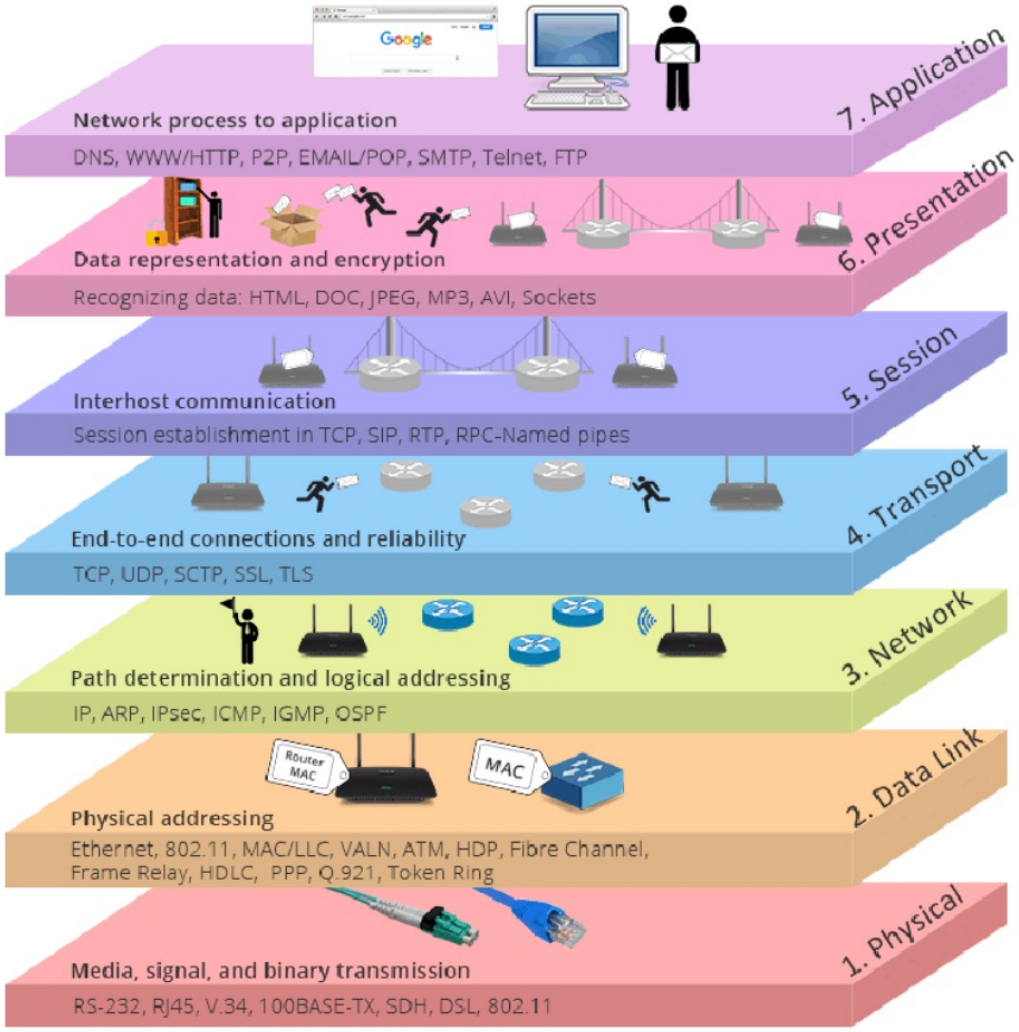
Лекция 4
HTTP

Проектирование систем и продуктовая веб-разработка

Канев Антон Игоревич

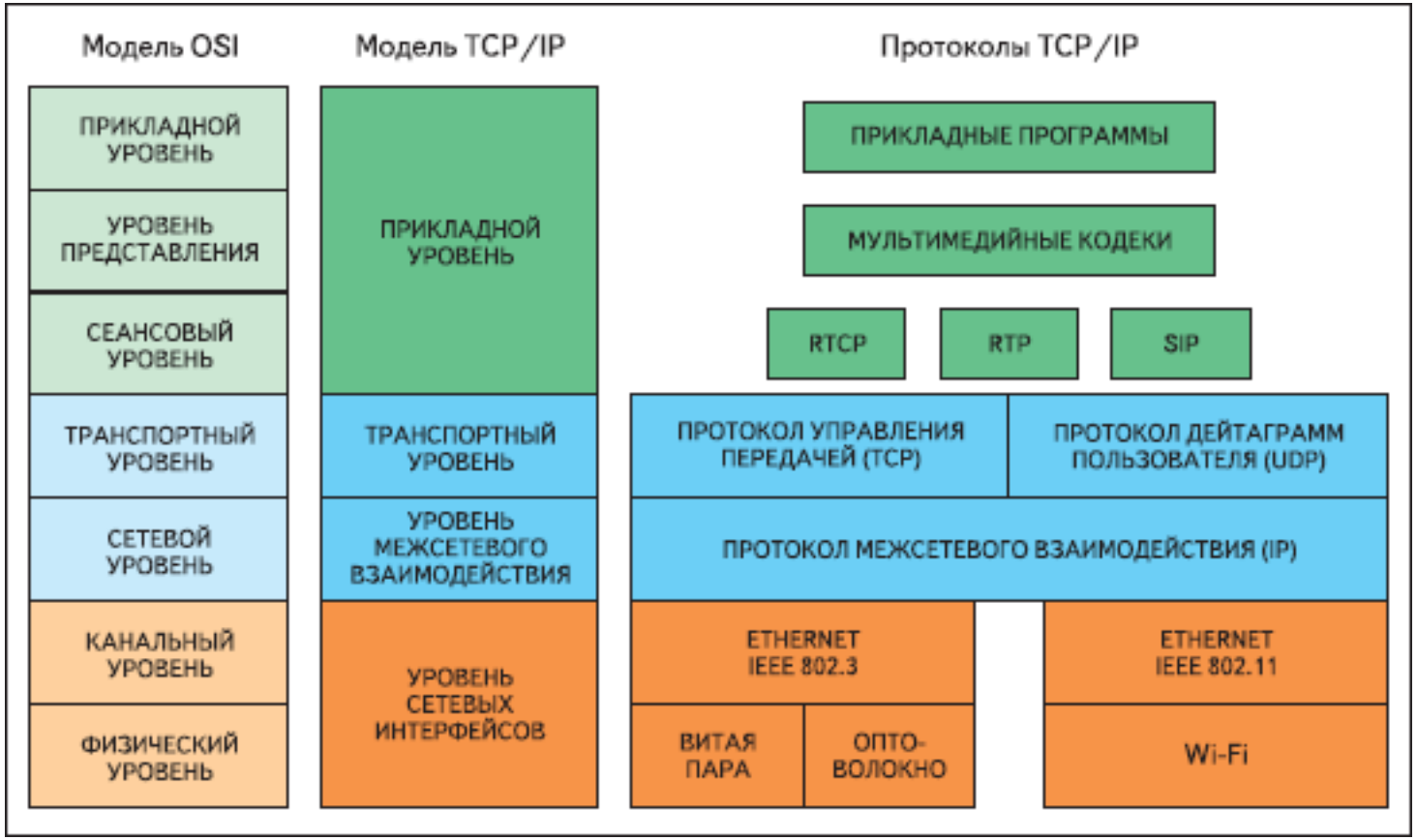
7-ми уровневая модель OSI

- Приложения (браузер, веб-сервер, наши мобильные приложения) работают на самом высоком 7-ом **Прикладном** уровне
- Уровень **Представления** – форматы данных, HTML, mp4 и тд
- **Сеансовый** для соединения в TCP
- **Транспортный** делит сообщение на универсальные сегменты
- **Сетевой** позволяет найти устройство (IP как почтовый адрес)
- 4-3-2 уровни – это как матрешка сегмент-пакет-кадр
- Физическая среда передачи на **Физическом** уровне (какие провода, длина света), на **Канальном** уровне организация канала (сегмент Wi-Fi) + постоянный MAC адрес (как паспорт)



Модель TCP/IP

- TCP/IP – стек протоколов на которых базируется Интернет
- TCP и IP появились с самого начала, находясь в середине – это **основа глобальной сети**
- Для приложений могут быть разные протоколы
- Физическая среда тоже может быть разная



Стандарты интернета

- В отличие от корпоративных систем, интернет изначально строится на открытых стандартах. Эти стандарты открыто опубликованы, любое заинтересованное лицо может принять участие в их разработке.
- Разработкой стандартов занимается IETF
  - Официальный сайт https://www.ietf.org
  - Список RFC опубликован здесь https://www.rfc-editor.org/rfc-index.html
- Стандарты для URL, HTTP, FTP.

Стандарты интернета – RFC Long Polling

Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP

RFC 6202

Status | IESG evaluation record | IESG writeups | Email expansions | History

Versions:

07 RFC 6202

Document	**Type**	RFC - Informational (April 2011) Was draft-loreto-http-bidirectional (individual in app area)
	Authors	Peter Saint-Andre, Salvatore Loreto, Stefano Salsano, Greg Wilkins
	Last updated	2015-10-14
	RFC stream	Internet Engineering Task Force (IETF)
	Formats	txt html pdf htmlized bibtex
	Reviews	SECDIR Last Call review by Julien Laganier
Stream	**WG state**	(None)
	Document shepherd	(None)
IESG	**IESG state**	RFC 6202 (Informational)
	Action Holders	(None)
	Consensus boilerplate	Unknown
	Telechat date	(None)
	Responsible AD	Alexey Melnikov
	Send notices to	Martin.Thomson@andrew.com

experienced by client applications due to the frequent opening and closing of connections necessary to periodically poll for data.

The two most common server-push mechanisms are HTTP long polling and HTTP streaming:

HTTP Long Polling: The server attempts to "hold open" (not immediately reply to) each HTTP request, responding only when there are events to deliver. In this way, there is always a pending request to which the server can reply for the purpose of delivering events as they occur, thereby minimizing the latency in message delivery.

HTTP

- HTTP – протокол передачи гипертекста. Сейчас для произвольных данных

Протокол является клиент-серверным, то есть существуют:

- Клиент – потребитель, он инициирует взаимодействие
- Сервер – поставщик, ждет запроса, обрабатывает его и возвращает обратно ответ

- Данный тип взаимодействия накладывает ограничение при получении уведомлений, сообщений в чате и тд
- HTTP – прокол без хранения состояния между разными запросами. Но компоненты могут хранить например куки (клиенты) или сессии (сервер)

HTTP request/response

- Методы

GET, POST, PUT, …

- Коды состояний

200 OK

404 Not Found

- Заголовки

параметр: значение

File Edit View Search Terminal Help

```
[osboxes@osboxes ~]$ telnet iu5.bmstu.ru 80
Trying 195.19.50.252...
Connected to iu5.bmstu.ru.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Server: nginx
Date: Mon, 09 Nov 2020 08:53:01 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 985
Connection: close
Last-Modified: Fri, 12 Apr 2019 09:22:18 GMT
ETag: "3d9-58651d6d73b52"
Accept-Ranges: bytes

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"><head>
    <title>hoster1.uimp.bmstu.ru &mdash; Coming Soon</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta name="description" content="This is a default index page for a new domain."/>
    <style type="text/css">
        body {font-size:10px; color:#777777; font-family:arial; text-align:center;}
        h1 {font-size:64px; color:#555555; margin: 70px 0 50px 0;}
        p {width:320px; text-align:center; margin-left:auto;margin-right:auto; margin-top: 30px }
        div {width:320px; text-align:center; margin-left:auto;margin-right:auto;}
        a:link {color: #34536A;}
        a:visited {color: #34536A;}
        a:active {color: #34536A;}
        a:hover {color: #34536A;}
    </style>
</head>
```

Методы

- OPTIONS
- GET - запрос содержимого ресурса («select»)
- HEAD
- POST – передача ресурсу данных пользователя («insert»)
- PUT – загрузка содержимого запроса на ресурс («update»)
- PATCH – PUT к фрагменту ресурса (например статус)
- DELETE – удаление ресурса
- TRACE
- CONNECT

Коды состояния

Код	Класс	Назначение
1xx	Информационный (**informational**)	Информирование о процессе передачи. В HTTP/1.0 — сообщения с такими кодами должны игнорироваться. В HTTP/1.1 — клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно.
2xx	Успех (**Success**)	Информирование о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса, сервер может ещё передать заголовки и тело сообщения.
3xx	Перенаправление (**Redirection**)	Сообщает клиенту, что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location.
4xx	Ошибка клиента (**Client Error**)	Указание ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.
5xx	Ошибка сервера (**Server Error**)	Информирование о случаях неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

URI

URI – схема HTTP

- http:// хост : порт / путь и имя файла ? параметры # якорь гиперссылки
- Пример: http://iu5.bmstu.ru:8080/cat1/cat2/script.asp?param1=1&param2=2#anchor1
- Порт по умолчанию – 80.

URI – схема FTP

- ftp://пользователь : пароль @ хост : порт / путь и имя файла
- Пример: ftp://user:password@host1.com/public/1.txt
- Порт по умолчанию – 21.

URI – схема mailto

- Предполагает использование протокола SMTP
- mailto:adr1@mail.ru?cc=adr2@mail.ru&subject=тема &body=тело письма

URI

- Рекомендуется использовать наиболее общий термин URI, хотя во многих спецификациях можно также встретить термин URL. Фактически, все адреса в WWW обозначающие ресурсы, являются URL.

- URI (URL) используется в гипертекстовых ссылках для обозначения ресурсов. С помощью URL можно адресовать как гипертекстовые документы формата HTML, так и другие ресурсы, например электронную почту, ftp.

- Для создания URI на национальных языках разрабатывается стандарт IRI.

URI

Нормализация URI

- не показывать порт, имя файла
- нижний регистр
- IP заменить на домен

Семантический URI – понятный пользователю

HTTP://www.Example.com

Non-semantic URL	Semantic URL
http://example.com/index.php?page=name	http://example.com/name
http://example.com/index.php?page=consulting/marketing	http://example.com/consulting/marketing
http://example.com/products?category=2&pid=25	http://example.com/products/2/25
http://example.com/cgi-bin/feed.cgi?feed=news&frm=rss	http://example.com/news.rss

История HTTP

- 1991 HTTP/0.9 Только GET
- 1996 HTTP/1.0 Появился POST и др методы, заголовки, кеширование
- 1999 HTTP/1.1 Keep alive, host (виртуальный хостинг)

- 2015 HTTP/2 На основе SPDY: Бинарный, сжатие заголовков, мультиплексирование, приоритезация ресурсов, отмена загрузки, Server Push

- Draft HTTP/3 На основе QUIC, из-за развития беспроводных сетей и возникающих из-за этого проблем в стеке (миграции IP и тд): ID соединения, отказ от TCP

Доля трафика

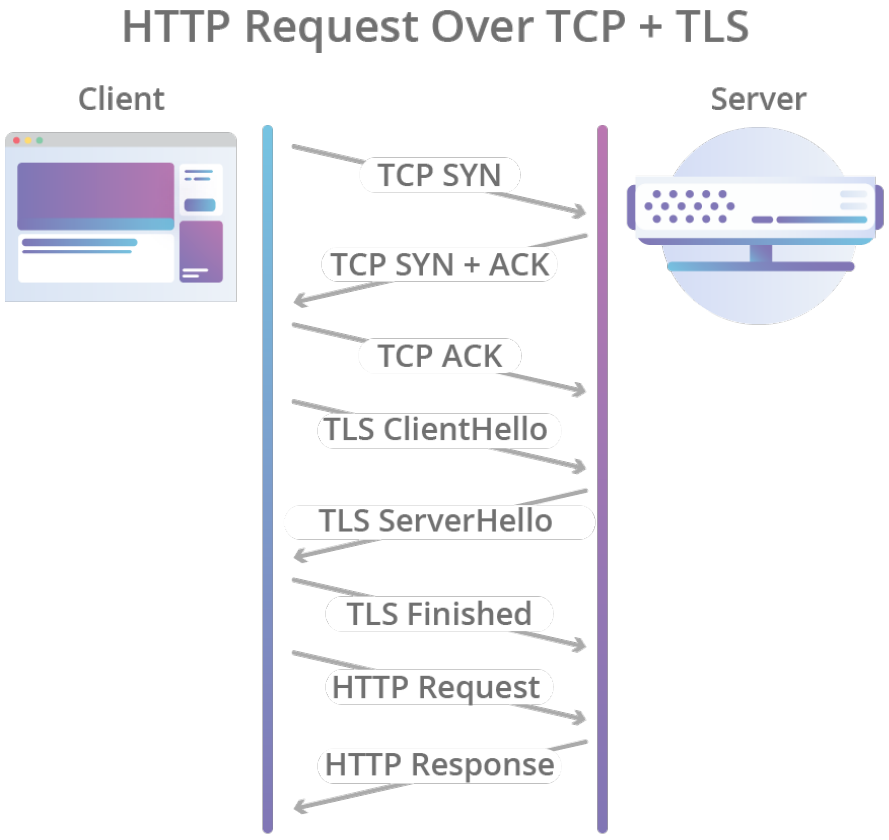
	2014 1 Jan	2015 1 Jan	2016 1 Jan	2017 1 Jan	2018 1 Jan	2019 1 Jan	2020 1 Jan	2021 1 Jan	2022 1 Jan	2023 1 Jan	2024 1 Jan	2025 1 Jan	2025 22 Sep
None	4.8%	5.1%	2.4%	0.8%	0.6%	0.6%	0.8%	1.6%	1.6%	1.1%	0.8%	0.6%	0.8%
CSS	90.0%	90.0%	93.7%	95.9%	95.6%	95.8%	95.5%	96.1%	96.1%	96.7%	97.0%	96.9%	96.2%
Compression	52.9%	59.1%	64.3%	70.3%	73.6%	77.2%	80.7%	83.0%	87.0%	87.8%	88.6%	89.8%	90.1%
Default protocol https					26.9%	45.9%	57.4%	68.8%	77.4%	81.2%	84.9%	87.7%	88.7%
Cookies	46.6%	46.0%	49.3%	50.6%	49.8%	46.6%	45.1%	44.9%	36.7%	40.8%	43.0%	41.8%	40.3%
Default subdomain www					42.2%	42.7%	40.8%	38.9%	38.1%	40.6%	40.3%	38.7%	38.6%
HTTP/3							2.3%	4.1%	24.2%	25.1%	27.8%	33.6%	35.8%
HTTP/2			5.6%	11.2%	23.1%	32.5%	42.6%	49.9%	46.9%	40.0%	35.5%	34.5%	33.3%
HTTP Strict Transport Security	0.1%	0.4%	1.0%	2.4%	5.3%	9.2%	12.1%	16.9%	22.8%	24.9%	27.3%	29.7%	31.7%
IPv6	2.9%	5.2%	6.1%	9.6%	11.4%	13.3%	15.0%	17.5%	20.6%	21.0%	22.6%	26.0%	27.3%
ETag	15.7%	15.1%	15.2%	14.5%	14.1%	14.4%	14.1%	14.4%	16.2%	17.3%	21.9%	24.1%	26.3%
QUIC					0.6%	1.4%	2.8%	5.1%	7.2%	8.5%	7.7%	8.5%	8.9%
Frameset	1.8%	0.9%	0.7%	0.7%	0.6%	0.6%	0.5%	0.3%	0.2%	0.2%	0.2%	0.2%	0.2%
SPDY	0.6%	3.2%	6.3%	7.5%	9.1%	0.6%	0.2%	0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%

https://w3techs.com/technologies/history_overview/site_element/all/y

Записать
2014, 2020, 2025 годы

HTTP

- Красный знак не доставленного сообщения в телеграм: мы нумеруем все сегменты и получаем ACK в ответ
- 3-е рукопожатие – договориться о номерах



- Протокол HTTP (поверх TCP)
- Протокол HTTPS (поверх TLS и TCP)
- Протокол HTTP/2
- Протокол HTTP/3 (поверх QUIC)

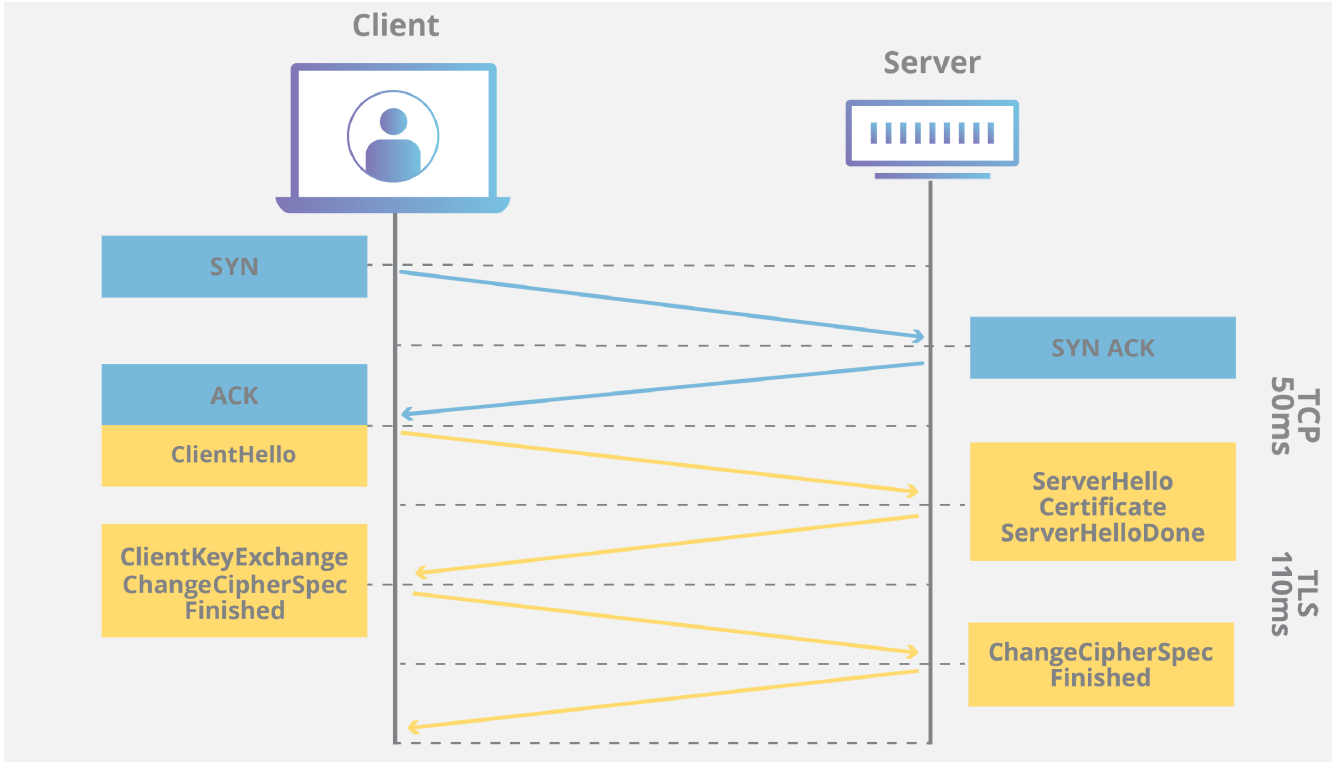
- Протоколы на основе HTTP:

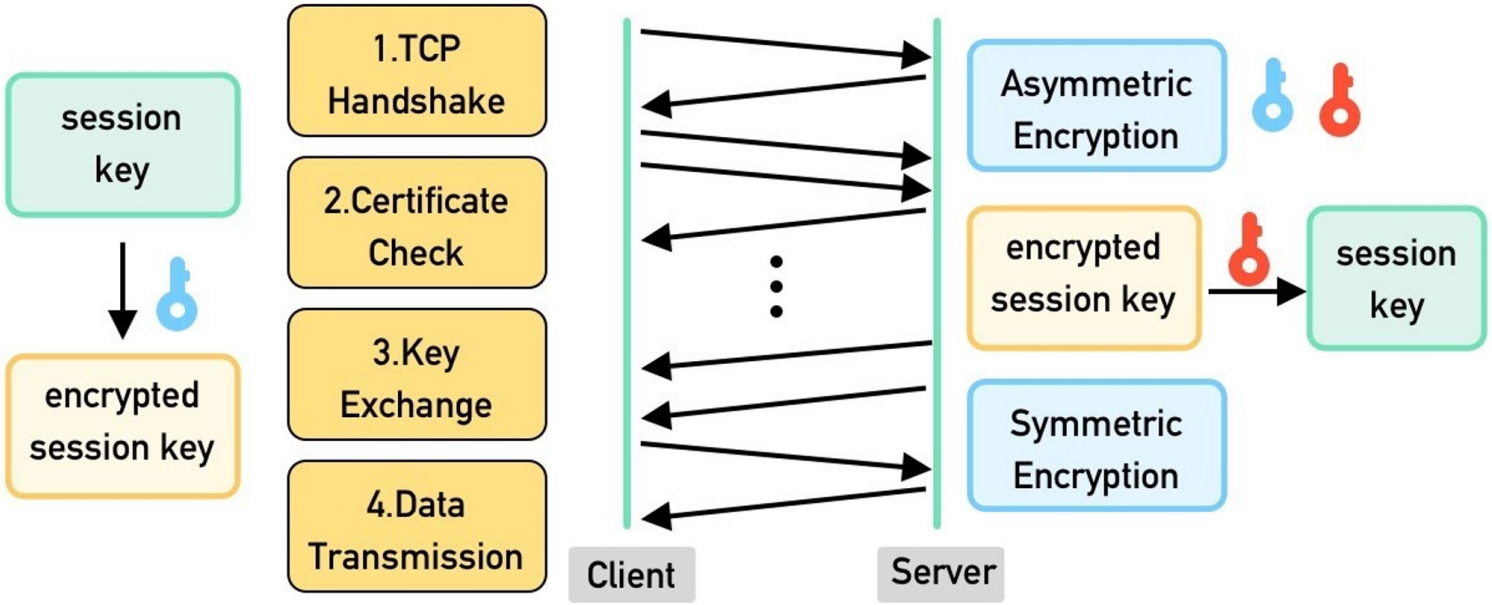
– JSON-RPC (AJAX), XML-RPC, SOAP, WebDAV, gPRC

HTTPS

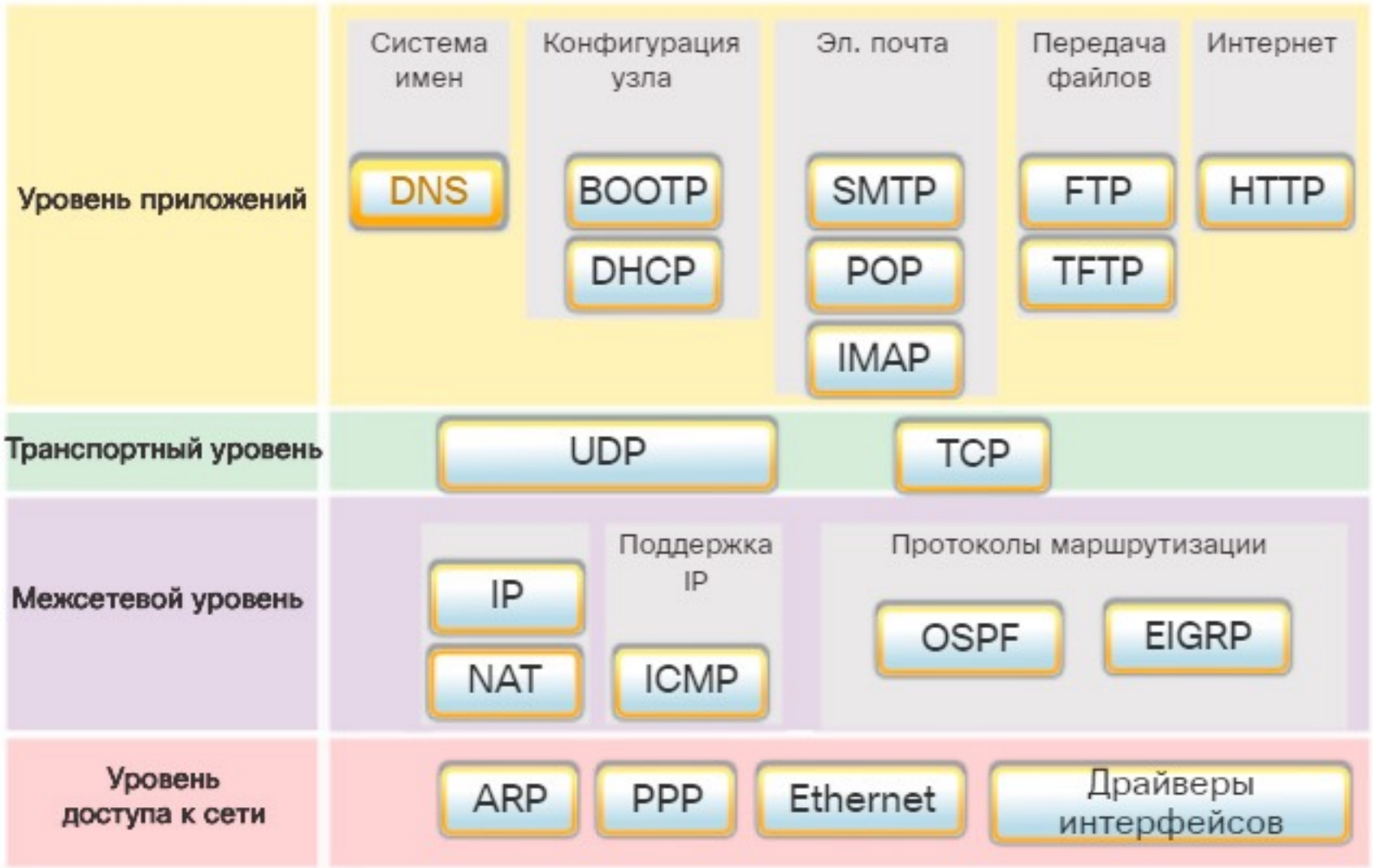
- HTTPS - протокол на основе TLS (ранее SSL) для аутентификации (асимметричное) и конфиденциальности (симметричное шифрование)

- Разработан 1994 году Netscape (родилась из браузера Mosaic)
- Порт по умолчанию 443



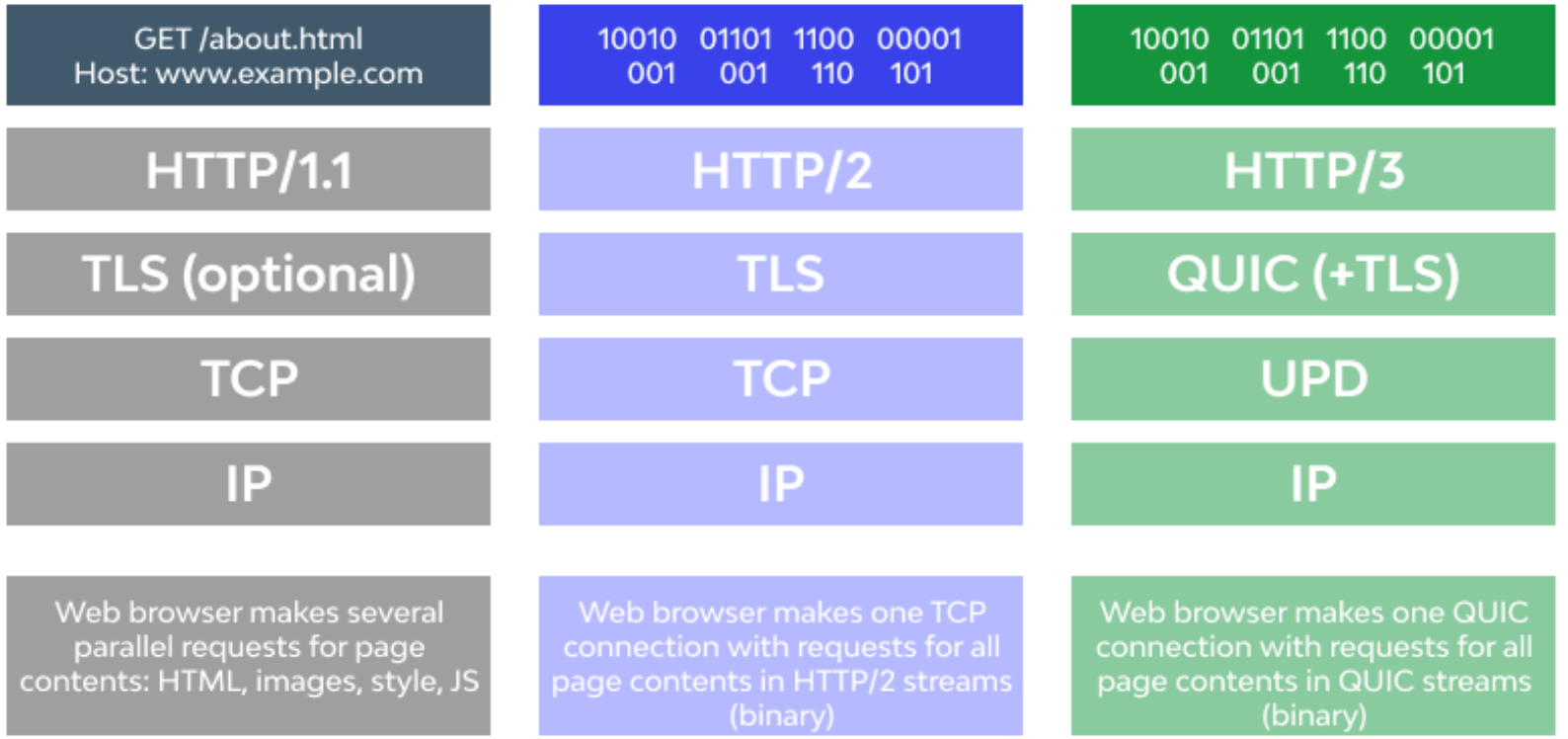


Протоколы модели TCP/IP

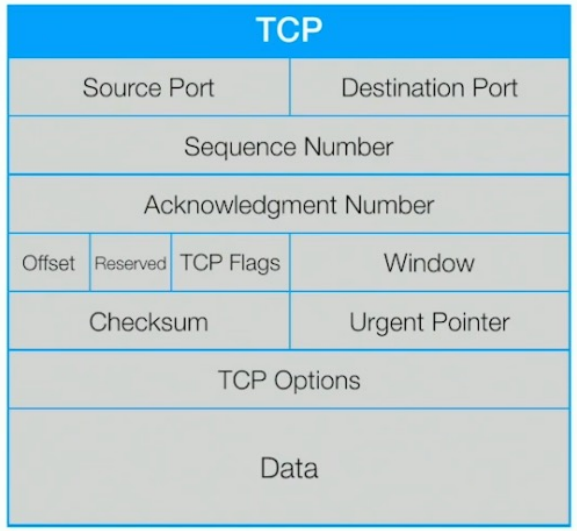


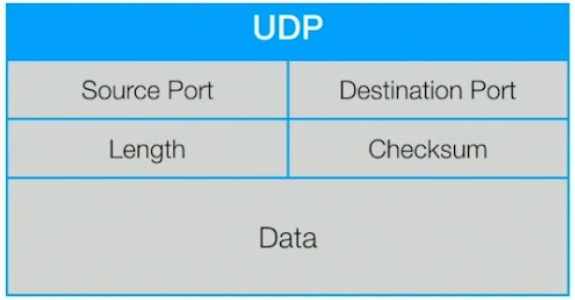
HTTP/2

- Протокол стал бинарным
- Появились потоки Streams в рамках одного TCP соединения для всего содержимого страницы
- Проблема возникает если теряем один пакет одного потока. Часто в беспроводных сетях
- HTTP/2 используется в качестве транспорта для gRPC



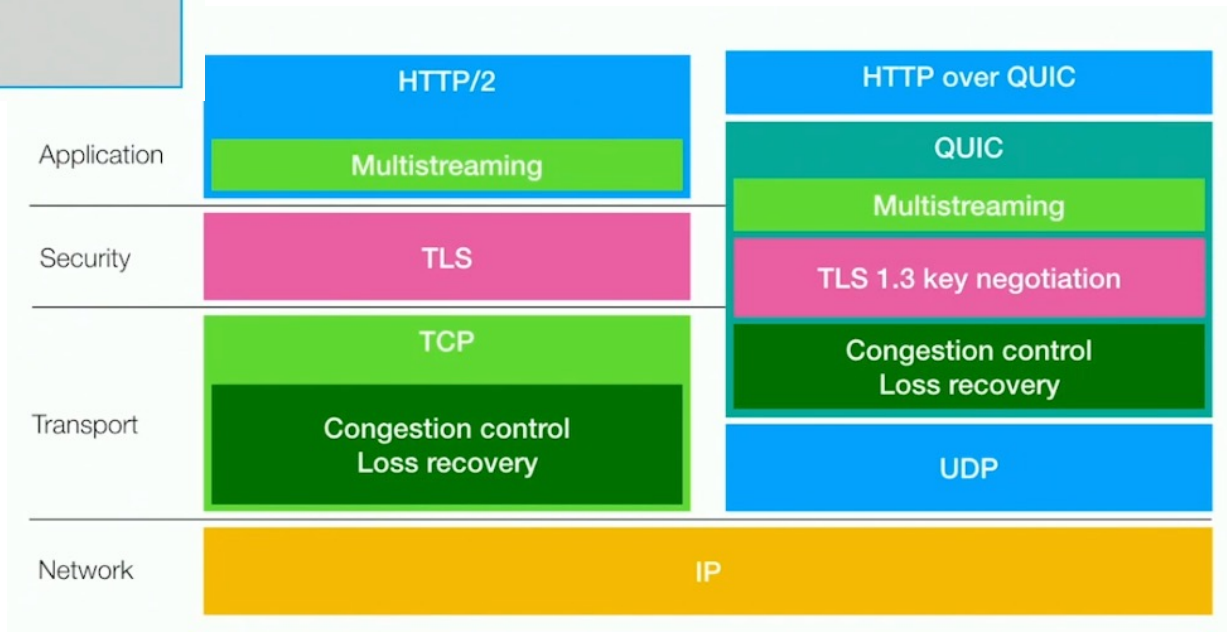
HTTP/3





- Отказываемся от TCP, переходим на UDP
- Заголовки UDP гораздо проще

- QUIC сочетает потоки, TLS, гарантированная доставка



HTTP/2 vs QUIC

- В HTTP/2 все файлы в одном соединении
- В QUIC делаем несколько потоков для разных файлов. Два получим и начнем обрабатывать, третий продолжим ждать