

Sistemas Distribuídos

José Orlando Pereira

Departamento de Informática
Universidade do Minho



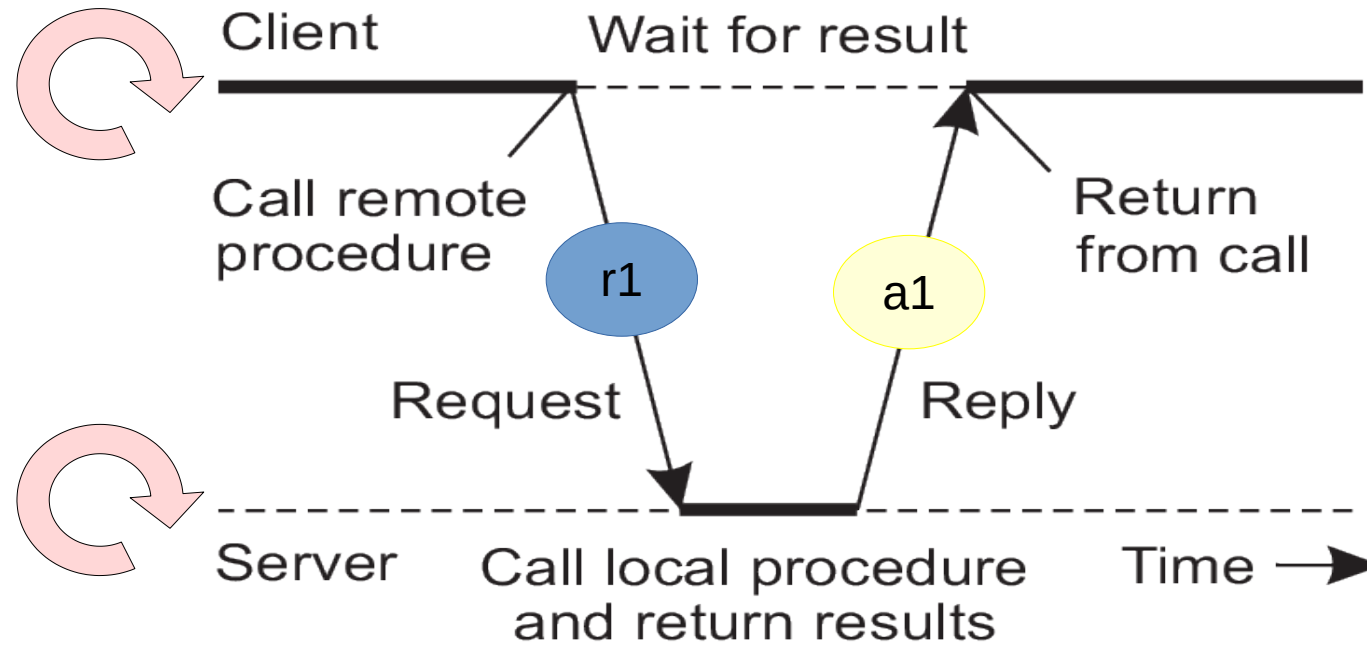
Operating Systems 101

- Threads:
 - Processor context + stack
- Process:
 - Threads + Memory context + OS context (open files, sockets, ...)
- Operating system:
 - Named shared resources (files, ports, ...)

Motivation

- A distributed system adds two new concerns:
 - Where is the thread executing?
 - How does it cross process and host boundaries?

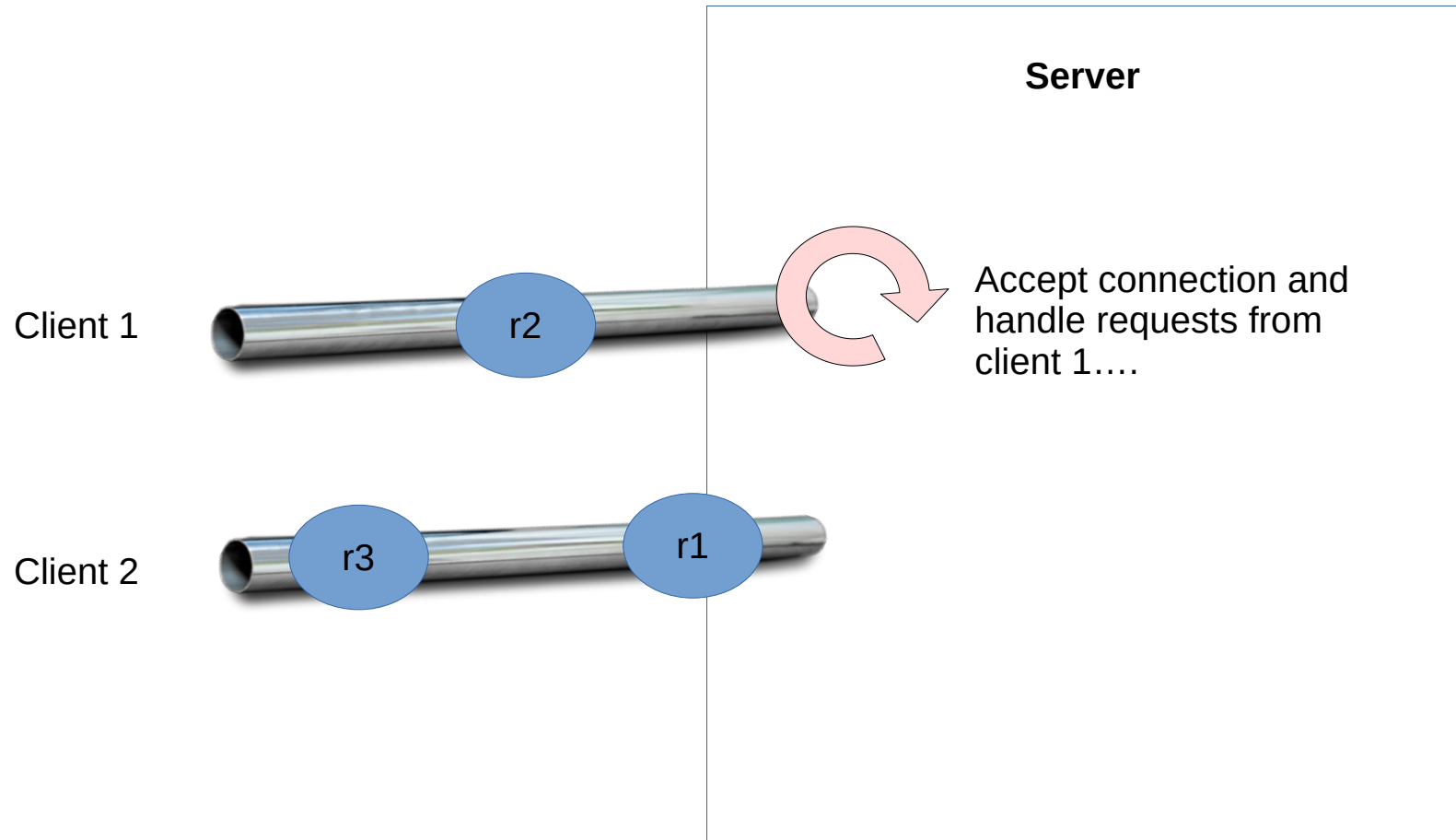
Example: Remote procedure call (RPC)



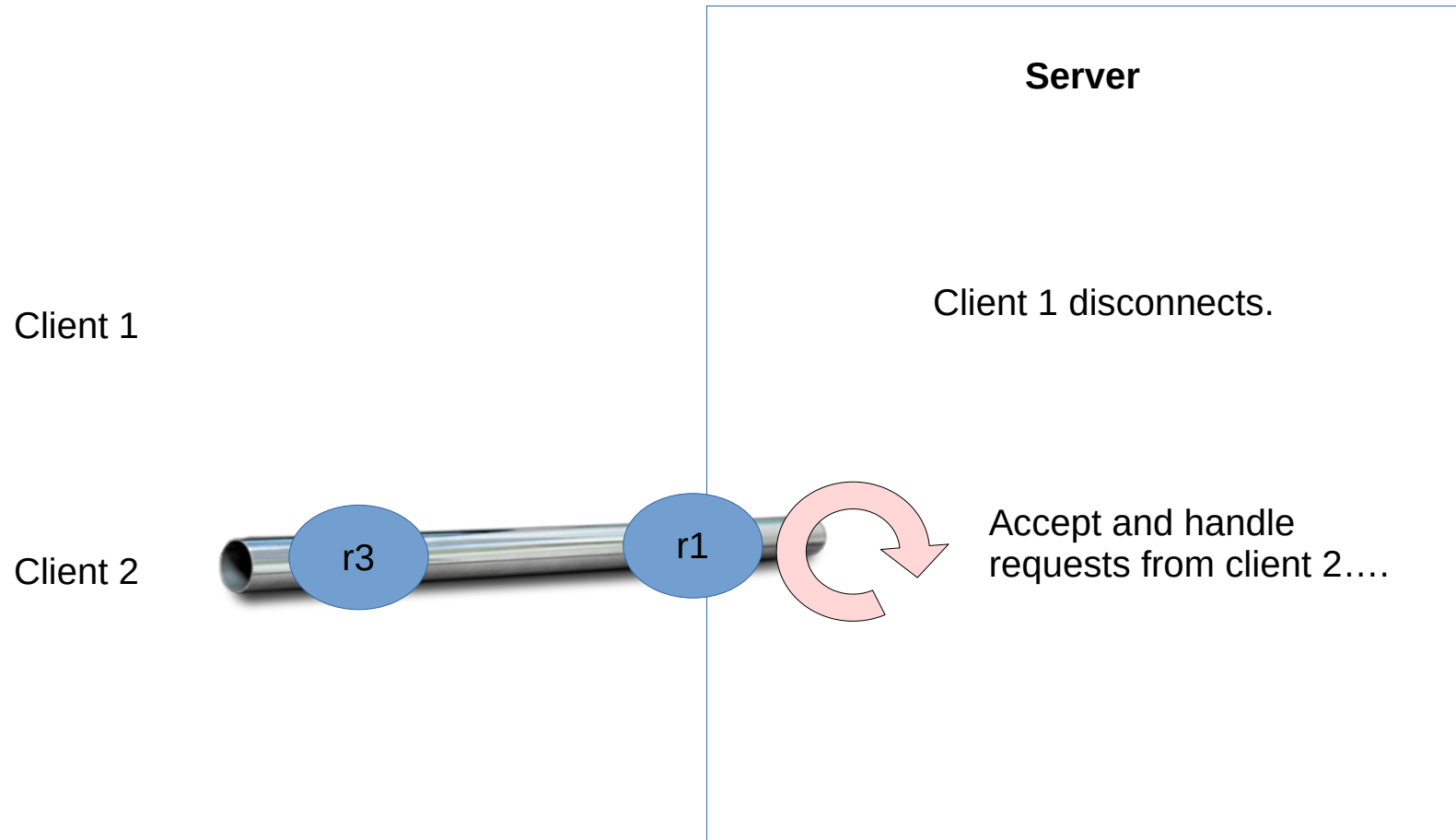
Example: Remote procedure call (RPC)

- Can be regarded as the client thread migrating to the server to perform a task
- Requires the cooperation of multiple operating system threads at both sides
 - Interacts with connection management

Single-threaded



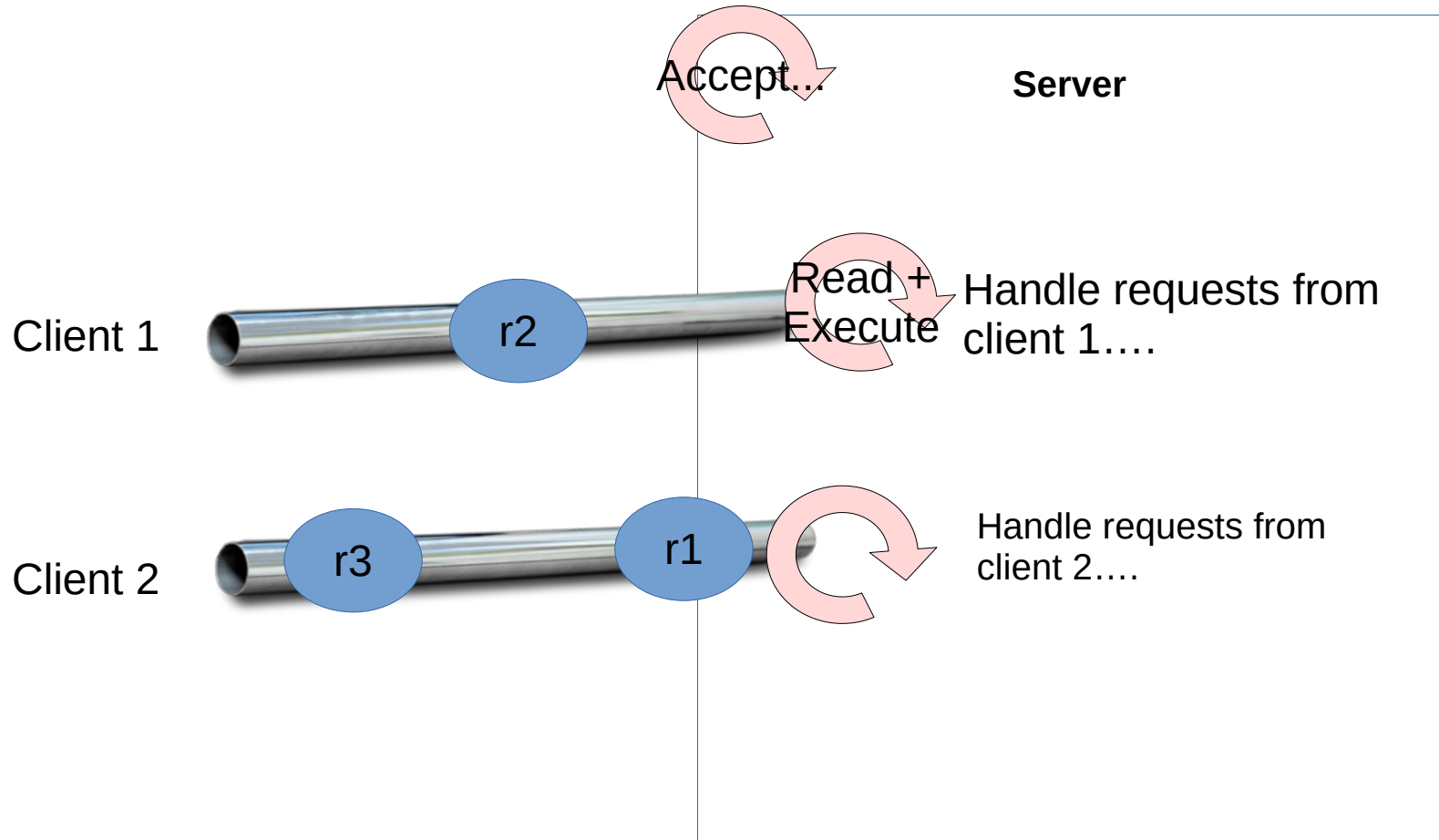
Single-threaded



Single-threaded

- Not useful as a general model, as a server handles only one connection
- Can be used with:
 - Connection-less communication protocols
 - Meta-servers

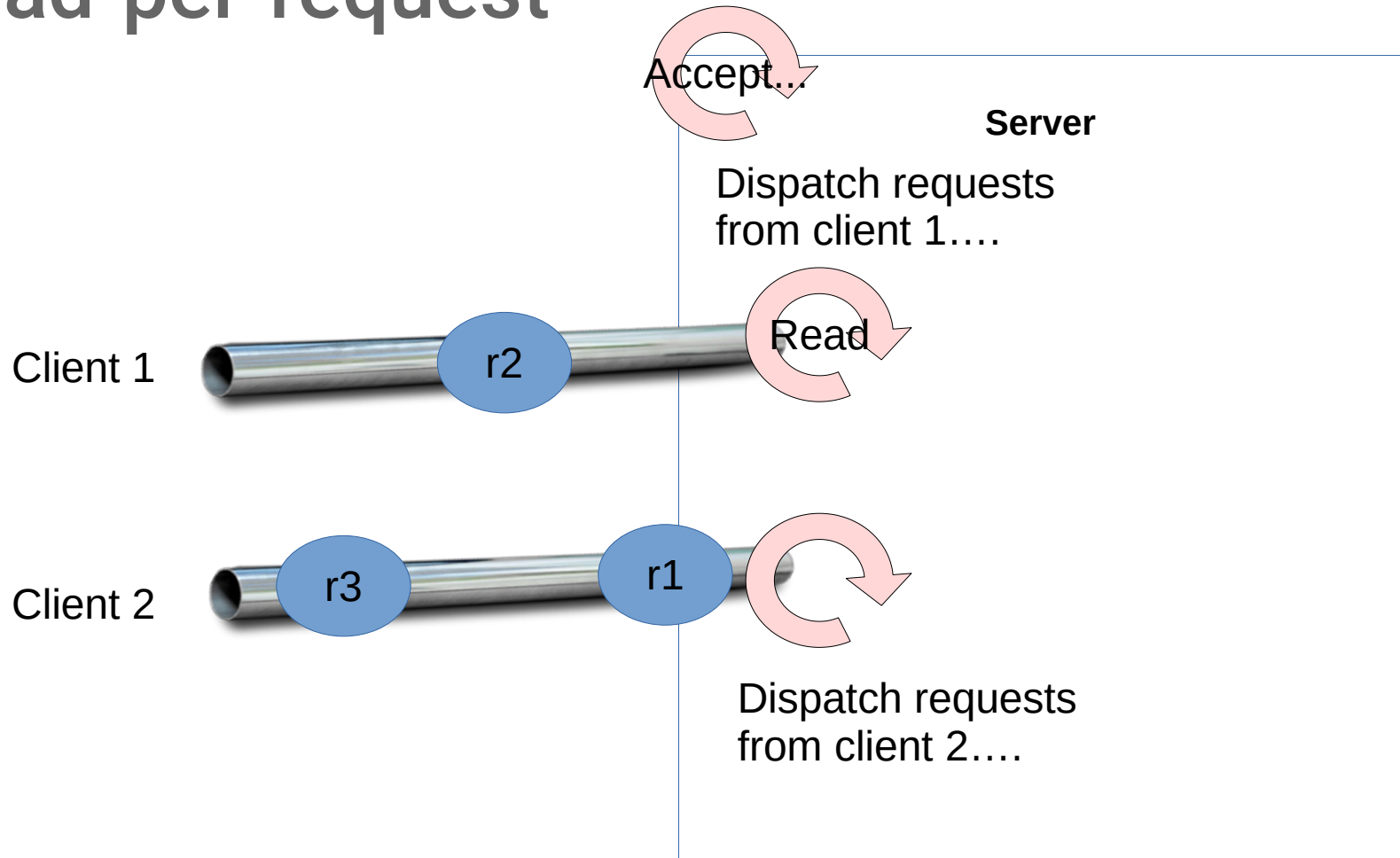
Thread-per-connection



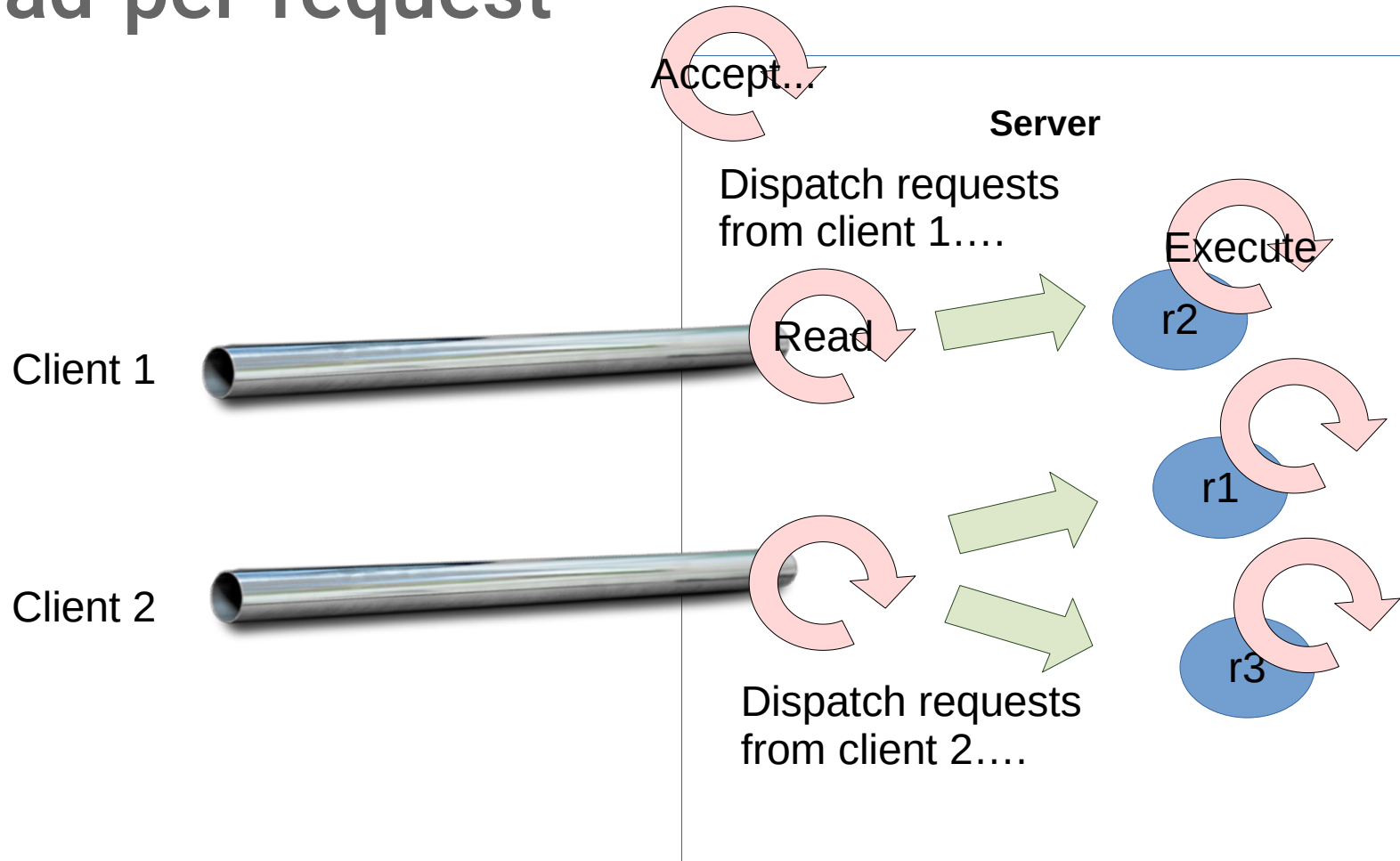
Thread-per-connection

- Generally useful / typical model
- Adequate to single-threaded clients
- Adequate for fast non-blocking requests

Thread-per-request



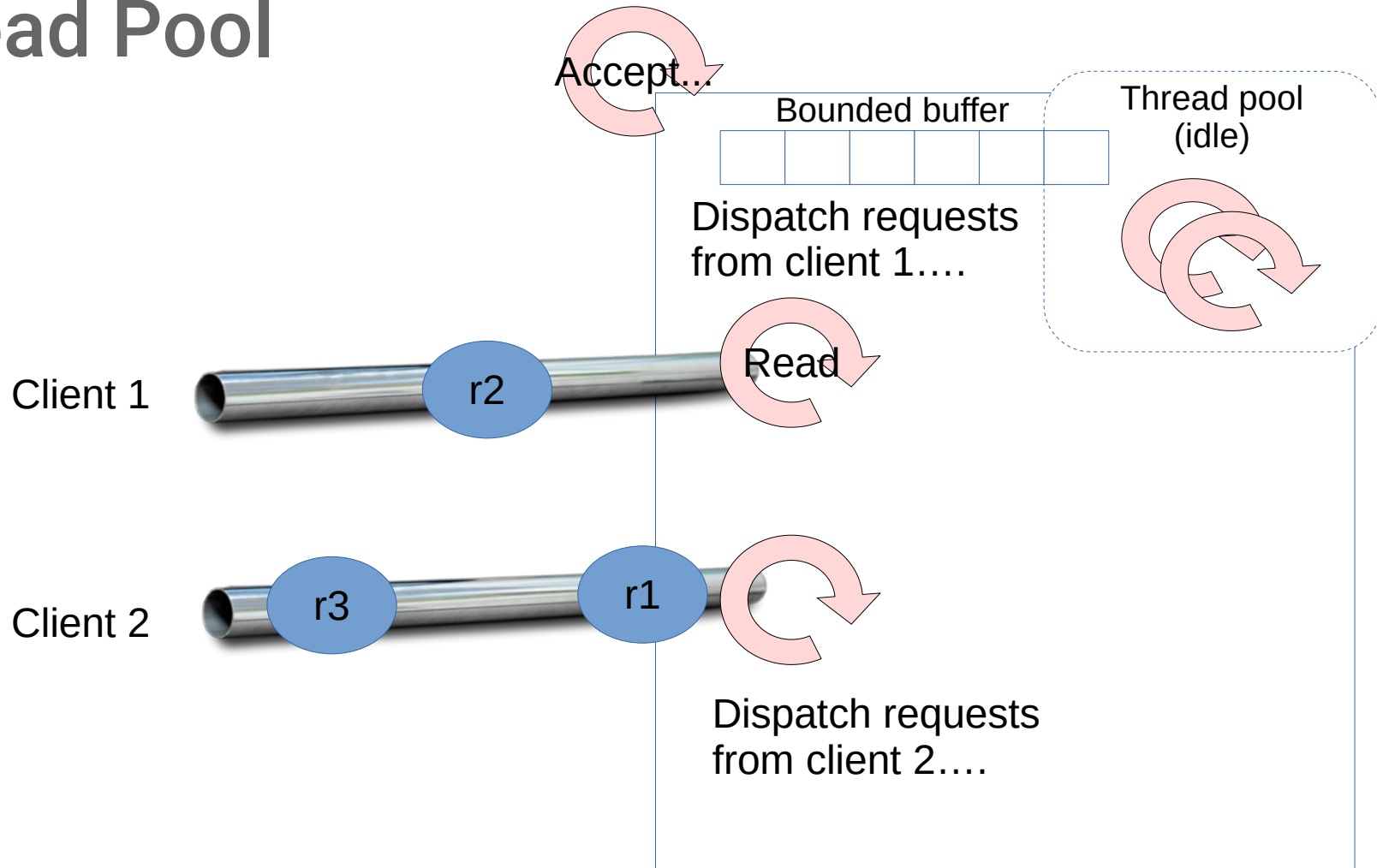
Thread-per-request



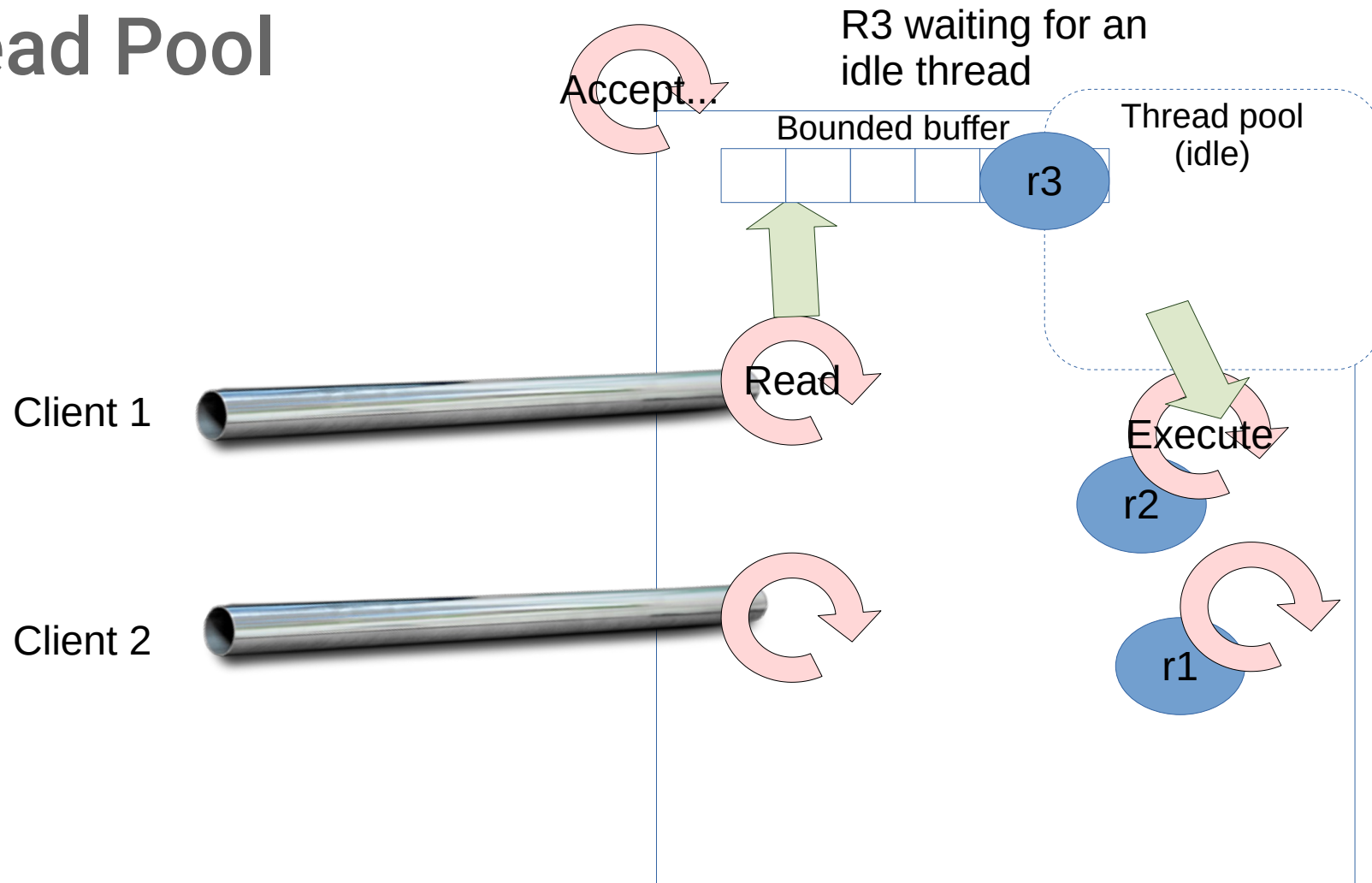
Thread-per-request

- General model for multi-threaded and long-lived requests
- Requests may be answered in an arbitrary order
 - Need identifiers!
- Additional overhead for each request:
 - Creating a new thread
 - Context-switching and synchronization to hand-off request

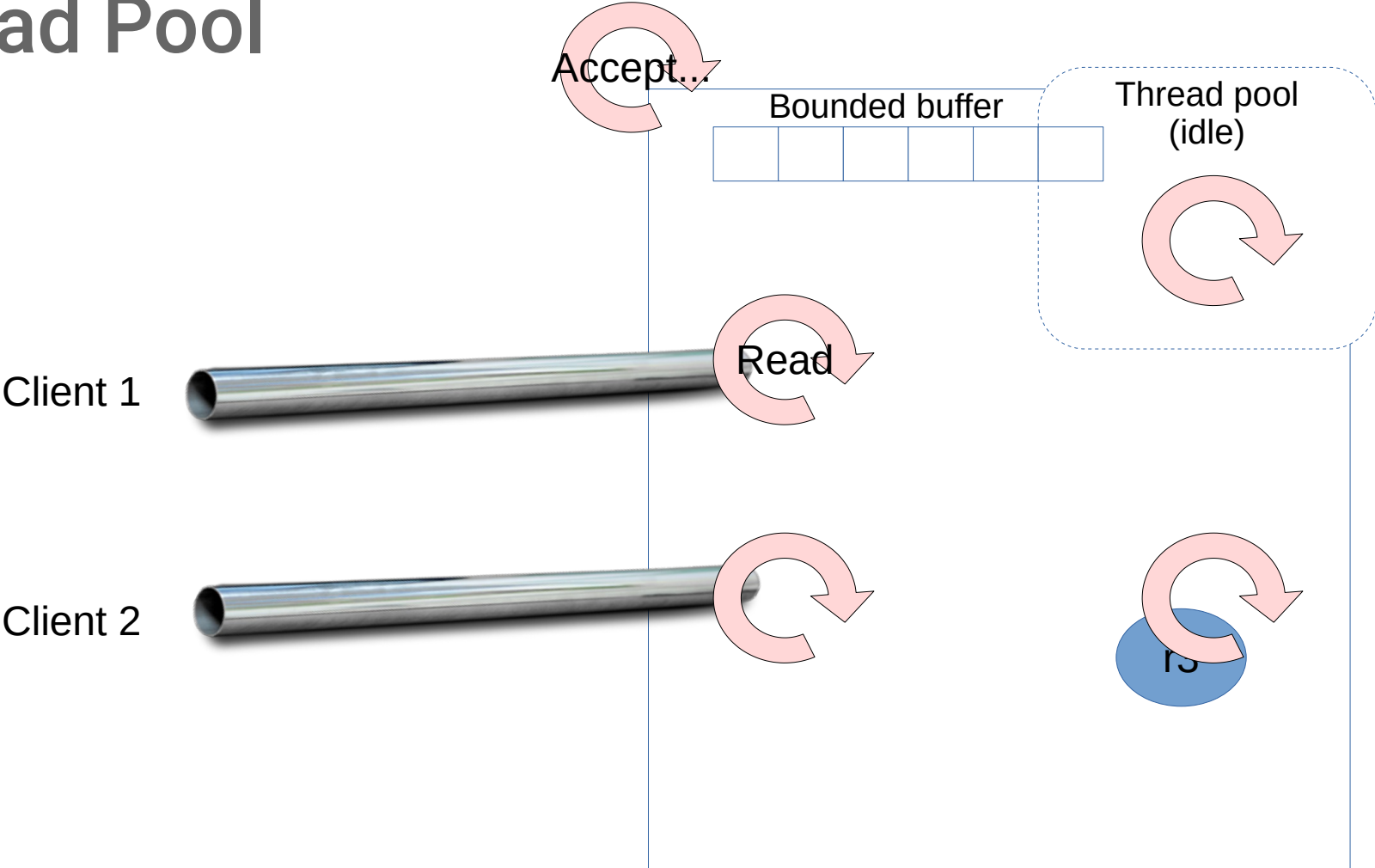
Thread Pool



Thread Pool



Thread Pool



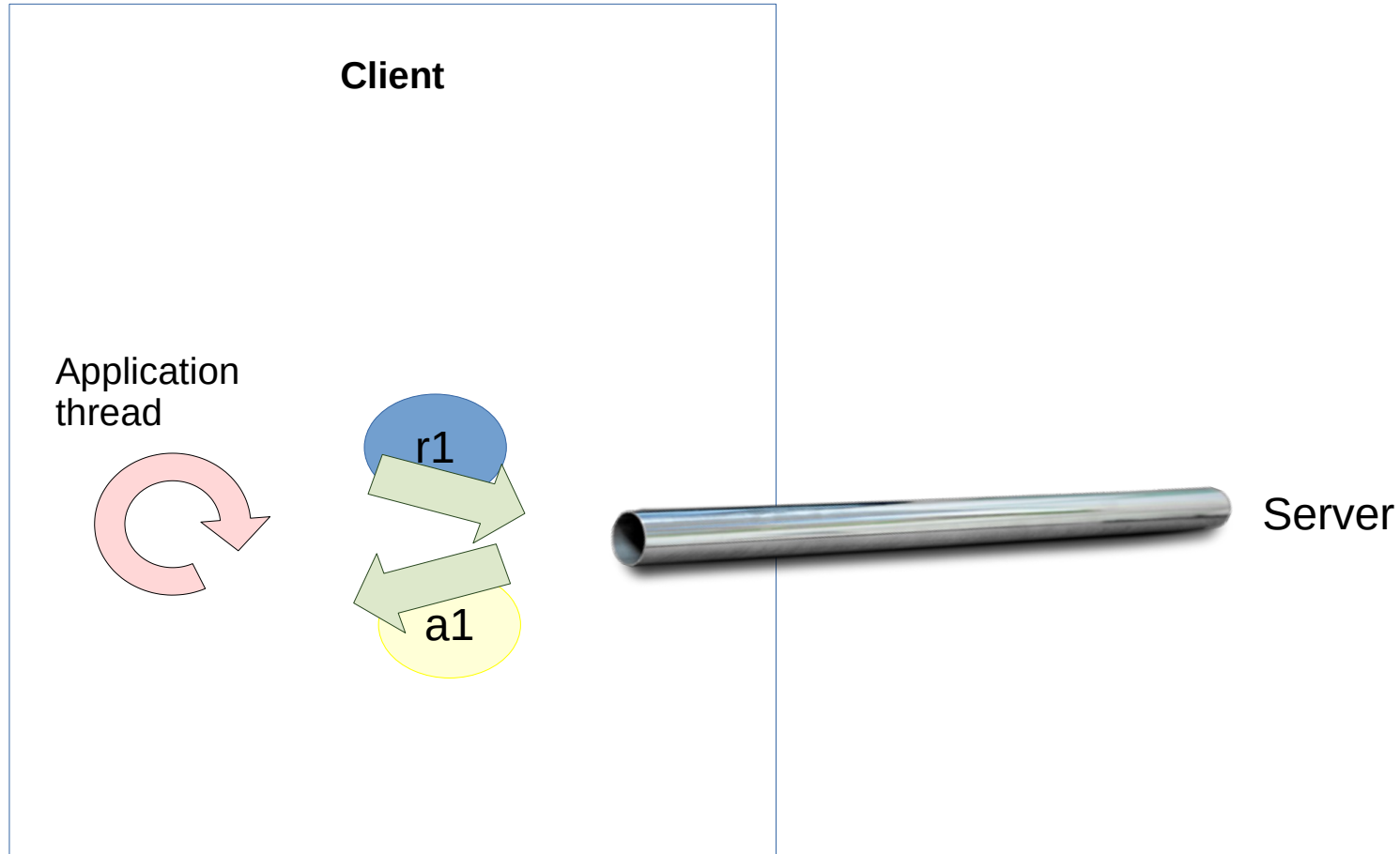
Thread Pool

- Similar to thread-per-request, but...
- Requests are queued in a bounded buffer
- Reduces request overhead by reusing threads
- Provides admission-control to restrict the amount of resources used in the server

1-to-n Notifications

- What if a request triggers replies for multiple clients?
- Writing directly on each socket:
 - Sequential and can block
- The solution is to have a second thread and a outgoing queue for each connection and:
 - The request enqueues a reply to each destination
 - Writer threads awake and write to sockets
- How do clientes handle notifications?

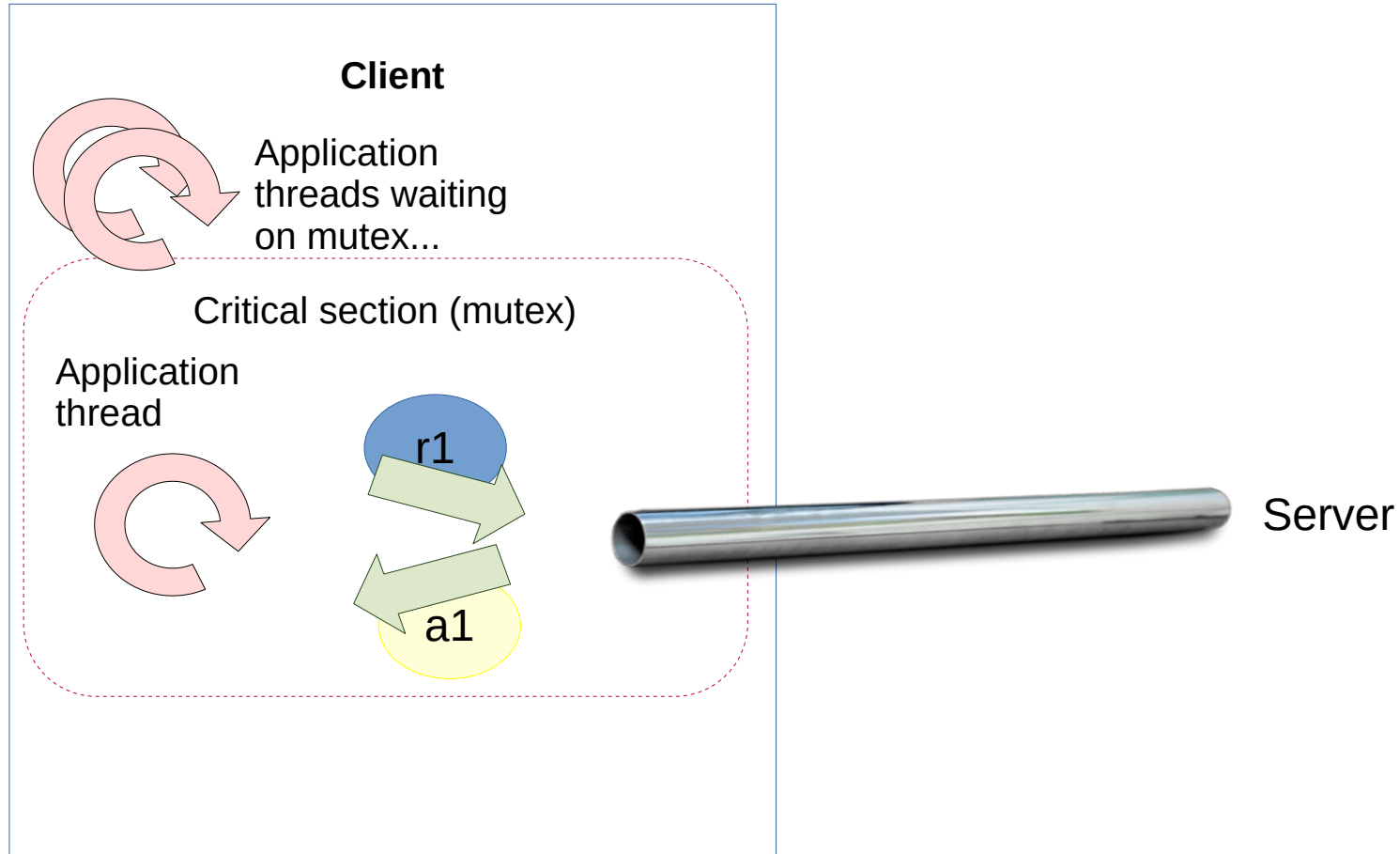
Single-threaded



Single-threaded

- Application has a single thread
- Application thread sends requests and waits for answers

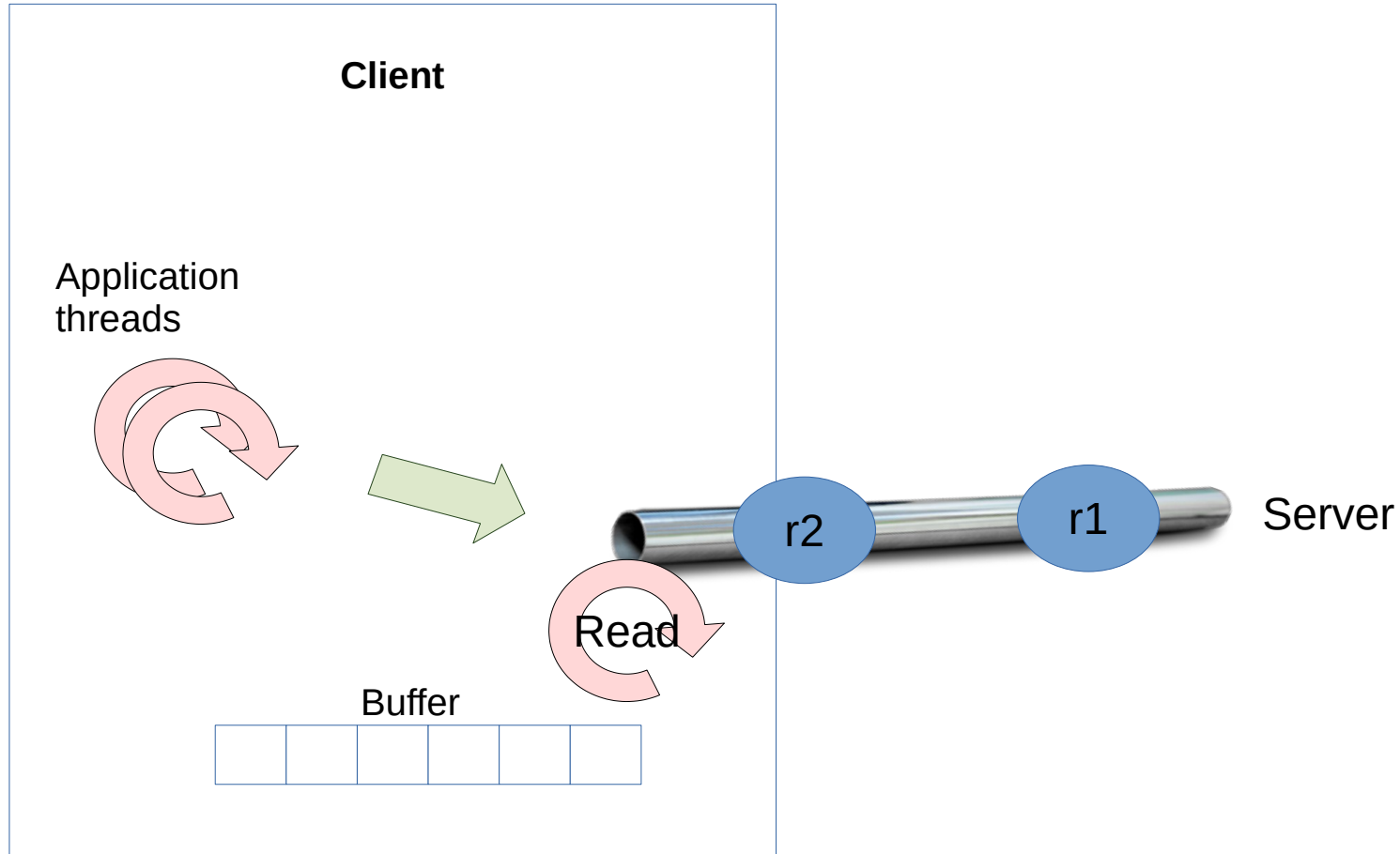
Multi-threaded



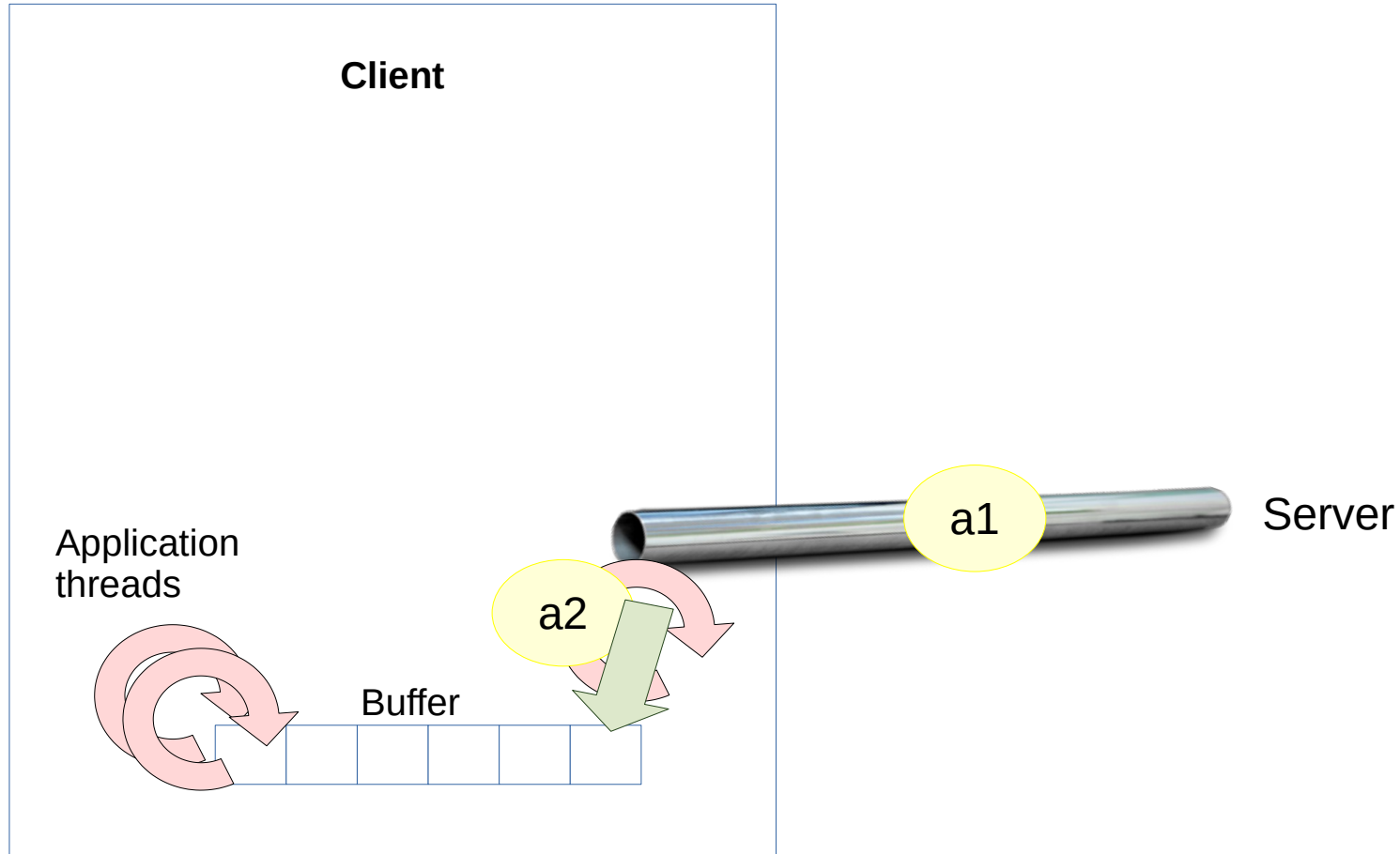
Multi-threaded

- Application has multiple threads
- The client code is protected in a critical section
 - Only one thread uses the connection
 - Threads queue for remote service
- Remote invocations likely to take a long time...

Multi-threaded with Dispatcher



Multi-threaded with Dispatcher



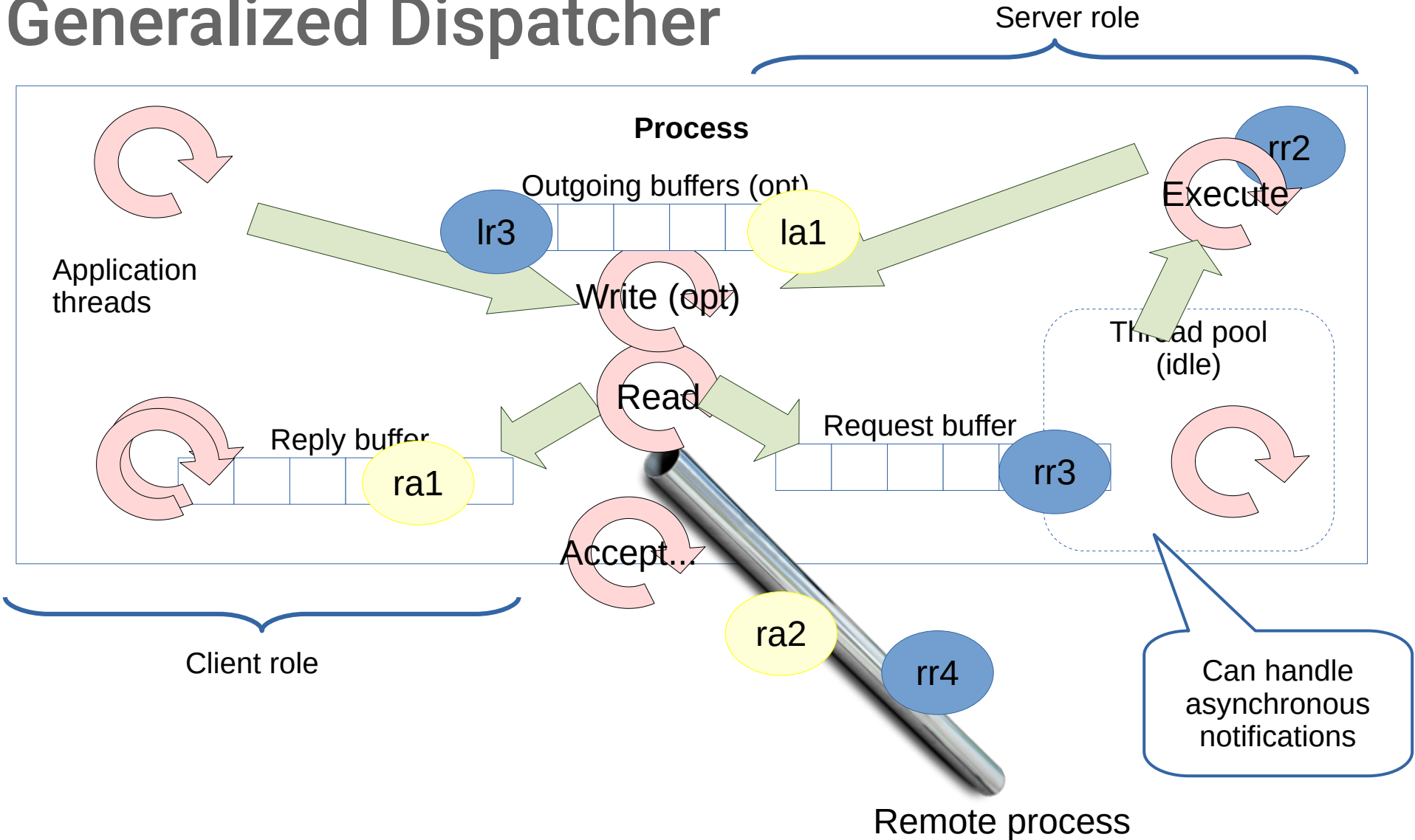
Multi-threaded with Dispatcher

- Multiple application threads issue requests without waiting for answers
 - Requests must be labeled with ids
- A single thread is dedicated to collecting answers and buffering them
- Application threads collect the correct answer from the buffer

Callbacks

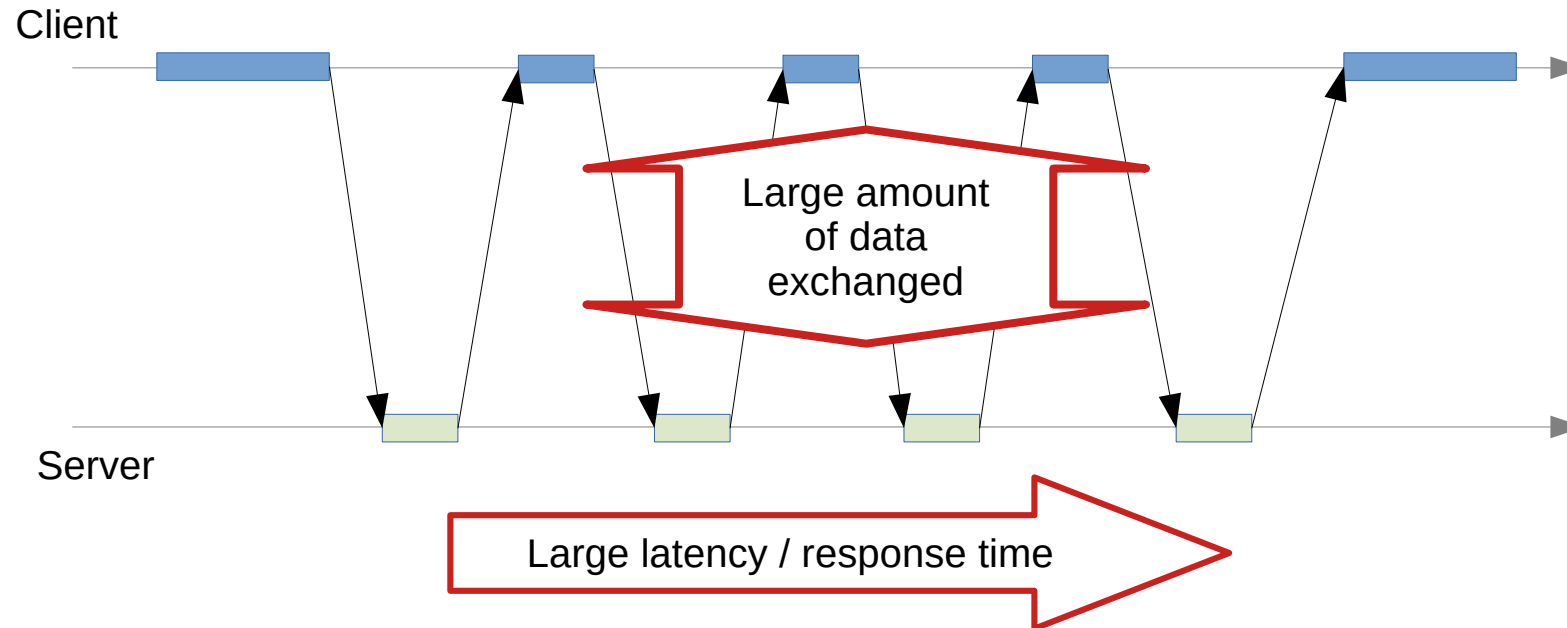
- What if the client dispatcher gets a request from a server?
 - The client dispatcher can use a local thread-pool to execute it
 - And then reply to the server
- This is a generalized symmetric model, in which the server can “callback” the client at any time

Generalized Dispatcher



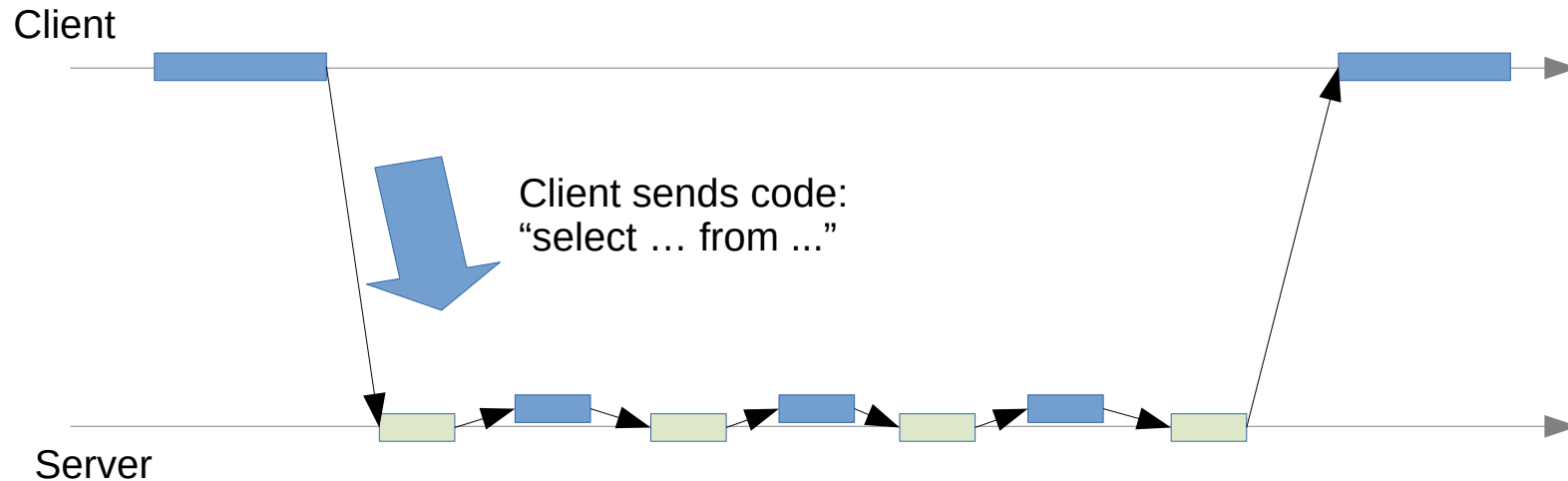
Limitations to transparent migration

- Examples: User interfaces, data base query, ...



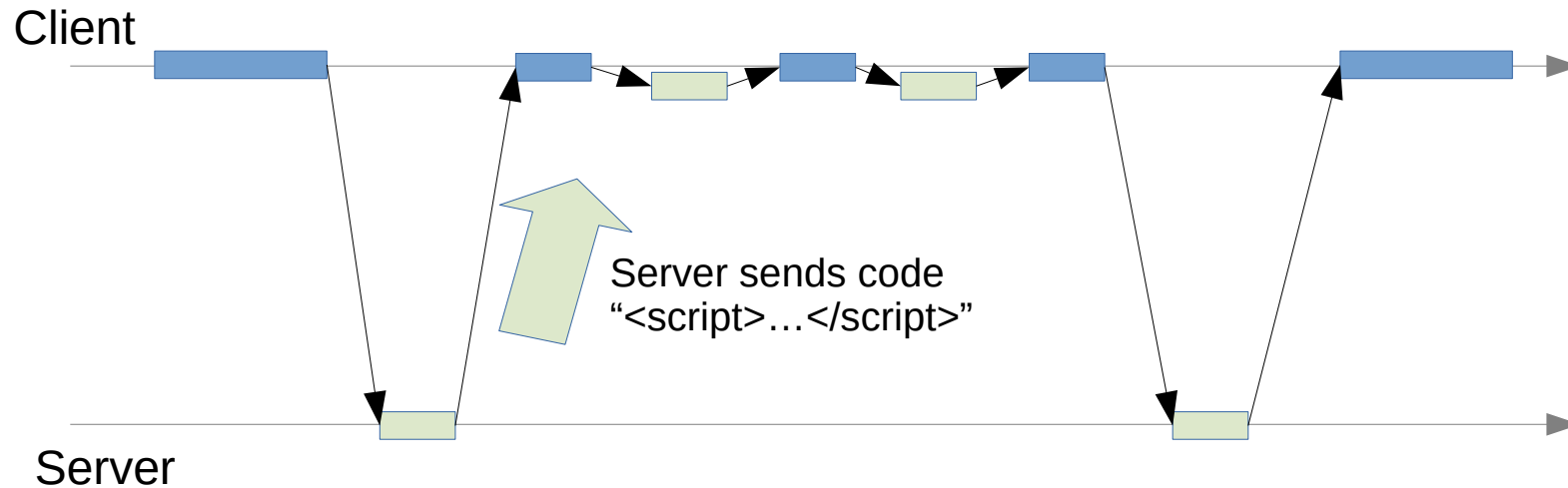
Solution: Code Migration to Server

- Example: SQL DBMS
- Avoids data being shipped over the network



Solution: Code Migration to Client

- Example: Web application with JavaScript
- Avoids latency of multiple round-trips to server

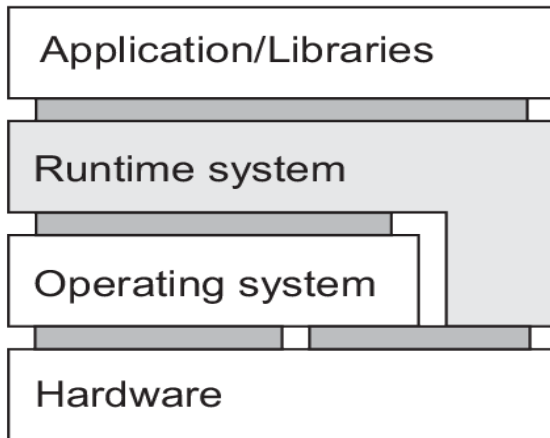


Challenges

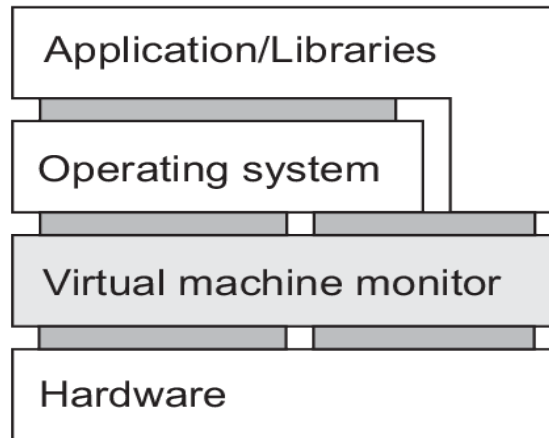
- Conflicting main issues:
 - Security: Constrain what the code can do
 - Efficiency: Allow code to run closer to the hardware
- Heterogeneity:
 - Different processor architectures
 - Different operating systems and libraries

Virtualization

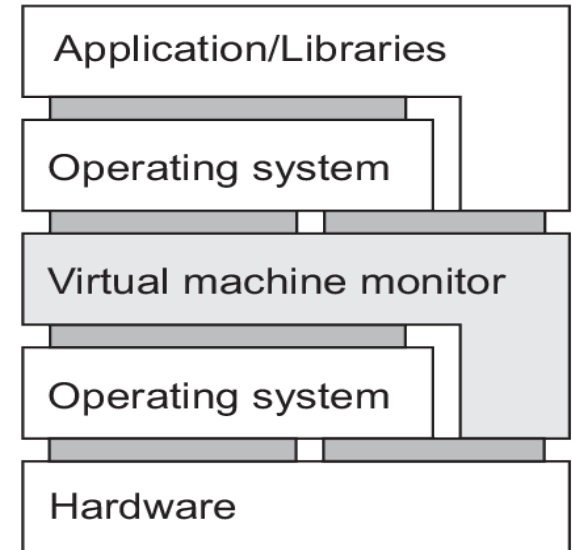
- Creates isolated virtual machines
 - Many in a single physical host
 - Access to resources is controlled



Examples: JVM, JS, WASM

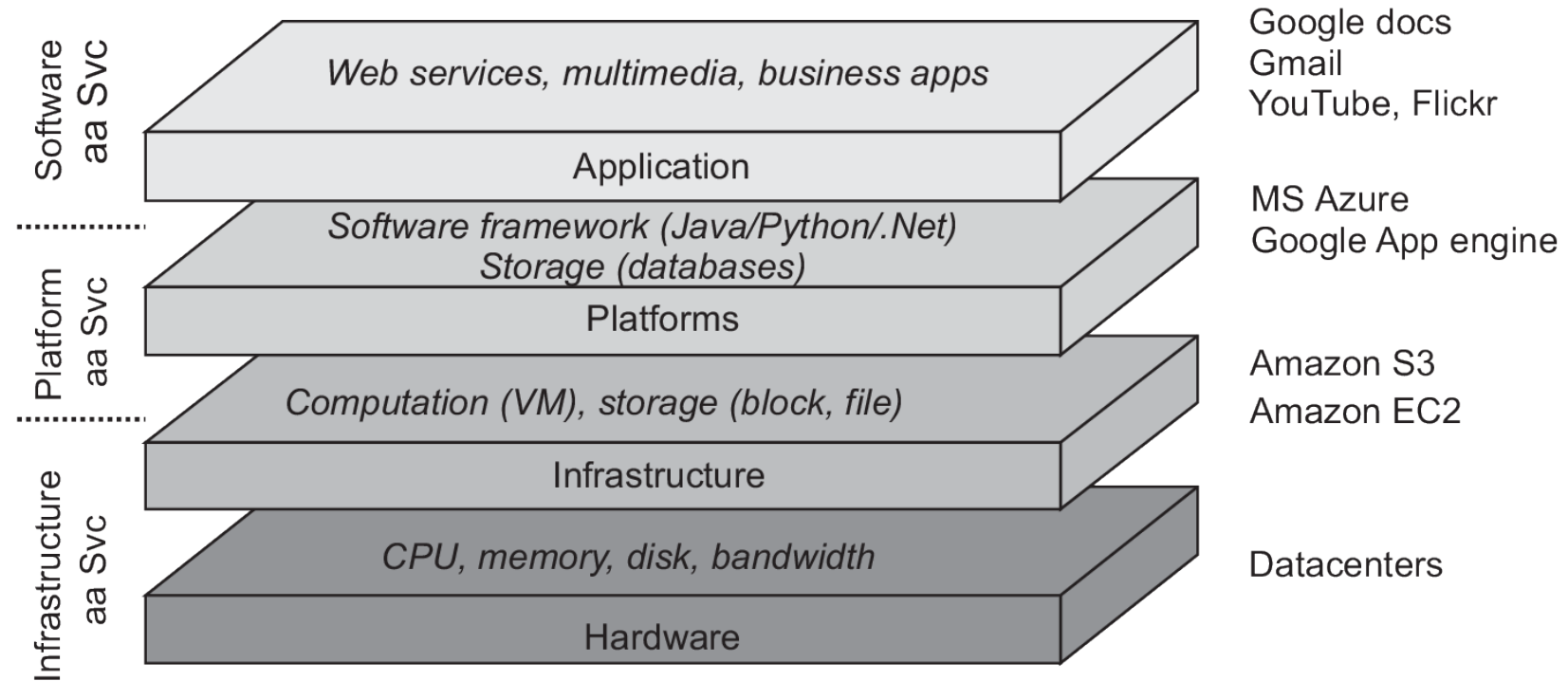


Example: Xen, VMware, VirtualBox, KVM



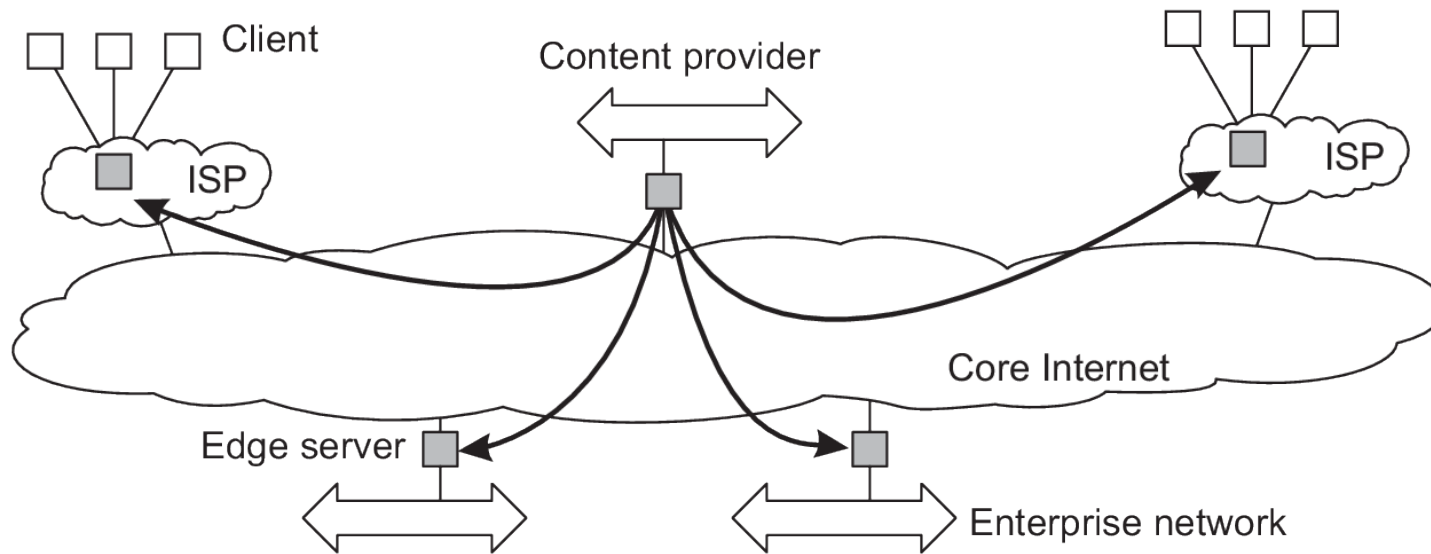
Cloud computing

- Combines virtualization with programmable pay-per-use infrastructure



Edge and Fog computing

- Servers placed at the edge of the network (e.g., CDN)
- Fog computing combines cloud and edge



Summary

- Cloud and fog make it easy to deploy various solutions to place services closer to clients
- Portable code enables code migration:
 - Java bytecode
 - JavaScript / ECMAScript
 - WebAssembly
- Cloud and fog computing provide new opportunities to place application components