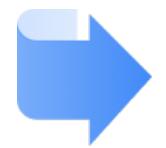


Типовое устройство backend-приложения

О себе



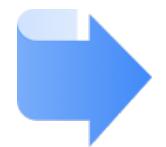
10 лет работал в НИИ



**В 2017 влюбился в Java, решил все изменить и уволился,
чтобы стать разработчиком**



**Работаю в Тинькофф в группе планирования встреч
backend-разработчиком**



Нравится делиться знаниями

HTTP протокол и все, что с ним связано

Лекция 1



В сегодняшней лекции

1

Сравнение сетевых моделей OSI и TCP/IP

2

Описание основных сетевых протоколов

3

Демо (Wireshark)

4

Подробное рассмотрение протокола HTTP

5

Демо (Postman и Chrome)

Модели OSI и TCP/IP

Протокол

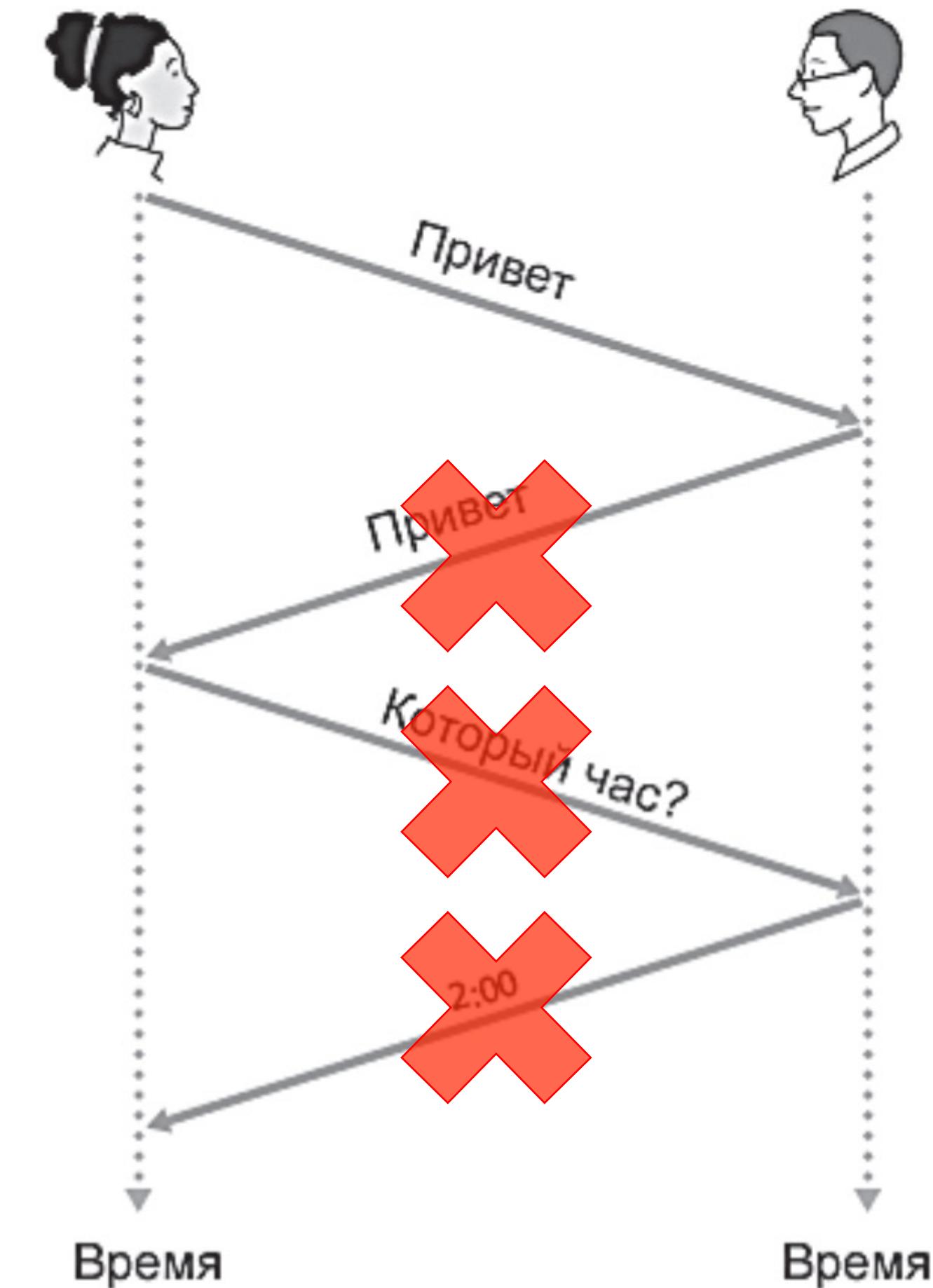
Протоколы нужны для взаимодействия объектов.

В первую очередь для обмена информацией.

Люди постоянно используют различные протоколы.

Пример: узнать сколько времени.

Для успешного обмена информацией стороны должны использовать **одинаковый протокол**.



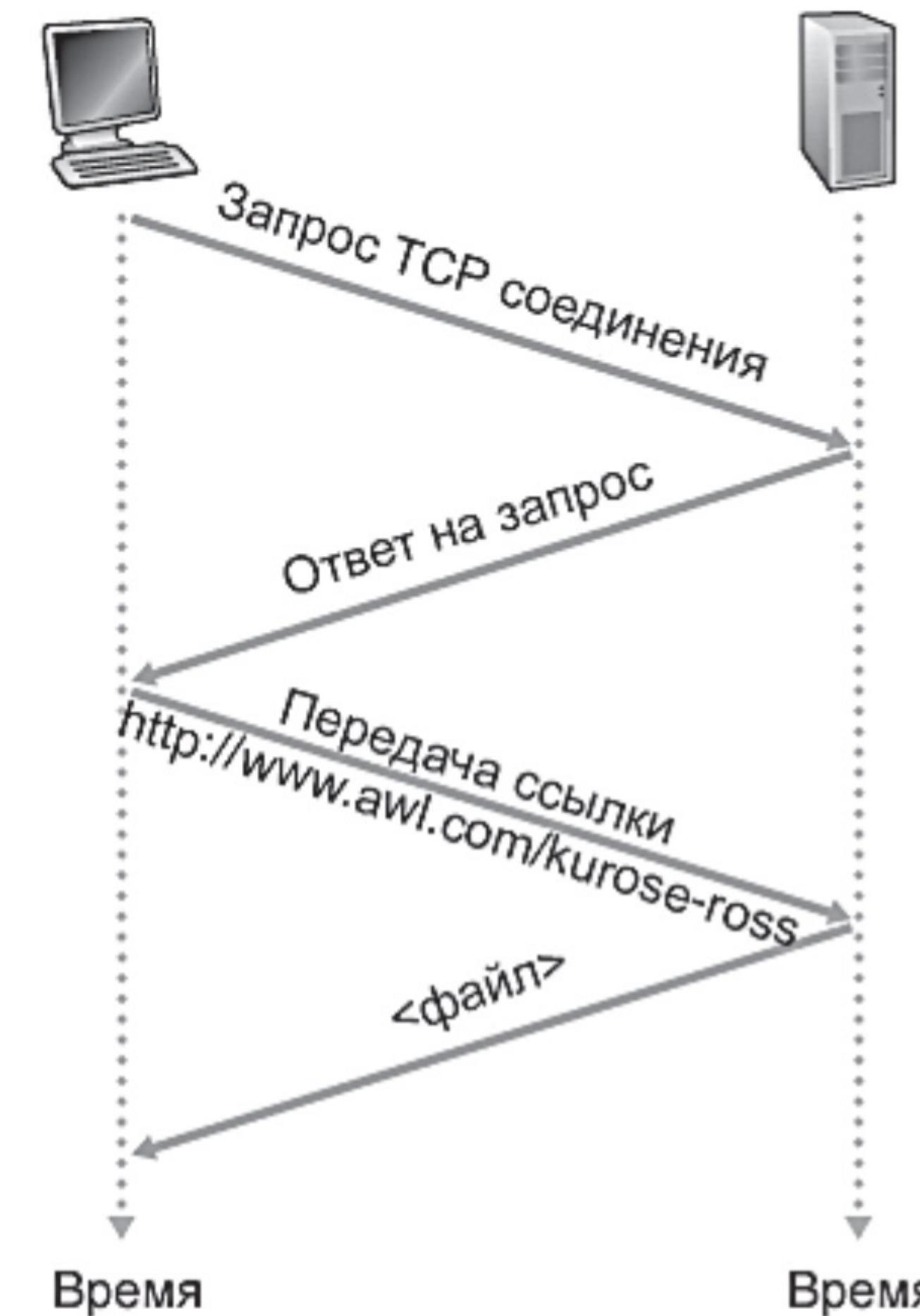
Сетевой протокол

Сетевой протокол - это набор правил, по которым данные передаются по сети.

С помощью сетевых протоколов связываются между собой компьютеры в сети, разные устройства и программы. Они выполняют определенные правила и поэтому понимают друг друга.

Чаще всего сетевые протоколы описывают в каком формате будут передаваться данные, как будет выглядеть процедура их передачи, а также что делать, если возникнут ошибки.

Пример: получение веб-страницы по ссылке.



Сетевая модель

Существует огромное количество различных протоколов, каждый из которых выполняет свою часть работы.

Сетевые протоколы объединяются в **сетевую модель**, которая описывает их и определяет их взаимодействие.

Сетевая модель разделена на уровни. Каждый уровень представляет одну категорию протоколов.

Прикладной	HTTP , SMTP , SNMP , FTP , Telnet , SSH , SCP , SMB , NFS , RTSP
Представления	XDR , AFP , TLS , SSL
Сеансовый	ISO 8327 / CCITT
Транспортный	X.225, RPC , NetBIOS , ASP
Сетевой	TCP , UDP , SCTP , SPX , RTP , ATP , DCCP , GRE IP , PPP , ICMP , IGMP , CLNP , OSPF , RIP , IPX , DDP, ARP , RARP , BGP
Канальный	Ethernet , Token ring , HDLC , X.25 , Frame relay , ISDN , ATM , MPLS
Физический	<u>электрические</u> <u>проводы</u> , радиосвязь , волоконно-оптические провода , Wi-Fi

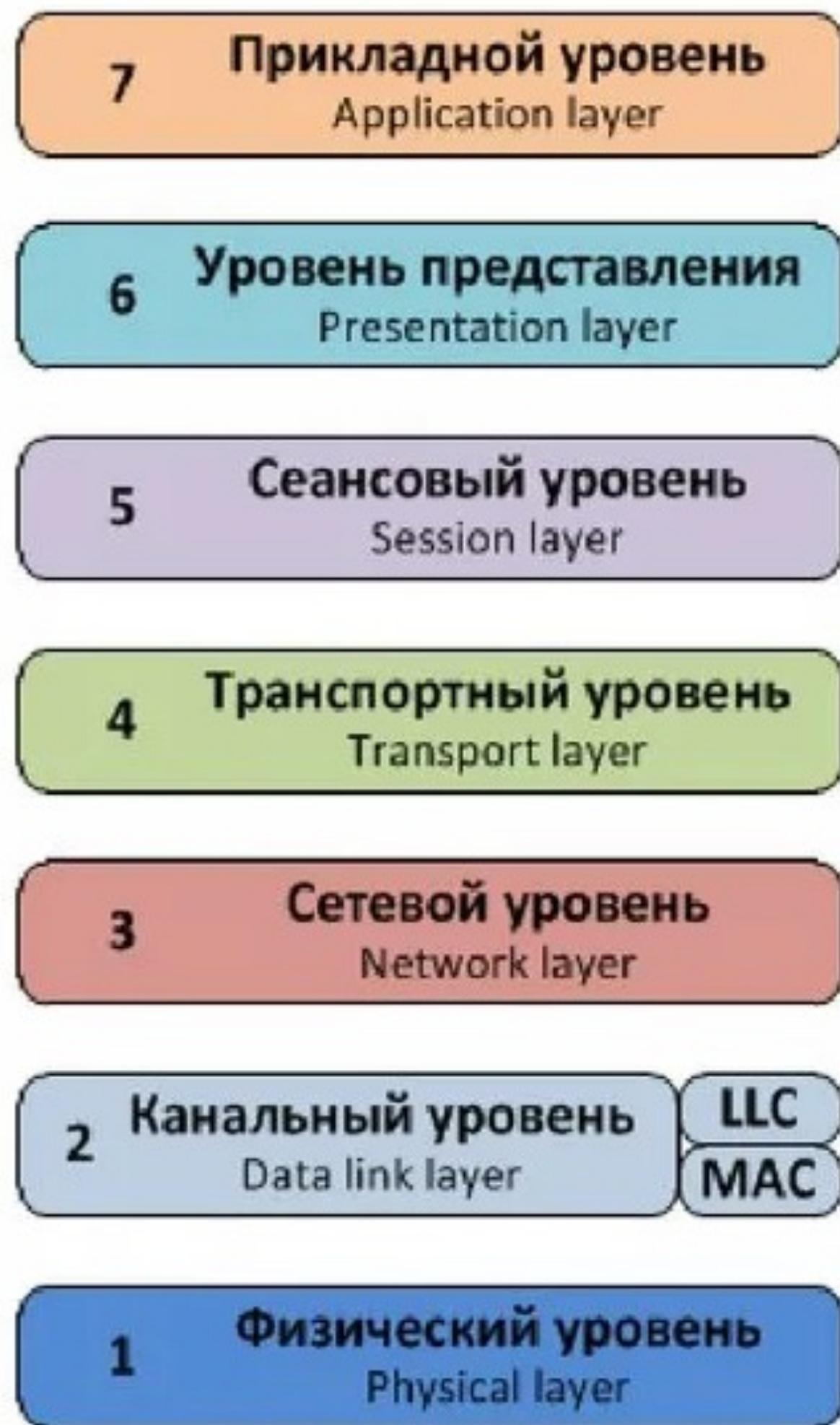
Модель OSI

Модель OSI (Open Systems Interconnection model) – модель взаимодействия открытых систем.

Это концептуальная модель, еще ее называют [эталонной](#).

Она определяет какие уровни должны быть в сети и какие функции должны выполняться на каждом уровне, но при этом не определяет как именно.

Состоит из 7 уровней.



Модель TCP/IP

Модель TCP/IP – сетевая модель передачи данных.

Она описывает как нужно строить сети на основе разных технологий, чтобы в них работал стек протоколов TCP/IP.

Состоит из 4 уровней.

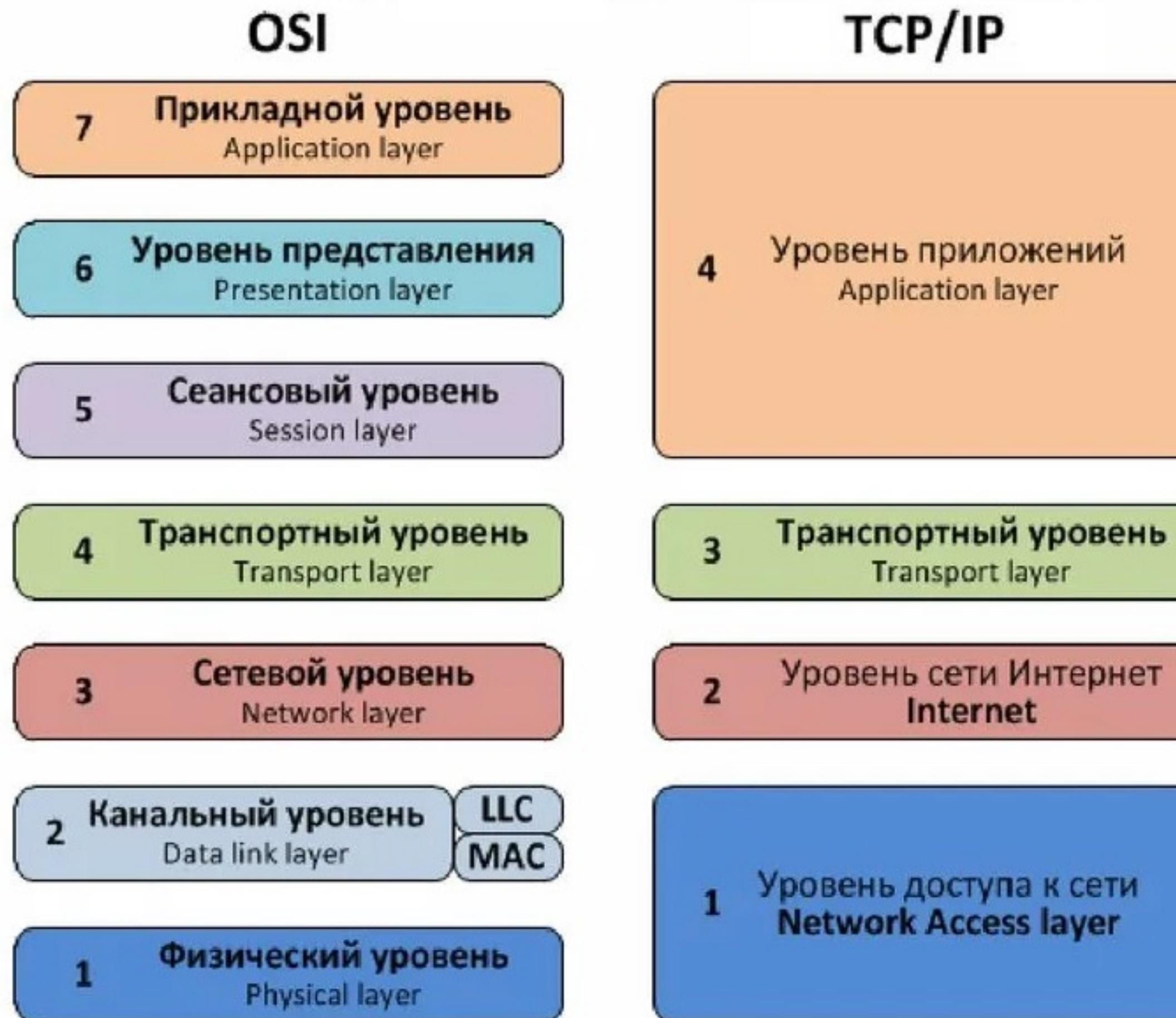
4 Уровень приложений
Application layer

3 Транспортный уровень
Transport layer

2 Уровень сети Интернет
Internet

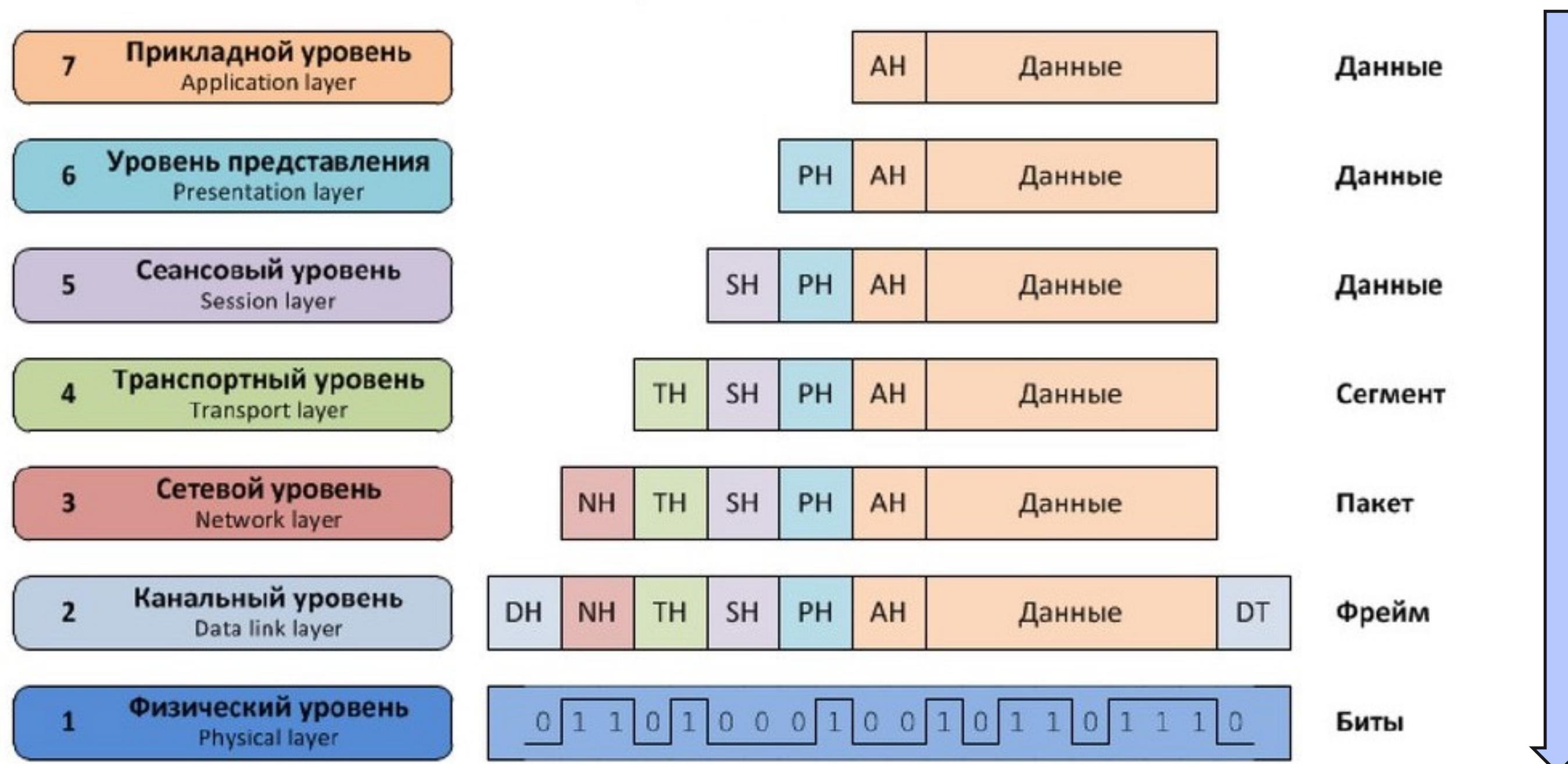
1 Уровень доступа к сети
Network Access layer

Модели OSI и TCP/IP



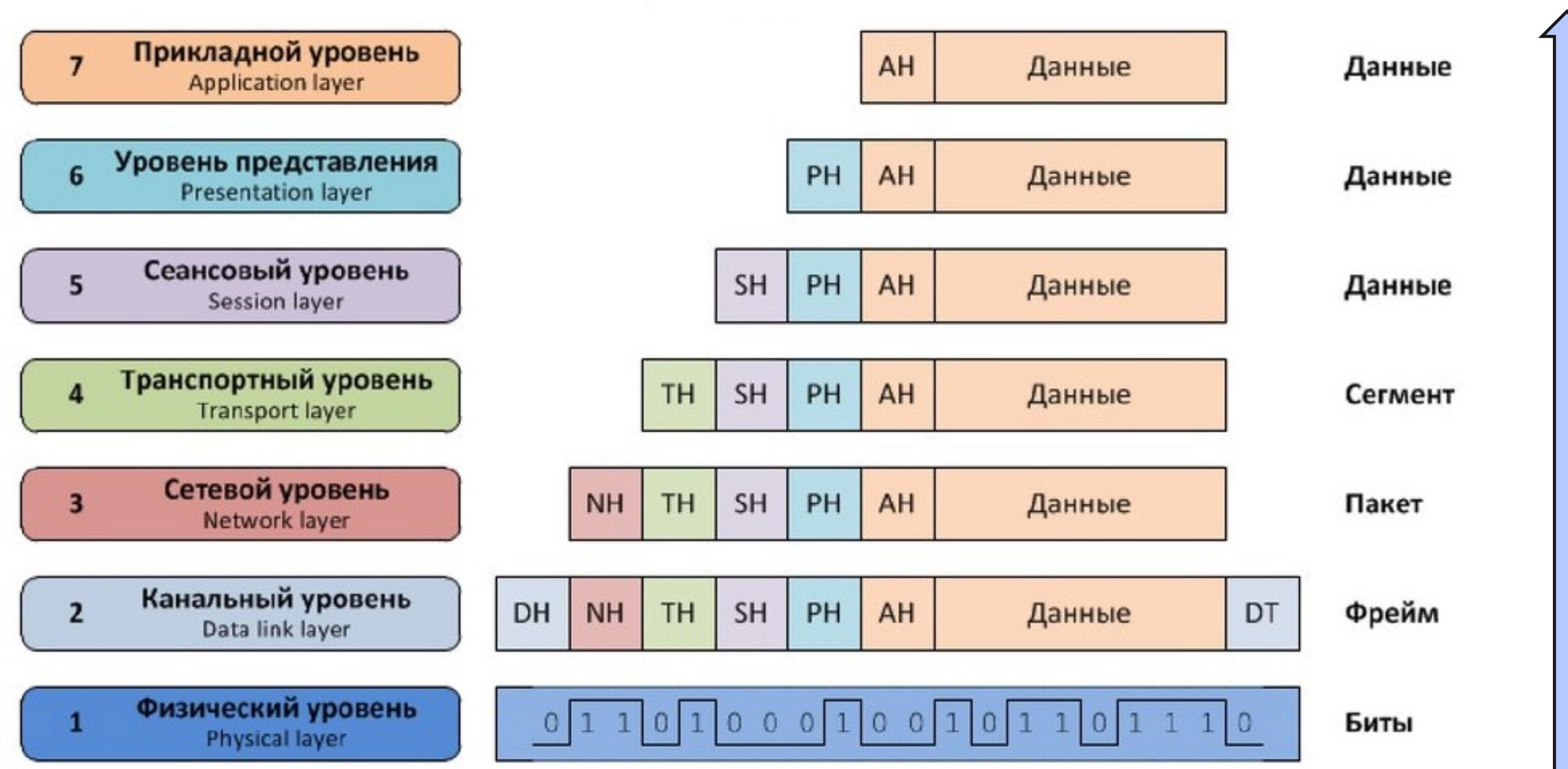
Инкапсуляция данных

Для отправки сообщения по сети на каждом уровне происходит упаковка данных верхнего уровня в собственный контейнер, к которому добавляется служебная информация. Этот процесс называется **инкапсуляция** данных.



Деинкапсуляция данных

При получении сообщения происходит извлечение служебной информации и распаковка данных. Этот процесс называется **деинкапсуляция** данных.

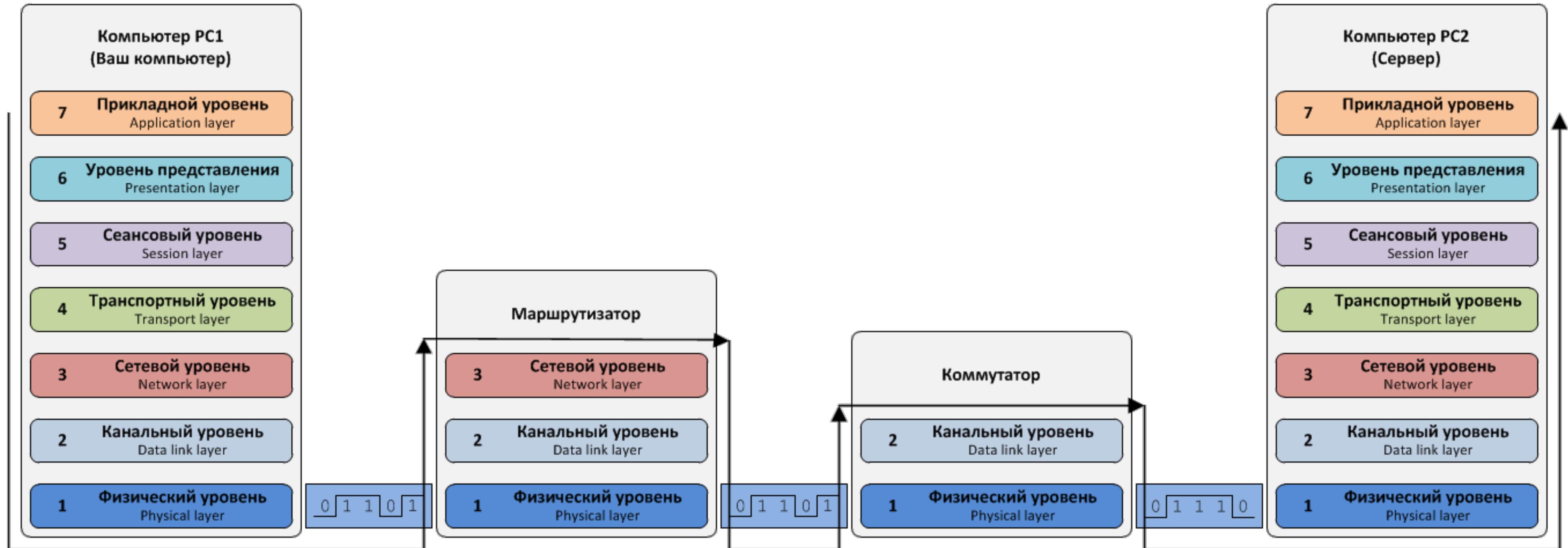


Инкапсуляция и деинкапсуляция данных



Инкапсуляция и deinкапсуляция данных

Во время передачи сообщения происходит многократная распаковка/упаковка служебной информации, которая используется для доставки сообщения получателю.





Вопросы

Прикладной

Находится ближе всего к конечным пользователям.

Представления

Протоколы прикладного уровня должны посыпать сообщения другим приложениям и получать от них сообщения. Некоторые из этих сообщений содержат пользовательские данные, а некоторые являются служебными.

Транспортный

Протоколы:

- **HTTP** (web-приложения)
- **FTP** (передача файлов)
- **SMTP, POP3** (электронная почта)
- **SSH** (соединение с удалённой машиной)
- **DNS** (преобразование символьных имён в IP-адреса)

Сетевой

Канальный

Сетевой

Прикладной

Представления

Сеансовый

Транспортный

Сетевой

Канальный

Физический

На этом уровне происходит преобразование данных.
Например кодирование/декодирование или
сжатие/распаковка.

Данные должны быть представлены в определённом виде, чтобы принимающее приложение смогло их понять.

Стандарты кодирования данных:

- ASCII
- Unicode
- JPEG

Прикладной

Представления

Сеансовый

Транспортный

Сетевой

Канальный

Физический

Отвечает за поддержание сеанса связи, позволяя приложениям взаимодействовать между собой длительное время.

Уровень управляет созданием/завершением сеанса, обменом информацией, синхронизацией задач, определением права на передачу данных и поддержанием сеанса в периоды неактивности приложений.

Протоколы:

- PPTP (Point-to-Point Tunneling Protocol)
- RPC (Remote Procedure Call)
- SIP (Session Initiation Protocol)

Прикладной

Предназначен для доставки данных. При этом неважно, какие данные передаются, он предоставляет сам механизм передачи.

Представления

В заголовок добавляются номера портов отправителя и получателя.

Сеансовый

Номер порта отправителя	Номер порта получателя	Данные
-------------------------	------------------------	--------

Транспортный

Порт необходим для определения какому именно приложению предназначаются данные.

Сетевой

Номер порта – число от **0** до **65535**.

Канальный

Физический

Прикладной

Представления

Сеансовый

Транспортный

Сетевой

Канальный

Физический

Протоколы:

- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)

	TCP	UDP
Единица передачи данных	Сегмент	Датаграмма
Устанавливает соединение	Да	Нет
Гарантия доставки	Да	Нет
Гарантия последовательности	Да	Нет

TLS/SSL

Протокол **TLS (Transport Layer Security)** основан на протоколе **SSL (Secure Sockets Layer)** и работает поверх **TCP**.

Для работы с **UDP** была разработана специальная версия **TLS**, получившая название **DTLS (Datagram Transport Layer Security)**.

TLS обеспечивает:

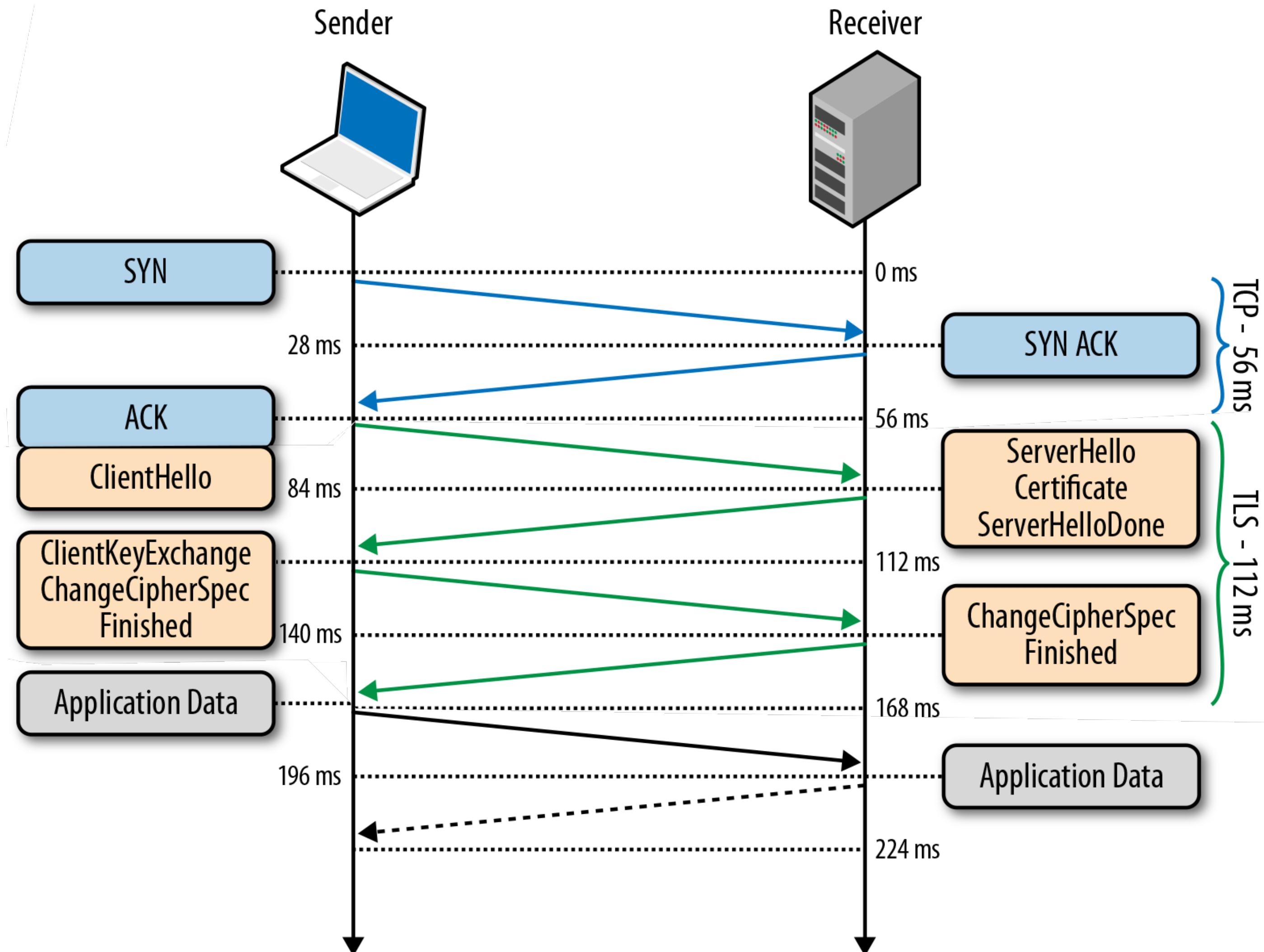
- Скрытие информации, передаваемой от одного компьютера к другому (шифрование)
- Проверку подлинности клиента/сервера
- Обнаружение подмены информации

К протоколам прикладного уровня принято добавлять **s**, если они используют **TLS** при передаче данных. Что означает **secure**.

Пример: **HTTPs (HTTP secure)**

TLS handshake

Для того чтобы установить криптографически безопасный канал данных, узлы соединения должны согласовать используемые методы шифрования и ключи. Этот процесс называется рукопожатие ([handshake](#)).



Симметричное и асимметричное шифрование

TLS использует комбинацию симметричной и асимметричной криптографии, поскольку это обеспечивает хороший компромисс между производительностью и безопасностью при передаче данных.

Симметричное шифрование - это когда используется один и тот же ключ для шифрования и дешифрования данных. Оно работает эффективно и быстро, но требует предварительного обмена ключом между компьютером и сервером, в ходе которого ключ могут [перехватить](#).

Асимметричное шифрование использует два ключа: публичный для шифрования и приватный для дешифровки. Публичный ключ можно свободно распространять, а приватный должен быть хорошо защищён. Асимметричное шифрование [безопаснее](#), но требует больше вычислительных ресурсов и работает медленнее, чем симметричное.

В протоколе TLS симметричное шифрование используют для шифрования непосредственно сообщений, а асимметричное шифрование - во время рукопожатия.

Прикладной

Отвечает за передачу данных между сетями (маршрутизацию).

Представления

Протокол IP (Internet Protocol)

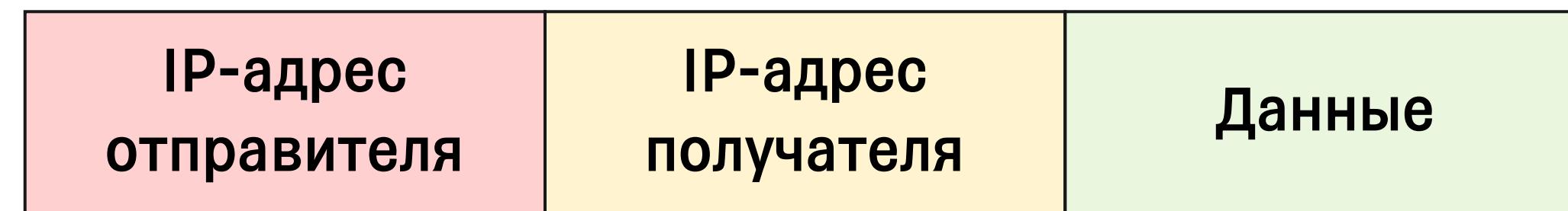
Сеансовый

Единица передачи данных – пакет.

Транспортный

В заголовок добавляются IP-адреса отправителя и получателя.

Сетевой



IP-адрес – это логический сетевой адрес устройства.

Примеры:

- IPv4 (192.168.0.1)
- IPv6 (2001:0db8:85a3:0000:0000:8a2e:0370:7334)

Физический

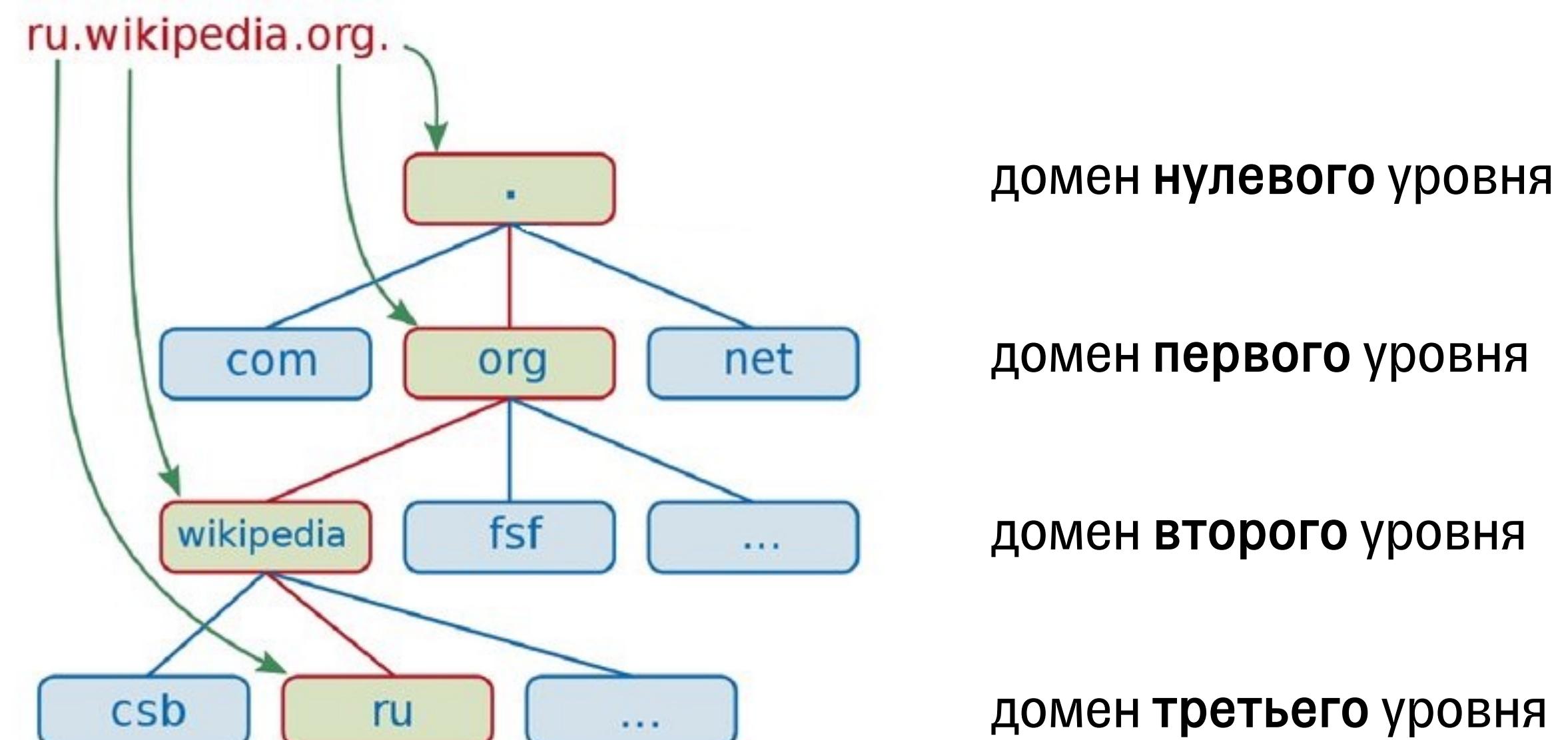
DNS

Для преобразования символьных имен в IP-адреса используется **DNS (Domain Name System)**.

Каждому цифровому IP-адресу присваивается понятное буквенное имя – **домен**.

Например, IP-адресу **142.250.68.36** соответствует домен **google.com**.

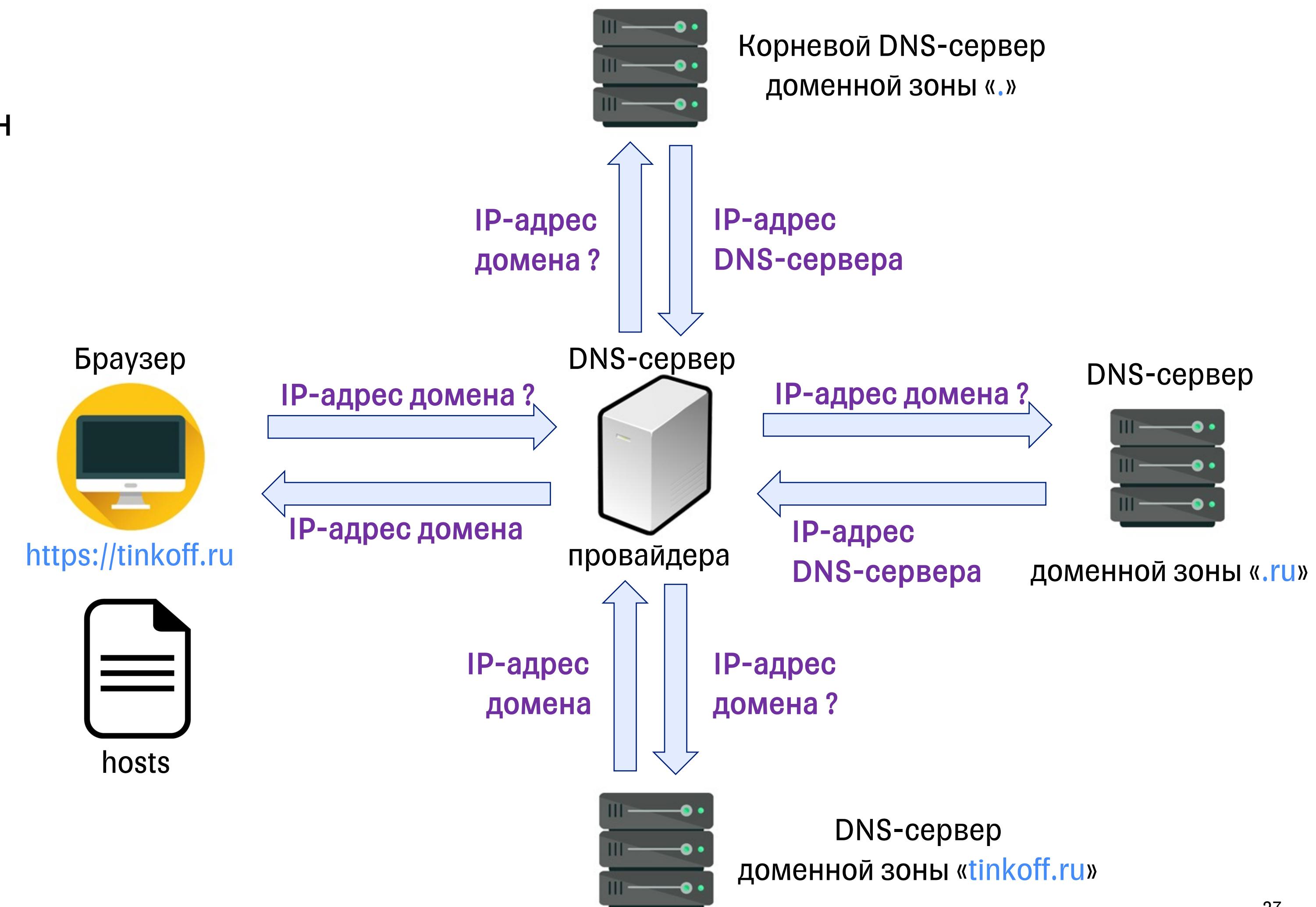
Доменное имя состоит из нескольких частей, разделённых точками. Эти части образуют иерархическую структуру, в которой уровни доменов перечисляются справа налево.



DNS-серверы

Система доменных имён работает благодаря DNS-серверам. Именно они хранят таблицы соответствий вида «имя домена» — «IP-адрес».

Каждый DNS-сервер содержит лишь часть записей, которые относятся к его доменной зоне, а также он содержит адреса DNS-серверов других зон.



Прикладной

Отвечает за передачу данных между устройствами внутри одной сети.

Представления

Протокол Ethernet

Единица передачи данных – фрейм.

Сеансовый

В заголовок добавляются MAC-адреса отправителя и получателя.

Транспортный

MAC-адрес отправителя	MAC-адрес получателя	Данные	КС
--------------------------	-------------------------	--------	----

Сетевой

MAC-адрес – это уникальный идентификатор устройства.

Канальный

Пример: **00:26:57:00:1f:02**

Физический

В концевик фрейма добавляется контрольная сумма. Если при получении контрольная сумма не совпадет, то фрейм будет отброшен.

Прикладной

Выполняет реальную физическую передачу бит данных.

Представления

Определяет электрические и физические характеристики соединения данных. Например, расположение штырей разъема, рабочие напряжения электрического кабеля, спецификации оптоволоконного кабеля и частоту беспроводных устройств.

Транспортный

Протоколы:

- **Bluetooth**
- **Wi-Fi**
- **USB**
- **RJ**

Канальный

Физический



Вопросы

Демо

(Wireshark)

Протокол HTTP

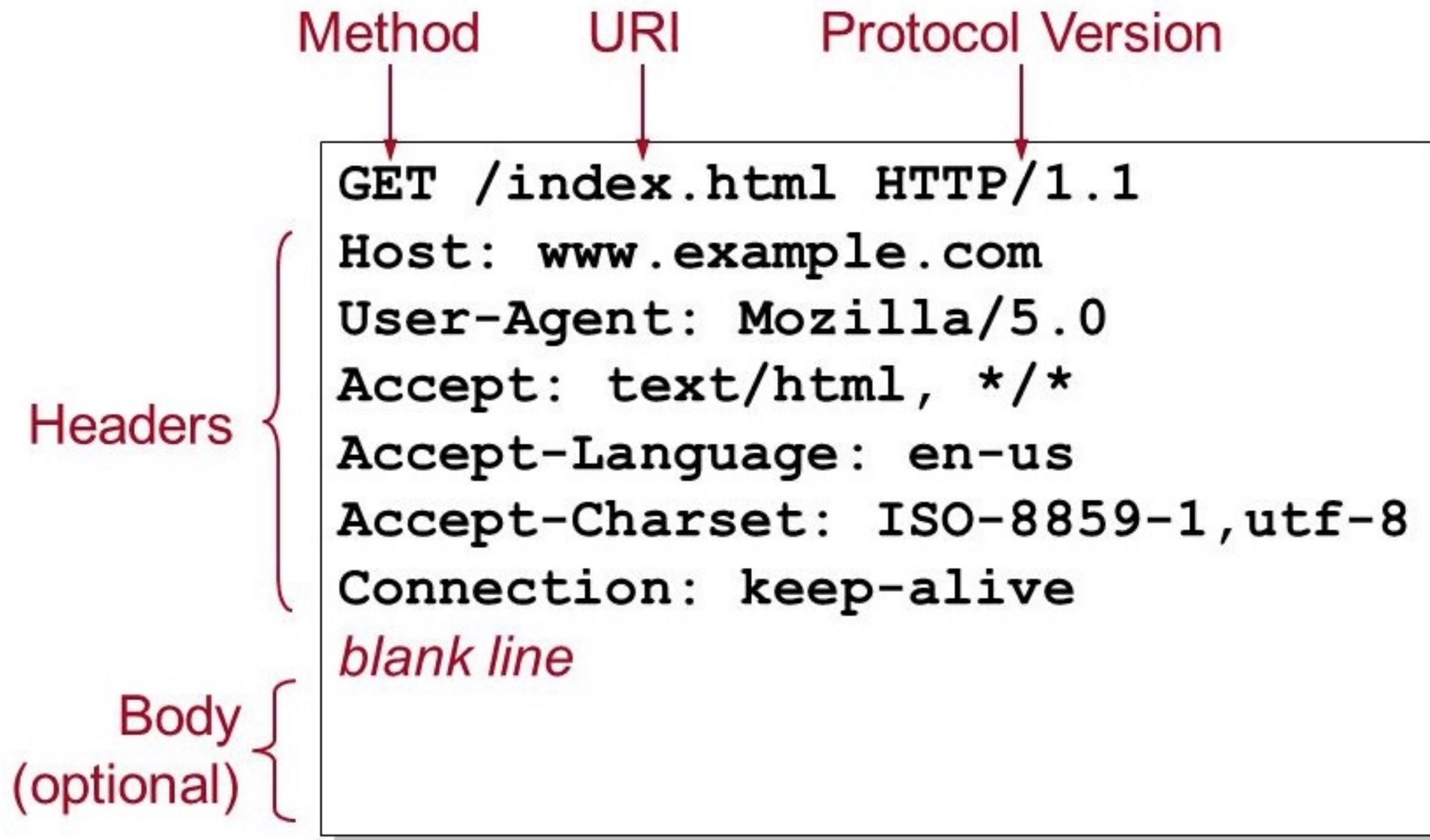
Описание HTTP

Протокол **HTTP** (**H**yper**T**ext **T**ransfer **P**rotocol) — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

Задача, которая традиционно решается с помощью протокола HTTP — обмен данными между пользовательским приложением, осуществляющим доступ к веб-ресурсам (обычно это веб-браузер) и веб-сервером. На данный момент именно благодаря протоколу HTTP обеспечивается работа Всемирной паутины.

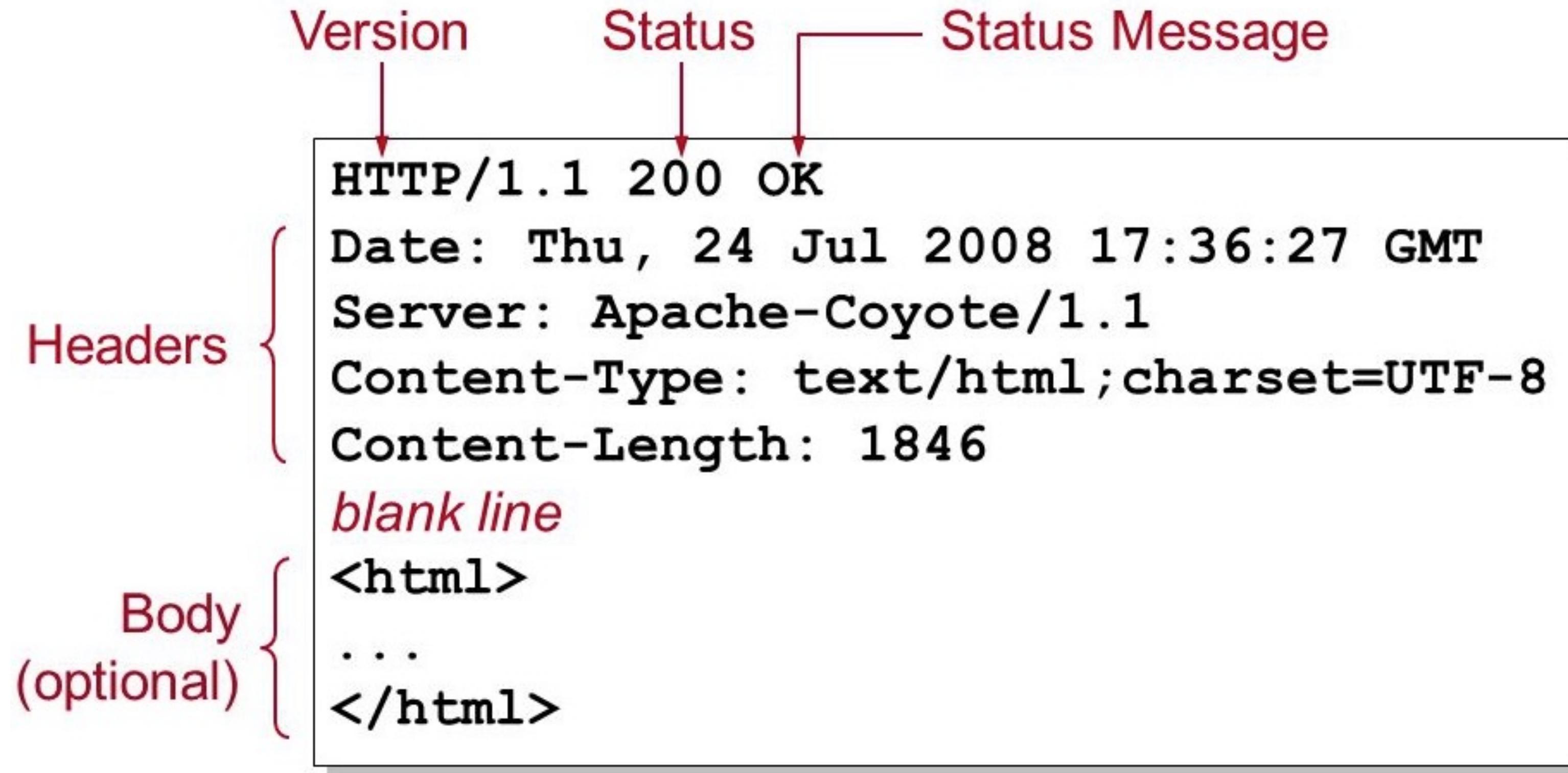
HTTP предполагает использование [клиент-серверной](#) структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту.

Структура HTTP запроса



URI (Uniform Resource Identifier) – унифицированный идентификатор ресурса

Структура HTTP ответа



Методы

Метод	Запрос имеет тело	Ответ имеет тело	Идемпотентный	Описание
OPTIONS	Нет	Да	Да	Определение возможностей веб сервера
GET	Нет	Да	Да	Получение данных от сервера
HEAD	Нет	Нет	Да	Получение данных от сервера (только заголовки)
POST	Да	Да	Нет	Передача данных, запуск процедур, загрузка файлов
PUT	Да	Нет	Да	Создание или изменение ресурсов
PATCH	Да	Да	Нет	То же, что PUT, но применяется к части данных
DELETE	Может	Может	Да	Удаление ресурса

Идемпотентный запрос — это запрос, который при многократном повторении приводит к одному и тому же результату.

Коды состояний

1xx – Информационные

1XX Informational	
100	Continue
101	Switching Protocols
102	Processing

Коды состояний

2xx – Успешные

2XX Success	
200	OK
201	Created
202	Accepted
203	Non-authoritative Information
204	No Content
205	Reset Content
206	Partial Content
207	Multi-Status
208	Already Reported
226	IM Used

Коды состояний

3xx – Ресурс перемещен

3XX Redirection	
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
308	Permanent Redirect

Коды состояний

4xx – Ошибки клиента

4XX Client Error	
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout

Коды состояний

5xx – Ошибки сервера

5XX Server Error	
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported
506	Variant Also Negotiates
507	Insufficient Storage
508	Loop Detected
510	Not Extended
511	Network Authentication Required
599	Network Connect Timeout Error

Основные заголовки запроса

- **Host** - имя домена или IP-адрес, к которому выполняется запрос (обязательный заголовок)
- **Connection** - позволяет удерживать соединение открытым после завершения запроса (keep-alive) для экономии сетевых ресурсов
- **User-Agent** - название и версия клиента
- **Accept** - типы содержимого (MIME Type), которые клиент может обработать
- **Accept-Language** - язык, который клиент ожидает
- **Accept-Encoding** - список способов кодирования
- **Accept-Charset** - список кодировок. Например utf-8

```
Accept:text/html,application/xhtml+xml  
Accept-Encoding:gzip, deflate, br  
Accept-Language:ru-RU,ru  
Connection:keep-alive  
Host:webhook.site  
User-Agent:Chrome/120.0.0.0 Safari/537.36
```

Основные заголовки ответа

- **Content-Type** – тип содержимого в body и кодировка
- **Content-Length** - размер body
- **Content-Language** - Язык контента в body
- **Content-Encoding** - список способов кодирования, который находится в body
- **Server** - программное обеспечение сервера, отправившего ответ

Content-Encoding:gzip

Content-Length:345

Content-Type:text/html; charset=UTF-8

Server:nginx

HTTP cookie

HTTP cookie (web cookie, куки браузера) - это небольшой фрагмент данных, который сервер отправляет браузеру пользователя.

Браузер может сохранить этот фрагмент у себя и отправлять на сервер с каждым последующим запросом. Это позволяет узнать, от одного ли клиента пришли несколько запросов.

Поскольку HTTP-протокол сам по себе [не имеет состояния](#), с помощью кук можно сохранить любую информацию о состоянии.

Куки часто используются для:

- Управления сеансом (логины, корзины для виртуальных покупок)
- Персонализации (пользовательские предпочтения)
- Трекинга (отслеживания поведения пользователей)

HTTP cookie

Получив HTTP-запрос, вместе с ответом сервер может отправить заголовок **Set-Cookie**.

HTTP/1.0 200 OK

Content-type: text/html

Set-Cookie: JSESSIONID=94F71DB44BDC3C9C41E0931845DA89DD; Path=/; HttpOnly

Куки обычно запоминаются браузером и посылаются в HTTP-заголовке **Cookie** с каждым новым запросом к одному и тому же серверу.

GET /sample_page.html HTTP/1.1

Host: www.example.org

Cookie: JSESSIONID=94F71DB44BDC3C9C41E0931845DA89DD

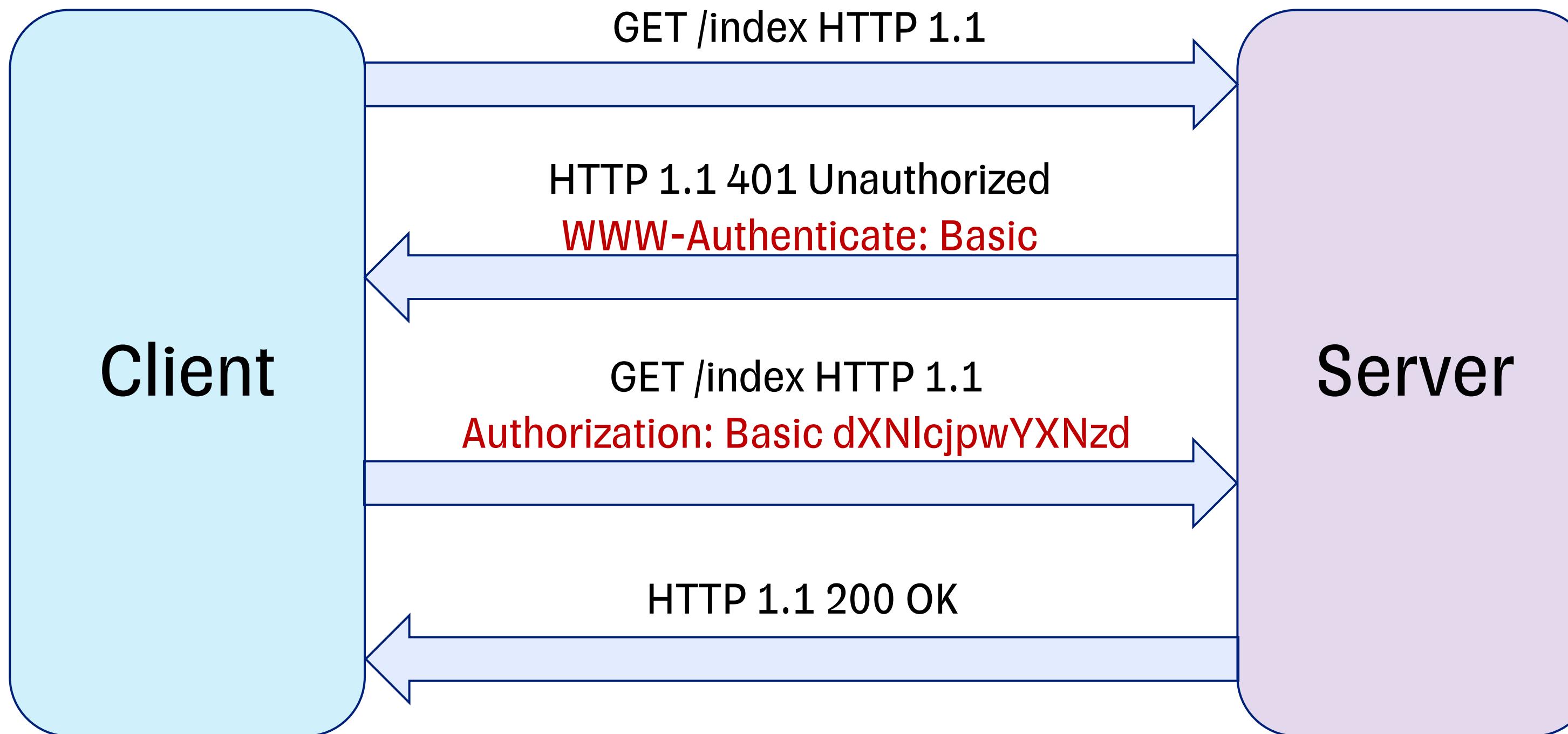
Идентификация, Аутентификация, Авторизация

- **Идентификация** – процедура, в результате выполнения которой для субъекта выявляется его уникальный признак, однозначно определяющий его в информационной системе.
- **Аутентификация** – процедура проверки подлинности, например, проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.
- **Авторизация** – предоставление определенному лицу прав на выполнение определенных действий.



HTTP basic authentication

Basic – наиболее простая схема, при которой username и password пользователя передаются в заголовке Authorization в незашифрованном виде ([base64-encoded](#)).



Сравнение HTTP/1.1 и HTTP/2

	HTTP/1.1	HTTP/2
Когда появился	1999 г.	2015 г.
Формат передачи данных	Текст	Бинарные данные
Загрузка ресурсов	Последовательная	Параллельная
Приоритизация данных	Отсутствует	Есть
Server Push	Отсутствует	Есть
Обязательное использование TLS	Нет	Да

В целом HTTP/2 кардинально изменил протокол и повысил его производительность и безопасность. Использование HTTP/2 может ускорить загрузку страницы на 30-50% по сравнению с HTTP/1.1.



Вопросы

Демо

(Postman и Chrome)

Дополнительные материалы

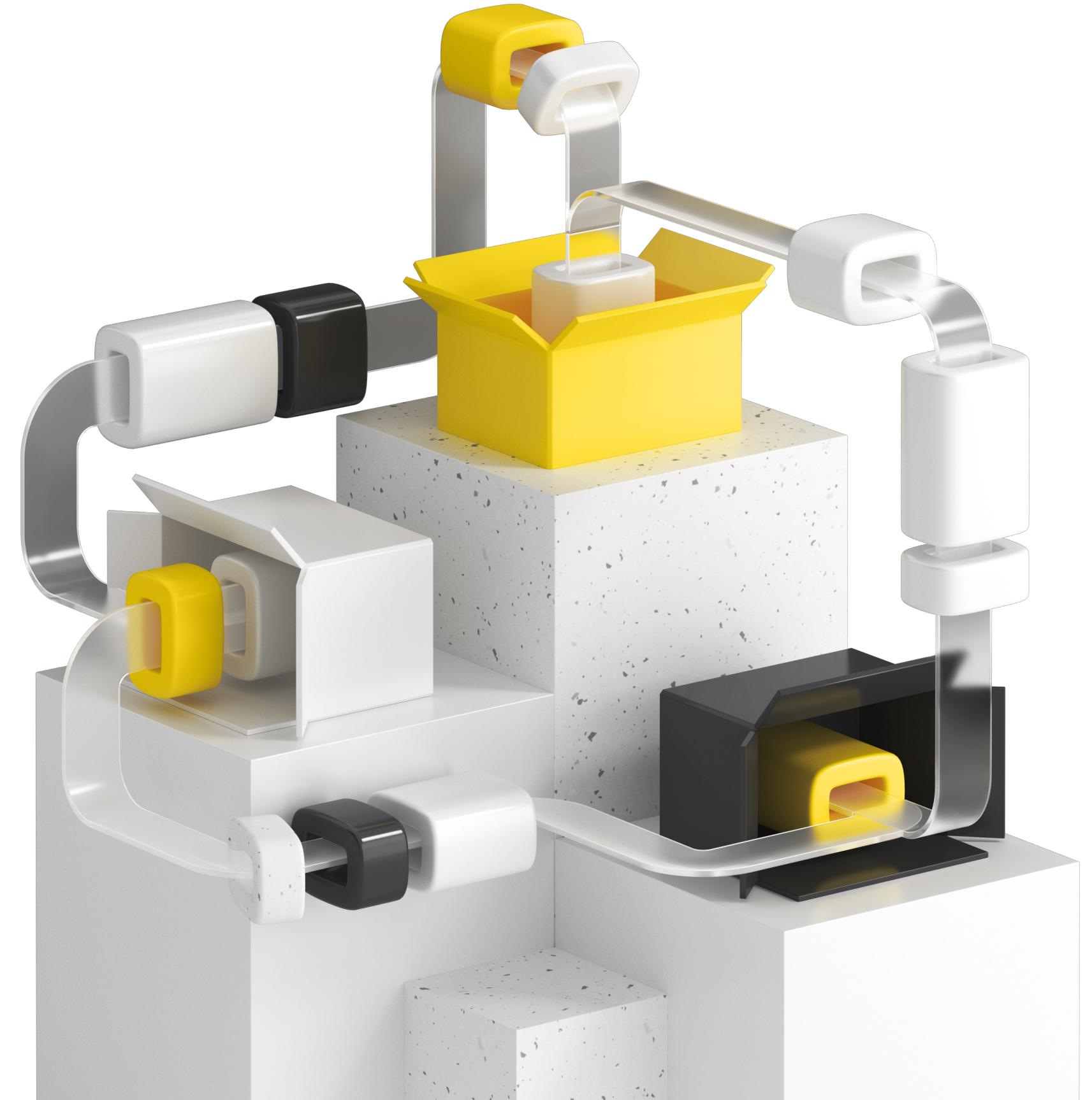
- Книга Б. Кришнамурти, Дж. Рексфорд - «Web-протоколы. Теория и практика»
- Книга Д. Куроуз, К. Росс - «Компьютерные сети. Нисходящий подход»
- Книга Э. Таненбаум, Д. Уэзеролл - «Компьютерные сети»
- Статья https://1cloud.ru/blog/www_first_liks- История World Wide Web Часть I
- Статья https://1cloud.ru/blog/www_standarts_up - История World Wide Web Часть II
- Статья https://1cloud.ru/blog/www_history_finish - История World Wide Web Часть III
- Статья <https://selectel.ru/blog/http-request/> - HTTP-запросы: структура, методы, строка статуса и коды состояния
- Статья <https://habr.com/ru/companies/dataart/articles/262817/> - Обзор способов и протоколов аутентификации в веб-приложениях
- Статья <https://habr.com/ru/companies/avito/articles/710678/> - Ультимативный гайд по HTTP. HTTP/1.1 vs HTTP/2

Обратная связь



Жизненный цикл (конвейер) запроса

В следующей лекции



**ТИНЬКОФФ
ОБРАЗОВАНИЕ**

Спасибо!

