Principais Protocolos de Comunicação da Internet

Protocolo de comunicação Web HTTP

- *Como funciona o protocolo HTTP?*

HTTP (HyperText Transfer Protocol) é um protocolo de comunicação, que faz o intermédio da comunicação entre cliente e servidor.

O cliente é responsável por requisitar recursos que precisa consumir, e o servidor vai responder com uma mensagem por meio do protocolo TCP.

Mensagens HTTP: request & response

Método GET chama o HTTP, quando precisa "pegar" algum conteúdo. O navegador envia uma requisição HTTP, e quando o servidor recebe a mensagem, procura nos arquivos e nos diretórios a solicitação que foi pedida pelo cliente.

Método POST é uma submissão em que o servidor é obrigado a acatar aquele conteúdo. Por exemplo um formulário, onde o cliente envia informações (como nome, data de nascimento, contato) e o servidor faz a atualização no banco de dados com as atualizações passadas. Após isso o servidor responde se conseguiu ou não fazer o update.

Estrutura de mapeamento servem para conseguir armazenar, transmitir e reconstruir os dados que são enviados ou recebidos. Os mais utilizados são o XML e o JSON. Determinamos isso através do "Content-Type". É comum esses tipos de dados serem auto descritivos, hierárquicos, independentes de linguagem de programação e cada uma com suas vantagens.

Para escolher a estrutura de mapeamento vai depender da tecnologia utilizada, se serão aplicações leves, nível de complexidade e metadados.

Particularidades do HTTP versão 1.1

TCP – Transmission Control Protocol. O cliente envia um "SYN" (mostrando que quer iniciar uma conexão), o servidor responde com "SYN ACK" (concordando) e o cliente manda "ACK" (conexão iniciada). Tem uma comunicação persistente e não-persistente associada.

Envia uma request e espera a resposta, bloqueando qualquer outra requisição enquanto não recebe a resposta. Por exemplo, recebe primeiro o arquivo main.html e depois faz outro request recebendo o file.js. Isso não é muito eficiente com recursos da web.

Mas então os navegadores fizeram atualizações que permitem que o cliente consiga fazer até 6 requisições HTTP simultâneas. Nesse caso não é preciso esperar pelo response, a menos que tenha mais de 6 requests.

É possível estabelecer uma conexão TCP Persistente, onde mantém a conexão cliente e servidor aberta. Se precisar de mais conexões, há a fila de espera.

- *Mensagens HTTP – Request & Response*

- A estrutura da mensagem HTTP Request GET é composta de:

1. Método HTTP (Get)
2. Versão do http
3. URL
4. Tipo de conexão
5. User-agent, que é o agente que realiza a requisição. Essa aplicação pode estar sendo feita via navegador, postman, aplicação java e entre outros
6. Accept-language, que é o campo do header que vai definir o idioma adotado pelo usuário

A maioria das requisições na internet utilizam o método GET, onde as pessoas buscam as informações.

Sobre a estrutura da mensagem HTTP Request, podem existir outras características em sua composição, como:

- Accept, que é o tipo de informação que o usuário aceita receber. Aí há relacionado um parâmetro de qualidade.
- No Accept-language também pode ter um fator de qualidade.
- Accept-Enconding, que quer dizer que é aceito codificação. Por exemplo, pode ser aceito gzip no modo deflate.

- A estrutura da mensagem HTTP Request POST tem informações similares ao anterior, mas novas características:

- Content-Type, que é o tipo de conteúdo recebido
- Content-Length, que é o tamanho do conteúdo
- (More data – Entity body)

Existem inúmeros métodos HTTP, mas alguns deles são estes:

Métodos HTTP	**O que faz?**
GET	Solicita um recurso do servidor
HEAD	GET sem corpo de resposta (solicita apenas o cabeçalho)
POST	Submete uma entidade a um recurso (precisa tratar os dados)
PUT	Substituição parcial de recursos pelos dados da requisição
DELETE	Deleta um recurso
TRACE	Chamada de loop-back a um determinado recurso, ótimo para fazer diagnóstico da rede e alguns problemas que podem ser encontrados.
OPTION	Retorna quais opções de comunicação há com o recurso

CONNECT	Insere um tunelamento identificado pelo recurso (faz uma ponte entre o cliente e o servidor para determinado recurso)
PATCH	Modificação parcial

Métodos seguros que não acarretam modificação do servidor: GET, HEAD e OPTION. (Operações de leitura)

- A estrutura da mensagem HTTP Response é composta de:

1. Status line, que tem a versão do protocolo, status code e status da mensagem
2. Header lines
3. Entity body

Há diversos Status code, mas alguns deles são:

- 200 OK: request bem-sucedida
- 301 Moved Permanently: objeto realocado nova URL no campo Location
- 400 Bad Request: servidor não entendeu a mensagem
- 404 Not Found: o documento solicitado é inexistente
- 505 HTTP Version Not Supported: versão do protocolo não suportada pelo servidor

Classificação dos Status Code:

- **Information** response (100 – 199)
- **Successful** response (200 – 299)
- **Redirection** response (300 – 399)
- **Client error** response (400 – 499)
- **Server error** response (500 – 599)

WebDav (Web Distributed Authoring and Versioning) – é uma ferramenta que pode ser utilizado para manipular as informações. É possível adicionar, remover, alterar, copiar, mover etc. os recursos dentro do servidor.

- *Para que servem os Cookies e Cache?*

Cookies

A ideia do cookie é fazer um track, rastrear as informações do cliente, onde possui pequenos pedaços ou blocos de dados criados e utilizados pelo servidor para persistir dados no dispositivo do cliente.

Os cookies possuem um header file, onde toda requisição e toda response carregam as informações dos cookies.

Existem os cookies de sessão e cookies persistentes. Servem por exemplo para: Manter logins ativos, informação de website e carrinho de sites comerciais.

Esses cookies podem ser invasivos, já que possui vários dados do cliente. Por isso ele deve estar de acordo com a LGPD.

Caching

Uma Web Cache também é chamada de Proxy server. Sempre que fizer uma requisição no proxy, ele vai checar se tem essas informações. Quando o cliente manda mensagem para o proxy requisitando algum cliente, o proxy é um servidor; e quando ele envia para o servidor web http uma solicitação, ele é um cliente.

- *HTTP 2.0 – Atualizações do protocolo*

Melhoria no HOL – Head of Line Blocking. Estabelece uma conexão persistente onde pode ser feita várias requisições e recebê-la de maneira diferente.

Priorização de recursos utilizando pesos, onde na versão anterior não possuía isso.

Push. Quando o cliente solicita o html e o servidor responde este, o servidor faz um Push, que é quando ele responde com todos os arquivos relacionados com a página solicitada pelo cliente. Por exemplo, o cliente solicita main.html e o servidor, além de responder com esse arquivo, também responde com main.js e img.jpg. Isto deve ser configurado com o cliente, porque as vezes pode não suportar esta opção.

HTTPS por padrão – TLS.

Negociação no handshake. No momento que está sendo estabelecido a conexão do cliente com o servidor, é feito uma verificação automática e comunicação eficiente para rodar o sistema na versão HTTP que o cliente suporta.

- *Servidores/Sistemas de aplicação*

Quais são os servidores mais utilizados? Apache, Nginx e Xampp.

Wireshark mostra a comunicação do cliente com o servidor.

HTTPS – O que muda no protocolo?

- *Conceitos básicos de segurança da informação*

Criptografia por chave. Mapear um texto legível para ilegível, “bloqueando” o conteúdo, tendo só acesso por meio da chave.

Chave assimétrica – chave privada e chave pública.

Chave simétrica – chave única privada.

Certificado digital. Verificar a autenticidade por meio da chave que a pessoa possui. É possível visualizar o certificado digital de determinado site através do cadeado ao lado da url, dizendo que a conexão é segura.

- *Protocolo SSL – Secure Socket Layer*

SSL é um protocolo de segurança e de conexão que funciona por cima do TCP. Garante confidencialidade, integridade e autenticidade end-point.

Quando utiliza HTTPS, não permite ataques ou invasões que possam prejudicar os usuários.

Operações do SSL:

Handshake – o cliente inicia a conexão TCP, recebe uma resposta e depois confirma que irá estabelecer uma conexão. Envio da EMS (Master Secret Key).

Key Derivation – chave simétrica.

DataTransfer – Transferência efetiva de dados.

Protocolo de comunicação WebSocket

- *Contextualização*

Utilizar o HTTP para aplicações de navegação web (como jogos online por exemplo) é inviável, por causa da enxurrada de requisições para manter essa conexão ativa. Conexões subjacentes, alto overhead e mapeamento via script.

A solução viável é utilizar WebSocket. Este roda encapsulado no HTTP, sem precisar criar toda uma estrutura de protocolos http. O WebSocket é voltado para aplicações web.

- *Como funciona o WebSocket?*

A API WebSocket roda por cima do HTTP, criando uma conexão bidirecional cliente e servidor, onde ambos os lados podem enviar ou receber dados. O WebSocket opera com duas fases: *Handshake* e *DataTransfer*.

Porém o Handshake é mais simplificado, de duas vias, tornando a comunicação mais rápida. Em uma série de requisições distintas, isso faz toda a diferença!

Data Transfer, ou transferência de dados.

O WebSocket, de maneira geral, possui: o modelo de origin-base security, onde a segurança é baseada na origem; endereçamento e protocolo onde o WebSocket se aproveita da estrutura HTTP para criar os endereçamentos; camadas que serão refletidos em enquadramento sobre o TCP e encerramento do handshake.

- *Como funciona a comunicação com a internet?*

Ao fazer a requisição, o DNS responde, fazendo com que o cliente seja capaz de iniciar uma conexão com o servidor HTTP. Primeiramente, ele inicia um socket TCP, onde há uma identificação IP e porta.

Quando temos diversos clientes diferentes, há o Load Balancer, que pode fazer a distribuição de carga, recebendo essas várias requisições e distribuir de acordo com os servidores HTTP disponíveis. Um outro papel dele é de multiplexação, onde recebe uma enxurrada de requisições http e multiplexar várias requisições em uma mesma conexão TCP com o servidor Apache/http, que envia as informações de volta para o cliente requisitante.

Dependendo do projeto, há empresas que, mesmo desenvolvendo aplicações web, não utilizam WebSocket, mas preferem o HTTP direto.