




Golang Developer. Professional

otus.ru

 Проверить, идет ли запись

Меня хорошо видно && слышно?

 Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема вебинара

Контекст и низкоуровневые сетевые протоколы

Рубаха Юрий



Правила вебинара



Активно
участвуем



Off-topic обсуждаем
в учебной группе

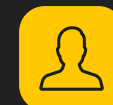


Задаем вопрос
в чат или голосом



Вопросы вижу в чате,
могу ответить не сразу

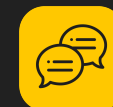
Условные обозначения



Индивидуально



Время, необходимое
на активность



Пишем в чат



Говорим голосом



Документ



Ответьте себе или
задайте вопрос

О чем будем говорить

1. `context.Context` .
2. Сетевые протоколы.
3. Работа с ними в Go.

Что это?

- Что уже знаете?
- Какие идеи исходя из названия?

```
type Context interface {  
    Deadline() (deadline time.Time, ok bool)  
    Done() <-chan struct{}  
    Err() error  
    Value(key any) any  
}
```

<https://pkg.go.dev/context>

<https://go.dev/blog/context>

Пакет context

```
func Background() Context
func TODO() Context
```

```
func WithCancel(parent Context) (ctx Context, cancel CancelFunc)
func WithoutCancel(parent Context) Context
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
```

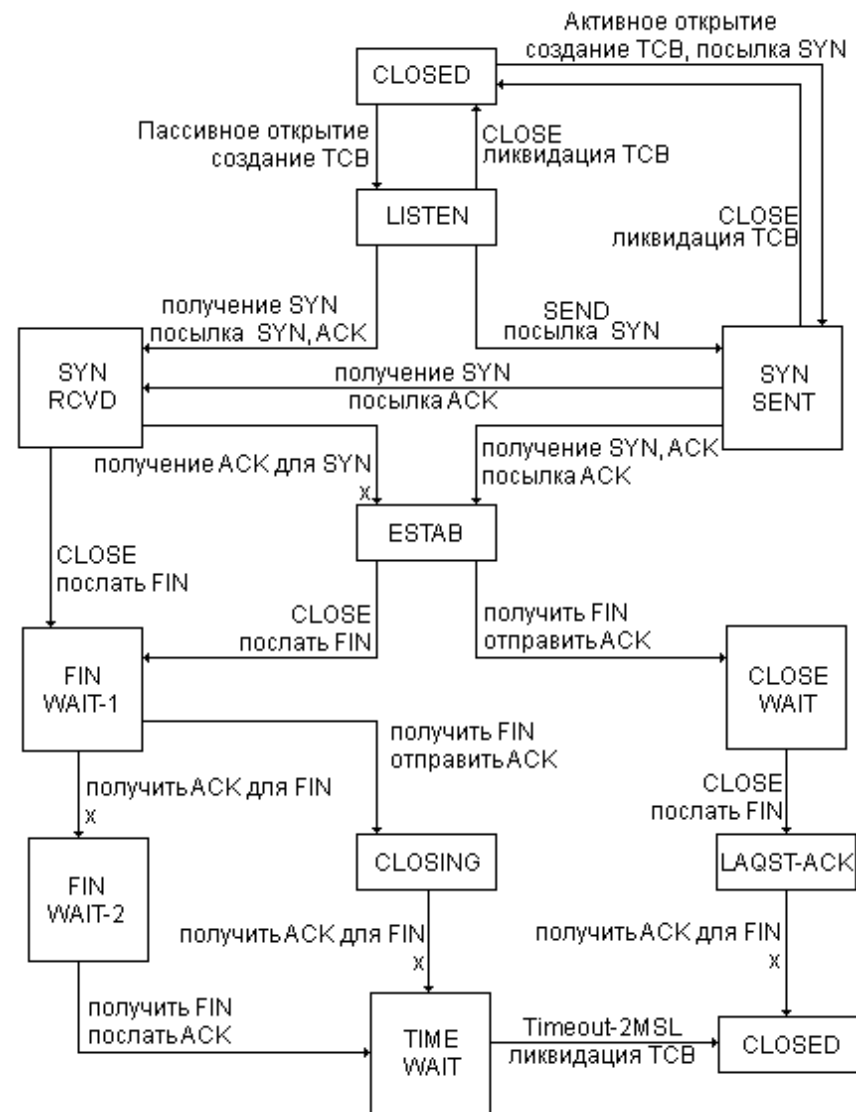
```
func AfterFunc(ctx Context, f func()) (stop func() bool)
```


Пишем эмулятор долгих операций

Какие особенности вы знаете?

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

TCP - диаграмма состояний



TCP - установка соединения



Отправитель



Размер окна = 1

Получатель



Отправляет 1

Получает ACK 2
Отправляет 2

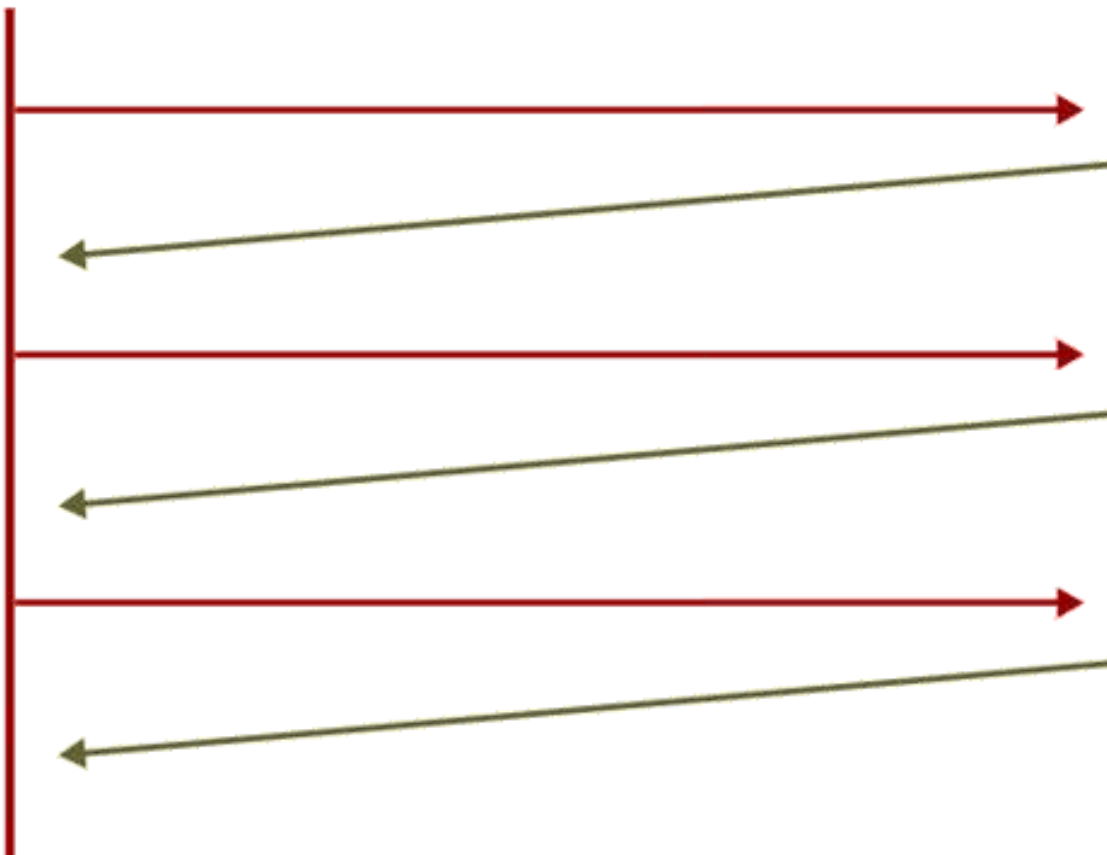
Получает ACK 3
Отправляет 3

Получает ACK 4

Получает 1
Отправляет ACK 2

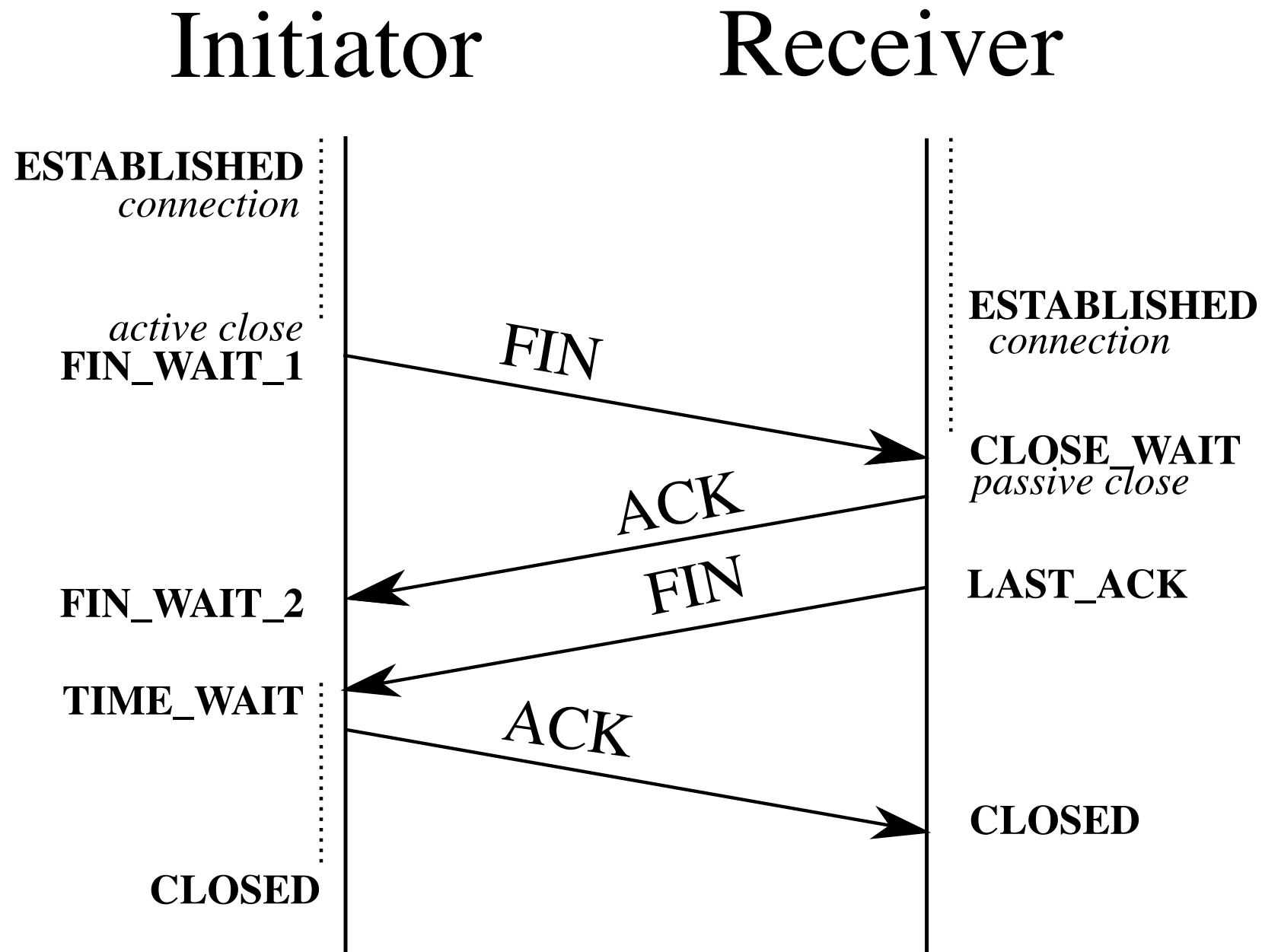
Получает 2
Отправляет ACK 3

Получает 3
Отправляет ACK 4



301P_162





Какие особенности вы знаете?

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

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 значения 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")
```

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

- Пишем чат сервер
- Учимся создавать многопоточные сервера

Типичные сетевые проблемы

- Какие знаете?
- С какими сталкивались? Что делали?

Типичные сетевые проблемы

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

Как с ними быть?

Вопросы?



Ставим “+”,
если вопросы есть



Ставим “-”,
если вопросов нет



**Заполните, пожалуйста,
опрос о занятии
по ссылке в чате**

Следующий вебинар

3 марта

Работа с SQL



Ссылка на вебинар будет в ЛК за 15 минут



Материалы к занятию в ЛК — можно изучать



Обязательный материал обозначен красной лентой



Спасибо за внимание!

Приходите на следующие вебинары

Дмитрий Копылов

Software Engineer at OZON

