1. <u>Request-Response</u>:

- Request-Response is a communication model in which the client sends requests to the server and the server responds to the requests.
- Figure below shows the client-server interactions in the request-response model

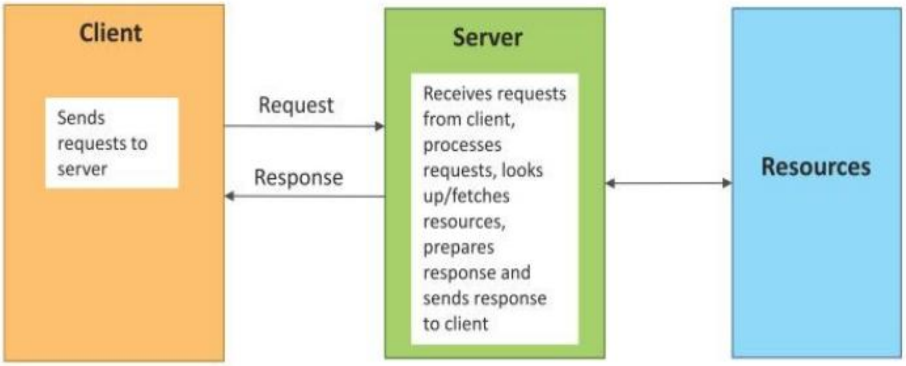
Client
Sends requests to server
Request
Response
Server
Receives requests from client, processes requests, looks up/fetches resources, prepares response and sends response to client
Resources

Figure 5: Request-Response Communication Model

- When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client.
- Request-Response model is a stateless communication model and each request-response pair is independent of others.

1. <u>Representational State Transfer (REST) - based Communication APIs</u>:

- Representational State Transfer is a software architectural style that defines a set of constraints to be used for creating web services. Web services that conform to the REST architectural style, termed RESTful web services, provide interoperability between computer systems on the Internet.
- It is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred.
- REST APIs follow the request-response communication model.
- The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.

- The REST architectural constraints are as follows

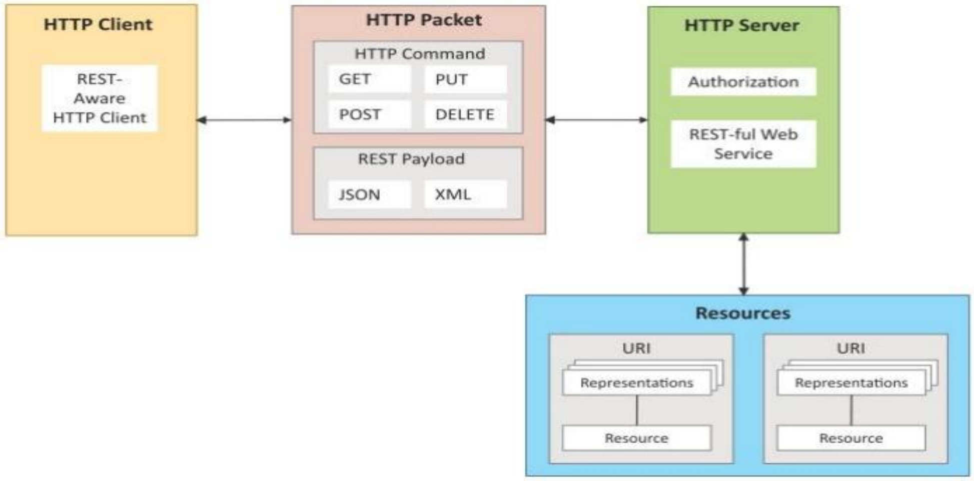

Figure 9: Communication with REST APIs

1. **Client-Server:**
   - The principle behind the client-server constraint is the separation of concerns.
   - Ex: clients should not be concerned with the storage of data which is a concern of the server. Similarly, the server should not be concerned about the user interface, which is a concern of the client. Separation allows client and server to be independently developed and updated.
2. **Stateless:**
   - Each request from client to server it must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server.
   - The session state is kept entirely on the client.
3. **Cache-able:**
   - Cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cache-able or non-cache-able.
   - If a response is cache-able, then a client cache is given the right to reuse that response data for later, equivalent requests.

- Caching can partially or completely eliminate some interactions and improve efficiency and scalability.

4. **Layered System:**
   - Layered system constrains the behavior of components such that each component cannot see beyond the immediate layer with which they are interacting.
   - Ex: a client cannot tell whether it is connected directly to the end server, or to an intermediary along the way.
   - System scalability can be improved by allowing intermediaries to respond to requests instead of the end server, without the client having to do anything different.
5. **Uniform Interface:**
   - Uniform Interface constraint requires that the method of communication between a client and a server must be uniform.
   - Resources are identified in the requests (by URIs in web based systems) and are themselves separate from the representations of the resources that are returned to the client.
   - When a client holds a representation of a resource it has all the information required to update or delete the resource (provided the client has required permissions). Each message includes enough information to describe how to process the message.
6. **Code on demand:**
   - Servers can provide executable code or scripts for clients to execute in their context. This constraint is the only one that is optional.

- A RESTful web service is a "web API" implemented using HTTP and REST principles.
- Figure 9 above shows the communication between client and server using REST APIs. Figure 10 shows the interactions in the request-response model used by REST.

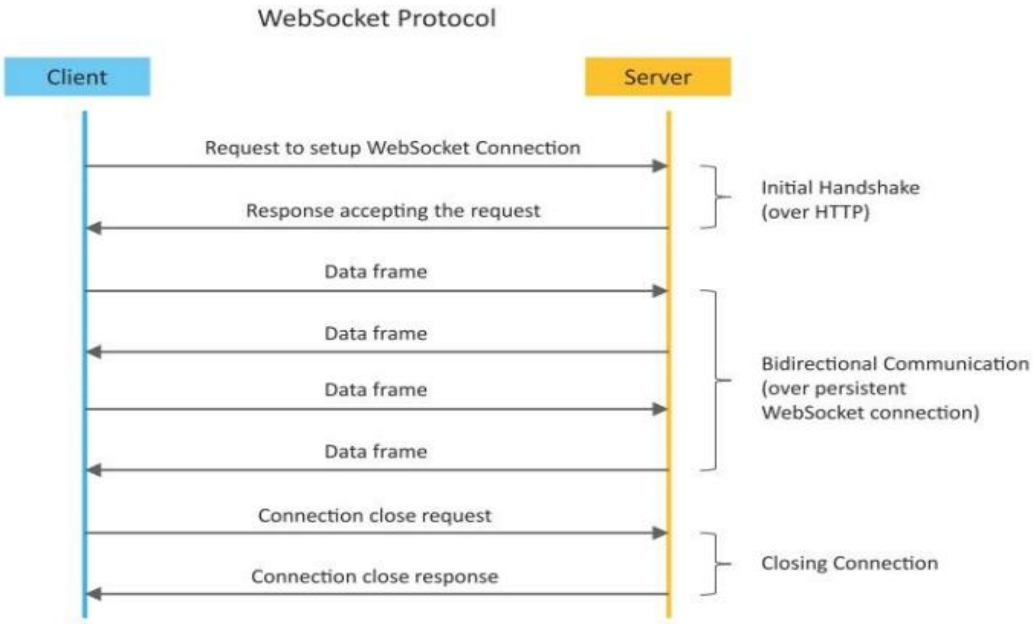
WebSocket Protocol
Client
Server
Request to setup WebSocket Connection
Response accepting the request
Initial Handshake (over HTTP)
Data frame
Data frame
Data frame
Data frame
Bidirectional Communication (over persistent WebSocket connection)
Connection close request
Connection close response
Closing Connection

Figure 10: Request-response model used by REST

- RESTful web service is a collection of resources which are represented by URLs. RESTful web API has a base URL (e.g. http://example.com/api/tasks/). The clients send requests to these URIs using the methods defined by the HTTP protocol (e.g., GET, PUT, POST, or DELETE).
- A RESTful web service can support various Internet media types (JSON being the most popular media type for RESTful web services). IP for Smart Objects Alliance (IPSO Alliance) has published an Application Framework that defines a RESTful design for use in IP smart object systems.

BY:
RAGHUDATHESH G P
Asst Professor
Dept of ECE, GMIT
Davangere - 577004
Cell: +91-7411459249
Mail: datheshraghu@gmail.com
Website: raghudathesh.weebly.com
Youtube: dathu’s class