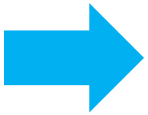
M1 L1
Introduction to Computer and Software Architectures

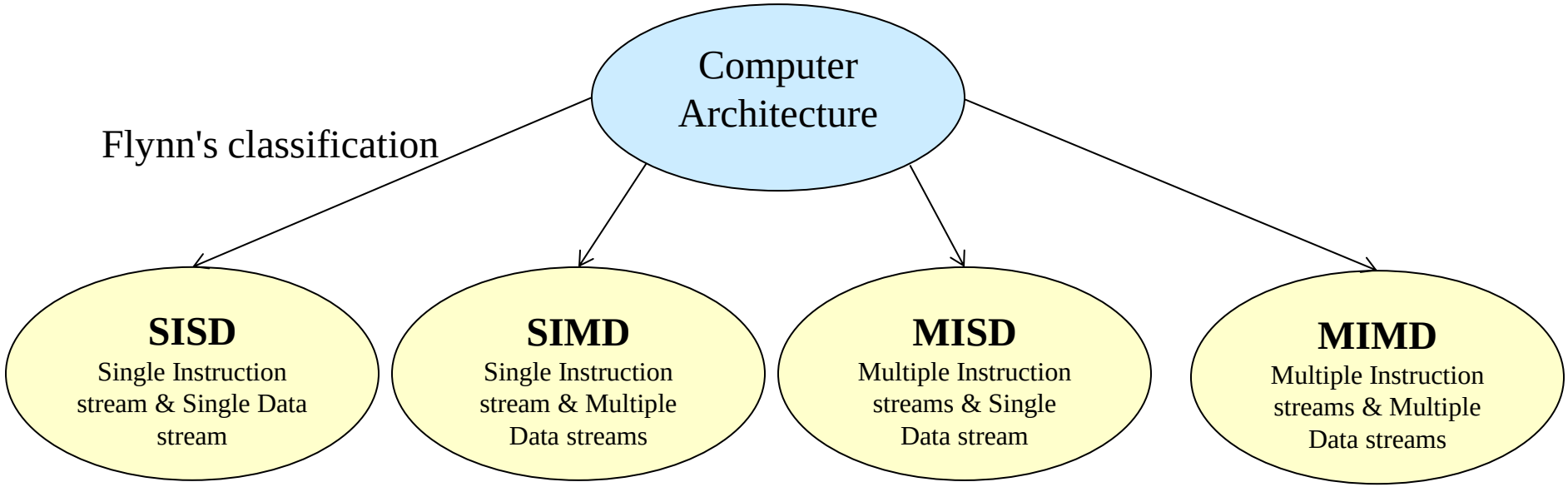
Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Overview



- Computer Architectures
- Software Architectures
- Distributed Software Architectures
- Tiered (N-Tier) Architectures
- Fallacies of Distributed Computing

Computer Architecture



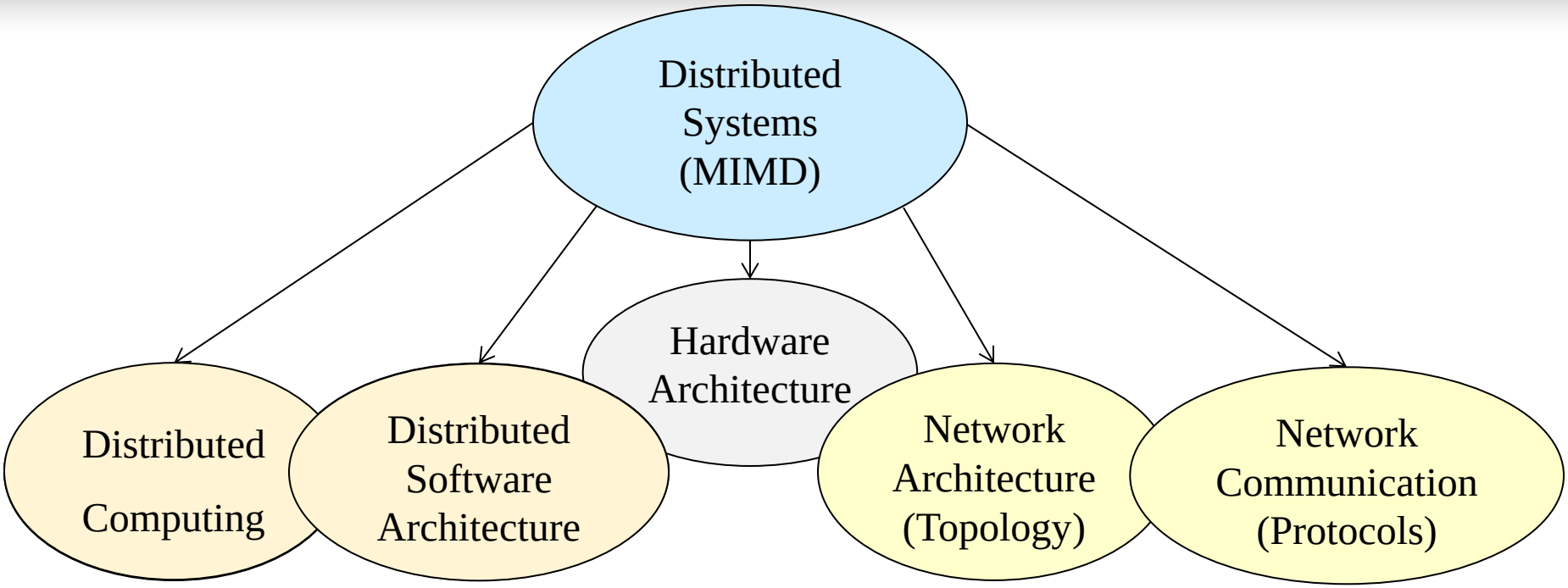
Simple computers

Vector or array computers

Fault-tolerant computer systems performing redundant computing on the same data stream and voting on the results

Computers with multi-core processors
Distributed systems
Networks

Topics in Distributed Systems (MIMD)



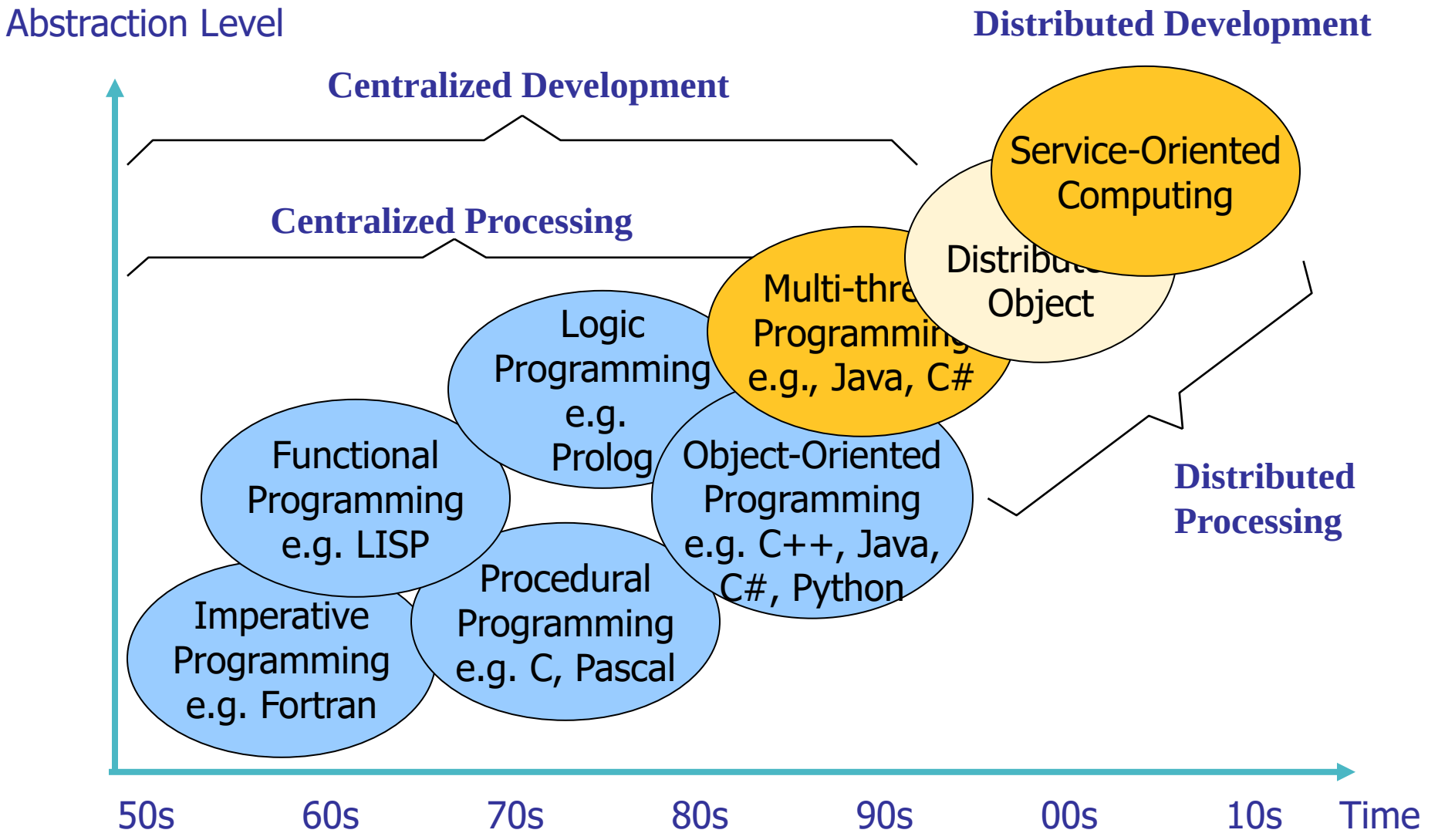
Paradigm of Parallelism and distribution of computing

Client server
Tiered
Object orientation
Service orientation

Topology and connectivity of nodes. Star, ring, peer-to-peer, shared links

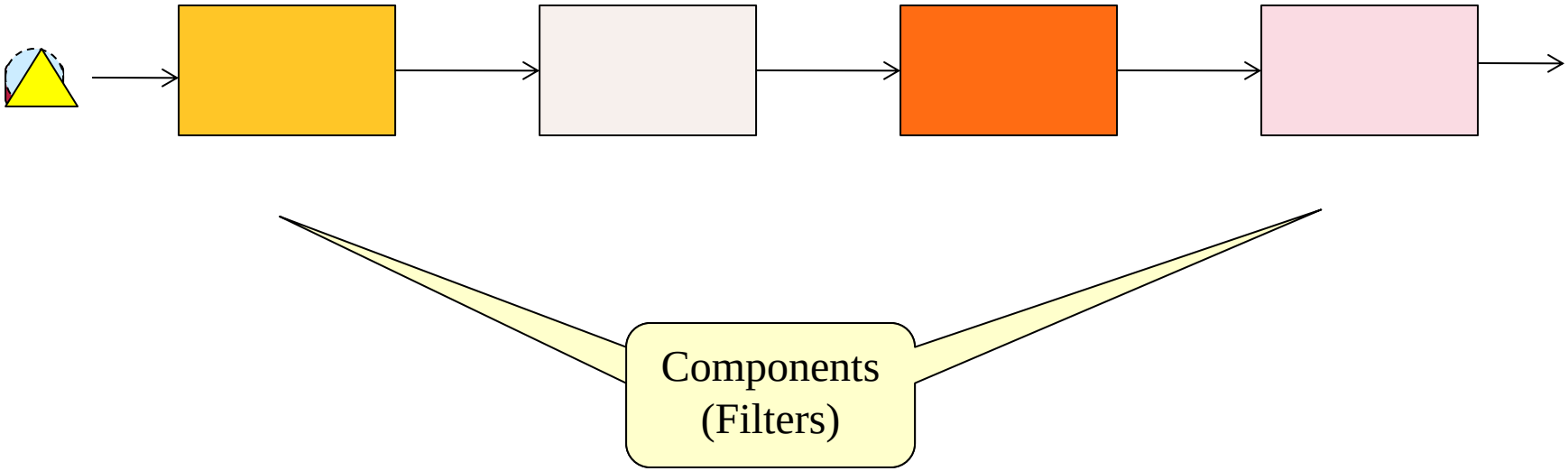
Layers of protocols,
Signal format
Packet format,
Error correction
Routing
Flow control
Session
Presentation

Review: Paradigms of Computing



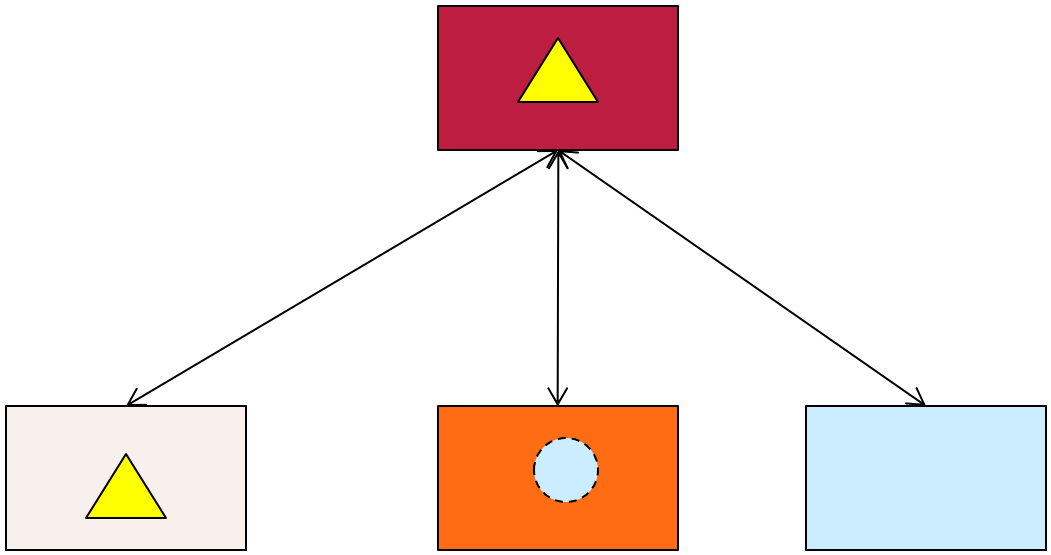
Software Architectures

Pipeline Architecture (Pipe-and-Filter) Architecture: Messages are sent to its neighbor components, processed (filtered), and the result sent to the next component.



Software Architectures

Blackboard Architecture: A component is assigned to be the center for communication. All components communicate through the central component.



Software Architectures

Event-Driven Architecture

- The execution flows are determined (triggered) by the occurrences of events in parallel.
- An event-generating component takes subscription of other components;
- When an event occurs, the event-generating component will notify the components subscribed to the event;
- The notified components will call back / process
- Will be discussed in more detail in later modules

Distributed Software Architectures

- Architectural designs allowing software to execute on more than one logical processors;
- Computing/processing is distributed over several computers rather than confined to a single machine;
- Virtually, all large software systems today are distributed;
- Distributed software architecture describes how the application functionality is distributed over a number of logical components and how these components are distributed across processors.
- Choosing the right architecture for an application is essential to achieve the desired quality of service.

Characteristics of Distributed Systems

- Resource sharing and synchronization
- Openness and standardization
- Concurrency and parallelism
- Scalability
- Dependability and Fault tolerance
- Transparency

Issues to be Addressed in Distributed Systems

Compared to a centralized system, issues and challenges to be addressed in distributed systems include:

- Complexity
- Communication and connectivity
- Security and reliability
- Unpredictability and non-deterministic problems

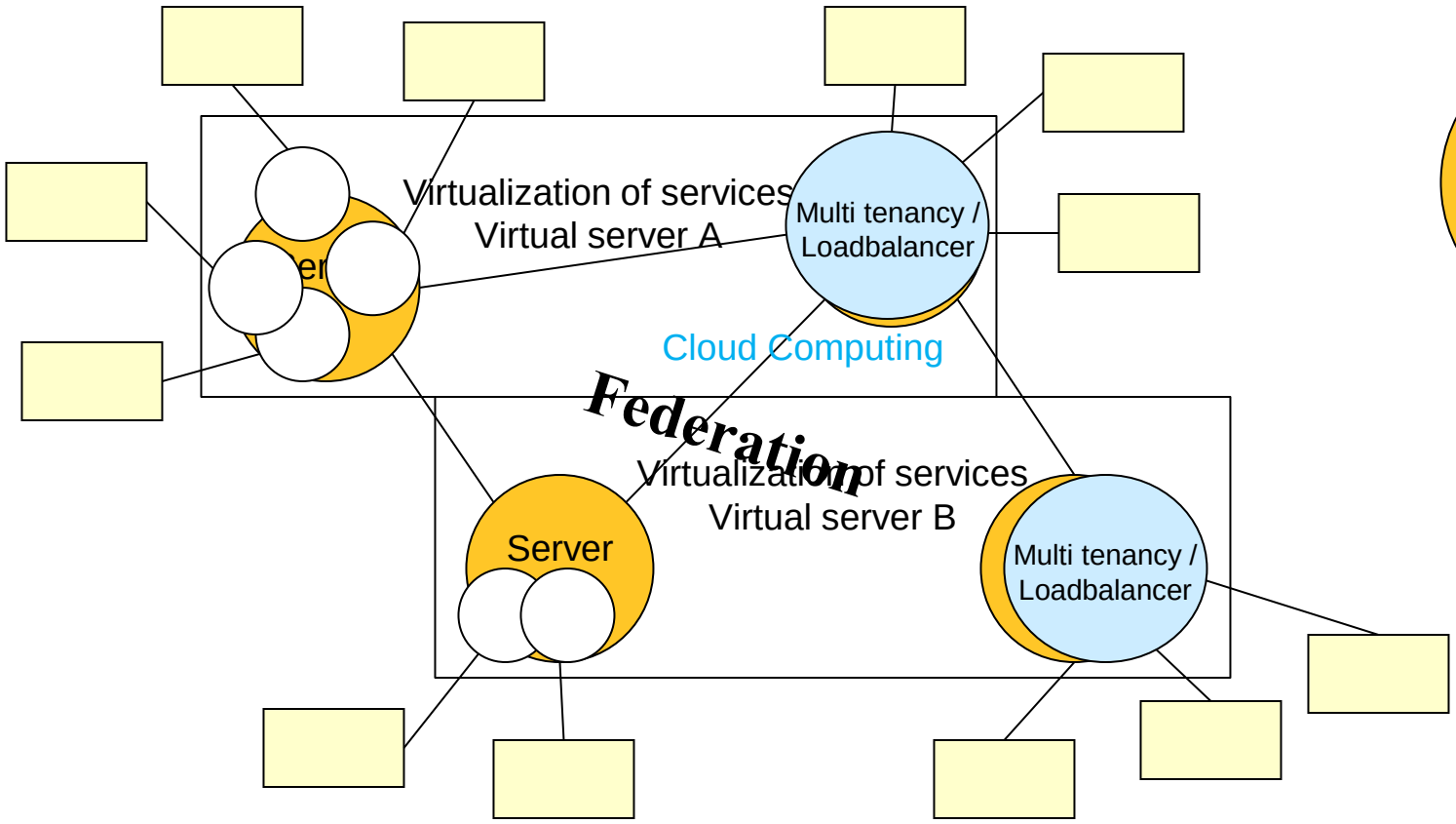
Lecture Overview

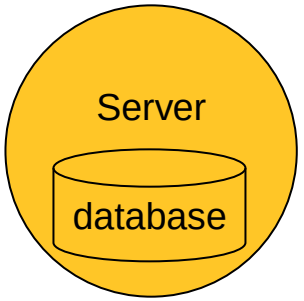
- Computer Architectures
- Software Architectures
- Distributed Software Architectures
- **Tiered (N-Tier) Architectures**
  - Two-tier architecture (client-server)
  - Three-tier architecture
  - Four-tier architecture
- **Fallacies of Distributed Computing**

Client-Server (Two-Tier) Architectures

- The application is modelled as a set of services that are provided by servers and a set of clients that use these services;
- Clients know of servers, but servers need not to know of clients;
- Clients and servers are logical processes:
  They could reside/running on different processors or on the same processor;
- The mapping of processors to processes is not necessarily one to one.

Client-Server Systems and their Federation





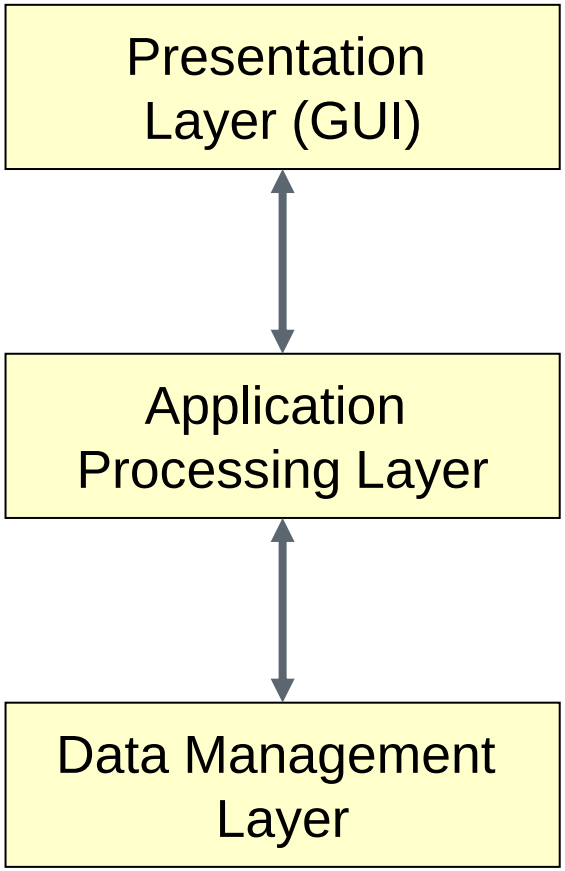
A server also manages the database

Thin-Client and Fat-Client Architectures

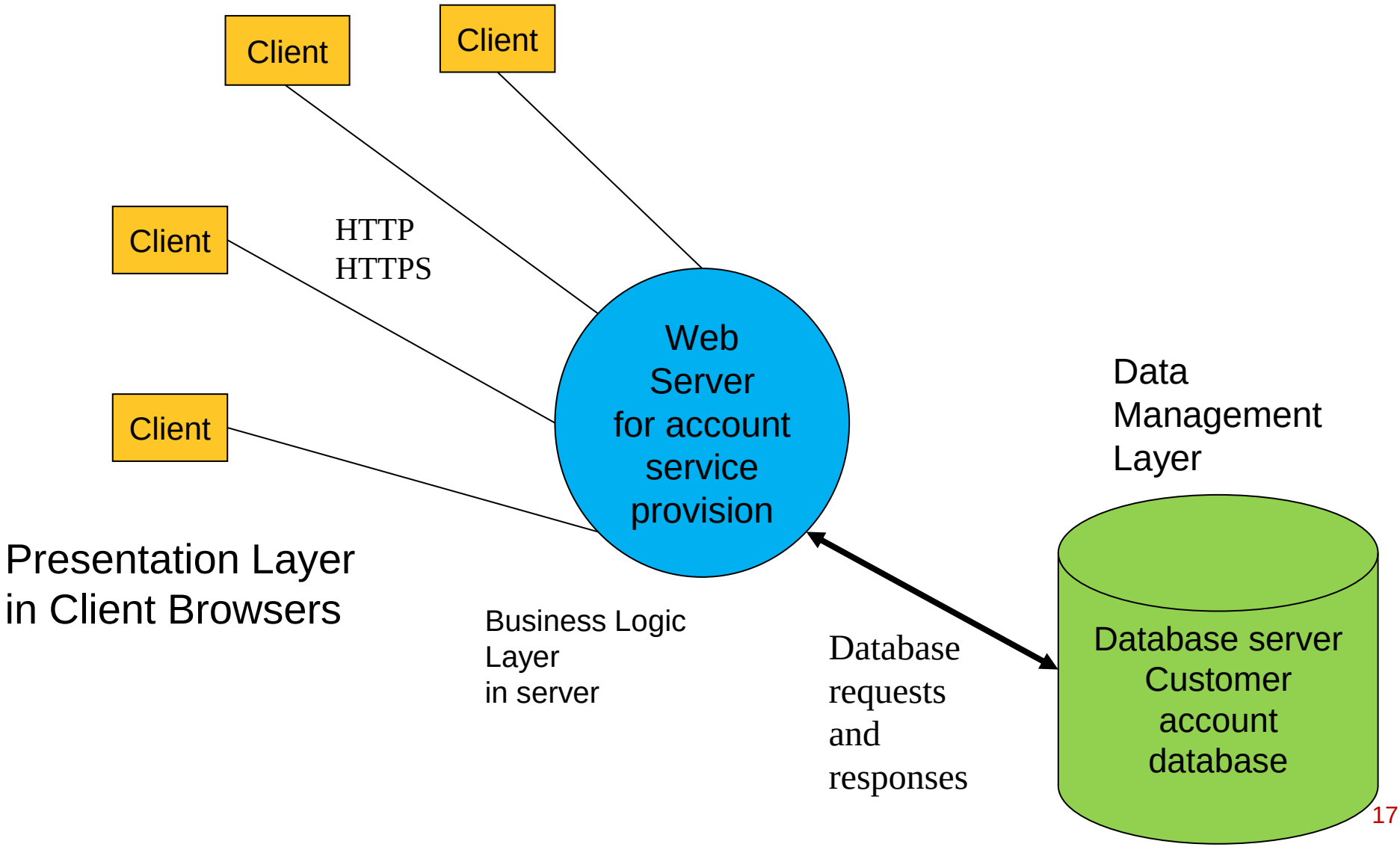
- Thin-client architecture
  - All of the application processing and data management are carried out on the server.
  - The client is simply responsible for running the presentation/GUI software.
- Fat-client (Thick-client) architecture
  - The software on the client implements the application logic and the interactions with the users.
  - The server is responsible for data management (database) only.
  - Example: Many game programs, Adobe Flash and MS Silverlight support Fat Clients, HTML5 can also support Fat Clients, where intensive processing and responsiveness are required.

Traditional Three-Tier Architectures

- Each layer may execute on a separate processor;
- A more balanced approach, which allows for better performance than a thin-client approach and is simpler to manage than a fat-client approach;
- A more scalable architecture - as demands increase, extra servers can be added.

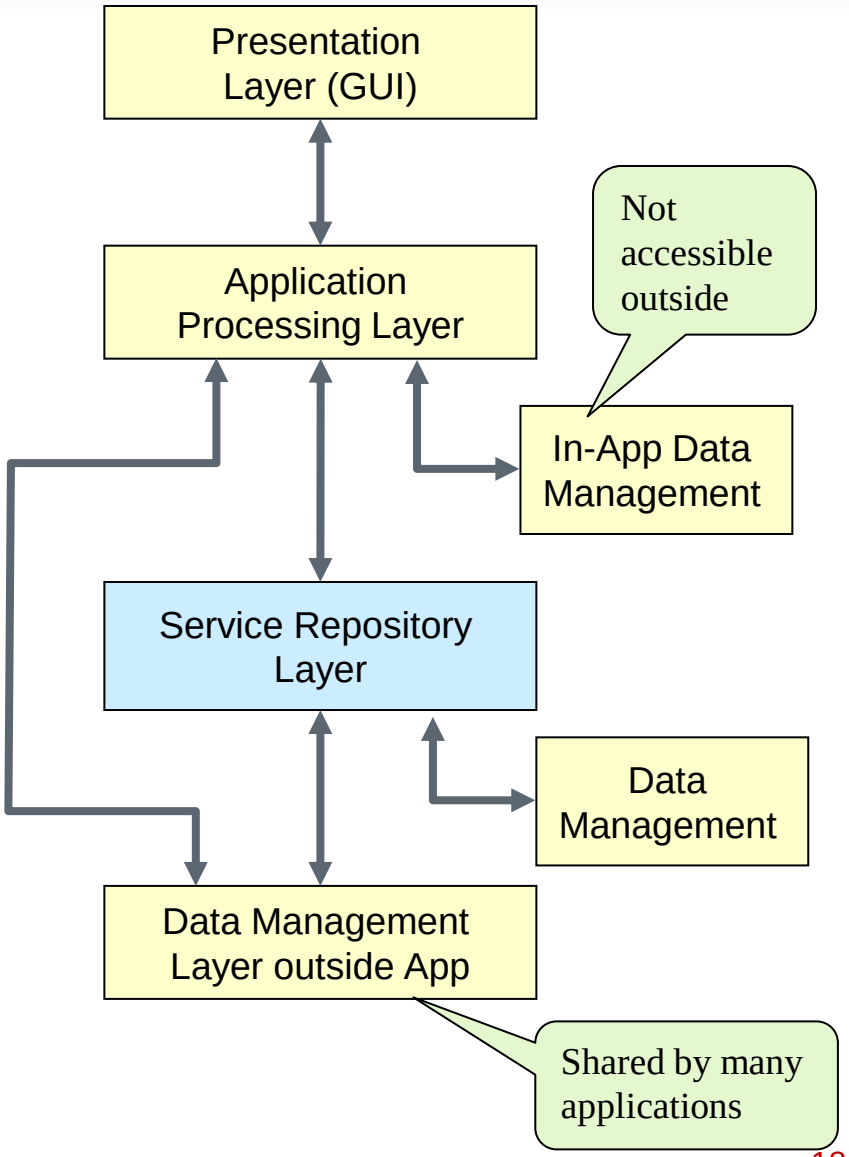


Example: A 3-Tier Internet Banking System

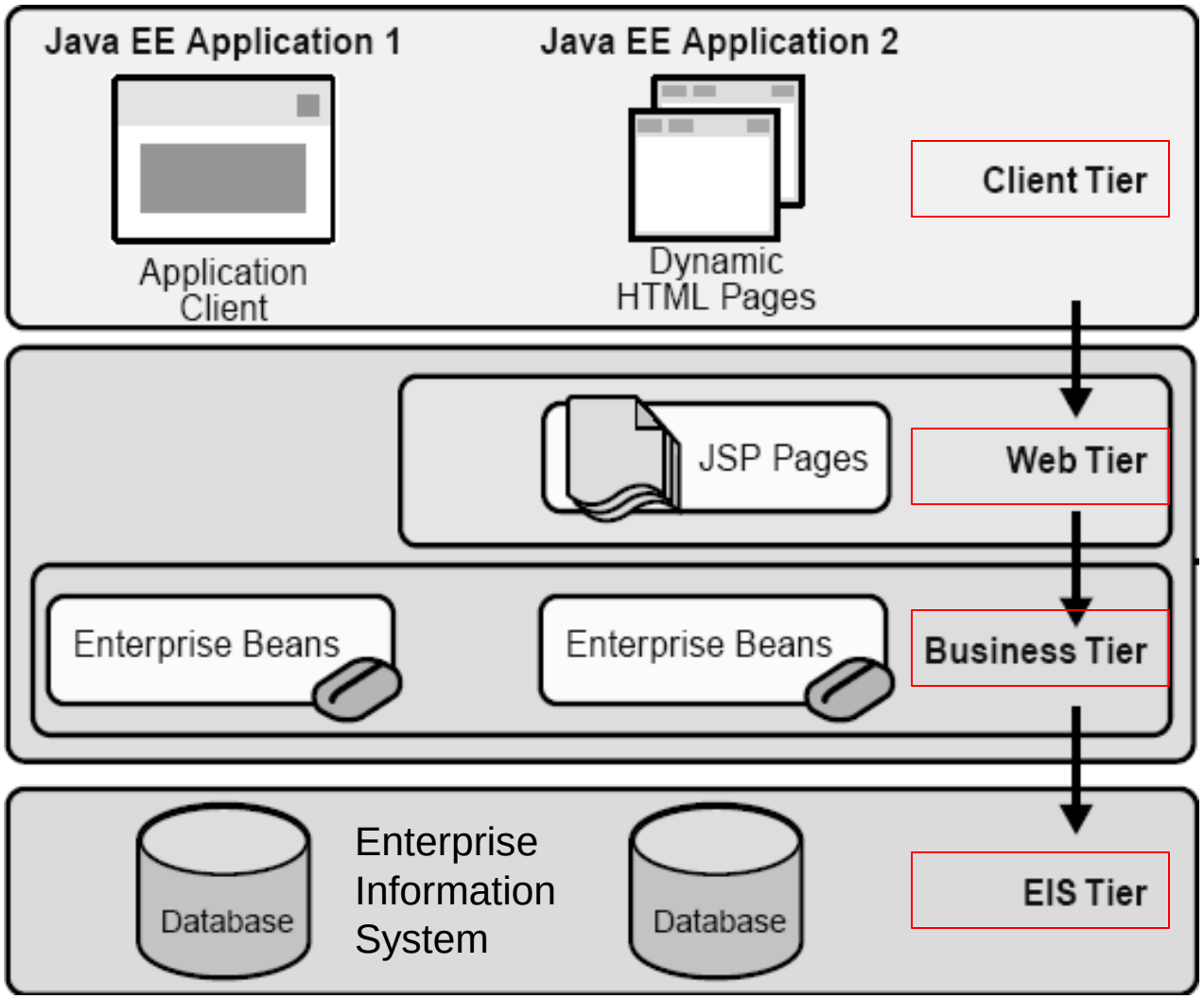


Four-Tier Architecture with a Service Layer

- Each layer may execute on a separate processor;
- Application is largely composed of existing services, running on a different server (repository), provided by external service providers;
- This is an example of Service-Oriented Architecture (SOA);
- SOA does not have to be in N-Tier. It can be a tree or graph too.



Four-Tier Architecture: Java EE Applications



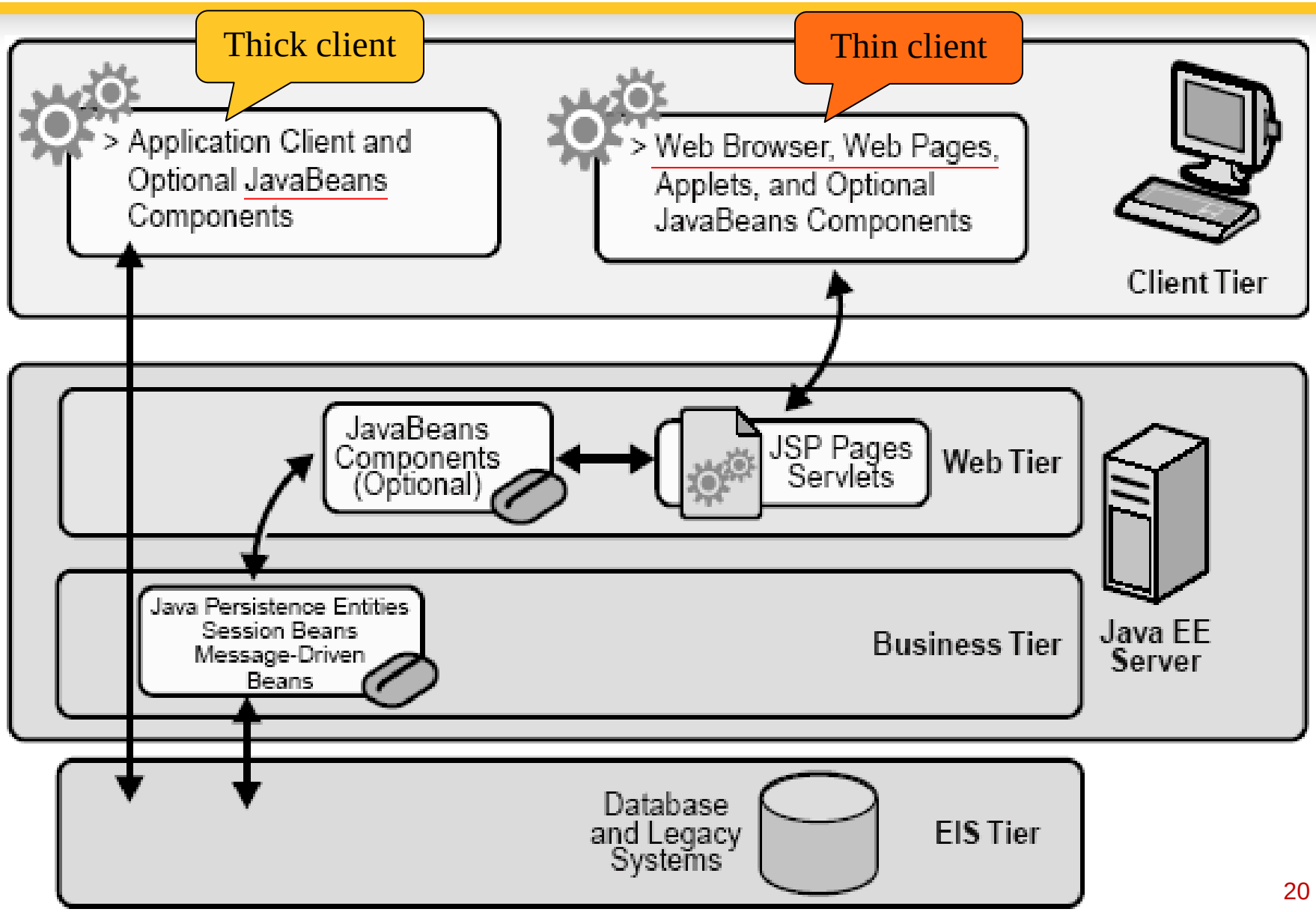
Client Machine

Java EE Server

EIS Database Server

Source: Java Enterprise Edition

Java EE Application (not strictly tiered)



Eight Fallacies of Distributed Computing

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous

Bill Joy and Tom Lyon, Sun Microsystems, 1991

Peter Deutsch, Sun Microsystems, 1994

Seven fallacies

James Gosling, added the 8th


Ira A. Fulton Schools of
Engineering
Arizona State University