# « Homework 1 - HTTP »


**استاد:** مهندس الوانی

**نام و نام‌خانوادگی:** نگار کرمی

**شماره دانشجویی:** ۹۷۲۲۰۳۹




**دانشگاه صنعتی امیرکبیر**

( پلی تکنیک تهران )

# Contents

## Question 1:

**A.** HTTP 1.0: **Building extensibility** - browsers and servers made http more versatile

HTTP 1.1: **The standardized protocol** - keeps all requests and responses in plain text format

HTTP 2: **A protocol for greater performance** - uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics

HTTP 3: **HTTP over QUIC** - use QUIC instead TCP/TLS for the transport layer portion

**B.** HTTP 3 benefits from the features of the UDP protocol and defines many of the new features that were in previous HTTP versions on the TCP layer. This allows to **remove the limitations existing within the Internet's infrastructure**. While HTTP/1.1 and HTTP/2 used TCP on the transport layer, HTTP/3 uses QUIC, aiming to solve one of the main problems of HTTP/2 regarding the linear progression of the downloads.
In HTTP 2 when one of them suffers a packet loss, the whole connection stops while waiting for TCP to retransmit the lost packet. HTTP 3 is **not limited by this blocking problem**, thanks to its integration on the connectionless UDP protocol. **By providing multiplexing natively, QUIC allows that lost packets only impact those transmissions where data has been lost**. Moreover, TCP handshake overhead is also removed and, as a consequence, **latency is reduced**.

## Question 2:

**A. Stateless:**

In this protocol Client send request to the server and server response according to current state. It does not require the server to retain session information or a status about each communicating partner for multiple requests. Stateless Protocol simplify the design of Server, they require less resources and the session details and each information packet travel on its own without reference to any other packet. Each communication is discrete and unrelated to those that precedes or follow.

**Stateful:**

In this Protocol If client send a request to the server, then it expects some kind of response, if it does not get any response then it resends the request. Stateful Protocols provide better performance to the client by keeping track of the connection information. Stateful Application require Backing storage. Stateful request are always dependent on the server-side state. TCP session follow stateful protocol because both systems maintain information about the session itself during its life.

B. HTTP is a stateless protocol. In order to solve the problems that may occur for some applications, it uses cookies and JSON web tokens. The information that needs to be remembered will be kept at client side and Client gives back the information to server in every request.

## Question 3:

A. Server sending a message to the client without a client initiating a request to the server asking for a specific payload is referred to as Push Request.

In Push Requests the server has to receive clients willing to receive updates/subscription. It is obvious that the client initiates the communication, by subscribing to a certain kind of events. to avoid flooding the network with broadcast messages, the server should have a way to keep track of subscribed clients. The system that keeps track of connected client together with events subscribed to uses a Queue and a session value helps to identify which clients registered to which event. Using the session, subscription key and associated payloads, a task runs either sequentially or reactively to push specific payload to registered clients every time there is a change in a topic of their interest.

This is possible because there is open communication channel and URL to the clients. The clients on another hand have to keep an open communication channel.

The server makes a decision to push a payload to subscribed clients. But unless the server broadcasts to an entire network, most client server communications are initiated by a client for identification, authentication and registration(subscription) purposes.

B.

1. The default way for this problem is using “WebSockets” (server push)

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. WebSocket is distinct from HTTP. Both protocols are located at layer 7 in the OSI model and depend on TCP at layer 4. Although they are different WebSocket is designed to work over HTTP ports 443 and 80 as well as to support HTTP proxies and intermediaries. To achieve compatibility, the WebSocket handshake uses the HTTP Upgrade header to change from the HTTP protocol to the WebSocket protocol.

The WebSocket protocol enables interaction between a client application and a web server with lower overhead than half-duplex alternatives such as HTTP polling, facilitating real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the client without being first requested by the client, and allowing messages to be passed back and forth while keeping the connection open. In this way, a two-way ongoing conversation can take place between the client and the server.

2. Long/short polling (client pull)

Polling is a technique by which the client asking the server for new data regularly. We can do polling in two ways: Short Polling and Long Polling. In simple terms, short polling is an AJAX-based timer that calls at fixed delays whereas long polling is based on Comet, server will send data to the client when the server event happens with no delay.

The server has no data and waits for some amount of time before sending the response. If something pops up during the wait, the server sends it and closes request. If there is nothing to send and the maximum wait time is achieved, the server sends a response that there is no data. In both cases, the client opens the next request for data. Lather, rinse, repeat.

3. Server-Sent events (SSE) (server push)

SSE is a real-time event emitted by the server and received by the browser. They're similar to WebSockets in that they happen in real time, but they're very much a one-way communication method from the server. The EventSource interface is used to receive Server-Sent Events. It connects to a server over HTTP and receives events in text/event-stream format without closing the connection.

The main difference to polling is that we get only one connection and keep an event stream going through it. Long polling creates a new connection for every pull.

## Question 4:

ss://asghar:1234!!@ss.myproxy.com:1234\#shadowSocks1

1 2 3 4

1. <Protocol (Scheme)>
2. <user>:<pass> => user = asghar, pass = 1234!!
3. <host>:<port> => host = ss.myproxy, port = 1234
4. <frag> => Fragment = shadowSocks1

## Question 5:

1. **502 Bad Gateway**: Gateway has successfully received login request and then it sends a request to database in order to confirm the login information and since database has faced a problem then gateway will receive an invalid response from database that will cause this error.
2. **401 Unauthorized**: It is for when authentication is required and has failed or has not yet been provided.
3. **301 Moved Permanently**: The website has been moved and needs to be redirected.

4. **429 Too Many Requests**: The user has sent too many requests in a given amount of time.
5. **200 OK**: The request was successful. The response will contain a JSON body.
6. **403 Forbidden**: Client is not allowed to access the content.

## Question 6:

1. Data center proxy
2. Forward proxy
3. Reverse proxy