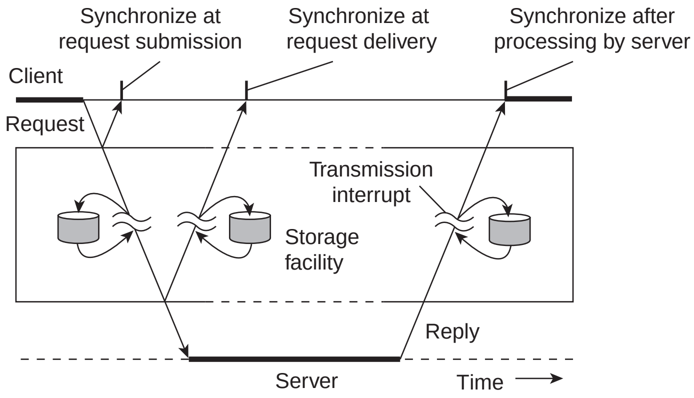
Message Oriented Midleware (MOM)

February 18, 2017

Roadmap

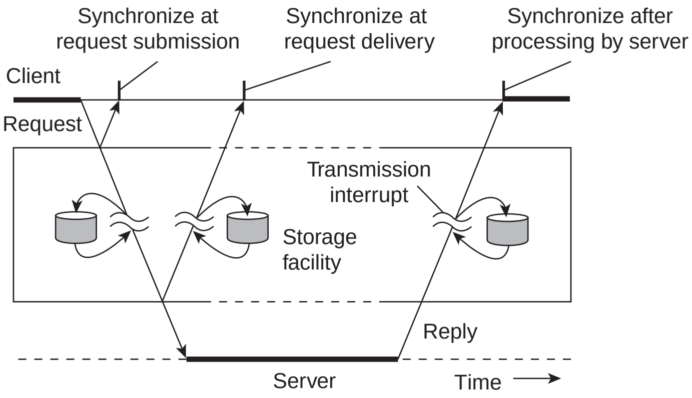
Roadmap

Message-based Communication


Synchronize at request submission
Synchronize at request delivery
Synchronize after processing by server
Client
Request
Transmission interrupt
Storage facility
Reply
Server
Time

1. Persistent vs. transient communication

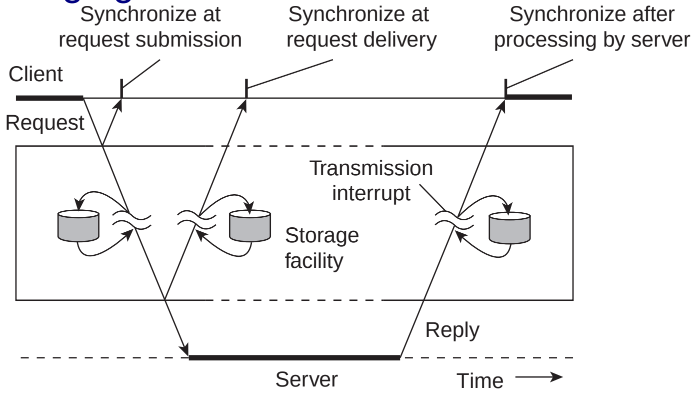
Message-based Communication



Synchronize at request submission
Synchronize at request delivery
Synchronize after processing by server
Client
Request
Transmission interrupt
Storage facility
Reply
Server
Time

1. Persistent vs. transient communication
2. Synchronization
    - with the communication channel/service
    - with the remote communicating party

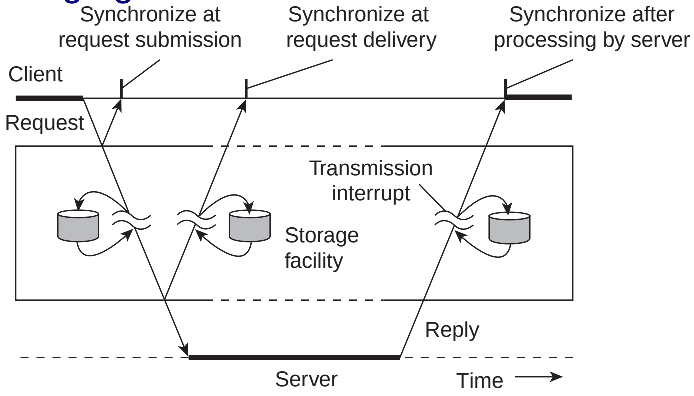
Messaging: Transient vs. Persistent Communication



Synchronize at request submission
Synchronize at request delivery
Synchronize after processing by server
Client
Request
Transmission interrupt
Storage facility
Reply
Server
Time

Transient communication The channel may discard a message in transit, if there are problems with its delivery to the next node (whether or not it is the final destination). Example ?

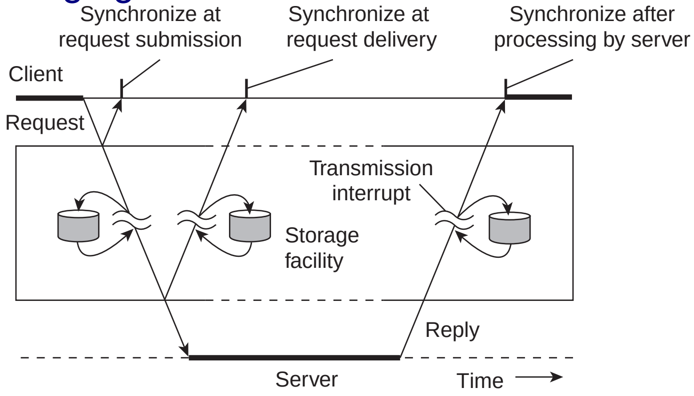
Messaging: Transient vs. Persistent Communication


Synchronize at request submission
Synchronize at request delivery
Synchronize after processing by server
Client
Request
Transmission interrupt
Storage facility
Reply
Server
Time

Transient communication The channel may discard a message in transit, if there are problems with its delivery to the next node (whether or not it is the final destination). Example ? TCP/UDP

Persistent communication The channel keeps a message until it delivers it, even if there are problems with the destination. Example?

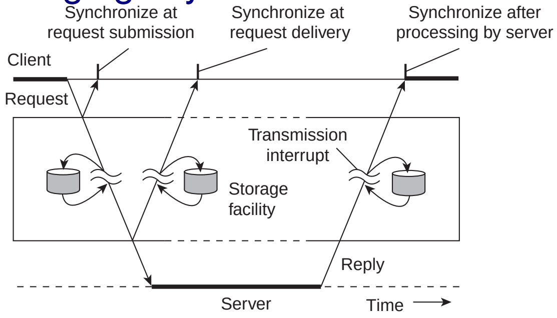
Messaging: Transient vs. Persistent Communication


Synchronize at request submission
Synchronize at request delivery
Synchronize after processing by server
Client
Request
Transmission interrupt
Storage facility
Reply
Server
Time

Transient communication The channel may discard a message in transit, if there are problems with its delivery to the next node (whether or not it is the final destination). Example ? TCP/UDP

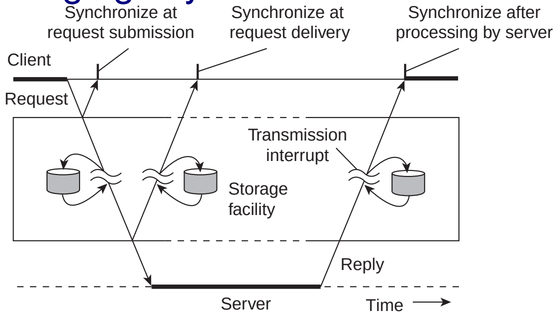
Persistent communication The channel keeps a message until it delivers it, even if there are problems with the destination. Example? SMTP.

Messaging: Synchronization with the Channel


Synchronize at request submission
Synchronize at request delivery
Synchronize after processing by server
Client
Request
Transmission interrupt
Storage facility
Reply
Server
Time

Sender

Messaging: Synchronization with the Channel

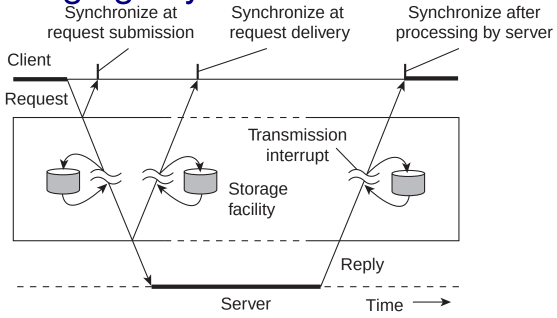


Synchronize at request submission
Synchronize at request delivery
Synchronize after processing by server
Client
Request
Transmission interrupt
Storage facility
Reply
Server
Time

Sender

1. Upon requesting the communication service from the channel. Example?

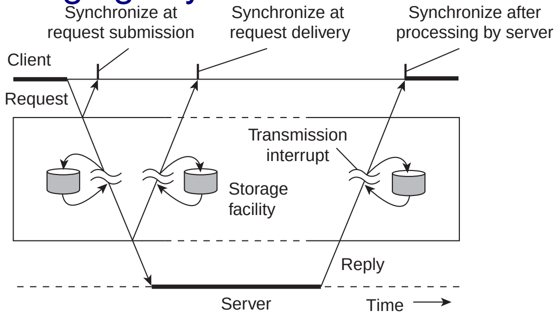
Messaging: Synchronization with the Channel


Synchronize at request submission
Synchronize at request delivery
Synchronize after processing by server
Client
Request
Transmission interrupt
Storage facility
Reply
Server
Time

Sender

1. Upon requesting the communication service from the channel. Example?
2. Upon delivery of the message to the destination. Example?

Messaging: Synchronization with the Channel


Synchronize at request submission
Synchronize at request delivery
Synchronize after processing by server
Client
Request
Transmission interrupt
Storage facility
Reply
Server
Time

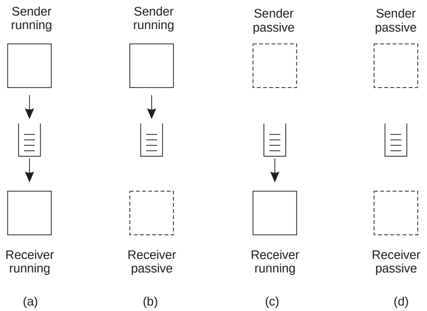
Sender

1. Upon requesting the communication service from the channel. Example?
2. Upon delivery of the message to the destination. Example?
3. Upon reception of the response (by the destination). Example?

Receiver Upon reception of the request

- May be blocking/non-blocking

Messaging: Asynchronous vs. Synchronous

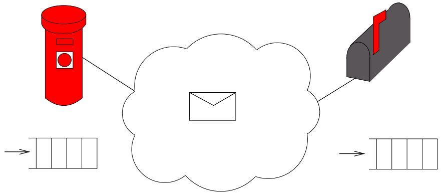


- The sender and the receiver need not be running simultaneously

Roadmap

Message Oriented Middleware (MOM)

- **Asynchronous** and **persistent** message-based communication
  - Processes need not synchronize when they exchange messages
  - Communication service (middleware) stores the messages as long as needed to deliver them
- The service is close to that of the mail service:



- The service properties may vary:
  - order;
  - reliability (persistent or not);
- Some systems provide also an abstraction similar to dicussion fora/news groups
  - **publishers** may send messages
  - **subscribers** may receive messages.

Java Message Service (JMS)

- JMS is a J2EE API :
  - Allows Java applications to access MOM in a portable way
  - It provides a maximum common divisor of the functionality provided by well known MOM providers
- JMS illustrates the MOM functionalities that may be useful for developing enterprise applications
- JMS can be integrated with the Java Transaction Service, and therefore take advantage of transactions

JMS Architecture and Model

- JMS defines 2 fundamental components:

  JMS Provider i.e. the MOM service implementation;

  JMS Client i.e. an application that sends/receives messages to a **destination** via the **JMS provider**

JMS Architecture and Model

- JMS defines 2 fundamental components:

  JMS Provider i.e. the MOM service implementation;

  JMS Client i.e. an application that sends/receives messages to a **destination** via the **JMS provider**
- To use the JMS, a client must first set up a **connection** with the provider

JMS Architecture and Model

- JMS defines 2 fundamental components:

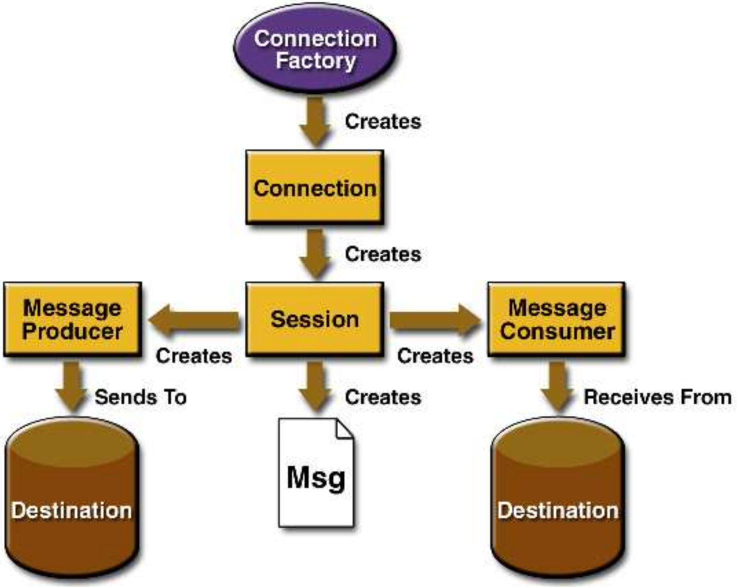
  JMS Provider i.e. the MOM service implementation;

  JMS Client i.e. an application that sends/receives messages to a **destination** via the **JMS provider**
- To use the JMS, a client must first set up a **connection** with the provider
- Cliens send/receive messages in the context of a **session**, to/from **destinations** – similar to a mailbox/message-queue

JMS Architecture and Model

- JMS defines 2 fundamental components:
  - JMS Provider i.e. the MOM service implementation;
  - JMS Client i.e. an application that sends/receives messages to a **destination** via the **JMS provider**
- To use the JMS, a client must first set up a **connection** with the provider
- Cliens send/receive messages in the context of a **session**, to/from **destinations** – similar to a mailbox/message-queue
- Sessions are usually created in the context of a connection

JMS Model



Source: Sun

Destinations and Messages

- JMS defines 2 types of *destinations*:

  Queues similar to mailboxes;

  Topics similar to news groups, i.e. messages can be read by more than one **subscriber**.

Destinations and Messages

- JMS defines 2 types of *destinations*:

  Queues similar to mailboxes;

  Topics similar to news groups, i.e. messages can be read by more than one **subscriber**.

- JMS messages have 3 parts:

  Header: has the necessary fields for identifying and routing messages;

  Properties: these are optional fields that logically belong to the header – i.e. they are meta-data

  Body: data to exchange. Can be typed.

Properties

Reliability

- Messages whose delivery mode is `PERSISTENT` are delivered **exactly-once** to a destination
- `NON_PERSISTENT` messages ensure **at-most-once** semantics

Properties

Reliability

- Messages whose delivery mode is `PERSISTENT` are delivered **exactly-once** to a destination
- `NON_PERSISTENT` messages ensure **at-most-once** semantics

Acknowledgement there are several options, including:

`AUTO_ACKNOWLEDGE` API methods automatically acknowledge delivery

`CLIENT_ACKNOWLEDGE` clients acknowledge the delivery of messages by explicitly invoking the `acknowledge` method

Properties

Reliability

- Messages whose delivery mode is `PERSISTENT` are delivered **exactly-once** to a destination
- `NON_PERSISTENT` messages ensure **at-most-once** semantics

Acknowledgement there are several options, including:

`AUTO_ACKNOWLEDGE` API methods automatically acknowledge delivery

`CLIENT_ACKNOWLEDGE` clients acknowledge the delivery of messages by explicitly invoking the `acknowledge` method

Order

- messages sent to each destination in a session;
- messages read in a session (possibly by different **consumers**);
- depends on priority, filters(**selectors**) and reliability.

JMS Topics

- Asynchronism of subscribers and publishers:
  - a message sent after a subscription may not be delivered;
  - a message sent before a subscription may be delivered.
- Subscriptions may be **durable**:
  - `PERSISTENT` messages are always delivered to a client that makes a durable subscription
  - Clients with non-durable subscriptions may loose `PERSISTENT` messages (in principle, pnly if they are inactive).

What JMS Is Not/Does Not

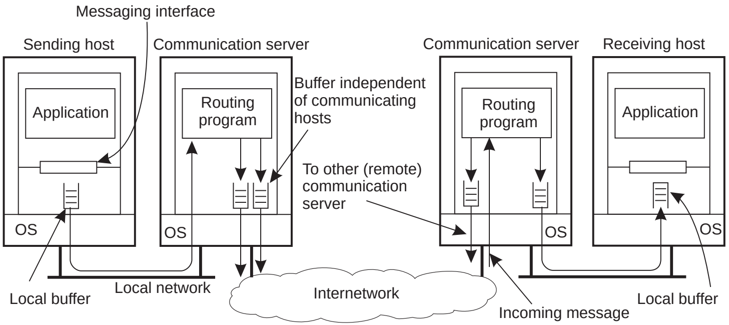
- A service: JMS is an API
  - Oracle's J2EE implementation comprises a *JMS provider*.

What JMS Is Not/Does Not

- A service: JMS is an API
  - Oracle's J2EE implementation comprises a *JMS provider*.
- Does not support:
  - Client replication (either for fault-tolerance or load balancing, e.g.)
  - Error notification
  - JMS Provider administration
  - Security – i.e. it does not offer an API to manage security attributes of exchanged messages

Messaging Service Implementation

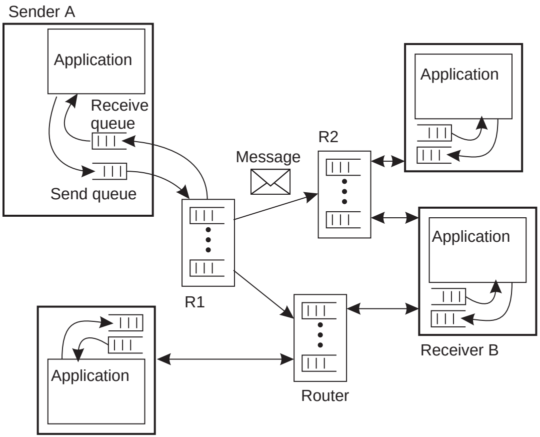
- Asynchronous communication is provided by a messaging service



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

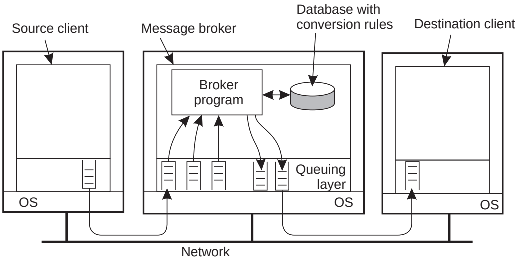
Architecture

- More advanced systems may use **message relays** to route messages to their destinations



Message Brokers

- MOM is often used for enterprise application integration. Sometimes:
  - These applications may have been designed independently
  - The syntax of the messages used by each of them may be different from one another
- **Message brokers** convert the format of the messages used by one application to the format used by another application
  - Strictly, they are not part of the communication service



MOM and Email

- The architecture of MOM services is very similar to that of the email service on the Internet.

MOM and Email

- The architecture of MOM services is very similar to that of the email service on the Internet.
- Internet's email service can be used by applications that need a plain message service:
  - SOAP, *WebServices* protocol specified by the W3C, can use either HTTP or SMTP as transport protocol
  - Some people advocate the use of SOAP/SMTP as MOM on the Internet
    - Supposedly XML facilitates the development of *message brokers*.
    - Isidro Vila Verde has a different view

Asynchronous Communication Applications

- This type of communication is appropriate for applications when the sender and receiver are **loosely coupled**. Some examples:

  Documents submission applications

  - Masks failures on the server
    - The submitter need not try again if the server is down – the messaging server hides the problem from the submitter

  Message based communication

  - *Email*;
  - *SMS* and *MMS*.

  Enterprise Application Integration

  Workflow applications

(Workflow Applications)

- These applications are related to **business processes**
  - for example, the handling of mortgage requests on a bank
- A business process can be decomposed on a set of activities whose execution depens on:
  - other activities of that process;
  - external events, which may be generated by other processes
- The different activities may be executed by independent applications
- The communication among activities can use MOM:
  - The receiving/consuming activity may not exist at the time the message is sent, because the preconditions for its execution may not yet be satisfied

Roadmap

Further Reading

- Tanenbaum e van Steen, *Distributed Systems, 2nd Ed.*
  - Section 4.3 *Message Oriented Communication*
- Oracle, *JMS Specification v1.1*