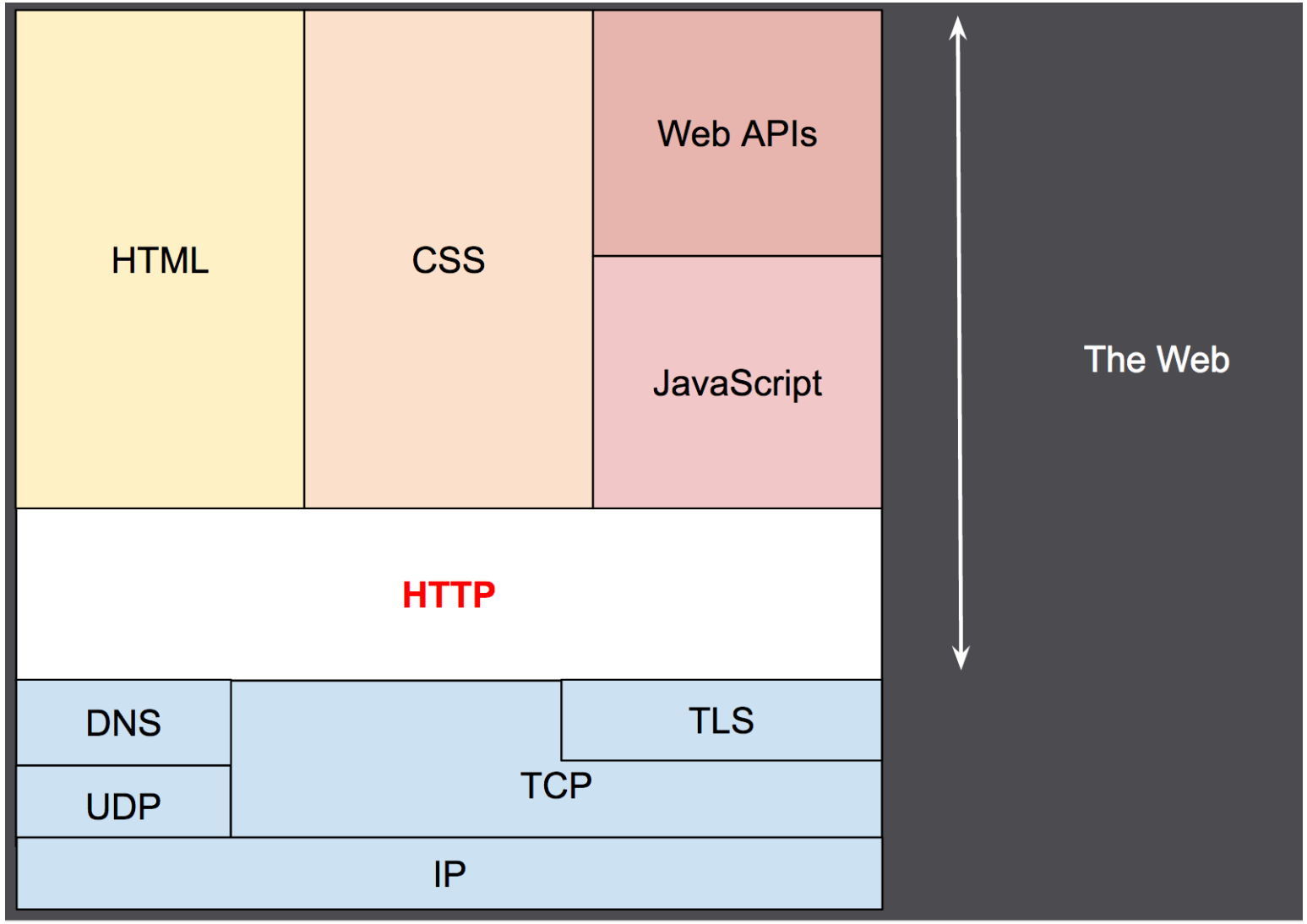
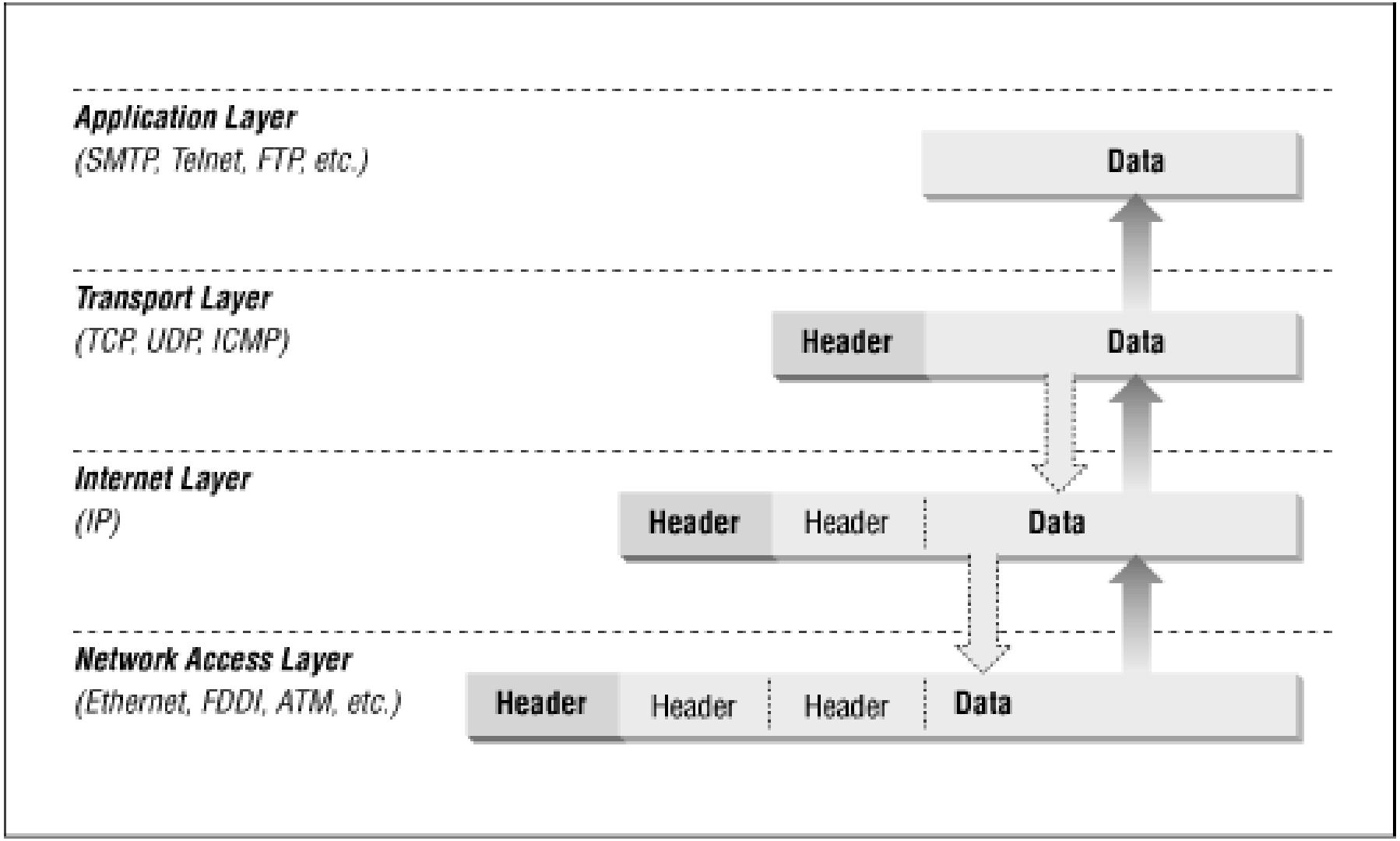
TÓPICO 09 - PROTOCOLO HTTP

Disciplina de Backend - Professor Ramon Venson - SATC 2026.1

Protocolos

- `TCP` e `IP`, responsáveis pelo endereçamento e transmissão de pacotes;
- `DNS`, responsável pela tradução de endereços para nomes de domínio;
- `HTTP` e `HTTPS`, responsáveis pelo tráfego de mensagens entre aplicações;


HTML
CSS
Web APIs
JavaScript
The Web
HTTP
DNS
TLS
TCP
UDP
IP


Application Layer
(SMTP, Telnet, FTP, etc.)
Data
Transport Layer
(TCP, UDP, ICMP)
Header
Data
Internet Layer
(IP)
Header
Header
Data
Network Access Layer
(Ethernet, FDDI, ATM, etc.)
Header
Header
Header
Data

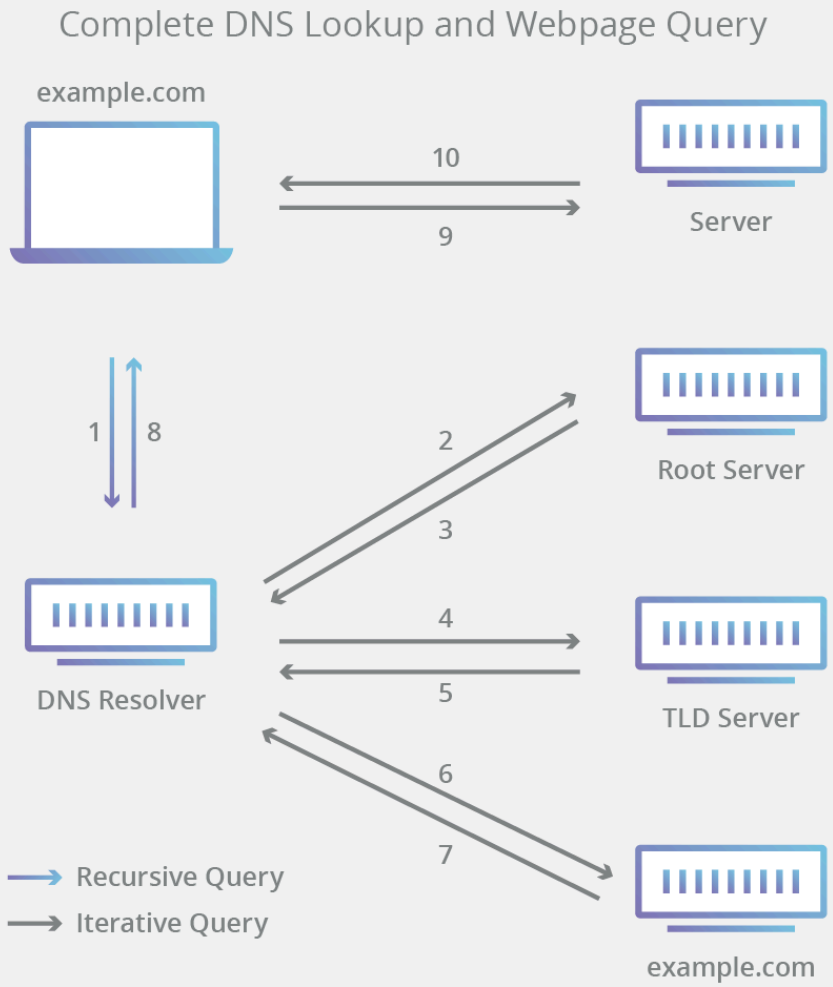
TCP e IP

Os protocolos **TCP** e **IP** são geralmente utilizados como base para todas as trocas de dados em web services

- **TCP**: Garante integridade, ordem e entrega final dos pacotes
- **IP**: roteamento dos pacotes através do endereço IP

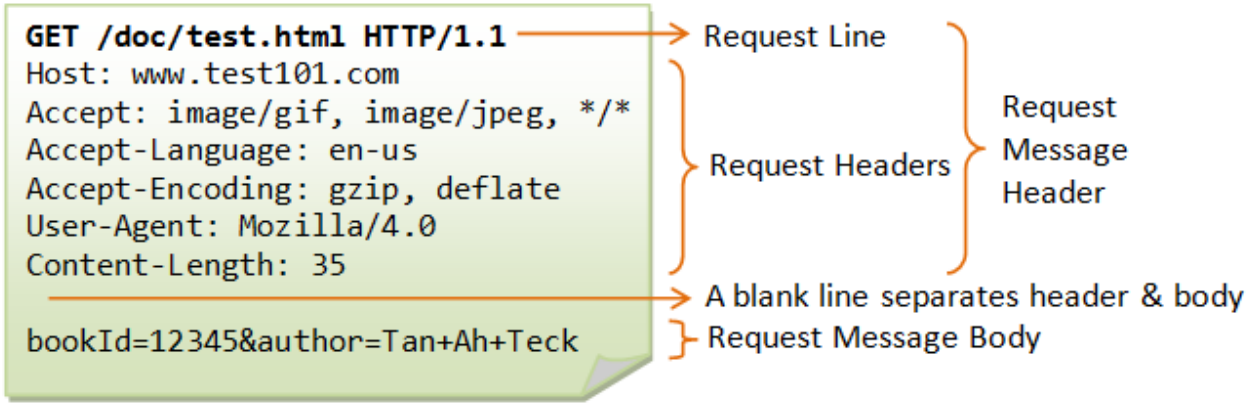
DNS

Domain Name System, responsável pela tradução dos nomes de domínio (ex.: exemplo.com.br) para um endereço IP (ex.: 20.30.2.1)



Protocolo HTTP

- Camada de Aplicação
- Requisição/Resposta
- Protocolo sem estado (stateless)
- Porta 80



Exemplo de Requisição HTTP

- Método + Caminho + Versão + Cabeçalhos + Corpo

> A conexão com o servidor acontece nas camadas de rede e transporte, mas em `HTTP/1.1` o cabeçalho `Host` continua sendo parte importante da requisição.

```
POST /login HTTP/1.1
Host: example.com
Content-Type: application/json
Connection: keep-alive

{"user": "teste", "password": "123456"}
```

Exemplo de Resposta HTTP

- Versão do HTTP (`HTTP/1.1`)
- Status Code (`200 OK`)
- Cabeçalhos (`Content-Type: text/html Date: Fri, 06 Aug 2030 01:31:51 GMT`)
- Corpo (`<html><body>Ola Mundo!</body></html>`)

```
HTTP/1.1 200 OK
Content-Type: text/html
Date: Fri, 06 Aug 2030 01:31:51 GMT

<html><body>Ola Mundo!</body></html>
```

Cabeçalhos Comuns

Cabeçalho	Descrição	Exemplo
Host	Nome do servidor	`example.com`
Content-Length	Tamanho do corpo da mensagem em bytes	`73`
Content-Type	Tipo do corpo da mensagem	`text/html`
User-Agent	Identificação do cliente	`Mozilla/5.0 (Windows NT 10.0; Win64; x64`
Authorization	Credenciais de autenticação	`Bearer <token>`

Cabeçalho	Descrição	Exemplo
Accept	Tipos de conteúdo aceitos pelo cliente	`text/html`
Cookie	Dados de sessão do cliente	`sessionId=abc123; userId=1`
Connection	Controle de conexão	`keep-alive` ou `close`
Cache-Control	Controle de cache	`no-cache, no-store, must-revalidate`
Last-Modified	Data e hora da última modificação do recurso	`Wed, 21 Oct 2015 07:28:00 GMT`
DNT	Preferência de não rastreamento do usuário	`1` (não rastrear) ou `0` (permitir rastrear)

Tipos de Conteúdo

O formato do corpo da mensagem geralmente é identificado usando um cabeçalho chamado `Content-Type`. Exemplos:

- Texto plano genérico (`text/plain`)
- Áudio (`audio/mpeg`), Vídeo (`video/mp4`) e Imagem (`image/png`)
- Binário (`application/octet-stream`)
- HTML (`text/html`)
- Formulário (`multipart/form-data`)
- JSON (`application/json`)

`Content-Type` x `Accept`

- `Content-Type` indica o formato do corpo enviado
- `Accept` indica os formatos que o cliente aceita receber

```
GET /produtos HTTP/1.1
Host: example.com
Accept: application/json
```

Métodos seguros e idempotentes

- Métodos **seguros** não devem alterar o estado do servidor (`GET`, `HEAD`, `OPTIONS`)
- Métodos **idempotentes** produzem o mesmo efeito quando repetidos (`GET`, `PUT`, `DELETE`)
- `POST` normalmente **não** é idempotente

`401` x `403`

- `401 Unauthorized`: cliente não se autenticou corretamente
- `403 Forbidden`: cliente está autenticado, mas não tem permissão

Exemplos:

- token inválido ou ausente -> `401`
- usuário comum tentando acessar rota administrativa -> `403`

Cache HTTP

Cabeçalhos de cache ajudam a evitar transferências desnecessárias.

- `Cache-Control`: define políticas de cache
- `ETag`: identifica uma versão específica do recurso

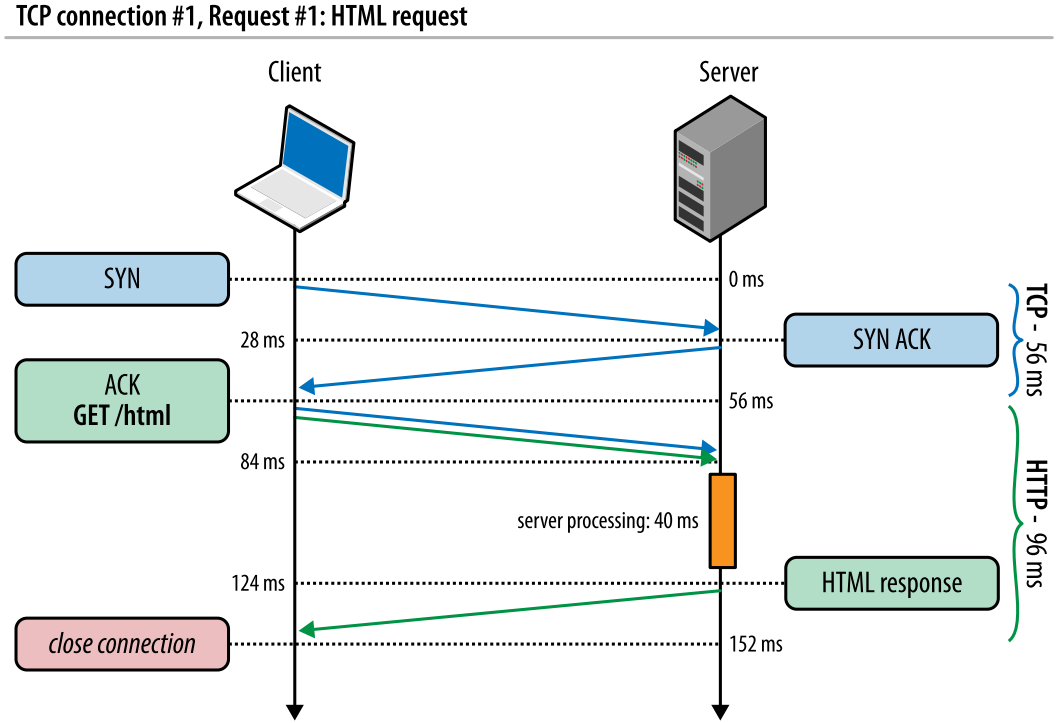
Exemplo:

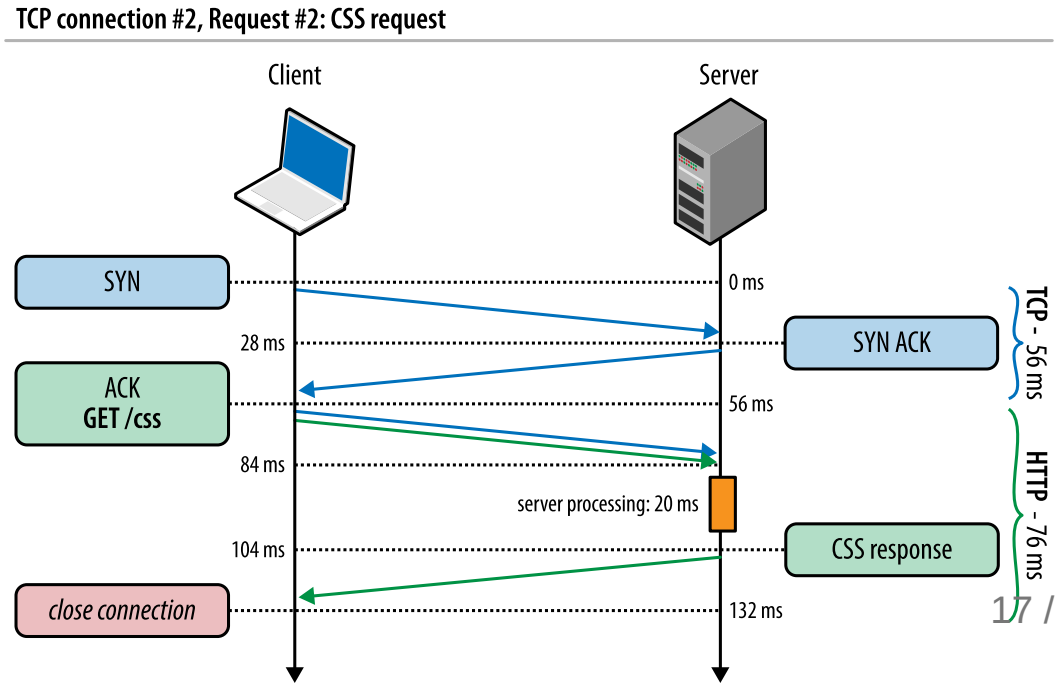
```
HTTP/1.1 200 OK
Cache-Control: max-age=3600
ETag: "produto-10-v3"
```

Estabelecendo uma conexão

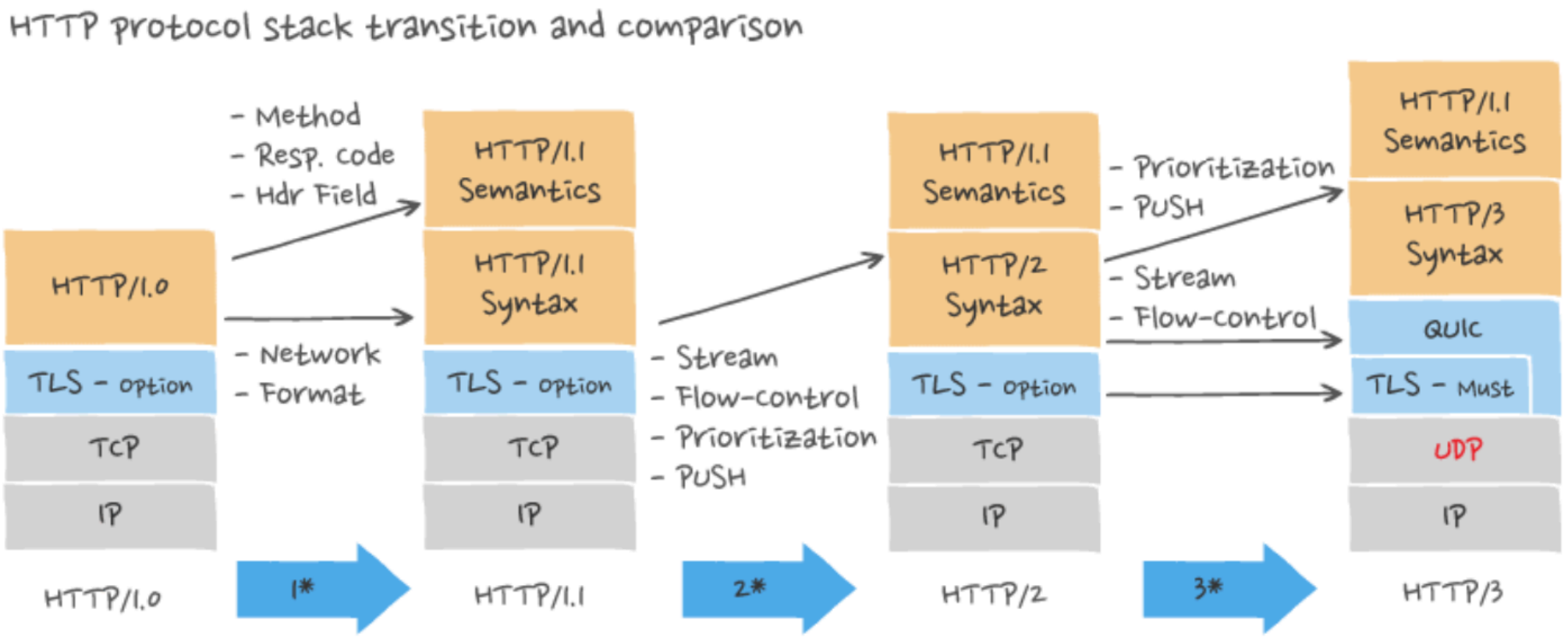
Sempre que uma requisição HTTP é feita, é preciso que as camadas de rede/transporte realizem a conexão com o servidor antes de realizar qualquer requisição.

Conexões do tipo ***Keep-Alive*** podem ser utilizadas para múltiplas requisições.





Comparação entre versões HTTP



- `1.0`: 1996
  - Pouco/Nenhum suporte a compressão
  - Apenas `GET`, `HEAD` e `POST`
- `1.1` (mais utilizado): 1997
  - Novos Métodos e Códigos de Resposta
  - Keep-Alive
  - Compressão aprimorada do corpo da mensagem

- `2.0`: 2015
  - Introduz protocolo binário ao invés de texto
  - Compressão do cabeçalho
  - Resposta e Requisições Multiplexada na camada de aplicação
    - Permite "enfileirar" requisições e respostas
    - Pacotes perdidos na camada de rede/transporte ainda bloqueiam as requisições
  - Server Push existiu na especificação, mas teve adoção prática limitada

- `3.0`: 2018
  - Troca do TCP+TLS para UDP+QUIC
  - Multiplexação não bloqueante na camada de transporte
    - Pacotes perdidos na camada de rede/transporte não bloqueiam outros pacotes
  - Criptografia obrigatória (TLS 1.3)
  - Conexão identificada por `Connection ID` ao invés de apenas endereço IP e porta
  - Menor latência

HTTPS

- Extensão do procolo HTTP
- Utiliza uma camada de segurança chamada *Transport Layer Security* (TLS)
- Negociação da criptografia usando chaves públicas
- Iniciada durante a fase de conexão
- Porta 443

> O TLS é uma versão mais recente do *Secure Sockets Layer* (SSL), termo ainda usado para descrever a mesma camada.

O que aprendemos nessa aula?

- O que são serviços web e exemplos
- Quais os protocolos mais utilizados por aplicações web
- Como são realizadas as requisições HTTP
- Quais ferramentas vamos utilizar para construir aplicações web