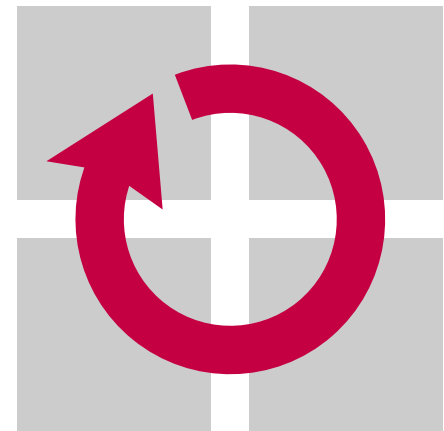


# The JX Operating System: An Overview

---

Michael Golm

University of Erlangen-Nuremberg  
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

---

JX Architecture

*Single Address Space*

*Domain A*

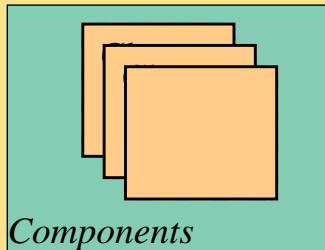
*Domain B*

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

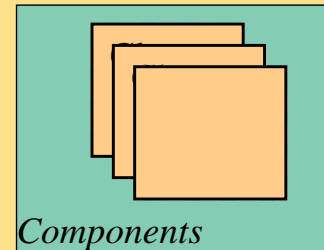
*Single Address Space*

*Domain Zero*

# Components

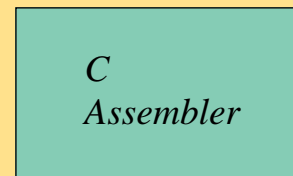


*Domain A*



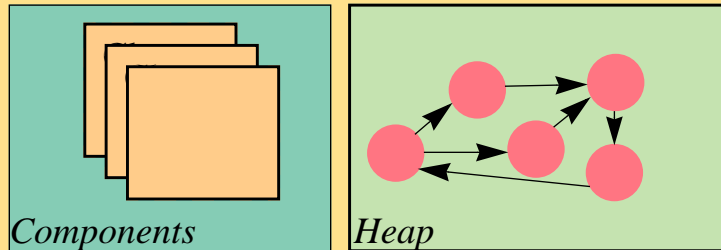
*Domain B*

- 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

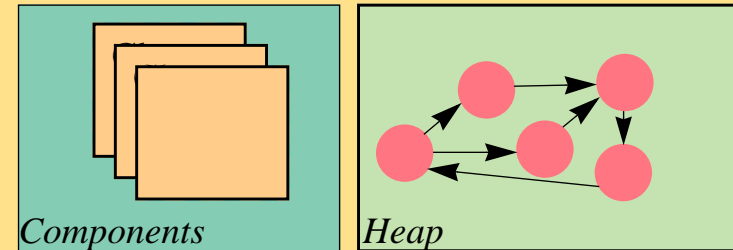


*Domain Zero*

# Objects & Heap

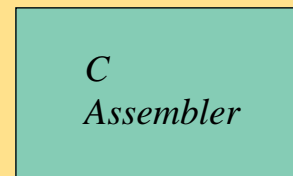


*Domain A*



*Domain B*

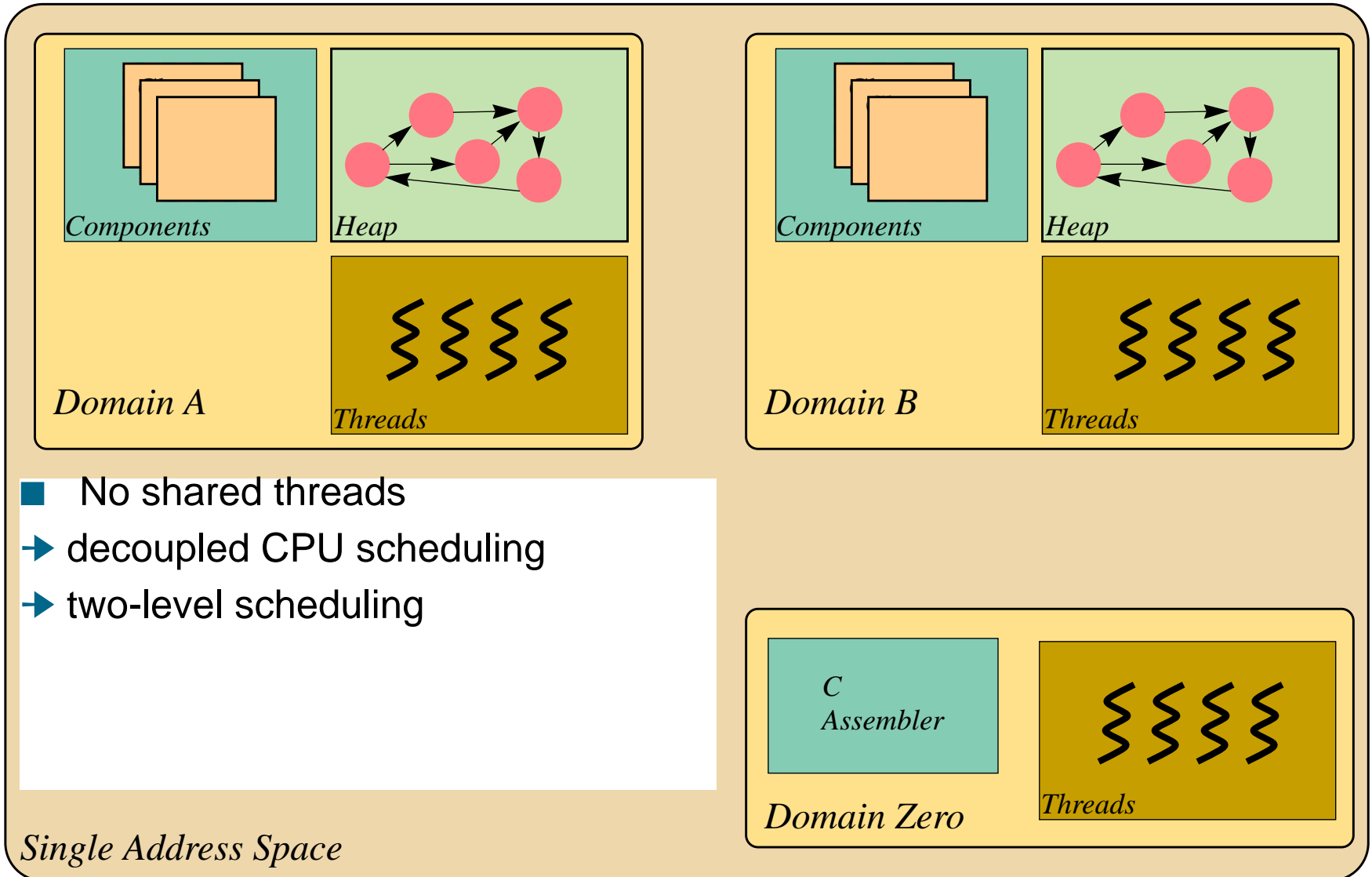
- No shared objects
- ➔ decoupled garbage collection
- ➔ no uncontrolled information flow



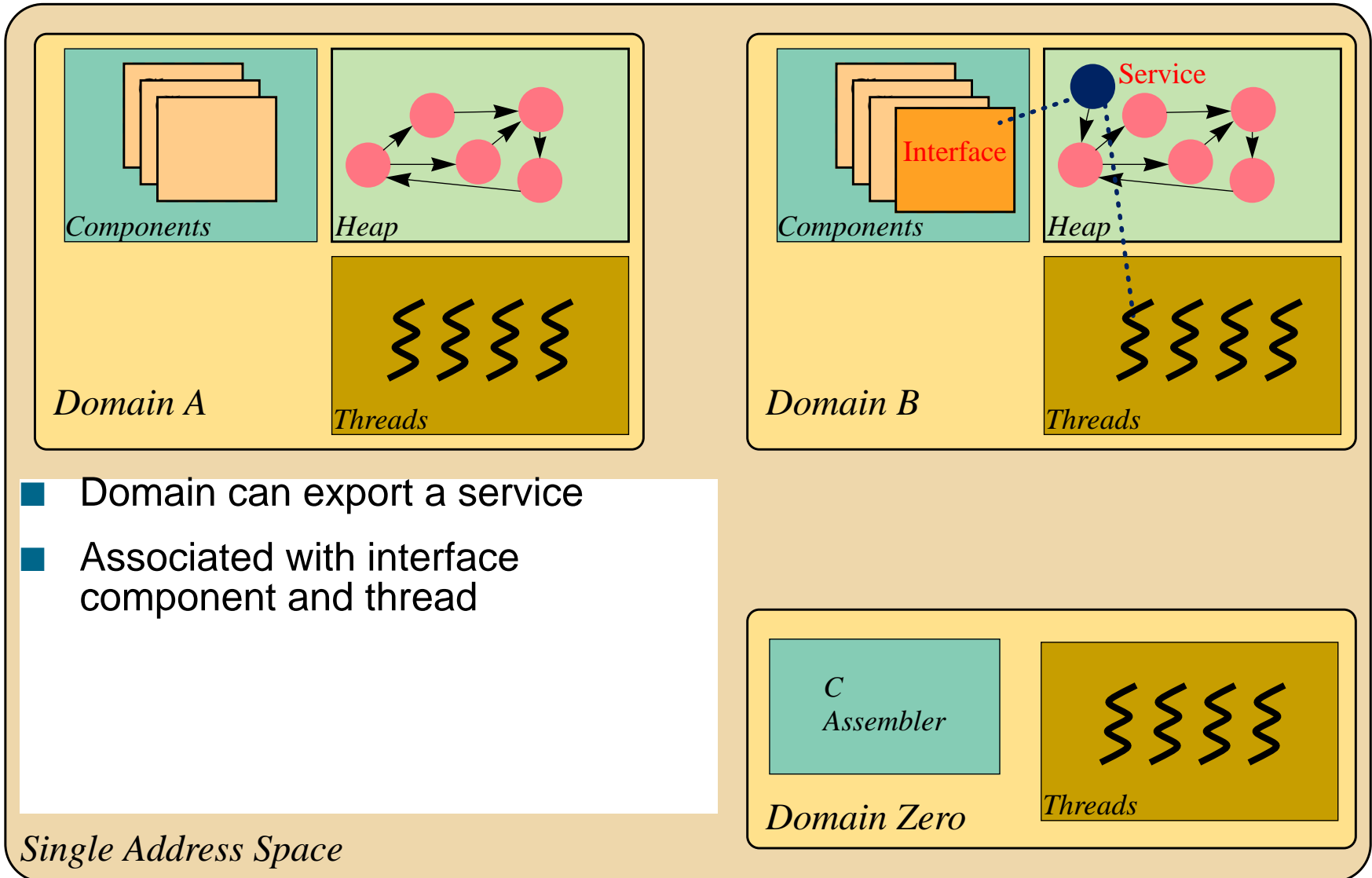
*Domain Zero*

*Single Address Space*

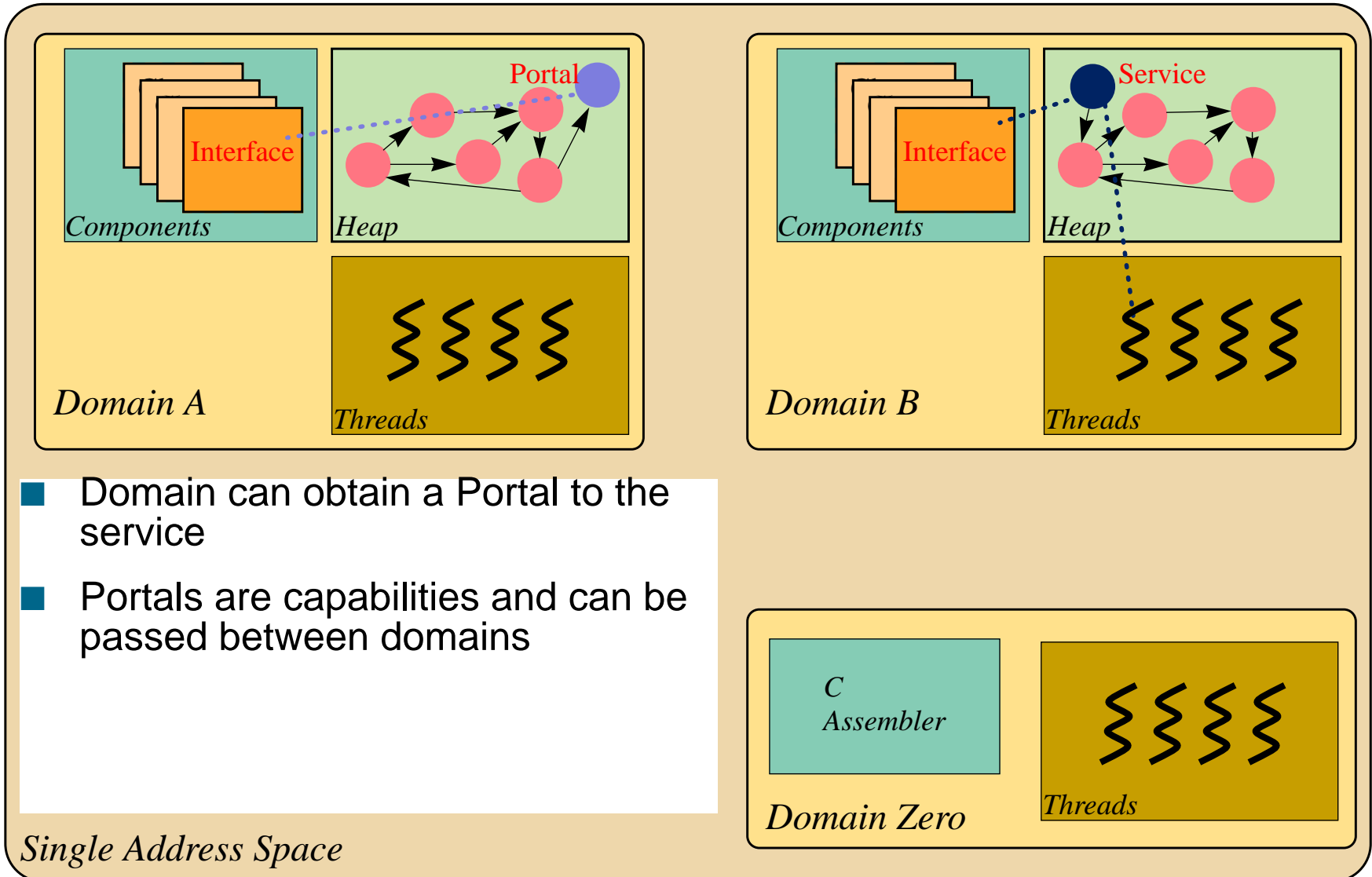
# Threads



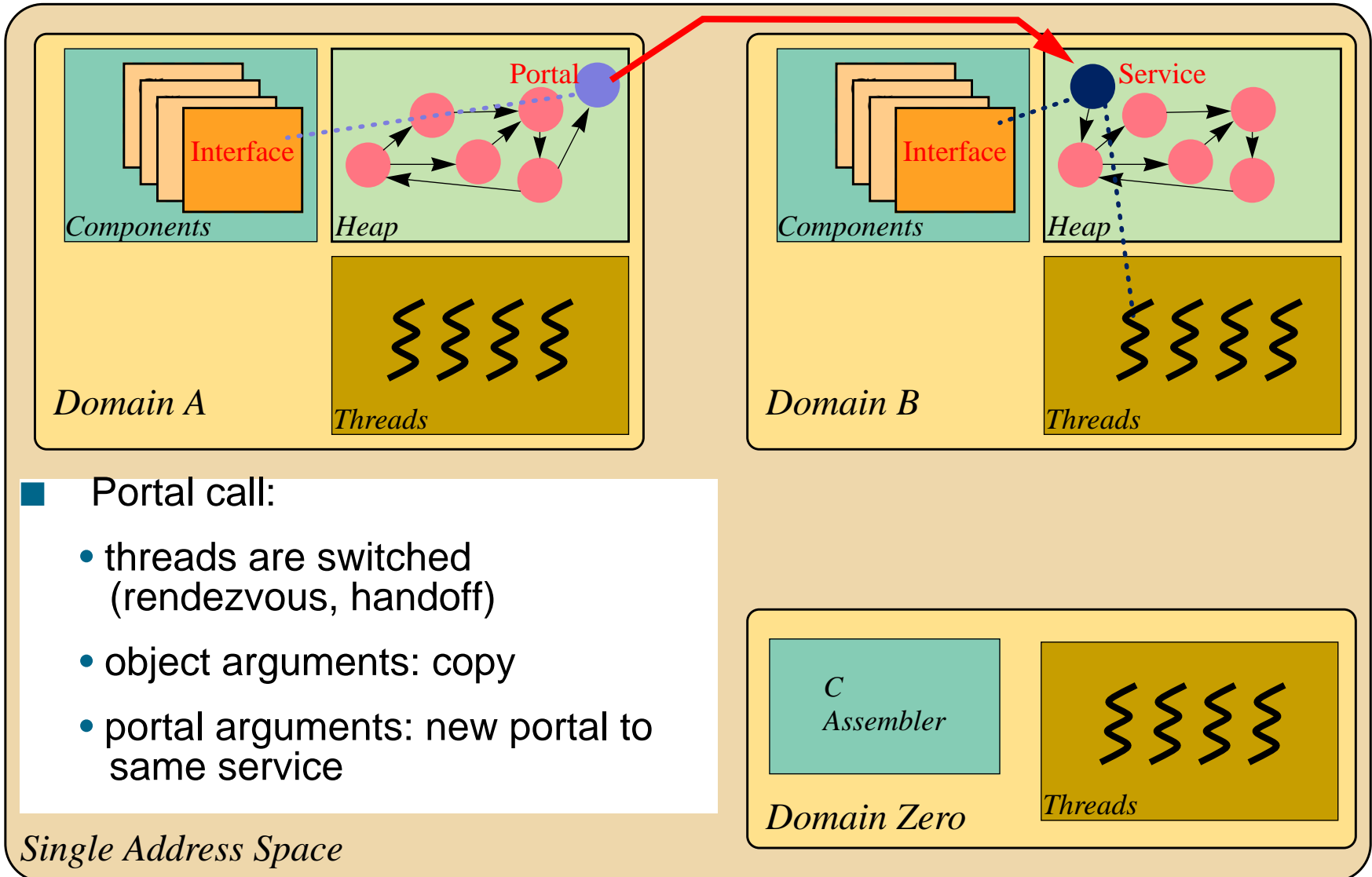
# Communication: Portals



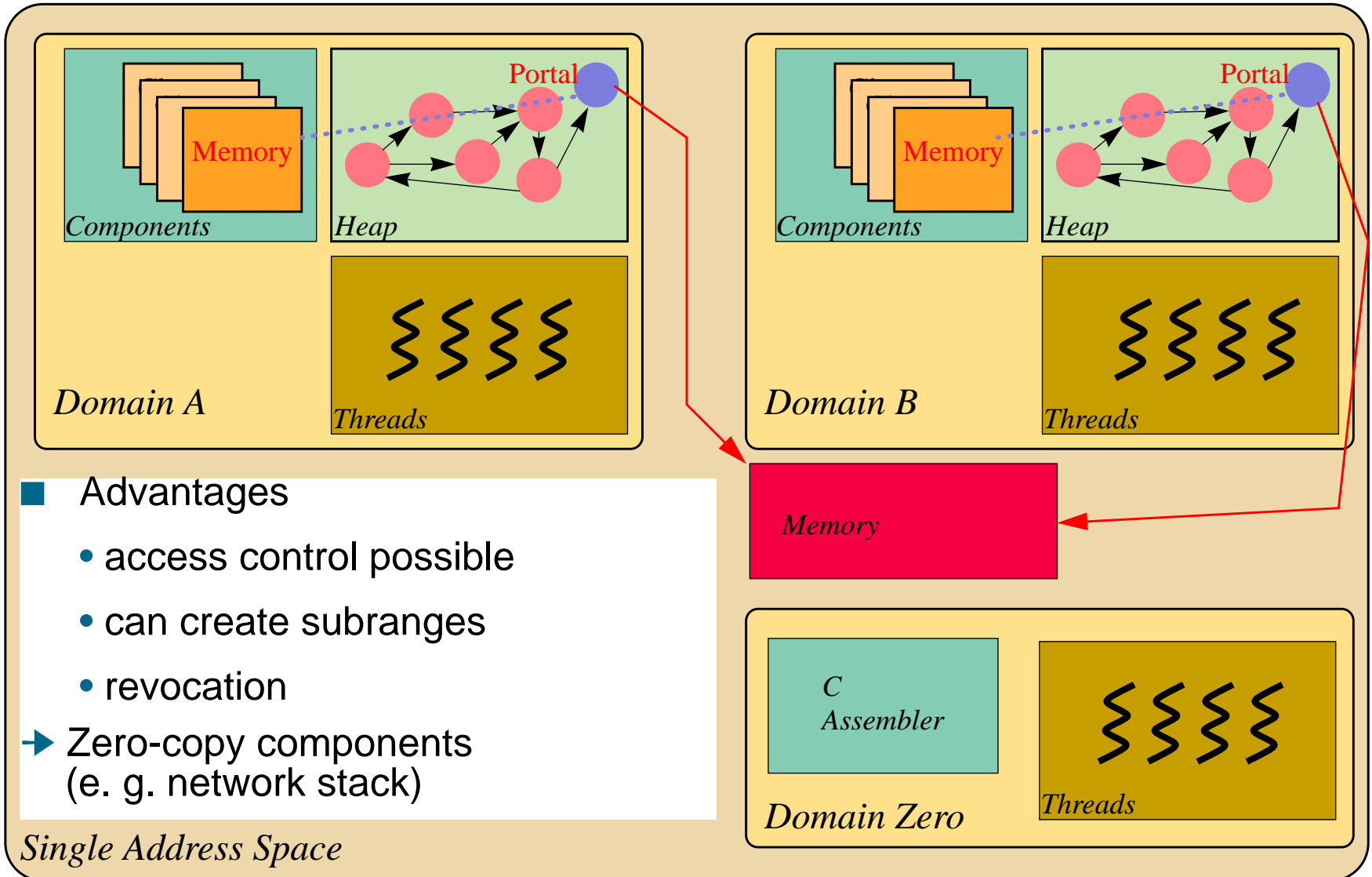
# Communication: Portals



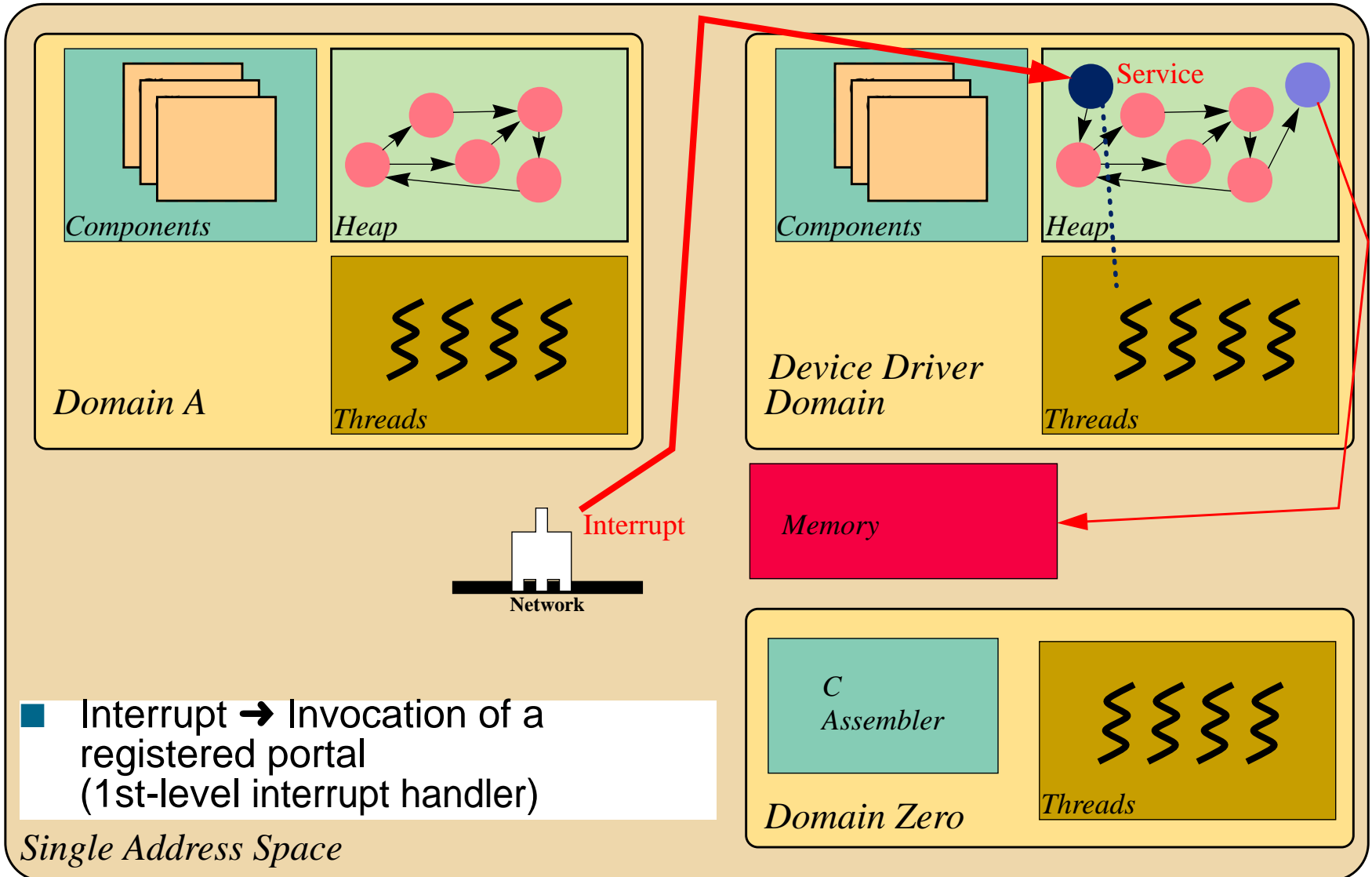
# Communication: Portals



# Communication: Memory



# Device Driver



# Protection: Interrupt Handler

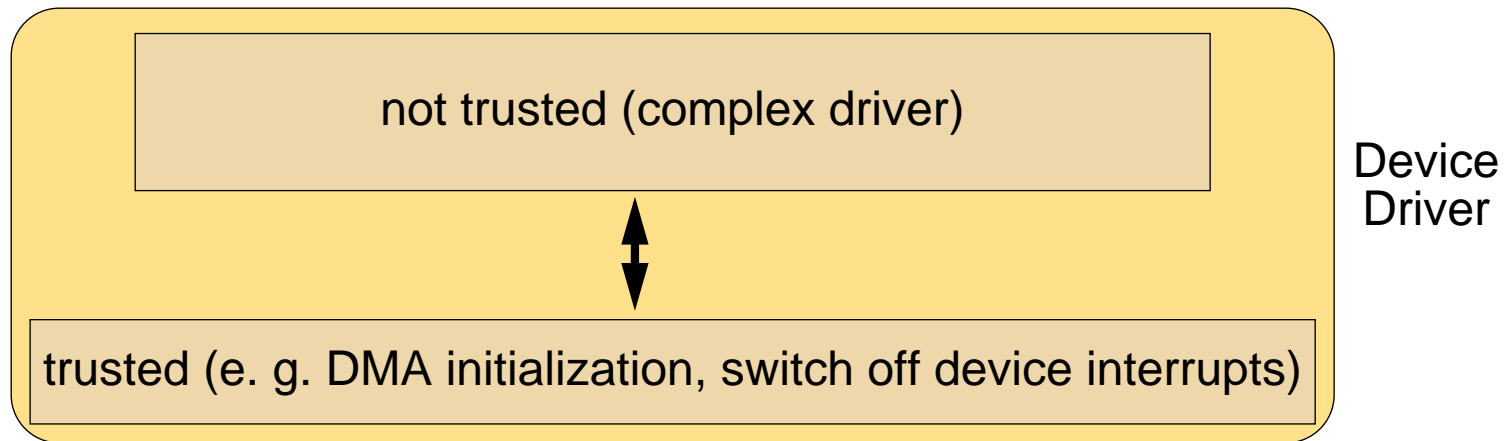
---

- 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)

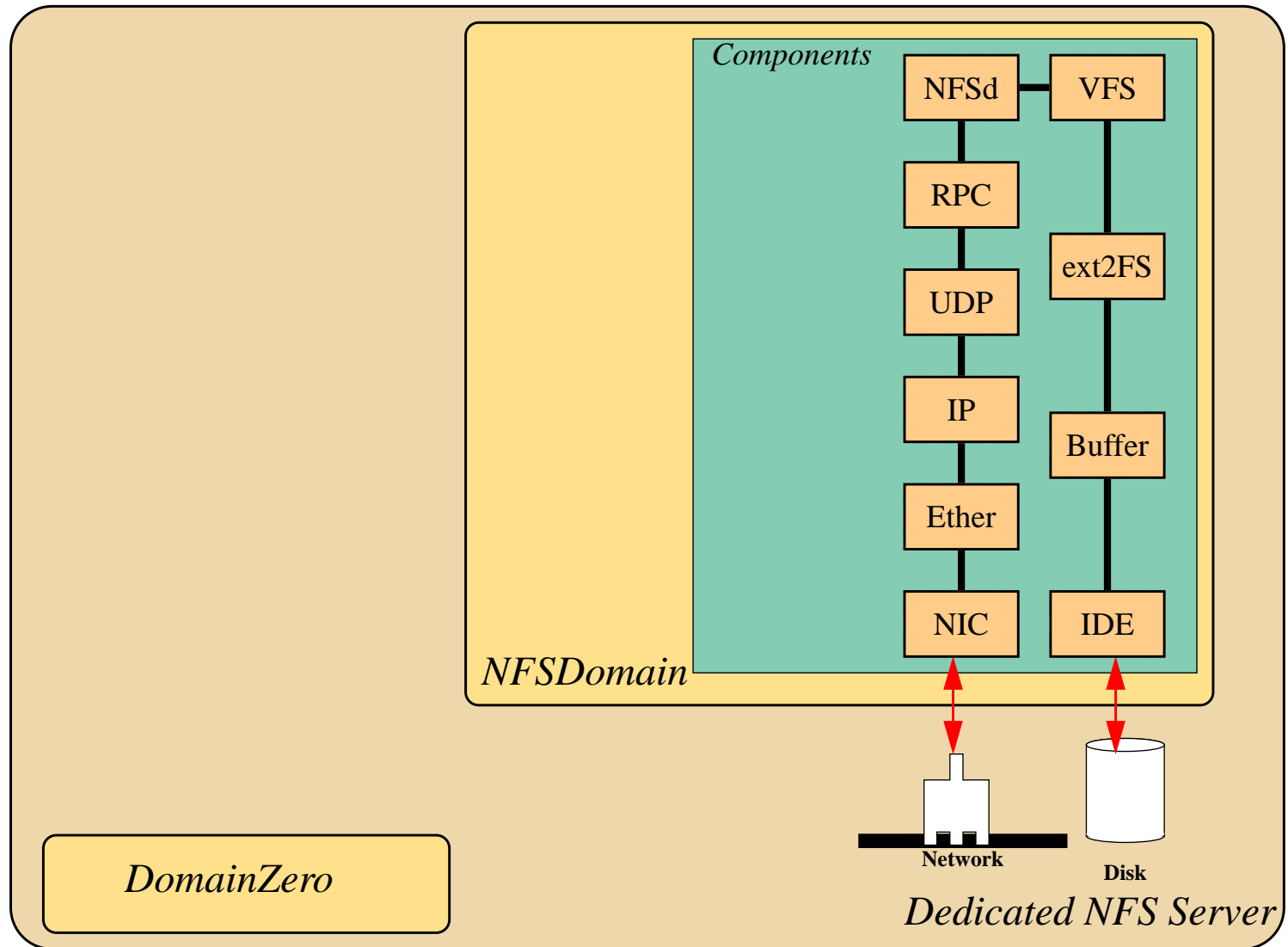
# Protection: Device Driver

---

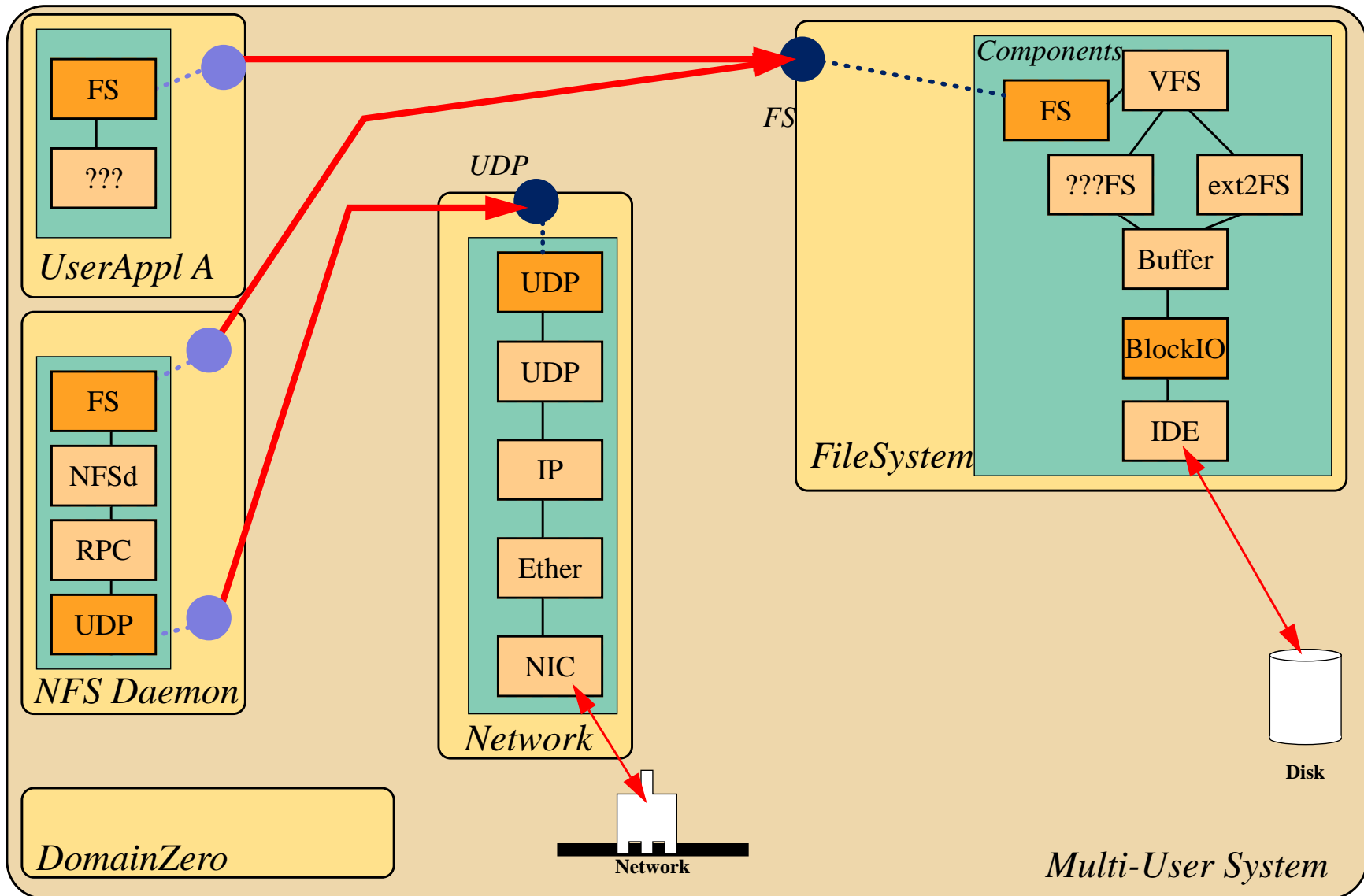
- JX protection is based on type safety and portals
- Some domains can circumvent these mechanisms
  - DomainZero, Translator, Verifier → Trust
  - (some) device drivers



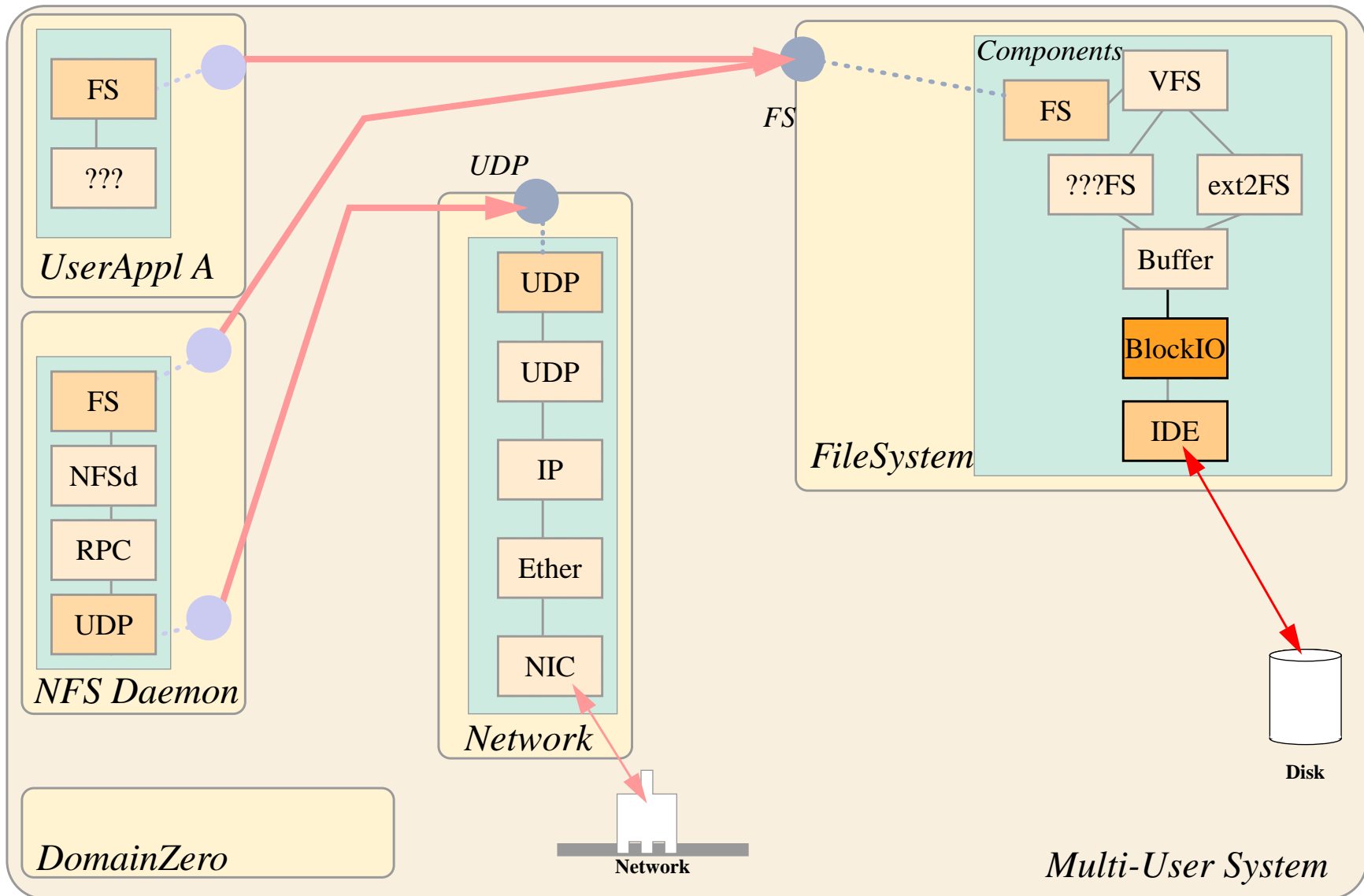
# Building an OS: A Dedicated System



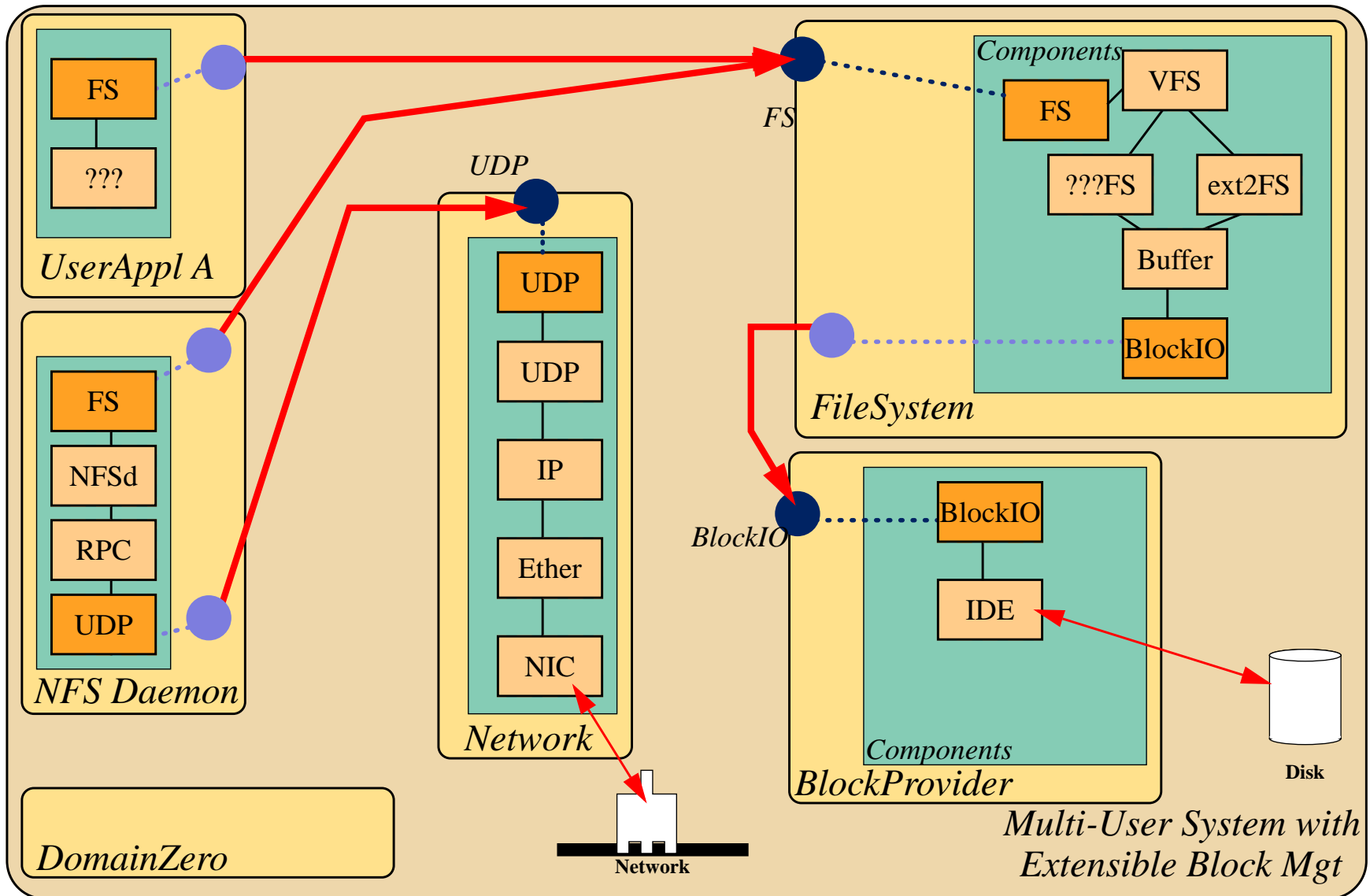
# Building an OS: A Multiuser System



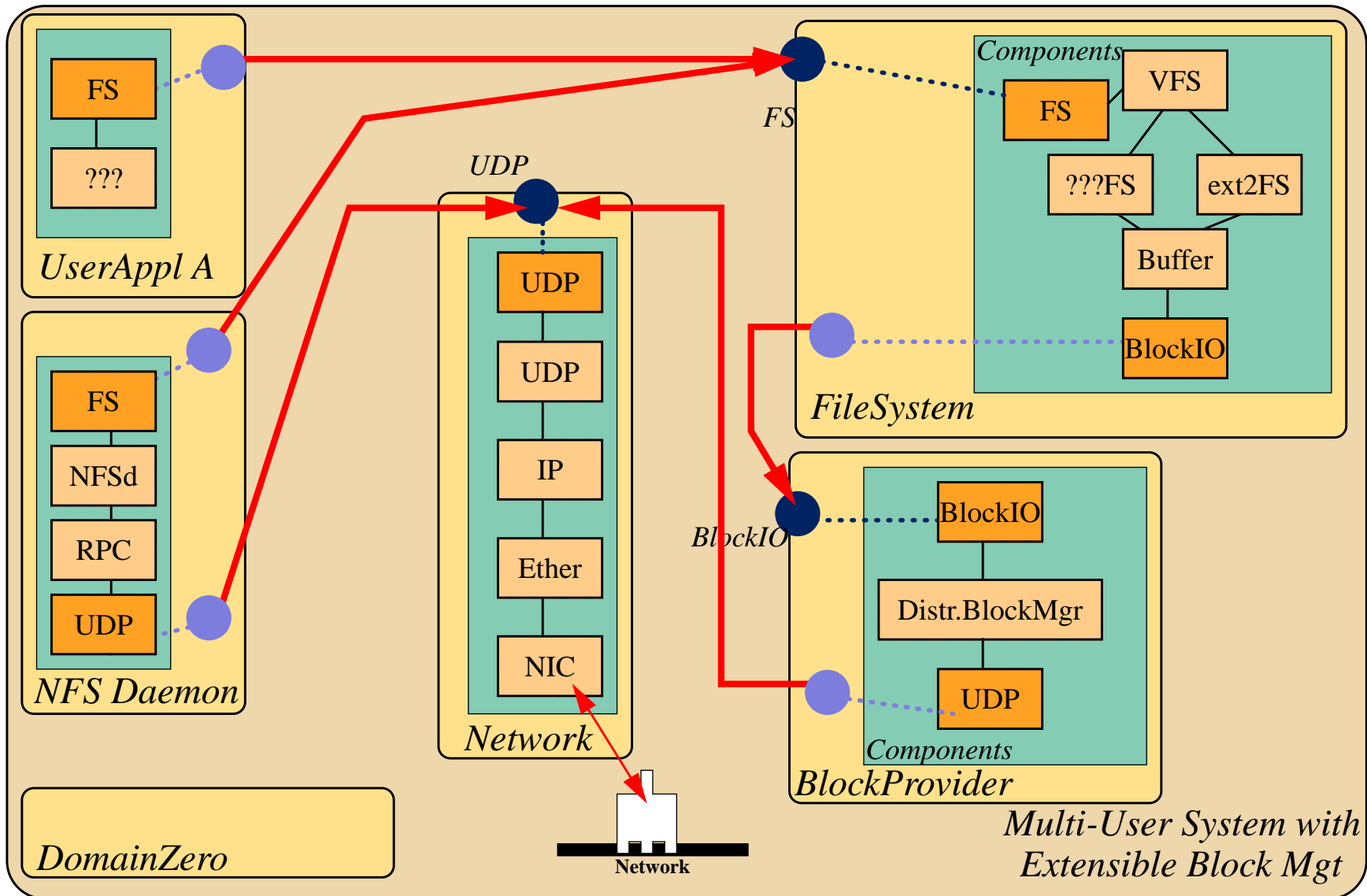
# Building an OS: A Multiuser System



# Building an OS: Extensibility



# Building an OS: Extensibility



# Performance

---

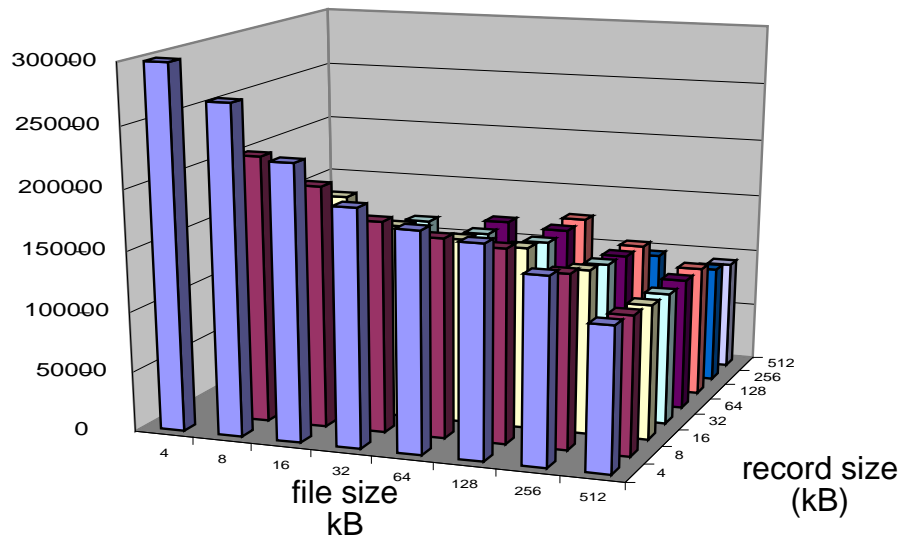
- Hardware:
  - 500MHz PIII, 128MB RAM,
  - IDE: Maxtor 91303D6, 12427MB, 512kB Cache
  - NIC: 3C905B 100 MB/s
  
- IPC:
  - Portal round trip 650 cycles
  - L4 (including RPC stubs): 800 cycles
  - KaffeOS: 27270 cycles

# Performance

## ■ lozone-like benchmark: re-read

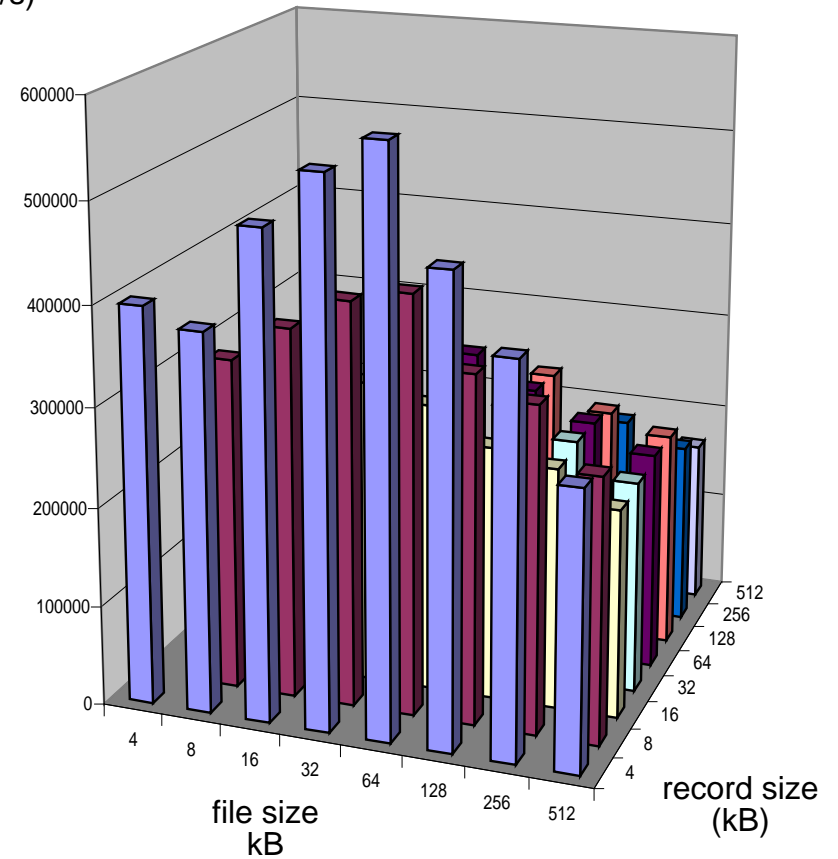
JX

throughput  
(kB/s)



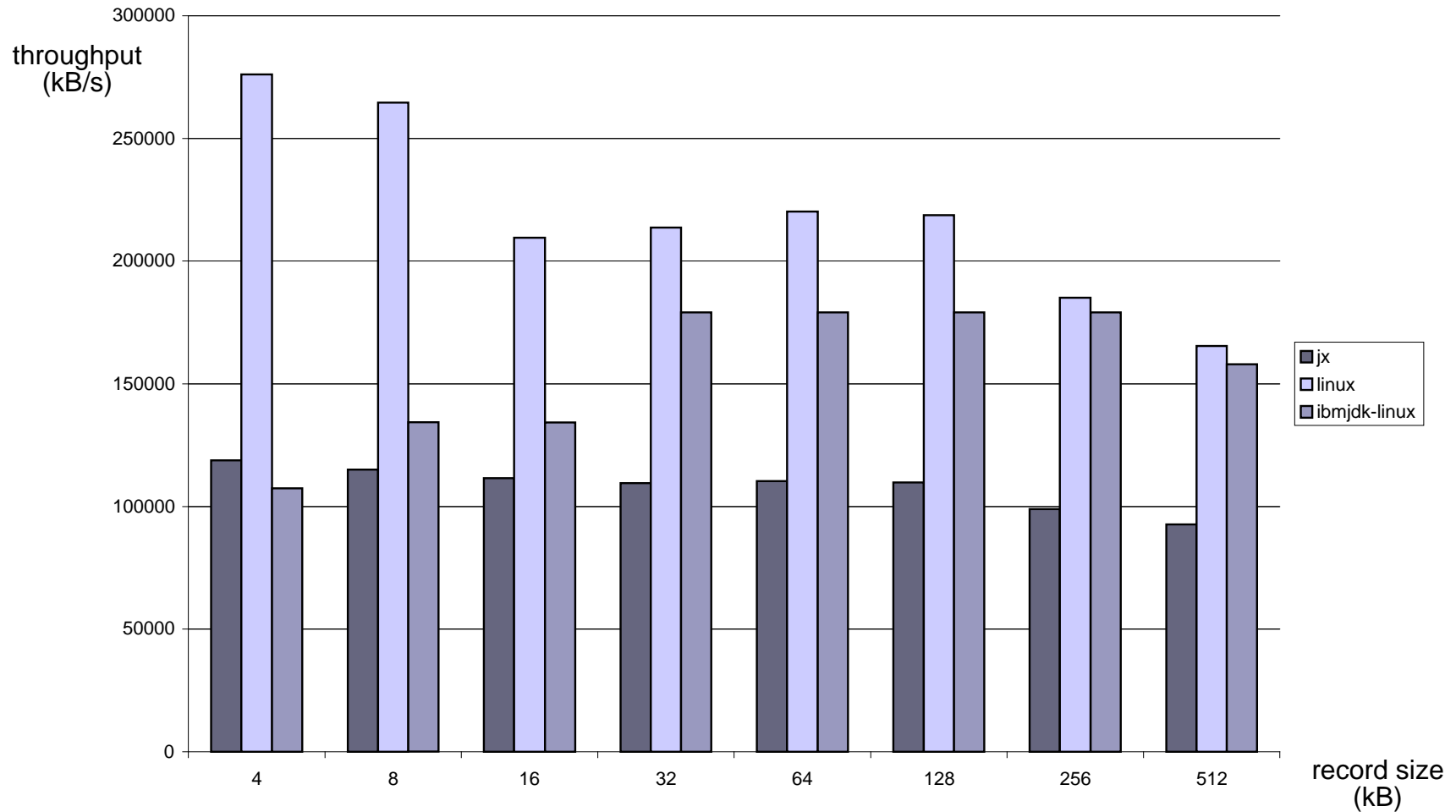
Linux

throughput  
(kB/s)



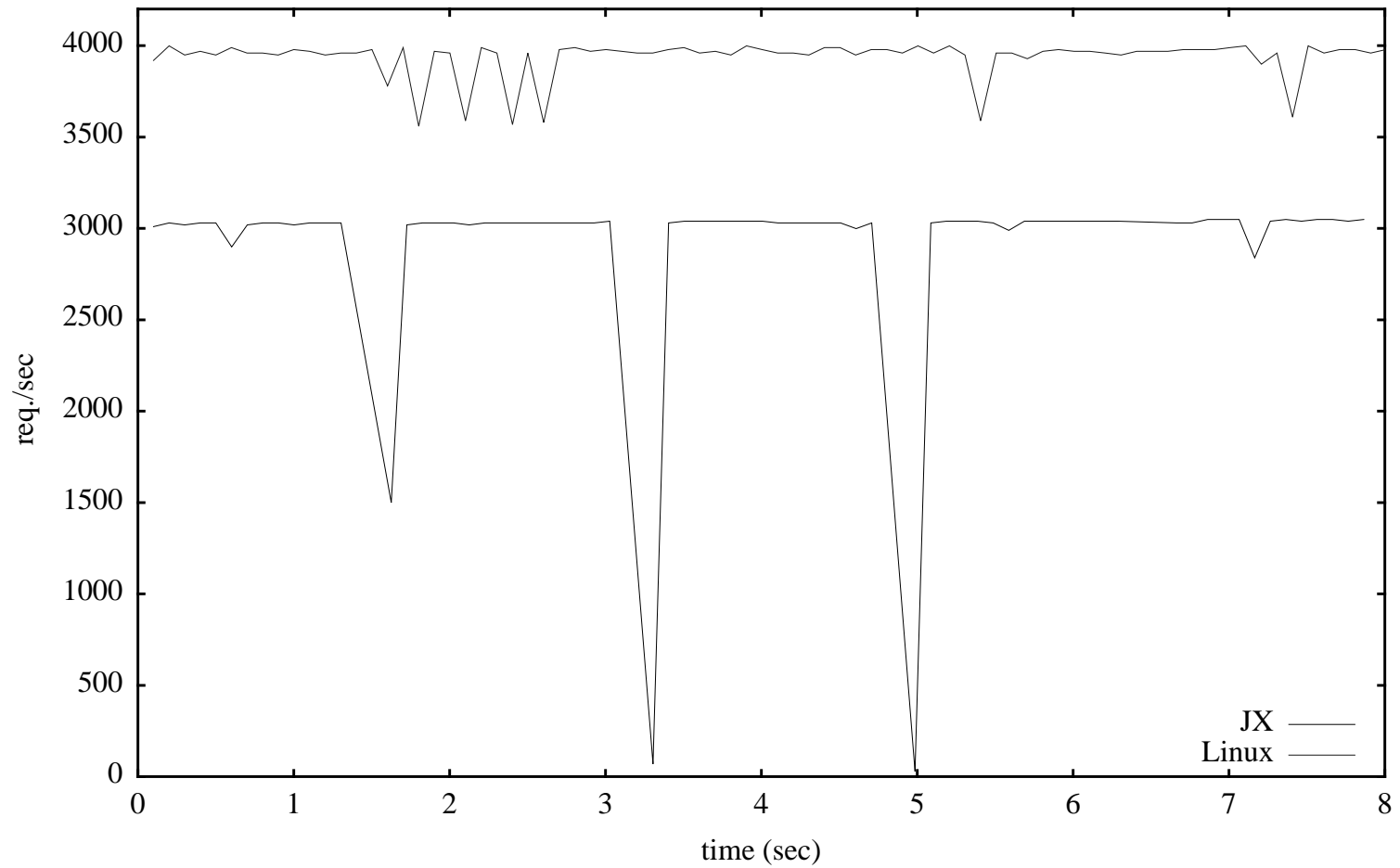
# Performance

## ■ lozone-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/>