CLIENT – SERVER PATTERN

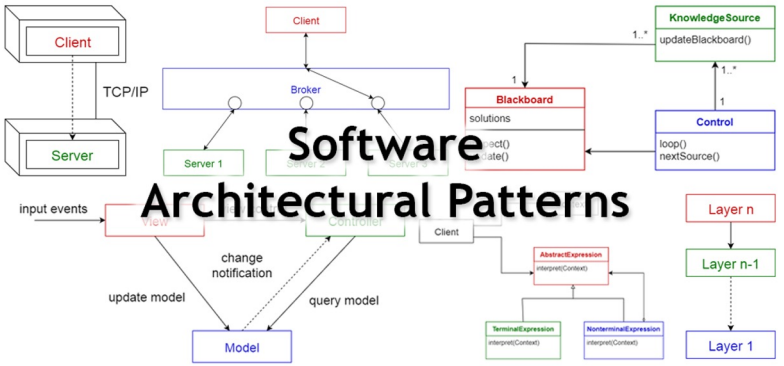
CS-3704 Intermed Software Design

By Shuai Lin, Yizhe Liu, Chenyu Mao, Reece Yankey

Group 13

Architectural Pattern

- An architectural pattern is a general, reusable solution to a common occurring problem in software architecture within a given context. Architectural patterns are similar to software design pattern but have a broader scope.
- The architectural patterns address various issues in software engineering, such as computer hardware performance limitations, high availability and minimization of a business risk.



Early History

An early form of client-server architecture is remote job entry, dating at least to 1964, where the request was to run a job, and the response was the output.

While formulating the client–server model in the 1960s and 1970s, computer scientists building ARPANET (predecessor of Internet) used the terms server-host (or serving host) and user-host (or using-host), and these appeared in the early documents.

One context in which researchers used these terms was in the design of a computer network programming language called Decode-Encode Language. The purpose of this language was to accept commands from one computer (the user-host), which would return status reports to the user as it encoded the commands in network packets. Another DEL-capable computer, the server-host, received the packets, decoded them, and returned formatted data to the user-host. A DEL program on the user-host received the results to present to the user. This is a client-server transaction.

CLIENT SERVER PATTERN

- **What is Client Server Pattern?**

  Client server pattern is a network architecture that consist of a server and multiple clients. Servers are powerful and it will provide service to multiple client components.

- **What is its precondition?**

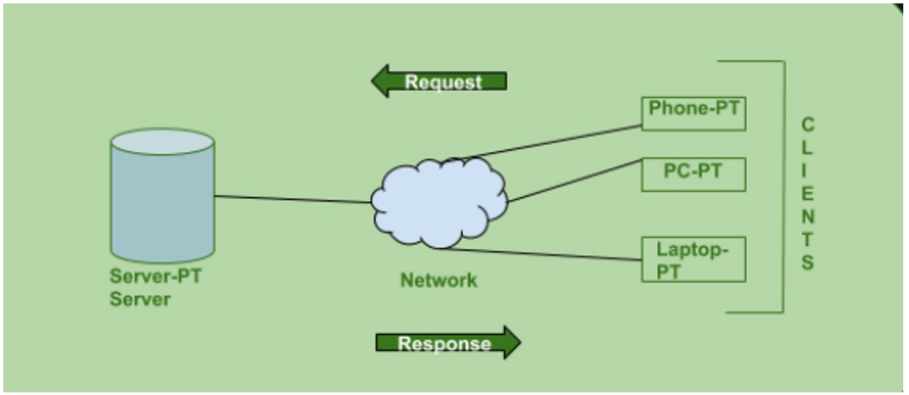
  Clients rely on servers for resources such as files, devices & processing power.

- **Where it’s used?**

  This pattern is commonly used for online applications such as file sharing, email & banking.

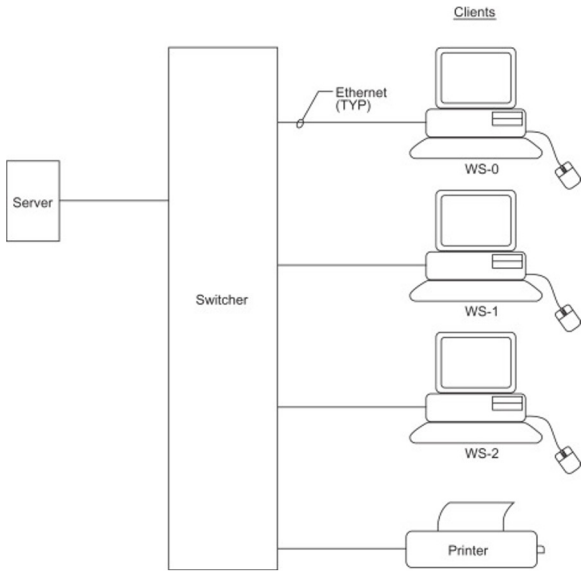
Interaction with server

- User enter URL of the website
- Look up IP address of the website
- Send request to the web server
- Server sends back necessary files
- Browser render files



Example - Hospital data processing

One client computer can use the application program for entering patient information while the server computer is using another program that manages the database in which the information is permanently stored. At the same time many other clients computer can access the server's information and do other tasks such as sending email simultaneously.



Example - Bank

- Real World Example:
  - A bank customer accesses online banking services with a web browser (the client)
  - The client initiates a request to the bank's web server
  - Since the customer's login credentials may be stored in a database such that the web server accesses the database server as a client
  - An application server interprets the returned data by applying the bank's business logic and provides the output to the web server
  - Lastly, the web server returns the result to the client web browser for display.

Example - Bank (Continued)

In each step of the sequence, the client-server exchanges message through computer processing a request and returning data. This is the request-response messaging pattern. Once all the requests are met, the sequence is complete and the web browser presents the data to the customer.

This example illustrates a design pattern applicable to the client–server model which is the separation of concerns.

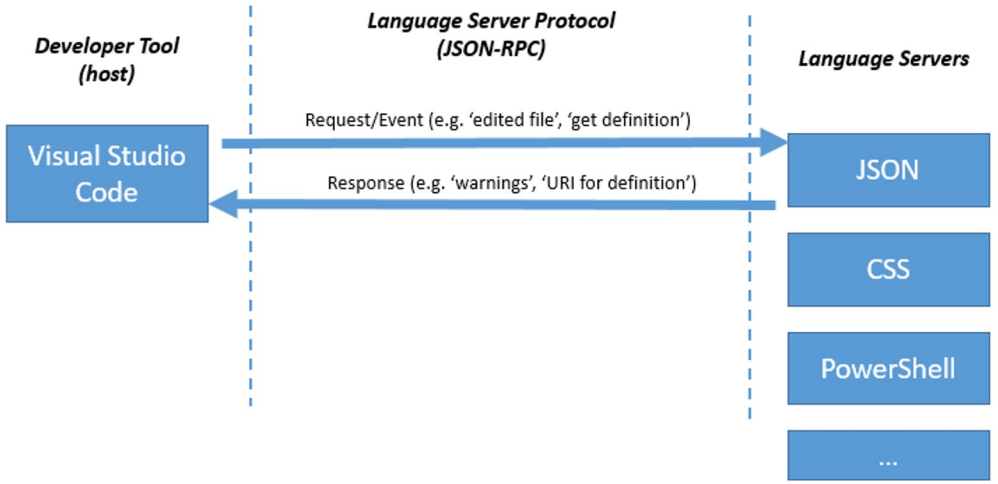
Example - Language Servers

A less conventional example is language servers

A code editor can connect to local language servers running on your machine by using the Language Server Protocol

The language server processes your code in the background to provide linting and suggestions

This model makes this process expandable and parallelizable



Advantages

- Centralization
  - All data placed in a single location
- Security
  - Well protected due to its centralization
- Scalability
  - Increase the number of servers
- Management
  - Easy to manage due to its property
- Accessibility
  - No need of processor or terminal mode

Disadvantages

- Traffic Congestion
  - Overloaded server
- Robustness
  - Main server happens to undergo failure or interference
- Cost
  - Setting up and maintaining the server
- Maintenance
  - Need specialized network manager

Reference

- https://www.topcoder.com/blog/architectural-patterns-to-consider-for-building-enterprise-applications/
- https://www.britannica.com/technology/client-server-architecture
- https://en.wikipedia.org/wiki/Client%E2%80%93server_model