

# Middleware

Maham Khan

Lecture 10&11

**Systems-Integration&  
Architecture**

## What is Middleware?

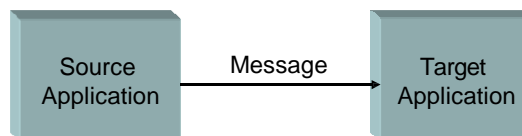
- Middleware is any type of software that facilitates communication between two or more software systems
  - Can be simple communication connection between applications
  - Can be as sophisticated as information sharing and logic execution mechanisms
- Middleware is a technology that allows us to move information between multiple enterprises

## Middleware Models

- There two types of middleware models
  - Logical middleware model
    - Depicts how information moves throughout the enterprise conceptually
  - Physical
    - Depicts both the actual method of information movement and the technology employed

## Point-to-Point Middleware

- Point-to-point middleware uses a simple pipe to allow one application to link to another application
  - Provides point-to-point connection between a source and a target application
- Disadvantages
  - Inability to bind more than two applications
  - Lacks ability to house application logic
  - Lacks ability to change messages as they flow through the pipe



## Many-to-Many Middleware

- Links many applications to many other applications
  - Can deal with more than two source or target applications
- This capability make it the best option for application integration
  - It provides flexibility and applicability to the application integration problem domain

## Synchronous vs Asynchronous

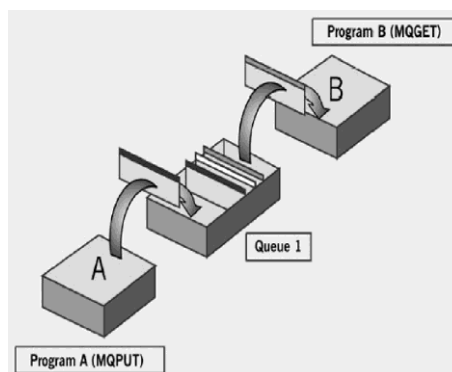
- Asynchronous middleware moves information between one or many applications in an asynchronous mode
  - i.e., the middleware software is decoupled from the source or target applications
  - Applications are not dependent on other connected applications for processing.
  - Application can always continue processing, regardless of the state of the other applications
- Synchronous middleware is tightly coupled to applications
  - The applications are dependent on the middleware to process one or more functions calls at a remote application
  - Calling application must halt processing to wait for the remote application to respond
- Asynchronous is preferred over synchronous application integration solution
  - Synchronous middleware faces problems such as network or remote server problems
    - Therefore, application has to stop processing
  - Synchronous middleware eats up bandwidth because several calls must be made across the network in support of a synchronous function call

## Communication Models

- Connection-oriented communication
  - Two parties connect, exchange messages and then disconnect
  - Typically synchronous process, but it can be asynchronous
- Connectionless communication
  - Calling program does not enter into a connection with the target process
  - Receiving application simply acts on the request, respond if required
- Direct communication
  - Middleware layer accepts the message from the calling program and passes it directly to the remote program
    - Usually synchronous in nature
- Fire and forget
  - This model allows the middleware user to “fire off” a message and then “forget” about it,
    - without worrying about who receives it or even if the message is ever received
  - The purpose of this model is to allow a source or target application to broadcast specific type of messages to multiple recipients

## Communication Model: Queued communication

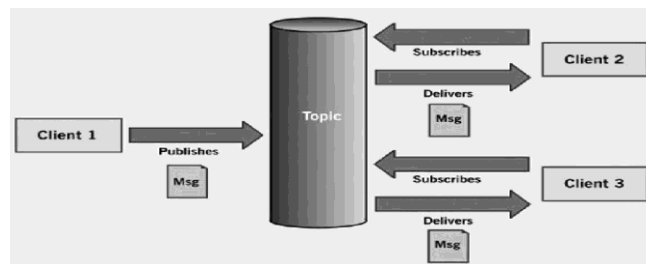
- Generally requires a queue manager to place a message in a queue
- The remote application then retrieves the message, either shortly after it has been sent, or at any time in the future
- Receiving application need not to be active when calling application sends the message
- Does not block neither remote nor calling application from proceeding with processing



Source for figure:  
<http://www.cw360ms.com/research/butler/middlewareoptions.pdf>

## Communication Model: Publish/ subscribe

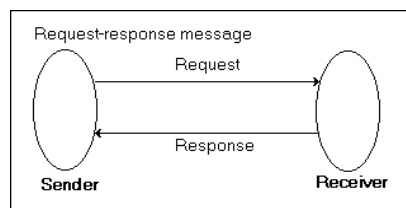
- Frees an application from the need to understand anything about the target application
- Publisher is the provider of the information about a topic
- Publisher sends information it desires to share to any interested applications (subscribers)
- Publisher does not need to understand anything about applications that are interested in the information



Source for figure: <http://www.cw360ms.com/research/butler/middlewareoptions.pdf>

## Communication Model: Request response

- As name implies, a request is made to an application using request response middleware, and it responds to the request
- Includes any middleware that can facilitate a response from a request between applications, such as integration servers or application servers

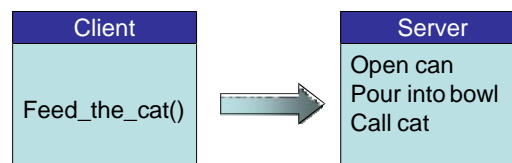


## Types of Middleware

- RPC
- Message-oriented middleware (MOM)
- Distributed objects
- Database-oriented middleware
- Transactional middleware
- Integration servers

## Remote Procedure Calls (RPC)

- Oldest type of middleware
- Provide ability to invoke a function within one program and have that function execute within another program on a remote machine
- RPC are synchronous
  - i.e., RPC must stop the execution of the program
- They also require more bandwidth than other types of middleware
- Advantage of RPC is its simplicity for mechanism and programming
- Disadvantage is are its huge performance cost and inability to scale

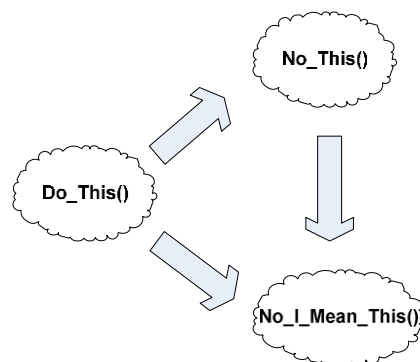


## Message-Oriented Middleware (MOM)

- MOM is queuing software that uses messages as a mechanism to move information from point to point
- MOM uses the notion of messages to communicate between applications,
  - Direct coupling with the middleware mechanism and the application is not required
- MOM rely on asynchronous paradigm
  - This allows application to function independently
    - i.e., continue processing after making a middleware service request
  - Message is dispatched to a queue message, which ascertainsthat message is delivered to its final destination.
  - Messages returning to the calling application are handled when the calling application finds the time
- Managing are easy to manage using MOM as it has structure (schema) and content (data)
  - MOM can be thought as one-record database that move between applications through message-passing mechanisms
- MOM supports two communication models
  - Point-to-point
  - Message queuing (MQ)

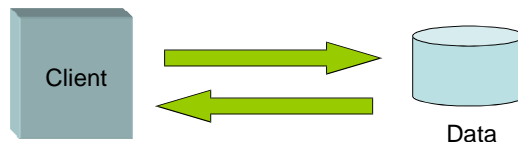
## Distributed Objects

- Small application programs that use standard interfaces and protocols to communicate with one another
- Provide mechanisms for application development, providing enabling technology for enterprise, or enterprise-wide method sharing
- There are two types of distributed objects in market today
  - Common Object Request Broker Architecture (CORBA)
  - Component Object Model (COM)



## Database-Oriented Middleware

- Facilitates communication with a database, whether from an application or between databases
- Can be used as mechanism to extract information to extract from either local or remote databases
- Works with two basic database types
  - Call-level interfaces (CLI)
    - Common APIs that span several types of databases, providing access to any number of databases through a well defined common interface
      - Open Database Connectivity (ODBC)
  - Native database middleware



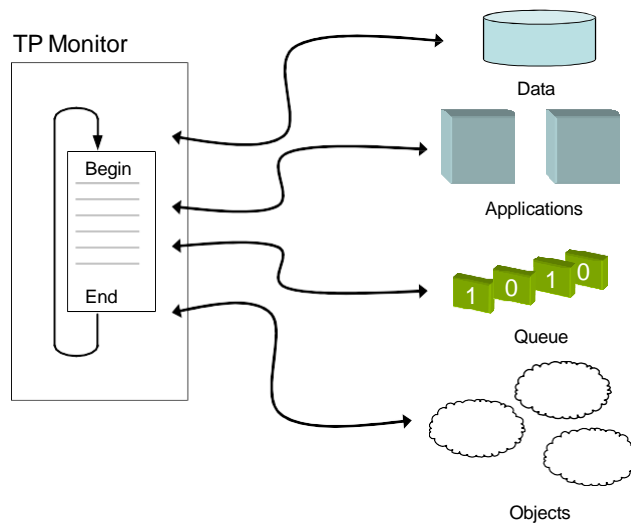
## Transaction-Oriented Middleware

- Provides mechanism for coordination information movement and method sharing between many different resources
  - Provides tightly coupled integration
    - Requires changes with source and target applications
- Based on concept of transaction
  - A unit of work with a beginning and an end
  - Application logic is encapsulated within a transaction
    - Transaction either completes or is rolled back completely
- Two types of transaction-oriented middleware
  - TP monitors
  - Application servers

## TP Monitors

- Provide mechanism to facilitate the communication between two or more applications as well as a location for application logic
- Provides scalability by sharing and processing transactions among other connected TP monitors
- Provide connectors to databases, other applications and queues
  - These connectors requires some application development in order to communicate with these various resources
- Once connected these resources are integrated into the transaction and leveraged as part of the transaction
  - As a result, can recover, if failure occurs
- Provides two services
  - Guarantee the integrity of transactions
  - Resource management and run-time management services
- TP monitors greatest performance value is in their load-balancing feature
  - Allows them to respond gracefully to a barrage of transactions
  - As demand increases, the transaction manager launches more server processes to handle the load even as it kills processes that are no longer required

## TP Monitors



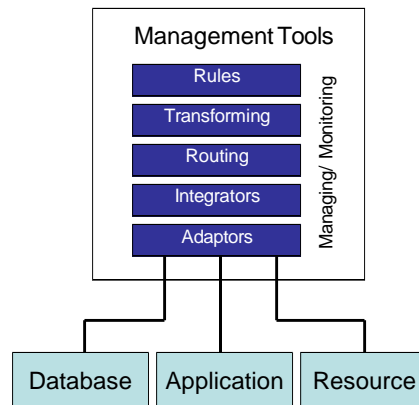
## **Application servers**

- Provide application logic sharing and processing and for connections to back-end resources
  - Resources such as databases, ERP applications and even traditional mainframe applications
- Provide user interface mechanisms
  - To deploy applications to the web platform

## **Integration Servers**

- Facilitates information movement between two or more resources and can account for differences in application semantics and platforms
  - Without any application necessarily understand anything about other applications it shares information with
- Can also join many applications by using common rules and routing engines
  - Can transform the schema and content of the information as it flows between various applications and databases
- Can broker messages between two or more source or target systems

## Integration Servers



## Tough Choices

- RPCs are slow, but their blocking nature provides best data integrity control
  - Updates are always applied in the correct order
- Asynchronous layer to access data may seem to be the best solution, as it does not block processing
  - But there is no way to guarantee that an update will occur in a timely manner
- Messaging could provide better performance because the queue manager offers sophisticated performance-enhancing features as such as load balancing
- Presence of easy-to-use interface will take the power of middleware and place it in hands of the business user