Trường Đại học Khoa Học Tự Nhiên
Khoa Công Nghệ Thông Tin
Bộ môn Công Nghệ Phần Mềm

CTT526 - Kiến trúc phần mềm

Middleware

PGS.TS. Trần Minh Triết
tmtriet@fit.hcmus.edu.vn

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

- Nội dung của bài giảng sử dụng:

  Session 4:

  A Guide to Middleware Architectures and Technologies

  trong bộ slide Software Architecture Essential

  của GS. Ian Gorton

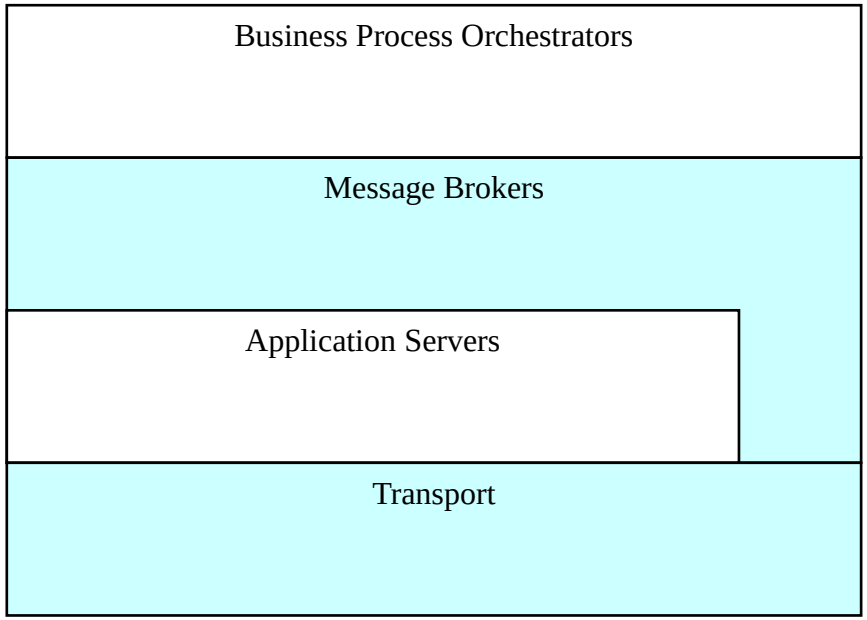
  Software Engineering Institute

  Carnegie Mellon University

Introduction

- Middleware is the plumbing or wiring of IT applications
- Provides applications with fundamental services for distributed computing
- Insulates applications from underlying platform (OS, DBMS, etc) APIs
- Lots of middleware exists
  - Different purposes
  - Different vendors
  - Different standards and proprietary technologies

Middleware Classification



BizTalk, TIBCO StaffWare, ActiveBPEL

BizTalk, WebSphere Message Broker, SonicMQ

J2EE, CCM, .NET

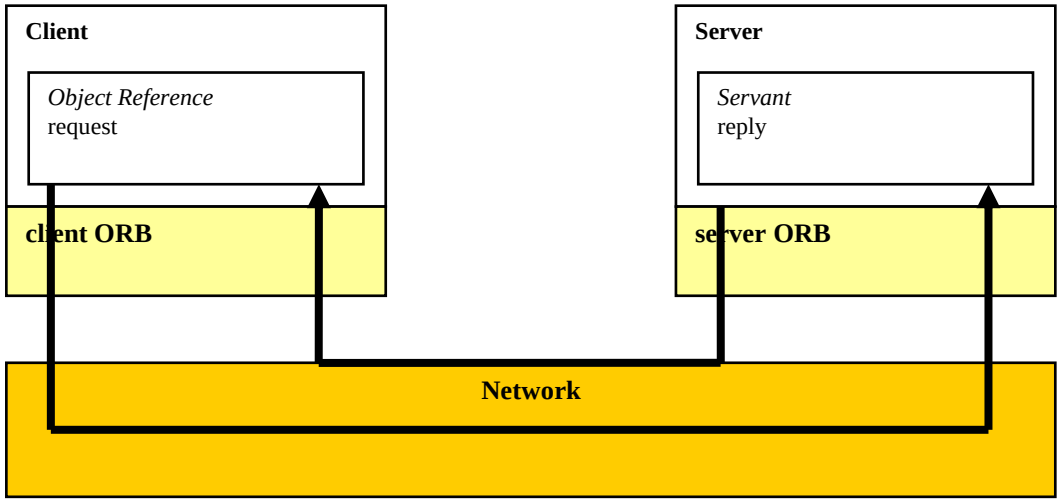
Message-Oriented Middleware, Distributed Objects Systems

Outline

- CORBA
- Message-oriented middleware
- J2EE
- Message brokers
- Business process orchestrators

CORBA

- Venerable distributed object technology
- Still widely used in telecomms, defense
- Many different implementations



CORBA Code Example

```
module ServerExample
{
interface MyObject    {       string isAlive();      };
};
```

CORBA IDL

Server

```
class MyServant extends _MyObjectImplBase
{  public String isAlive()   {    return "\nLooks like it…\n";    }
}
```

```
ORB orb = ORB.init(args, null);
MyServant objRef = new MyServant();
orb.connect(objRef);
ORB orb = ORB.init(args, null);
// Lookup is a wrapper that actually access the CORBA Naming
// Service directory – details omitted for simplicity
MyServant servantRef = lookup(“Myservant”)String
reply = servantRef.isAlive();
```

Client

CORBA – Some Thoughts

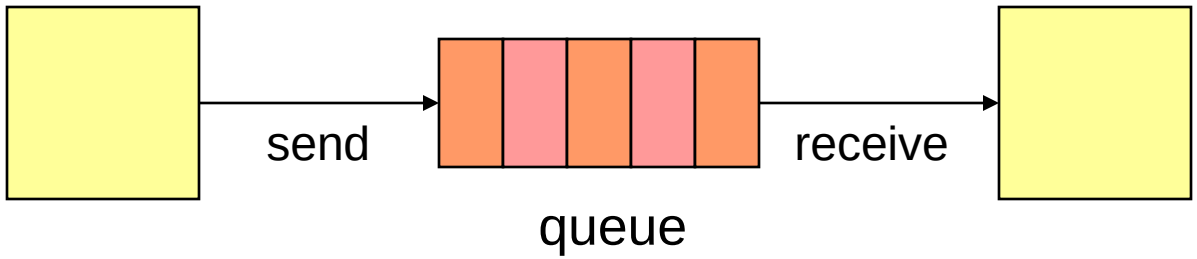
- Many associated services, eg
  - Naming
  - Notification
  - Transactions
- Synchronous technology, client-server relatively tightly coupled
- Remote calls can/will fail
- State management in server objects creates ‘interesting’ recovery issues

Messaging - MOM

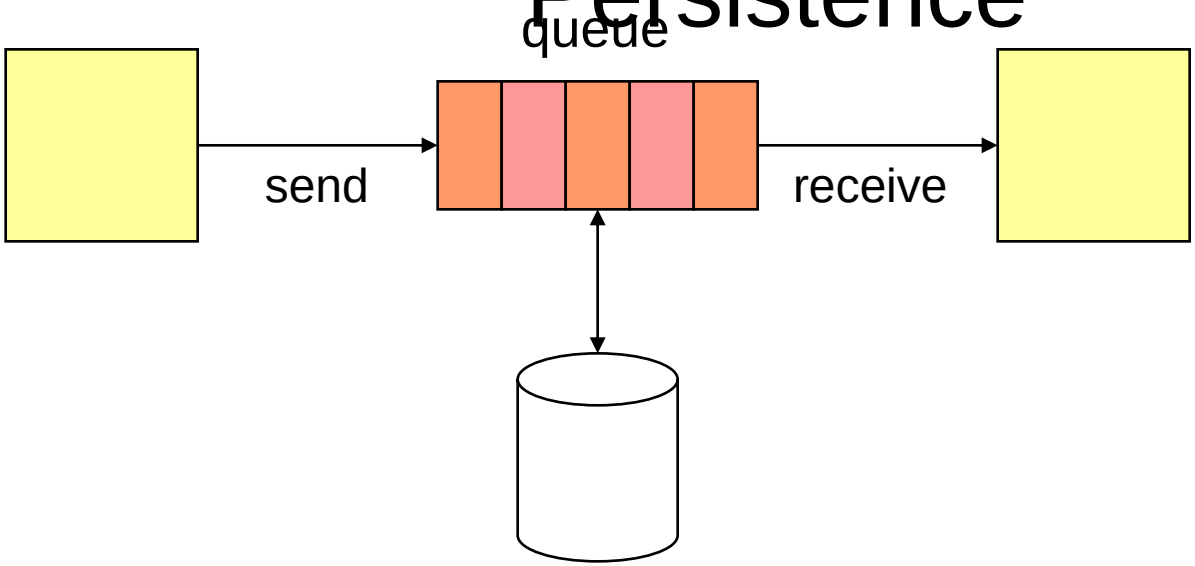
- Basic Message Oriented Middleware (MOM) provides features like:
  - Asynchronous communications between processes, applications and systems
  - Send-and-forget
    - Delivering messages despite failures
  - Transactional Messaging
    - Deliver all messages in a transaction, or none
  - Persistence
    - Messages can be logged at the server and hence survive server failure

Basic Messaging

- Send (queue, message)
  - Put message onto queue
- Receive (queue, message)
  - Get message from queue
- No dependency on state of receiving application on message send

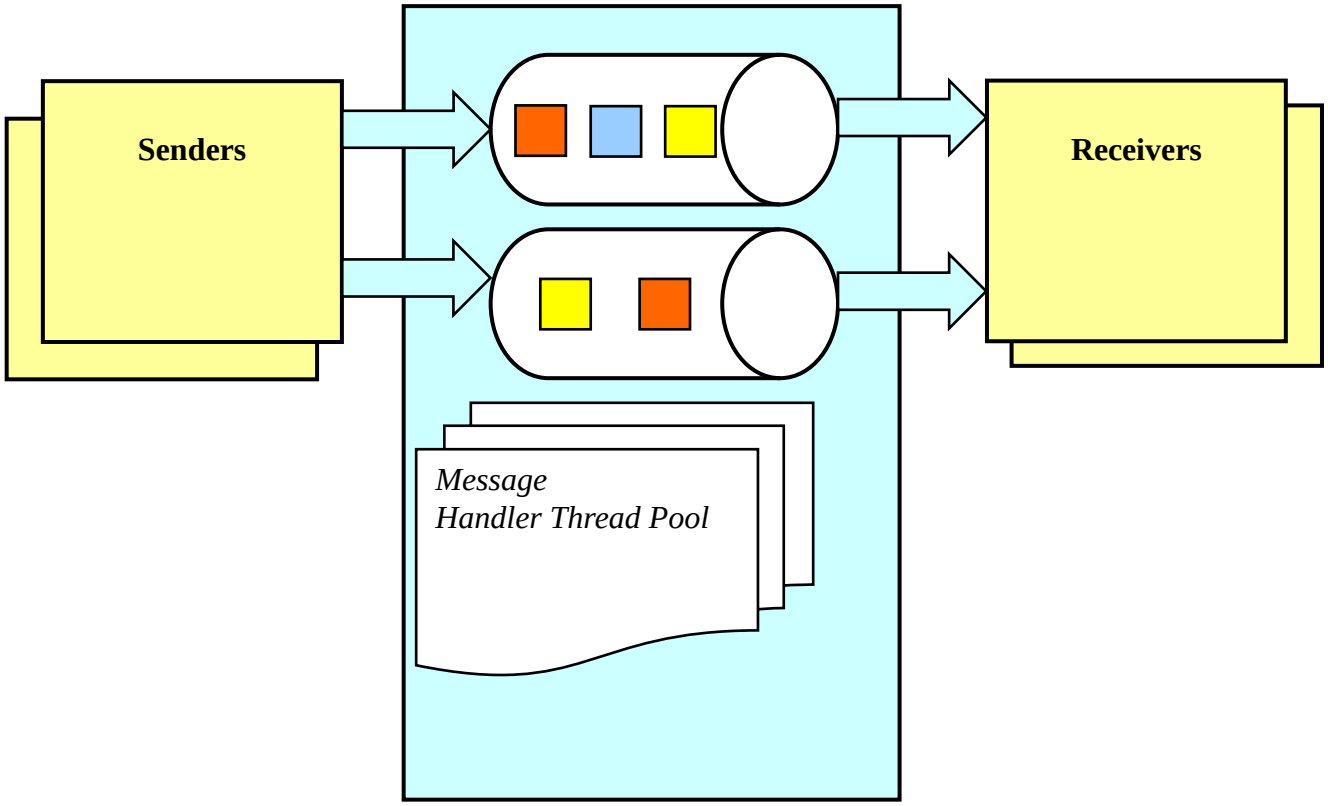


Persistence



- Receipt of message at queue implies message is written to disk log
- Removal of message from queue deletes message from disk log
- Trade-off performance versus reliability

MOM Server

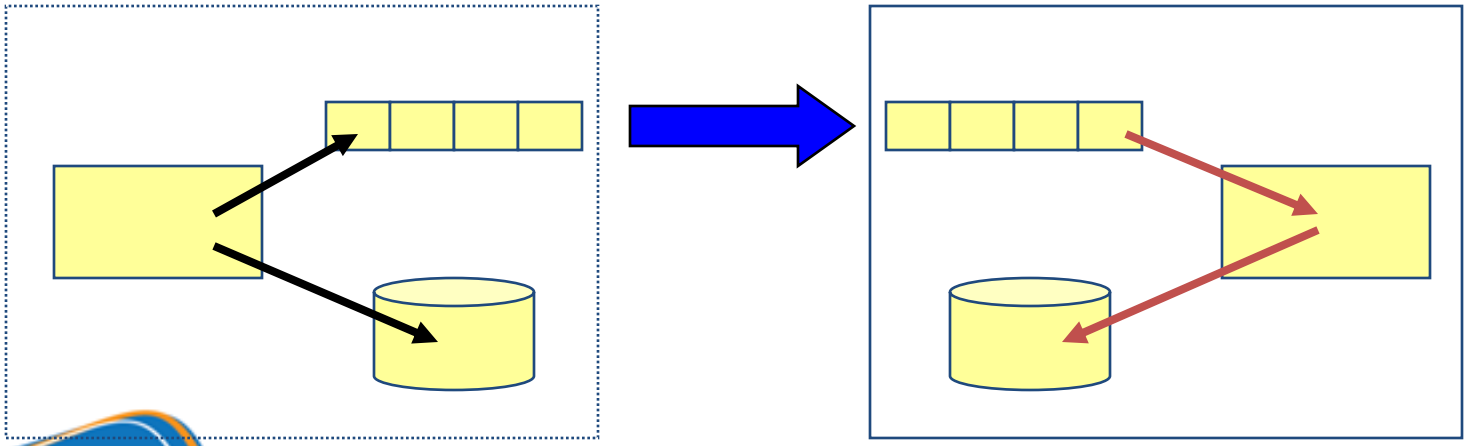


Peer-to-peer MOM technologies are the alternative design

MOM Transactions

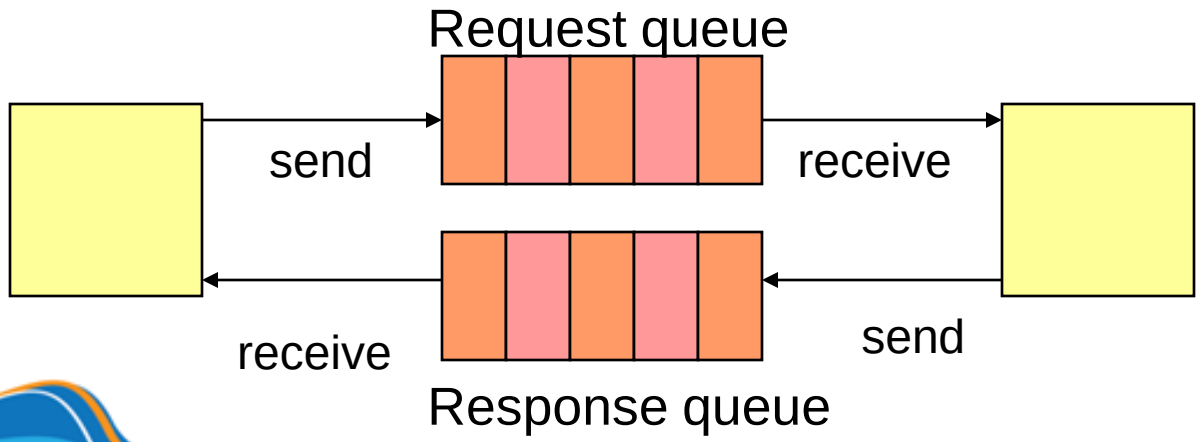
```
Begin transaction
...
update database record
put message on queue
...
commit transaction
```

```
Begin transaction
...
get message from queue
update database record
...
commit transaction
```

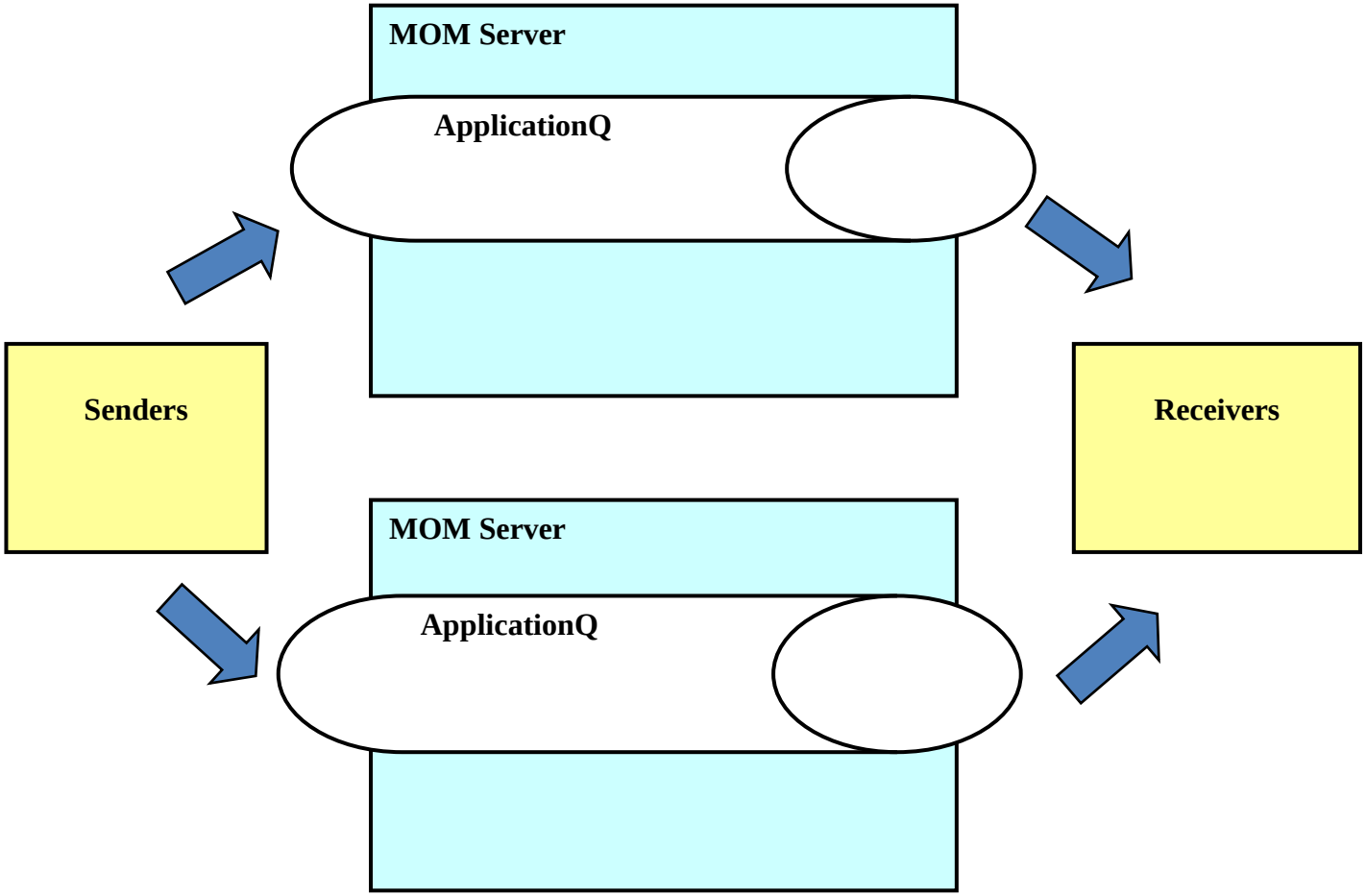


MOM Transactions

- Sender and receiver do *not* share a transaction
  - Rollback on receiver does not affect the sender (already committed)
  - 'Synchronous' operations are not atomic
    - Request/response is 3 transactions not 1
      - Put to request queue
      - Get from request queue, put to response queue
      - Get from response queue



Scaling MOM

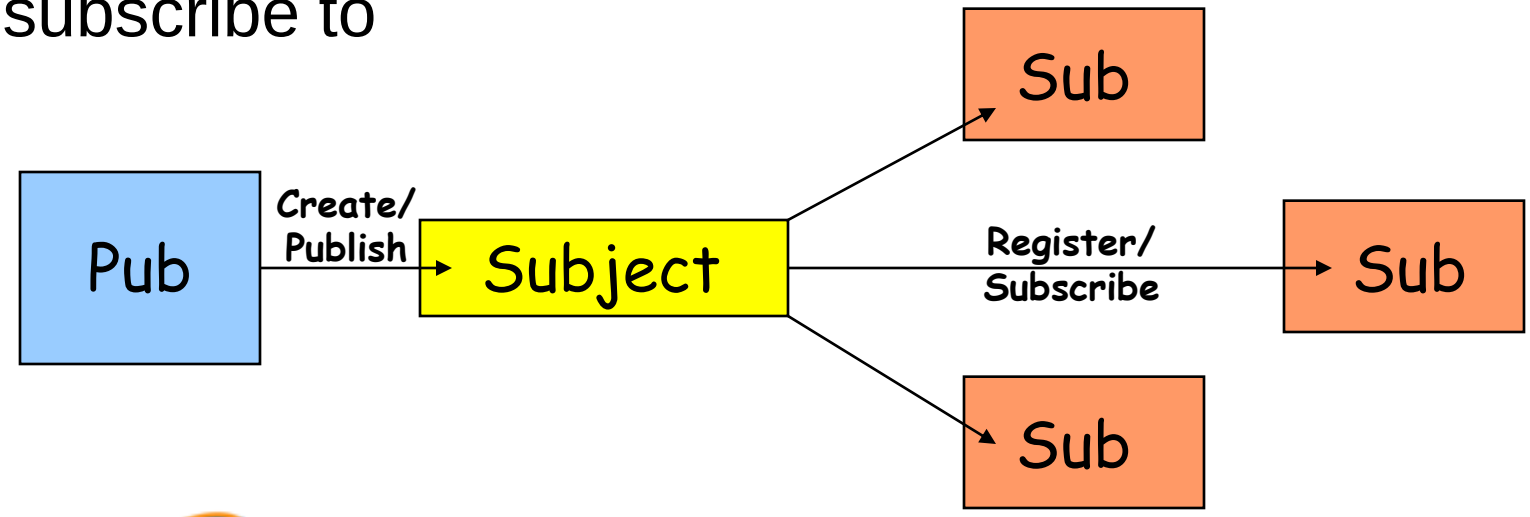

MOM Server
ApplicationQ
Senders
Receivers
MOM Server
ApplicationQ

Messaging – Some thoughts

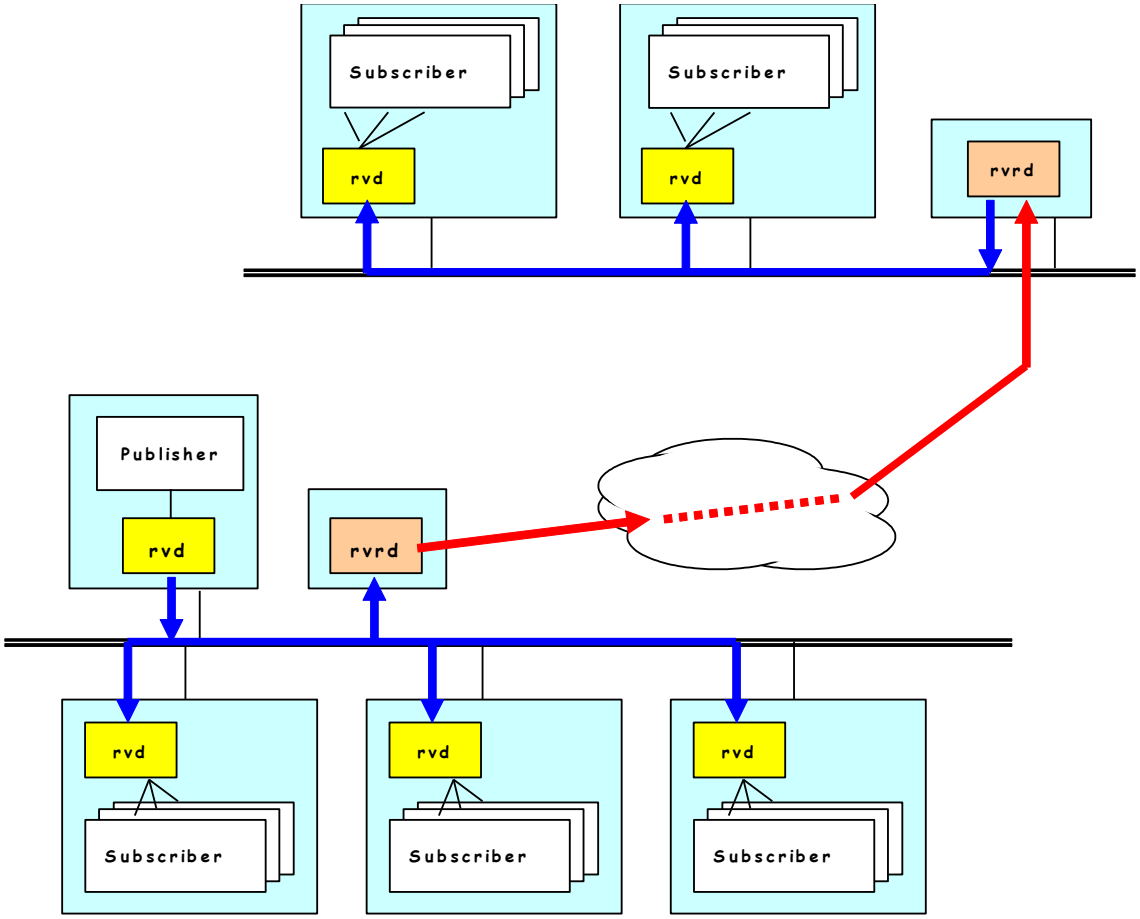
- Highly attractive asynchronous technology
- Supports loosely-coupled, dynamic applications
- Scales well, high throughput possible
- Many implementations, various qualities of service
  - caveat emptor

Publish-Subscribe Messaging

- Extension of MOM to provide 1-to-N, N-to-1, and N-to-N communications
- Messages are ‘published’ to logical *subjects* or *topics*
- Subscribers receive all messages from subjects they subscribe to

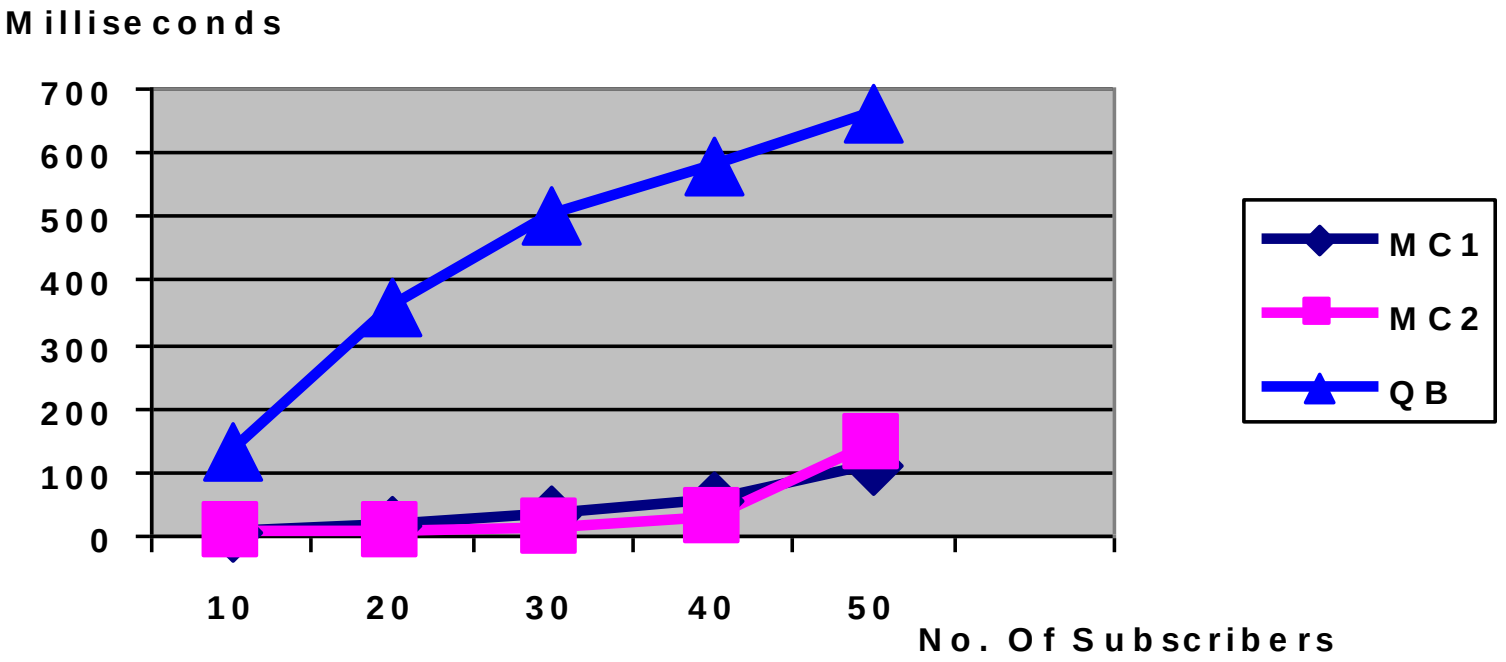


Publish-Subscribe with Multicast

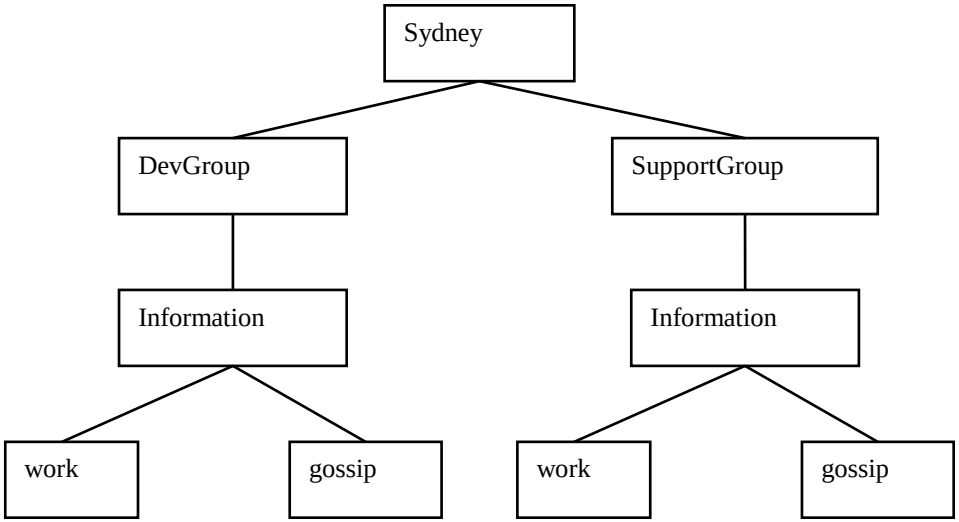


Based on TIBCO Rendezvous

Performance



Subject/Topic Naming



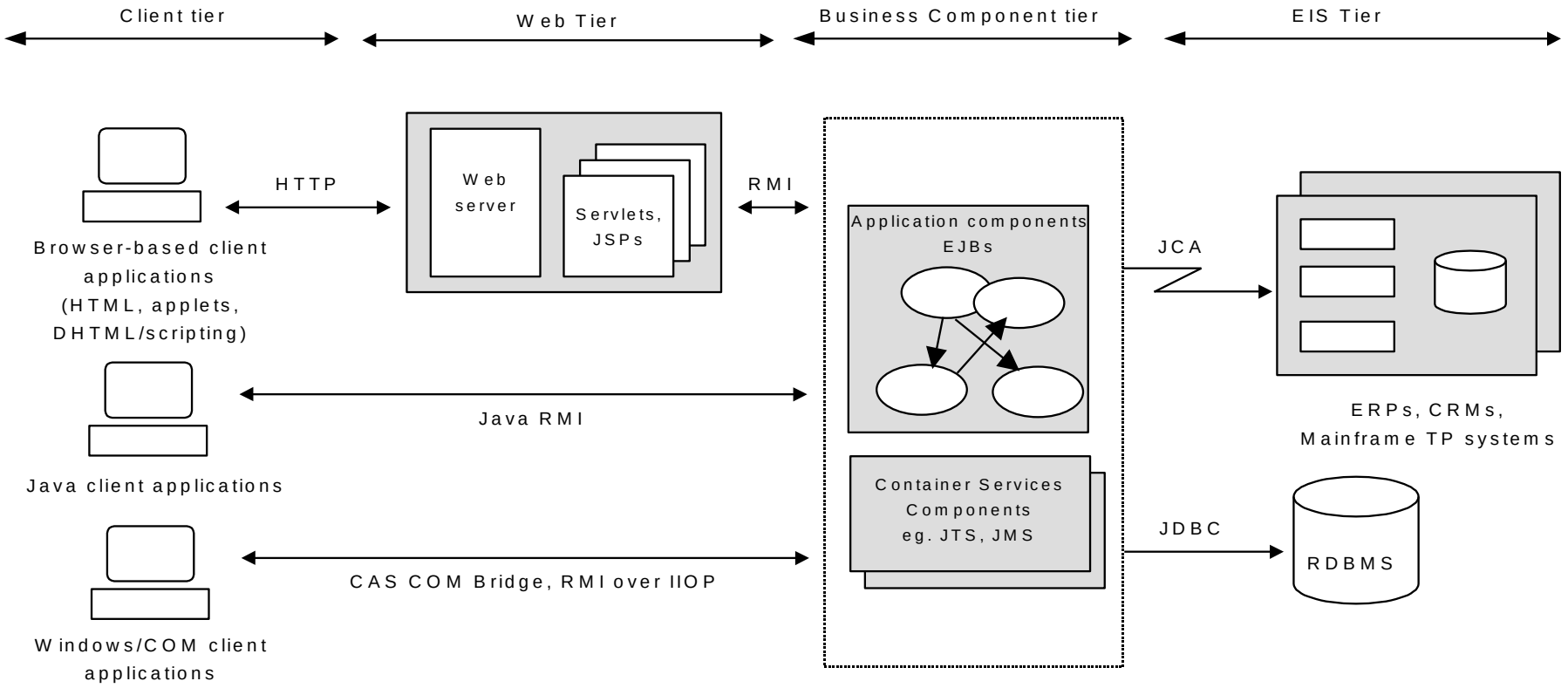
Sydney
Sydney/DevGroup
Sydney/DevGroup/Information
Sydney/DevGroup/Information/work
Sydney/DevGroup/Information/gossip
Sydney/SupportGroup
Sydney/SupportGroup/Information
Sydney/SupportGroup/Information/work
Sydney/SupportGroup/Information/gossip

Sydney/*/Information
Sydney/DevGroup/*/*
Sydney/DevGroup/**

Publish-Subscribe – Some Thoughts

- Highly decoupled messaging style
  - Publishers don't know about subscribers
  - Subscribers don't know who is publishing
  - Publishers and Subscribers can dynamically appear and disappear
- Issues –
  - Reliability
  - Transactions
  - Security
  - Performance

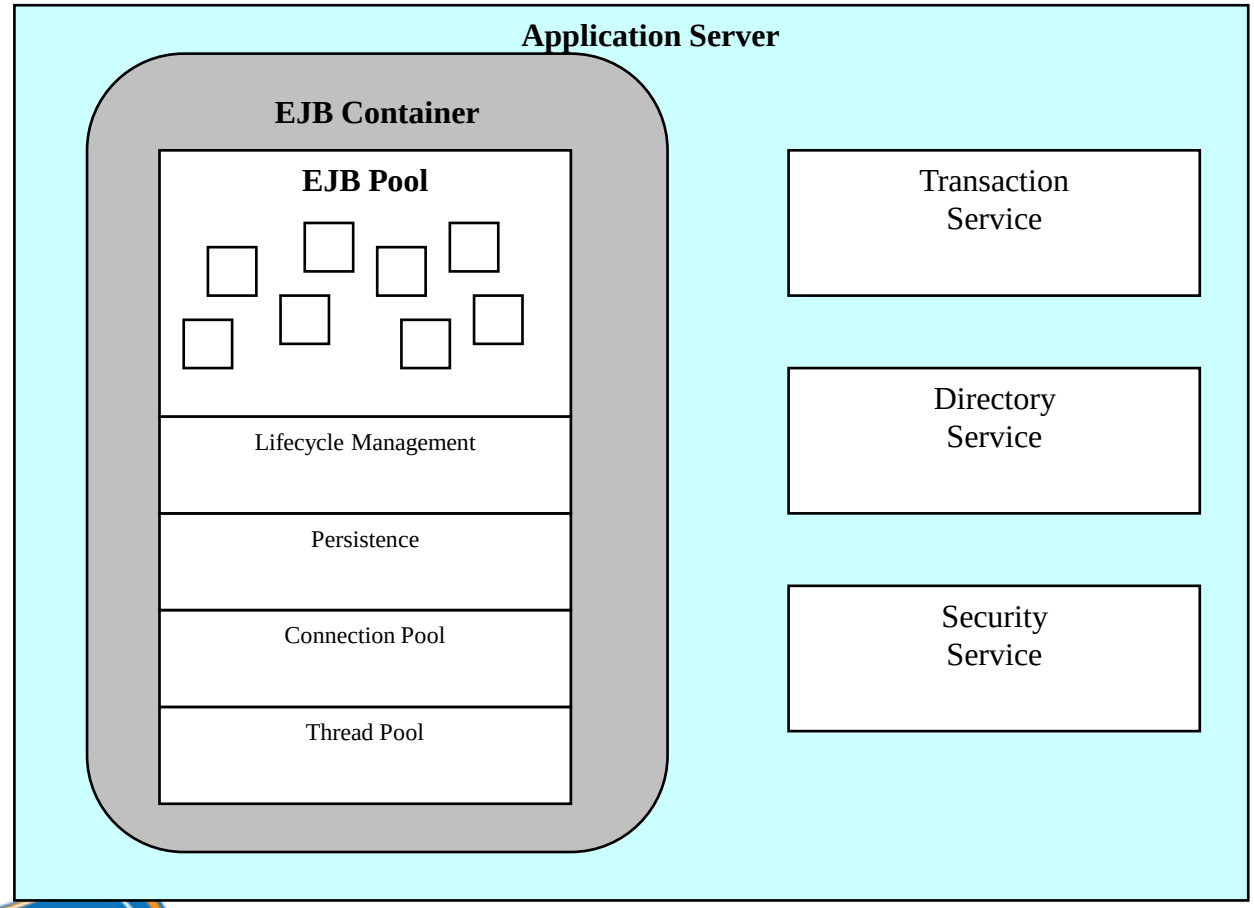
J2EE Overview



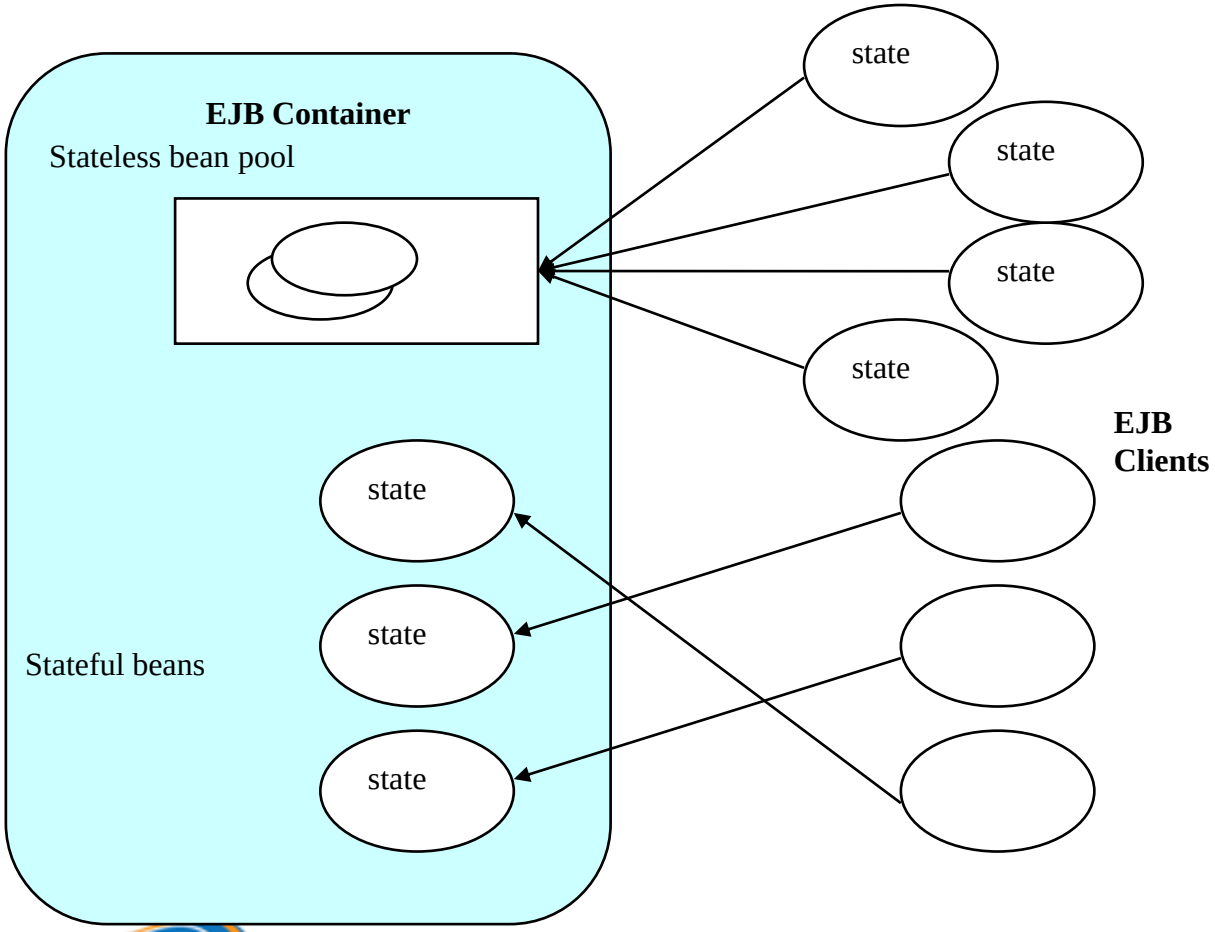
J2EE Application Server

- In J2EE, the application server container provides the execution environment for the J2EE-specific components
  - EJBs
  - Message-driven beans
  - Connectors
- Container provides additional services for hosted components
  - Transactions
  - Security
  - Directory
  - Threading
  - Connection pooling

EJB Container



Beans and State



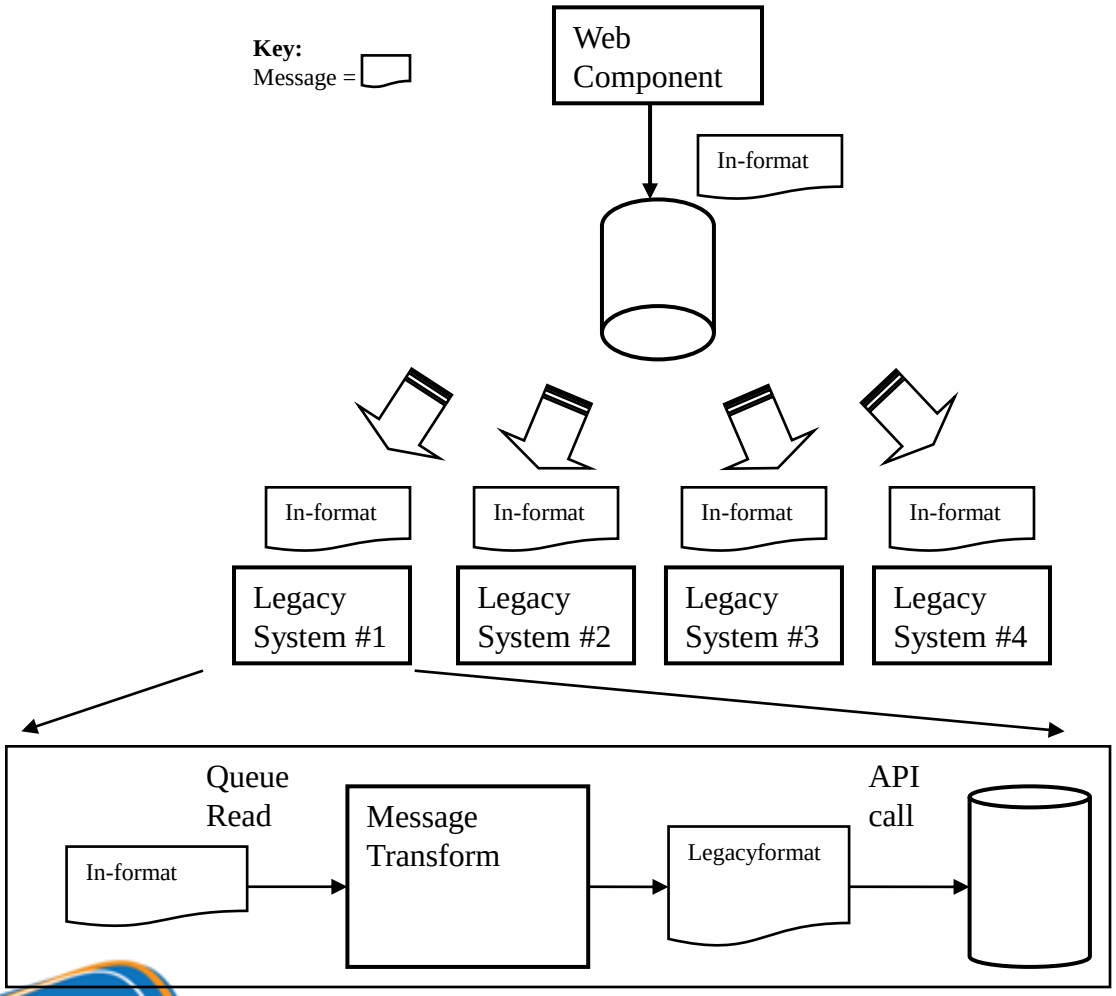
Deployment Descriptors

```
<ejb-jar>
<enterprise-beans>
 <session>
                    <ejb-name>EntityStock.BrokerHome</ejb-name>
                    <home>db.entitystock.BrokerHome</home>
                    <remote>db.entitystock.Broker</remote>
                    <ejb-class>db.entitystock.BrokerBean</ejb-class>
                    <session-type>Stateless</session-type>
                    <transaction-type>Container</transaction-type>
  </session>
</enterprise-beans>
<assembly-descriptor>
  <container-transaction>
            <method>
              <ejb-name>EntityStock.BrokerHome</ejb-name>
              <method-intf>Remote</method-intf>
              <method-name>*</method-name>
            </method>
            <trans-attribute>Required</trans-attribute>
  </container-transaction>
</assembly-descriptor>
</ejb-jar>
```

J2EE – Some Thoughts

- Standards-based, multiple vendors, portable
- Good open source technology available
- Quality of implementations varies considerably
- Java only, wide platform support
- Performance is good, but varies between suppliers
- Scalable, fail over support through clustering
- Good integration with web technologies
- Supports various design patterns, flexible but more complex (e.g. stateful beans/scalability, entity beans)
- Standards evolving, need to monitor

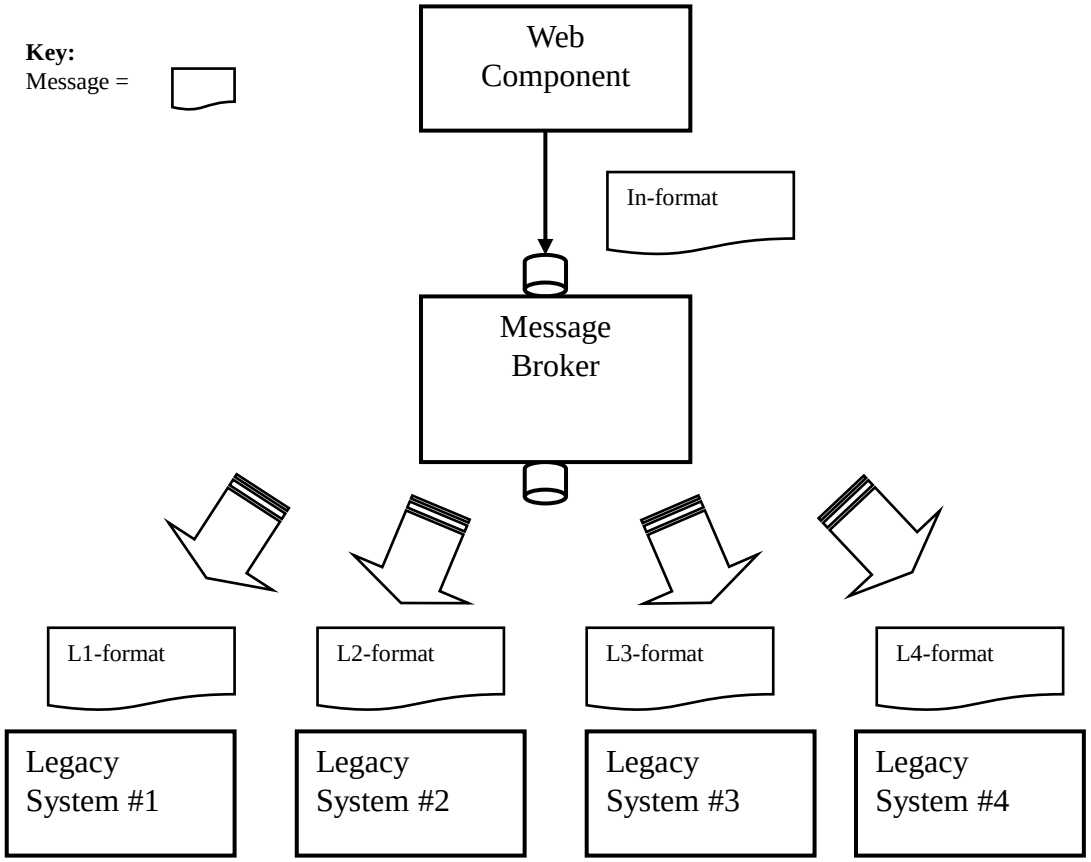
Message Brokers - Motivation



What if …

- the common *In-format* message format changes?
- any legacy system API changes?
- any of the transformations needs modifying?

Alternative Solution



- Transformations in broker
- Simplified endpoints
- Decouples Web and legacy components

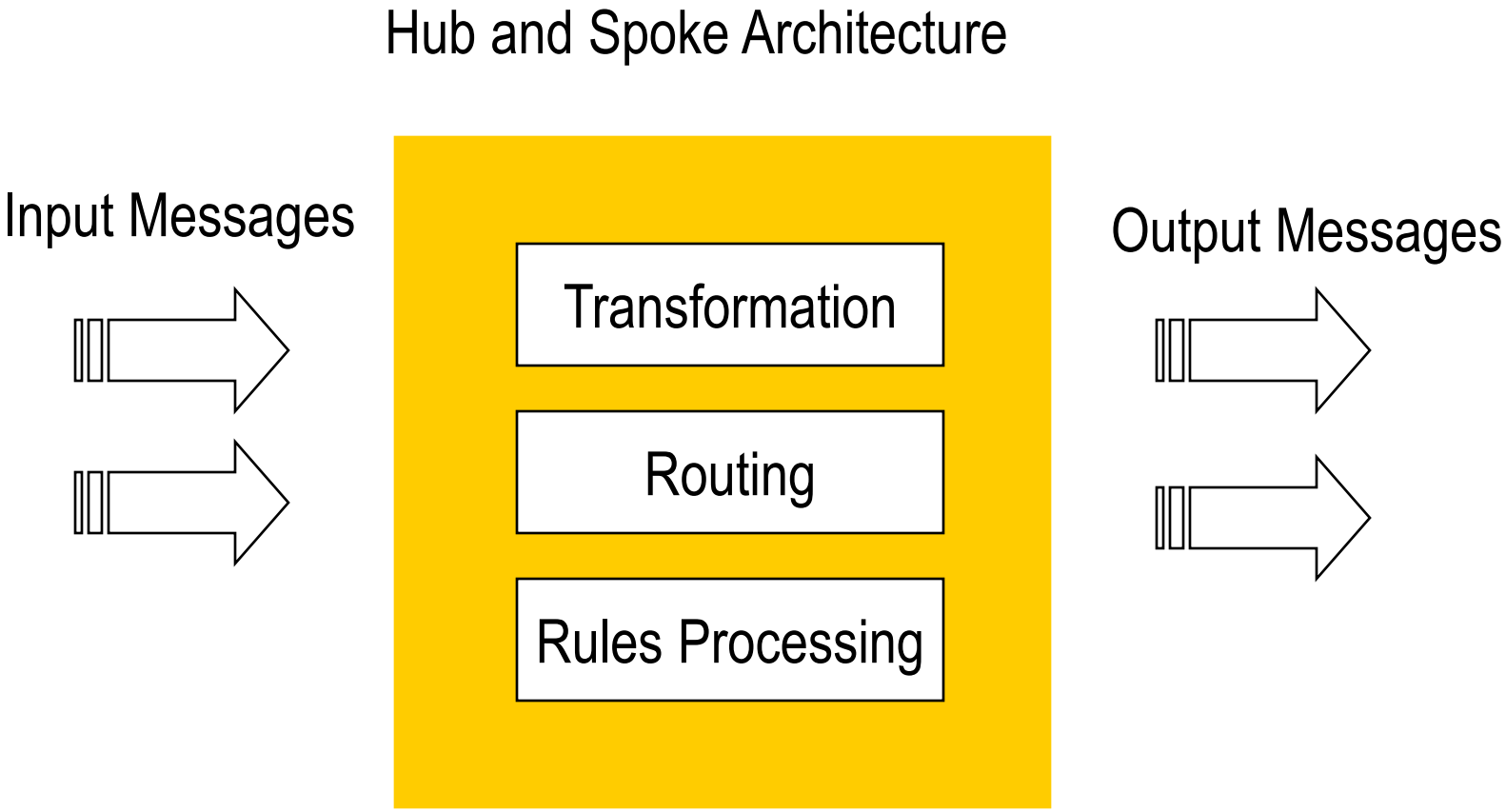
Message Brokers

- Developed specifically for Enterprise Application Integration (EAI)
- Add new layers of functionality to MOM
  - Message transformation
  - Rules engine
  - Intelligent routing
  - Adapters
- Typically (but not necessarily) built on top of a MOM layer

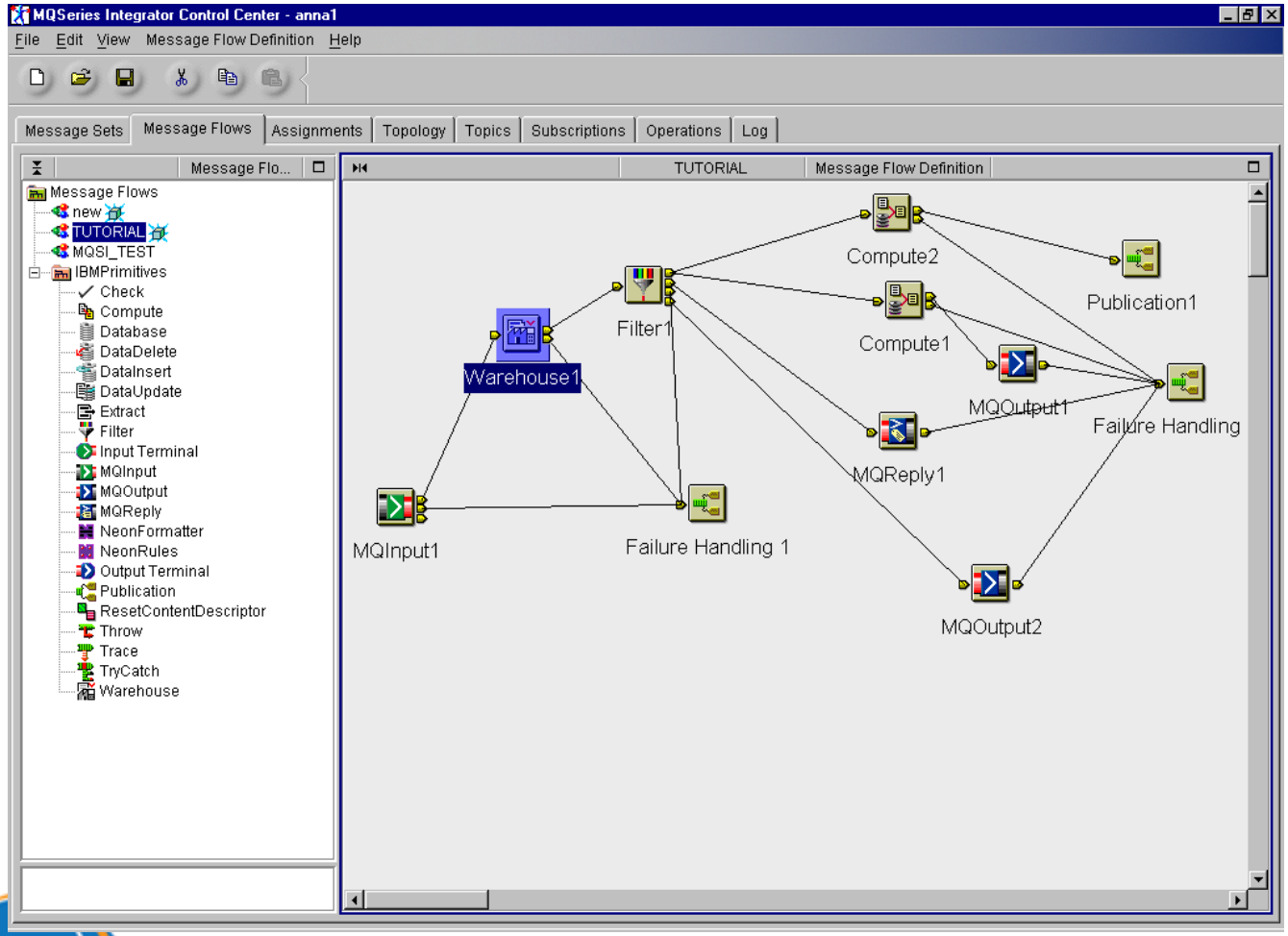
Message Broker Features

- Message transformation – transform between different source/target formats
  - Graphical message format definition and mapping tools
  - High performance transformation engines
  - Message format repositories
- Intelligent routing
  - Route messages based on message content
- Rules Engine
  - Scripting language, built-in functions
  - Application programming environment

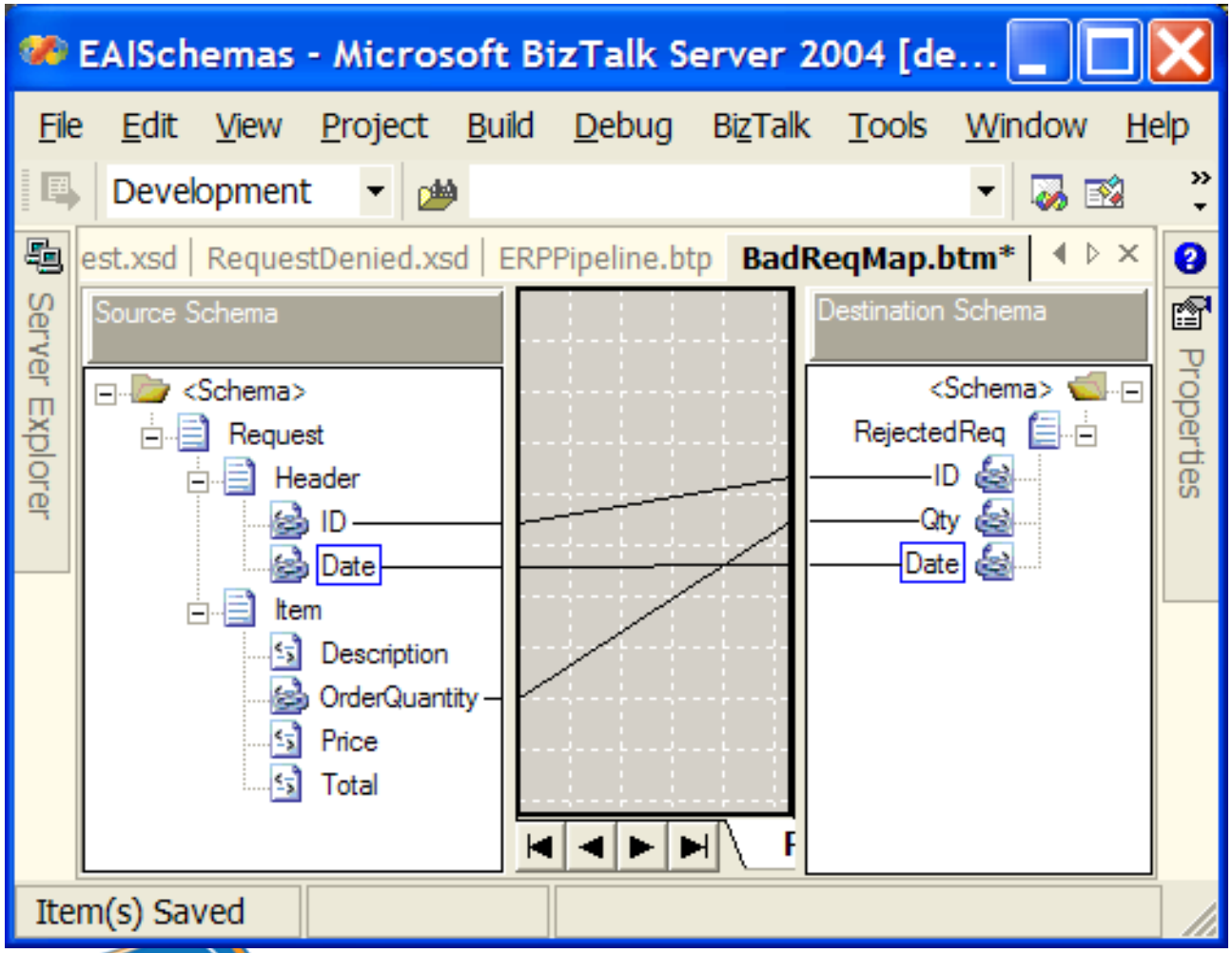
Message Brokers



Example - WMQI



BizTalk Mapping Tool



Adapters

- An adapter is a component that resides between the message broker and the source/target systems
- Simplify complexity of end system interface through an abstraction layer
- Thin adapters - simple wrappers
- Thick adapters
  - Programmable
  - Abstract representation of services and meta-data
- Centralized adapters co-located with broker
- Distributed adapters execute in own process and may be located with source/target system

Message Brokers – Some Thoughts

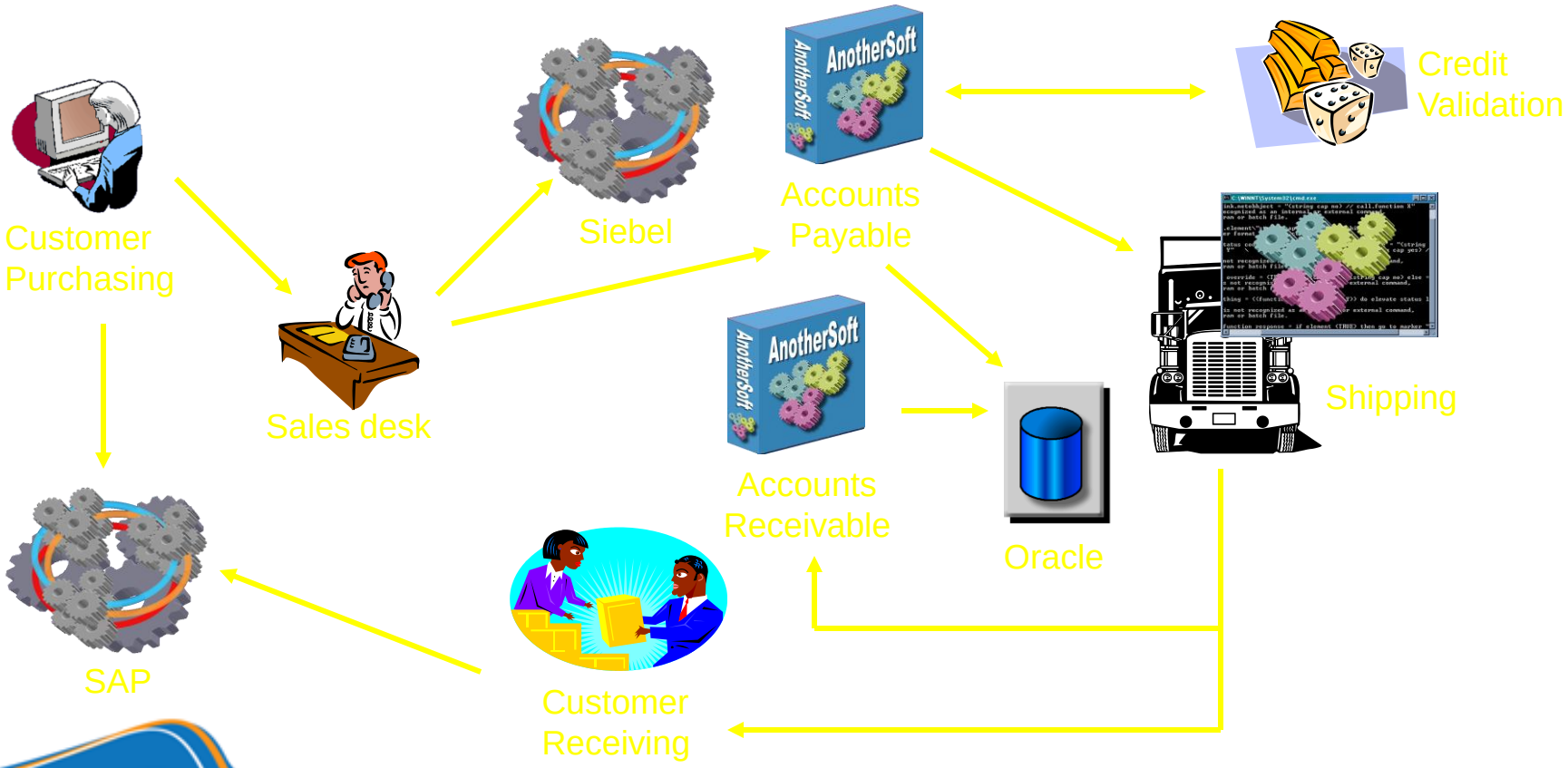
- Embeds transformations/routing in broker
  - Can get complex
- Possible scaling issues
  - Need to replicate brokers
- Failure handling
  - Lightweight, rarely designed to recover from failure
- Often proprietary technology
  - Good open source, standards-based like Mule now available

Business Process Orchestration

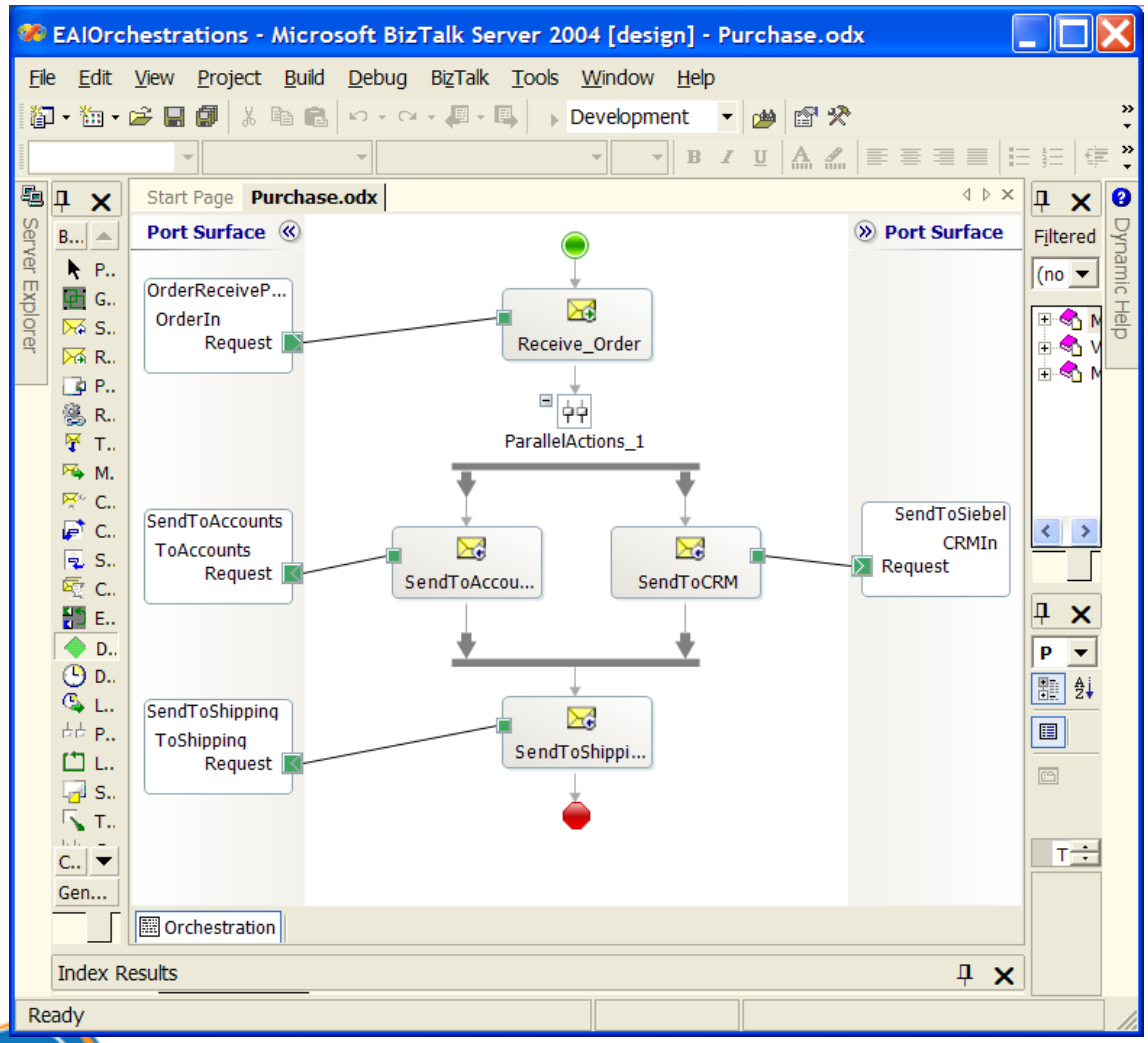
- Commonly known as workflow
- Aim is to automate business processes which need to access data and business logic across disparate back-end applications
- Builds on EAI to ensure business processes are executed in the defined order using the required data
- Builds on middleware providing:
  - Process execution engine
  - Visual process definition tools
  - Process monitoring tools

Typical Scenario

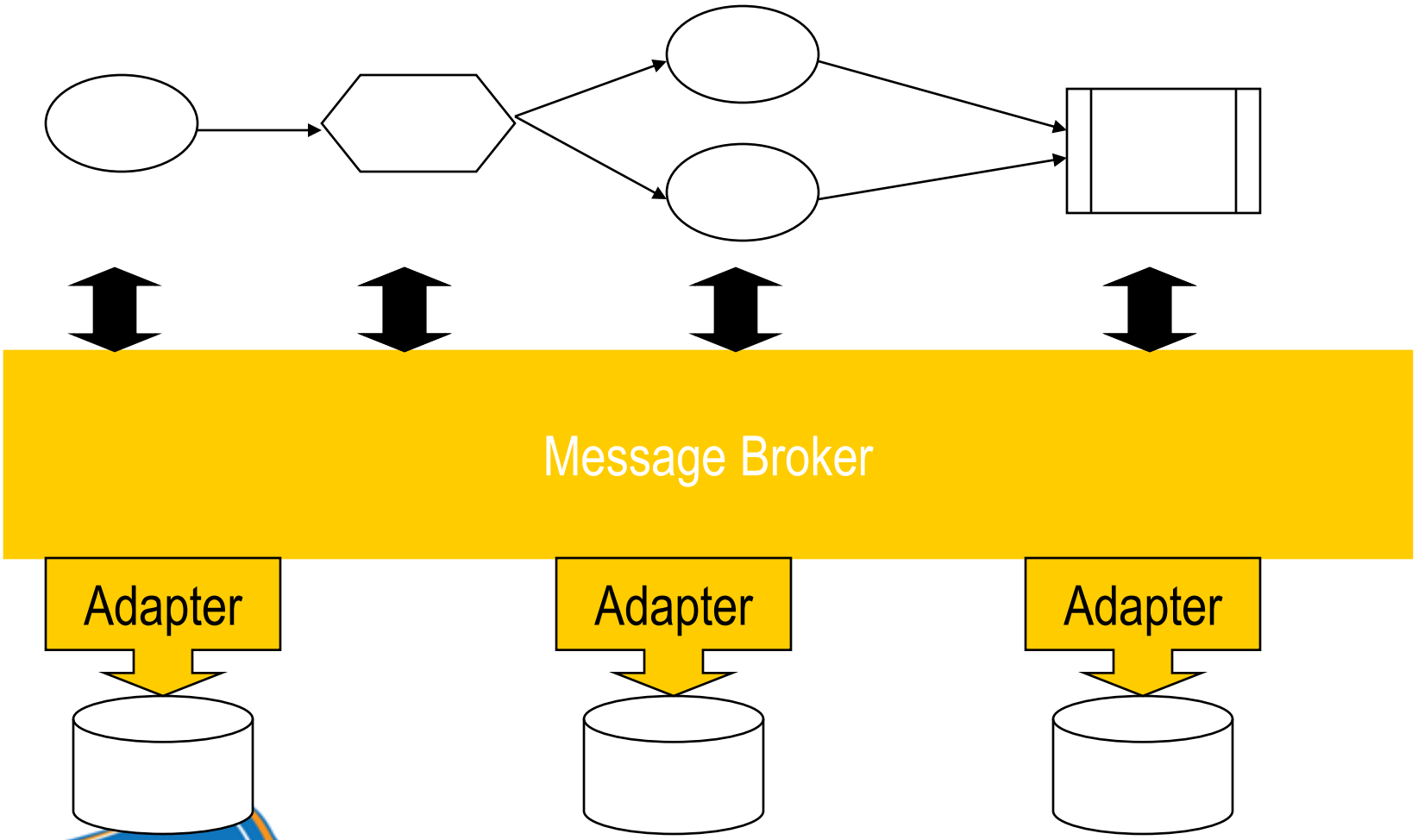
- Business process automation



Example - BizTalk



BPO Architecture

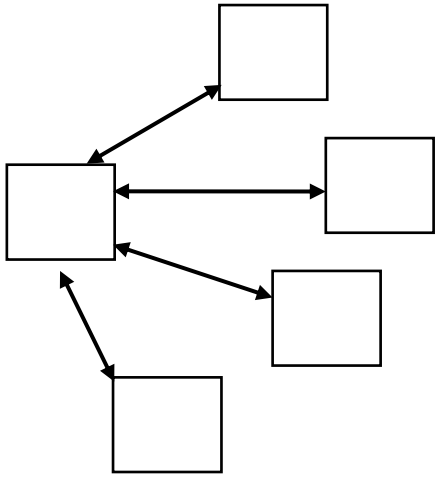


BPEL

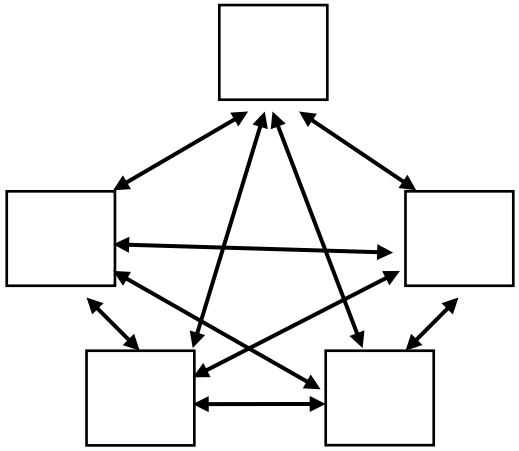
- Web Services standard for describing workflows
- Many design and execution tools
  - Eg ActiveBPEL
- Version 2.0 is a significant improvement

Integration Issues – Point-to-Point

- Point-to-Point evolution
- Spaghetti architecture, hard to modify potentially (N^2-N) interfaces



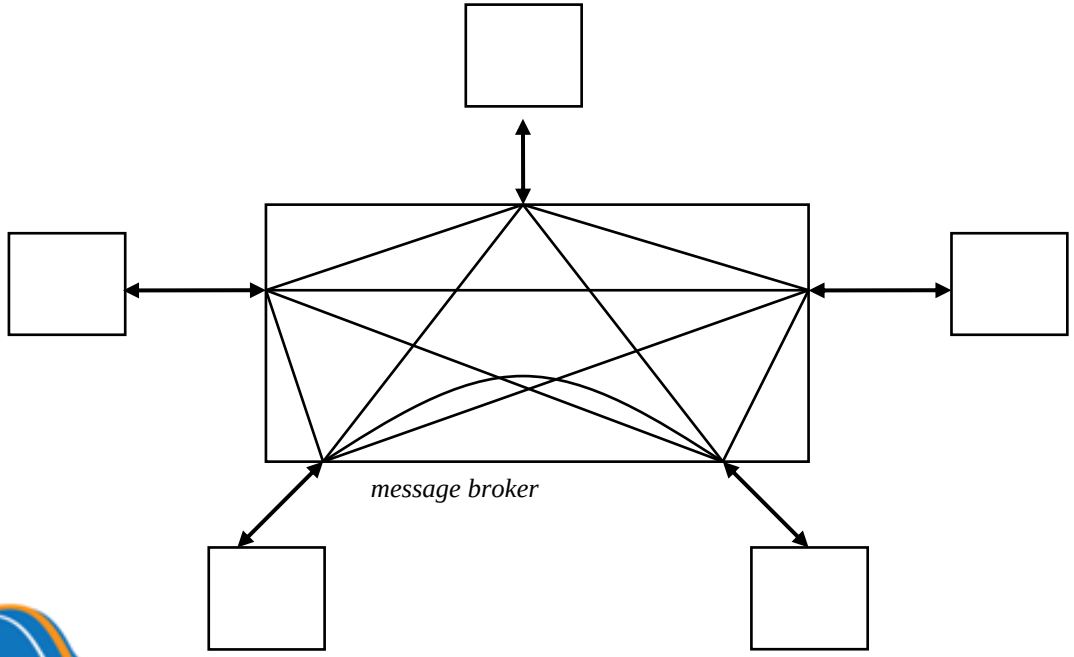
1 business process =
4 interfaces



5 business processes =
20 interfaces

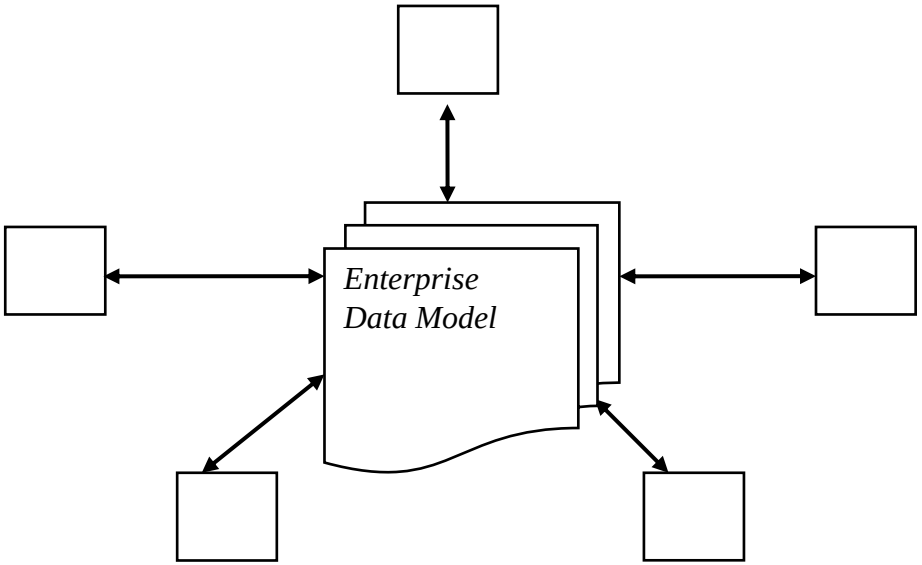
Broker Spaghetti

- No free lunch …
- Just relocates the spaghetti



Enterprise Data Model

- Source sends message to target with common message format as payload.
- Target receives message and transforms common format into its own local data representation.
- 2xN transformations, no broker needed
- Getting agreement is the tough bit …



Summary

- Middleware:
  - makes building complex, distributed, concurrent applications simpler.
  - institutionalizes proven design practices by supporting them in off-the-shelf middleware technologies.
- Architect's job is to 'mix n'match' technologies to create appropriate solutions
  - Analyze trade-offs
  - Open-minded (no hammer/nail thinking)
  - No good/evil, its just technology