Message Oriented Systems

- MOM
  - [ACKM04] Section 2.5
  - [BN97] Ch 4

- Message Brokers, EAI, Workflow
  - [ACKM04] Ch 3
Async RPC as two messages

Message: quoteRequest {
  QuoteReferenceNumber: 325
  Customer: Acme, INC
  Item: #115 (Ball-point pen, blue)
  Quantity: 1200
  RequestedDeliveryDate: Mar 16, 2003
  DeliveryAddress: Palo Alto, CA
}

Message: quote {
  QuoteReferenceNumber: 325
  ExpectedDeliveryDate: Mar 12, 2003
  Price: 1200$
}

(a)

(b)
Message Oriented Middleware

MOM Core

client application

quotations tool

inbound queue

queued messages

Message-Oriented Middleware (MOM)

MOM routes messages and maintains input queues
Multiple Servers

Queue can act as worklist for multiple server processes
Multiple Clients and Servers

- Queue can be output for multiple clients
- Load balancing on both sides of queue
But FIFO or request priority may be affected
Transactional Server

• Careful here:

• Everything the server does that affects permanent state must commit/abort as part of the same transaction as the dequeue operation

• So in general transactional MOM requires distributed txns, 2PC just like TP monitor
Service Ordering

- Server A fails/aborts after dequeue of req 1 (by server A) and req 2 (by server B)
- No message lost, but service not FIFO
Queue is not just a Relation

Suppose a DEQUEUE(Q) operation is the SQL

DELETE FROM Q WHERE Q.pri = (SELECT MAX(pri) FROM Q)

Then the sequence

DEQUEUE(Q1)
DEQUEUE(Q2)
COMMIT
DEQUEUE(Q2)
DEQUEUE(Q1)
COMMIT

is NOT SERIALIZABLE (second dequeue in each queue is a dirty read).

This pair of transactions is a DEADLOCK!

With only a single queue, second server would block behind first.
Transactional RPC

- Want a recoverable RPC
- Client and/or server may fail and be recovered
- State of RPC can be recovered from the durable transactional queues
Multi-Transaction RPC

Idea: suppose client or server fails; on recovery, can we recover the state of current RPC?
Server Recovery

• Server dequeues some inputs, does some processing, enqueues some outputs
• Suppose all processing reads/writes data under control of Txn 2 -- thus either commits or aborts along with dequeue and enqueue
• We are done!
Client Recovery

• Client may read inputs and produce outputs that are not undoable
  • read from terminal, print boarding pass, dispense $20 bills, ...

• So client maintains log on persistent storage
  • as in 2PC resource managers, log contains read results or intentions to write
  • crash recovery restores state from log
  • continue operation with minimal loss of data
Client Recovery

Assume: client runs one txn at a time.

On recovery: 4 possible states of current txn:

A: Txn 1 did not commit
   Client should resubmit request if possible

B: Txn 2 did not commit
   Client should wait

C: Txn 3 did not commit
   Client should restart txn 3

D: Txn 3 committed
   No client action required
Client Recovery

- GUID for each transaction
  - known to both client and queue mgr
- Client logs GUID of each enqueued request and each dequeued reply just before commit of its txn
- Queue mgr retains GUID of last enqueued request and dequeued reply committed by client
Client Recovery

- On recovery, client compares GUIDs in its own log with GUIDs provided by queue mgr
  - enqueue mismatch => state A
    - resubmit request using data from log
  - dequeue match => state D
    - do nothing
  - reply queue not empty => state C
    - retry -- see below
  - else
    - wait for reply message
Recovering from state C

• We don’t know how far client got in previous run ...
• Might have done something that cannot be undone/redone
  • print boarding pass
  • dispense cash
• How do we handle this? Two ways:
  • device state
  • idempotency
Recovering Using Device State

This is how to run T3 so it works properly even during recovery ... 

1. start transaction T3
2. dequeue reply
3. earlier logged device state different from current state => ask operator
4. write current device state to log
5. perform operation on device
6. commit

Assumption: every operation changes state of device!
Idempotency

- Idempotent => no harm if done twice
  - e.g. “move robot arm to this position”
- Sometimes a small tweak makes an operation idempotent:
  - send message “You just bought 100 shares of IBM” is not idempotent, but
  - send message “your request number 1234 to buy 100 shares of IBM has executed” is idempotent!
Queue Manager Summary

- Queue “DDL”
  - create/destroy/alter queue

- Queue operations
  - enqueue
  - dequeue (blocking/nonblocking, priority/FIFO)
  - enumerate

- Nontransactional queue operations
Queue Manager Summary

• Persistent sessions
  • associate queues with clients
  • persistent state aids client recovery

• Queue forwarding/routing
  • enqueue at one site
  • queue mgr transactionally forwards messages to remote queue(s) in priority or FIFO order
  • dequeue at remote site
Workflow

• A Supply Chain ...

• This is really a sequence of atomic transactions
• Why do things this way? Lots of reasons ...
  • Availability
    • need all the resources at once
  • Resource contention
    • hold only the subset of resources you needed now => better throughput
• Legacy systems
  • may not have distributed commit
ACID Properties?

• Clearly fails Isolation
  • other txns can read state between committed steps
  • e.g. transfer funds by withdraw and deposit as separate txns
  • can view this as expanding set of “consistent” states
  • highly application dependent
ACID Properties?

• Atomicity?
  • not in short term -- same argument as for isolation!
  • long term -- to roll back a sequence of transactions, execute a *compensating transaction* for each committed step
  • but this is not always possible
  • S1 makes deposit to bank acct
  • S2 makes withdrawal
  • Later we try to compensate for S1’s deposit and find insufficient funds ...
Despite weak transactional semantics, workflow extremely useful for integration of legacy applications: Enterprise Application Integration (EAI)

Of course they all run independent transactions!
How add new apps without modifying the existing ones?
Message Broker -- Routing

- Message broker determines destinations
  - based on sender identity, message type, message content
- Senders do not specify or know who the receivers are!

In basic MOM it is the sender who specifies the identity of the receivers.

With message brokers, custom message routing logic can be defined at the message broker level or at the queue level.
EAI Using Message Broker

- Add new application by altering routing specification in Message Broker ...

```
inventory management (subscriber)  ERP (subscriber)  dispatcher (publisher)  shipping (subscriber)  month-end closing (subscriber)
```

```
new PO  new PO  new PO  new PO  new PO
```

message broker
EAI Using Message Broker

- Or create new application that makes use of legacy applications

integrating application (contains the composition logic)

message broker

SmartQuotation adapter

database adapter

SmartForecasting adapter

e-mail adapter

XYZ adapter

SmartQuotation

DBMS applications

SmartForecasting

XYZ
Workflow Specifications

- **Offered=false**
  - check if offered product
  - Offered=false
  - check if worth proceeding
    - GoAhead=true
    - get quote from quotation system
      - ContractExists=false
      - get quote from supplier
      - update quotation system
        - ContractExists=true
        - ContractExists=true
        - send quote to customer
        - enter quote in forecasting system

- **Offered=true**

**variables:**
- QuoteReferenceNumber: int
- Customer: String
- Item: String
- Quantity: int
- RequestedDeliveryDate: Date
- DeliveryAddress: String
- GoAhead: Bool
- ContractExists: Bool
- Offered: Bool
Workflow Definition

1. Check if offered product
   - If Offered=false:
     - Check if worth proceeding
     - GoAhead=true
   - If Offered=true:
     - Get quote from quotation system
6. Get quote from supplier
5. Update quotation system
4. Send quote to customer
3. Enter quote in forecasting system

Variables:
- QuoteReferenceNumber: int
- Customer: String
- Item: String
- Quantity: int
- RequestedDeliveryDate: Date
- DeliveryAddress: String
- GoAhead: Bool
- ContractExists: Bool
- Offered: Bool
Workflow System

1. completed work items
2. workflow definitions
3. inbound queue
4. workflow engine
5. resource broker

outbound queues
resource 1
resource 2
resource n
Combining ...

WfMS

WfMS adapter

message broker

SmartQuotation adapter

database adapter

SmartForecasting adapter

e-mail adapter

XYZ adapter

SmartQuotation

DBMS applications

SmartForecasting

XYZ