# Message Oriented Middleware (MOM)

## Message-based communication

### Distributed System

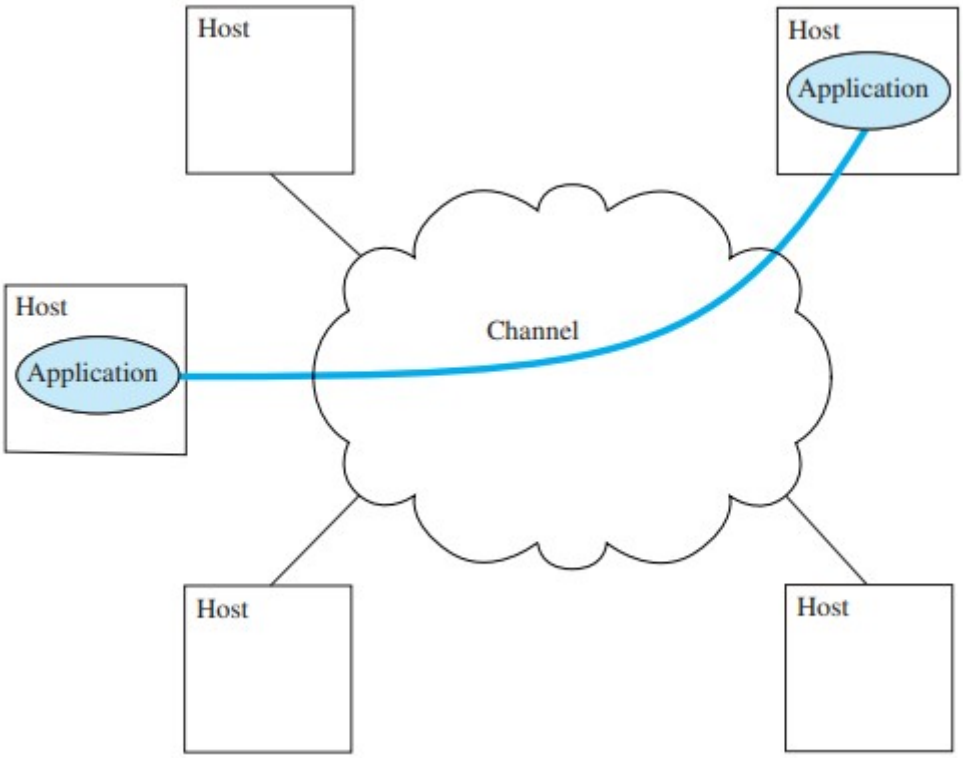
Consists of a collection of distinct processes which:

- are spatially separated
- communicate with one another by exchanging messages

**Message**: atomic bit string; its format and meaning are specified by a communications protocol

### Message-based Communication and Networking

The transport of a message from its source to its destination is performed by a computer networl
The network can be abstracted as a communication channel




### Internet Protocols

| Layer | Function |
|---|---|
| Aplication | Specific communication services |

| Layer | Function |
|---|---|
| Transport | Communication between 2 (or more) processes |
| Network | Communication between 2 computers not directly connected with each other |
| Interface | Communication between 2 computers directly connected |

On the Internet, the properties of the communication channel provided to an application depend on the transport protocol used (UDP/TCP):

- The design of a distributed application depends on the properties provided by the chosen transport protocol

## UDP vs. TCP

| Property | UDP | TCP |
|---|---|---|
| Abstraction | Message | Stream |
| Connection-based | ✗ | ✓ |
| Reliability (loss & duplication) | ✗ | ✓ |
| Order | ✗ | ✓ |
| Flow control | ✗ | ✓ |
| Number of recipients | 1 / n | 1 |

## TCP Reliability

### Message Loss

TCP guarantees that the aplliction will be notified if the local end is unable to communicate with the remoe end

- `send()`/`write()` or `recv()`/`read()` will return an error code
- TCP **cannot** guarantee that there is no data loss - the application should deal with this

### Message duplication

**Why not always retransmit?**
The re-transmitted message may have been delivered before the connection was closed

- TCP is not able to filter data duplicated by the application

- Issue when it's a request for a non-idempotent operation (e.g. credit/debit operation)
  The application may need to **synchronize** with the remote end to learn if there was some data loss in either direction

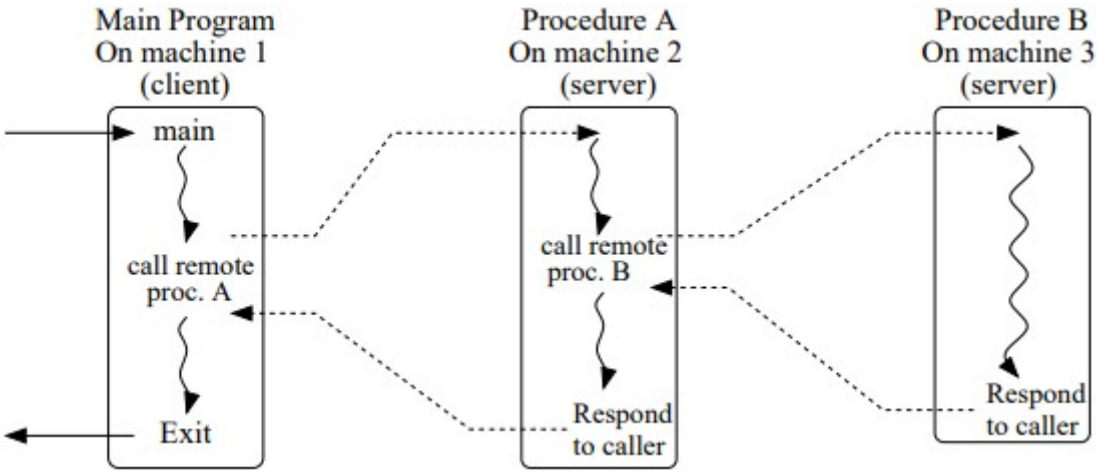
## RPC (Remote Procedure Call)

> Reference: *van Steen and Tanenbaum, Distributed Systems, 3rd Ed.*

**Idea:** allow programs to call procedures located on other machines

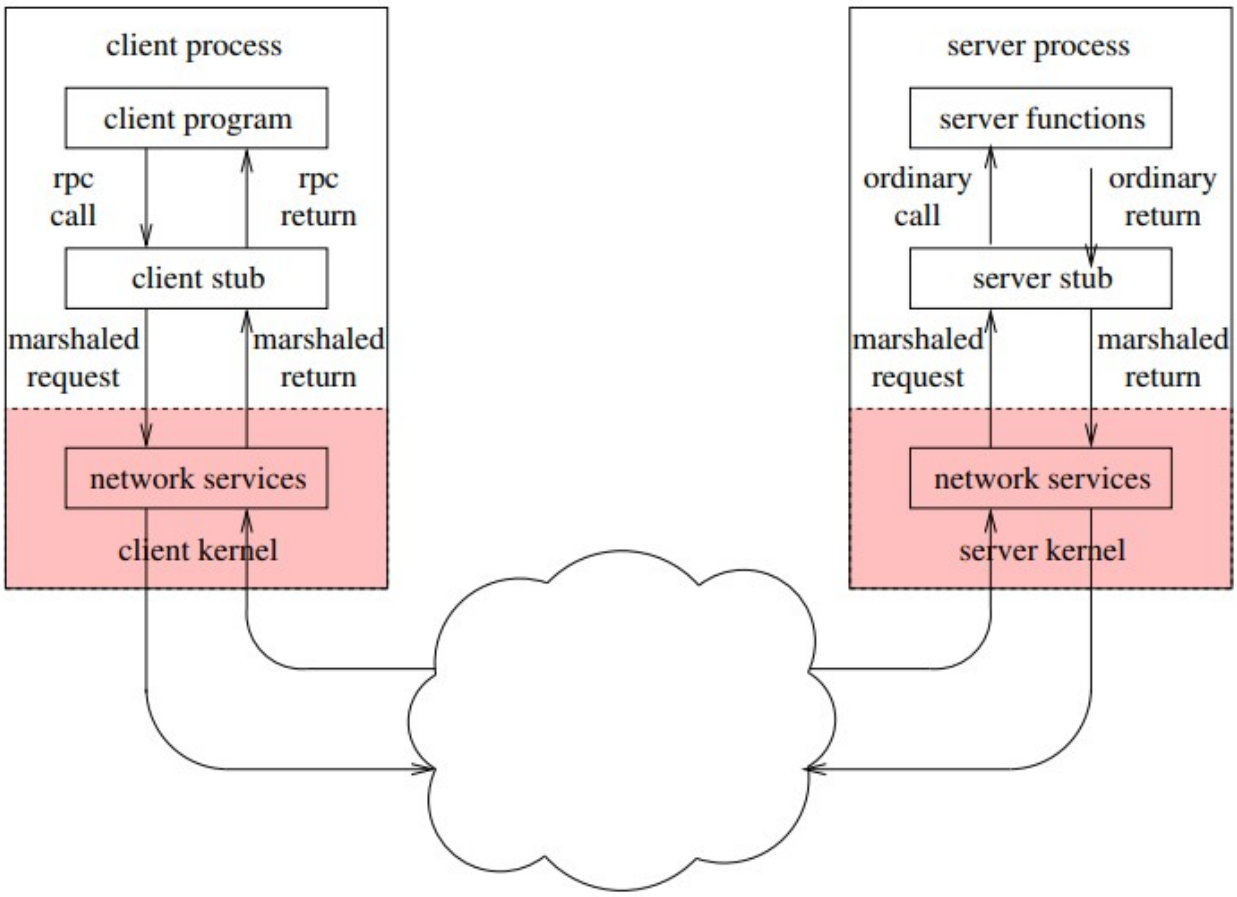
1. Process on machine A calls procedure on machine B
2. Calling process on A is suspended
3. Execution of the called procedure taks place on B

- Information can be transported from the caller to the calee in the parameters and can come back in the procedure result.
- No message passing at all is visible to the programmer.
- Great for request-reply communication patterns (but even then there may be better alternatives)




## Architecture

- RPC is typically implemented on top of the transport layer (TCP/IP)




## Client Stub

### Request

1. Assembles message: **parameter mashalling**
2. Send messages (`write()`/`sendto()`)
3. Blocks waiting for response (`read()`/`recvfrom()`)
    - Assuming synchronous RPC

### Response

1. Receives response
2. Extracts the results (**unmarshalling**)
3. Returns to client
    - Assuming synchronous RPC

## Server Stub

### Request

1. Receives message (`read()`/`recvfrom()`)
2. Parses message to determine arguments (**unmarshalling**)
3. Calls function

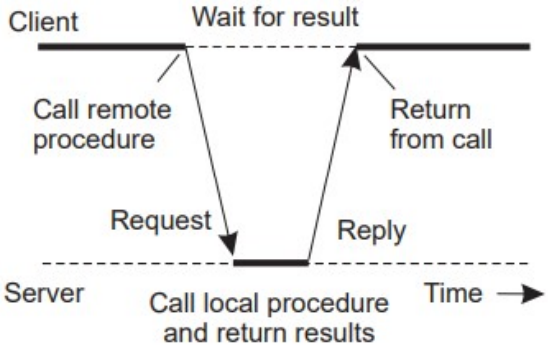
### Response

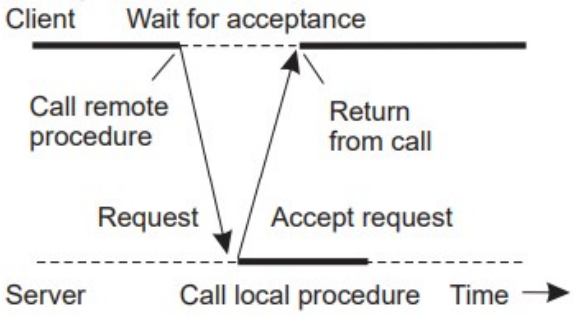
1. Assembles message with the return value of the function
2. Send message (`write()`/`sendto()`)
3. Blocks waiting for a new request

## Synchronous vs. Asynchronous

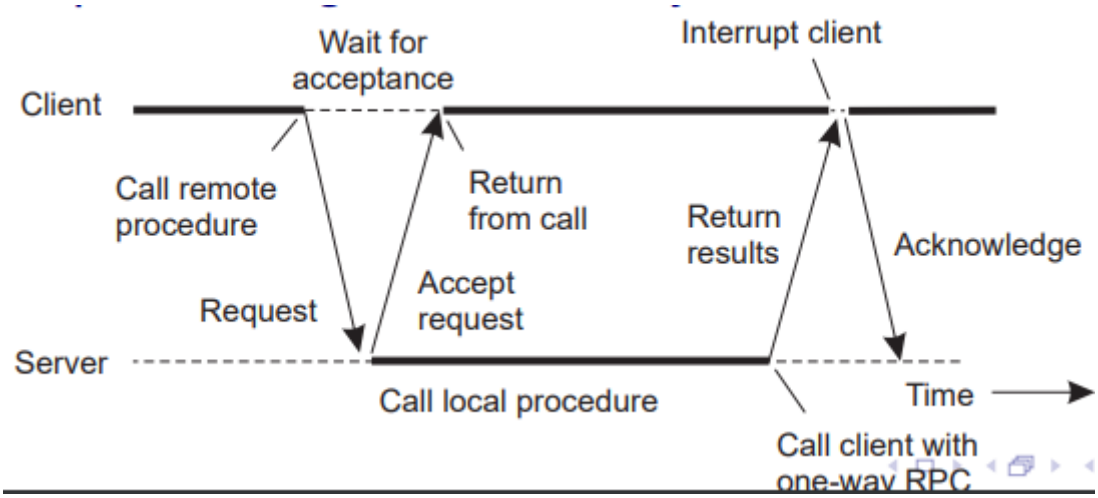
### Synchronous

- Client blocks until it receives the response
- Programming distributed applications with synchronous RPC would be almost as simple as programming non-distributed applications




### Asynchronous




# Asynchronous Communication (MOM)

**Problem**: The communication parties may not always be simultaneously available
**Solution**: Use *asynchronous communication*

## Message Oriented Middleware (MOM)

**Asynchronous** message-based communication

- Communication service (middleare) stores the messages as long as needed to deliver them
- Guarantees **asynchronicity** and **anonymity**
  **Publishers**: may send messages
  **Subscribers**: may receive messages

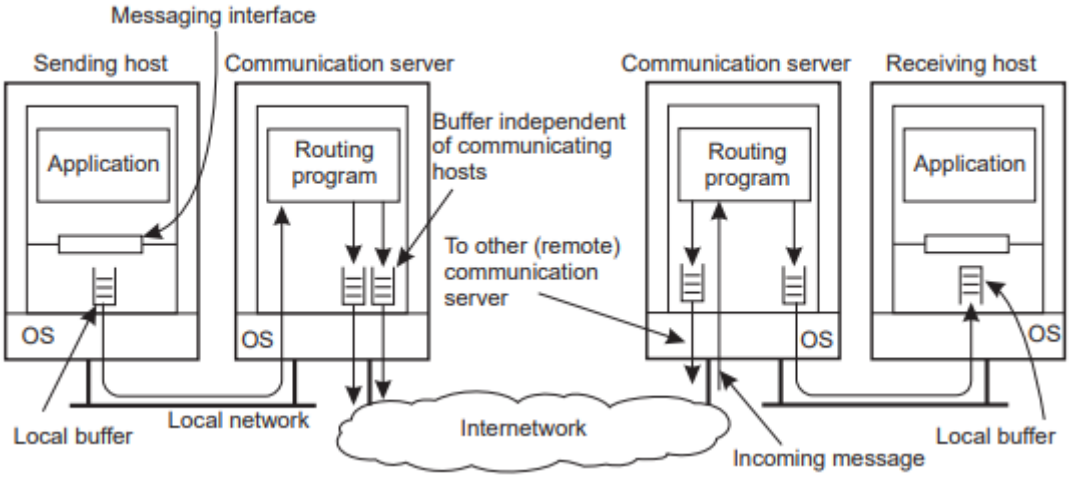
## Basic Patterns

| Pattern | Structure | Roles | Delivery |
|---|---|---|---|
| **Point-to-point** | Queue | Senders and Receivers | Each message is delivered to at most one process |
| **Publish-Subscribe** | Topic | Publishers and Subscribers | A message is delivered to more than one process |

## Messaging Service Implementation

Provides asynchronous communication

- At the lowest communication level, there must be synchronization between sender and receiver




## Asynchronous Communication Applications

Appropriate for applications when the send and receiver are **loosely coupled**:

- Enterprise Application Integration
- Workflow Applications
  - Related to **business processes**
- Microservices
- Message based communication between people (Email, SMS, Instant Messaging)

## Java Message Service (JMS)

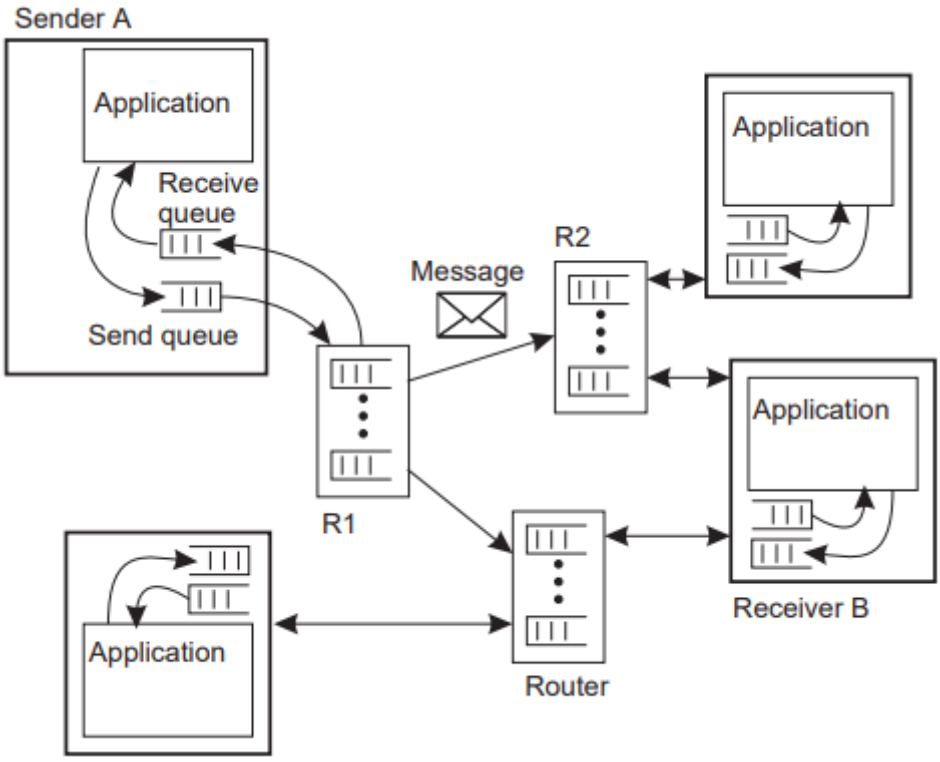
Temos se saber cenas especificas de java?.....
TODO

## Message Queing Protocols

**AMQP** (Advanced Message Queuing Protocol): open-standard protocol (approved by OASIS)
**MTTQ** (Message Queuing Telemetry Transport): also an OASIS transport
**OpenWire**: public protocol used by Apache ActiveMQ

### Architecture

- Larger scale systems may use **message relays** to route messages to their destinations
- Similar to SMTP, although nowadays almost every email message just traverses 2 serversD




### Message Brokers

Convert the format of the messages used by one application to the format used by another application

- They are not part of the communication service