Chapter 12

Communication services

Example communication services

- Messages
- Queuing and multiplexing
- Message with reply
  - Remote method invocation
- Conversation
- Broadcast

Some objectives

- Allow modules to interact in the same way across different hosts as on a single host
- Make it easier to develop applications by capturing common generic needs
- Support a range of applications with distinct needs

Message

- The smallest unit of information meaningful to the sender and recipient
  - Recipient must be ready and listening
- Directs or informs the recipient
  - Pure push model
- Employs a send-receive protocol

Supporting deferred applications

- Queuing
  - Queuing allows a message to be sent even though the recipient is not waiting
  - The message sits in a queue somewhere in the infrastructure until it is actively retrieved by the recipient
  - Combines send-receive and request-reply protocols
Participants in message with queuing

- Sender
- Recipient
- Queue
- Request/response

Sender can send message whenever it wants
Recipient can access message whenever it wants

Multiplexing

- Multiplexing
  - Allows recipient to receive messages from two or more senders
  - How are the senders identified?
  - Presumes queuing: what if two messages are sent simultaneously?
  - Supported by Message Oriented Middleware (MOM)

Multiplexing and queuing

- Sender1
- Sender2
- Mux & queue
- Recipient

Recipient deals with one message at a time even if coming from many senders

What message is not

- Delivery may not be guaranteed
- No shared context of messages from same sender to recipient
  - Application is free to create that context on its own
  - Messages may not be delivered in the same order as sent

Example application: workflow

- Workers
- Administrative assistants
- Purchasing
- Receiving

Message with reply

- Service couples two messages:
  - Request and coupled response message
  - Sender freed of burden of associating response
- Requests information or a service
  - Pure pull model
- Immediate
  - Recipient presumes sender is waiting for reply and responds as quickly as possible
An action object-oriented programming is called a method.

Remote method invocation (RMI)

Method invocation

method: parameters return_values

Computes method()

Sender is blocked waiting for response

method: parameters → return_values

Time

Blocked: can’t do anything else

Sender Recipient Sender Recipient

Comparison

• Two-way exchange of messages
• Messages in one direction are influenced by messages in the other
• To maintain conversation
  – Messages should be delivered reliably
  – Messages should be delivered in the same order as sent
  – No coupling of replies with messages

Session protocol

A session typically provides guaranteed ordered delivery to maintain proper context

Establish Conversation

Disestablish

Some advantages of sessions

• Aids application by managing a shared context for a conversation
  – Identification, ordering, reliable delivery
• Reduces overhead
  – Example: only one name server query
• Security (Chapter 8)
• Quality of service (Chapter 11)
Examples

- What are examples of applications that would naturally leverage:
  - Messages?
  - Queuing and multiplexing?
  - Message with reply?
  - Conversation?

Streaming multimedia

- Audio coder → Audio decoder
- Video coder → Video decoder

Delay and latency

- Message latency = time from sending of message until it is received in its entirety
- Delay = time elapsing between an event (or series of events) and its (their) representation

Multimedia record-store-playback

- Audio coder → Audio decoder
- Video coder → Video decoder
- Storage

A session supports streaming multimedia

- Accumulate audio in message → Send message → Receive message → Playback audio in message
- Time → Delay

Relationship of delay and latency

- Latency
- Delay

- Delay = latency (of network) + message accumulation time
- If delay is important, keep the messages short!