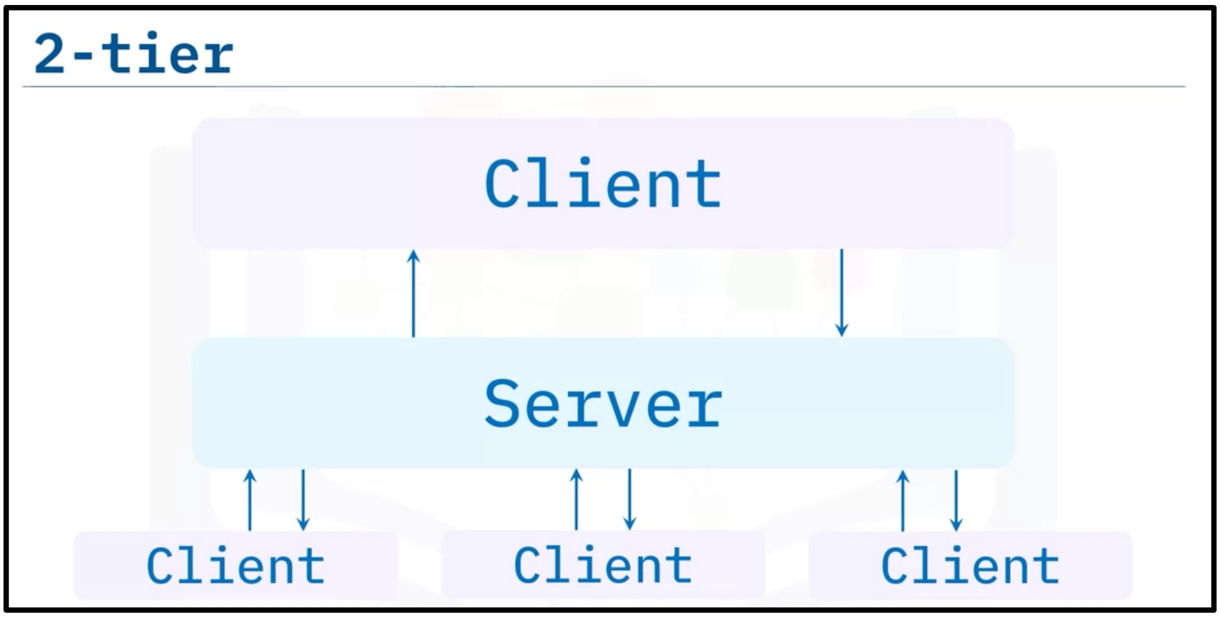
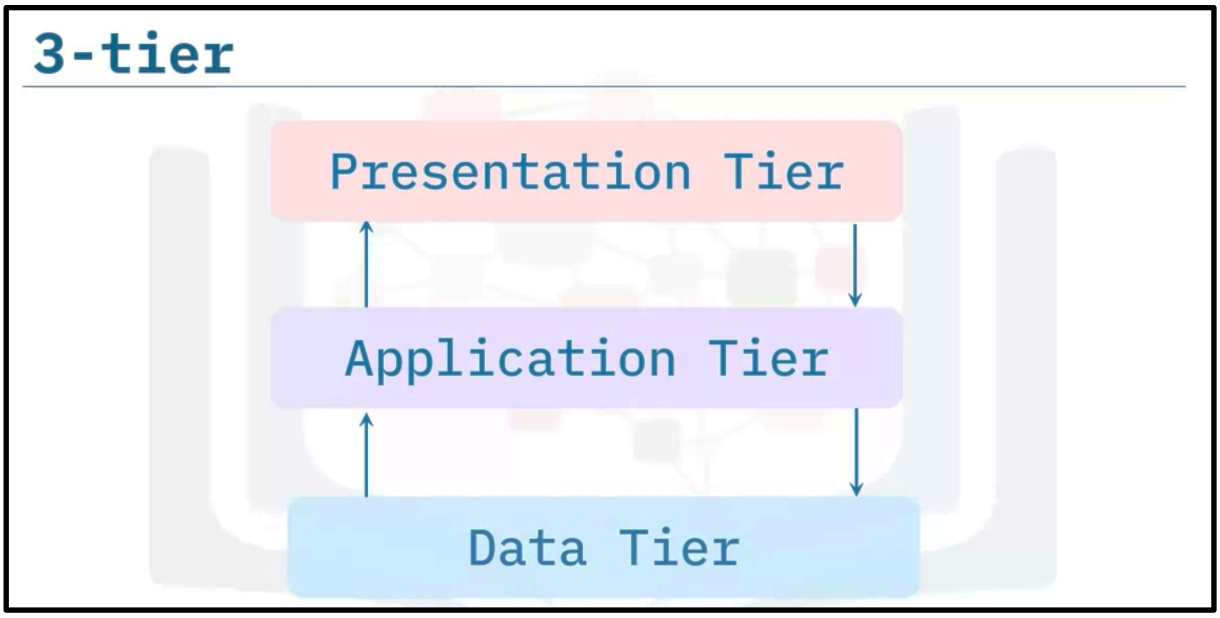
Content Summary:

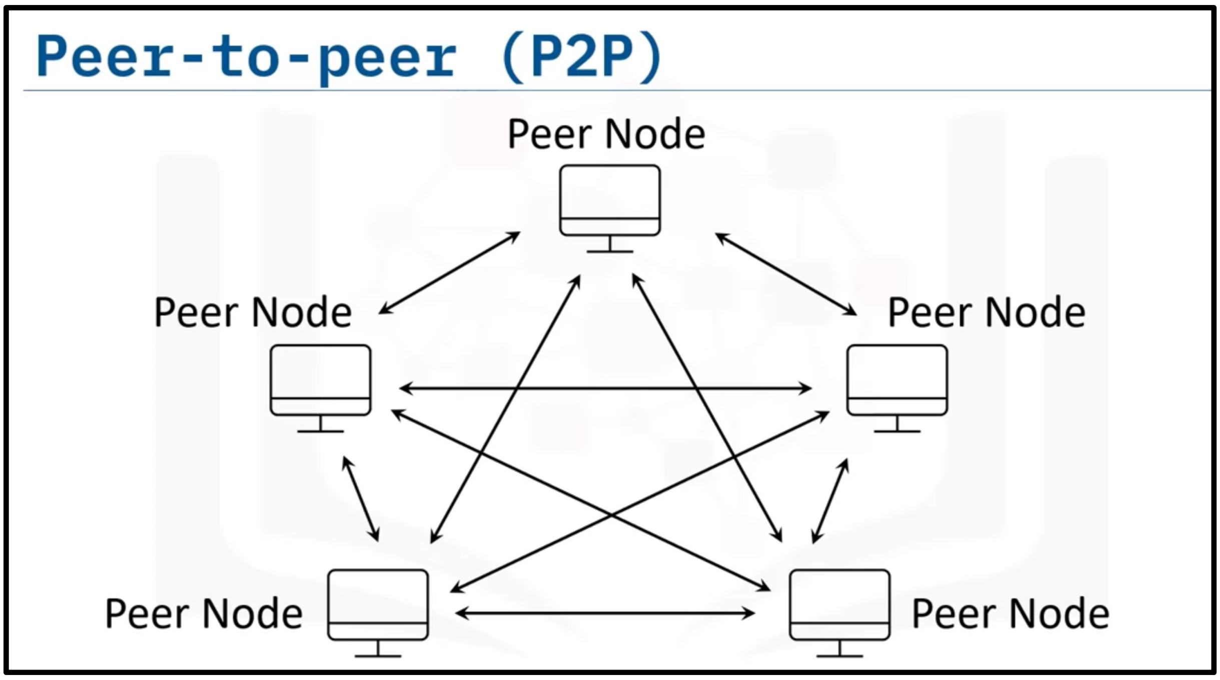
1. **2-Tier Architecture (Client-Server Model):**
   - Structure: Consists of a server that hosts, delivers, and manages most of the resources and services, with the client making requests for these resources.
   - Example: Text messaging applications where the client sends a message request to the server, which then delivers it to the recipient.



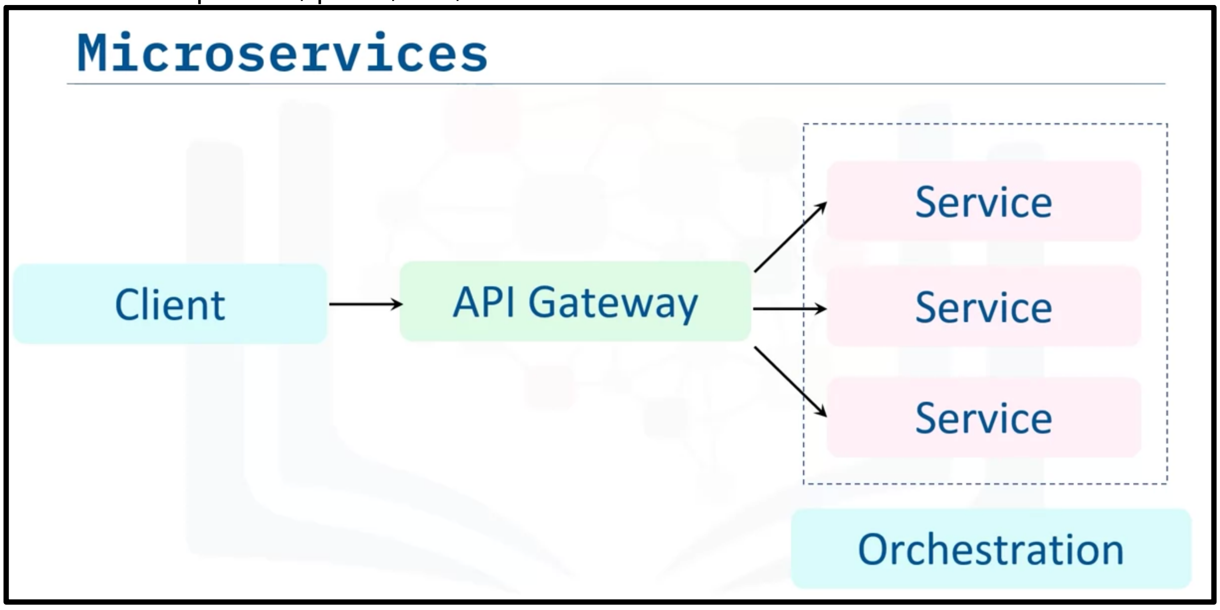
2. **3-Tier Architecture:**
   - Structure: Divided into three logical and physical tiers: the presentation tier (user interface), the application tier (business logic), and the data tier (data storage and management).
   - Example: Many web applications use this model, including e-commerce sites, where the web server manages the user interface, an application server processes the business logic, and a database server handles data storage.



3. **Peer-to-Peer (P2P) Architecture:**
   - Structure: A decentralized network where each node acts as both client and server, sharing resources such as processing power and storage directly with other nodes.
   - Example: Cryptocurrencies like Bitcoin and Ethereum, where each node in the blockchain network acts as both server and client.



4. **Event-Driven Architecture:**
    - Structure: Based on the production and consumption of events, where producers generate events and consumers (or event handlers) respond to them.
    - Example: Ride-sharing applications like Uber or Lyft, where a user's request for a ride triggers a series of events handled by different parts of the system.
5. **Microservices Architecture:**
    - Structure: Breaks down an application into a collection of smaller, interconnected services, each performing a specific function and communicating via APIs.
    - Example: Social media platforms where different services manage user profiles, posts, ads, and friend interactions.



Integration and Flexibility:

- Architectural patterns are often not mutually exclusive and can be combined to meet complex requirements. For instance, a three-tier architecture might be implemented using microservices, or a peer-to-peer system might incorporate event-driven components.
- It is crucial for architects to choose the appropriate patterns based on the specific needs and constraints of the project.

Conclusion: Understanding and selecting the right architectural patterns is crucial for designing robust, scalable, and maintainable software systems. This lecture provides you with the foundational knowledge to recognize and apply these patterns effectively in your projects, enabling you to tailor solutions that best fit the architectural challenges you might encounter.