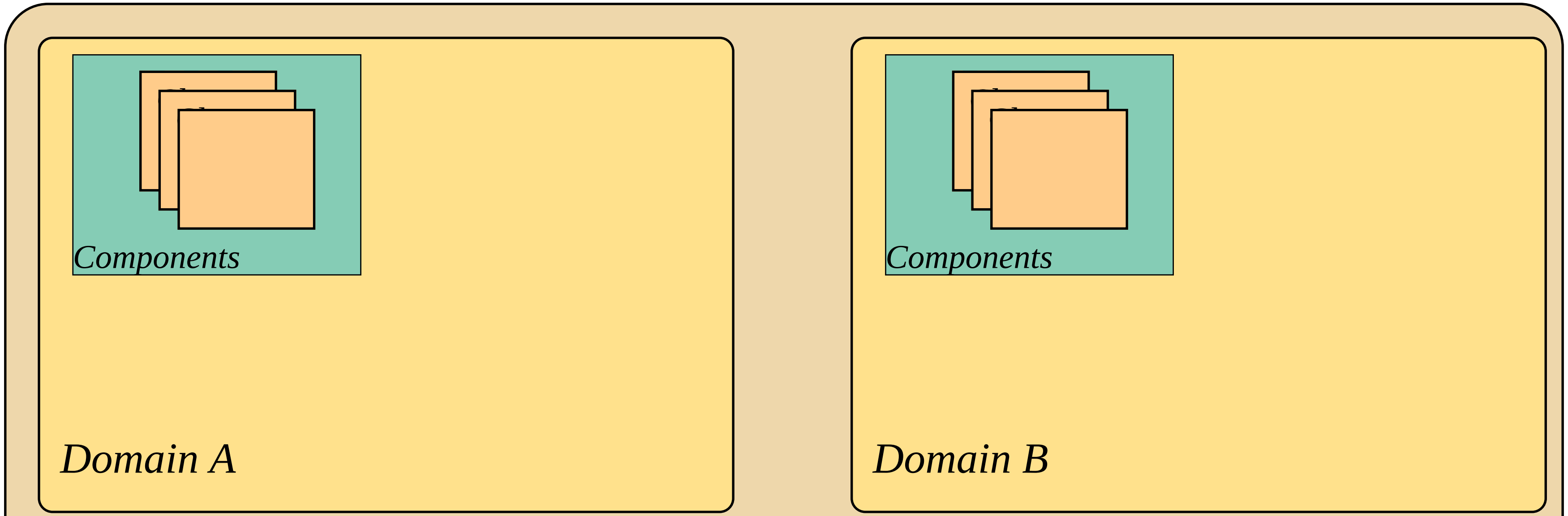
Single Address Space

Domains

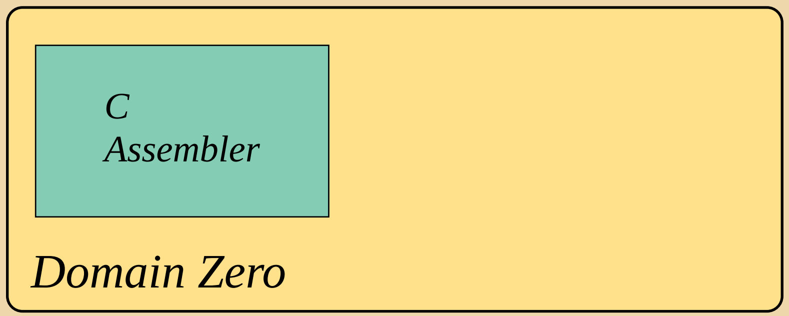
- Domain is unit of
  - protection
  - resource management
  - fault containment
  - termination



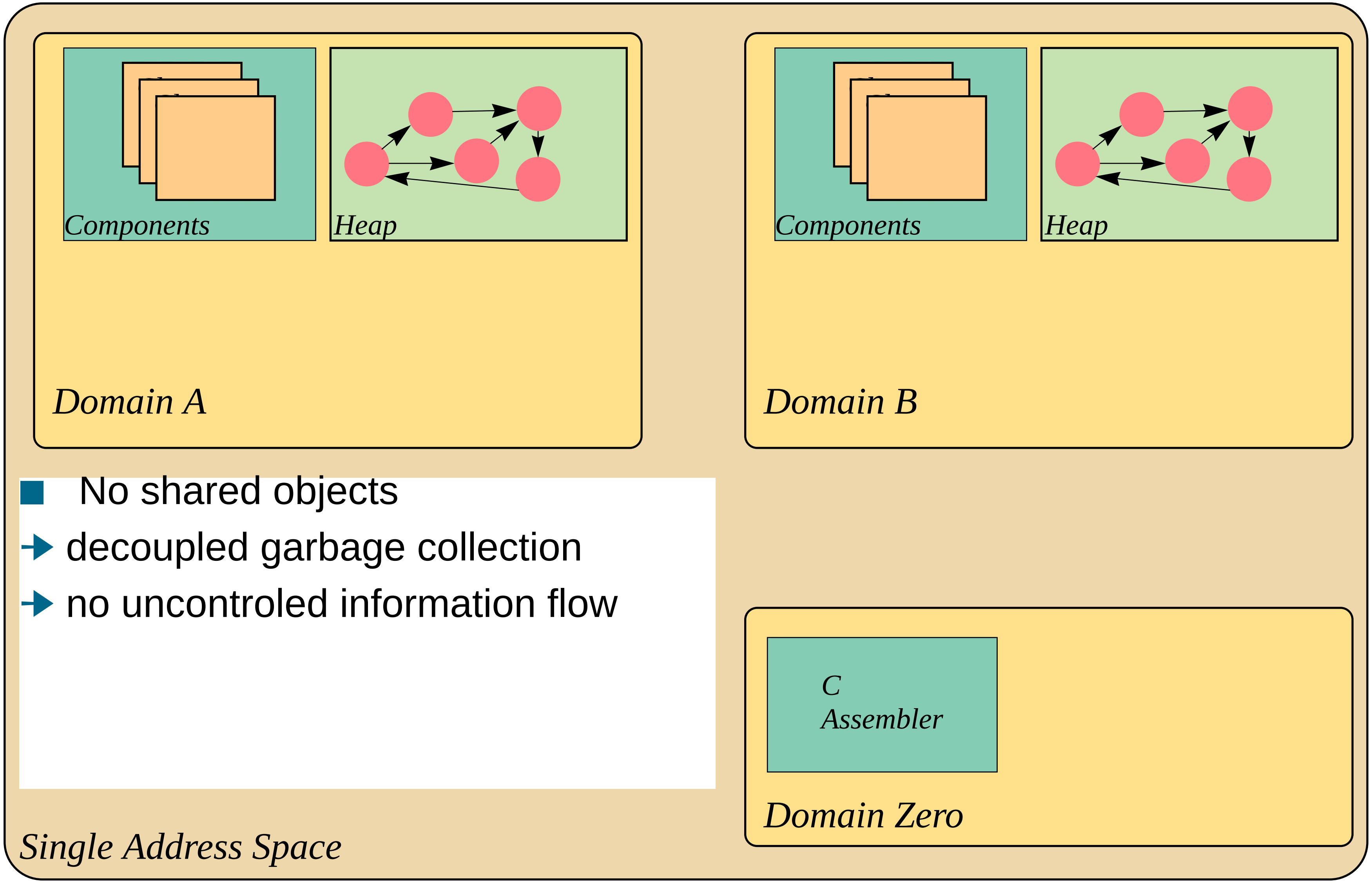
Components



- Components contain 100% Java
  - Bytecode is verified by an extended verifier
  - All components are compiled to native code at load time
- DomainZero contains core written in C and assembler

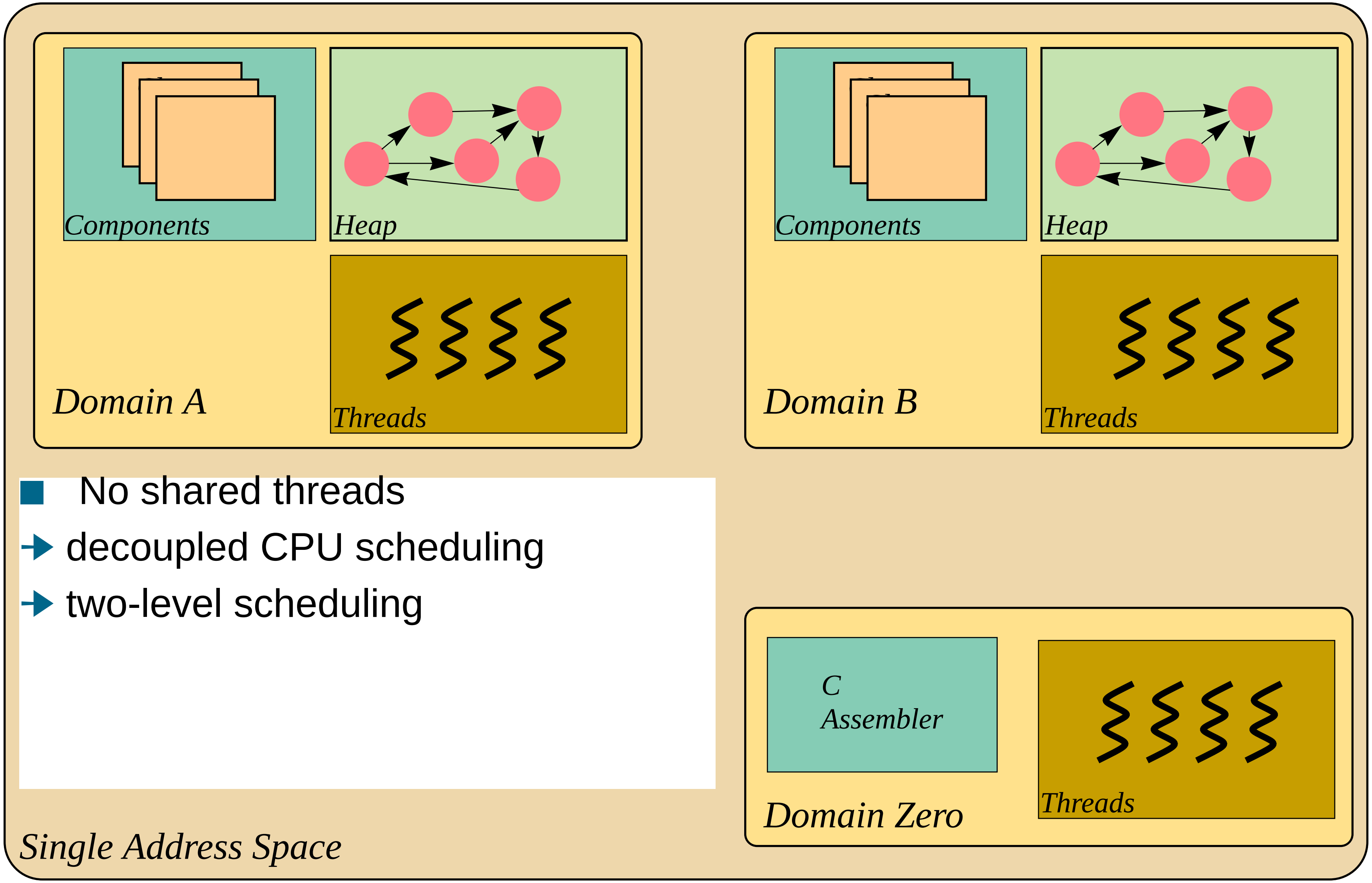


Objects & Heap



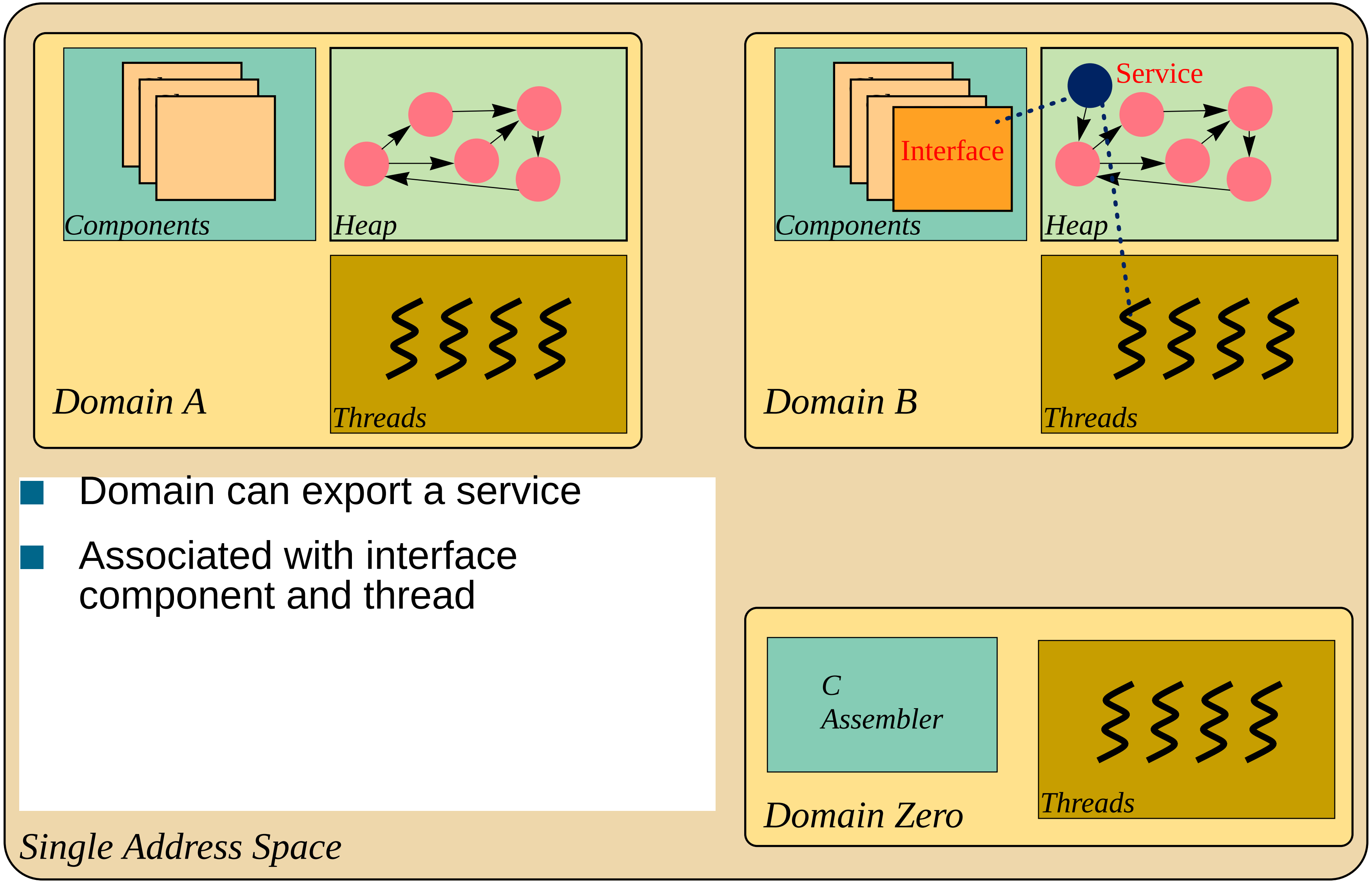
- No shared objects
  - → decoupled garbage collection
  - → no uncontroled information flow

Threads



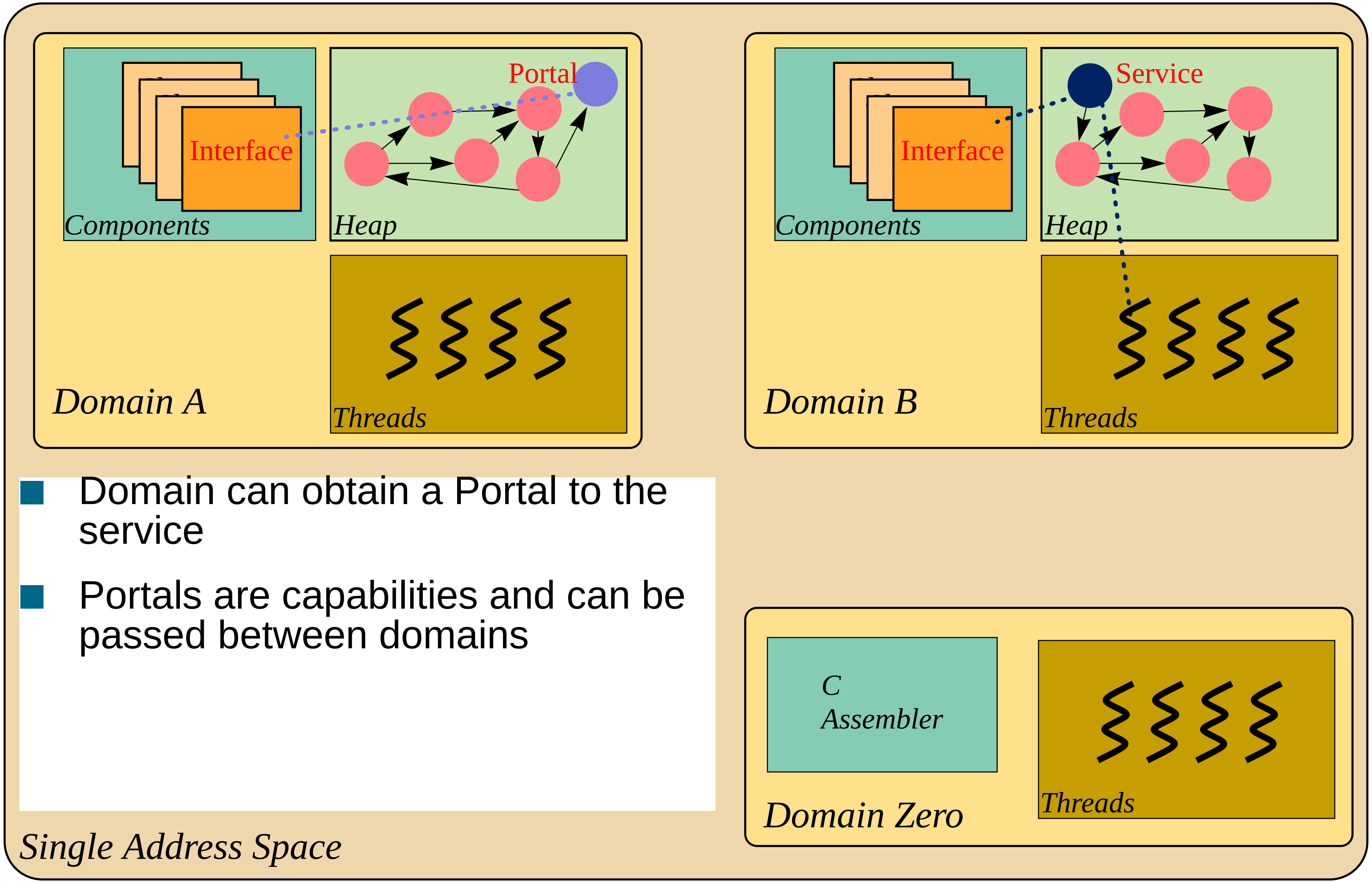
- No shared threads
  - → decoupled CPU scheduling
  - → two-level scheduling

Communication: Portals



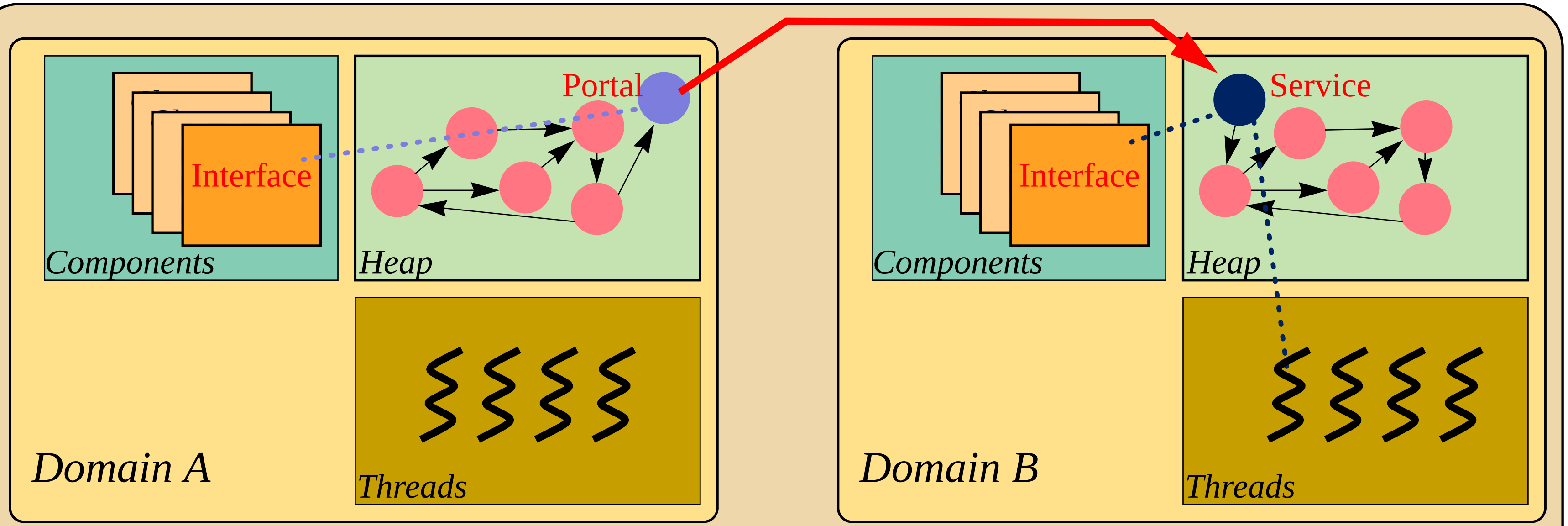
- Domain can export a service
- Associated with interface component and thread

Communication: Portals

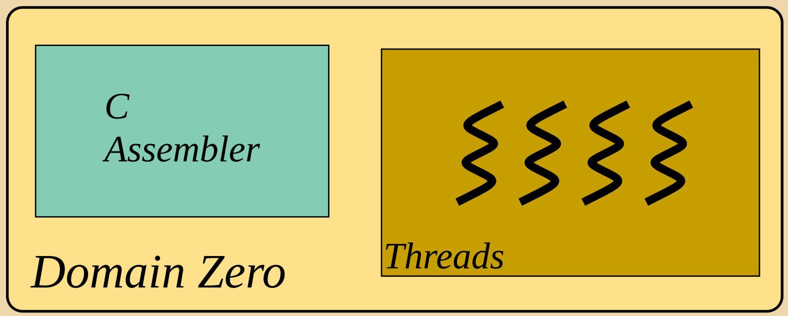


- Domain can obtain a Portal to the service
- Portals are capabilities and can be passed between domains

Communication: Portals

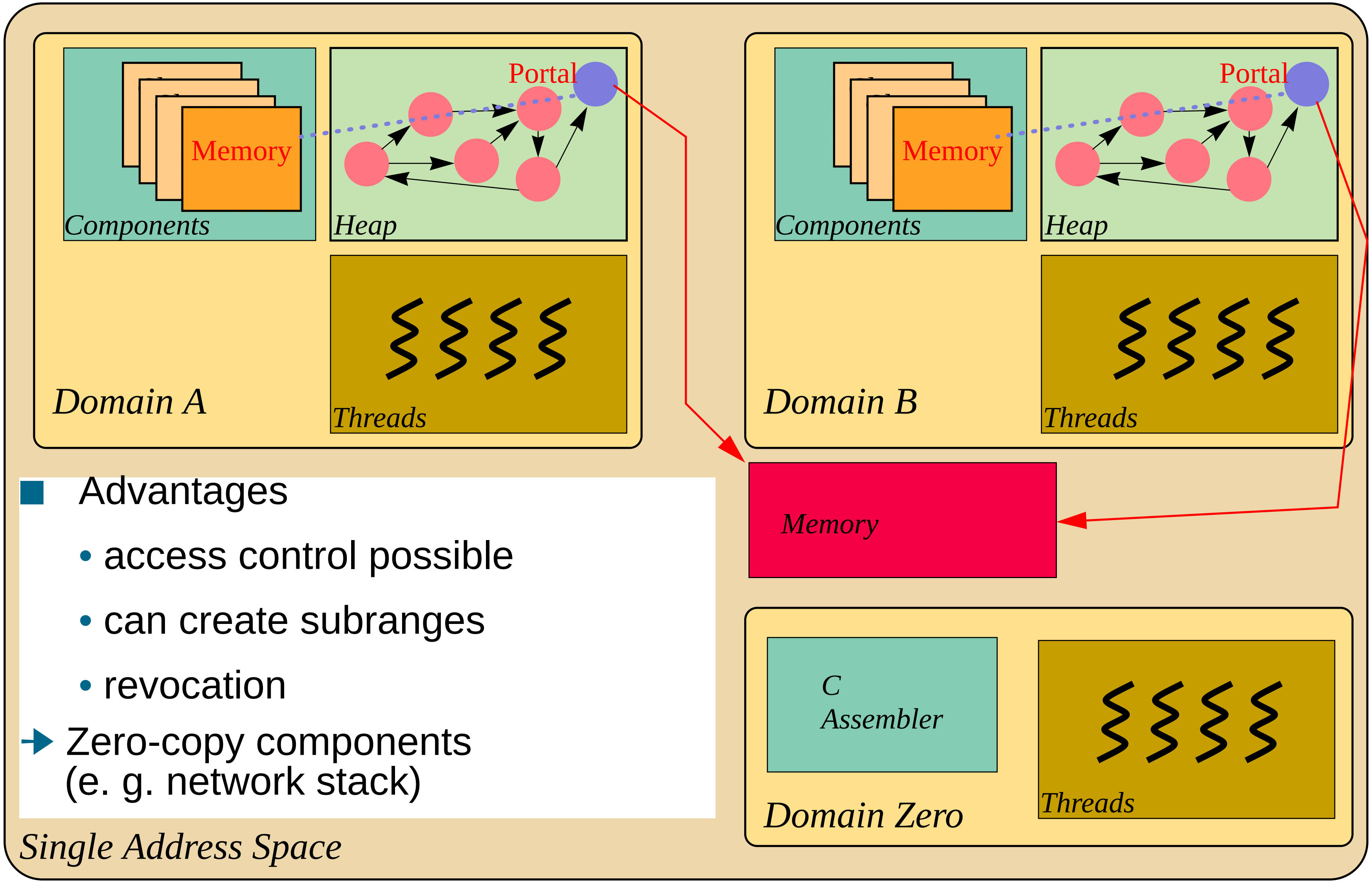


- Portal call:
  - threads are switched (rendezvous, handoff)
  - object arguments: copy
  - portal arguments: new portal to same service



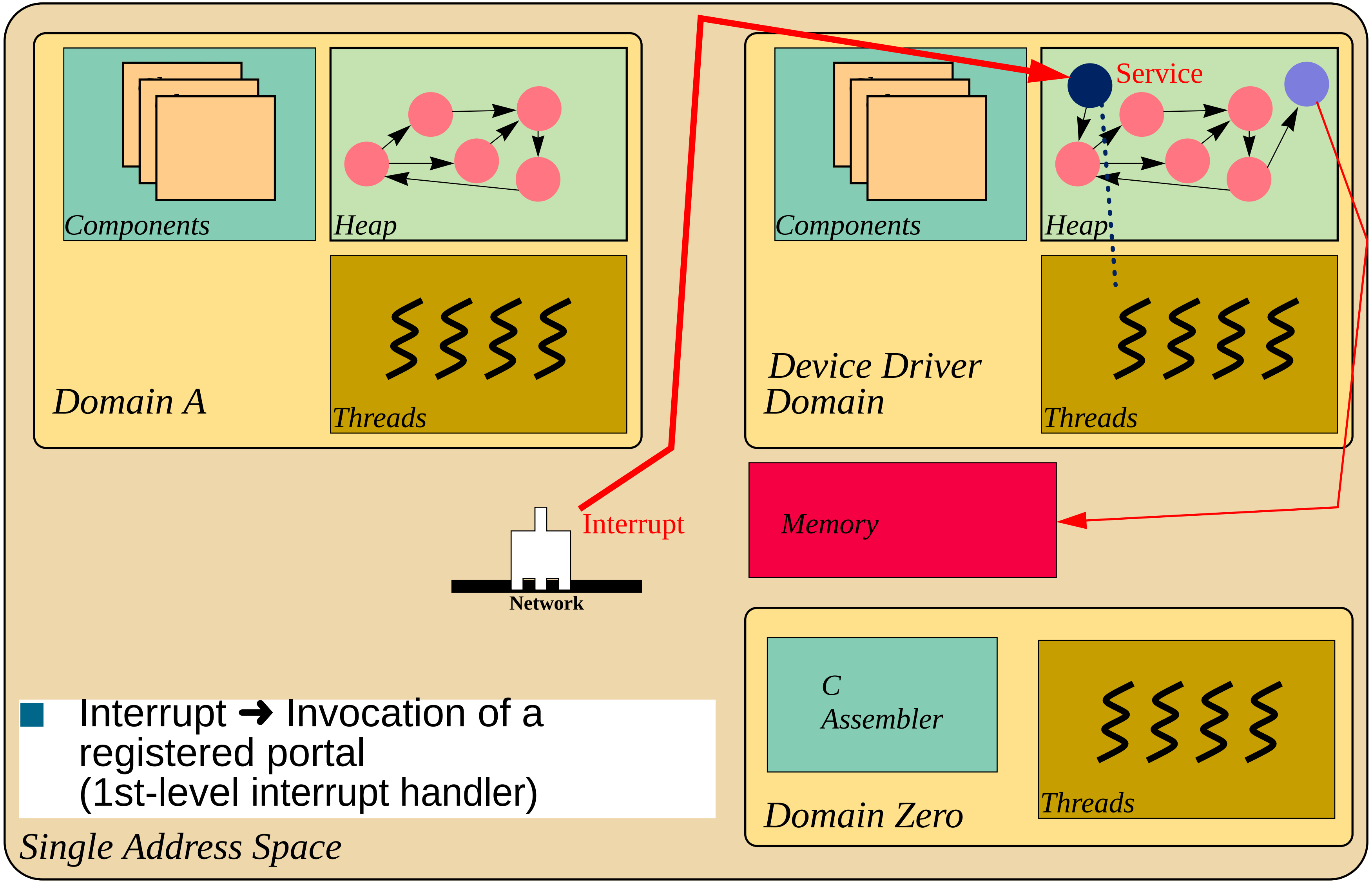
Single Address Space

Communication: Memory



- Advantages
  - access control possible
  - can create subranges
  - revocation
- → Zero-copy components (e. g. network stack)

Device Driver



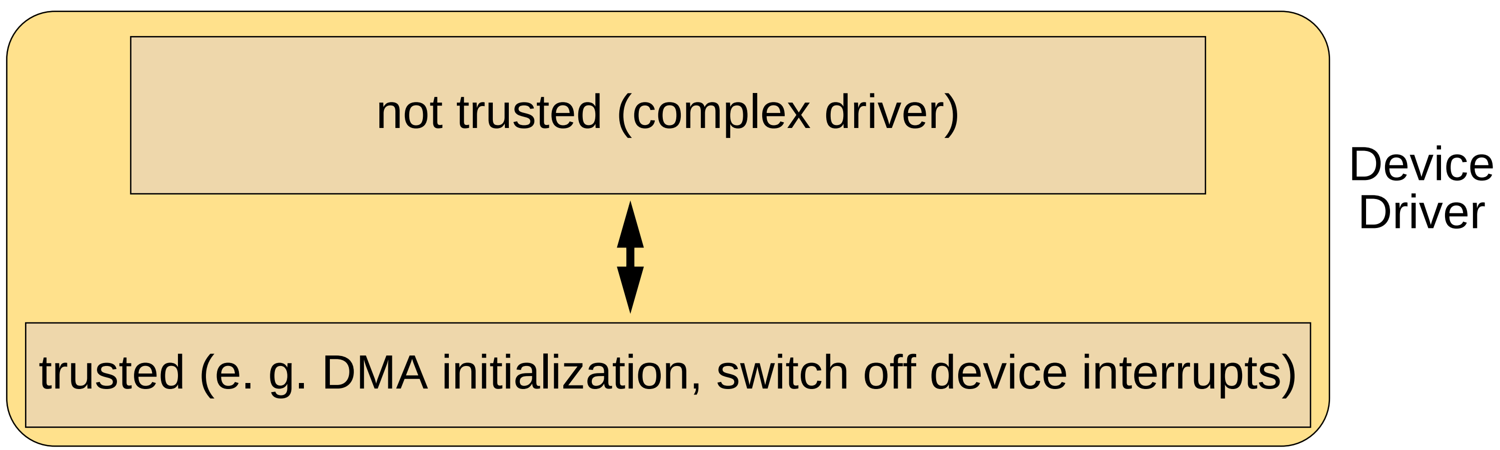
- Interrupt ➜ Invocation of a registered portal (1st-level interrupt handler)

Protection: Interrupt Handler

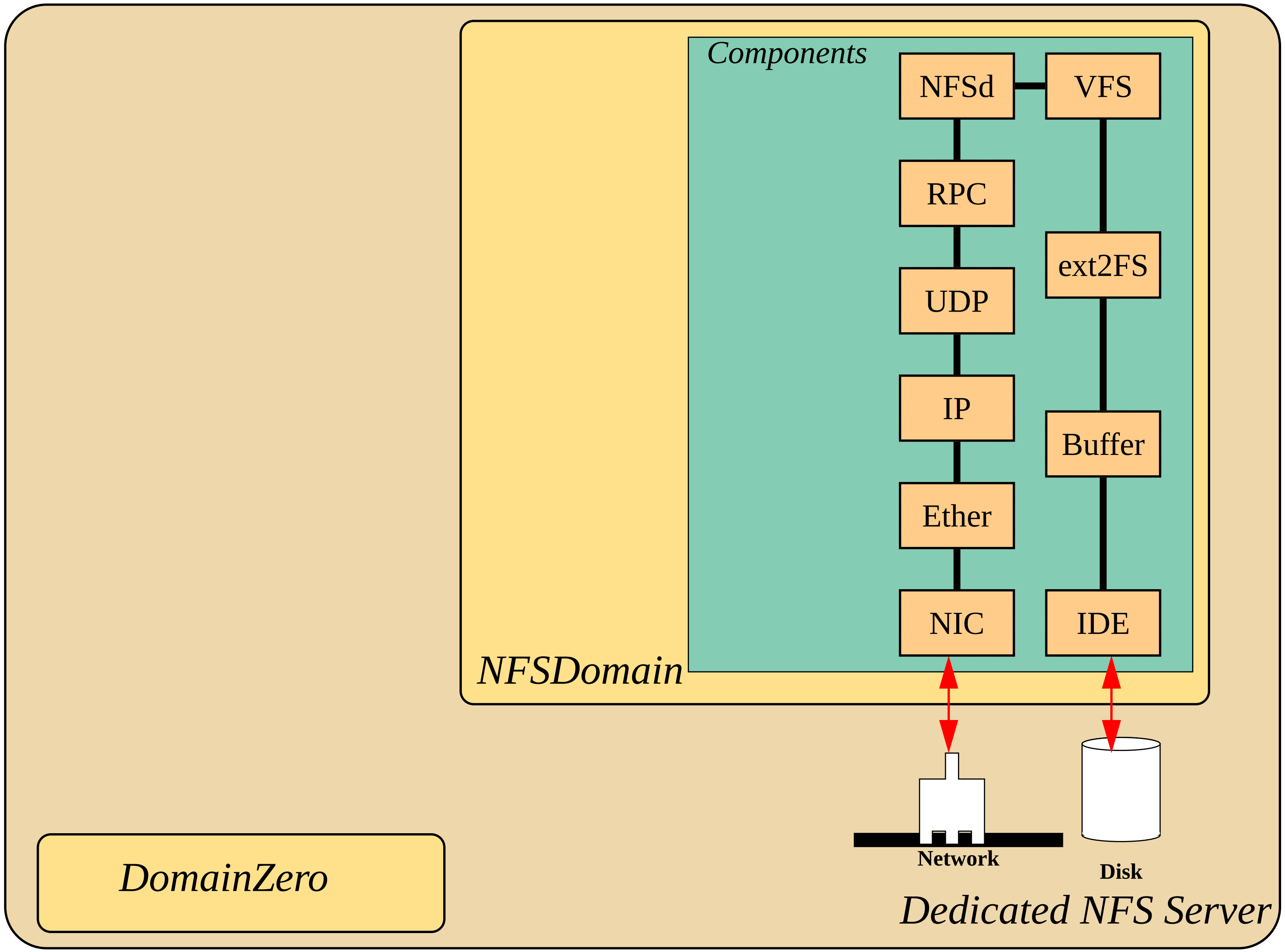
- Interrupts on the interrupted CPU are blocked during execution of the interrupt handler
- Verifier checks interrupt handler for upper limit of execution time
  - can insert runtime checks to ensure timely termination
  - runtime check can terminate interrupt handler and initiate counter measure (e. g., switch off device interrupts)

Protection: Device Driver

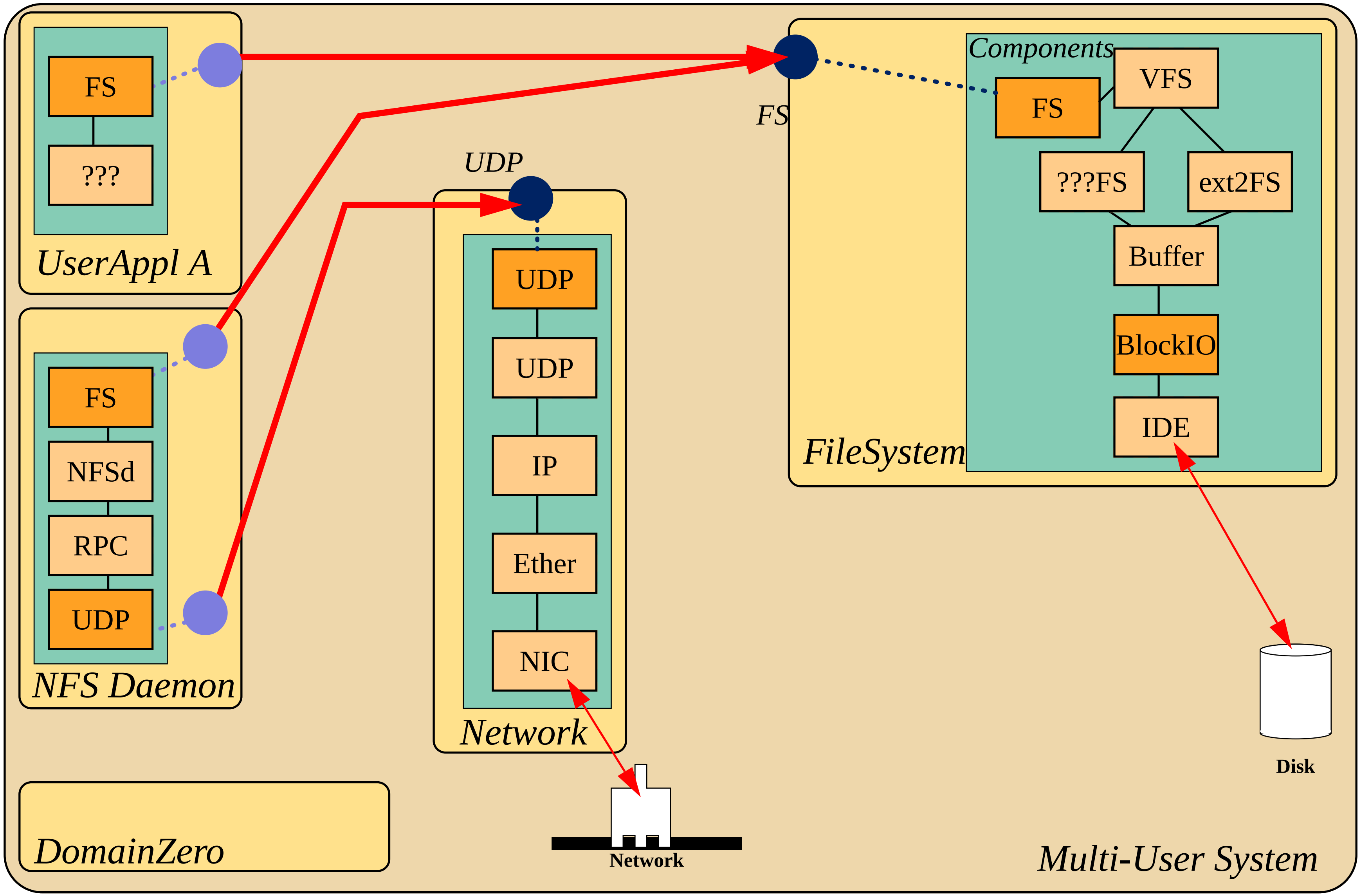
- JX protection is based on type safety and portals
- Some domains can circumvent these mechanisms
  - DomainZero, Translator, Verifier ➜ Trust
  - (some) device drivers



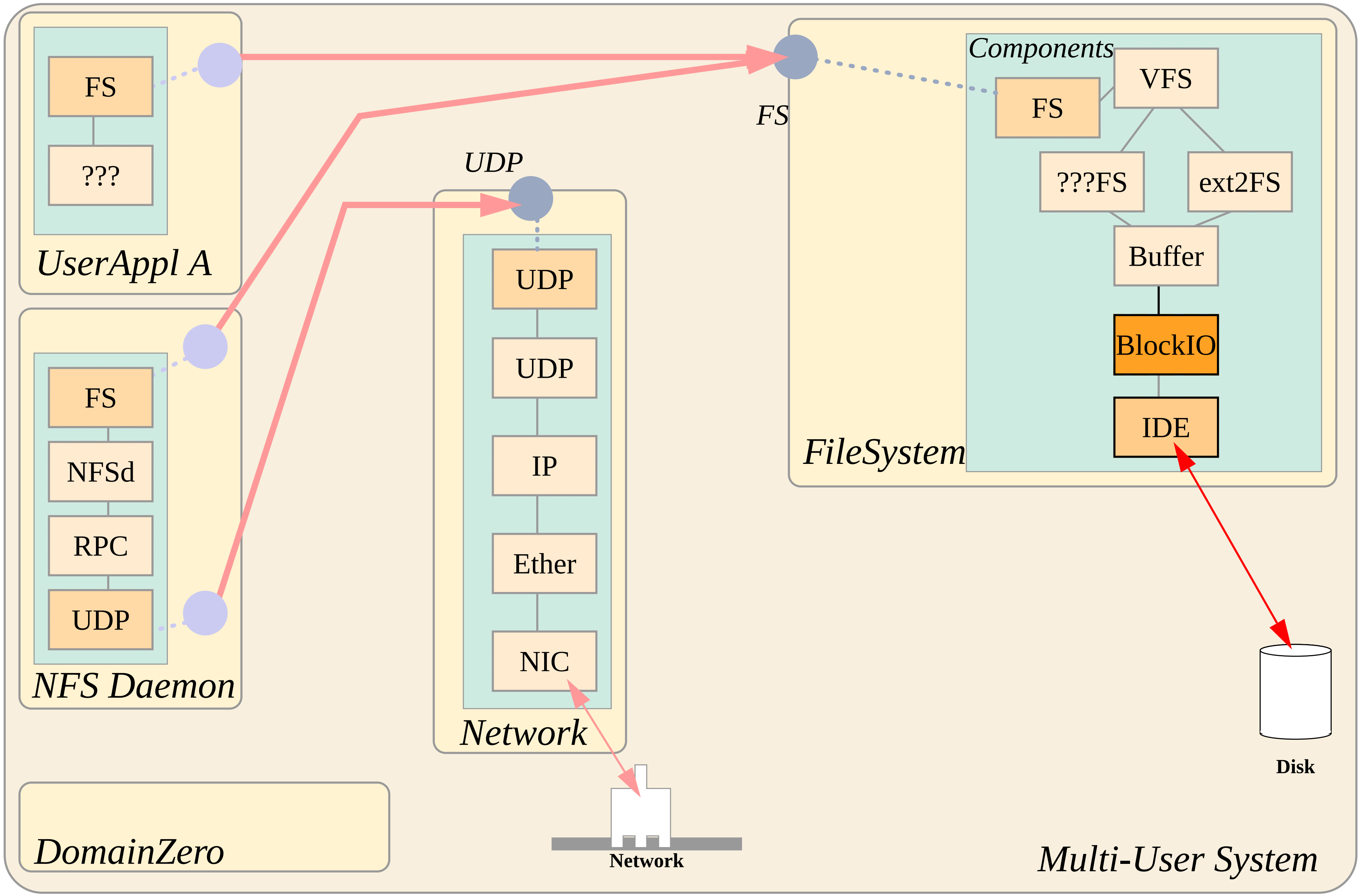
Building an OS: A Dedicated System



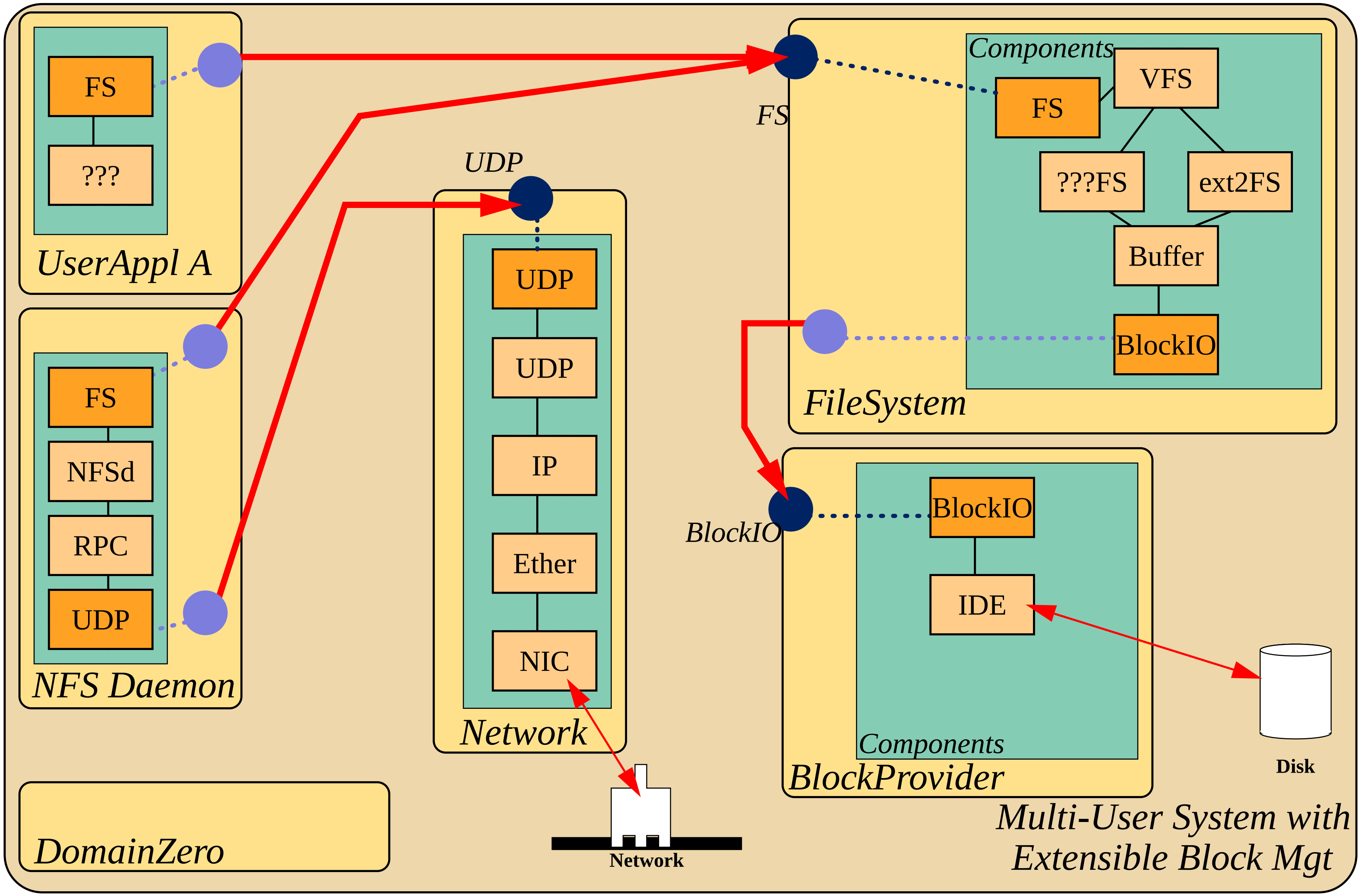
Building an OS: A Multiuser System



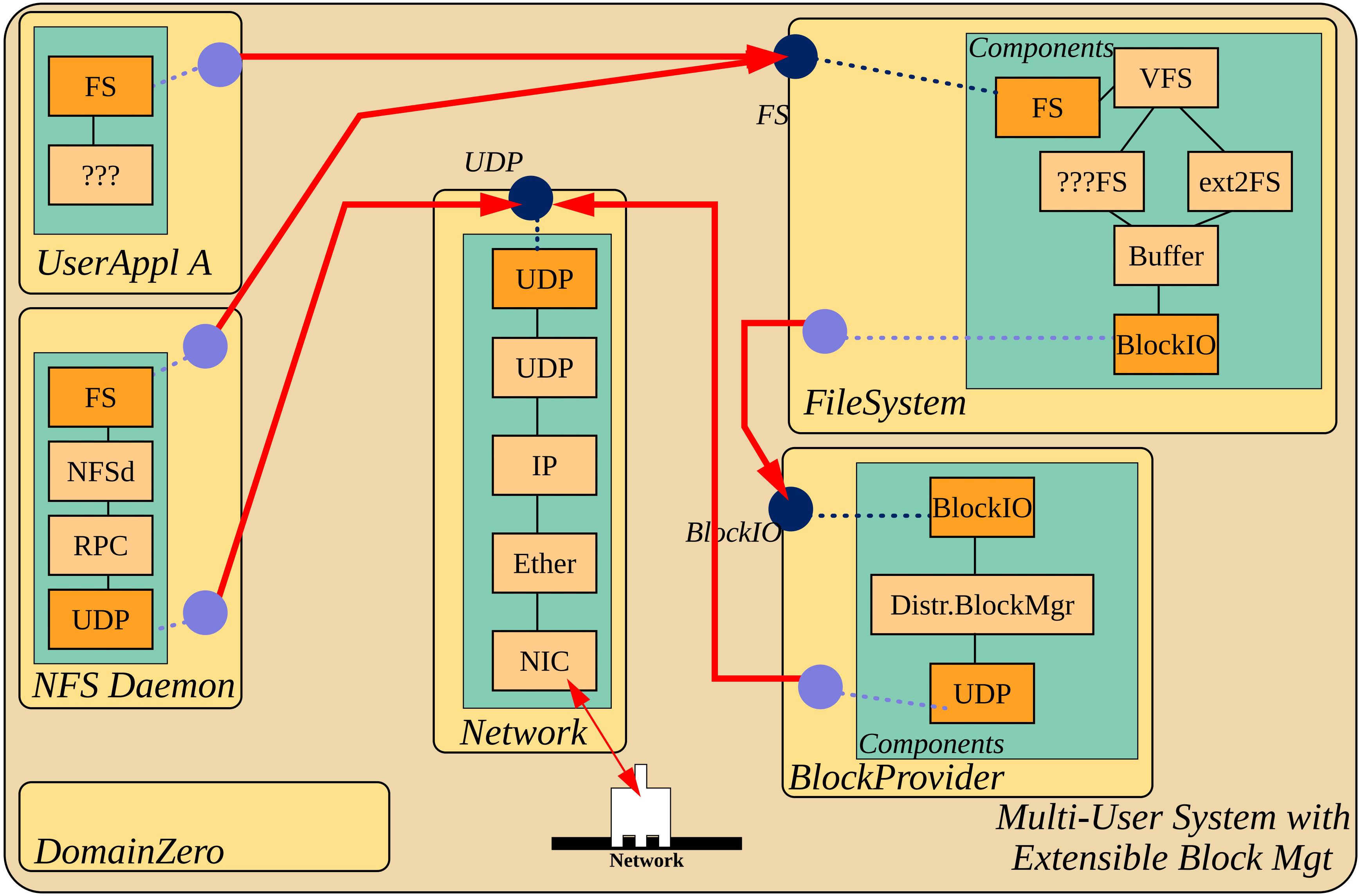
Building an OS: A Multiuser System



Building an OS: Extensibility



Building an OS: Extensibility



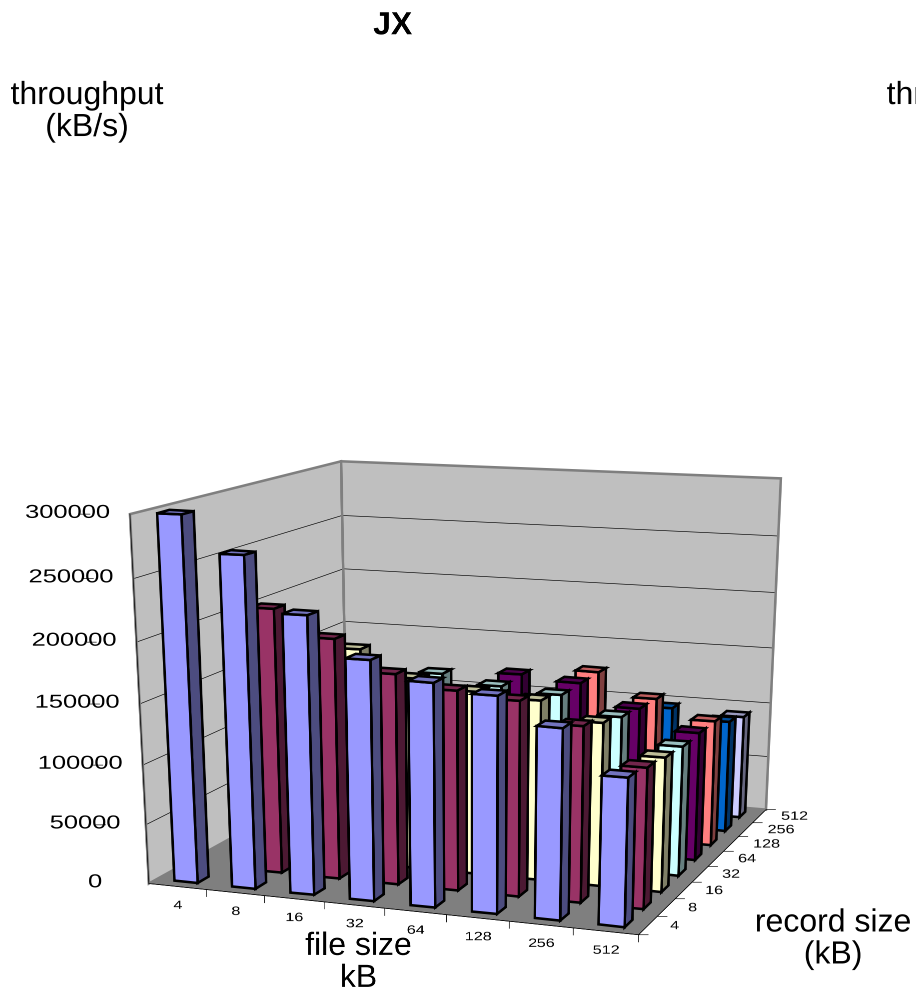
Performance

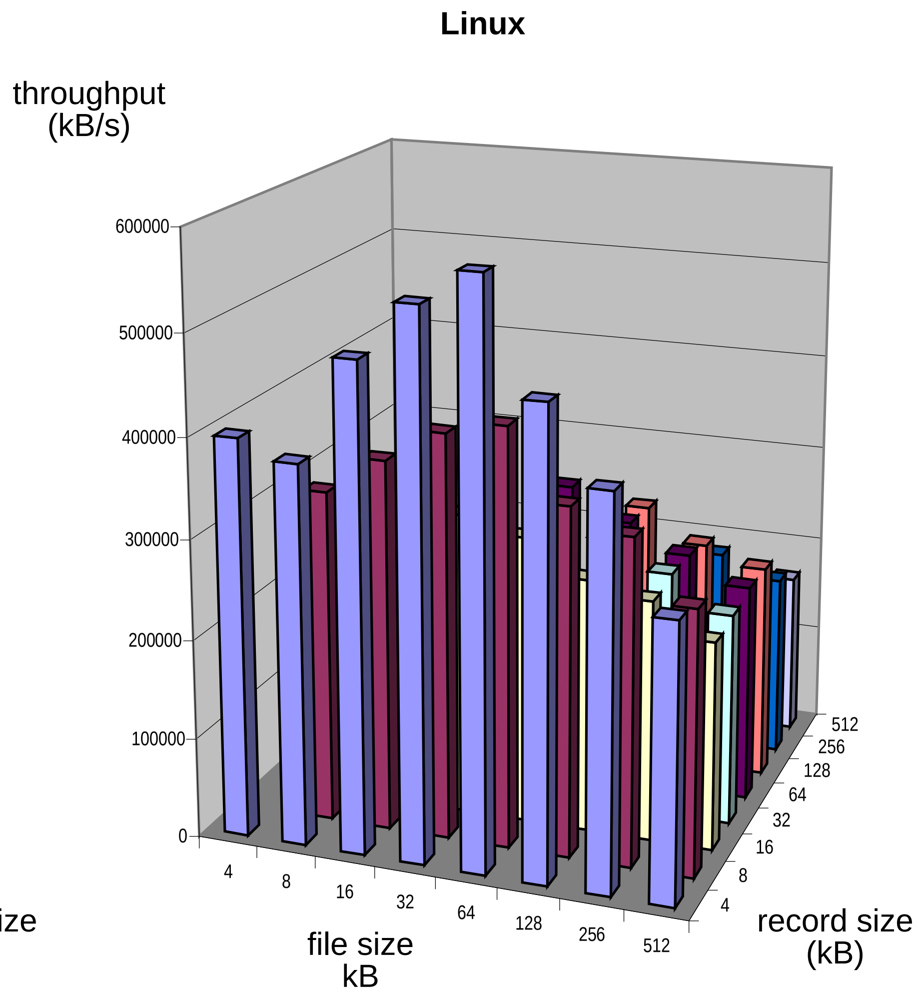
- Hardware:
  500MHz PIII, 128MB RAM,
  IDE: Maxtor 91303D6, 12427MB, 512kB Cache
  NIC: 3C905B 100 MB/s

- IPC:
  - Portal round trip 650 cycles
  - L4 (including RPC stubs): 800 cycles
  - KaffeOS: 27270 cycles

Performance

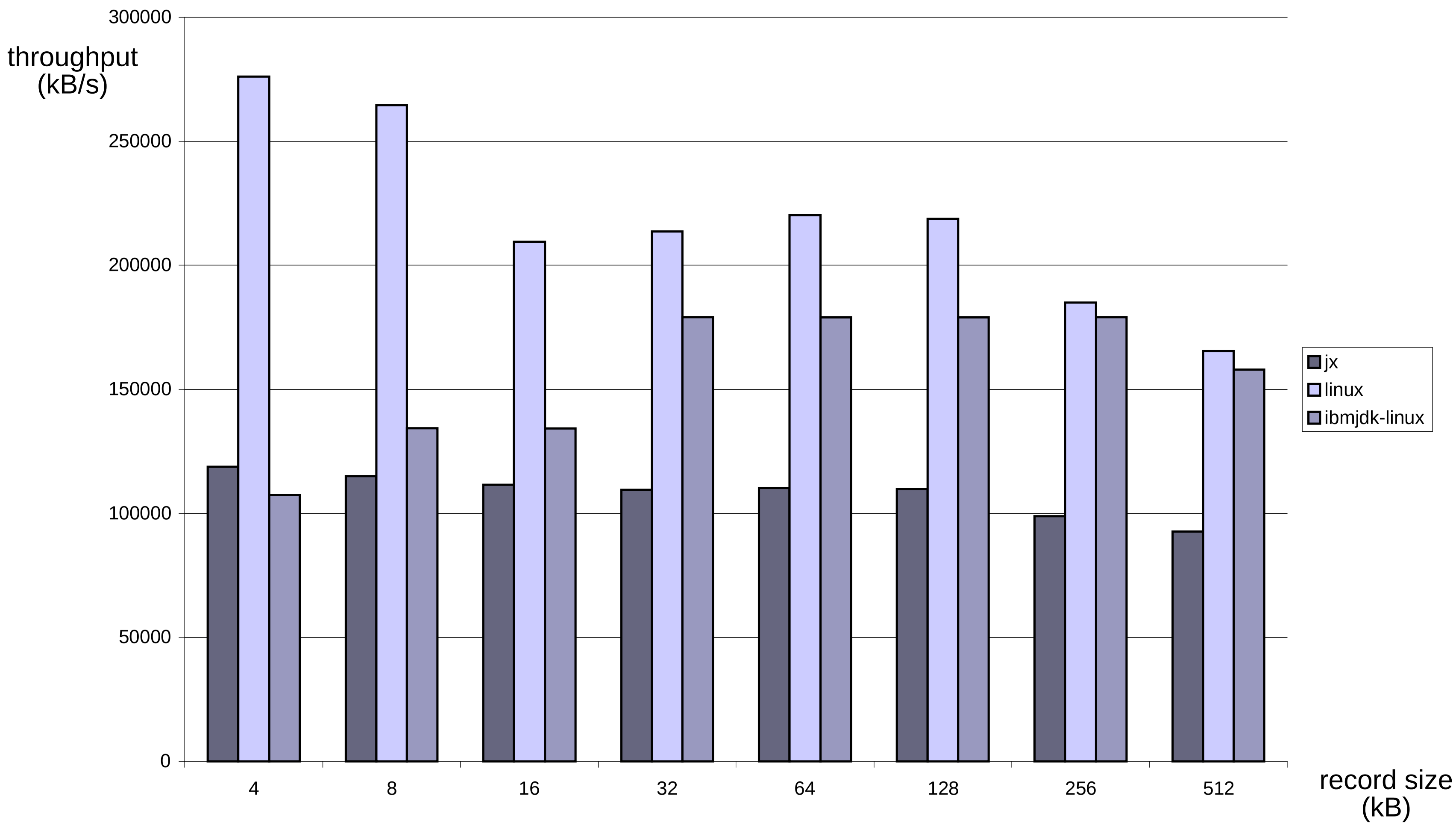
- Iozone-like benchmark: re-read





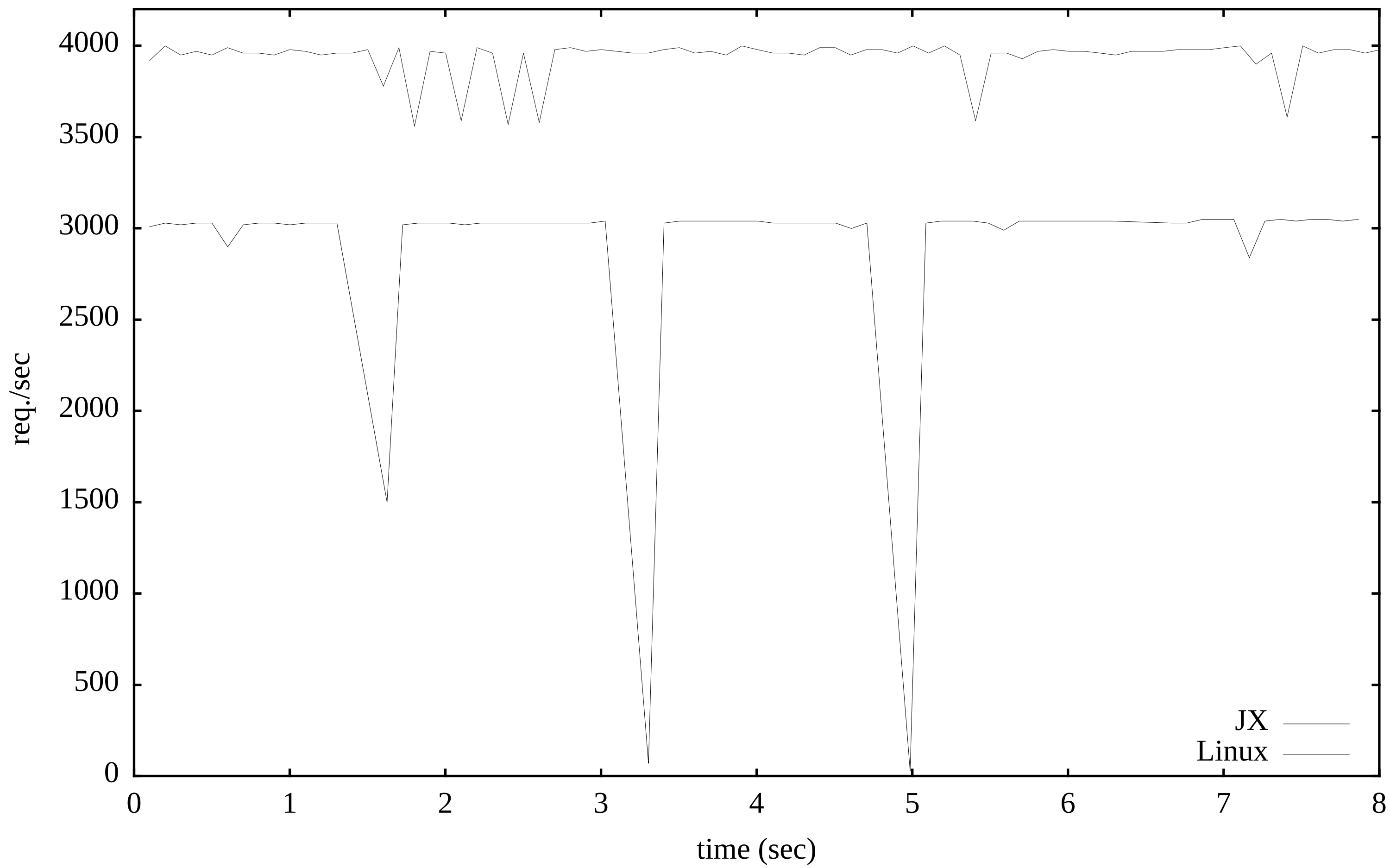
Performance

- Iozone-like benchmark: re-read of a 512 kB file



Performance

- JX as NFS server: *getattr* request rate



Performance:
JX Advantages and Limitations

- Advantages:
  - No expensive border crossings (JNI, OS border)
  - Safe inlining of OS-level code into application code
  - Avoid locking in favour of specialized scheduling
- Limitations:
  - Unavoidable safety checks
  - Semantic gap between stack machine and register machine

Conclusion

- Single Address Space
- Full Protection
  - completely decoupled domains
  - fast communication using portals or memory objects
- Reusable components
- Dynamic extensibility
- Good performance

➜ http://www4.cs.fau.de/Projects/JX/

Status

- JX runs on off-the-shelf PCs
- Drivers for: IDE, Matrox G200, 3COM 3C905B, BT848
- Ext2-FS
- UDP, TCP, RPC (client), NFS (client)
- SMP support

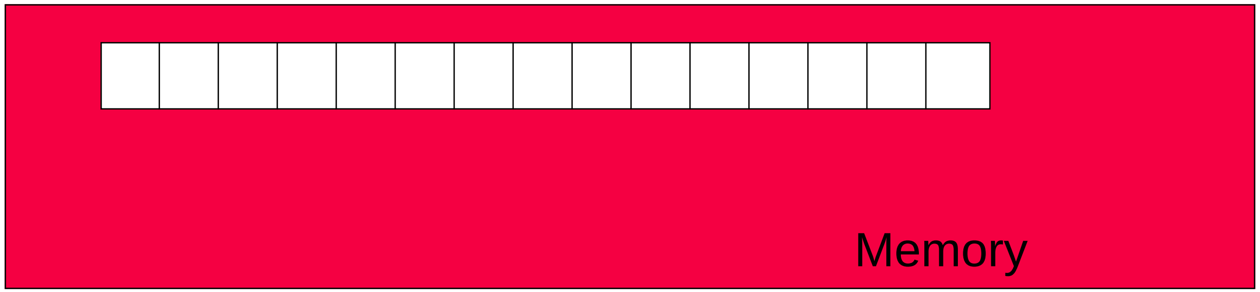
Component Interaction

	co-located components	dis-located components
parameter passing	by reference possible	parameter objects must be copied
thread	can execute in same thread	threads must be switched
resources	caller can be trusted to carefully use resources (e. g., Memory objects)	access rights for Memory objects must be restricted

- Scheduling: scheduling strategy of all co-located components must be compatible
- Execution engine of co-located components must be identical (translated or interpreted)

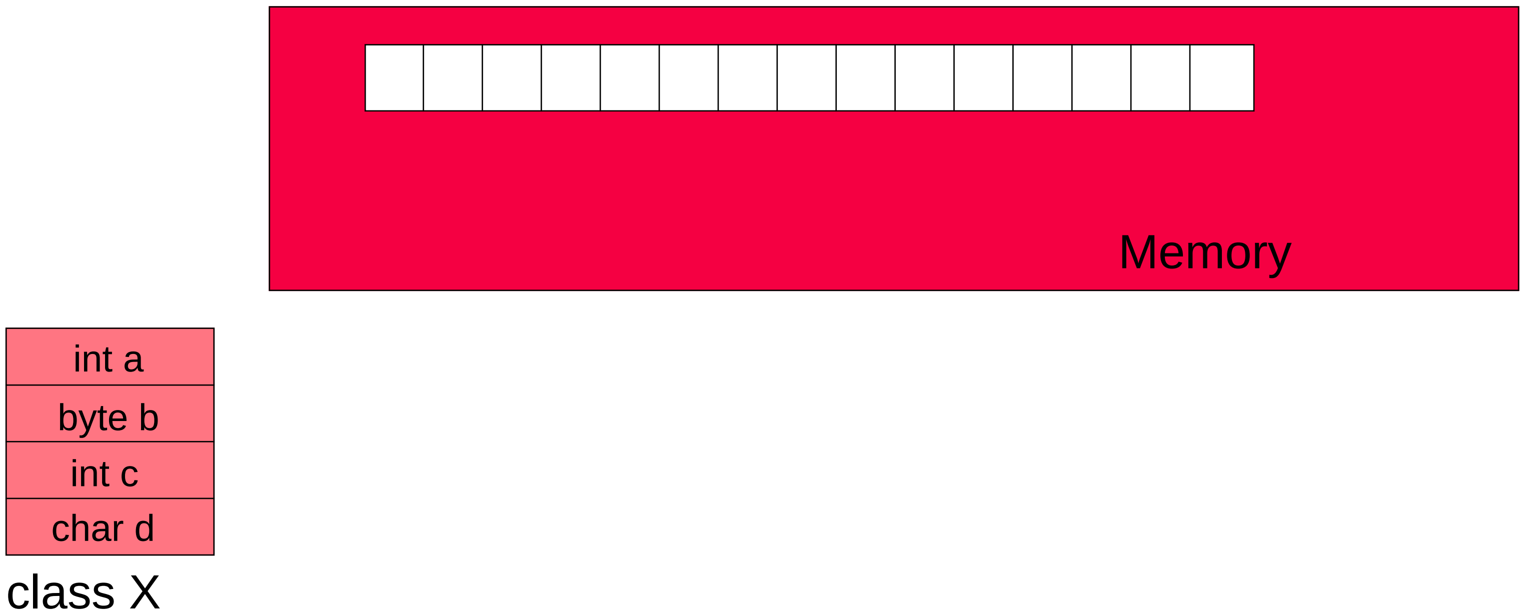
Memory Mapping

- Performance problem: range check
- (partial) solution: map objects to memory range



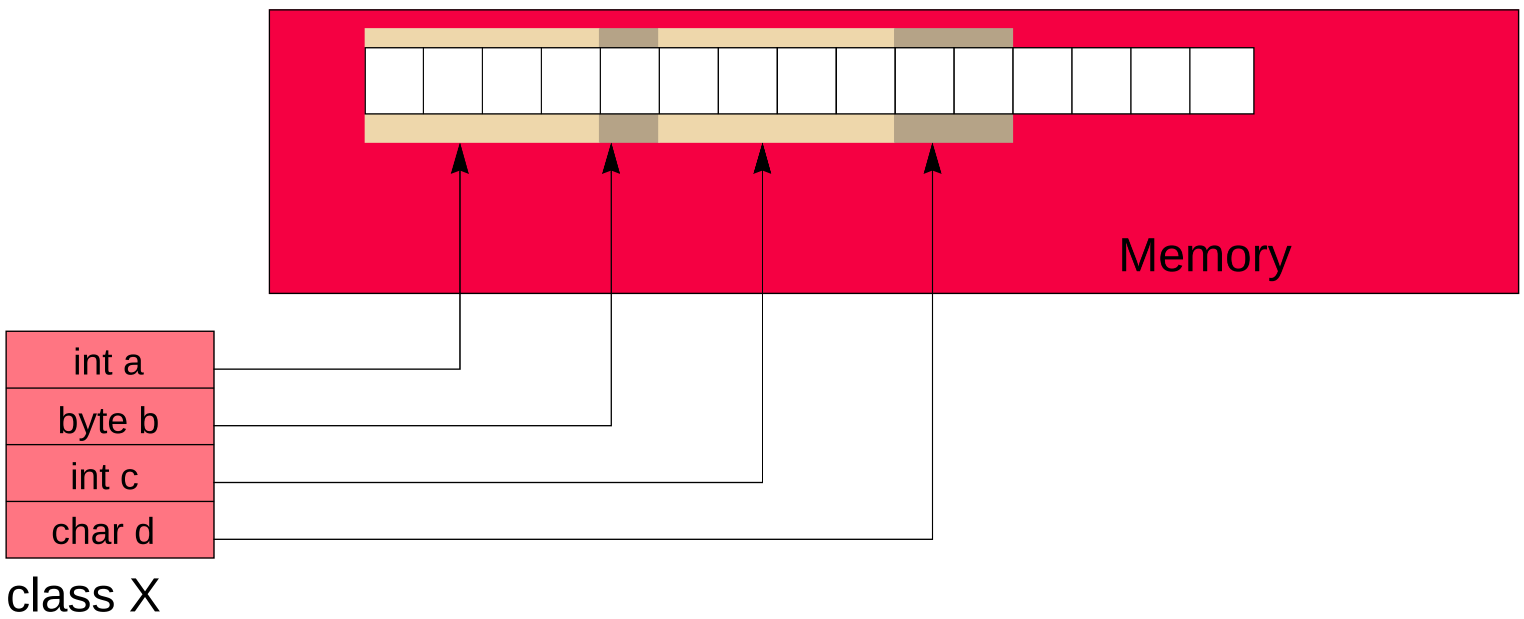
Memory Mapping

- Performance problem: range check
- (partial) solution: map objects to memory range

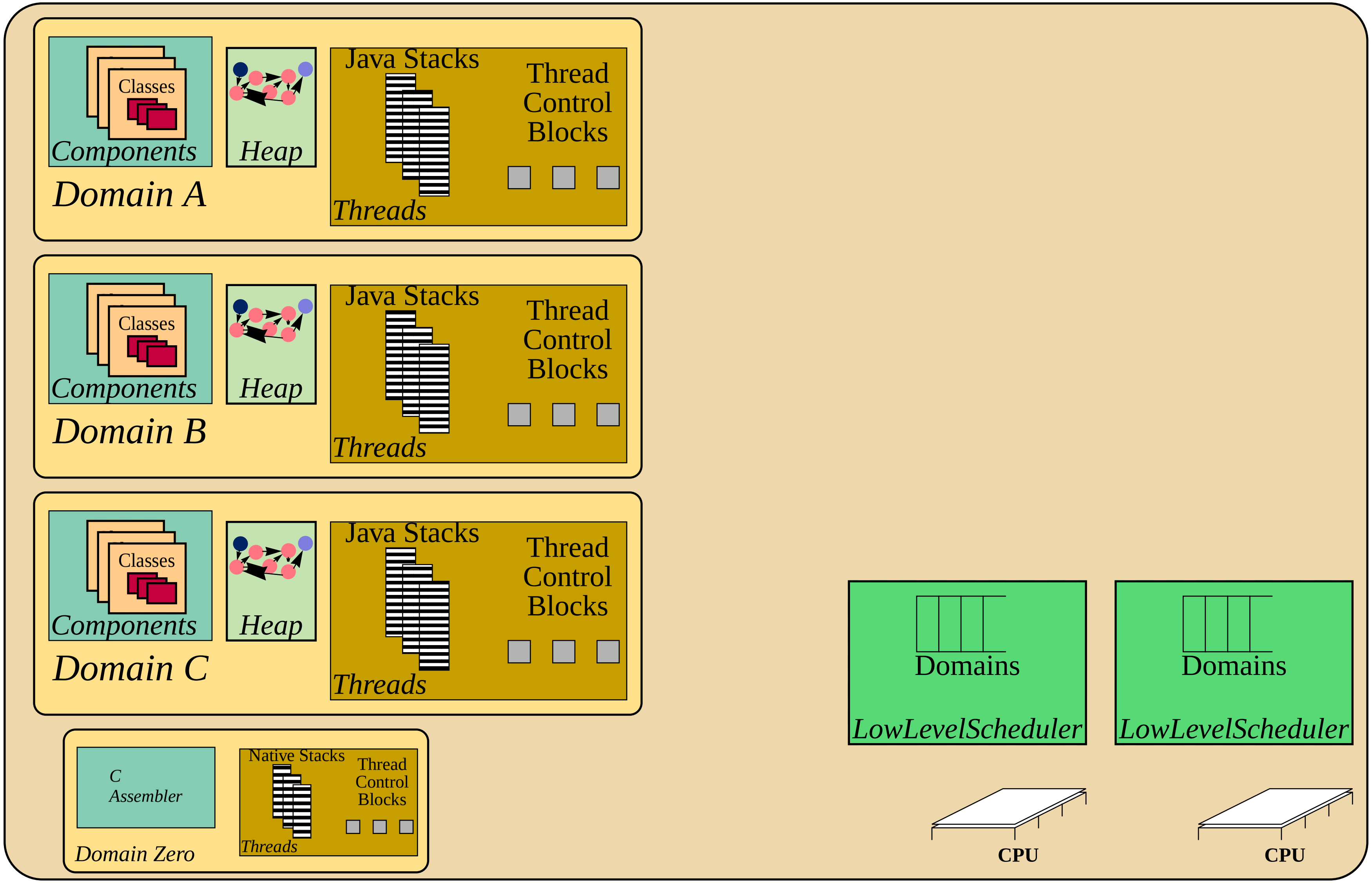


Memory Mapping

- Performance problem: range check
- (partial) solution: map objects to memory range

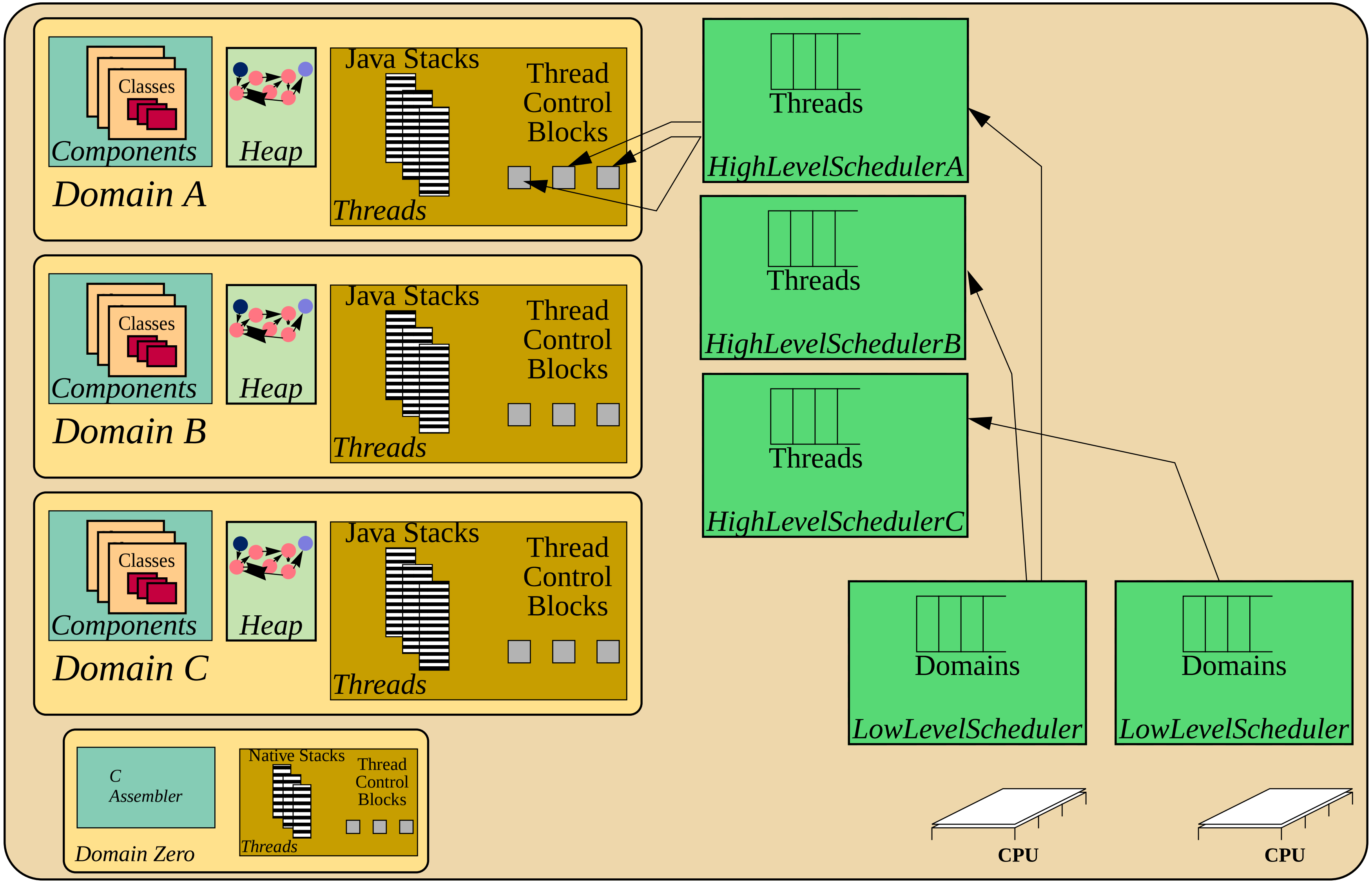


Scheduling



Classes
Components
Heap
Java Stacks
Thread
Control
Blocks
Threads
Domain A
Classes
Components
Heap
Java Stacks
Thread
Control
Blocks
Threads
Domain B
Classes
Components
Heap
Java Stacks
Thread
Control
Blocks
Threads
Domain C
C
Assembler
Native Stacks
Thread
Control
Blocks
Threads
Domain Zero
Domains
LowLevelScheduler
Domains
LowLevelScheduler
CPU
CPU

Scheduling

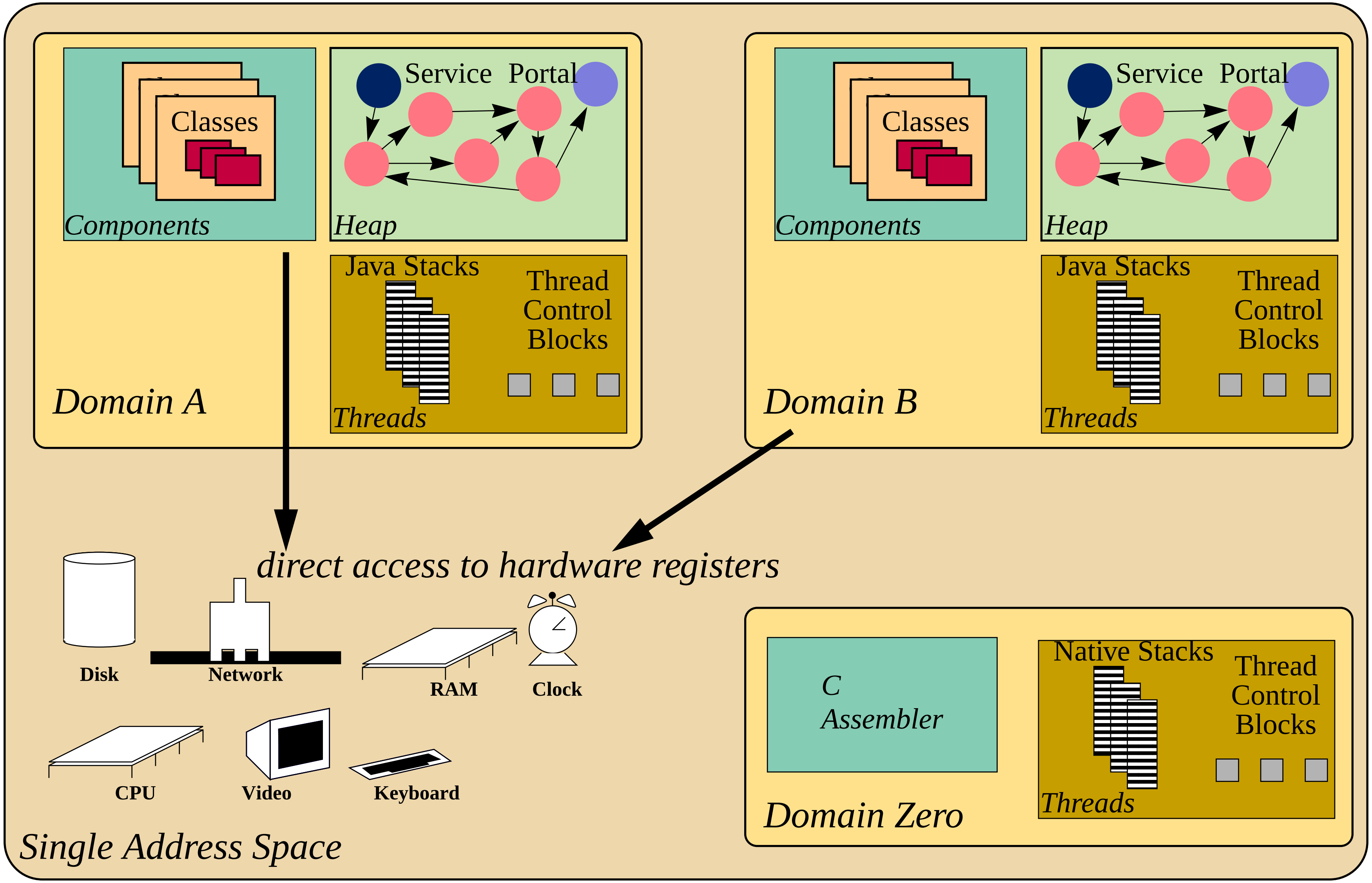


Classes
Components
Heap
Java Stacks
Thread
Control
Blocks
Threads
Domain A
Classes
Components
Heap
Java Stacks
Thread
Control
Blocks
Threads
Domain B
Classes
Components
Heap
Java Stacks
Thread
Control
Blocks
Threads
Domain C
C
Assembler
Native Stacks
Thread
Control
Blocks
Threads
Domain Zero
Threads
HighLevelSchedulerA
Threads
HighLevelSchedulerB
Threads
HighLevelSchedulerC
Domains
LowLevelScheduler
Domains
LowLevelScheduler
CPU
CPU

Resource Management

- Physical resources (CPU, Memory)
  - resource management is supported by DomainZero (but no policy)
- Device-specific physical resources (e. g. network bandwidth)
  - completely managed by a domain
- Virtual resources
  - managed by their respective domains (e. g. TCP port numbers, files)

Hardware Access



Zero-copy using Memory

- immediate processing or buffering possible