ОНЛАЙН-ОБРАЗОВАНИЕ

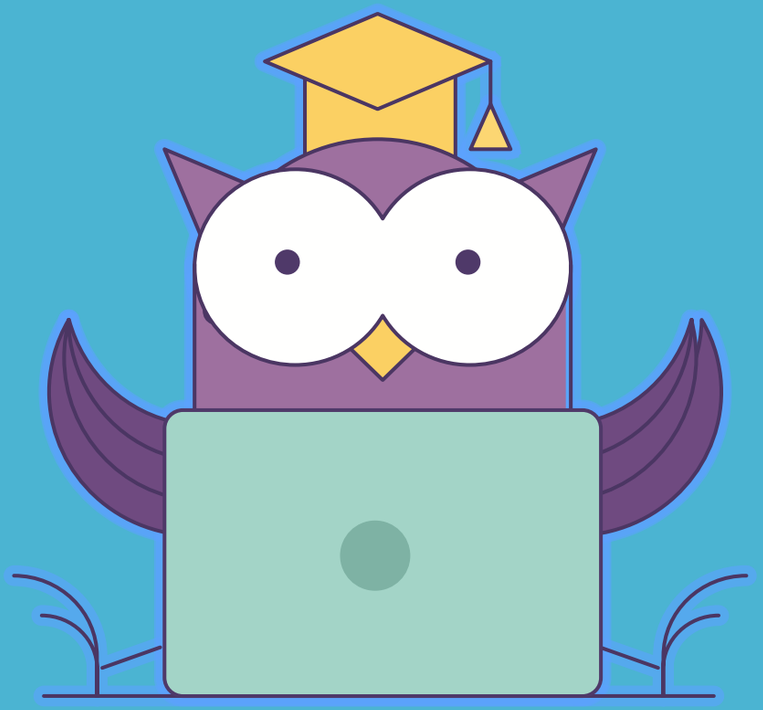
OTUS

Низкоуровневые протоколы

TCP, UDP, DNS

Артем Желтак

Как меня слышно и видно?



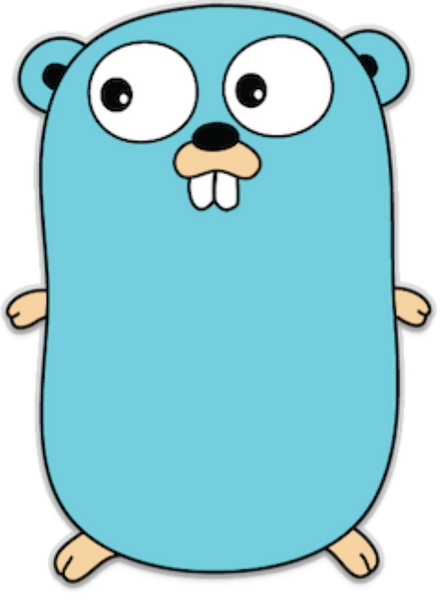
> Напишите в чат

+ если все хорошо

– если есть проблемы со звуком или с видео

Цель занятия

- Изучить что такое контекст
- Изучить особенности протоколов TCP и UDP
- Изучить стандартные типы Conn и Dialer
- Узнать о типичных сетевых проблемах
- Научиться обеспечивать тайм-ауты
- Научиться отлаживать сетевые проблемы



Контекст

Теперь это часть стандартной библиотеки

```
import "context"
```

Имеет природу матрешки: контексты вкладываются друг в друга

Виды контекстов

```
func Background() Context
func TODO() Context
func WithCancel(parent Context) (ctx Context, cancel CancelFunc)
func WithDeadline(parent Context, deadline time.Time) (Context, CancelFunc)
func WithTimeout(parent Context, timeout time.Duration) (Context, CancelFunc)
func WithValue(parent Context, key interface{}, val interface{}) Context
```

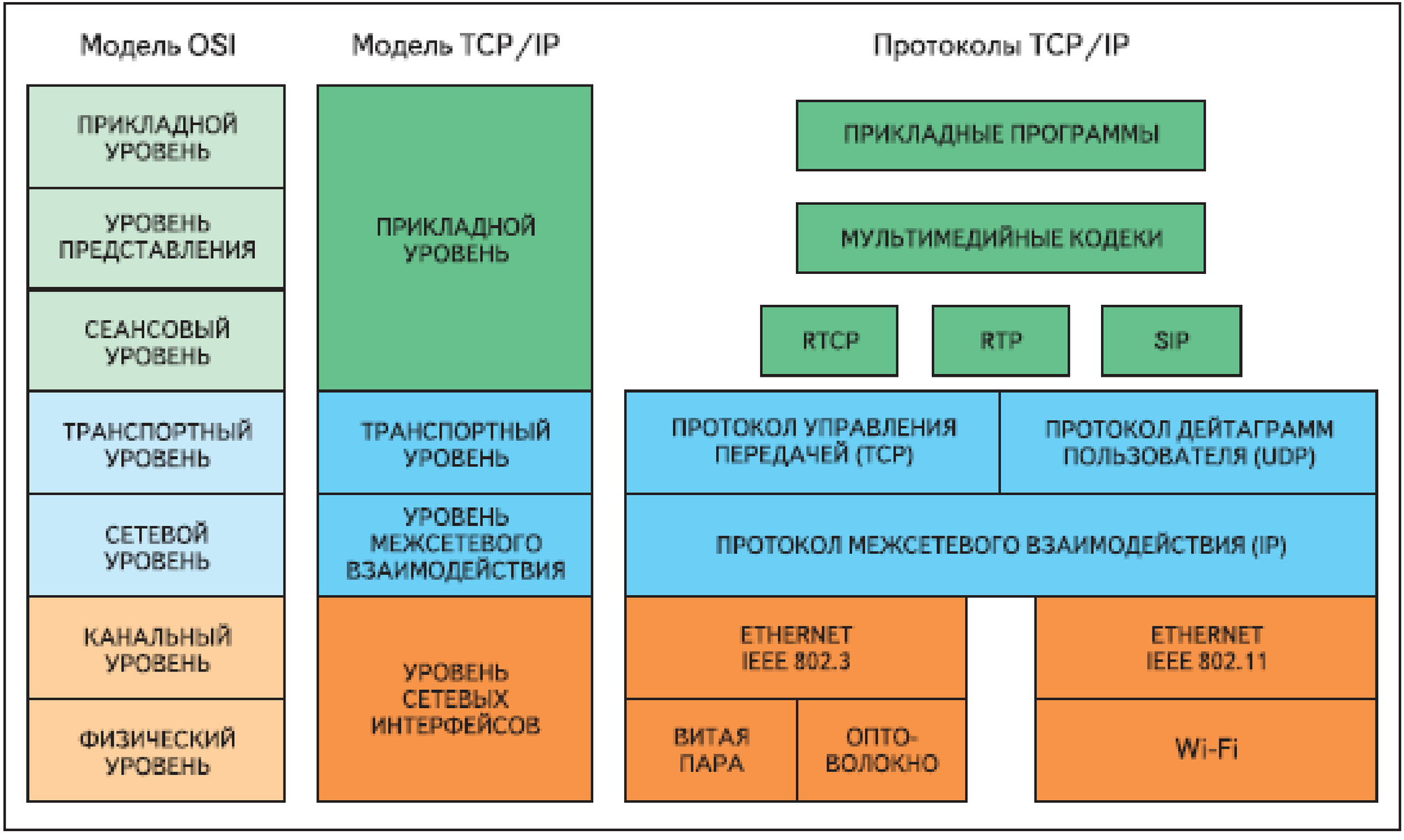
Практика

Пишем эмулятор долгих операций

Сетевая модель OSI

Модель OSI		
Тип данных	**Уровень (layer)**	**Функции**
Данные	7. Прикладной (application)	Доступ к сетевым службам
	6. Уровень представления (presentation)	Представление и шифрование данных
	5. Сеансовый (session)	Управление сеансом связи
Сегменты	4. Транспортный (transport)	Прямая связь между конечными пунктами и надежность
Пакеты (датаграммы)	3. Сетевой (network)	Определение маршрута и логическая адресация
Кадры	2. Канальный (data link)	Физическая адресация
Биты	1. Физический (physical)	Работа со средой передачи, сигналами и двоичными данными

Сетевая модель OSI



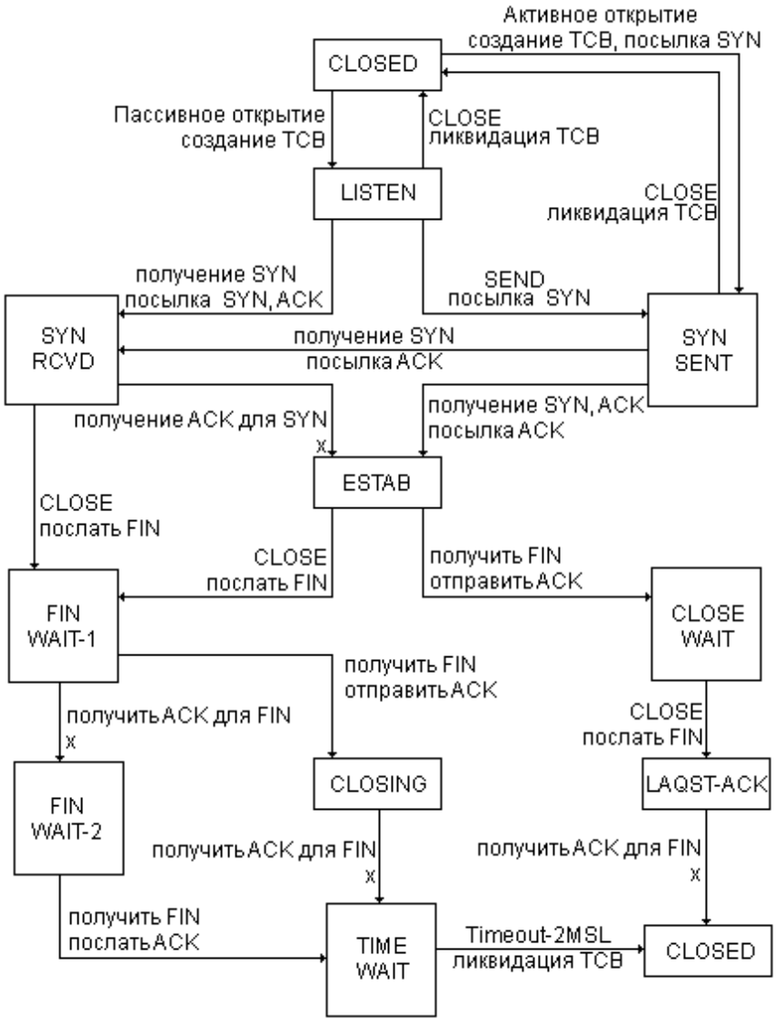
UDP

- Доставка пакета не гарантируется
- Порядок сообщений не гарантируется
- Соединение не устанавливается
- Быстрый
- Подходит для потокового аудио и видео, статистики, игр

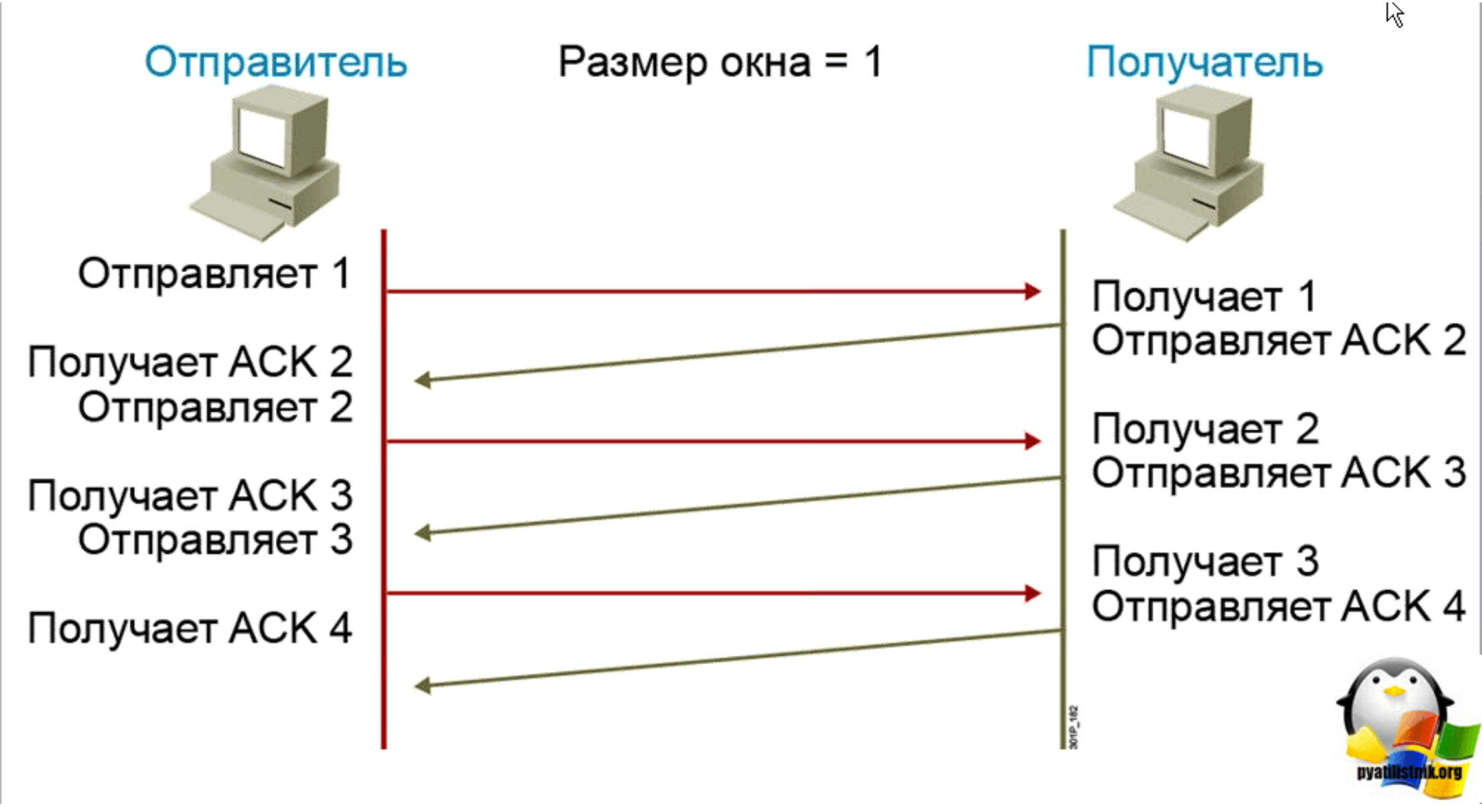
TCP

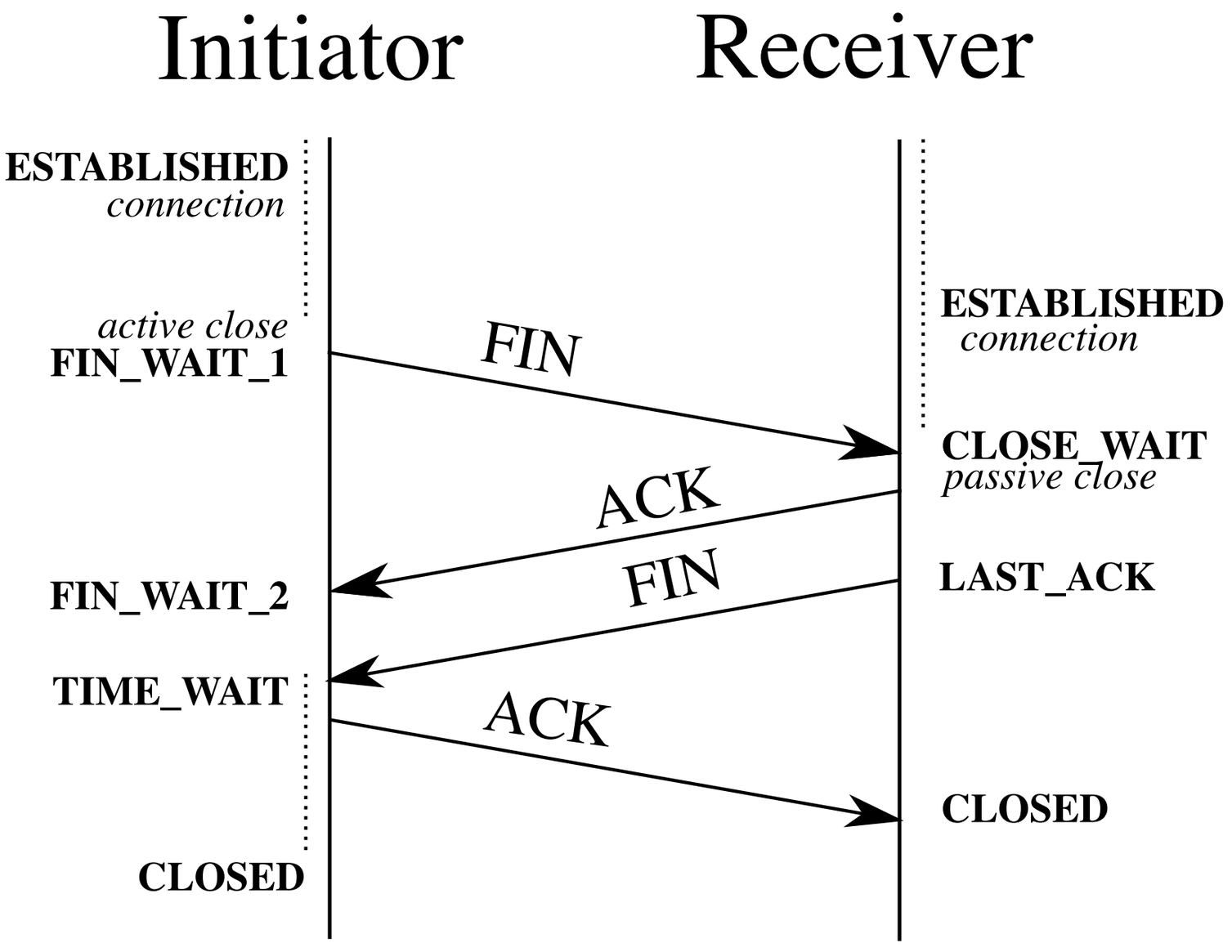
- Доставка пакета гарантируется (или получим ошибку)
- Порядок пакетов гарантируется
- Соединение устанавливается
- Есть overhead
- Подходит для http, электронной почты

TCP - диаграмма состояний



TCP - установка соединения

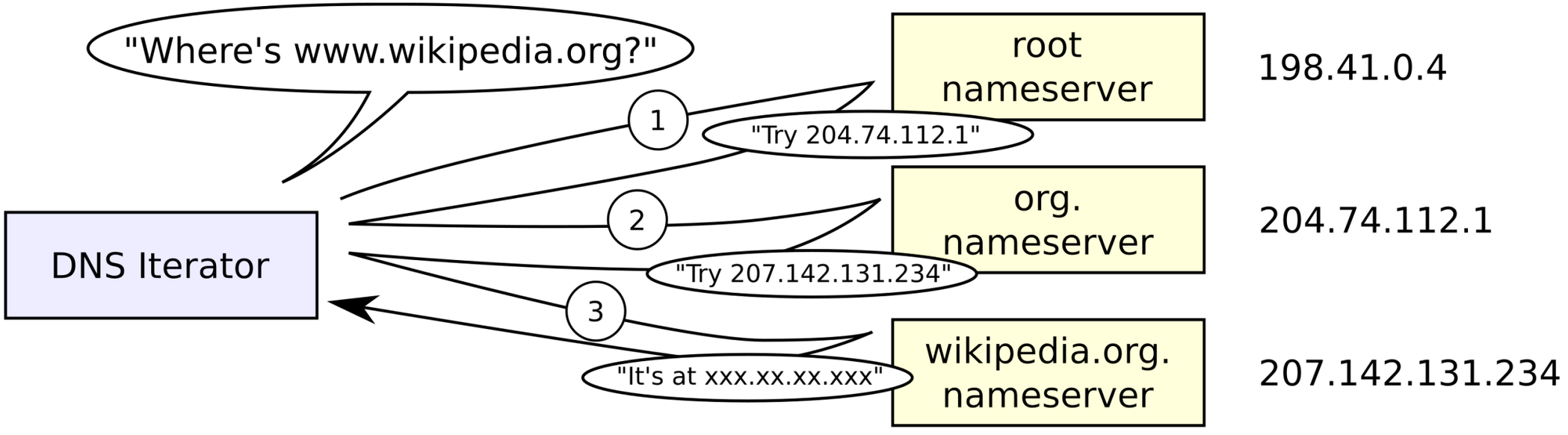



Initiator
Receiver
ESTABLISHED
connection
active close
FIN_WAIT_1
FIN
ESTABLISHED
connection
CLOSE_WAIT
passive close
ACK
FIN
FIN_WAIT_2
LAST_ACK
TIME_WAIT
ACK
CLOSED
CLOSED

DNS

- Служит для получение IP адреса по доменному имени (и не только)
- Работает как поверх UDP, так и поверх TCP
- Имеет рекурсивную природу
- Имеет механизмы для кеширования
- При высоких нагрузках можно использовать /etc/hosts

Рекурсивная природа DNS



Пакет net

В Go за сетевые возможности отвечает пакет net и его подпакеты

Dialer

Тип, задача которого установка соединений
Обладает следующим интерфейсом:

```
func (d *Dialer) Dial(network, address string) (Conn, error)

func (d *Dialer) DialContext(ctx context.Context,
                    network, address string) (Conn, error)
```

Можно использовать стандартные знаения параметров функцией

```
func Dial(network, address string) (Conn, error)
func DialTimeout(network, address string, timeout time.Duration) (Conn, error)
```

Примеры установки соединений

```
Dial("tcp", "golang.org:http")
Dial("tcp", "192.0.2.1:http")
Dial("tcp", "198.51.100.1:80")
Dial("udp", "[2001:db8::1]:domain")
Dial("udp", "[fe80::1%lo0]:53")
Dial("tcp", ":80")
```

Тип Conn

Является абстракцией над поточным сетевым соединением.
Является имплементацией Reader, Writer, Closer.
Это потокобезопасный тип.

Практика

- Пишем чат сервер
- Учимся создавать многопоточные сервера

Типичные сетевые проблемы

- Потеря пакетов
- Недоступность
- Тайм-ауты
- Медленные соединения

Как с ними бороться?

Практика

Изучаем инструменты отладки:

- tcpdump
- wireshark
- lsof
- netstat

На занятии

- Изучили что такое контекст
- Изучили особенности протоколов TCP и UDP
- Изучили стандартные типы Conn и Dialer
- Узнали о типичных сетевых проблемах
- Научились обеспечивать тайм-ауты
- Научились отлаживать сетевые проблемы

Вопросы?

Опрос

Не забудьте заполните заполнить опрос. Ссылка на опрос в чате

Спасибо за внимание!