

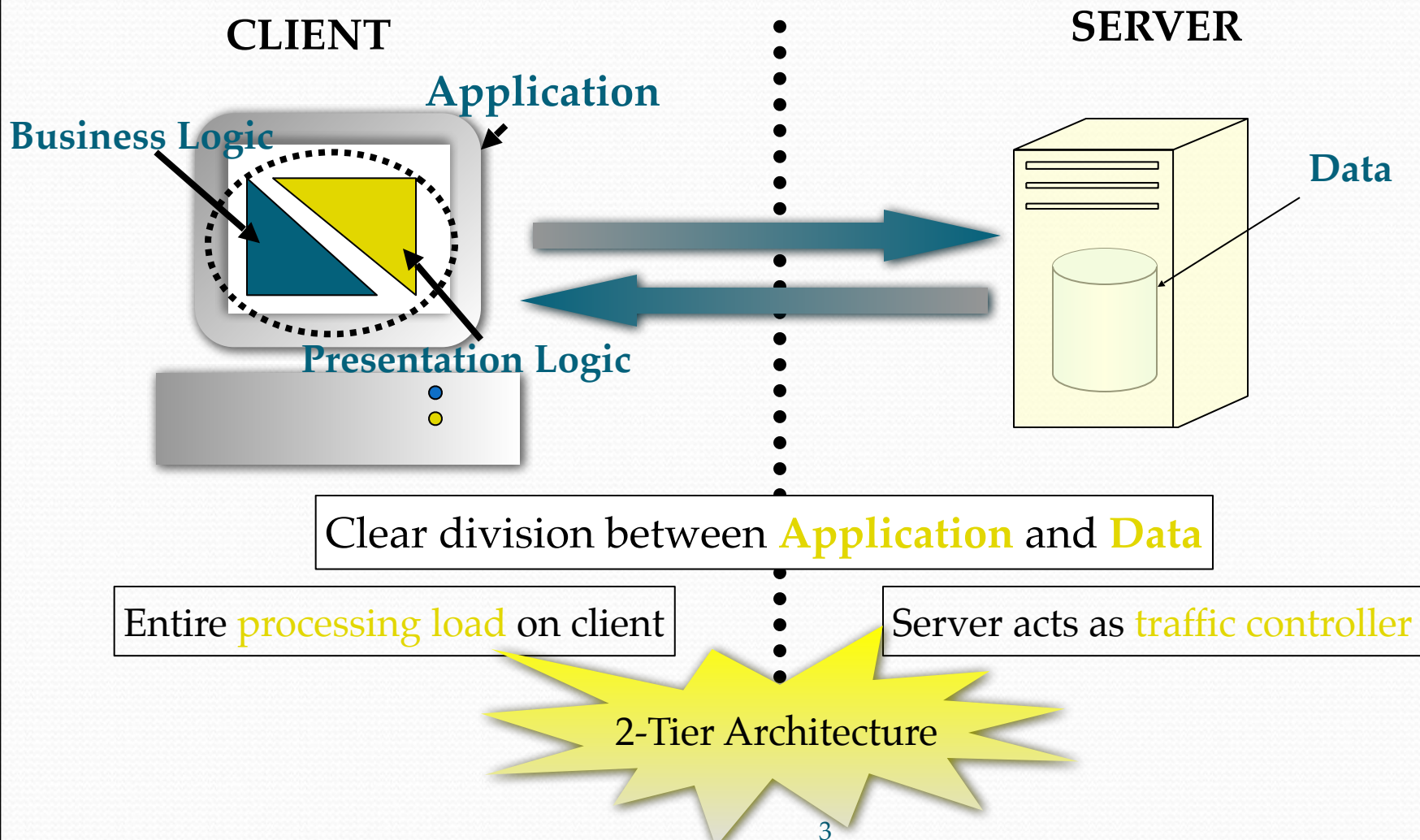
# Distributed Multi-tiered Applications

- The J2EE platform uses a multi-tiered distributed application model for both enterprise applications
- Application logic is divided into “components” according to function, and the various application components that make up a J2EE application are installed on different machines depending on the tier in the multi-tiered J2EE environment to which the application component belongs

- The J2EE technologies can be broadly classified into four different categories:
  - Client-side technologies
  - Component technologies
  - Service technologies
  - Communication technologies
- Component technologies include:
  - Servlets
  - Java Server Pages
  - Enterprise JavaBeans
    - Session Beans
    - Entity Beans
- Service Technologies include:
  - Java Database Connectivity
  - Java Transaction API and Service



# Client-Server Architecture



## Client-Server: The Drawbacks

Business logic present on each client

Client waits longer for response

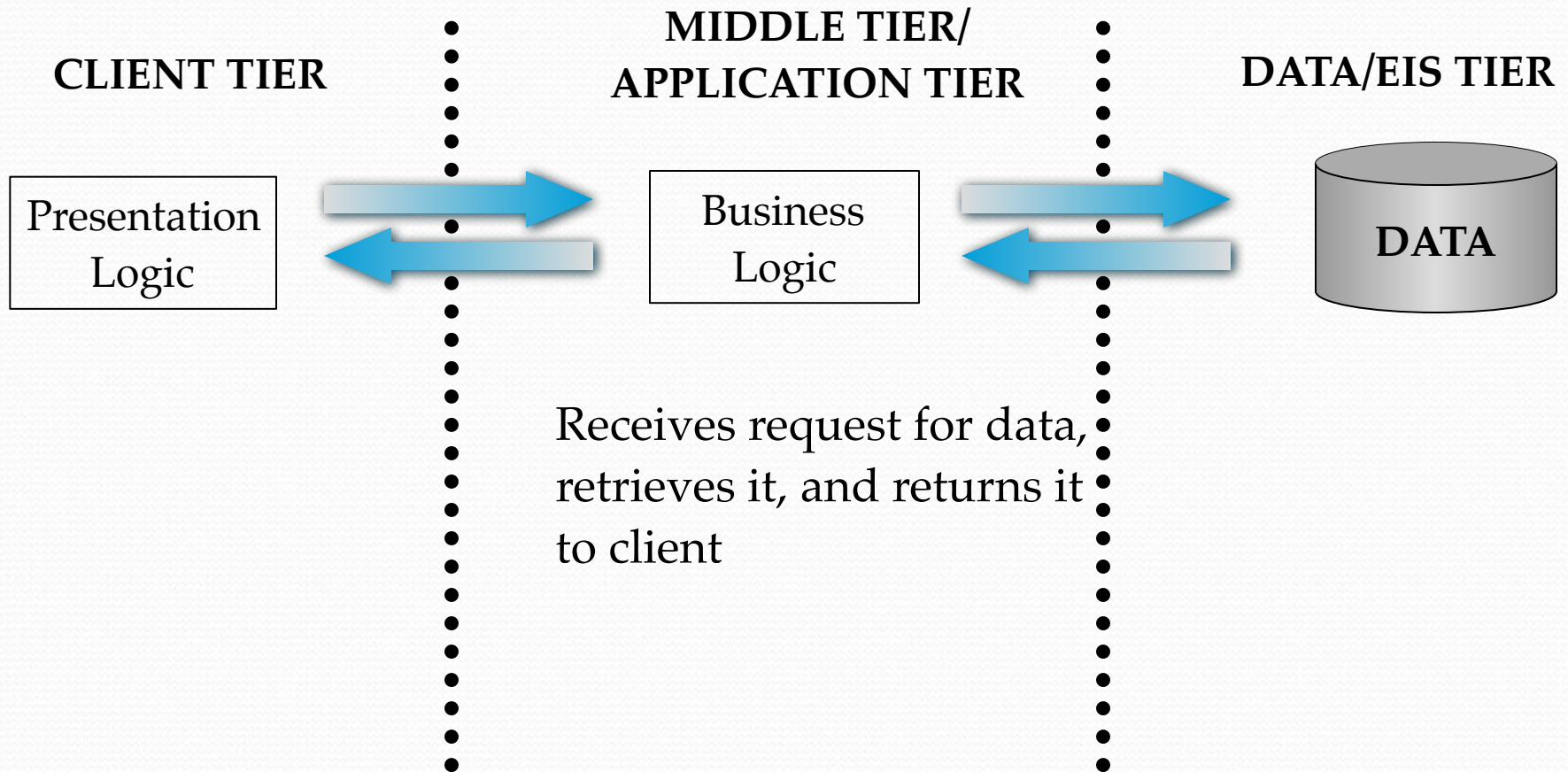
Business logic + presentation logic bundled together—therefore scalability problems

Load on server and network as all clients send request to 1 server

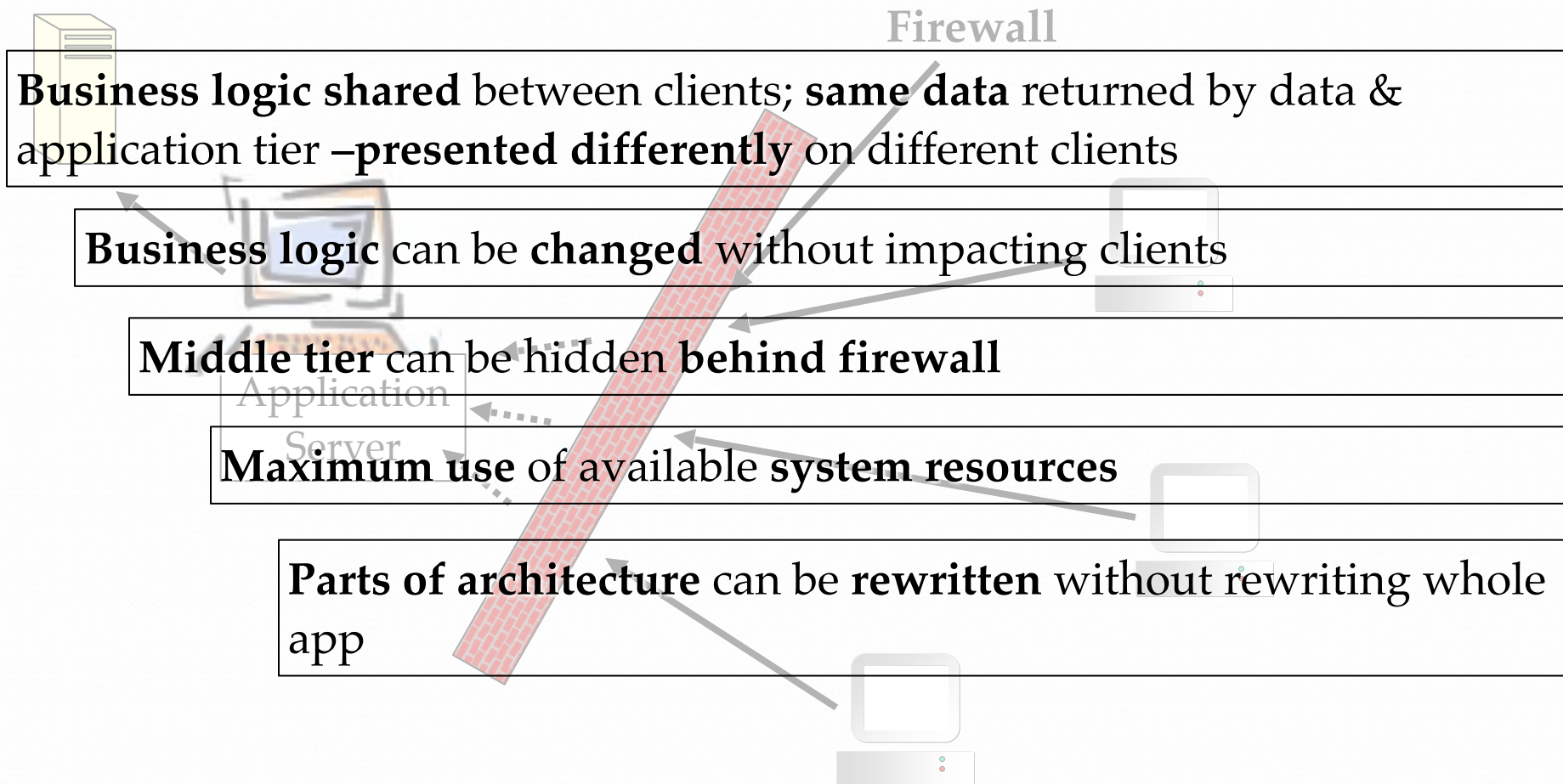
Tiny change to application—entire application has to be changed, and the clients upgraded



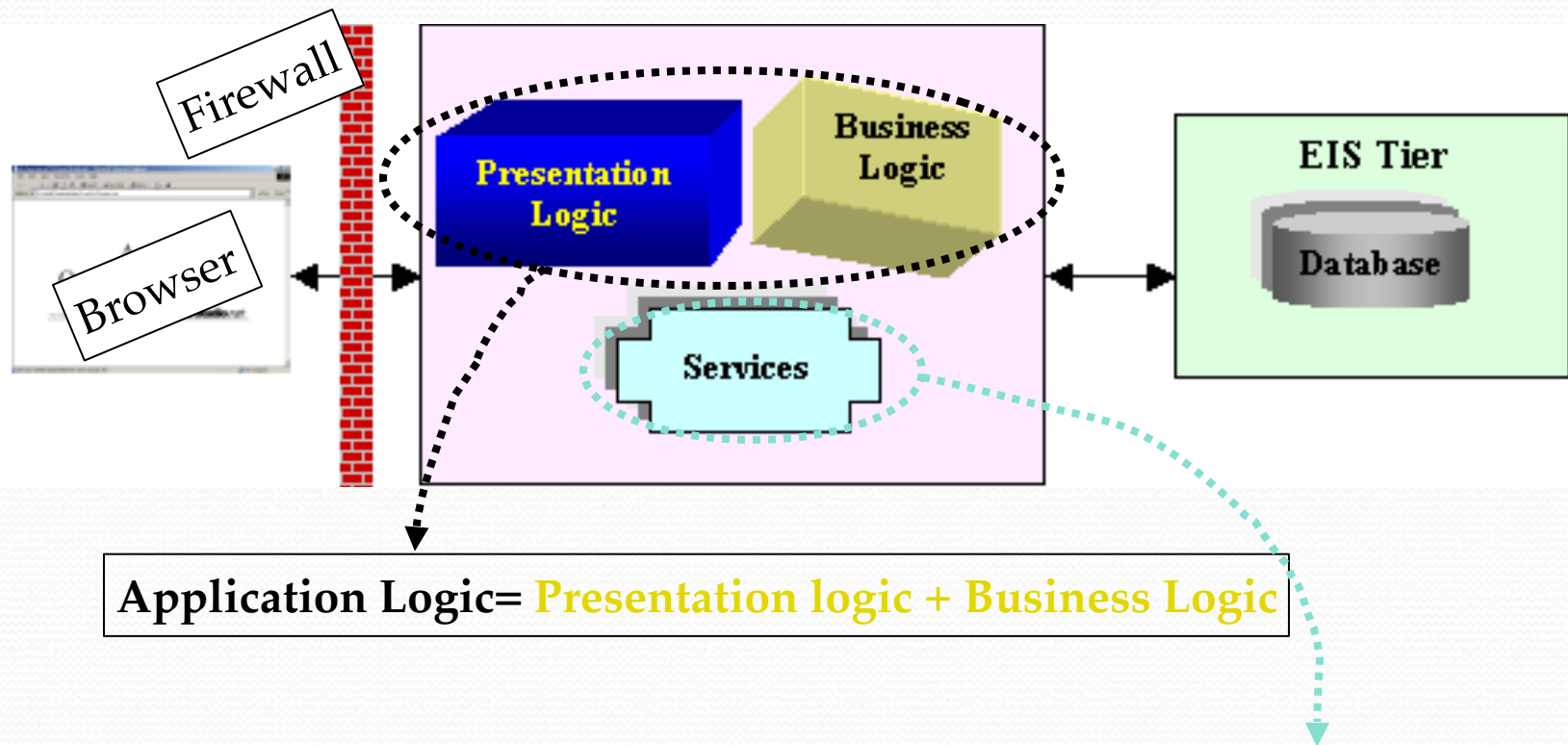
# 3-Tier Architecture



# 3-Tier Architecture: the benefits



# Traditional n-Tier Architecture



Infrastructure services provide additional functionalities required by application, such as messaging services and transactional services.



# Traditional n-Tier Architecture: The Characteristics

Business logic and presentation logic in same module

Database connectivity through same module

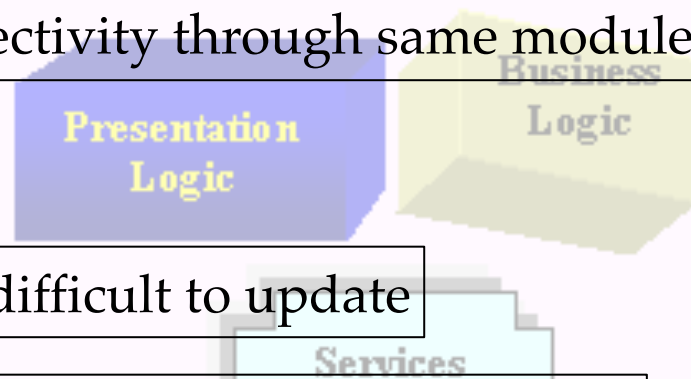
Scalability low

Business logic difficult to update

Client & Server – stateless communication

Business logic unaware of different client identities

Client has to maintain state





# Overcoming the drawbacks - Improving the system

**PROBLEM**

Middle Tier contains one  
App object

For different types of  
needs-different app objects  
required

Different application  
objects may not be able to  
communicate with each  
other

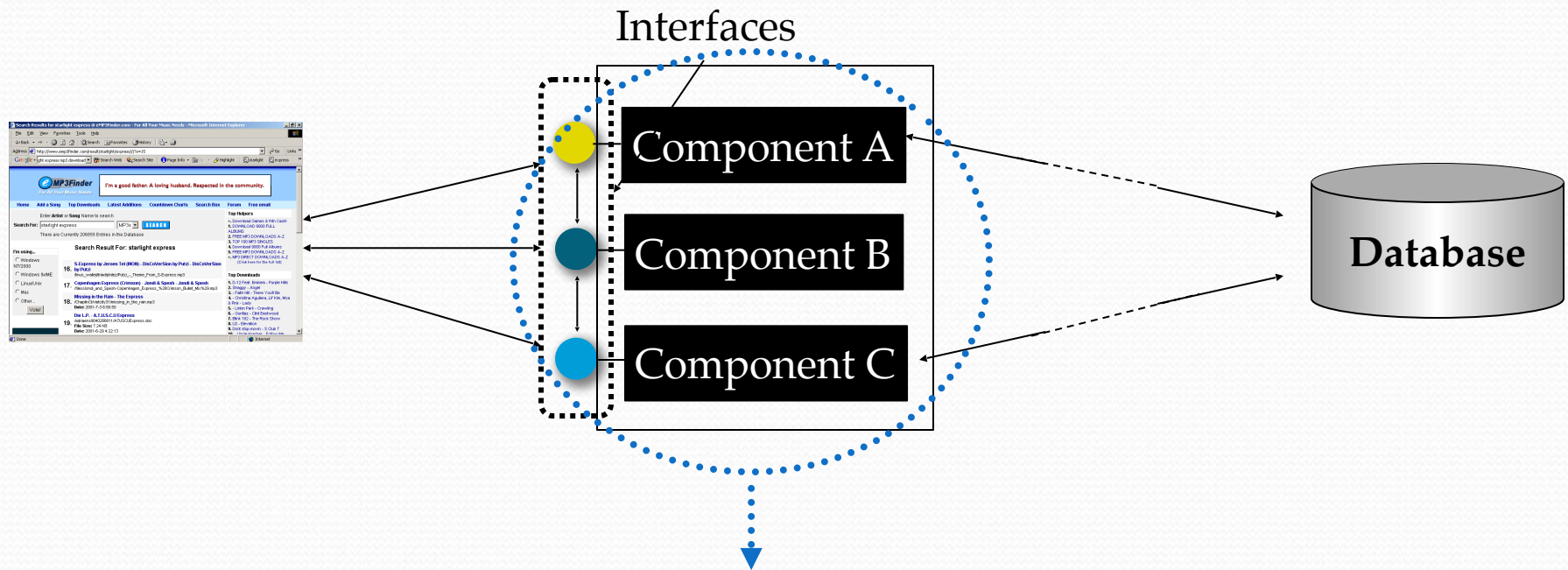
**SOLUTION**

Extend the middle tier to  
create one more layer

Allow multiple application  
objects to reside on the server

Use interfaces to  
communicate between  
application objects

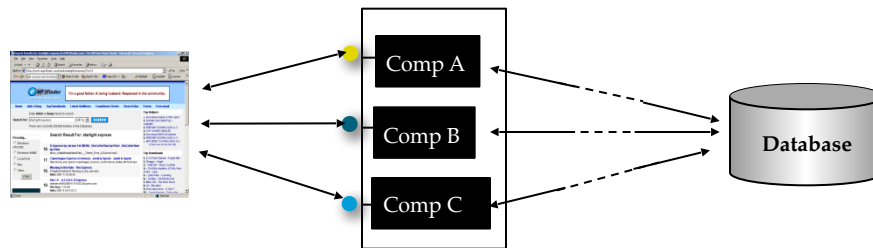
# Component n-tier Architecture



Application object broken into components that can communicate with each other, through interfaces



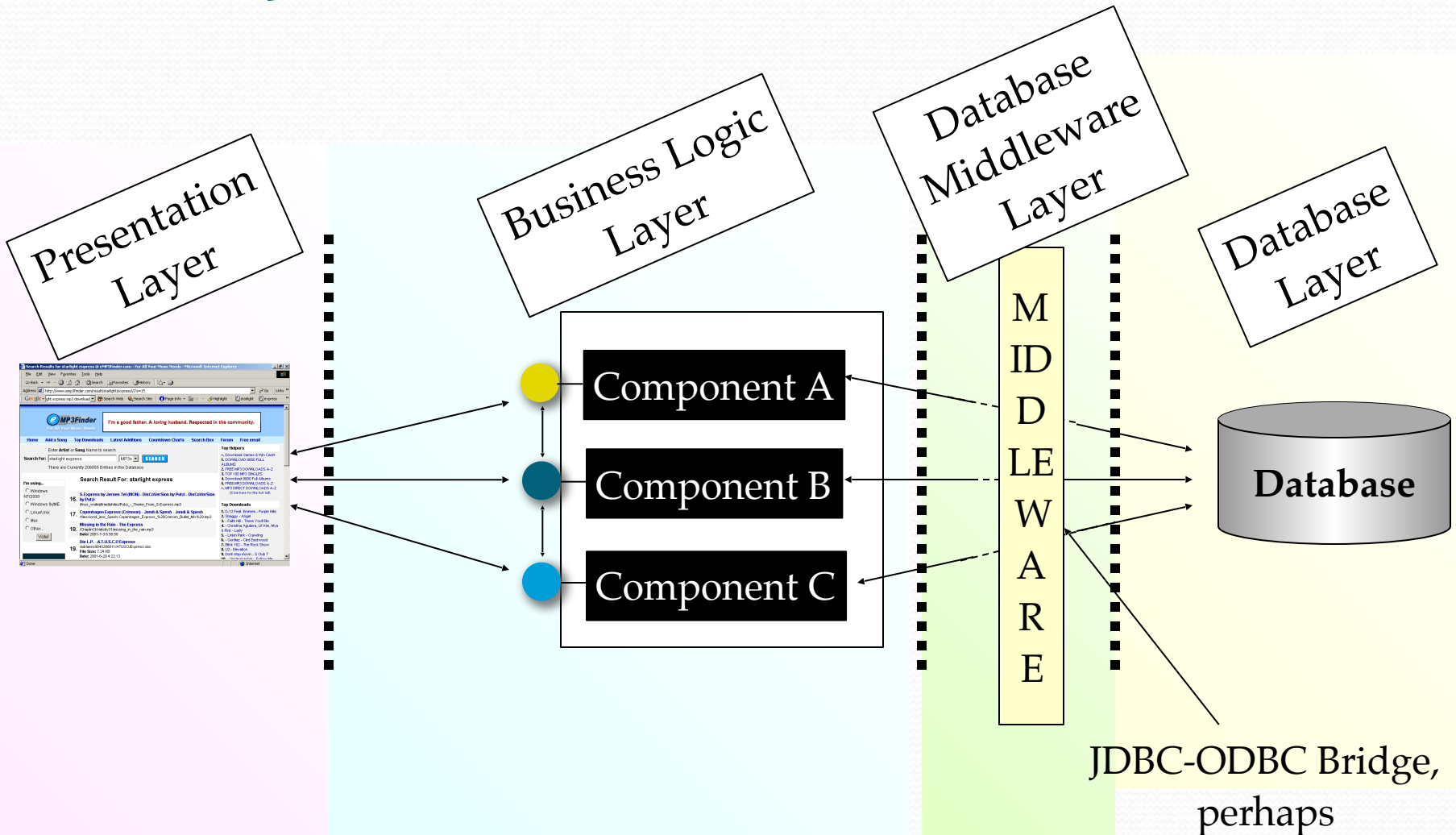
# Component Based n-Tier Systems



- Component objects maintain identity and encapsulate remote methods
- Components can be designed to maintain session state on server
- Business logic can be modified without affecting other logic



# Layered Architecture



# Various models of architecture

Client-Server

Traditional n-Tier

Component-based n-Tier

Layered

Which **architecture** would suit which scenario??

Depends on

**Distributed nature of application**

**Scalability**

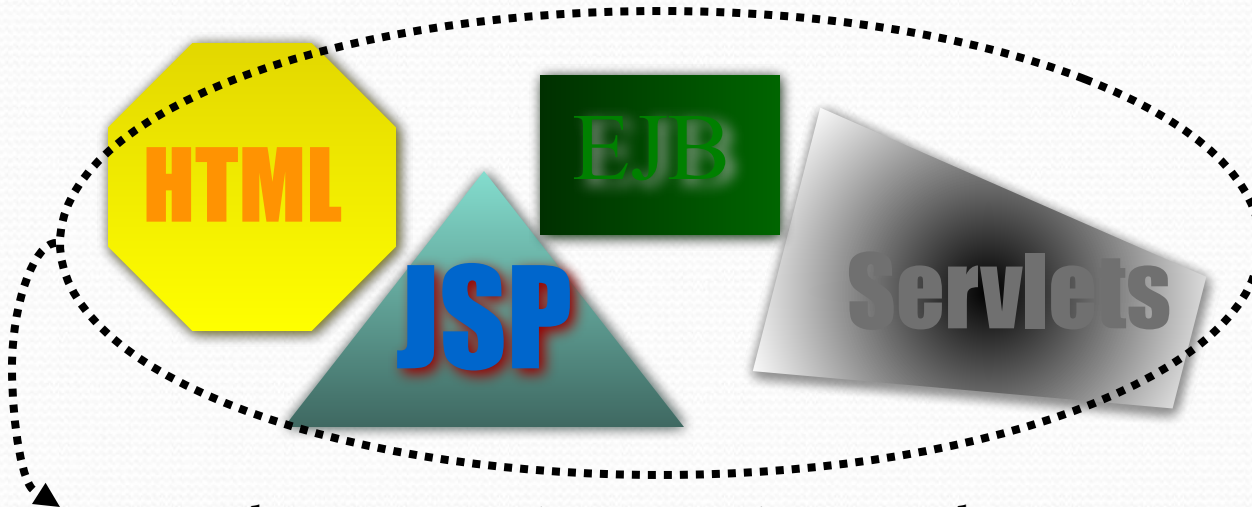
**Performance**

**Memory Management**

# J2EE Architecture

J2EE is a layered architecture

J2EE framework designed based on...



Using these we can design applications that are...

Flexible

Scalable

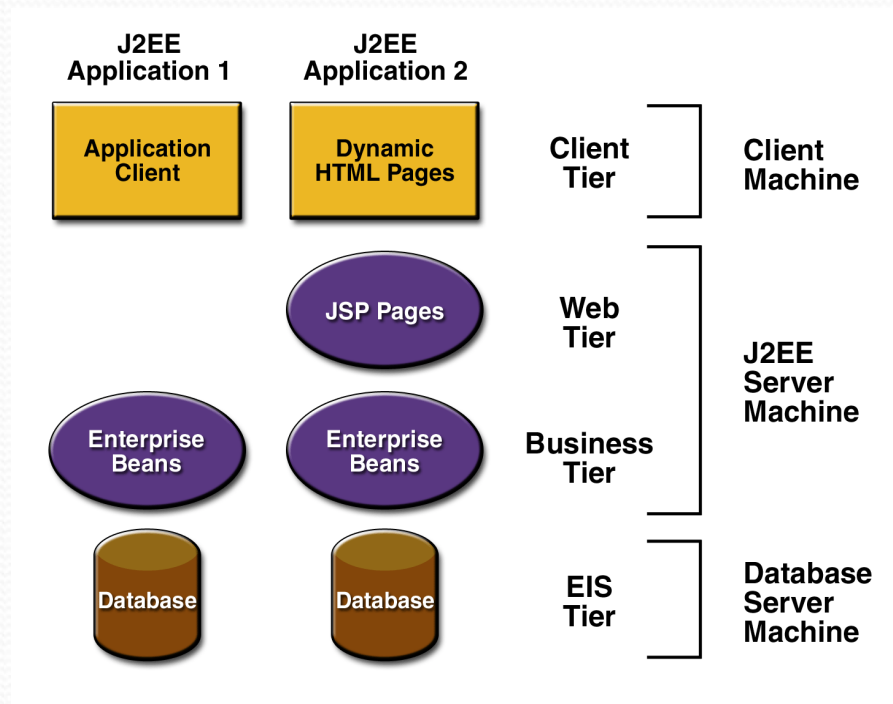
Distributed

Multi-tier

Component-based



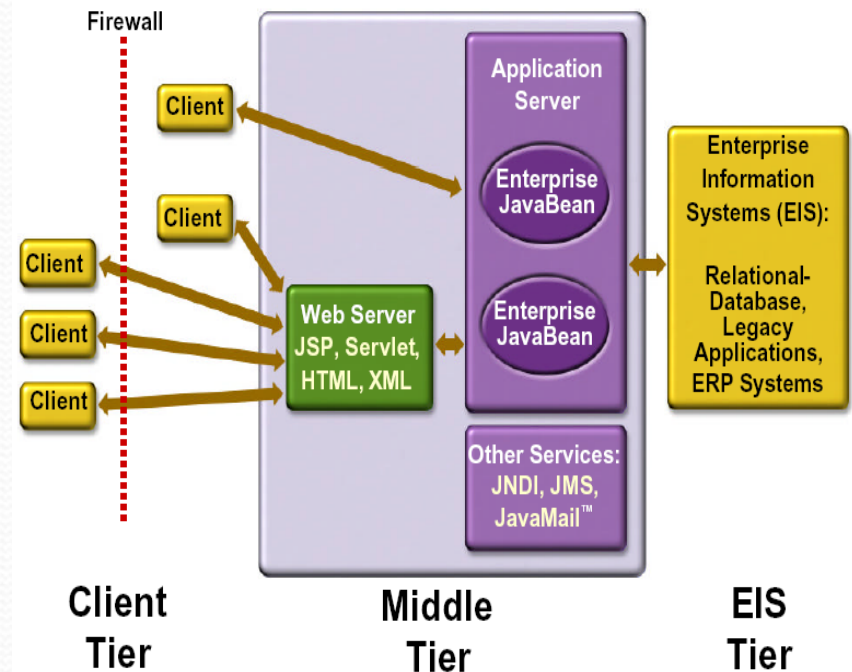
# J2EE Architecture



- J2EE multi-tiered applications are generally considered to be three-tiered applications because they are distributed over three different locations
  - client machines
  - the J2EE server machine
  - the database or legacy machines at the back end

# J2EE Architecture

- Three-tiered applications that run in this way extend the standard two-tiered client and server model by placing a multithreaded application server between the client application and back-end storage





# J2EE goals

- Robustness
- Scalability
- Simplicity
- Maintainability
- Testability
- Reusability



# J2EE Containers

- The application server maintains control and provides services through an interface or framework known as a *container*
- There are five defined container types in the J2EE specification

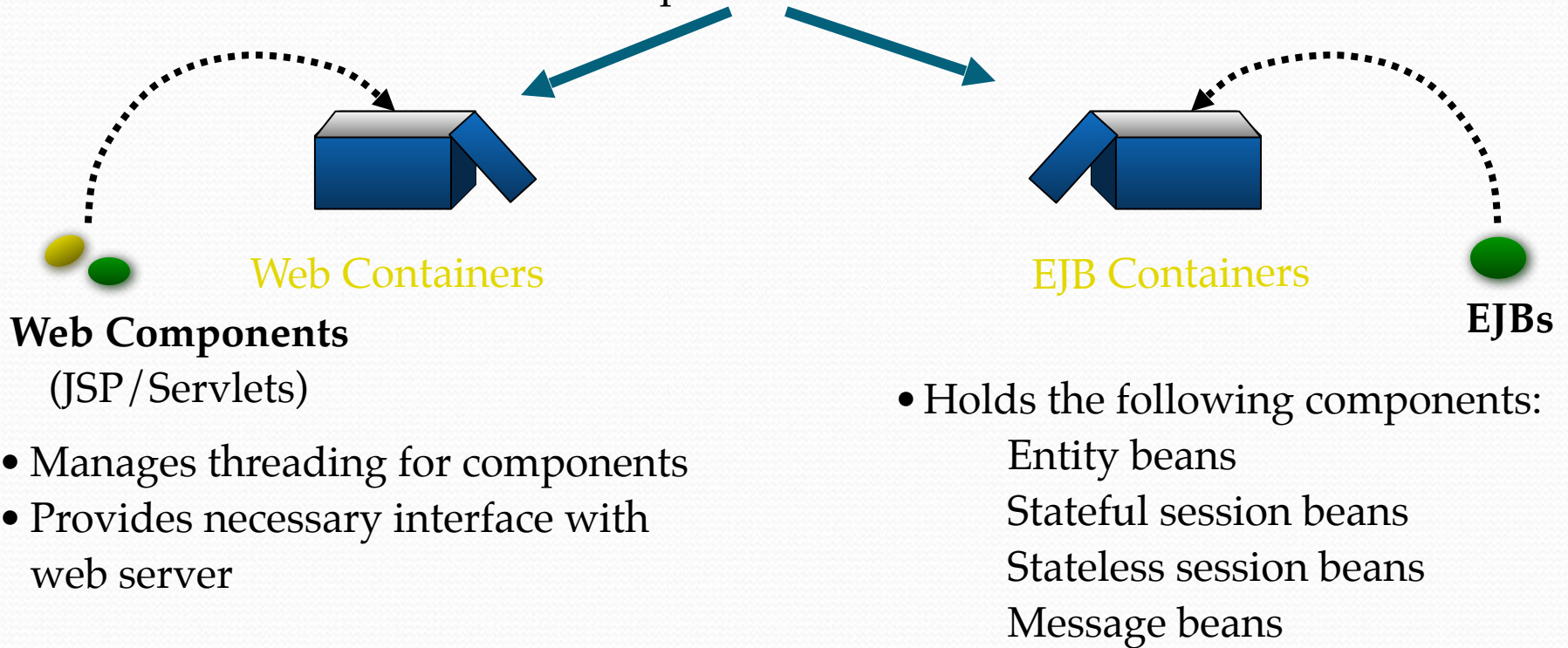
# J2EE Containers

- Three of these are server-side containers:
  - The server itself, which provides the J2EE runtime environment and the other two containers
  - An EJB container to manage EJB components
  - A Web container to manage servlets and JSP pages
- The other two container types are client-side:
  - An application container for stand-alone GUIs, console
  - An applet container, meaning a browser, usually with the Java Plug-in



# J2EE Container

Two most important containers:



Containers provide medium for services to communicate with domain layer



# J2EE Components

- As said earlier, J2EE applications are made up of components
- A *J2EE component* is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and that communicates with other components

# Components

- Client components run on the client machine, which correlate to the client containers
- Web components -servlets and JSP pages
- EJB Components



# Packaging Applications and Components

- Under J2EE, applications and components reside in Java Archive (JAR) files
- These JARs are named with different extensions to denote their purpose, and the terminology is important

# Various File types

- Enterprise Archive (EAR) files represent the application, and contain all other server-side component archives that comprise the application
- Client interface files and EJB components reside in JAR files
- Web components reside in Web Archive (WAR) files



# Deployment Descriptors

- Deployment descriptors are included in the JARs, along with component-related resources
- Deployment descriptors are XML documents that describe configuration and other deployment settings (remember that the J2EE application server controls many functional aspects of the services it provides)
- The statements in the deployment descriptor are declarative instructions to the J2EE container; for example, transactional settings are defined in the deployment descriptor and implemented by the J2EE container

# Deployment Descriptors

- Most J2EE Web Services vendors provide a GUI tool for generating deployment descriptors and performing deployment because creating manual entries is tedious and error prone
- The deployment descriptor for an EJB component must be named `ejb-jar.xml`, and it resides in the `META-INF` directory inside the EJB JAR file



# EJB Components

- EJB components are server-side, modular, and reusable, comprising specific units of functionality
- They are similar to the Java classes we create every day, but are subject to special restrictions and must provide specific interfaces for container and client use and access
- We should consider using EJB components for applications that require scalability, transactional processing, or availability to multiple client types

# EJB Components- Major Types

- **Session beans**

- These may be either *stateful* or *stateless* and are primarily used to encapsulate business logic, carry out tasks on behalf of a client, and act as controllers or managers for other beans

- **Entity beans**

- Entity beans represent persistent objects or business concepts that exist beyond a specific application's lifetime; they are typically stored in a relational database