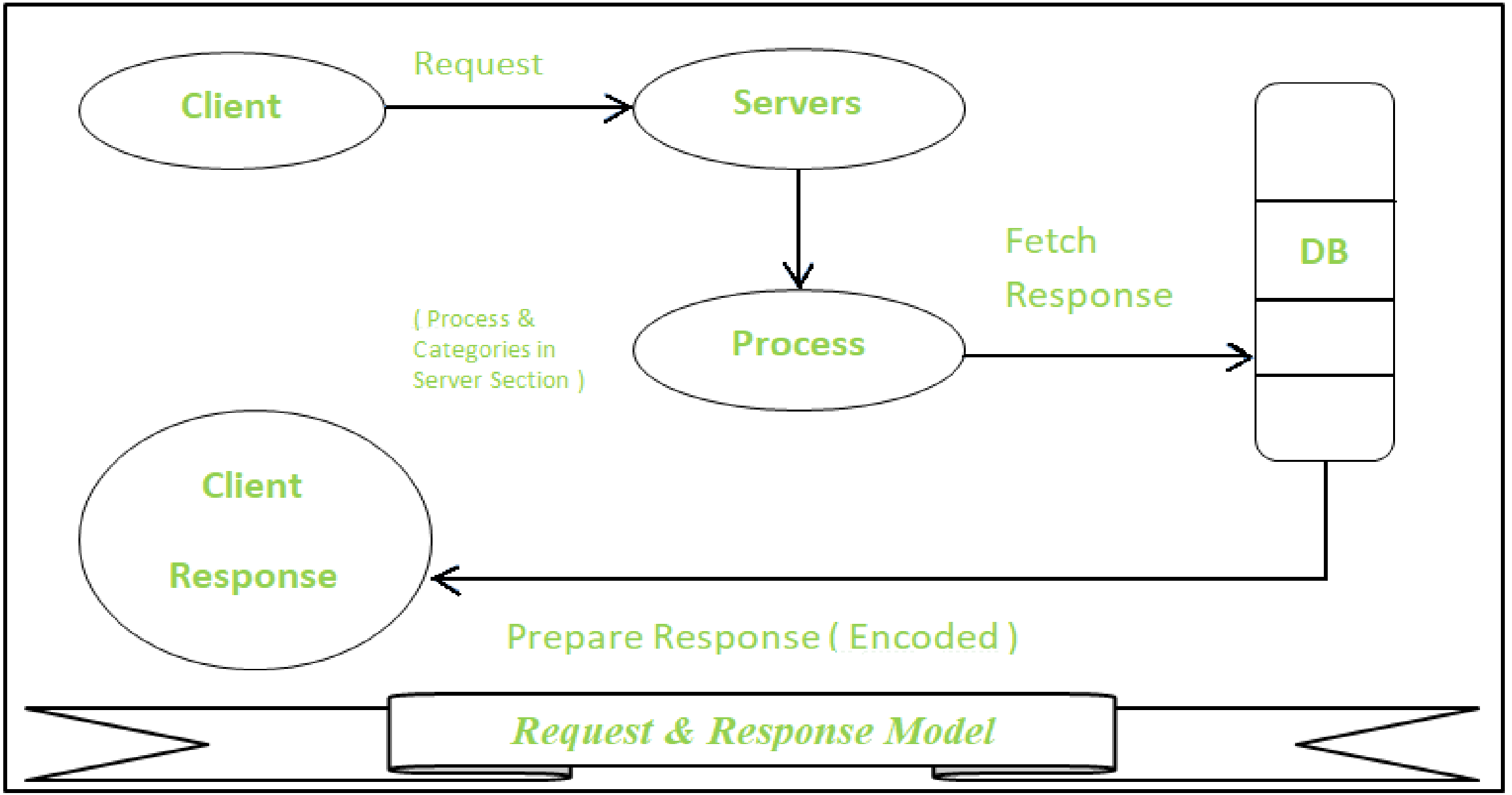
COMMUNICATION MODELS IN IOT (INTERNET OF THINGS)

- The Internet of Things (IoT) is a network of connected devices that can communicate with each other, share data, and perform tasks without human intervention. The importance of communication in IoT cannot be overstated, as it is the foundation on which the entire system is built. The devices that make up the IoT ecosystem need to be able to communicate with each other in order to function properly and achieve their intended purpose.
- Effective communication in IoT enables devices to share data, receive instructions, and respond to requests in a timely and accurate manner. This is critical for the successful implementation of IoT solutions across various industries, such as healthcare, manufacturing, transportation, and smart homes.
- For example, in a smart home, the communication between the devices (such as lights, thermostats, and security systems) allows them to work together to create a more convenient and secure living environment for the occupants. Similarly, in a healthcare setting, IoT devices can be used to monitor patients remotely and alert healthcare providers in case of an emergency, ensuring that timely medical intervention is provided.

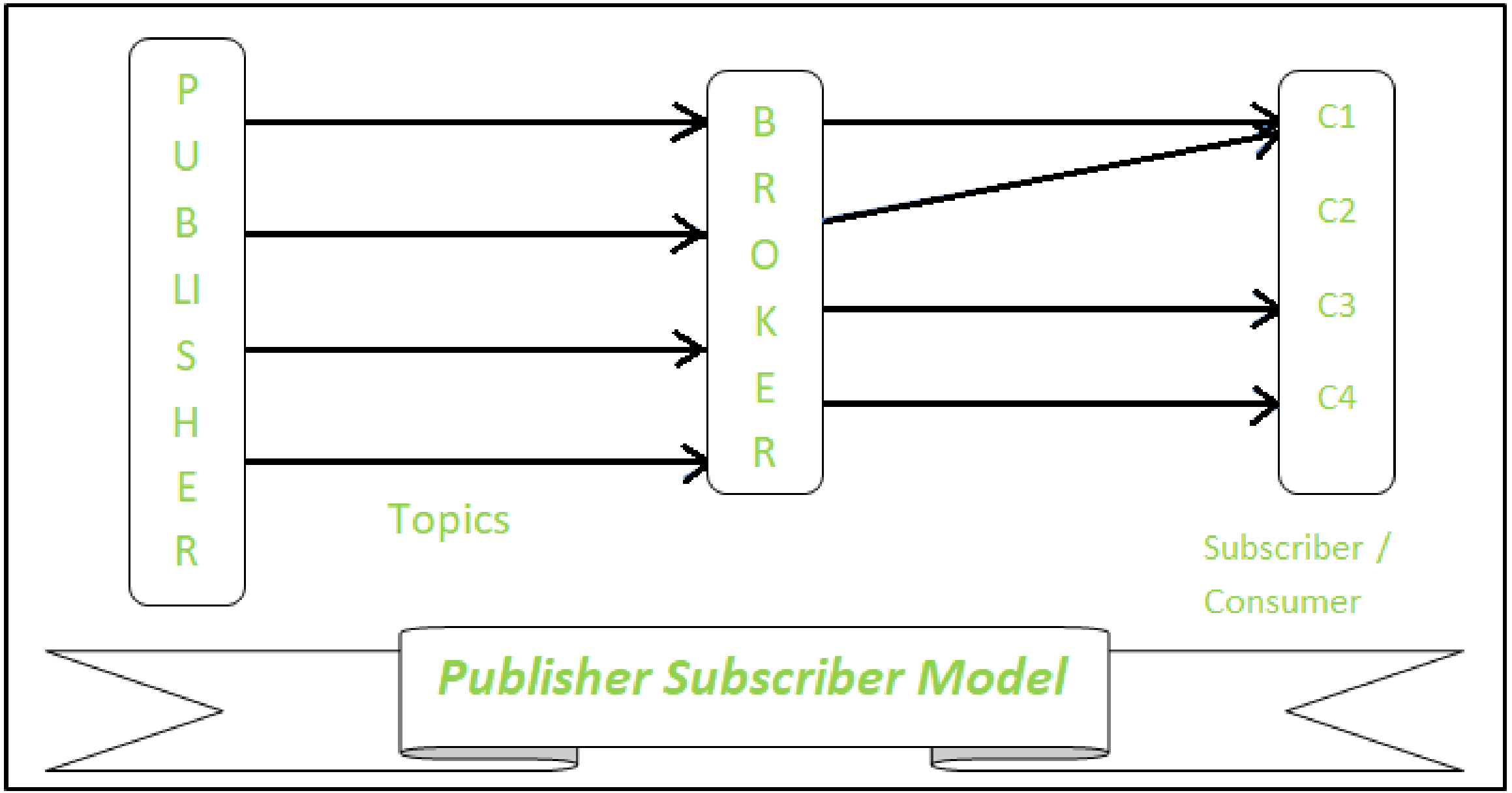
TYPES OF COMMUNICATION MODEL :

- **1. Request & Response Model** –
  This model follows a client-server architecture.
- The **client**, when required, requests the information from the server. This request is usually in the encoded format.
- This model is stateless since the data between the requests is not retained and each request is independently handled.
- The server Categories the request, and fetches the data from the database and its resource representation. This data is converted to response and is transferred in an encoded format to the client. The client, in turn, receives the response.
- On the other hand — In **Request-Response** communication model client sends a request to the server and the server responds to the request. When the server receives the request it decides how to respond, fetches the data retrieves resources, and prepares the response, and sends it to the client.


Client
Request
Servers
DB
Fetch
Response
(Process &
Categories in
Server Section)
Process
Client
Response
Prepare Response (Encoded)
Request & Response Model

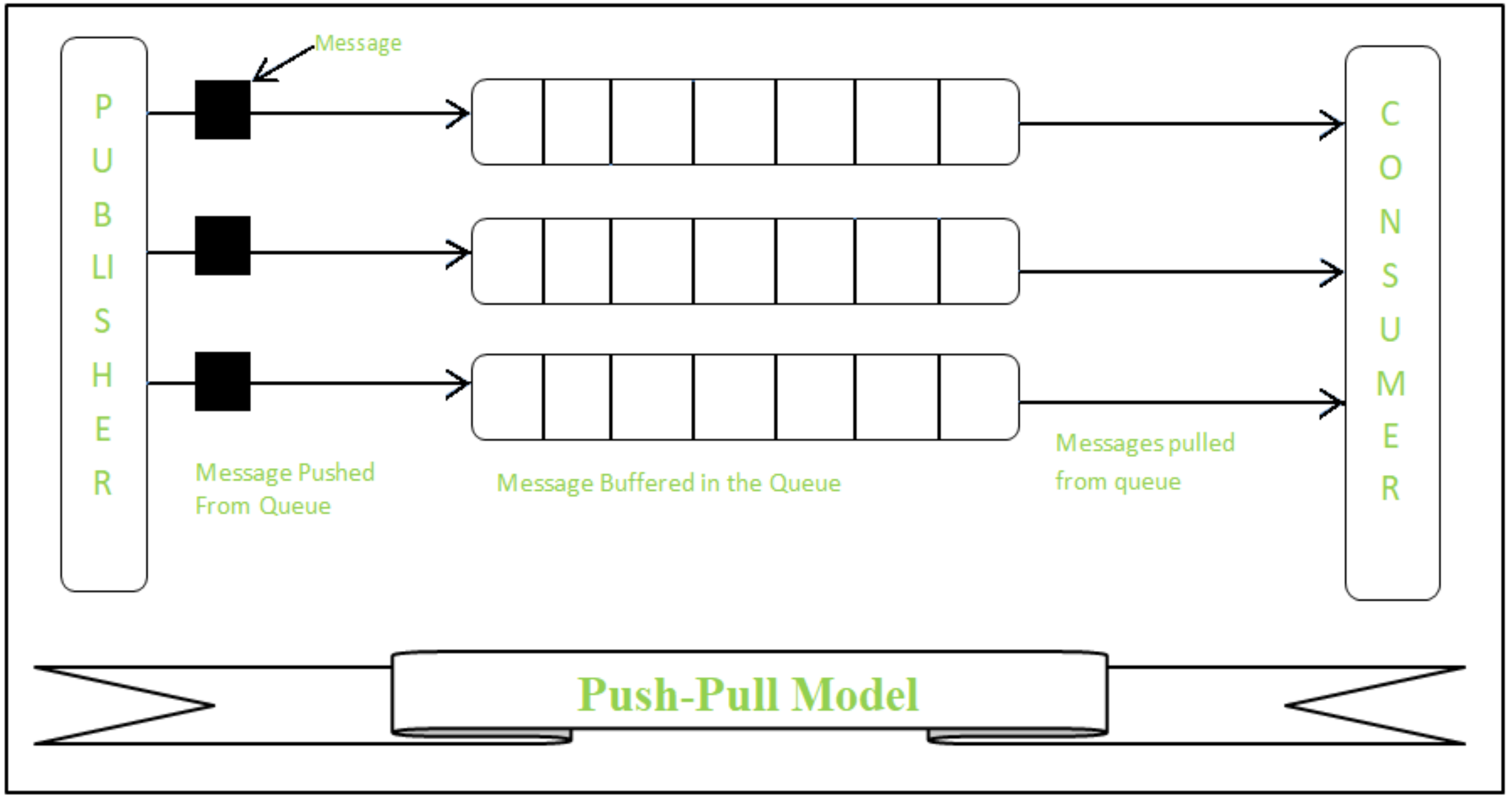
TYPES OF COMMUNICATION MODEL :

- **2. Publisher-Subscriber Model** –
  This model comprises three entities: Publishers, Brokers, and Consumers.
- **Publishers** are the source of data. It sends the data to the topic which are managed by the broker. They are not aware of consumers.
- **Consumers** subscribe to the topics which are managed by the broker.
- Hence, **Brokers** responsibility is to accept data from publishers and send it to the appropriate consumers. The broker only has the information regarding the consumer to which a particular topic belongs to which the publisher is unaware of.


PUBLISHER
BROKER
C1
C2
C3
C4
Topics
Subscriber / Consumer
Publisher Subscriber Model

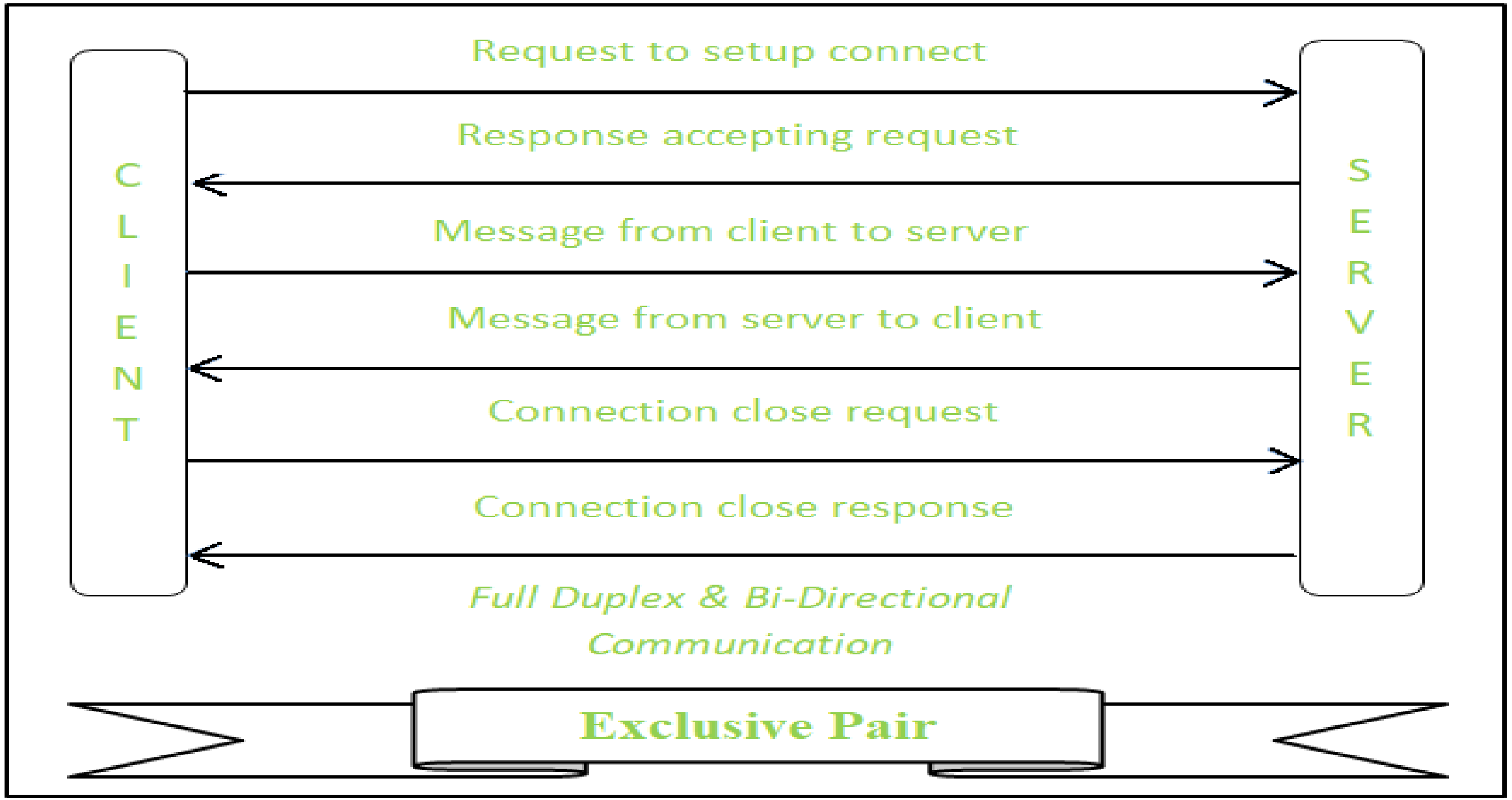
TYPES OF COMMUNICATION MODEL :

- **3. Push-Pull Model** –
  The push-pull model constitutes data publishers, data consumers, and data queues.
- **Publishers** and **Consumers** are not aware of each other.
- Publishers publish the message/data and push it into the queue. The consumers, present on the other side, pull the data out of the queue. Thus, the queue acts as the buffer for the message when the difference occurs in the rate of push or pull of data on the side of a publisher and consumer.
- **Queues** help in decoupling the messaging between the producer and consumer. Queues also act as a buffer which helps in situations where there is a mismatch between the rate at which the producers push the data and consumers pull the data.


Message
PUBLISHER
CONSUMER
Message Pushed
From Queue
Message Buffered in the Queue
Messages pulled
from queue
Push-Pull Model

TYPES OF COMMUNICATION MODEL :

- **4. Exclusive Pair –**
- **Exclusive Pair** is the bi-directional model, including full-duplex communication among client and server. The connection is constant and remains open till the client sends a request to close the connection.
- The **Server** has the record of all the connections which has been opened.
- This is a state-full connection model and the server is aware of all open connections.
- WebSocket based communication API is fully based on this model.


Request to setup connect
Response accepting request
Message from client to server
Message from server to client
Connection close request
Connection close response
CLIENT
SERVER
Full Duplex & Bi-Directional Communication
Exclusive Pair