RPC / RMI vs MOM / JMS

Comparison of Communication Styles

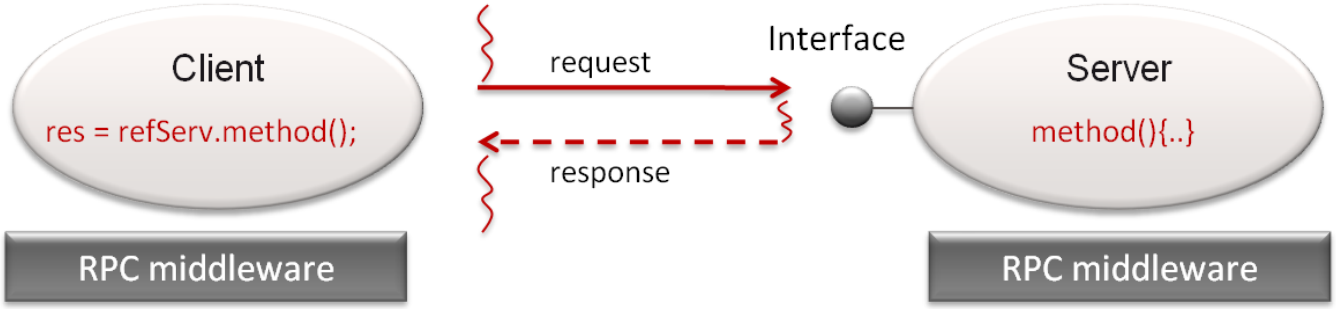
Ada Diaconescu
ada.diaconescu@telecom-paris.fr

Reminder

RMI & JMS

RPC / RMI – Reminder

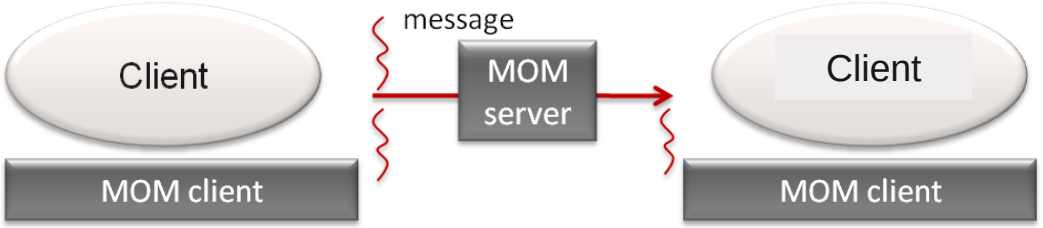
General Architecture and Main Characteristics



- Strong-coupling
  - The Client depends on the Server's Interface
  - The client must "know" the Server (Reference – e.g., IP address, port)
- Time dependency
  - The Client and the Server must be available simultaneously
- Synchronous/blocking communication (usually)
  - The Client is blocked after sending the request and until receiving the response

MOM / JMS – Reminder

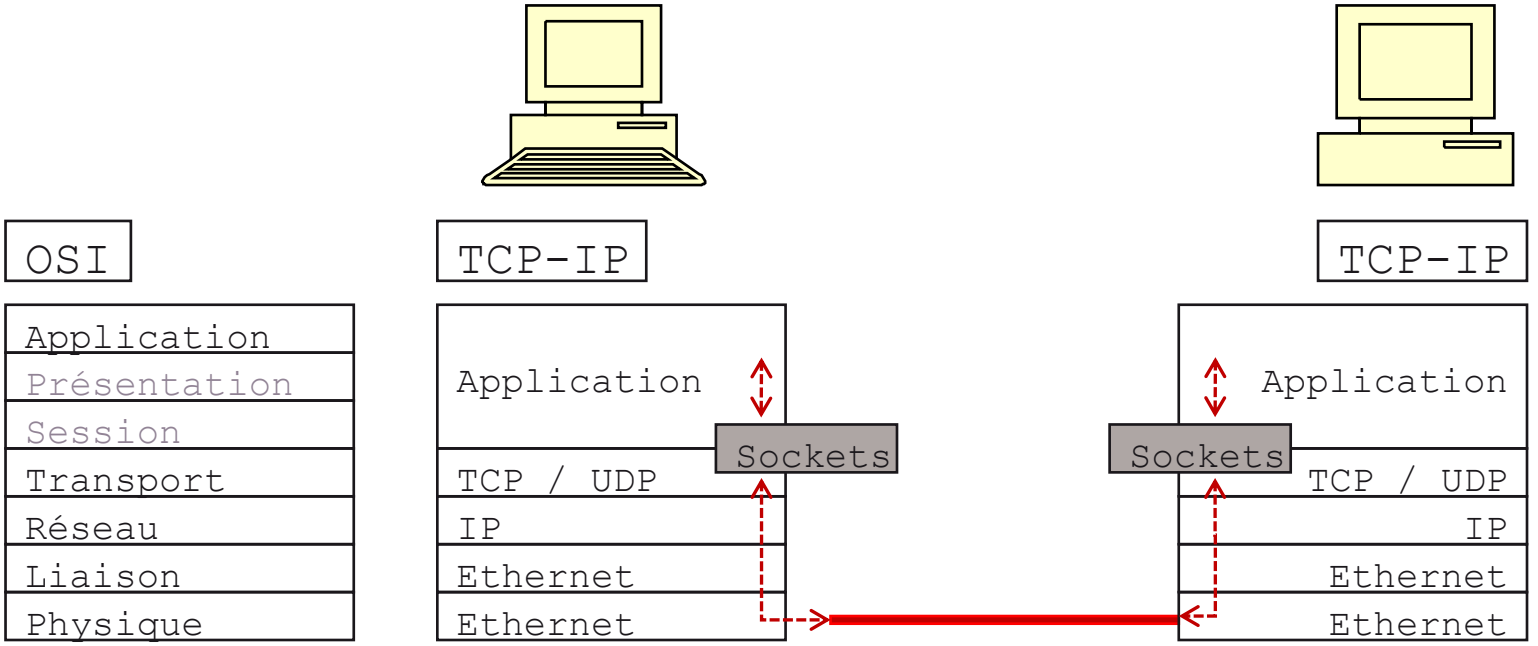
General Architecture and Main Characteristics



- Loose coupling (possible) => facilitates component (Client) replacement
  - A component (client) does not depend on the interface of other components (clients)
  - A Client does *not* need to know the other clients (using a MOM server/broker)
- No time dependency => clients do not need to be available simultaneously
  - Application components are not always available at the same time
  - Except for transitory communication (e.g; via topics and without durable subscriptions)
- Support for asynchronous / non-blocking communication
  - Producers are not blocked after sending a message (except if waiting for an acknowledgement)
  - Consumers can receive message in both blocking and non-blocking manners

Message-oriented Communication

- At the network level, all communication is message-based
- At the application level, there are different communication models (via specific middleware)



Message-oriented Communication
→ Application Level ←

Main Characteristics

- Synchronous vs. Asynchronous
- Persistent vs. Transitory
- Unicast vs. Multicast
- Push vs. Pull
- Communication patterns:
  Request/reply, Point-to-Point, Publish / Subscribe (Pub/Sub), …
- Message routing systems
- Message processing library
- ...

=> A wide range of message-based communication models

- Implementing various combinations of the aforementioned aspects

Message-oriented Communication

Synchronous vs. Asynchronous communication

- Synchronous / blocking
  - The message sender is blocked until the recipient acknowledges the message reception
- Asynchronous / non-blocking
  - The message sender continues to execute after sending the message

Message-oriented Communication

Persistent vs. Transitory data

- Persistent Communication
  - The MOM Server stores the message until it can be transmitted to the receiver(s)
  - A message is never lost or deleted (except if setting an expiration time)
  - => no time dependency: no need for simultaneous availability of senders and receivers

- Transient Communication
  - The MOM server only stores the message in transit from the sender(s) to the receiver(s)
  - => time dependency: the sender(s) and receiver(s) must be available at the same time

Message-oriented Communication

Unicast vs. Multicast

- Unicast
  - The message is sent to a single destination
- Multicast - Diffusion (or group)
  - The message can be sent to multiple destinations

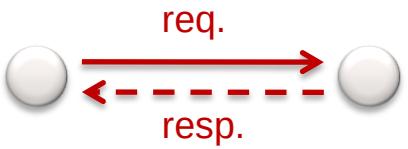
Push vs. Pull mode

- Push mode
  - The sender transmits, or “pushes”, the message to the receiver
  - E.g. By calling a “callback” method on the receiver
- Pull mode
  - The receiver fetches, or “pulls”, the message from the sender
  - E.g.:
    - Periodically;
    - Waiting in a blocked state until a message becomes available;
    - Upon receiving a notification from the sender (i.e. push and pull mode)

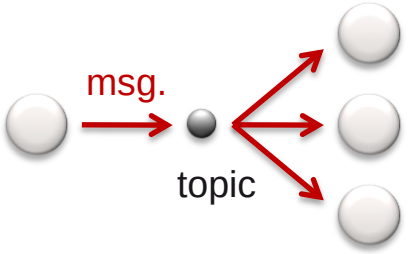
Models for Message-oriented Communication

- Point-to-Point

- Request / Response



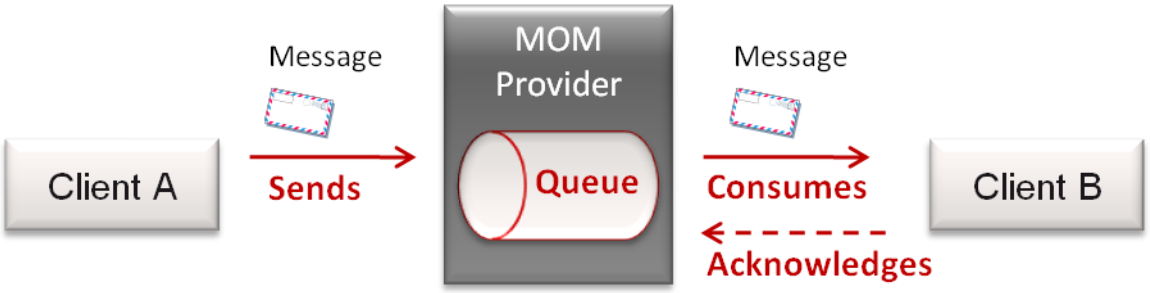
- Publish / Subscribe (Pub/Sub)



- …

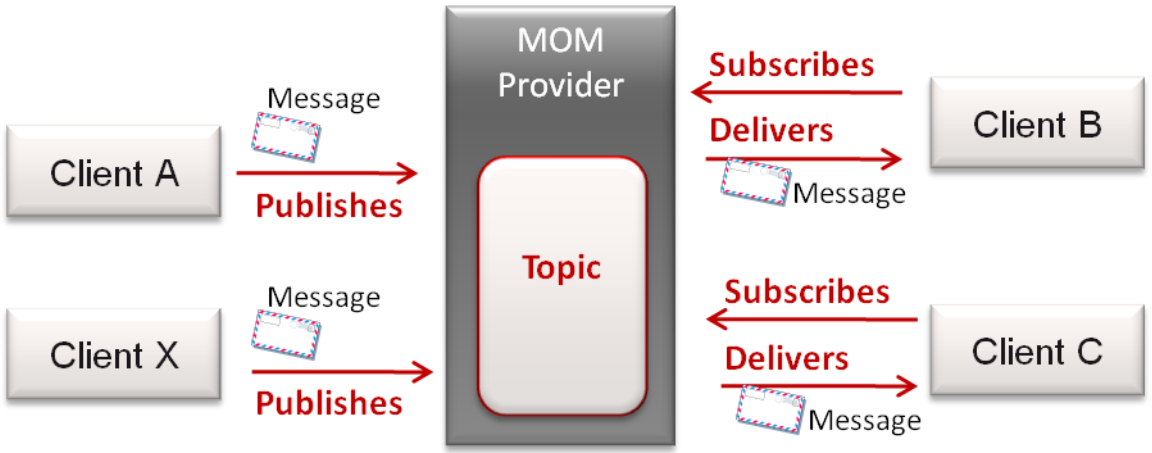
Models for Message-oriented Communication
→ Example 1 ←

- Point-to-Point (1 : 1)
  - Each message is stored in a Queue until a recipient reads it
  - Each message is consumed only once (by a single recipient)
  - No time dependency between the message sender and the receiver
  - The message receiver can acknowledge message reception



Models for Message-oriented Communication
→ Example 2 ←

- Publish/Subscribe (* : *)
  - The Topic manages message transmission to all subscribers
  - Space Decoupling between publishers and the subscribers
    - Publishers do not need to know the subscribers
  - Time decoupling between publishers and subscribers
    - A client can read messages only *after* having subscribed to the corresponding topic
    - A subscribed client must remain active to receive message from the corresponding topic(s)



RMI Characteristics

RMI Communication

- Synchronous
  - The Client is blocked until it receives a response from the Server
- Transitory
  - Requires simultaneous availability of the Client and the Server
- Unicast
  - The Client sends a single request at a time to a Server
- Push mode
  - The Client pushes the request towards the Server; the Server pushed the reply towards the Client
- Communication model:
  - Request / Reply

JMS Characteristics

JMS Communication

- Synchronous message producer
- Synchronous or asynchronous message consumer

- Persistent via Queues
- Transitory via Topics (except for durable configurations)

- Unicast via Queues
- Multicast via Topics

- Push mode between the producers and the MOM
- Push or Pull mode between the MOM and the consumers

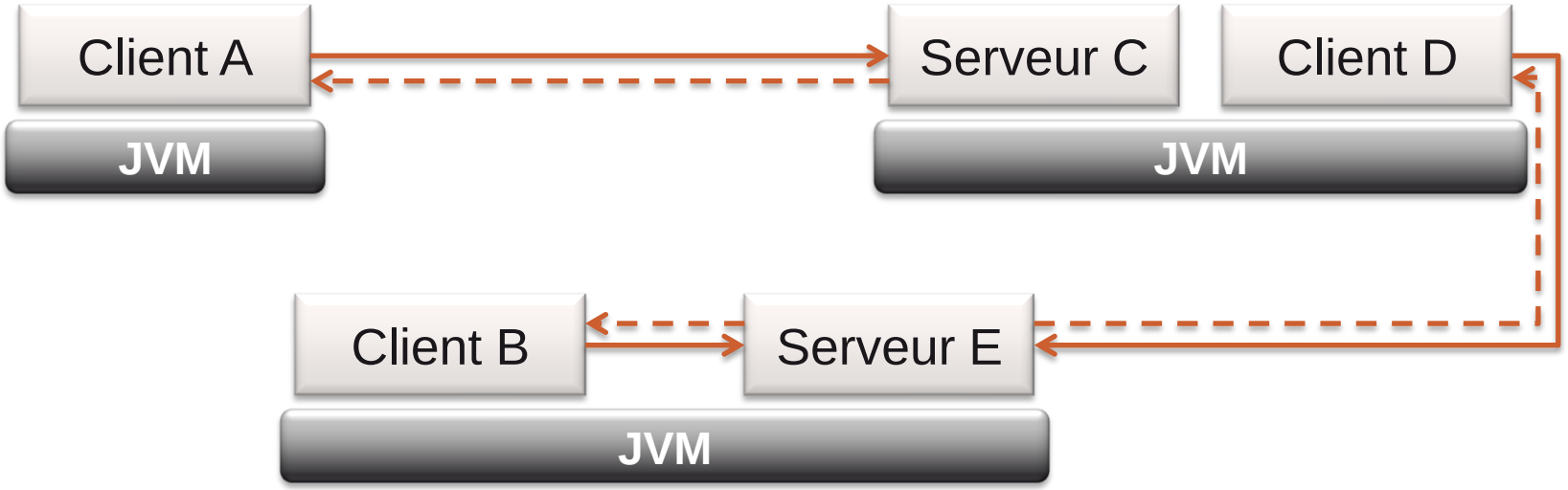
- Communication models: point-to-point or pub/sub

Middleware

TOPOLOGIES

RPC / RMI Topology

- Decentralised:
  - Needs one middleware instance on each machine of the distributed system
  - Ex : for RMI : in the JVM of each Java application
  - Ex : Java EE : on each machine, possibly with a centralised JNDI server

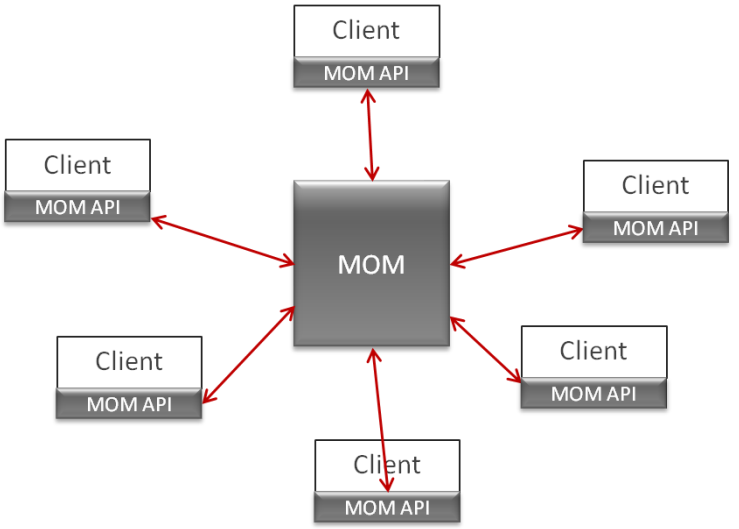


Topologies of MOM Providers

- Various topologies are possible for the Message Broker (MOM)
  - Centralised
  - Decentralised
  - Hybrid
- Each MOM product may support one or several topologies
  - E.g. Joram MOM supports both centralised and decentralised topologies

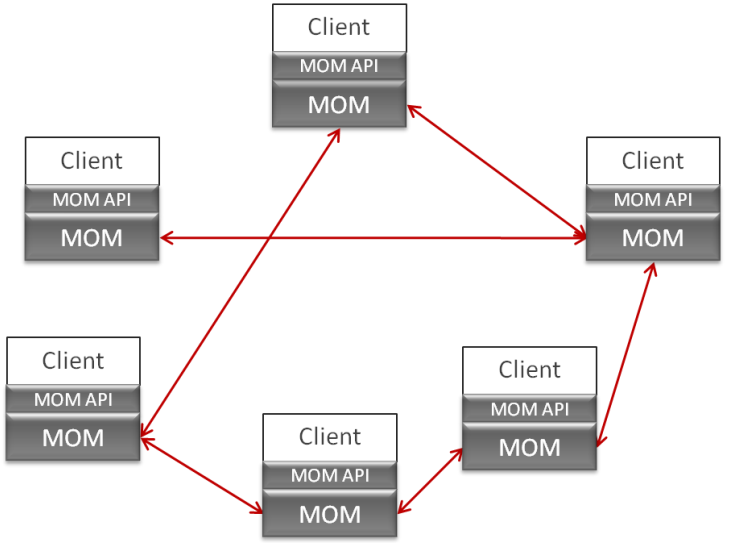
MOM – centralised topology (hub & spoke)

- Centralised server for message management
  - Routing, storage, transactions, ..
- Advantages
  - Client decoupling – adding or removing a client does not impact other clients directly
- Disadvantages
  - May ensue important traffic
  - May cause bottlenecks
    - Single point of failure
    - Limited scalability
    - May introduce high latency



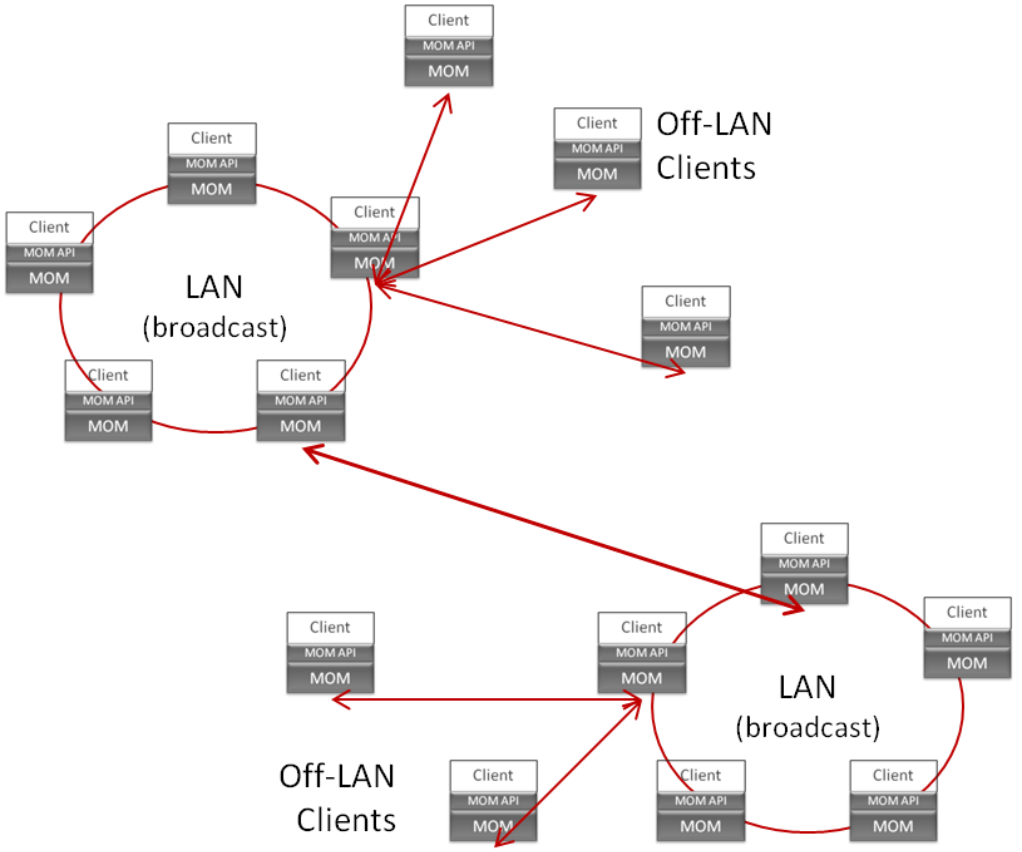
MOM – decentralised topology

- One MOM instznce installed on each Client machine (no central MOM instance)
  - Storing, transactions, security, …
  - Message routing
    - Ad-hoc, between MOMs
    - Based on network protocols - e.g. IP-Multicast
- Advantages
  - Distributes MOM functions du MOM across servers – e.g. persistence, security, …
- Disadvantages
  - Potential interoperability issues – e.g. between different MOM vendors
  - Heavier Clients, function duplication



MOM – hybrid topology

- Combines the two previous architectures
- Aims to combine their advantages and reduce their disadvantages, case-by-case



MOM – sample topology for Joram MOM

- Multiple Joram/MOM Servers
- Each Client connects to one of the Joram Servers
- Any Client may communicate to any other Client, irrespectively of the Joram Server they are connected to