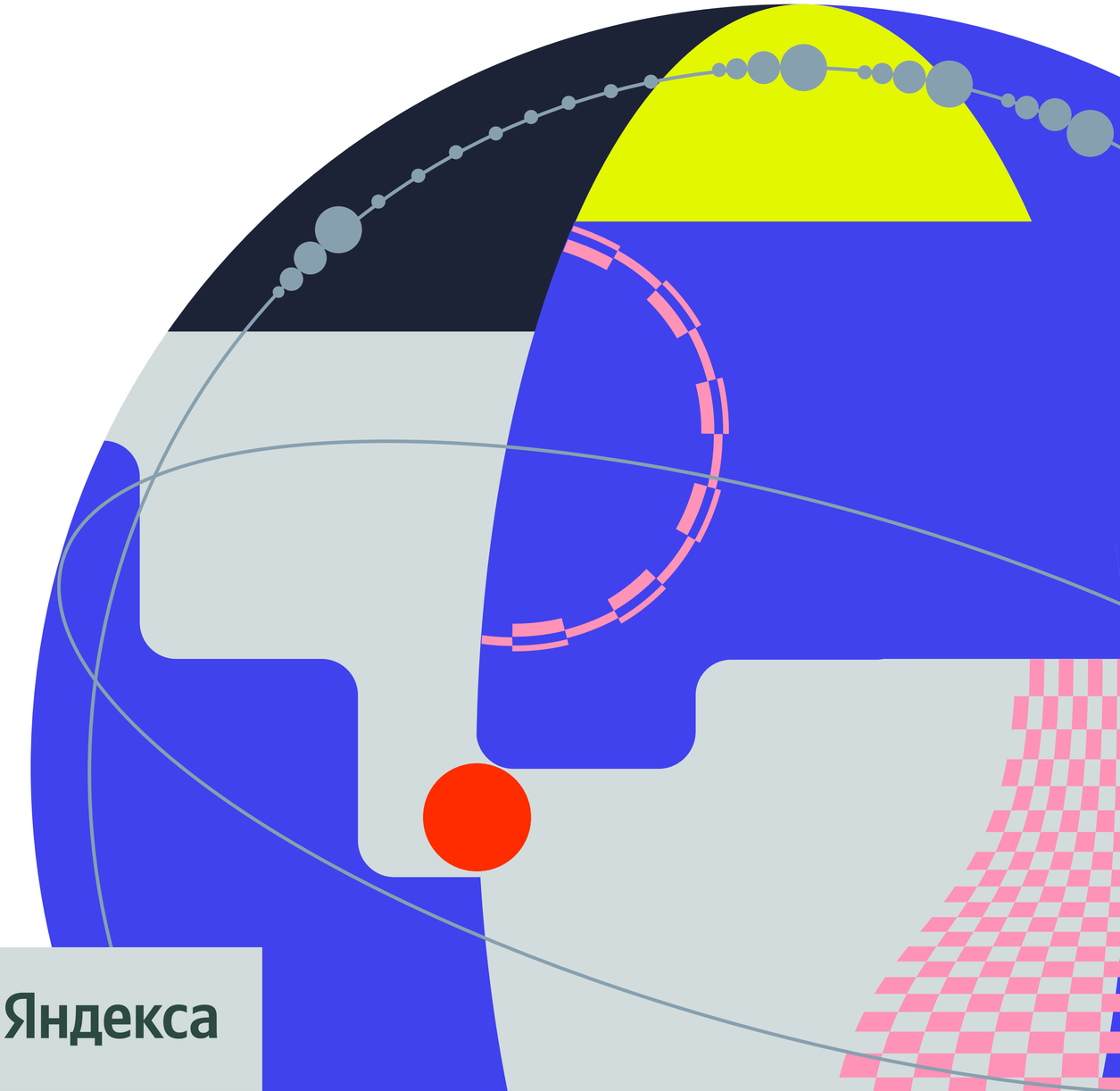
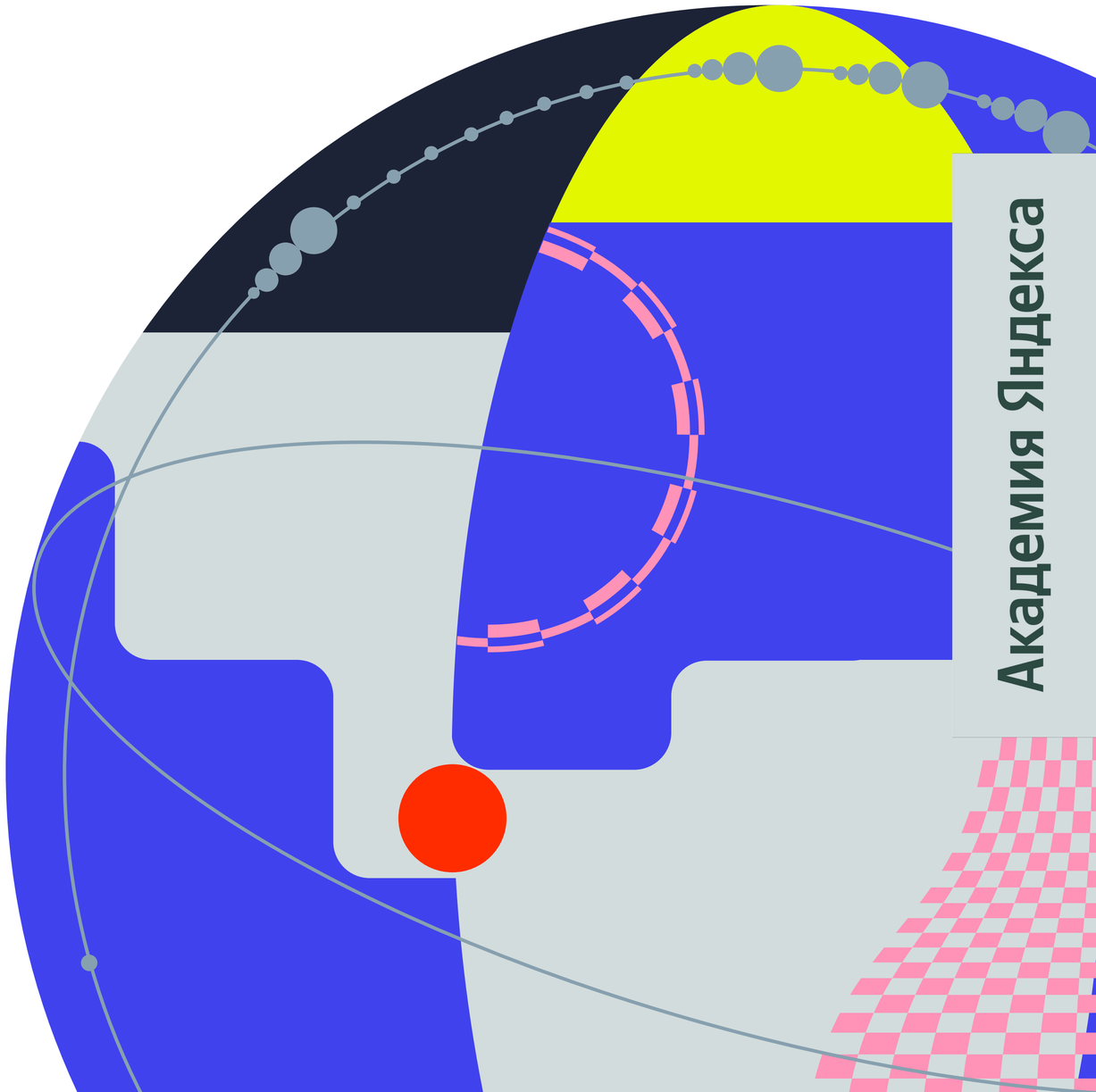
Вместе учимся, решаем сложные задачи, общаемся и строим свою карьеру в IT



Академия Яндекса

Высшая школа программной инженерии МФТИ



Академия Яндекса позволяет школьникам
и студентам освоить востребованные ИТ-профессии по программам,
разработанным экспертами компании

Введение в userver

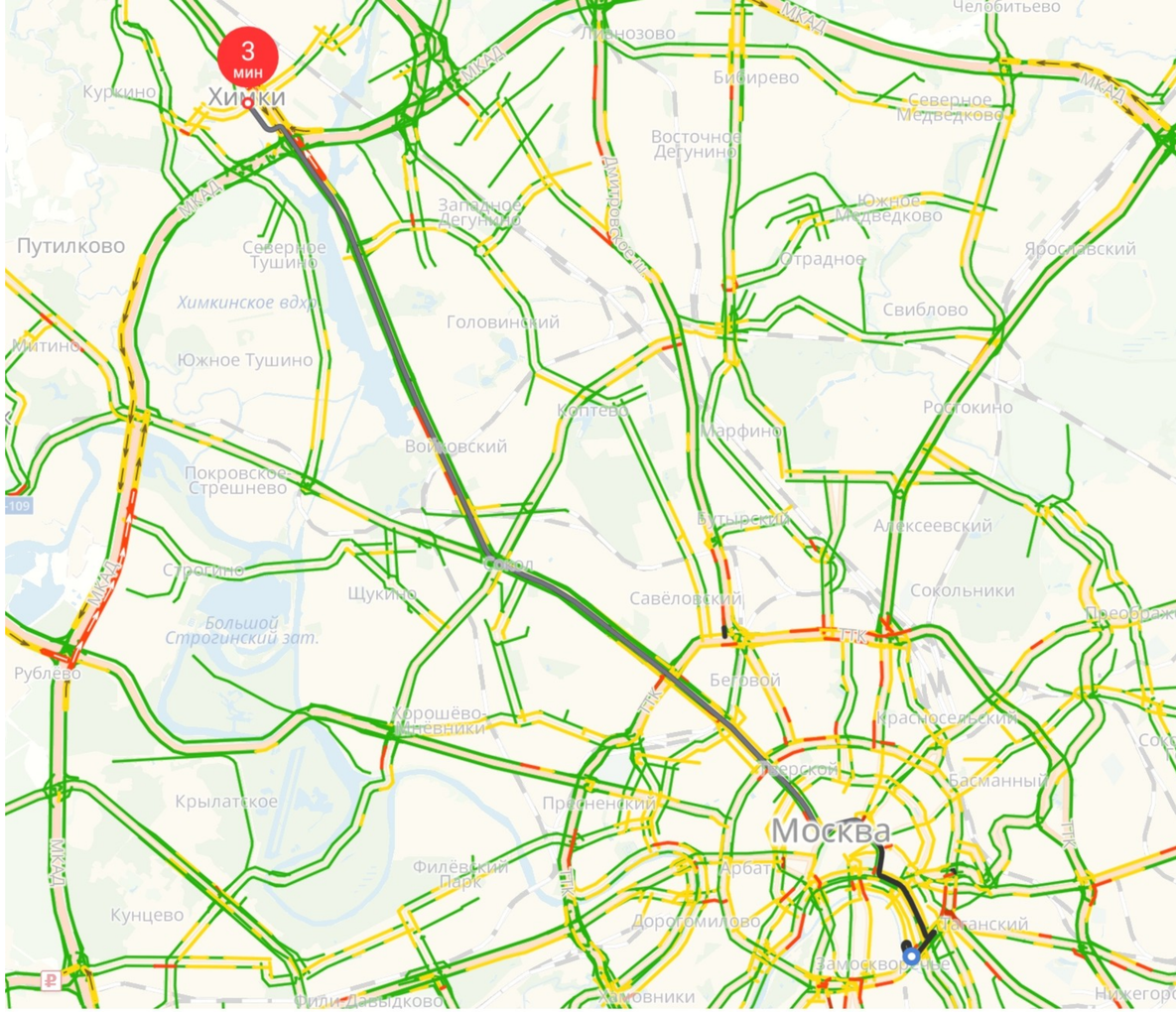
Полухин Антон

Antony Polukhin

Яндекс Go

Содержание

- Зачем?
  - Соверменные нагрузки
  - История про 1%
  - Проблема C10K
  - userver
- Как?
  - «Hello world»
  - Конфиги
  - Компоненты
  - gRPC



Современные нагрузки

Нагрузка

Нагрузка

- 700 000 активных водителей

Нагрузка

- 700 000 активных водителей
- Половина на линии

Нагрузка

- 700 000 активных водителей
- Половина на линии
  - 350к запросов в секунду статуса/положения/…

Нагрузка

- 700 000 активных водителей
- Половина на линии
  - 350к запросов в секунду статуса/положения/…

1 секунда / 350к запросов

Нагрузка

- 700 000 активных водителей
- Половина на линии
  - 350к запросов в секунду статуса/положения/…

1 секунда / 350к запросов => 0.0000003 сек. => 3us

Нагрузка

- 700 000 активных водителей
- Половина на линии
  - 350к запросов в секунду статуса/положения/…

1 секунда / 350к запросов => 0.0000003 сек. => 3us

- Динамическая аллокация — ~1us
- Системный вызов — ~2us

Логика

Логика

- авторизация (поход в базу?)

Логика

- авторизация (поход в базу?)
- запрос надо и заказчику перенаправить, чтобы отрисовать машинку

Логика

- авторизация (поход в базу?)
- запрос надо и заказчику перенаправить, чтобы отрисовать машинку
- новую часть карты подгрузить, при необходимости

Логика

- авторизация (поход в базу?)
- запрос надо и заказчику перенаправить, чтобы отрисовать машинку
- новую часть карты подгрузить, при необходимости
- из данных водителя обновить информацию о пробках

Логика

- авторизация (поход в базу?)
- запрос надо и заказчику перенаправить, чтобы отрисовать машинку
- новую часть карты подгрузить, при необходимости
- из данных водителя обновить информацию о пробках
- найти и нарисовать оптимальный маршрут

Логика

- авторизация (поход в базу?)
- запрос надо и заказчику перенаправить, чтобы отрисовать машинку
- новую часть карты подгрузить, при необходимости
- из данных водителя обновить информацию о пробках
- найти и нарисовать оптимальный маршрут
- найти исполнителя/заказчика

Логика

- авторизация (поход в базу?)
- запрос надо и заказчику перенаправить, чтобы отрисовать машинку
- новую часть карты подгрузить, при необходимости
- из данных водителя обновить информацию о пробках
- найти и нарисовать оптимальный маршрут
- найти исполнителя/заказчика
- ...

Что делать?

350к запросов в секунду

350к запросов в секунду

Можно обработать приблизительно вот так:

- Делаем 100 сервисов разного функционала
- Каждый из сервисов запускаем в 10ти экземплярах

350к запросов в секунду

Можно обработать приблизительно вот так:

- Делаем 100 сервисов разного функционала
- Каждый из сервисов запускаем в 10ти экземплярах

Получаем 1000 серверов для такси, на каждый около 35к запросов в секунду

История про 1%

Масштабы современных IT компаний

Масштабы современных IT компаний

- 1.000.000 серверов (Google в 2011)

Масштабы современных IT компаний

- 1.000.000 серверов (Google в 2011)

  1% => 10.000 серверов

Масштабы современных IT компаний

- 1.000.000 серверов (Google в 2011)

  1% => 10.000 серверов

- 1.000 серверов из наших расчётов

  1% => 10 серверов

Масштабы современных IT компаний

- 1.000.000 серверов (Google в 2011)

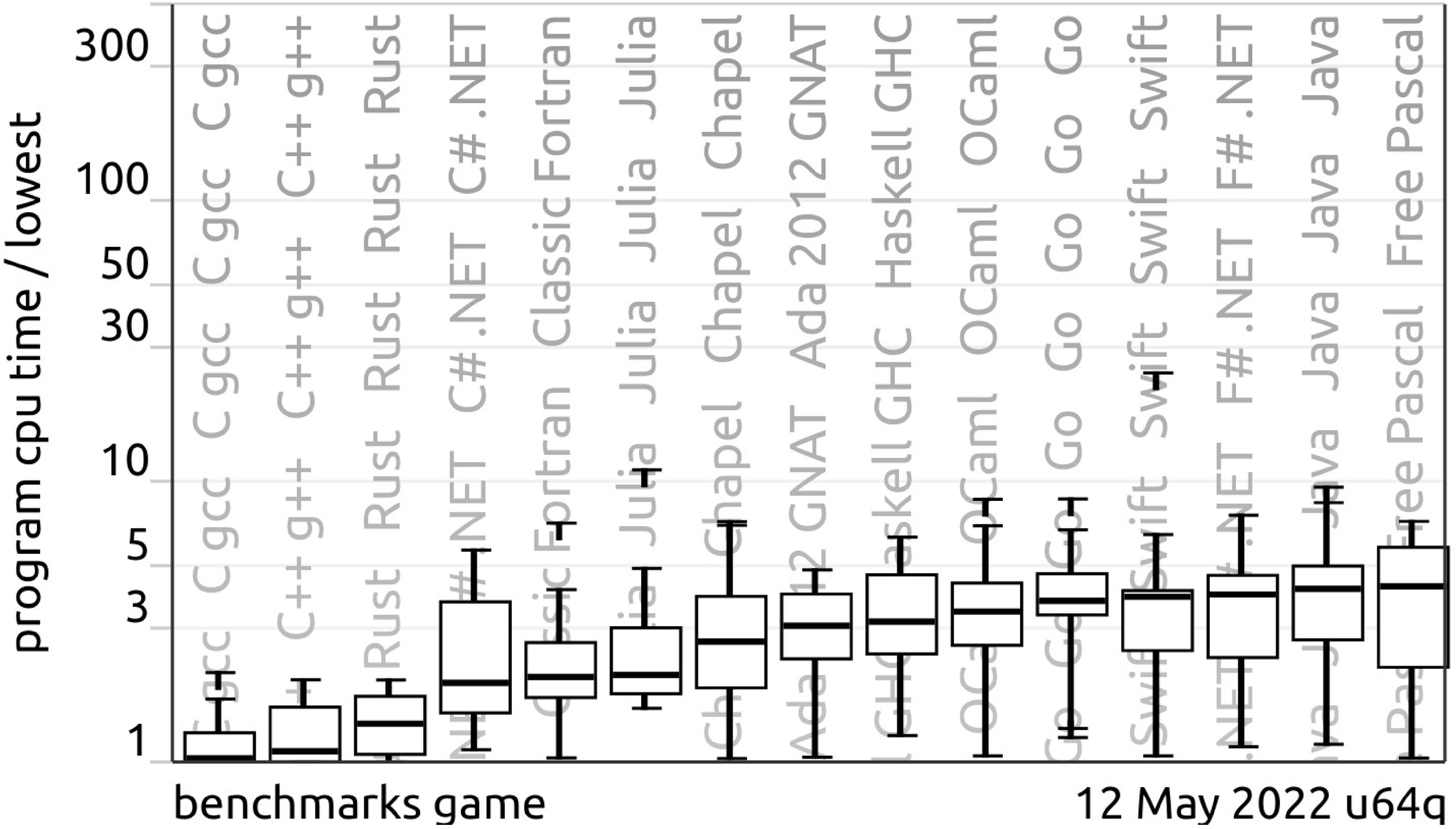
  1% => 10.000 серверов

- 1.000 серверов из наших расчётов

  1% => 10 серверов

Каждый сервер надо питать, охлаждать, обслуживать, ...

Современные языки программирования



Проблема C10K

350к запросов в секунду

Можно обработать приблизительно вот так:

- Делаем 100 сервисов разного функционала
- Каждый из сервисов запускаем в 10ти экземплярах

Получаем 1000 серверов для такси, на каждый около **35к запросов** в секунду

Синхронный сервер

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

Синхронный сервер

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

Синхронный сервер

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

Синхронный сервер

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

Синхронный сервер

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

Синхронный сервер

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

Синхронный сервер

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

Синхронный сервер

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

Синхронный сервер

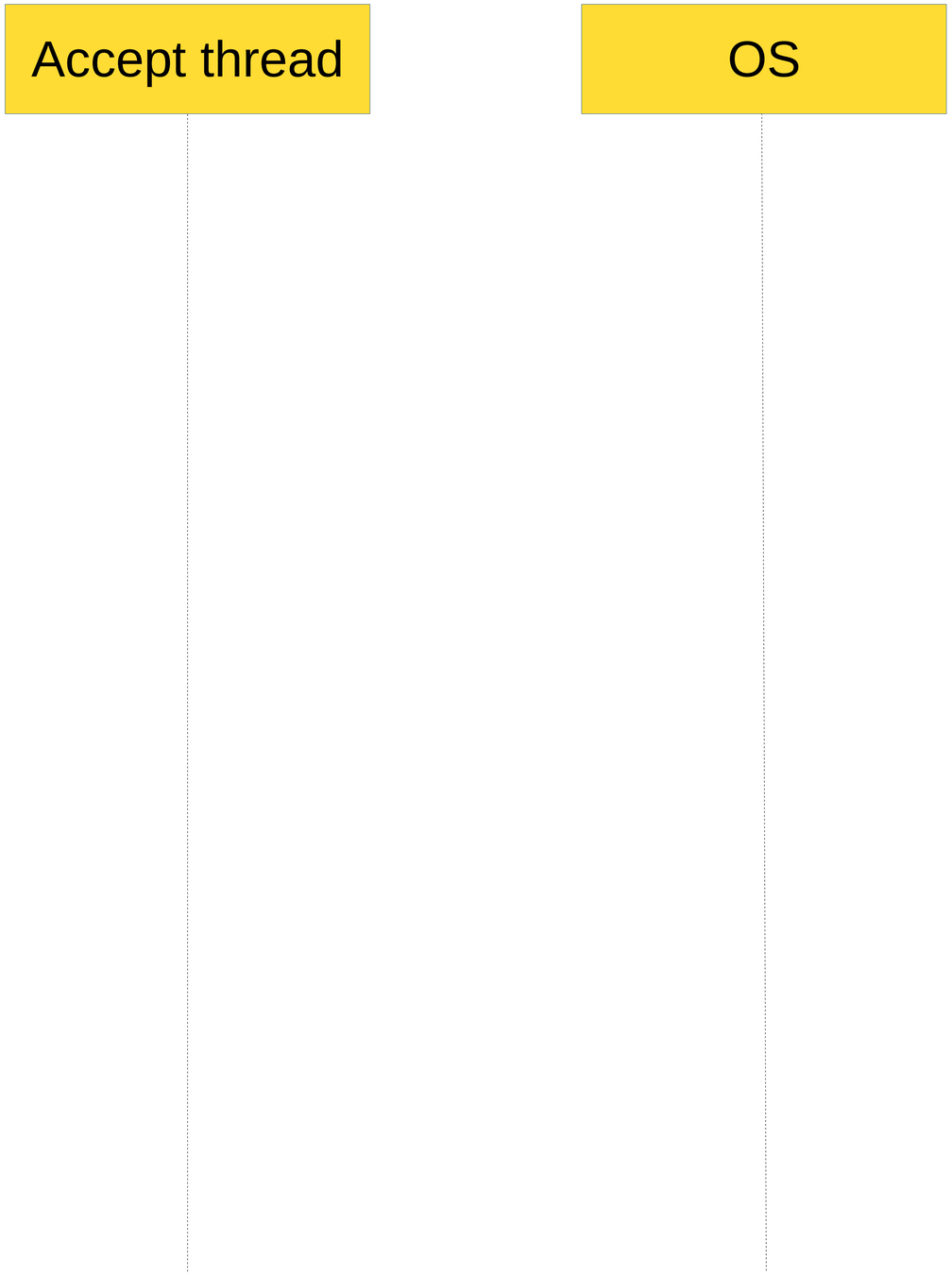
```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

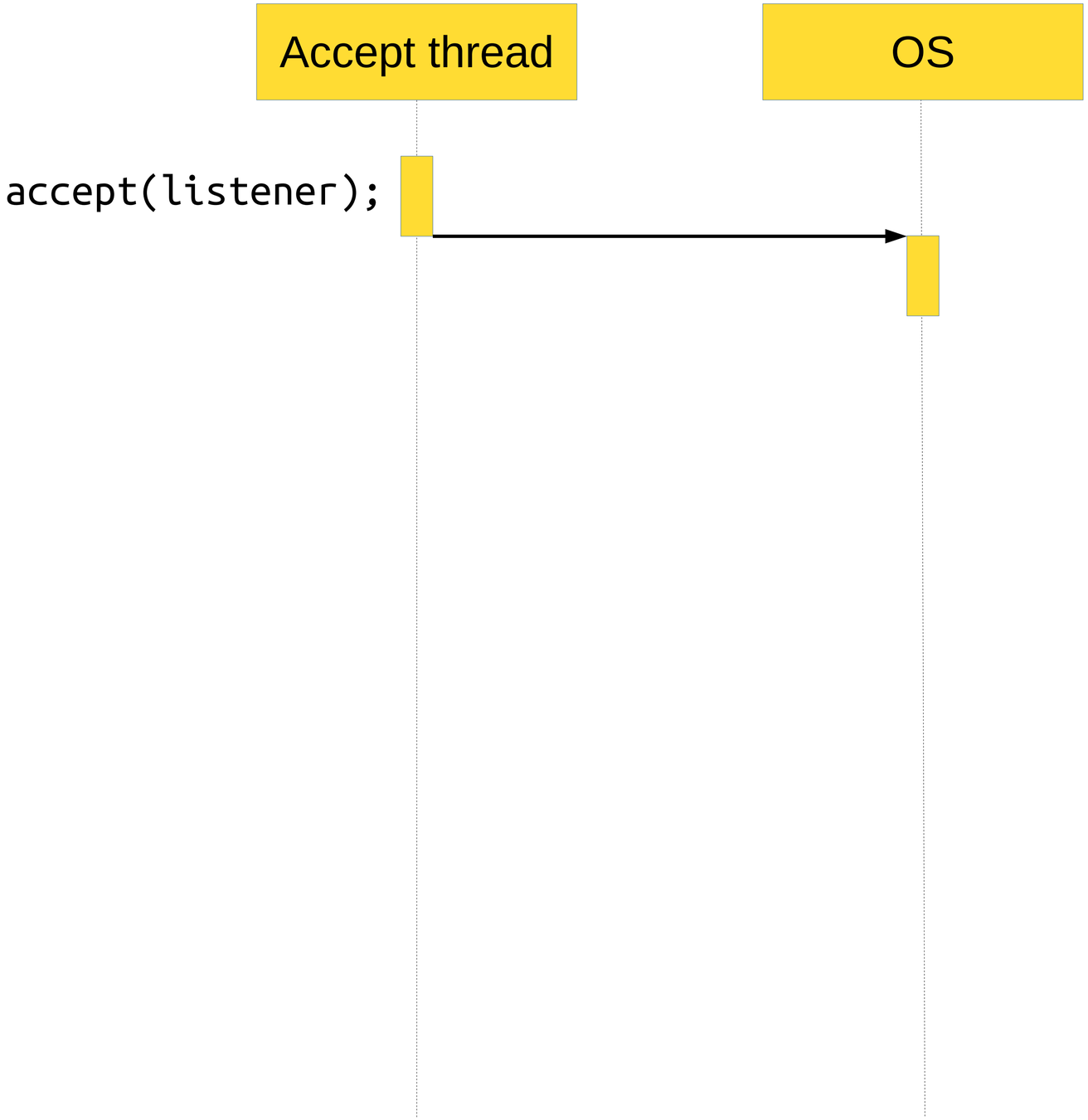
Синхронный сервер

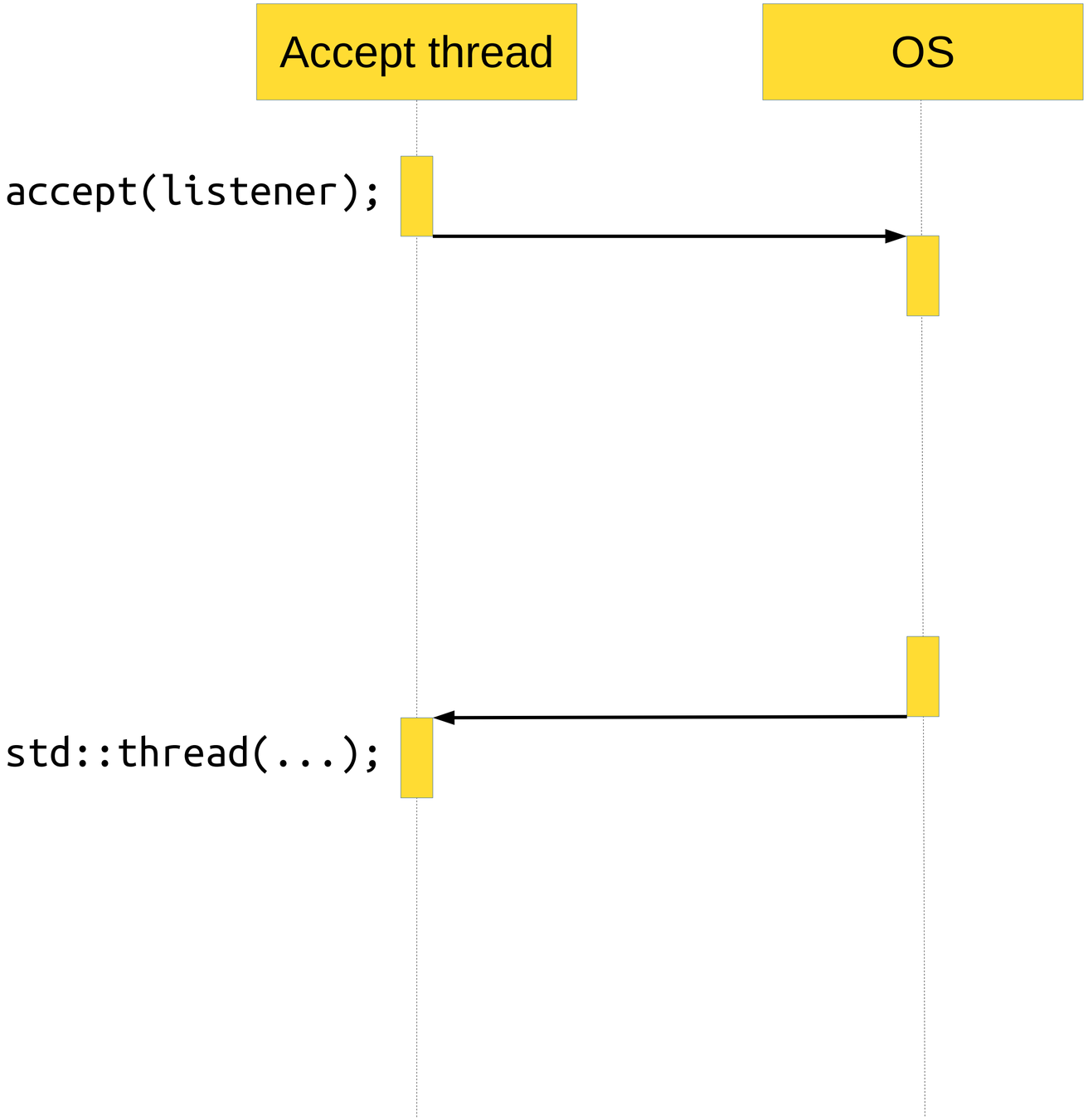
```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

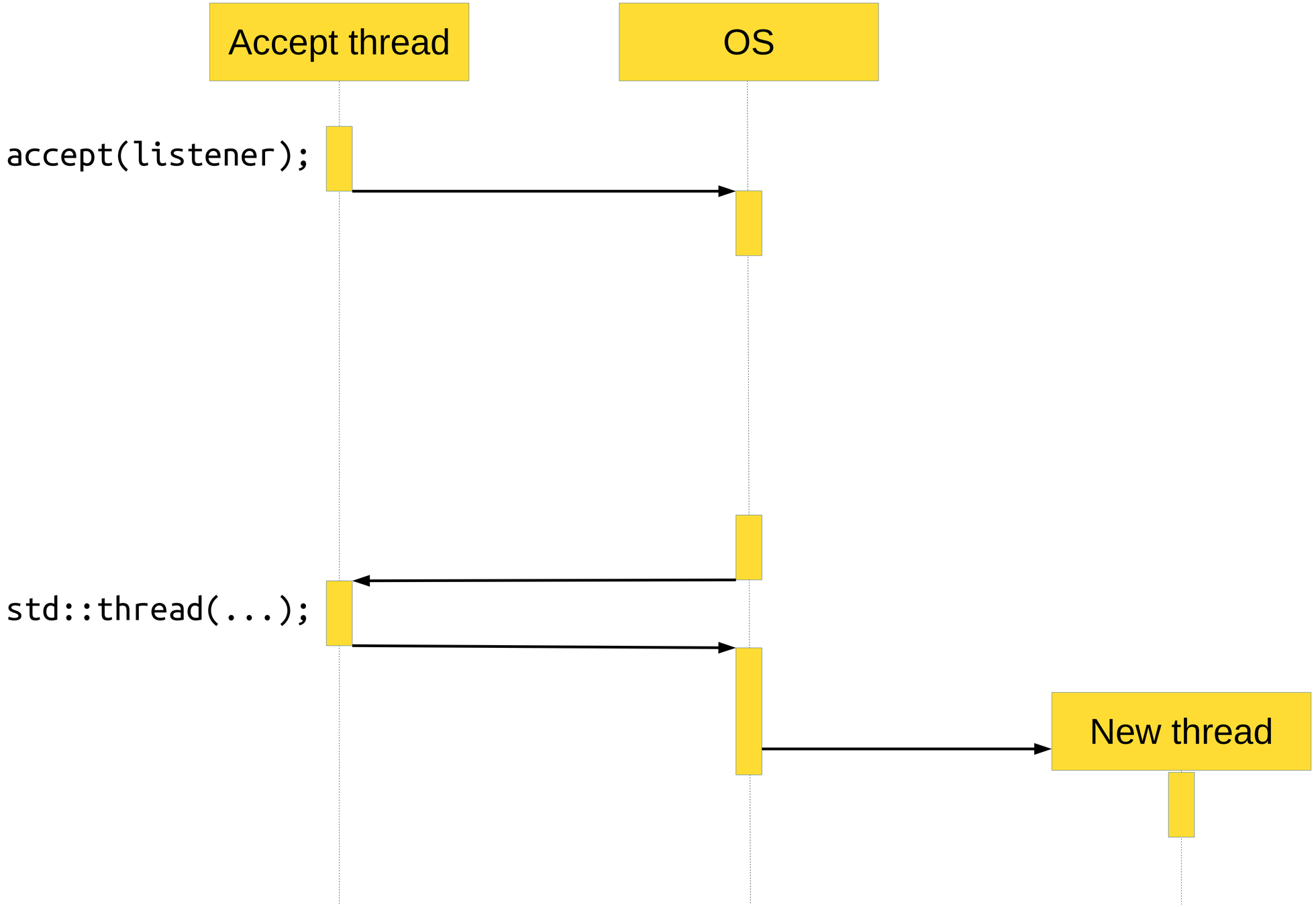
Синхронный сервер

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd([socket = std::move(new_socket)] {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```


Accept thread
OS

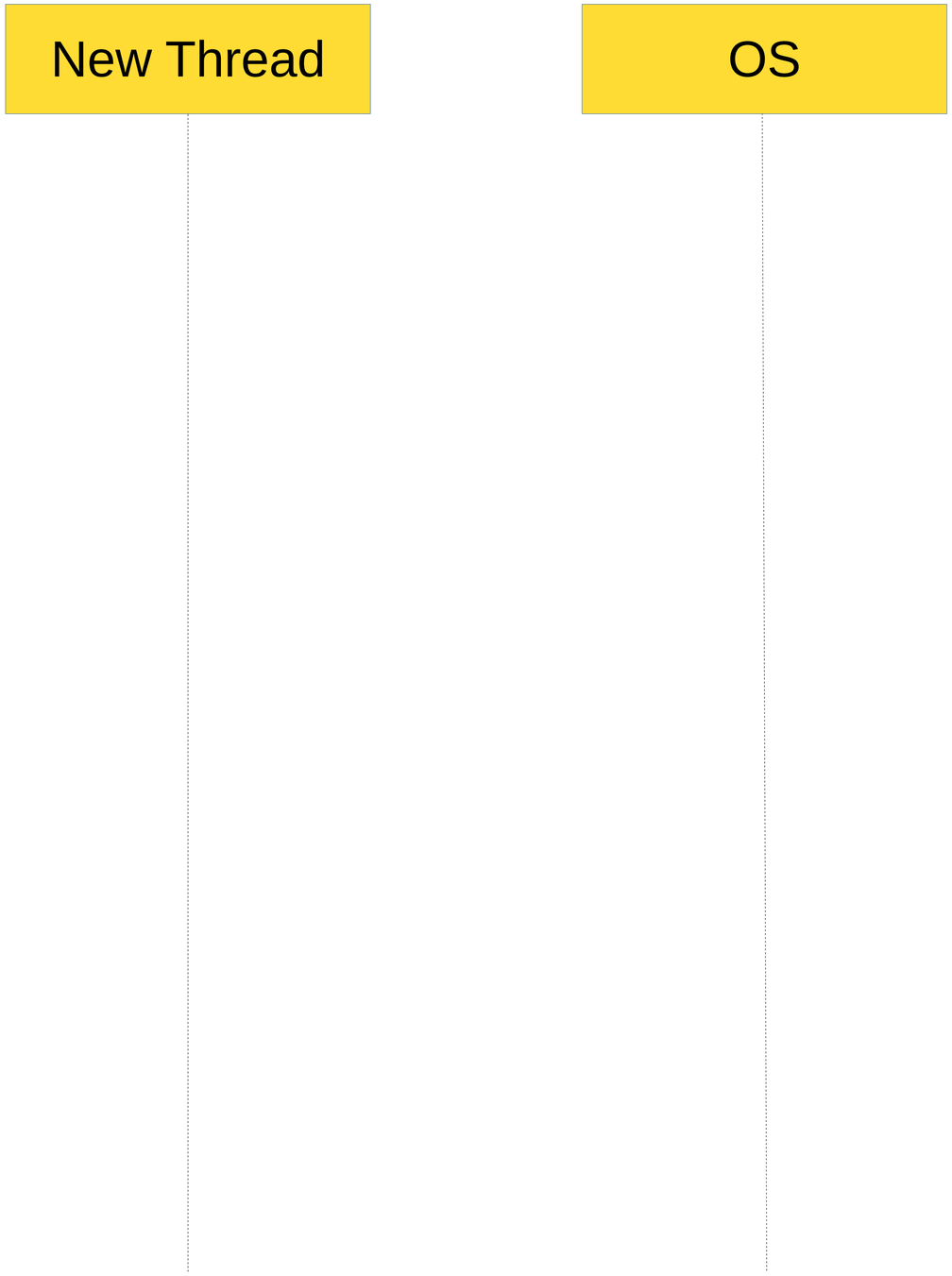

Accept thread
OS
accept(listener);

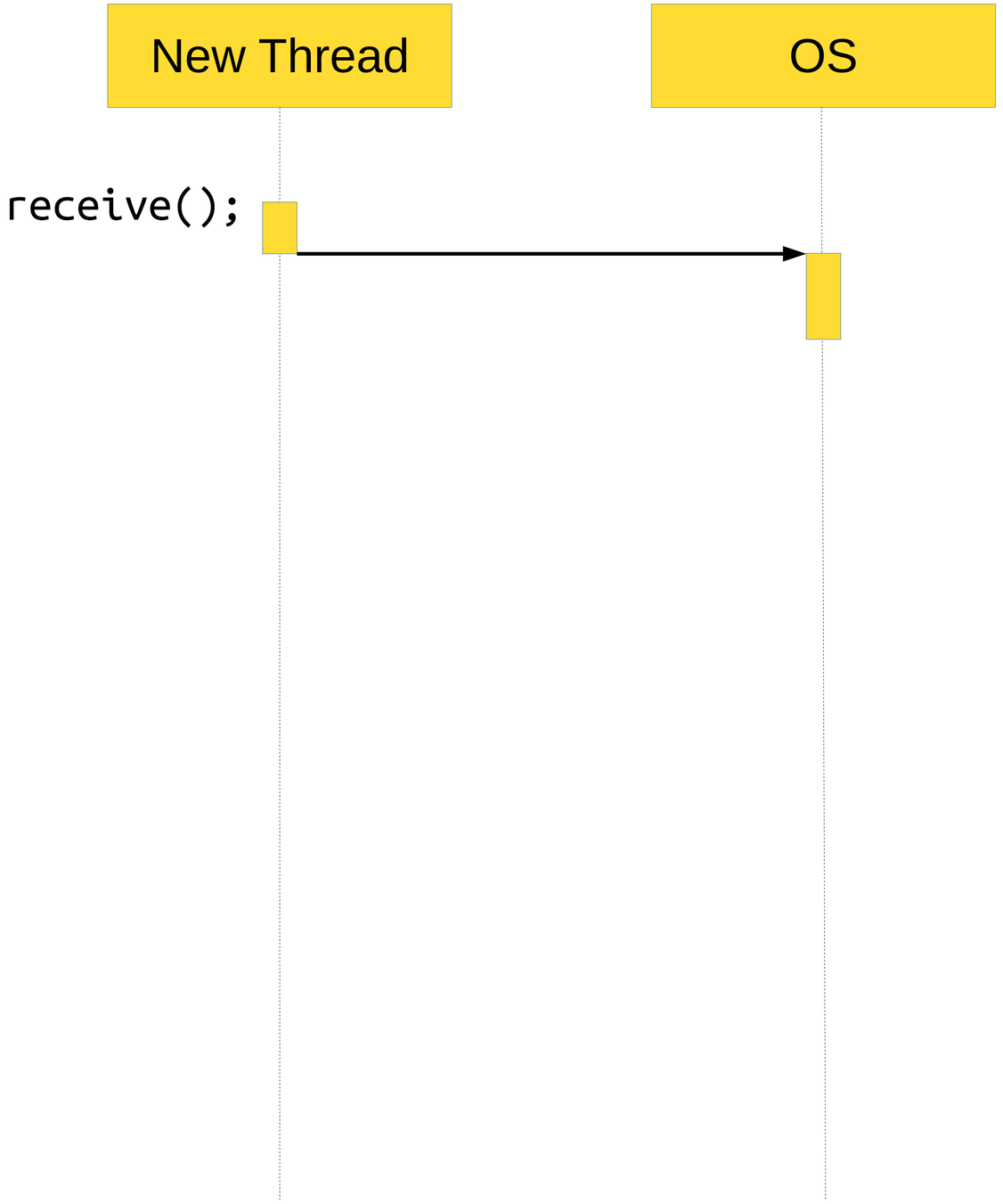

Accept thread
OS
accept(listener);
std::thread(...);

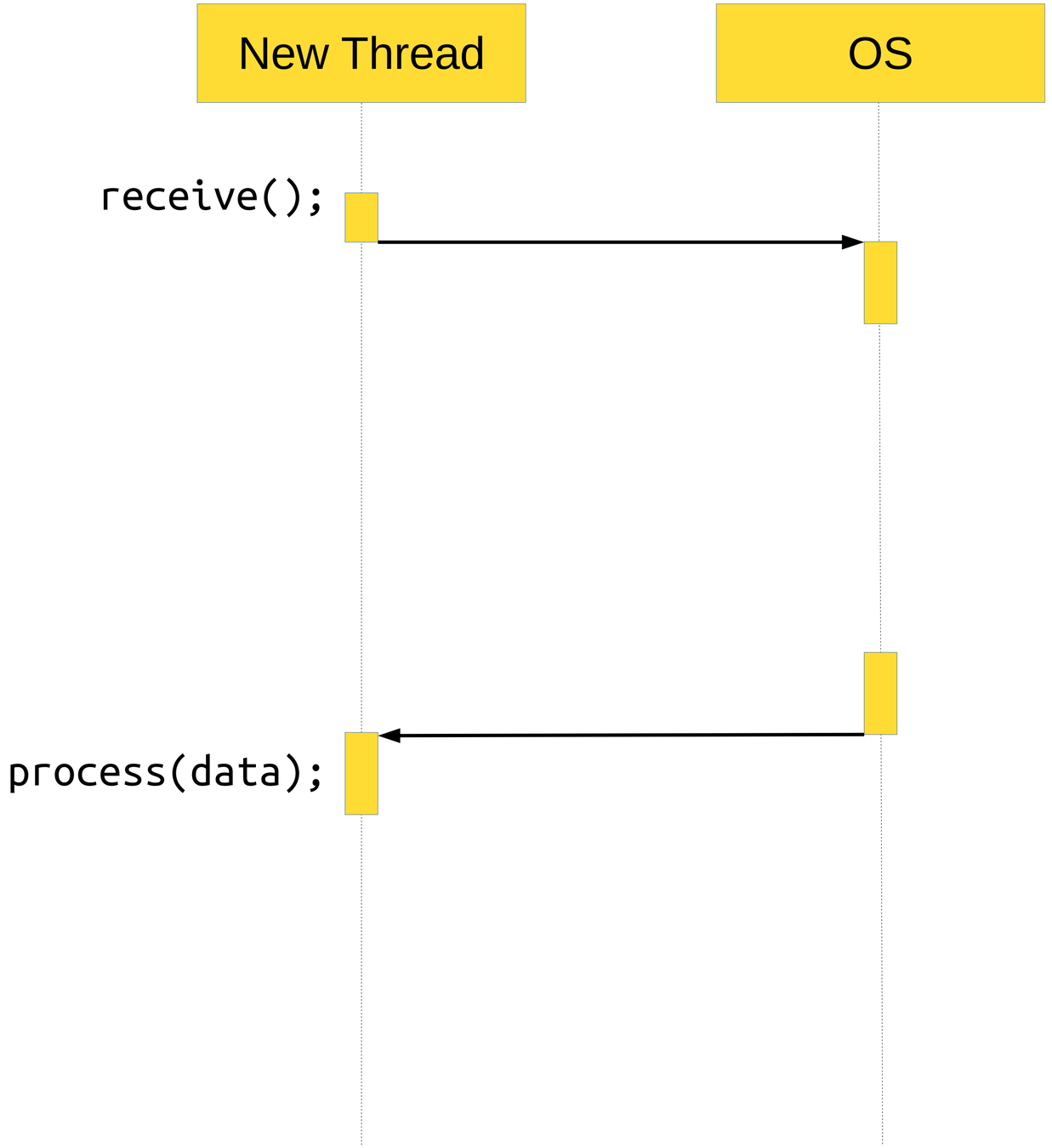

Accept thread
OS
accept(listener);
std::thread(...);
New thread

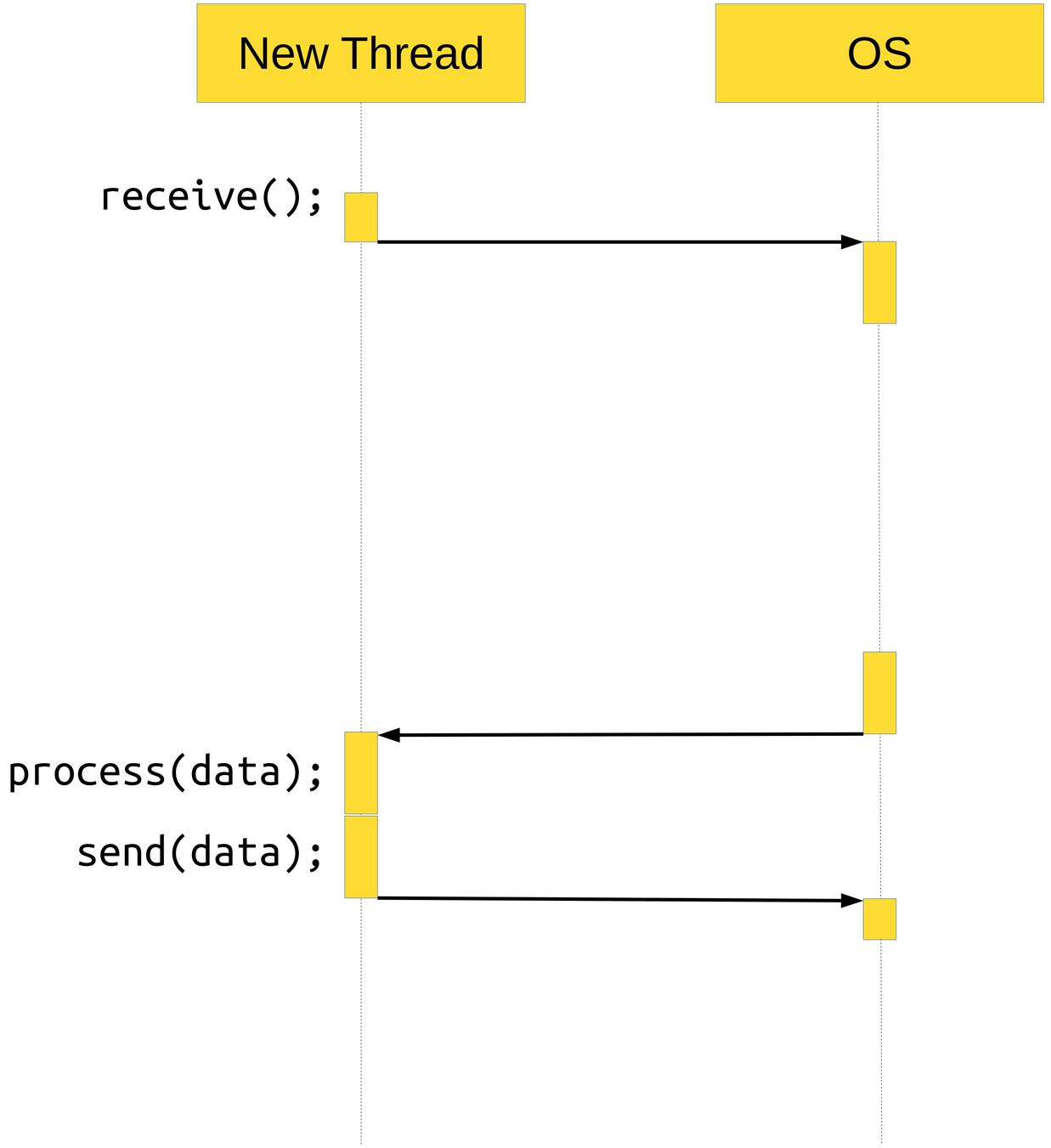

Accept thread
OS
accept(listener);
std::thread(...);
New thread
accept(listener);

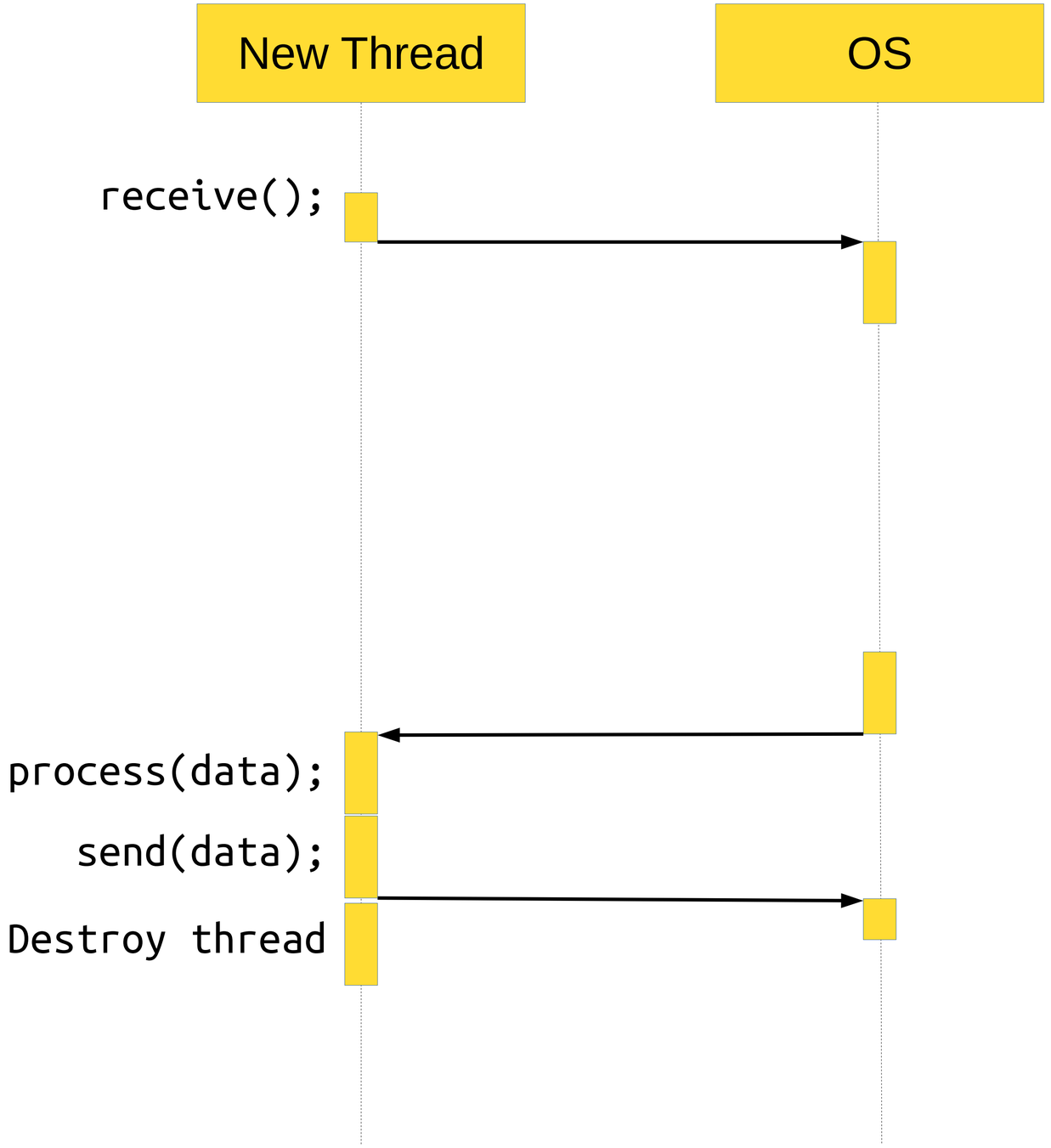
Что там делает новый поток?


New Thread
OS


New Thread
OS
receive();


New Thread
OS
receive();
process(data);


New Thread
OS
receive();
process(data);
send(data);


New Thread
OS
receive();
process(data);
send(data);
Destroy thread

Плюсы/минусы наивного подхода

Плюсы/минусы наивного подхода

Плюсы:

Плюсы/минусы наивного подхода

Плюсы:

- Всё просто и читаемо

Плюсы/минусы наивного подхода

Плюсы:

- Всё просто и читаемо

Минусы:

Плюсы/минусы наивного подхода

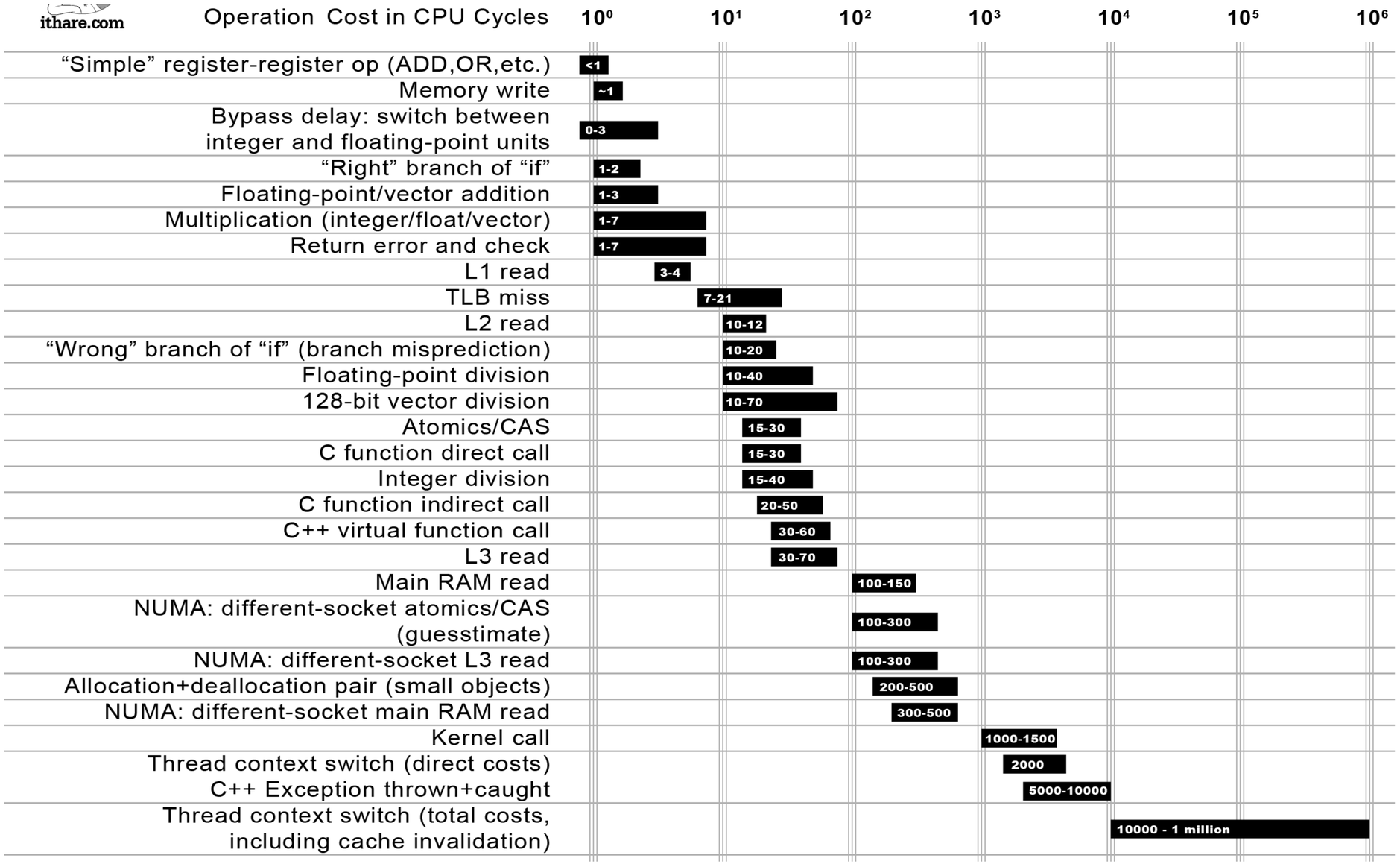
Плюсы:

- Всё просто и читаемо

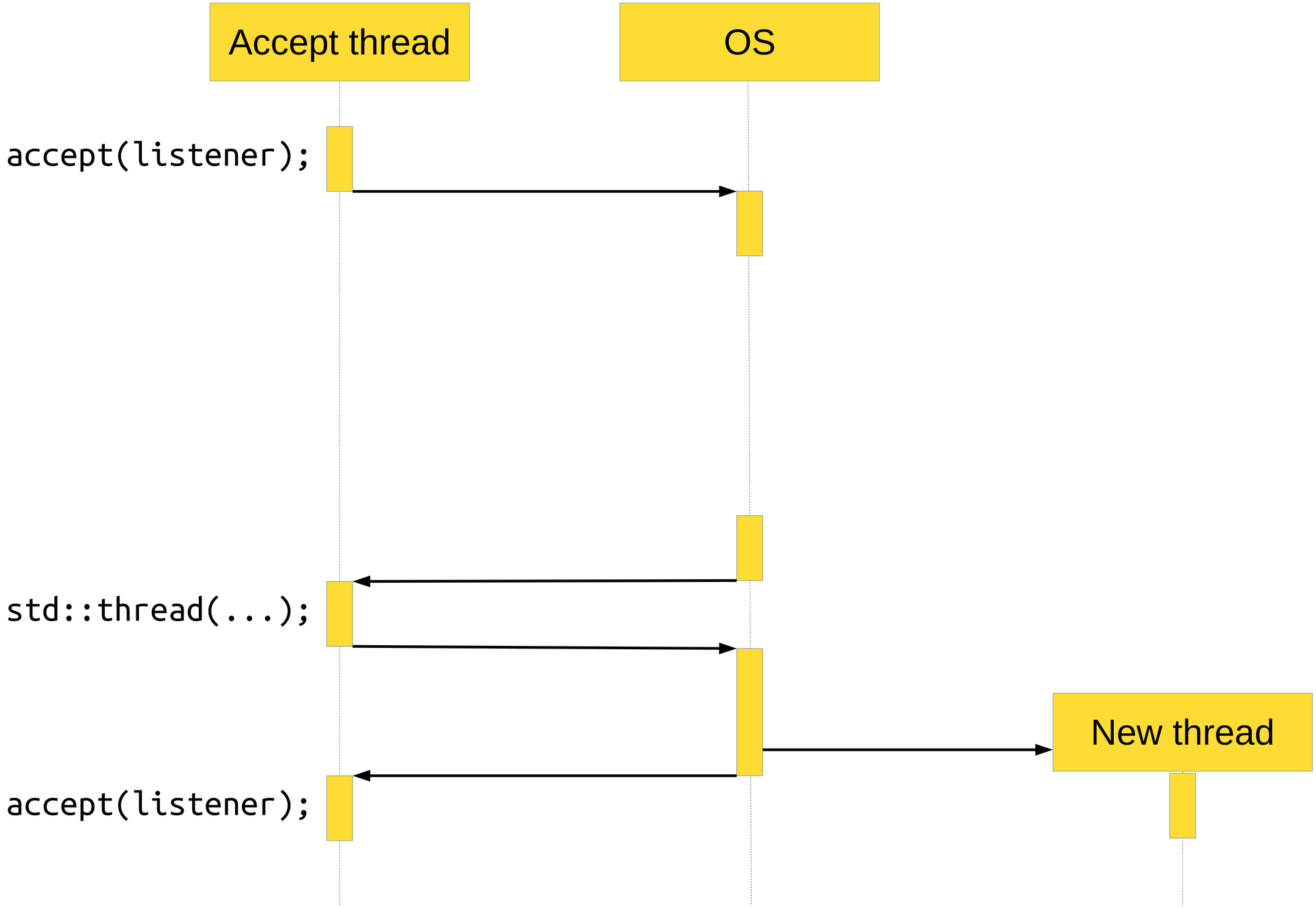
Минусы:

- Не эффективно, потому что...

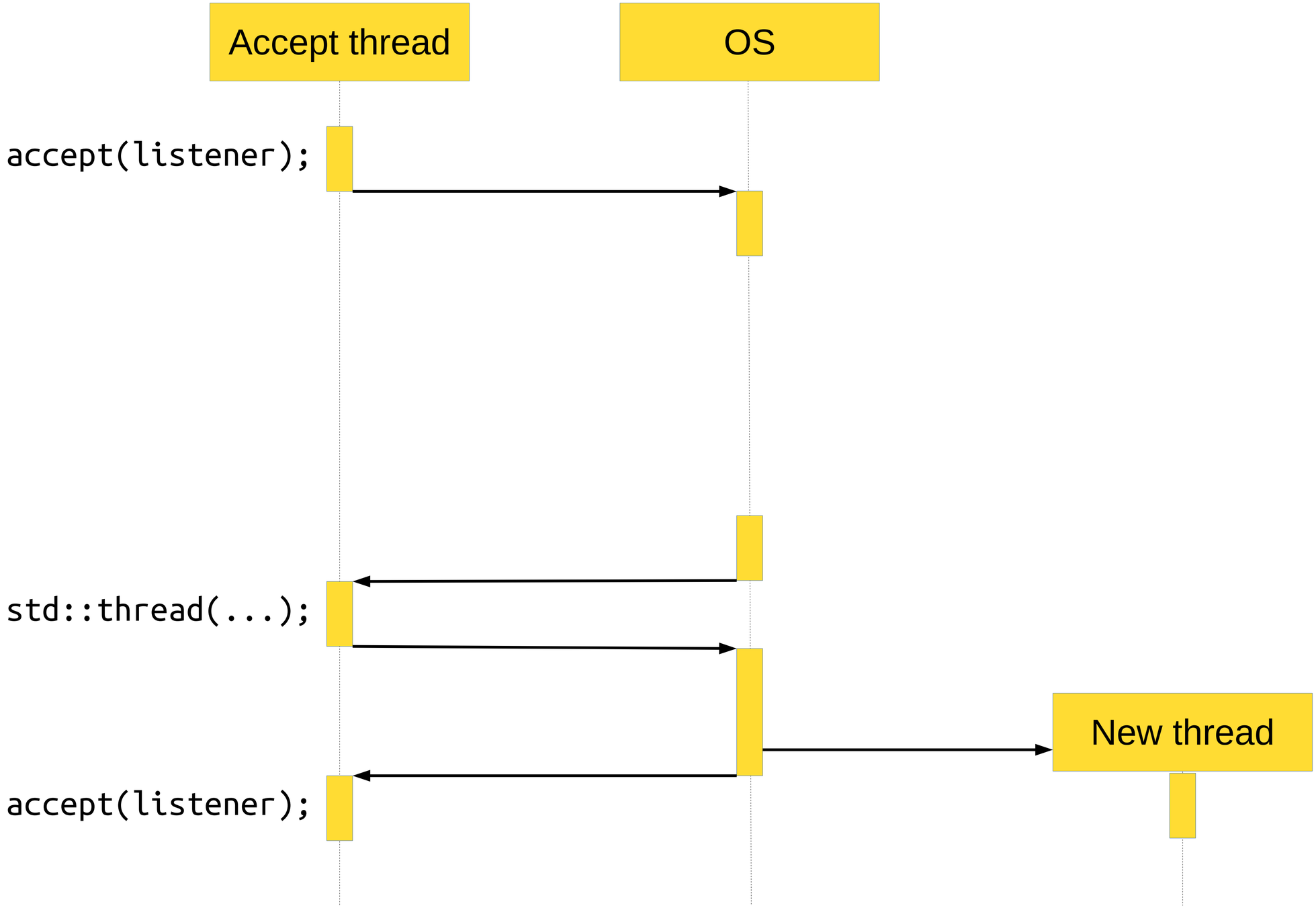
«Цена» операции



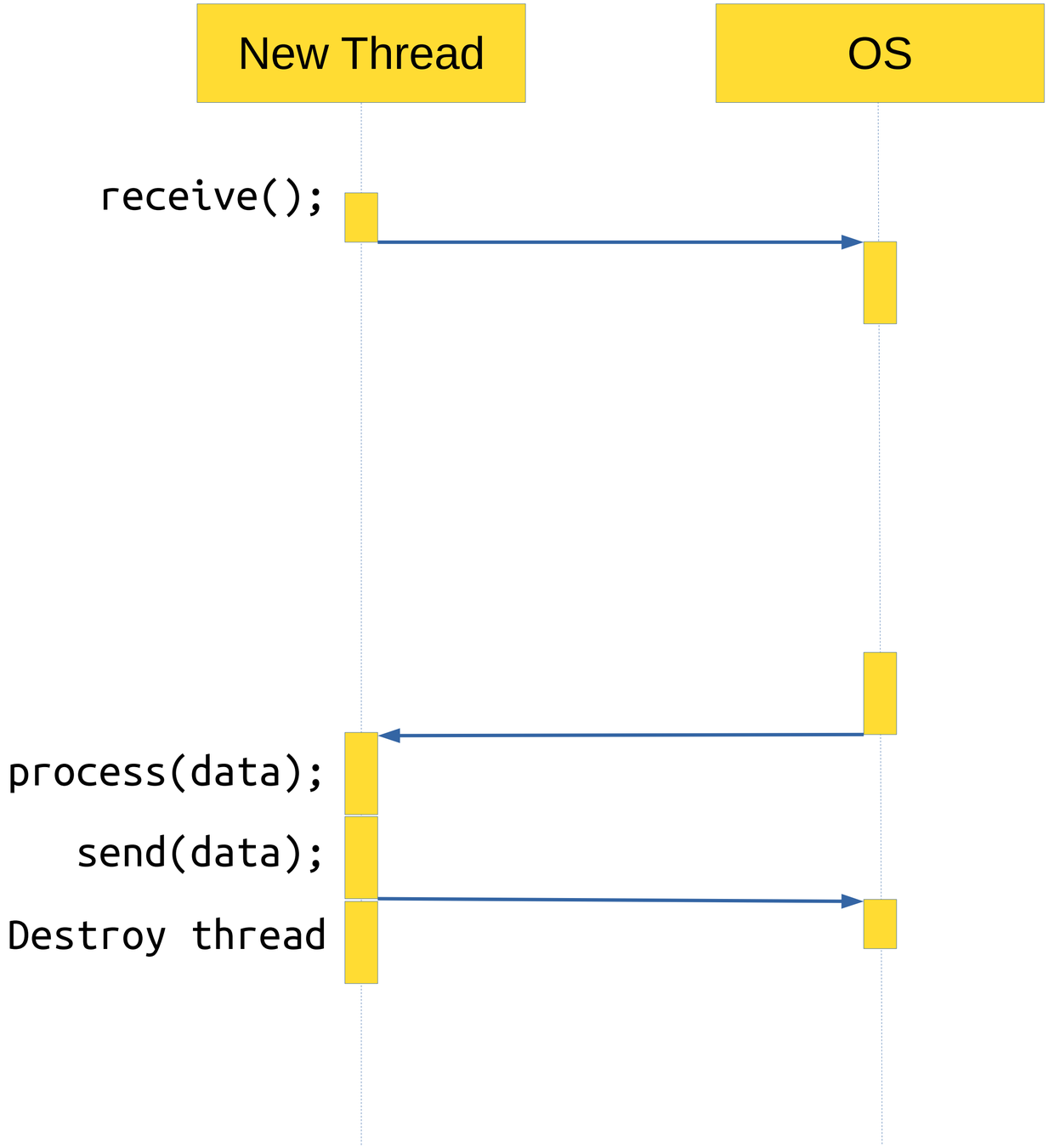
Kernel call / Context Switch

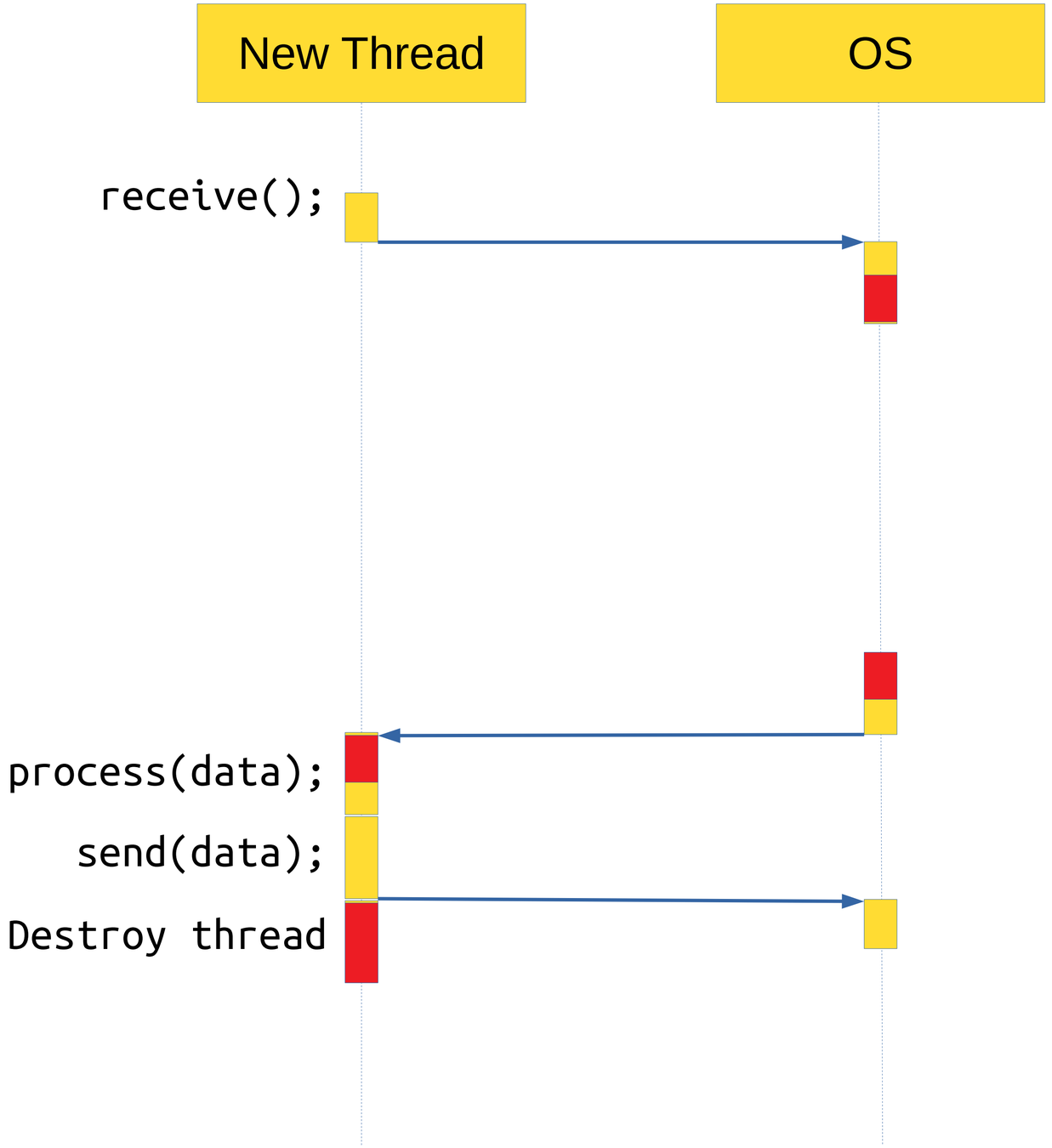

Accept thread
OS
accept(listener);
std::thread(...);
New thread
accept(listener);


Accept thread
OS
accept(listener);
std::thread(...);
New thread
accept(listener);


Accept thread
OS
accept(listener);
std::thread(...);
New thread
accept(listener);


Accept thread
OS
accept(listener);
std::thread(...);
New thread
accept(listener);


New Thread
OS
receive();
process(data);
send(data);
Destroy thread


New Thread
OS
receive();
process(data);
send(data);
Destroy thread

Пишем асинхронный сервер

Было/станет

Было/станет

Было:

Было/станет

Было:

- Отдаём управление ОС при системном вызове

Было/станет

Было:

- Отдаём управление ОС при системном вызове
- Ждём пока событие случится

Было/станет

Было:

- Отдаём управление ОС при системном вызове
- Ждём пока событие случится
- ОС будит поток

Было/станет

Было:

- Отдаём управление ОС при системном вызове
- Ждём пока событие случится
- ОС будит поток

Станет:

Было/станет

Было:

- Отдаём управление ОС при системном вызове
- Ждём пока событие случится
- ОС будит поток

Станет:

- «Забираем» **случившиеся** события

Было/станет

Было:

- Отдаём управление ОС при системном вызове
- Ждём пока событие случится
- ОС будит поток

Станет:

- «Забираем» **случившиеся** события
- Выполняем коллбеки, связанные с этим событиями

Асинхронный сервер

```
void async_accept() {
  accept(listener, [](socket_t socket) {
    async_accept();
    socket.receive(
        [socket](std::vector<unsigned char> data) {
          process(data);
          socket.send(data, kNoCallback);
        });
  });
}
```

Асинхронный сервер

```
void async_accept() {
  accept(listener, [](socket_t socket) {
    async_accept();
    socket.receive(
        [socket](std::vector<unsigned char> data) {
          process(data);
          socket.send(data, kNoCallback);
        });
  });
}
```

Асинхронный сервер

```
void async_accept() {
  accept(listener, [](socket_t socket) {
    async_accept();
    socket.receive(
        [socket](std::vector<unsigned char> data) {
          process(data);
          socket.send(data, kNoCallback);
        });
  });
}
```

Асинхронный сервер

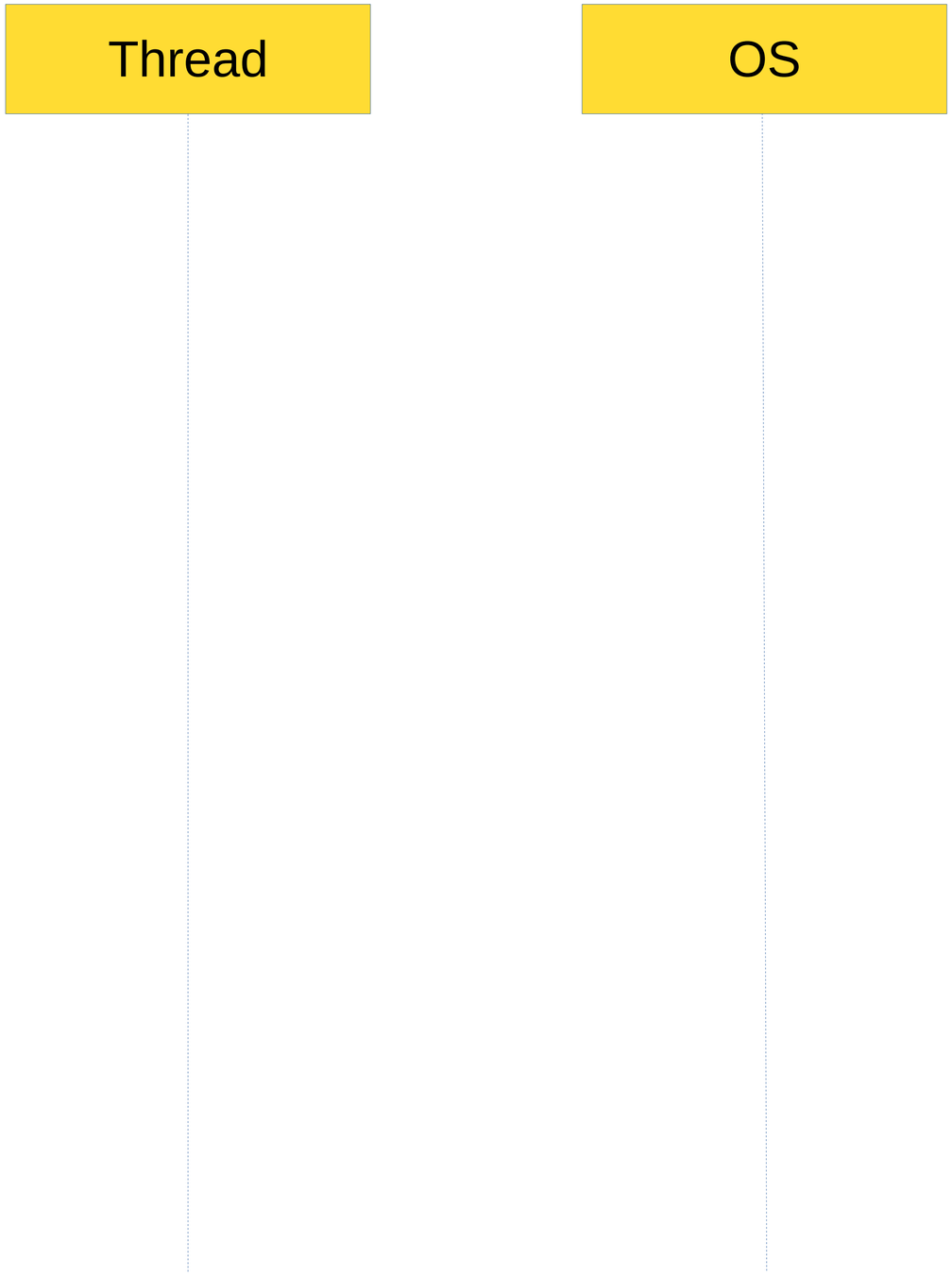
```
void async_accept() {
  accept(listener, [](socket_t socket) {
    async_accept();
    socket.receive(
        [socket](std::vector<unsigned char> data) {
          process(data);
          socket.send(data, kNoCallback);
        });
  });
}
```

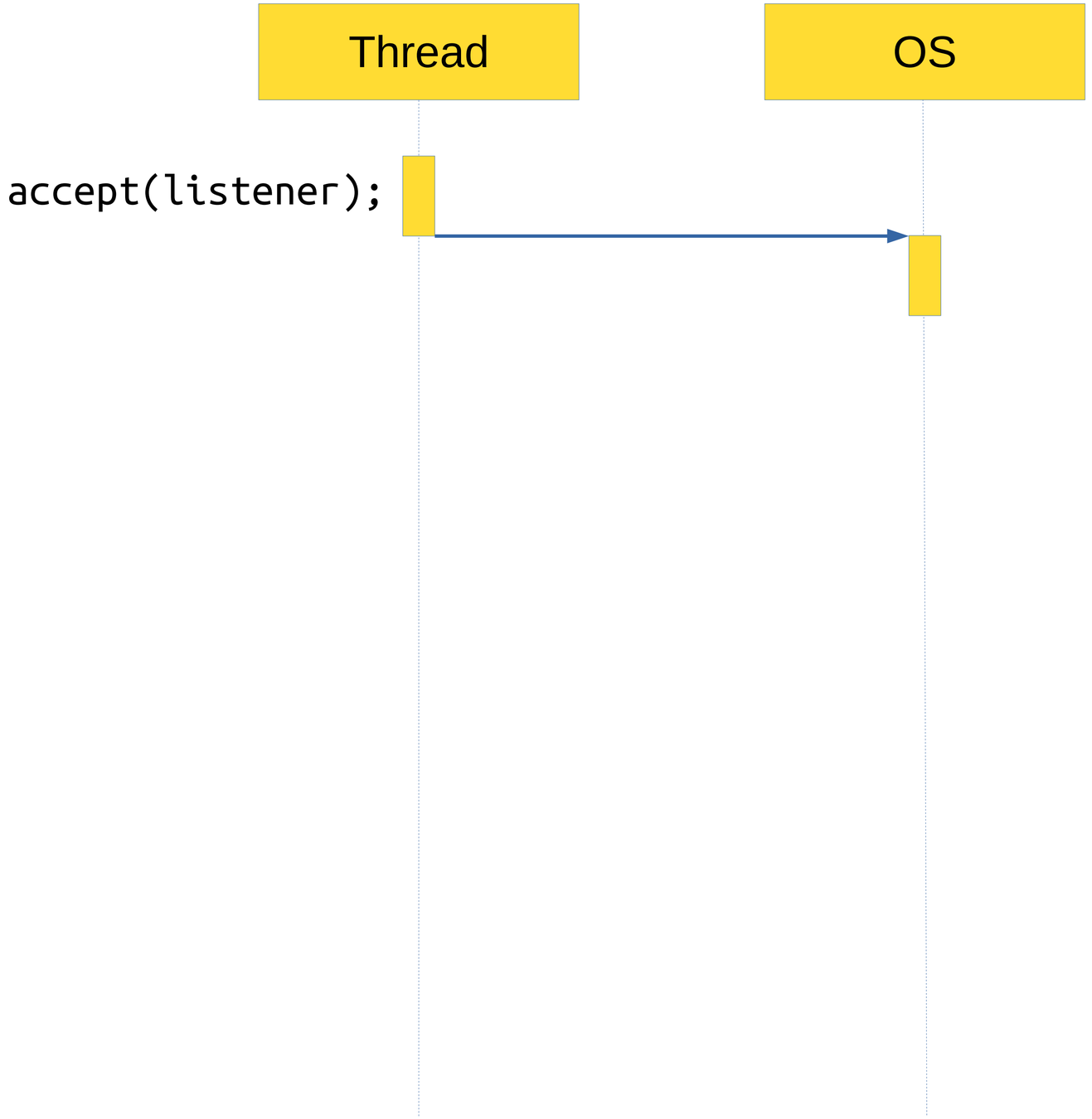
Асинхронный сервер

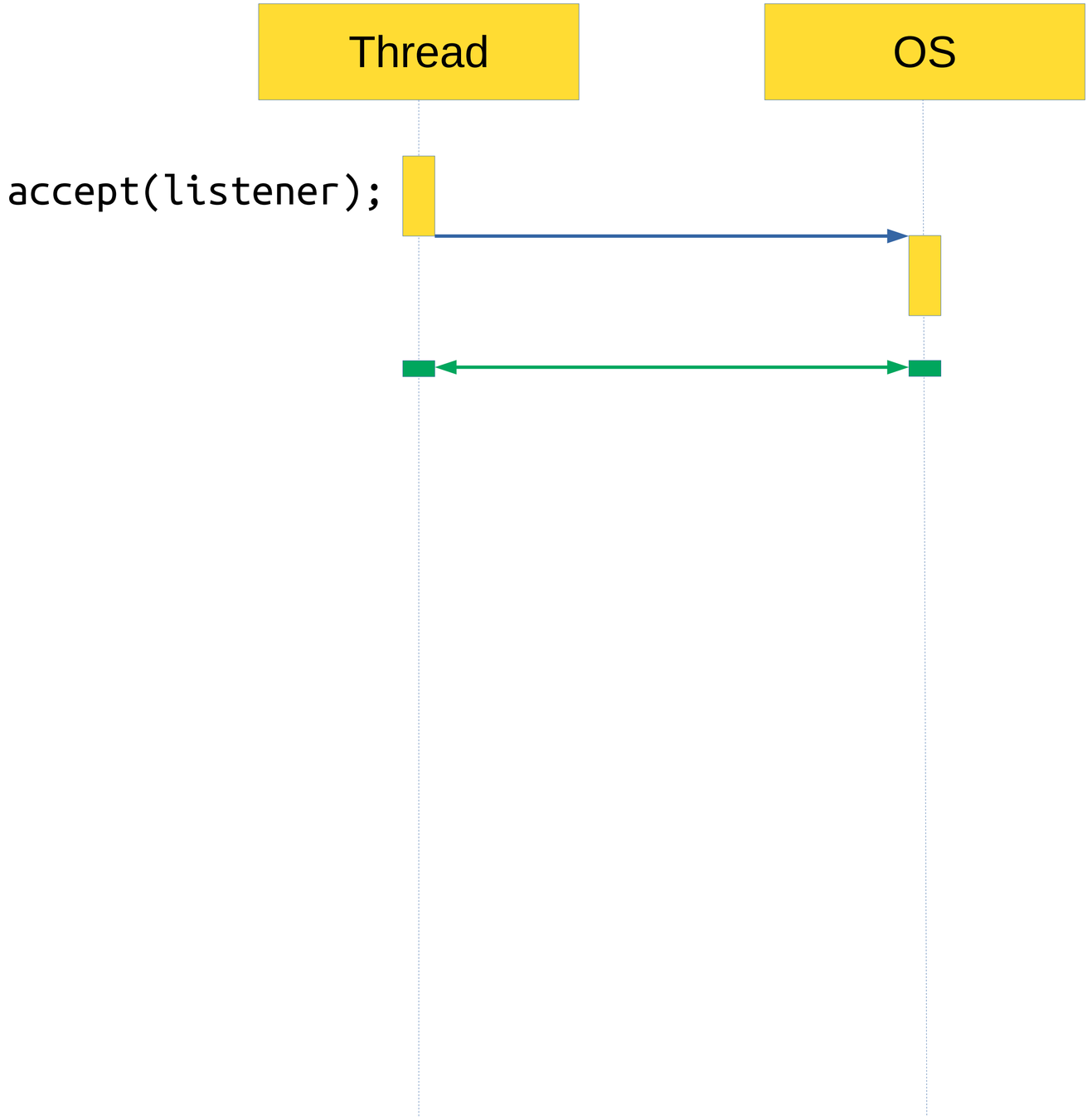
```
void async_accept() {
  accept(listener, [](socket_t socket) {
    async_accept();
    socket.receive(
        [socket](std::vector<unsigned char> data) {
          process(data);
          socket.send(data, kNoCallback);
        });
  });
}
```

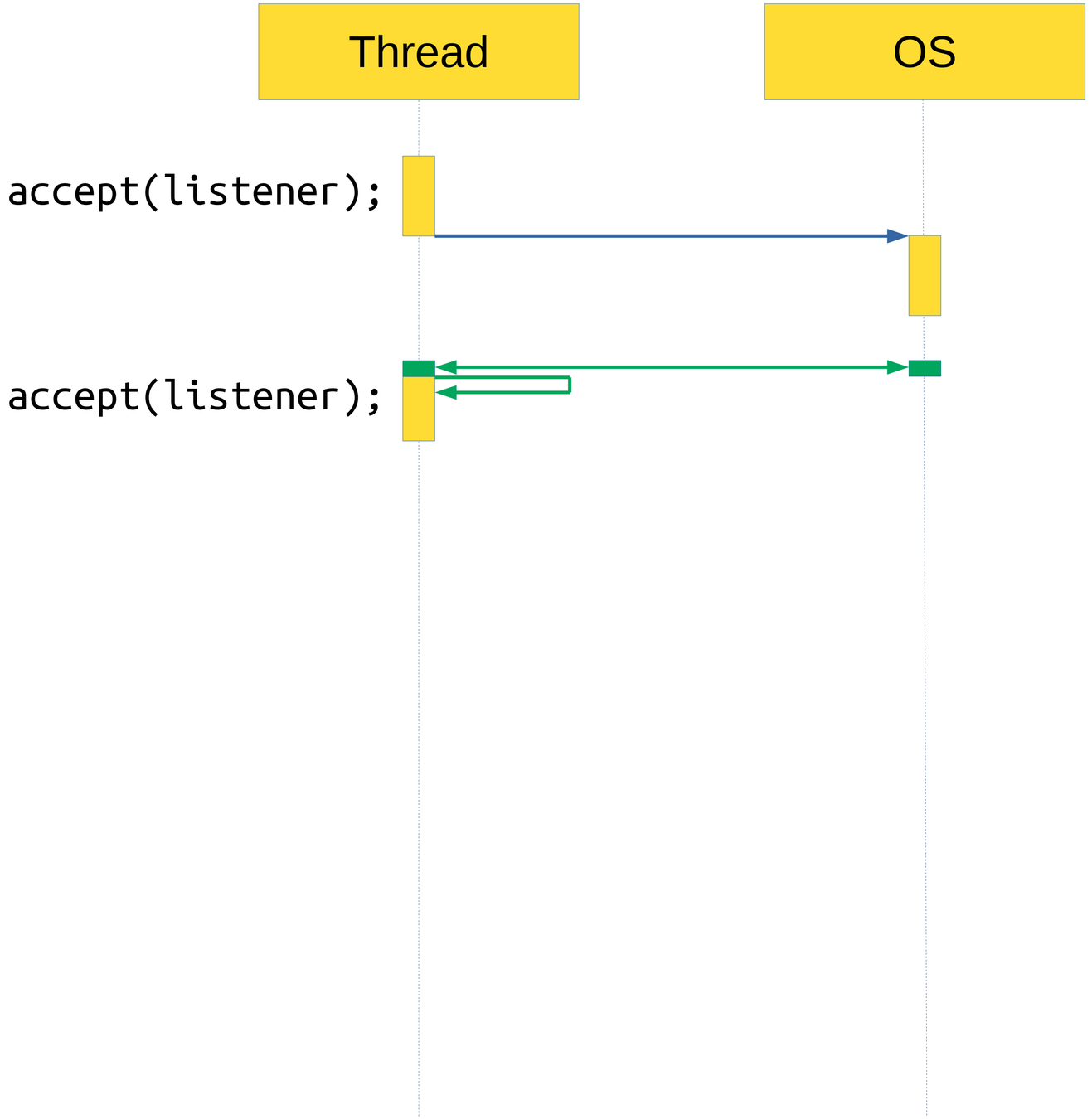
Асинхронный сервер

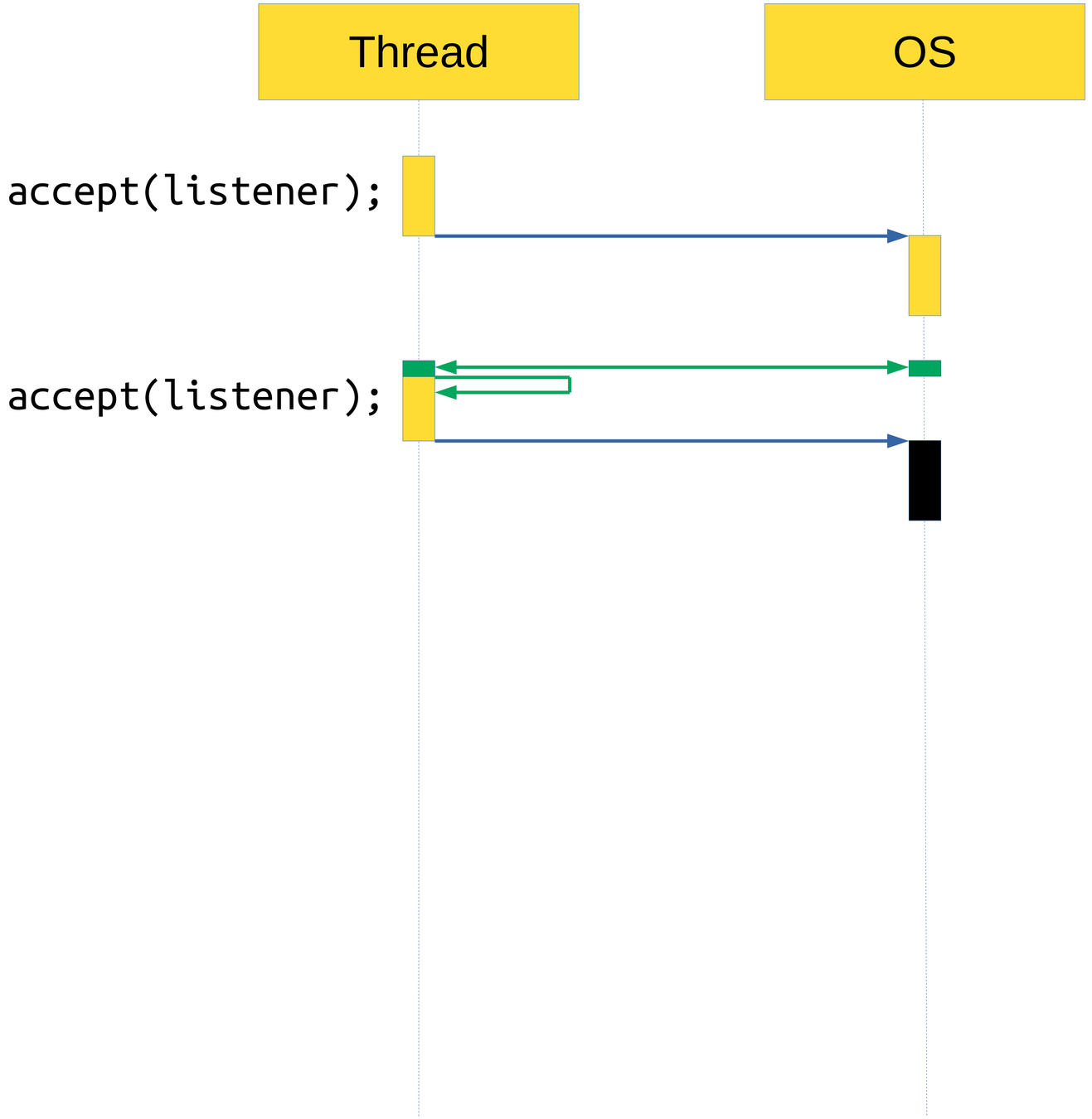
```
void async_accept() {
  accept(listener, [](socket_t socket) {
    async_accept();
    socket.receive(
        [socket](std::vector<unsigned char> data) {
          process(data);
          socket.send(data, kNoCallback);
        });
  });
}
```

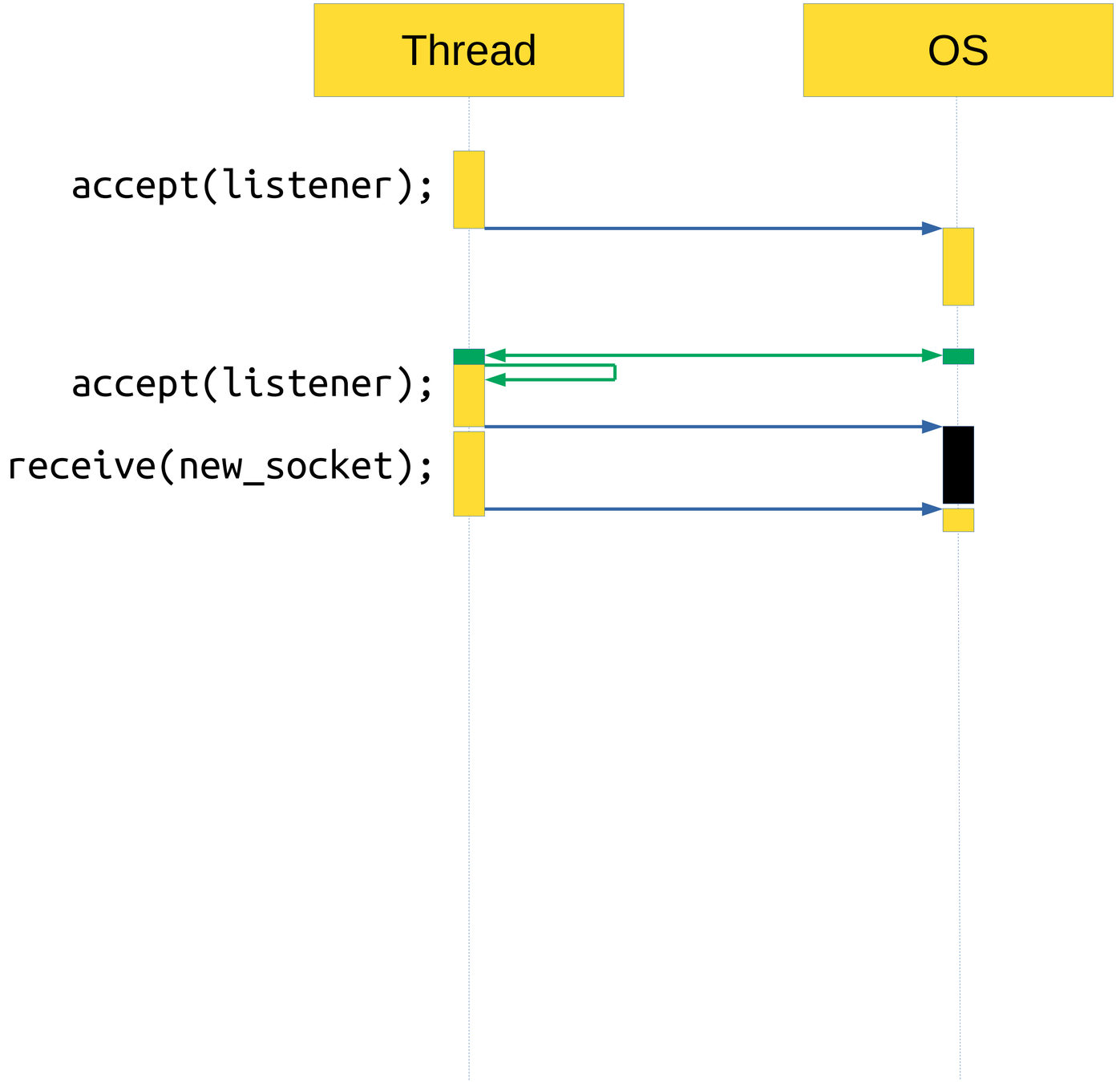

Thread
OS

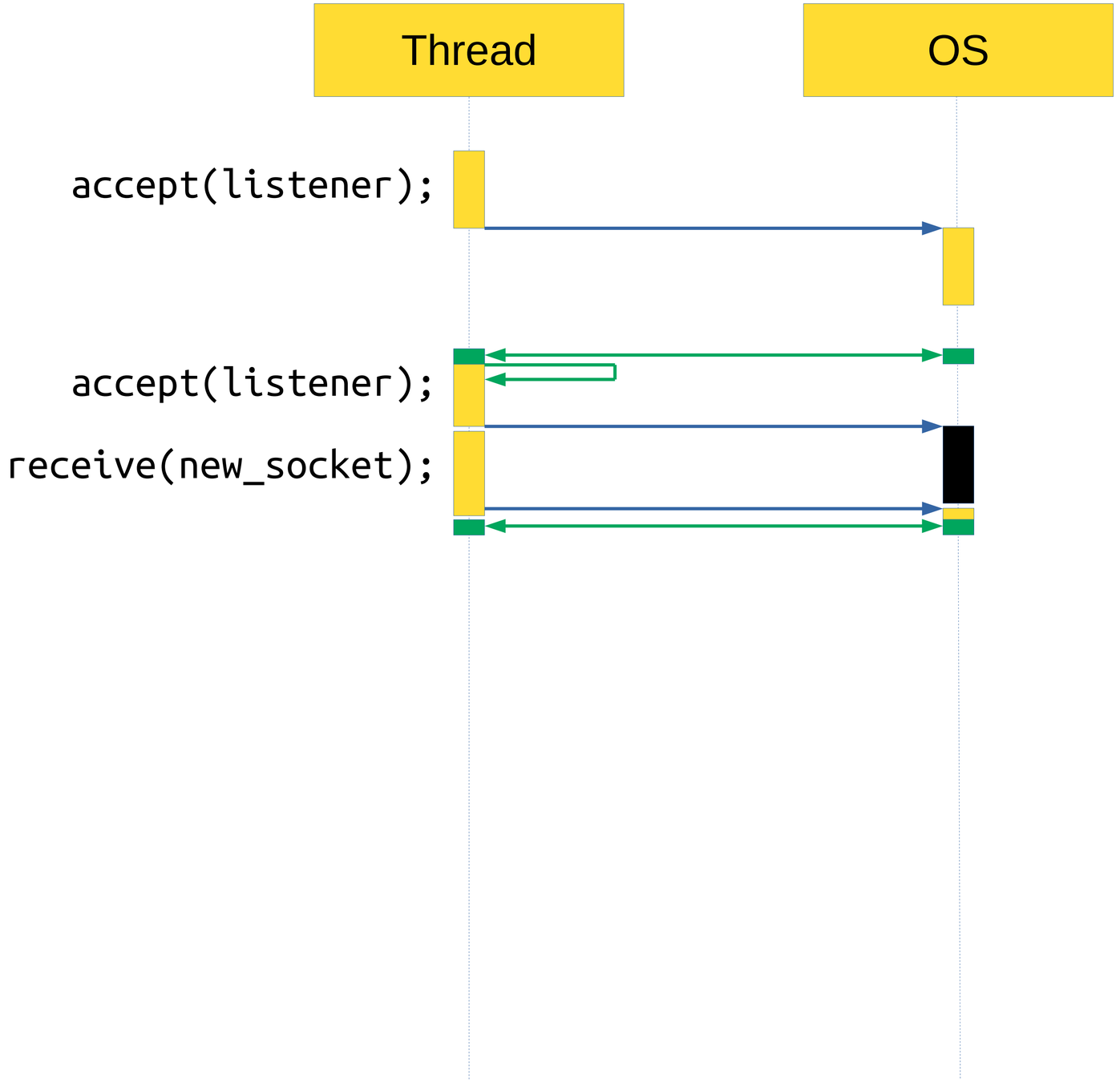

Thread
OS
accept(listener);

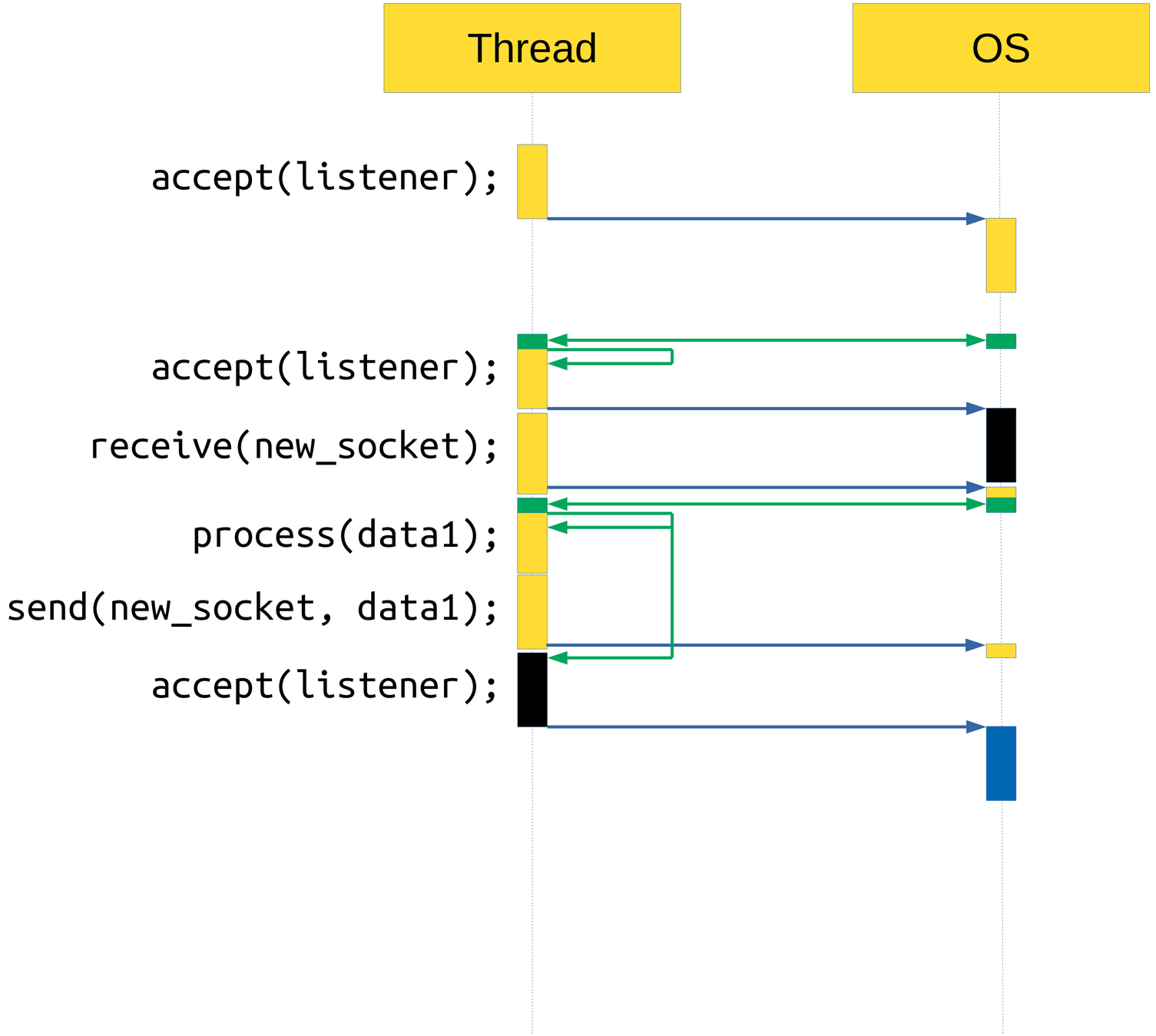

Thread
OS
accept(listener);

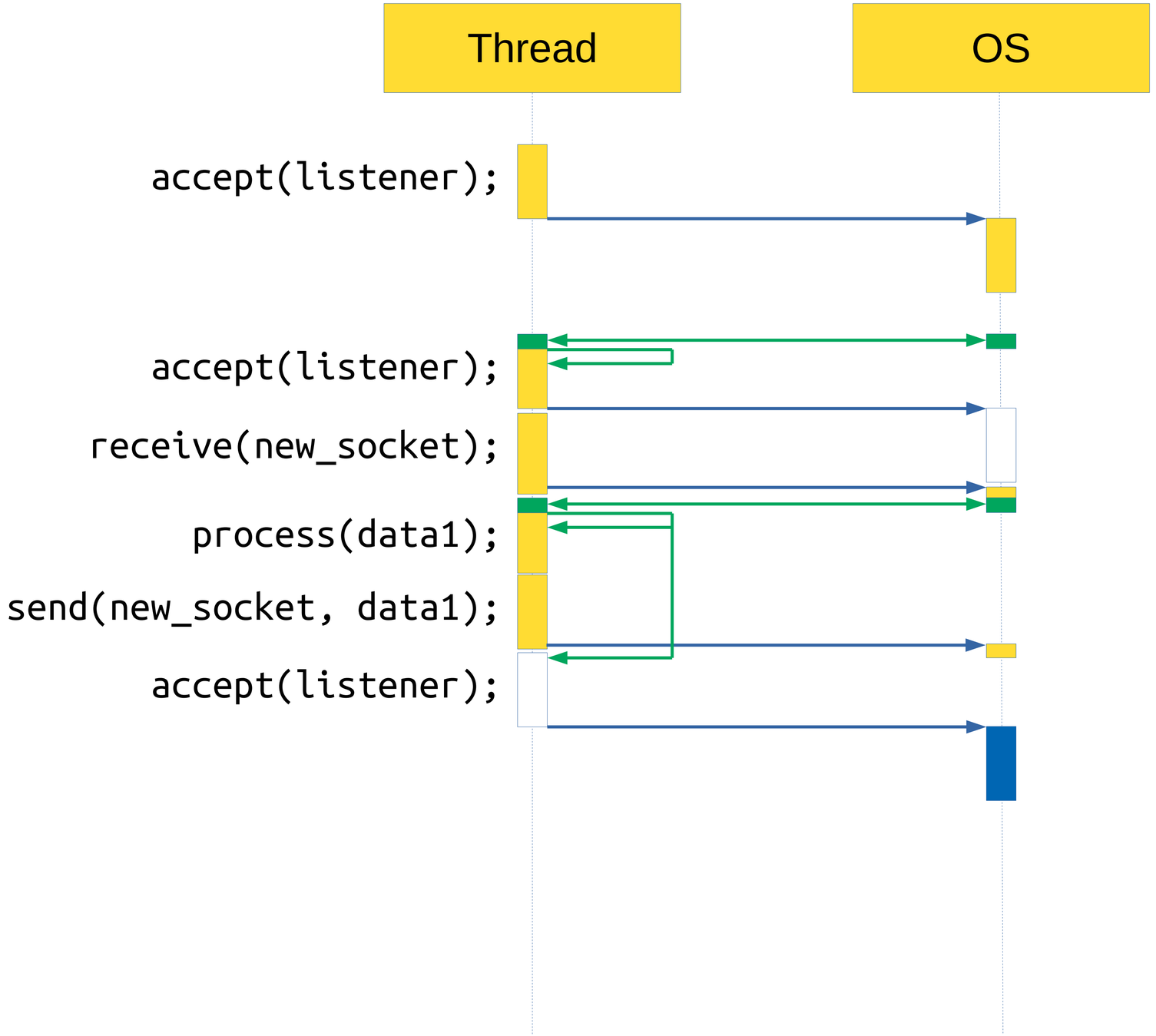

Thread
OS
accept(listener);
accept(listener);

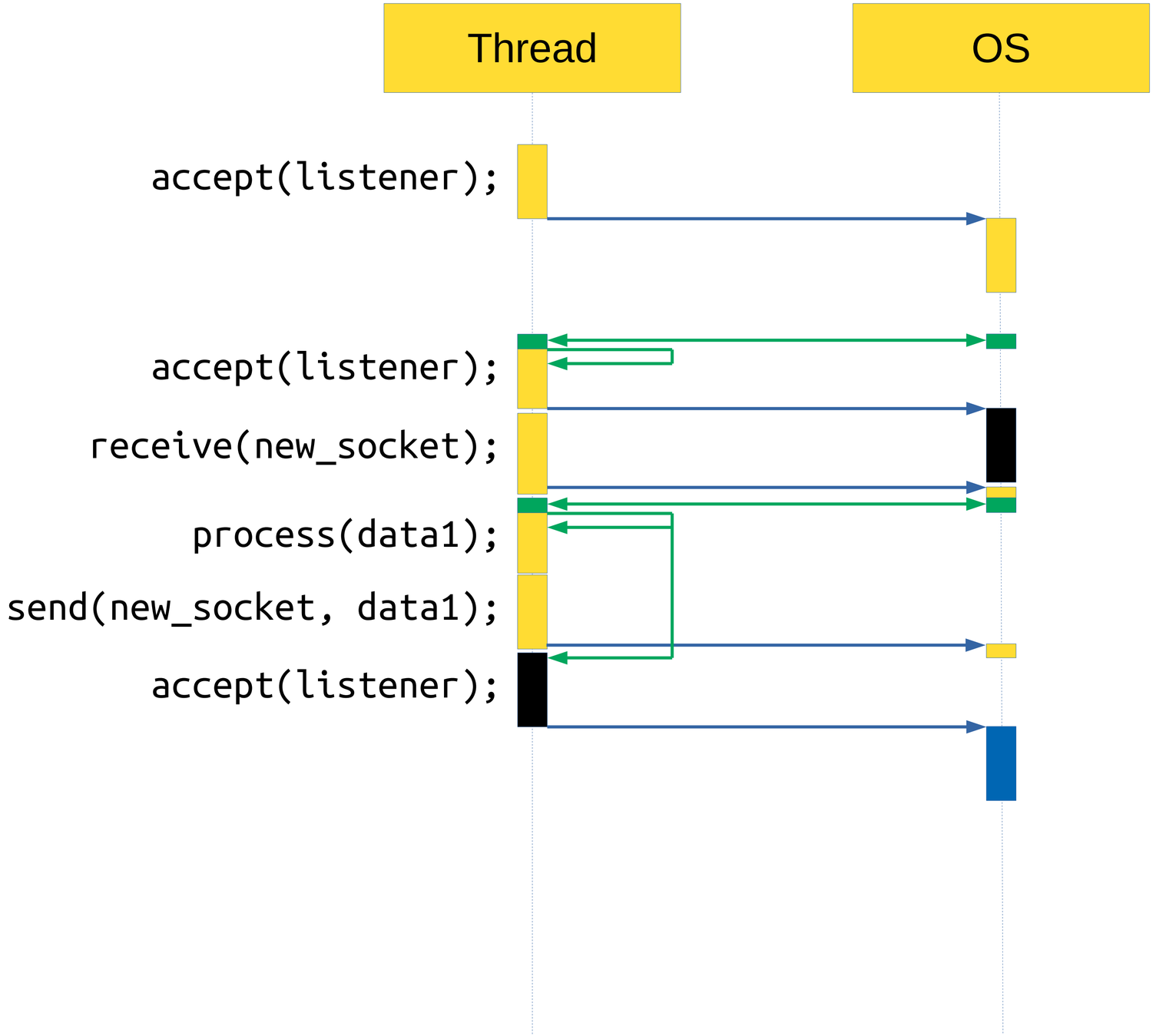

Thread
OS
accept(listener);
accept(listener);

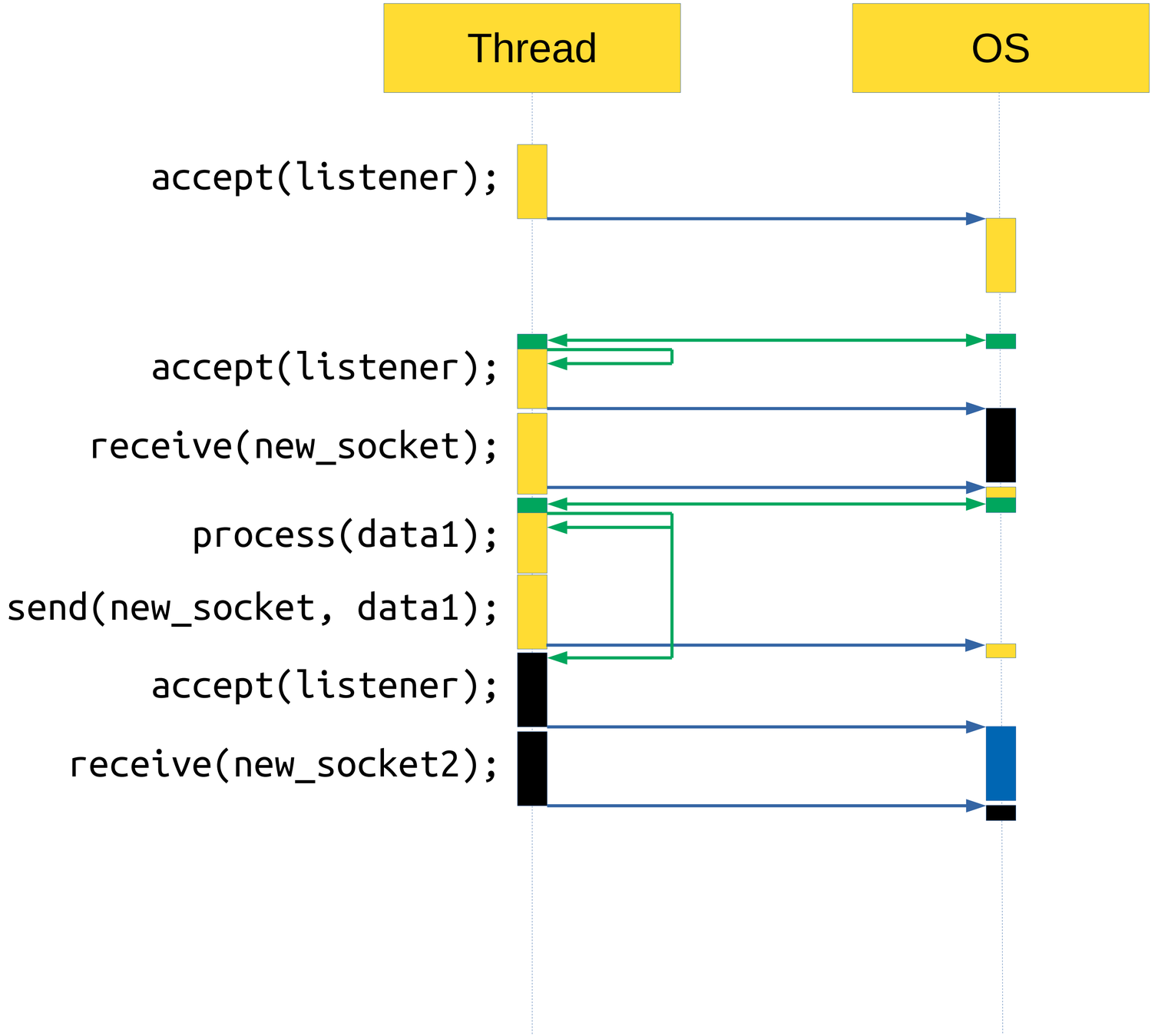

Thread
OS
accept(listener);
accept(listener);
receive(new_socket);


Thread
OS
accept(listener);
accept(listener);
receive(new_socket);


Thread
OS
accept(listener);
accept(listener);
receive(new_socket);
process(data1);
send(new_socket, data1);
accept(listener);


Thread
OS
accept(listener);
accept(listener);
receive(new_socket);
process(data1);
send(new_socket, data1);
accept(listener);


Thread
OS
accept(listener);
accept(listener);
receive(new_socket);
process(data1);
send(new_socket, data1);
accept(listener);


Thread
OS
accept(listener);
accept(listener);
receive(new_socket);
process(data1);
send(new_socket, data1);
accept(listener);
receive(new_socket2);

Плюсы/минусы асинхронности

Плюсы/минусы асинхронности

Плюсы:

Плюсы/минусы асинхронности

Плюсы:

- Всё очень эффективно

Плюсы/минусы асинхронности

Плюсы:

- Всё очень эффективно

Минусы:

Ну давай, прочти меня!

```
void async_accept() {
  accept(listener, [](socket_t socket) {
    async_accept();
    auto something = Async(process1, {42});
    auto& socket_ref = *socket; socket_ref.receive(
        [socket = std::move(socket), something = std::move(something)]
(std::vector<unsigned char> data) mutable {
          auto task = Async(process1, data);
          process(data);
          task.wait();
          auto& socket_ref = *socket; socket_ref.send(data, [data, socket =
std::move(socket), something = std::move(something)]() mutable {
              mutex.lock([data = std::move(data), socket = std::move(socket), something =
std::move(something)]() mutable {
                process2(shared_resource, data);
                socket->send(data, kNoCallback);
              });
          });
});
```

Плюсы/минусы асинхронности

Плюсы:

- Всё очень эффективно

Минусы:

- Нечитаемо...

Корутины спешат на помощь

Корутины ≈ колбеки

```
coro_future coro_accept_stackles() {
  for (;;) {
    auto new_socket = co_await accept(listener);

    auto task = Async([socket = std::move(new_socket)]() -> coro_future {
      auto data = co_await socket.receive();
      process(data);
      co_await socket.send(data);
      co_return;
    });

    task.Detach();
  }
}
```

Корутины ≈ колбеки

```
coro_future coro_accept_stackles() {
  for (;;) {
    auto new_socket = co_await accept(listener);

    auto task = Async([socket = std::move(new_socket)]() -> coro_future {
      auto data = co_await socket.receive();
      process(data);
      co_await socket.send(data);
      co_return;
    });

    task.Detach();
  }
}
```

Корутины ≈ колбеки

```
coro_future coro_accept_stackles() {
  for (;;) {
    auto new_socket = co_await accept(listener);

    auto task = Async([socket = std::move(new_socket)]() -> coro_future {
      auto data = co_await socket.receive();
      process(data);
      co_await socket.send(data);
      co_return;
    });

    task.Detach();
  }
}
```

Корутины ≈ колбеки

```
coro_future coro_accept_stackles() {
  for (;;) {
    auto new_socket = co_await accept(listener);

    auto task = Async([socket = std::move(new_socket)]() -> coro_future {
      auto data = co_await socket.receive();
      process(data);
      co_await socket.send(data);
      co_return;
    });

    task.Detach();
  }
}
```

Корутины ≈ колбеки

```
coro_future coro_accept_stackles() {
  for (;;) {
    auto new_socket = co_await accept(listener);

    auto task = Async([socket = std::move(new_socket)]() -> coro_future {
      auto data = co_await socket.receive();
      process(data);
      co_await socket.send(data);
      co_return;
    });

    task.Detach();
  }
}
```

Корутины ≈ колбеки

```
coro_future coro_accept_stackles() {
  for (;;) {
    auto new_socket = co_await accept(listener);

    auto task = Async([socket = std::move(new_socket)]() -> coro_future {
      auto data = co_await socket.receive();
      process(data);
      co_await socket.send(data);
      co_return;
    });

    task.Detach();
  }
}
```

Асинхронный сервер с корутинами vs синхронный

```
coro_future coro_accept_stackles() {
  for (;;) {
    auto new_socket = co_await accept(listener);

    auto task = Async(/*...*/ {
      auto data = co_await socket.receive();
      process(data);
      co_await socket.send(data);
      co_return;
    });

    task.Detach();
  }
}
```

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd(/*...*/ {
      auto data = socket.receive();
      process(data);
      socket.send(data);

    });

    thrd.detach();
  }
}
```

Асинхронный сервер с корутинами vs синхронный

```
coro_future coro_accept_stackles() {
  for (;;) {
    auto new_socket = co_await accept(listener);

    auto task = Async(/*...*/ {
      auto data = co_await socket.receive();
      process(data);
      co_await socket.send(data);
      co_return;
    });

    task.Detach();
  }
}
```

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd(/*...*/ {
      auto data = socket.receive();
      process(data);
      socket.send(data);

    });

    thrd.detach();
  }
}
```

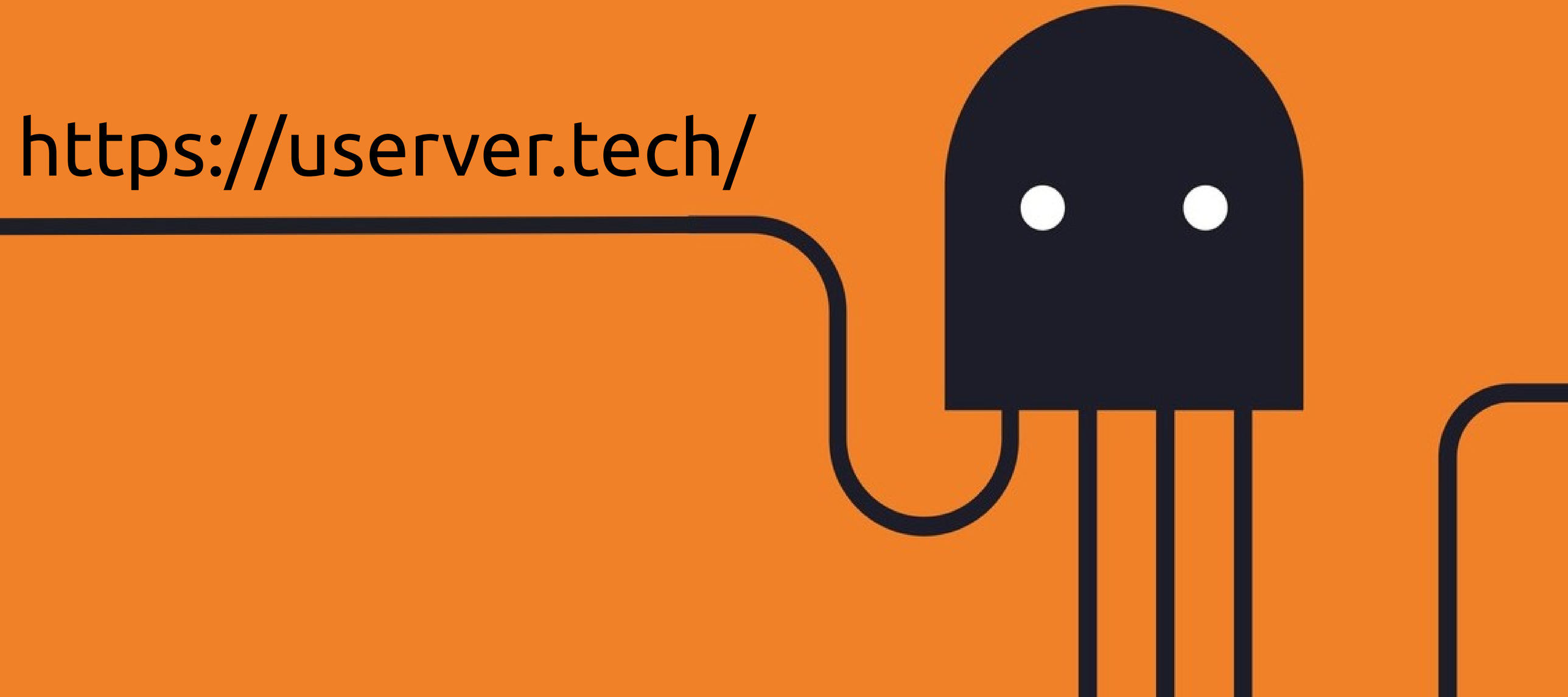
Асинхронный сервер с корутинами vs синхронный

```
void coro_accept_stackfull() {
  for (;;) {
    auto new_socket = accept(listener);

    auto task = Async(/*...*/ {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    task.Detach();
  }
}
```

```
void naive_accept() {
  for (;;) {
    auto new_socket = accept(listener);

    std::thread thrd(/*...*/ {
      auto data = socket.receive();
      process(data);
      socket.send(data);
    });

    thrd.detach();
  }
}
```

Продумали. Нужен фреймворк

https://userver.tech/



userver

userver

- C++

userver

- C++

  эффективный язык, чтобы логика могла не «тормозить»

userver

- C++

  эффективный язык, чтобы логика могла не «тормозить»

- Асинхронная работа

userver

- C++

  эффективный язык, чтобы логика могла не «тормозить»

- Асинхронная работа

  чтобы побороть C10K

userver

- C++

  эффективный язык, чтобы логика могла не «тормозить»

- Асинхронная работа

  чтобы побороть C10K

- Корутины

userver

- C++

  эффективный язык, чтобы логика могла не «тормозить»

- Асинхронная работа

  чтобы побороть C10K

- Корутины

  чтобы код выглядел линейно и его было просто писать

userver

- C++

  эффективный язык, чтобы логика могла не «тормозить»

- Асинхронная работа

  чтобы побороть C10K

- Корутины

  чтобы код выглядел линейно и его было просто писать

- Обширный функционал из коробки

userver

- C++

  эффективный язык, чтобы логика могла не «тормозить»

- Асинхронная работа

  чтобы побороть C10K

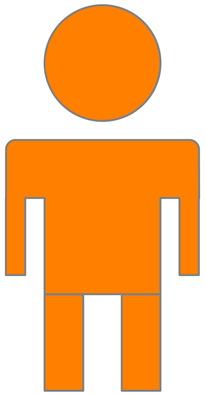
- Корутины

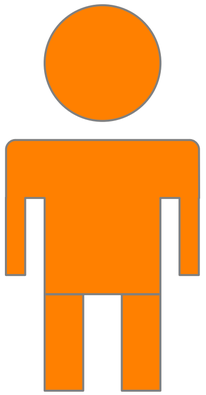
  чтобы код выглядел линейно и его было просто писать

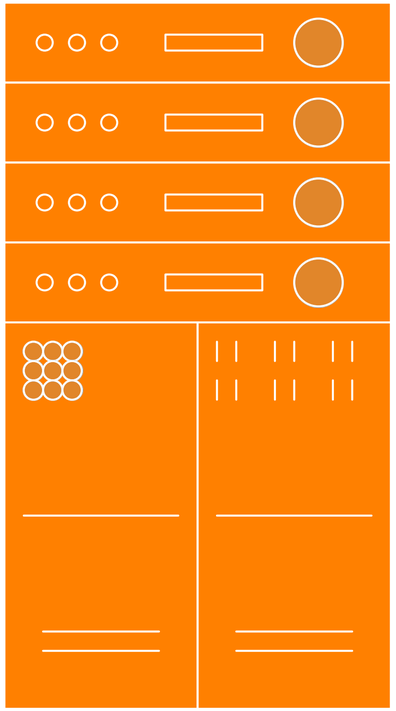
- Обширный функционал из коробки

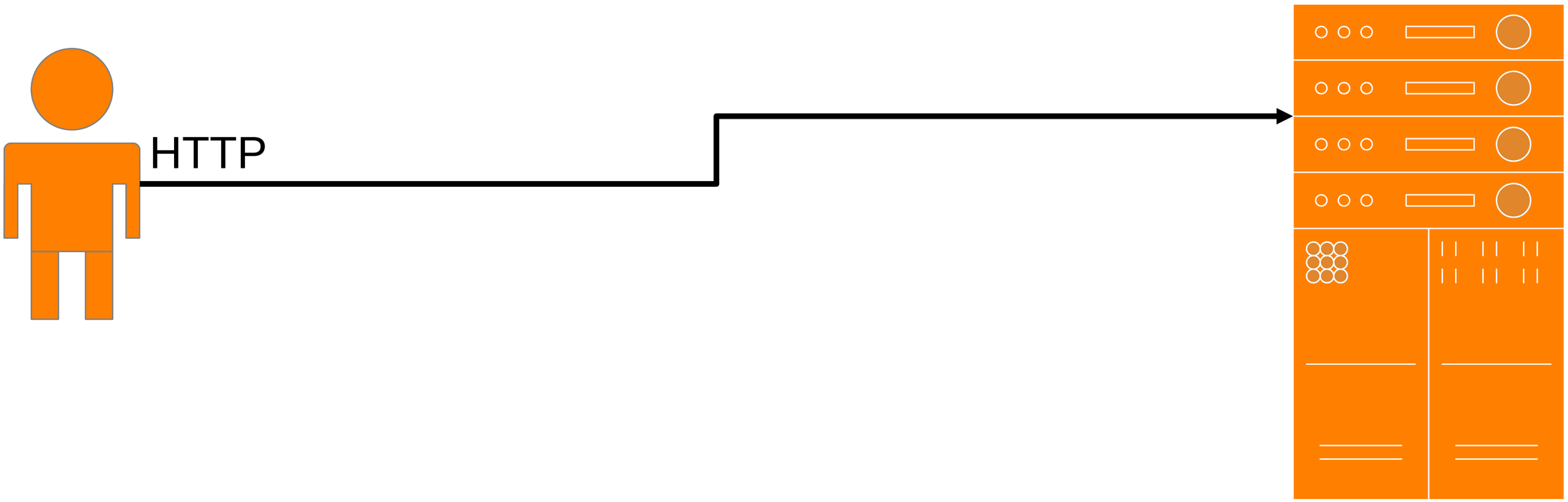
  чтобы быстро писать код

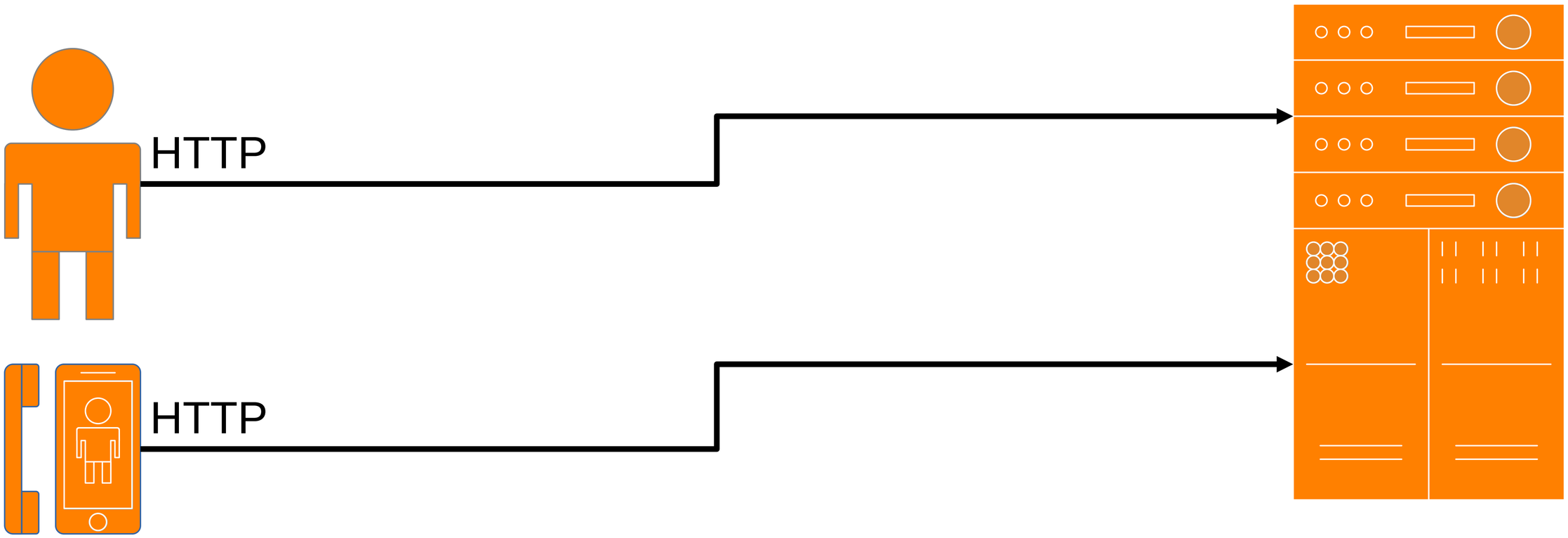
Hello world – введение

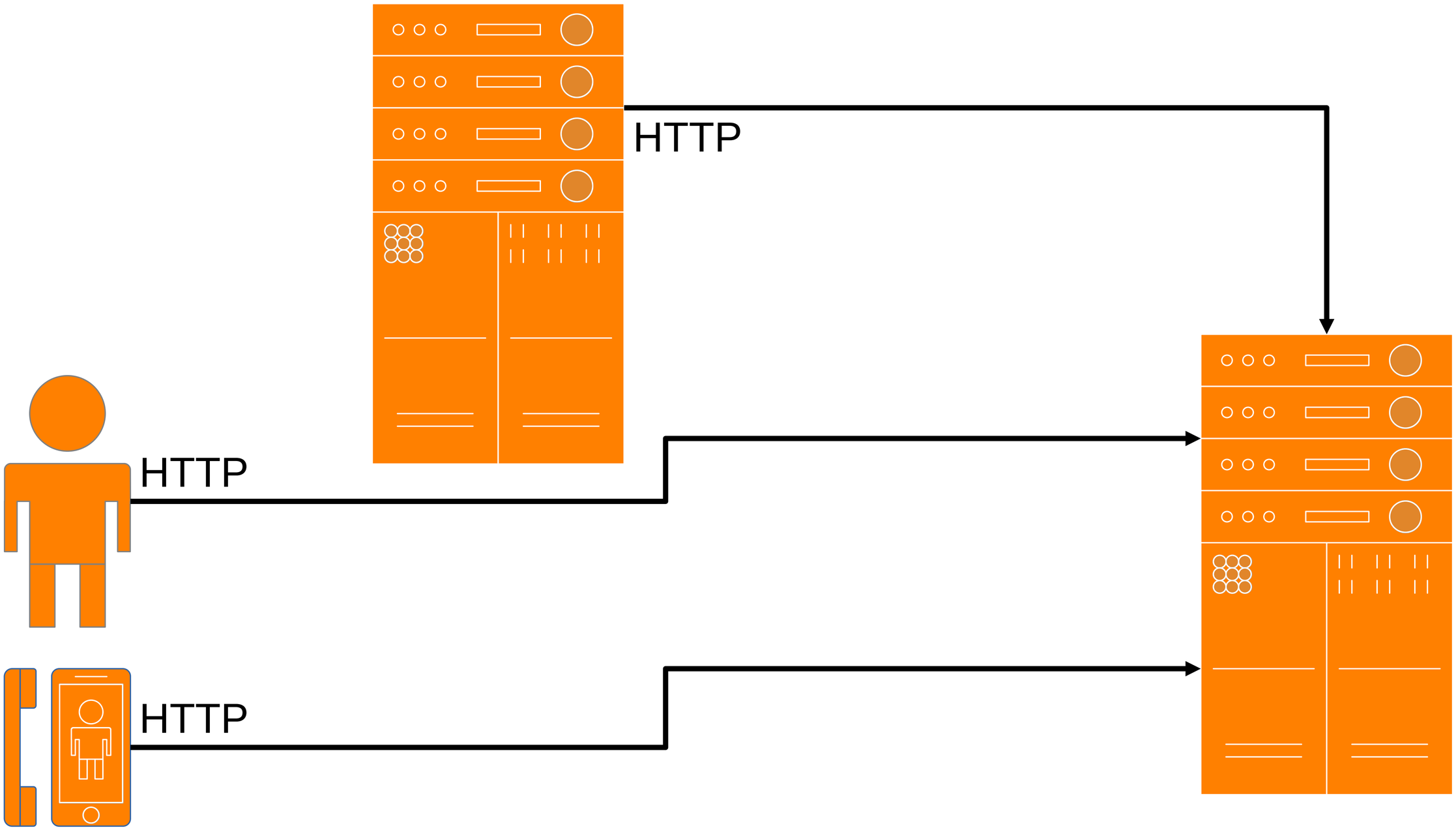


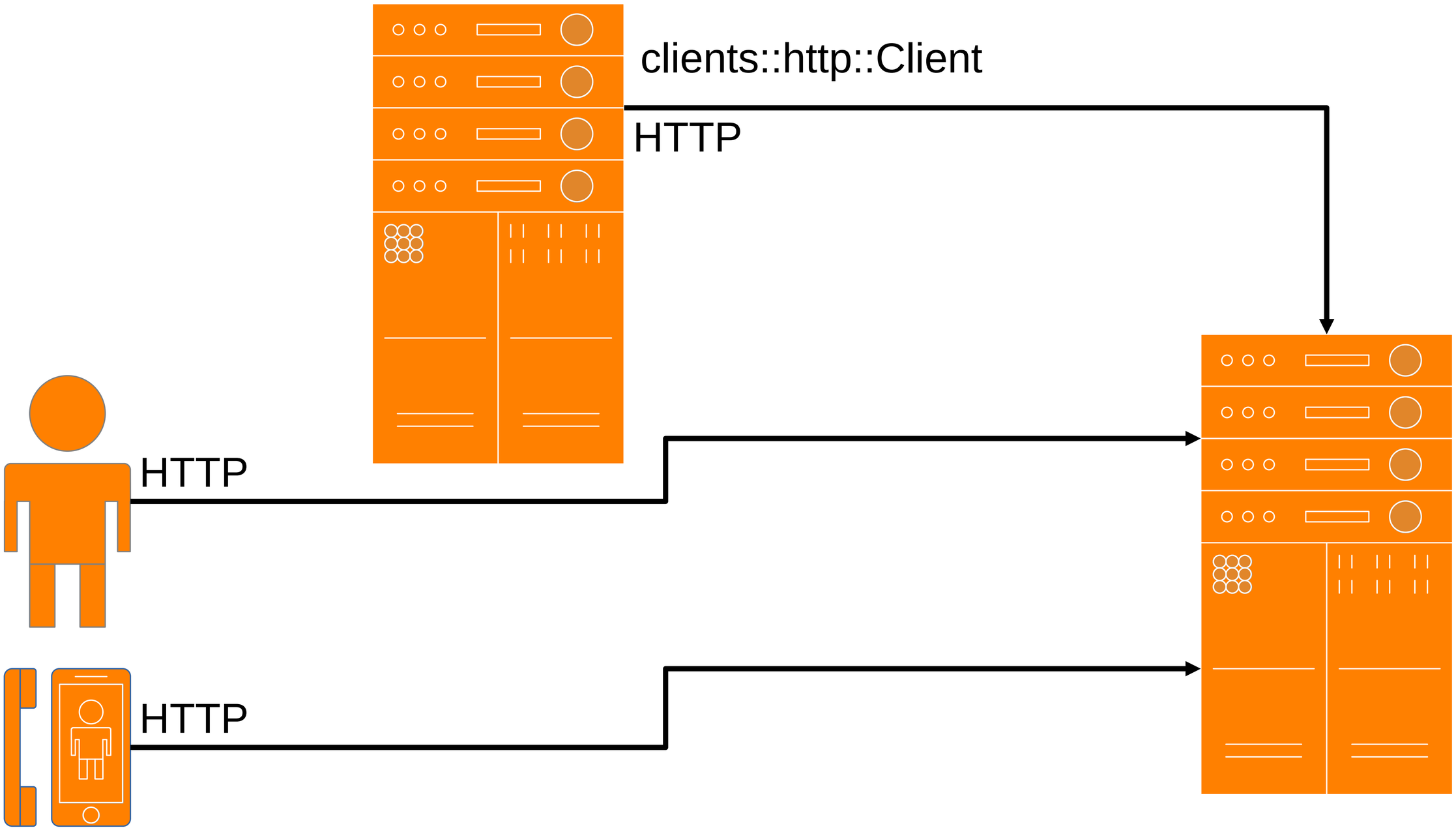


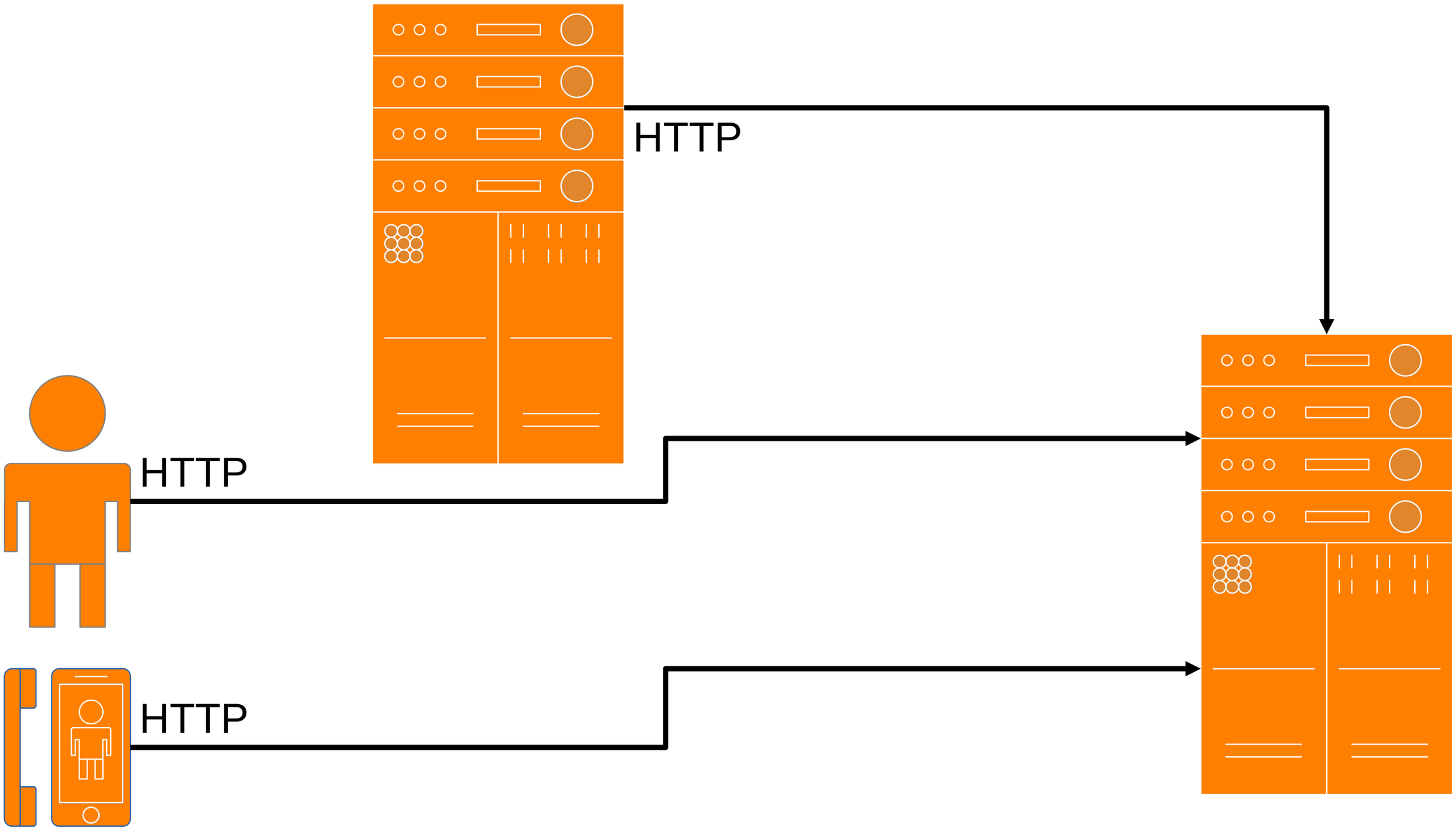


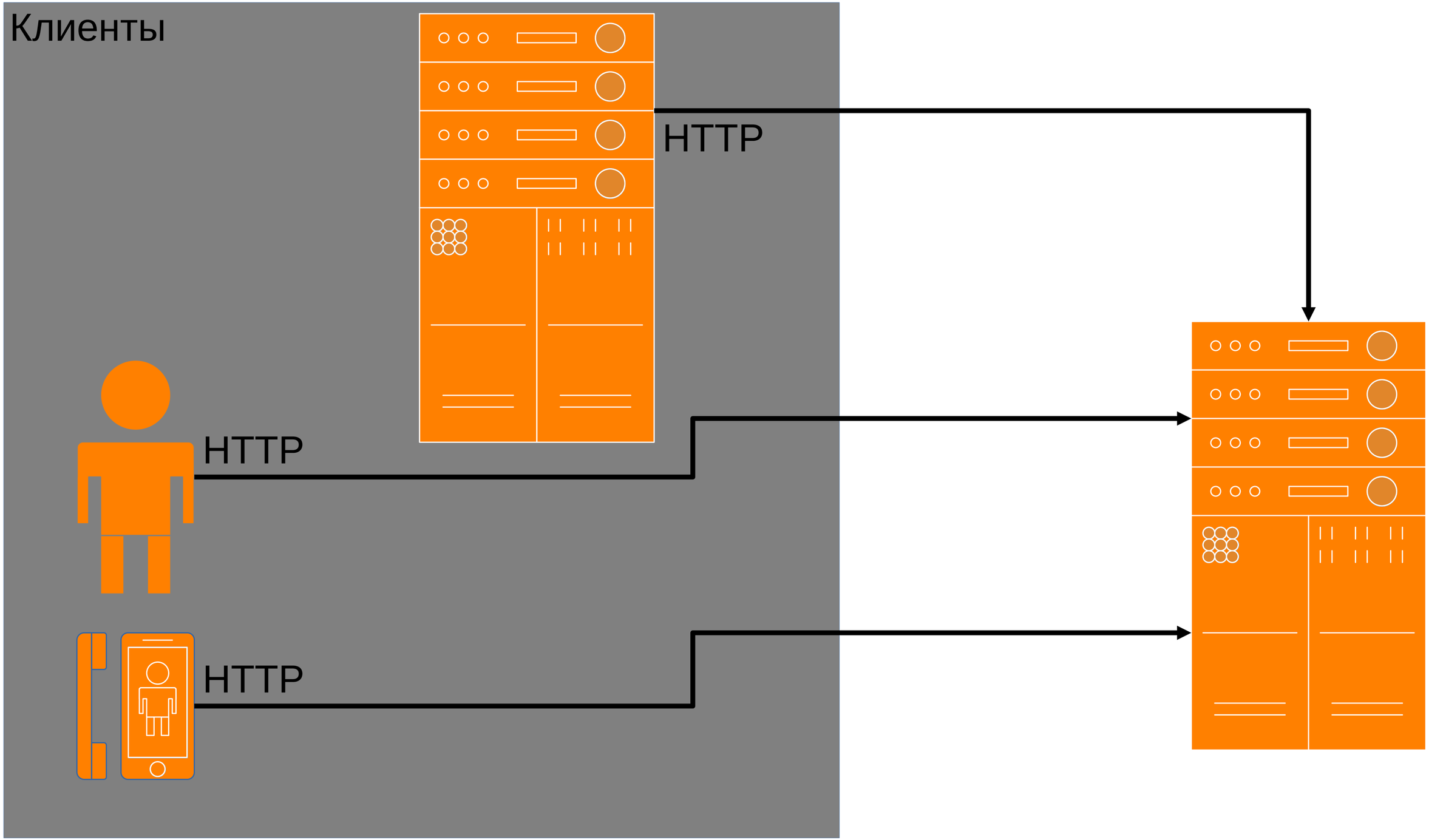

HTTP

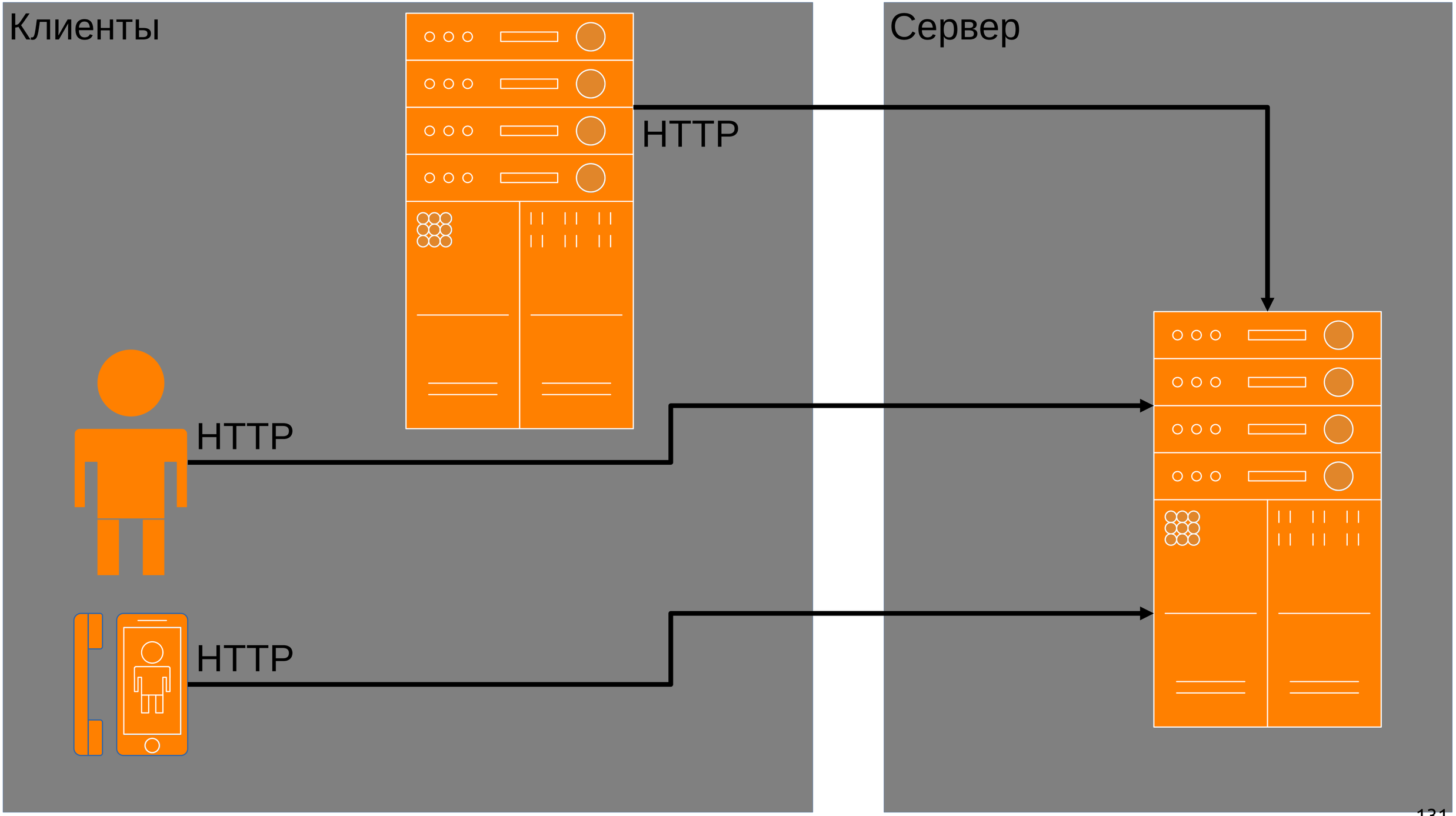

HTTP
HTTP


HTTP
HTTP
HTTP


clients::http::Client
HTTP
HTTP
HTTP


HTTP
HTTP
HTTP


Клиенты
HTTP
HTTP
HTTP


Клиенты
Сервер
HTTP
HTTP
HTTP

HTTP введение

HTTP введение

```
$ curl -v ya.ru/test
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*

< HTTP/1.1 404 Not found
< Content-Type: text/html; charset=UTF-8
< Date: Thu, 29 Sep 2022 13:46:05 GMT
< Last-Modified: Thu, 29 Sep 2022 13:46:05 GMT
< Content-Length: 9123
<
< <!DOCTYPE html><html   class="i-ua_js_no i-ua_css_standart i-ua_browser_unknown i-
ua_browser-engine_unknown i-ua_browser_desktop i-ua_platform_other"  lang="ru"><head
xmlns:og="http://ogp.me/ns#"><meta name="description" content="Яндекс — поисковая система и
интернет-портал. Поиск по интернету и другие сервисы: карты и навигатор, транспорт и такси,
погода, новости, музыка, телепрограмма, переводчик, покупки в интернете. Бесплатная
электронная почта и облачное хранилище. Найдется всё!"><meta property="og:title"
content="Яндекс"> ...
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*

< HTTP/1.1 404 Not found
< Content-Type: text/html; charset=UTF-8
< Date: Thu, 29 Sep 2022 13:46:05 GMT
< Last-Modified: Thu, 29 Sep 2022 13:46:05 GMT
< Content-Length: 9123
<
< <!DOCTYPE html><html   class="i-ua_js_no i-ua_css_standart i-ua_browser_unknown i-
ua_browser-engine_unknown i-ua_browser_desktop i-ua_platform_other"  lang="ru"><head
xmlns:og="http://ogp.me/ns#"><meta name="description" content="Яндекс — поисковая система и
интернет-портал. Поиск по интернету и другие сервисы: карты и навигатор, транспорт и такси,
погода, новости, музыка, телепрограмма, переводчик, покупки в интернете. Бесплатная
электронная почта и облачное хранилище. Найдется всё!"><meta property="og:title"
content="Яндекс"> ...
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*

< HTTP/1.1 404 Not found
< Content-Type: text/html; charset=UTF-8
< Date: Thu, 29 Sep 2022 13:46:05 GMT
< Last-Modified: Thu, 29 Sep 2022 13:46:05 GMT
< Content-Length: 9123
<
< <!DOCTYPE html><html   class="i-ua_js_no i-ua_css_standart i-ua_browser_unknown i-
ua_browser-engine_unknown i-ua_browser_desktop i-ua_platform_other"  lang="ru"><head
xmlns:og="http://ogp.me/ns#"><meta name="description" content="Яндекс — поисковая система и
интернет-портал. Поиск по интернету и другие сервисы: карты и навигатор, транспорт и такси,
погода, новости, музыка, телепрограмма, переводчик, покупки в интернете. Бесплатная
электронная почта и облачное хранилище. Найдется всё!"><meta property="og:title"
content="Яндекс"> ...
```

HTTP введение

```
$ curl -v ya.ru/test

> GET /test HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*

< HTTP/1.1 404 Not found
< Content-Type: text/html; charset=UTF-8
< Date: Thu, 29 Sep 2022 13:46:05 GMT
< Last-Modified: Thu, 29 Sep 2022 13:46:05 GMT
< Content-Length: 9123
<
< <!DOCTYPE html><html   class="i-ua_js_no i-ua_css_standart i-ua_browser_unknown i-
ua_browser-engine_unknown i-ua_browser_desktop i-ua_platform_other"  lang="ru"><head
xmlns:og="http://ogp.me/ns#"><meta name="description" content="Яндекс — поисковая система и
интернет-портал. Поиск по интернету и другие сервисы: карты и навигатор, транспорт и такси,
погода, новости, музыка, телепрограмма, переводчик, покупки в интернете. Бесплатная
электронная почта и облачное хранилище. Найдется всё!"><meta property="og:title"
content="Яндекс"> ...
```

HTTP введение

HTTP введение

```
$ curl -v -X POST -d 'Hello' ya.ru/hello
```

HTTP введение

```
$ curl -v -X POST -d 'Hello' ya.ru/hello

> POST /hello HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
> Content-Length: 5
> Content-Type: application/x-www-form-urlencoded
>
> Hello
```

HTTP введение

```
$ curl -v -X POST -d 'Hello' ya.ru/hello

> POST /hello HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
> Content-Length: 5
> Content-Type: application/x-www-form-urlencoded
>
> Hello
```

HTTP введение

```
$ curl -v -X POST -d 'Hello' ya.ru/hello

> POST /hello HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
> Content-Length: 5
> Content-Type: application/x-www-form-urlencoded
>
> Hello
```

HTTP введение

```
$ curl -v -X POST -d 'Hello' ya.ru/hello

> POST /hello HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
> Content-Length: 5
> Content-Type: application/x-www-form-urlencoded
>
> Hello
```

HTTP введение

```
$ curl -v -X POST -d 'Hello' ya.ru/hello

> POST /hello HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
> Content-Length: 5
> Content-Type: application/x-www-form-urlencoded
>
> Hello

< HTTP/1.1 302 Moved temporarily
< Content-Length: 0
< Location: https://ya.ru/hello
```

HTTP введение

```
$ curl -v -X POST -d 'Hello' ya.ru/hello

> POST /hello HTTP/1.1
> Host: ya.ru
> User-Agent: curl/7.81.0
> Accept: */*
> Content-Length: 5
> Content-Type: application/x-www-form-urlencoded
>
> Hello

< HTTP/1.1 302 Moved temporarily
< Content-Length: 0
< Location: https://ya.ru/hello
```

Hello world

Hello world

```cpp
#include <userver/components/minimal_server_component_list.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/utils/daemon_run.hpp>

struct Hello final : public server::handlers::HttpHandlerBase {
  static constexpr std::string_view kName = "handler-hello-sample";

  using HttpHandlerBase::HttpHandlerBase;

  std::string HandleRequestThrow(
      const server::http::HttpRequest&,
      server::request::RequestContext&) const override {
    return "Hello world!\n";
  }
};

int main(int argc, char* argv[]) {
  const auto component_list =
      components::MinimalServerComponentList().Append<Hello>();
  return utils::DaemonMain(argc, argv, component_list);
}
```

Hello world

```cpp
#include <userver/components/minimal_server_component_list.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/utils/daemon_run.hpp>

struct Hello final : public server::handlers::HttpHandlerBase {
  static constexpr std::string_view kName = "handler-hello-sample";

  using HttpHandlerBase::HttpHandlerBase;

  std::string HandleRequestThrow(
      const server::http::HttpRequest&,
      server::request::RequestContext&) const override {
    return "Hello world!\n";
  }
};

int main(int argc, char* argv[]) {
  const auto component_list =
      components::MinimalServerComponentList().Append<Hello>();
  return utils::DaemonMain(argc, argv, component_list);
}
```

Hello world

```
#include <userver/components/minimal_server_component_list.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/utils/daemon_run.hpp>

struct Hello final : public server::handlers::HttpHandlerBase {
  static constexpr std::string_view kName = "handler-hello-sample";

  using HttpHandlerBase::HttpHandlerBase;

  std::string HandleRequestThrow(
      const server::http::HttpRequest&,
      server::request::RequestContext&) const override {
    return "Hello world!\n";
  }
};

int main(int argc, char* argv[]) {
  const auto component_list =
      components::MinimalServerComponentList().Append<Hello>();
  return utils::DaemonMain(argc, argv, component_list);
}
```

Hello world

```
#include <userver/components/minimal_server_component_list.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/utils/daemon_run.hpp>

struct Hello final : public server::handlers::HttpHandlerBase {
  static constexpr std::string_view kName = "handler-hello-sample";

  using HttpHandlerBase::HttpHandlerBase;

  std::string HandleRequestThrow(
      const server::http::HttpRequest&,
      server::request::RequestContext&) const override {
    return "Hello world!\n";
  }
};

int main(int argc, char* argv[]) {
  const auto component_list =
      components::MinimalServerComponentList().Append<Hello>();
  return utils::DaemonMain(argc, argv, component_list);
}
```

Hello world

```cpp
#include <userver/components/minimal_server_component_list.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/utils/daemon_run.hpp>

struct Hello final : public server::handlers::HttpHandlerBase {
  static constexpr std::string_view kName = "handler-hello-sample";

  using HttpHandlerBase::HttpHandlerBase;

  std::string HandleRequestThrow(
      const server::http::HttpRequest&,
      server::request::RequestContext&) const override {
    return "Hello world!\n";
  }
};

int main(int argc, char* argv[]) {
  const auto component_list =
      components::MinimalServerComponentList().Append<Hello>();
  return utils::DaemonMain(argc, argv, component_list);
}
```

Hello world

```cpp
#include <userver/components/minimal_server_component_list.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/utils/daemon_run.hpp>

struct Hello final : public server::handlers::HttpHandlerBase {
  static constexpr std::string_view kName = "handler-hello-sample";

  using HttpHandlerBase::HttpHandlerBase;

  std::string HandleRequestThrow(
      const server::http::HttpRequest&,
      server::request::RequestContext&) const override {
    return "Hello world!\n";
  }
};

int main(int argc, char* argv[]) {
  const auto component_list =
      components::MinimalServerComponentList().Append<Hello>();
  return utils::DaemonMain(argc, argv, component_list);
}
```

Hello world - тесты

Hello world - тесты

```
async def test_ping(service_client):
    response = await service_client.get('/hello')
    assert response.status == 200
    assert response.content == b'Hello world!\n'
```

Hello world - тесты

```
async def test_ping(service_client):
    response = await service_client.get('/hello')
    assert response.status == 200
    assert response.content == b'Hello world!\n'
```

Hello world - тесты

```
async def test_ping(service_client):
    response = await service_client.get('/hello')
    assert response.status == 200
    assert response.content == b'Hello world!\n'
```

Hello world - тесты

```
async def test_ping(service_client):
    response = await service_client.get('/hello')
    assert response.status == 200
    assert response.content == b'Hello world!\n'
```

Hello world - тесты

```
async def test_ping(service_client):
    response = await service_client.get('/hello')
    assert response.status == 200
    assert response.content == b'Hello world!\n'
```

Hello world - конфиги

Hello world - конфиги

```yaml
components_manager:
    coro_pool:
        initial_size: 500             # Preallocate 500 coroutines at startup.
        max_size: 1000                # Do not keep more than 1000 preallocated coroutines.
    task_processors:                  # Task processor is an executor for coroutine tasks
        main-task-processor:          # Task processor for CPU-bound couroutine tasks.
            worker_threads: 4         # Process tasks in 4 threads.
            thread_name: main-worker
        fs-task-processor:            # Make a separate task processor for FS task.
            thread_name: fs-worker
            worker_threads: 4
    default_task_processor: main-task-processor
    components:                       # Components that were registered via component_list
        # ...
```

Hello world - конфиги

```yaml
components_manager:
    coro_pool:
        initial_size: 500             # Preallocate 500 coroutines at startup.
        max_size: 1000                # Do not keep more than 1000 preallocated coroutines.
    task_processors:                  # Task processor is an executor for coroutine tasks
        main-task-processor:          # Task processor for CPU-bound couroutine tasks.
            worker_threads: 4         # Process tasks in 4 threads.
            thread_name: main-worker
        fs-task-processor:            # Make a separate task processor for FS task.
            thread_name: fs-worker
            worker_threads: 4
    default_task_processor: main-task-processor
    components:                       # Components that were registered via component_list
        # ...
```

Hello world - конфиги

```
components_manager:
    coro_pool:
        initial_size: 500             # Preallocate 500 coroutines at startup.
        max_size: 1000                # Do not keep more than 1000 preallocated coroutines.
    task_processors:                  # Task processor is an executor for coroutine tasks
        main-task-processor:          # Task processor for CPU-bound couroutine tasks.
            worker_threads: 4         # Process tasks in 4 threads.
            thread_name: main-worker
        fs-task-processor:            # Make a separate task processor for FS task.
            thread_name: fs-worker
            worker_threads: 4
    default_task_processor: main-task-processor
    components:                       # Components that were registered via component_list
        # ...
```

Hello world - конфиги

```
components_manager:
    coro_pool:
        initial_size: 500             # Preallocate 500 coroutines at startup.
        max_size: 1000                # Do not keep more than 1000 preallocated coroutines.
    task_processors:                  # Task processor is an executor for coroutine tasks
        main-task-processor:          # Task processor for CPU-bound couroutine tasks.
            worker_threads: 4         # Process tasks in 4 threads.
            thread_name: main-worker
        fs-task-processor:            # Make a separate task processor for FS task.
            thread_name: fs-worker
            worker_threads: 4
    default_task_processor: main-task-processor
    components:                       # Components that were registered via component_list
        # ...
```

Hello world - конфиги

```
components_manager:
    coro_pool:
        initial_size: 500             # Preallocate 500 coroutines at startup.
        max_size: 1000                # Do not keep more than 1000 preallocated coroutines.
    task_processors:                  # Task processor is an executor for coroutine tasks
        main-task-processor:          # Task processor for CPU-bound couroutine tasks.
            worker_threads: 4         # Process tasks in 4 threads.
            thread_name: main-worker
        fs-task-processor:            # Make a separate task processor for FS task.
            thread_name: fs-worker
            worker_threads: 4
    default_task_processor: main-task-processor
    components:                       # Components that were registered via component_list
        # ...
```

Hello world - конфиги

```yaml
components_manager:
    coro_pool:
        initial_size: 500             # Preallocate 500 coroutines at startup.
        max_size: 1000                # Do not keep more than 1000 preallocated coroutines.
    task_processors:                  # Task processor is an executor for coroutine tasks
        main-task-processor:          # Task processor for CPU-bound couroutine tasks.
            worker_threads: 4         # Process tasks in 4 threads.
            thread_name: main-worker
        fs-task-processor:            # Make a separate task processor for FS task.
            thread_name: fs-worker
            worker_threads: 4
    default_task_processor: main-task-processor
    components:                       # Components that were registered via component_list
        # ...
```

Hello world - конфиги

```yaml
components_manager:
    coro_pool:
        initial_size: 500             # Preallocate 500 coroutines at startup.
        max_size: 1000                # Do not keep more than 1000 preallocated coroutines.
    task_processors:                  # Task processor is an executor for coroutine tasks
        main-task-processor:          # Task processor for CPU-bound couroutine tasks.
            worker_threads: 4         # Process tasks in 4 threads.
            thread_name: main-worker
        fs-task-processor:            # Make a separate task processor for FS task.
            thread_name: fs-worker
            worker_threads: 4
    default_task_processor: main-task-processor
    components:                       # Components that were registered via component_list
        # ...
```

Hello world - компоненты

Компоненты

Компоненты

Приложение «собирается» из различных компонент.

Компоненты

Приложение «собирается» из различных компонент.

Компоненты:

Компоненты

Приложение «собирается» из различных компонент.

Компоненты:

- Работают со статическим конфигом

Компоненты

Приложение «собирается» из различных компонент.

Компоненты:

- Работают со статическим конфигом
- Общаются с другими компонентами

Компоненты

Приложение «собирается» из различных компонент.

Компоненты:

- Работают со статическим конфигом
- Общаются с другими компонентами
- Предоставляют «клиентов» к компоненту

Компоненты

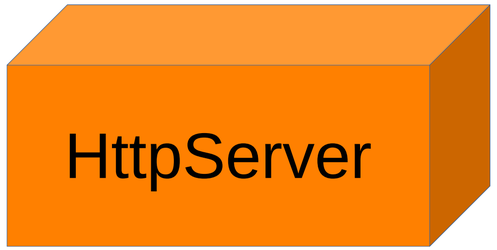
Приложение «собирается» из различных компонент.

Компоненты:

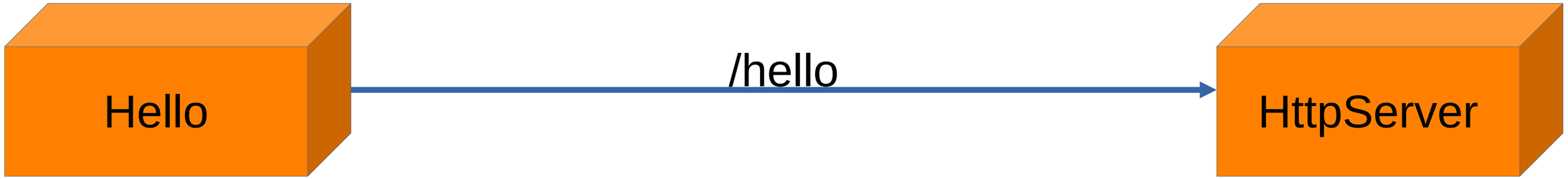
- Работают со статическим конфигом
- Общаются с другими компонентами
- Предоставляют «клиентов» к компоненту
- Гарантируют времена жизни зависимостей

Компоненты

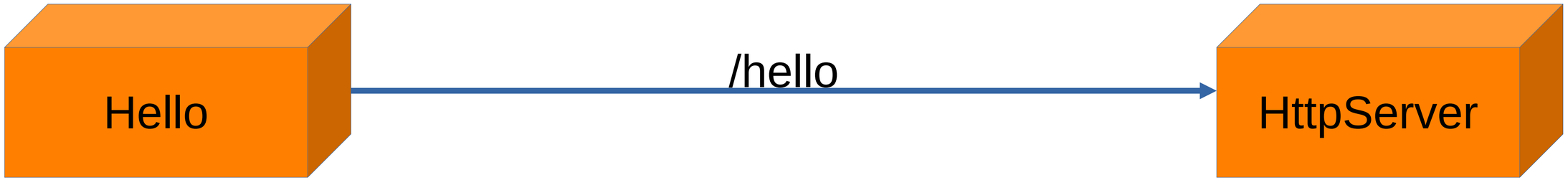
Компоненты



Компоненты



Компоненты



a::Client

a::Client::Client(b::Client& client, a::Options config)

a::Client::Client(b::Client& **client**, a::Options config)

b::Client

a::Client::Client(b::Client& client, a::Options config)

b::Client::Client(b::Options config)

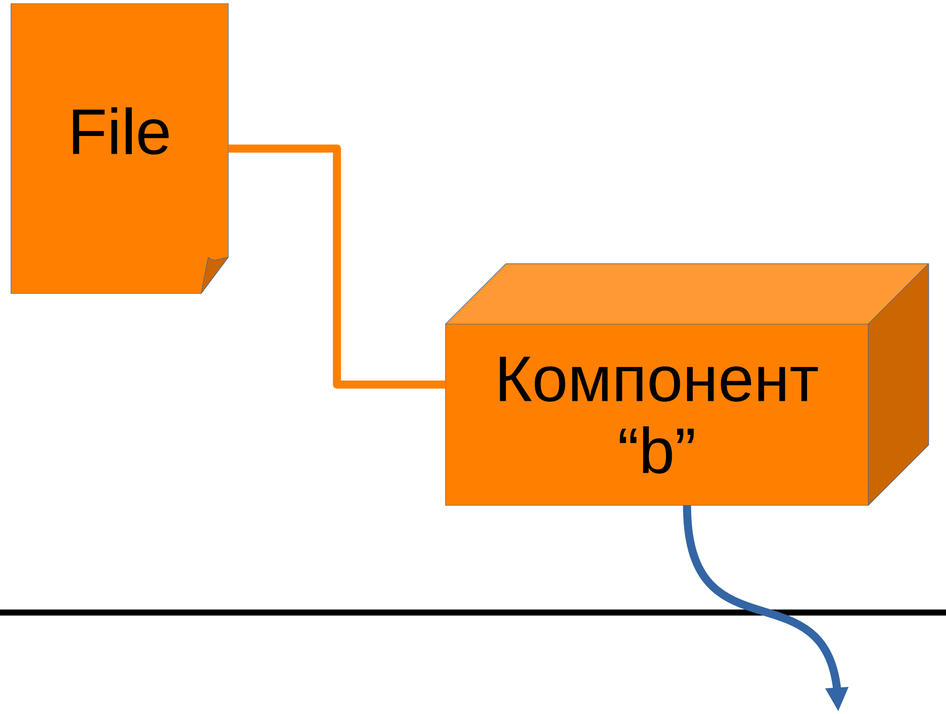
a::Client::Client(b::Client& client, a::Options config)

b::Client::Client(b::Options config)

a::Client::Client(b::Client& client, a::Options config)

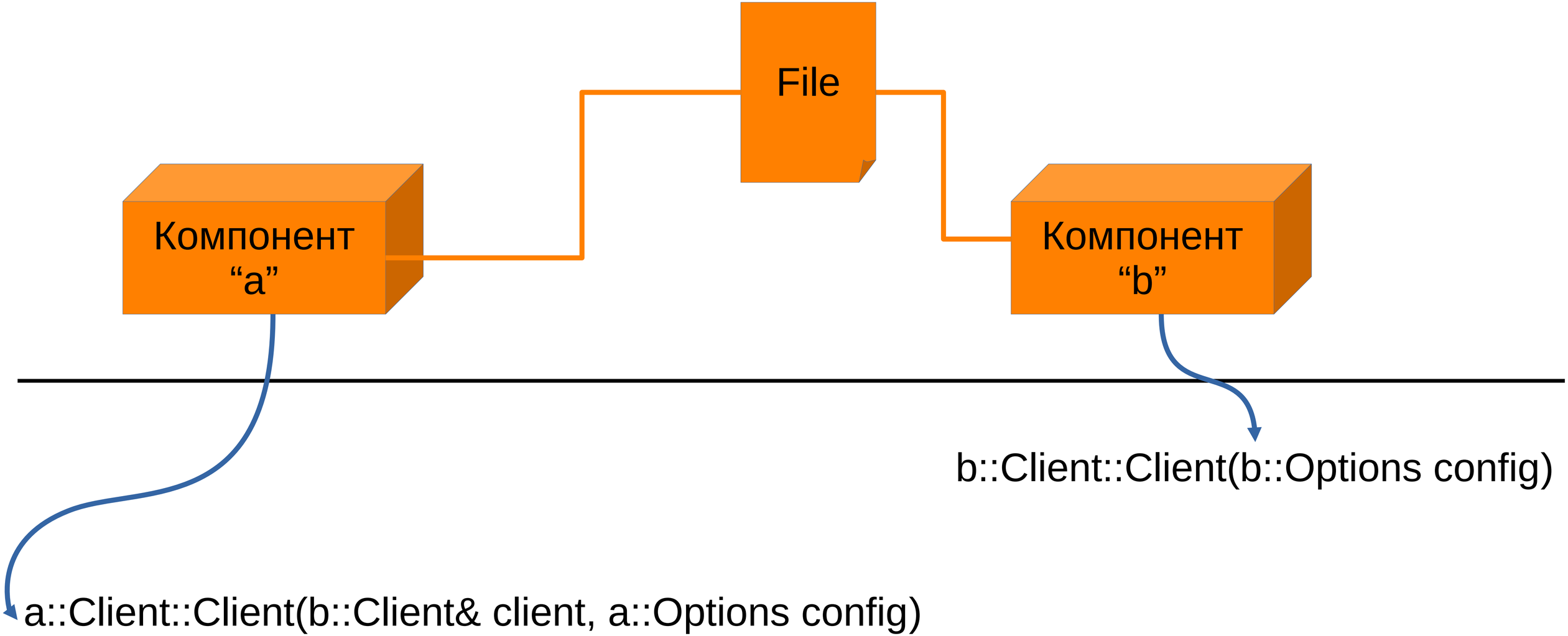
b::Client::Client(b::Options **config**)

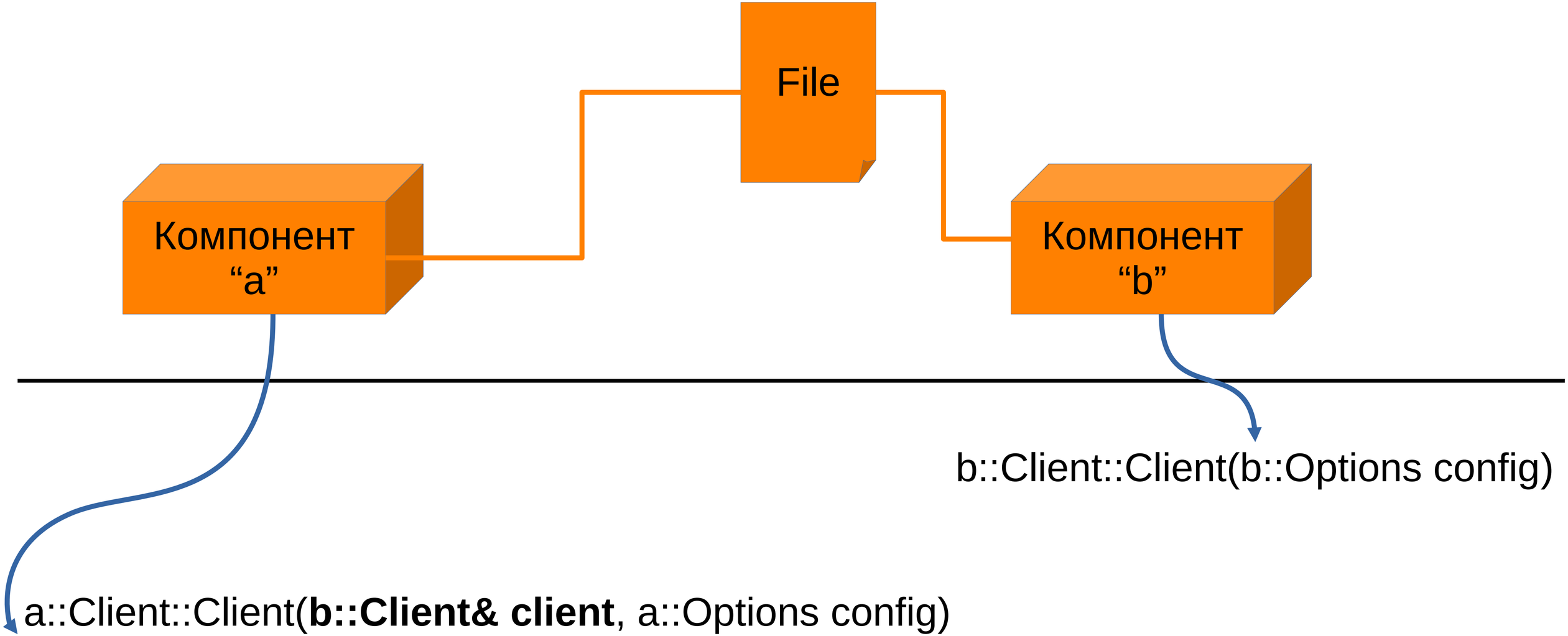
a::Client::Client(b::Client& client, a::Options **config**)

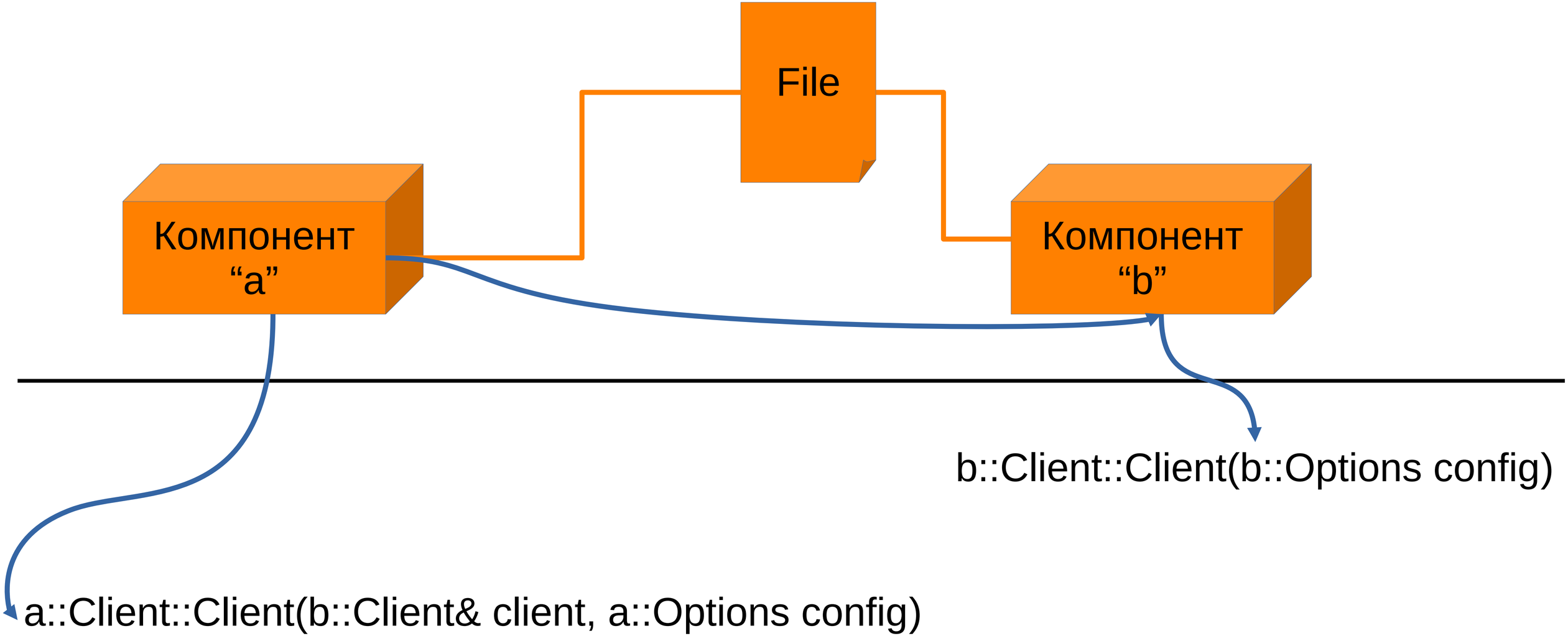


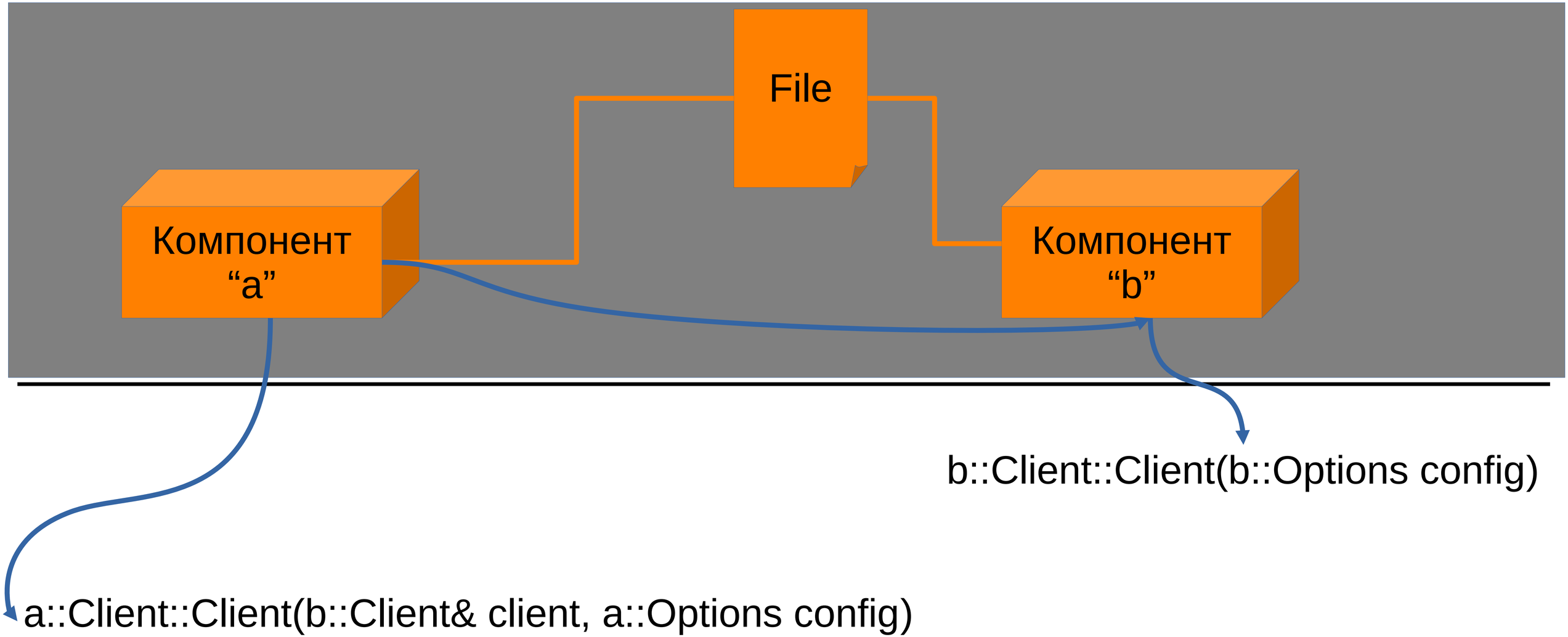
b::Client::Client(b::Options config)

a::Client::Client(b::Client& client, a::Options config)

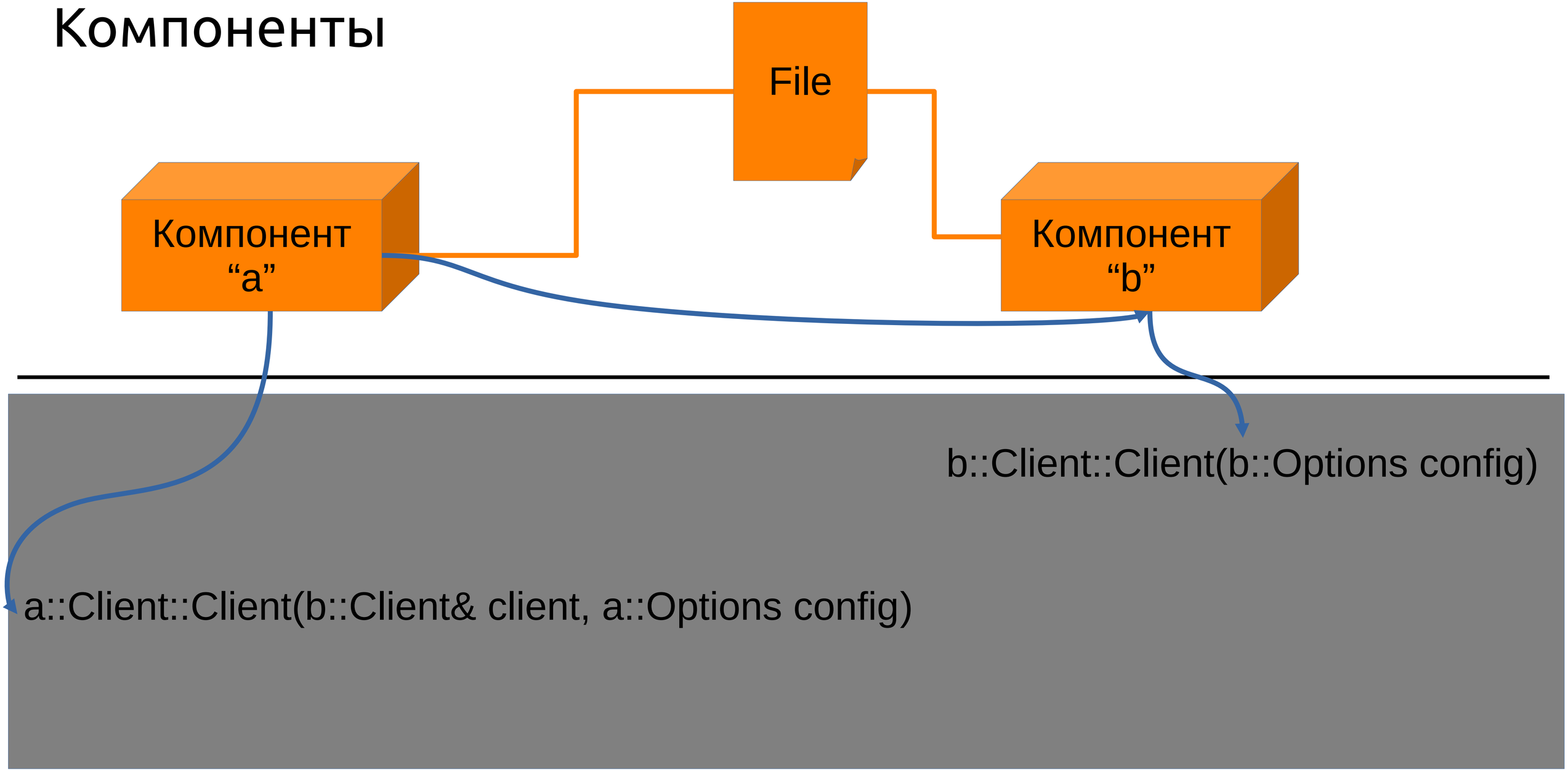

File
Компонент
“a”
Компонент
“b”
b::Client::Client(b::Options config)
a::Client::Client(b::Client& client, a::Options config)


File
Компонент
“a”
Компонент
“b”
b::Client::Client(b::Options config)
a::Client::Client(**b::Client& client**, a::Options config)


File
Компонент
“a”
Компонент
“b”
b::Client::Client(b::Options config)
a::Client::Client(b::Client& client, a::Options config)


File
Компонент
“a”
Компонент
“b”
b::Client::Client(b::Options config)
a::Client::Client(b::Client& client, a::Options config)

Компоненты


File
Компонент
“a”
Компонент
“b”
b::Client::Client(b::Options config)
a::Client::Client(b::Client& client, a::Options config)

Hello world - конфиги

```yaml
components_manager:
    components:

        server:
            listener:
                port: 8080
                task_processor: main-task-processor

        handler-hello-sample:
            path: /hello             # Registering handler by URL '/hello'.
            method: GET,POST         # It will only reply to GET (HEAD) and POST requests.
            task_processor: main-task-processor  # Run it on CPU bound task processor
```

Hello world - конфиги

```yaml
components_manager:
    components:

        server:
            listener:
                port: 8080
                task_processor: main-task-processor

        handler-hello-sample:
            path: /hello             # Registering handler by URL '/hello'.
            method: GET,POST         # It will only reply to GET (HEAD) and POST requests.
            task_processor: main-task-processor  # Run it on CPU bound task processor
```

Hello world - конфиги

```yaml
components_manager:
    components:

        server:
            listener:
                port: 8080
                task_processor: main-task-processor

        handler-hello-sample:
            path: /hello             # Registering handler by URL '/hello'.
            method: GET,POST         # It will only reply to GET (HEAD) and POST requests.
            task_processor: main-task-processor  # Run it on CPU bound task processor
```

Hello world

```
#include <userver/components/minimal_server_component_list.hpp>
#include <userver/server/handlers/http_handler_base.hpp>
#include <userver/utils/daemon_run.hpp>

struct Hello final : public server::handlers::HttpHandlerBase {
  static constexpr std::string_view kName = "handler-hello-sample";

  using HttpHandlerBase::HttpHandlerBase;

  std::string HandleRequestThrow(
      const server::http::HttpRequest&,
      server::request::RequestContext&) const override {
    return "Hello world!\n";
  }
};

int main(int argc, char* argv[]) {
  const auto component_list =
      components::MinimalServerComponentList().Append<Hello>();
  return utils::DaemonMain(argc, argv, component_list);
}
```

Hello world - конфиги

```
components_manager:
    components:

        server:
            listener:
                port: 8080
                task_processor: main-task-processor

        handler-hello-sample:
            path: /hello            # Registering handler by URL '/hello'.
            method: GET,POST        # It will only reply to GET (HEAD) and POST requests.
            task_processor: main-task-processor  # Run it on CPU bound task processor
```

Hello world - конфиги

```yaml
components_manager:
    components:

        server:
            listener:
                port: 8080
                task_processor: main-task-processor

        handler-hello-sample:
            path: /hello             # Registering handler by URL '/hello'.
            method: GET,POST         # It will only reply to GET (HEAD) and POST requests.
            task_processor: main-task-processor  # Run it on CPU bound task processor
```

Компоненты

```
Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {

  [[maybe_unused]] auto url = config["some-url"].As<std::string>();
  const auto fs_tp_name = config["fs-task-processor"].As<std::string>();

  auto& fs_task_processor = context.GetTaskProcessor(fs_tp_name);
  // ...
}
```

Компоненты

```
Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {

  [[maybe_unused]] auto url = config["some-url"].As<std::string>();
  const auto fs_tp_name = config["fs-task-processor"].As<std::string>();

  auto& fs_task_processor = context.GetTaskProcessor(fs_tp_name);
  // ...
}
```

Компоненты

```
Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {

  [[maybe_unused]] auto url = config["some-url"].As<std::string>();
  const auto fs_tp_name = config["fs-task-processor"].As<std::string>();

  auto& fs_task_processor = context.GetTaskProcessor(fs_tp_name);
  // ...
}
```

Компоненты

```
Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {

  [[maybe_unused]] auto url = config["some-url"].As<std::string>();
  const auto fs_tp_name = config["fs-task-processor"].As<std::string>();

  auto& fs_task_processor = context.GetTaskProcessor(fs_tp_name);
  // ...
}
```

Компоненты

```
Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {

  [[maybe_unused]] auto url = config["some-url"].As<std::string>();
  const auto fs_tp_name = config["fs-task-processor"].As<std::string>();

  auto& fs_task_processor = context.GetTaskProcessor(fs_tp_name);
  // ...
}
```

Hello world - конфиги

```yaml
components_manager:
    components:

        the-component:
            some-url: /bla-bla
            fs-task-processor: fs-task-processor
```

Компоненты

```
Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {

  [[maybe_unused]] auto url = config["some-url"].As<std::string>();
  const auto fs_tp_name = config["fs-task-processor"].As<std::string>();

  auto& fs_task_processor = context.GetTaskProcessor(fs_tp_name);
  // ...
}
```

Продолжаем с конфигами

Hello world - конфиги

```yaml
components_manager:
    components:

        # ...

        logging:
            fs-task-processor: fs-task-processor
            loggers:
                default:
                    file_path: '@stderr'
                    level: debug
                    overflow_behavior: discard  # Drop logs if the system is too busy

        dynamic-config:
            fs-cache-path: ''
        dynamic-config-fallbacks:
            # Load options from file and push them into the dynamic config storage.
            fallback-path: /etc/hello_service/dynamic_config_fallback.json
```

Hello world - конфиги

```
components_manager:
    components:

        # ...

        logging:
            fs-task-processor: fs-task-processor
            loggers:
                default:
                    file_path: '@stderr'
                    level: debug
                    overflow_behavior: discard  # Drop logs if the system is too busy

        dynamic-config:
            fs-cache-path: ''
        dynamic-config-fallbacks:
            # Load options from file and push them into the dynamic config storage.
            fallback-path: /etc/hello_service/dynamic_config_fallback.json
```

Hello world - конфиги

```yaml
components_manager:
    components:

        # ...

        logging:
            fs-task-processor: fs-task-processor
            loggers:
                default:
                    file_path: '@stderr'
                    level: debug
                    overflow_behavior: discard  # Drop logs if the system is too busy

        dynamic-config:
            fs-cache-path: ''
        dynamic-config-fallbacks:
            # Load options from file and push them into the dynamic config storage.
            fallback-path: /etc/hello_service/dynamic_config_fallback.json
```

Динамические конфиги

Вы делаете новый функционал

Вы делаете новый функционал

Допустим, вы сделали «возможность проезда по платным дорогам»

Вы делаете новый функционал

Допустим, вы сделали «возможность проезда по платным дорогам»

- Написали нужный код

Вы делаете новый функционал

Допустим, вы сделали «возможность проезда по платным дорогам»

- Написали нужный код
- Скомпилировали, прогнали тесты

Вы делаете новый функционал

Допустим, вы сделали «возможность проезда по платным дорогам»

- Написали нужный код
- Скомпилировали, прогнали тесты
- «Выкатили» на сервера

Вы делаете новый функционал

Допустим, вы сделали «возможность проезда по платным дорогам»

- Написали нужный код
- Скомпилировали, прогнали тесты
- «Выкатили» на сервера
-

Вы делаете новый функционал

Допустим, вы сделали «возможность проезда по платным дорогам»

- Написали нужный код
- Скомпилировали, прогнали тесты
- «Выкатили» на сервера
- обнаружилась фатальная проблема

Что делать?

Что делать?

- Очень плохой вариант: поправить код, перевыкатить

Что делать?

- Очень плохой вариант: поправить код, перевыкатить
- Терпимый вариант: отключить через правку статического конфига

Что делать?

- Очень плохой вариант: поправить код, перевыкатить
- Терпимый вариант: отключить через правку статического конфига
  - проблемы с обновлениями сервиса

Что делать?

- Очень плохой вариант: поправить код, перевыкатить
- Терпимый вариант: отключить через правку статического конфига
  - проблемы с обновлениями сервиса
  - проблемы с пропущенными серверами

Что делать?

- Очень плохой вариант: поправить код, перевыкатить
- Терпимый вариант: отключить через правку статического конфига
  - проблемы с обновлениями сервиса
  - проблемы с пропущенными серверами
  - надо и в конфиге исходнике поправить

Что делать?

- Очень плохой вариант: поправить код, перевыкатить
- Терпимый вариант: отключить через правку статического конфига
  - проблемы с обновлениями сервиса
  - проблемы с пропущенными серверами
  - надо и в конфиге исходнике поправить
- Хороший вариант: динамический конфиг

Динамический конфиг

Динамический конфиг

Сервис раздающий конфиги

Динамический конфиг

Сервис раздающий конфиги

- Меняем конфиг из браузера

Динамический конфиг

Сервис раздающий конфиги

- Меняем конфиг из браузера
- Конфиг автоматически применяется ко всем серверам

Динамический конфиг

Сервис раздающий конфиги

- Меняем конфиг из браузера
- Конфиг автоматически применяется ко всем серверам

Позволяет:

- Безопасно внедрять новый функционал

Динамический конфиг

Сервис раздающий конфиги

- Меняем конфиг из браузера
- Конфиг автоматически применяется ко всем серверам

Позволяет:

- Безопасно внедрять новый функционал
- Проводить эксперименты

Динамический конфиг

Сервис раздающий конфиги

- Меняем конфиг из браузера
- Конфиг автоматически применяется ко всем серверам

Позволяет:

- Безопасно внедрять новый функционал
- Проводить эксперименты
- Задавать лимиты/таймауты/логирование/…

Динамический конфиг

Сервис раздающий конфиги

- Меняем конфиг из браузера
- Конфиг автоматически применяется ко всем серверам

Позволяет:

- Безопасно внедрять новый функционал
- Проводить эксперименты
- Задавать лимиты/таймауты/логирование/…
- Менять proxy/авторизации и т.п.

Динамические конфиги

```
namespace {

int ParseRuntimeCfg(const dynamic_config::DocsMap& docs_map) {
  return docs_map.Get("SAMPLE_INTEGER_FROM_RUNTIME_CONFIG").As<int>();
}

constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace
```

Динамические конфиги

```
namespace {

int ParseRuntimeCfg(const dynamic_config::DocsMap& docs_map) {
  return docs_map.Get("SAMPLE_INTEGER_FROM_RUNTIME_CONFIG").As<int>();
}

constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace
```

Динамические конфиги

```
namespace {

int ParseRuntimeCfg(const dynamic_config::DocsMap& docs_map) {
  return docs_map.Get("SAMPLE_INTEGER_FROM_RUNTIME_CONFIG").As<int>();
}

constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace
```

Динамические конфиги

```
namespace {

int ParseRuntimeCfg(const dynamic_config::DocsMap& docs_map) {
  return docs_map.Get("SAMPLE_INTEGER_FROM_RUNTIME_CONFIG").As<int>();
}

constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace
```

Динамические конфиги

```
namespace {

int ParseRuntimeCfg(const dynamic_config::DocsMap& docs_map) {
  return docs_map.Get("SAMPLE_INTEGER_FROM_RUNTIME_CONFIG").As<int>();
}

constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace
```

Динамические конфиги

```
namespace {

int ParseRuntimeCfg(const dynamic_config::DocsMap& docs_map) {
  return docs_map.Get("SAMPLE_INTEGER_FROM_RUNTIME_CONFIG").As<int>();
}

constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace
```

Динамические конфиги

```
namespace {

int ParseRuntimeCfg(const dynamic_config::DocsMap& docs_map) {
  return docs_map.Get("SAMPLE_INTEGER_FROM_RUNTIME_CONFIG").As<int>();
}

constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace
```

Динамические конфиги

```
constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace

Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {
  // ...
}

int Component::DoSomething() const {
  // Getting a snapshot of dynamic config.
  const auto runtime_config = config_.GetSnapshot();
  return runtime_config[kMyConfig];
}
```

Динамические конфиги

```
constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace

Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {
  // ...
}

int Component::DoSomething() const {
  // Getting a snapshot of dynamic config.
  const auto runtime_config = config_.GetSnapshot();
  return runtime_config[kMyConfig];
}
```

Динамические конфиги

```
constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace

Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {
  // ...
}

int Component::DoSomething() const {
  // Getting a snapshot of dynamic config.
  const auto runtime_config = config_.GetSnapshot();
  return runtime_config[kMyConfig];
}
```

Динамические конфиги

```
constexpr dynamic_config::Key<ParseRuntimeCfg> kMyConfig{};

}  // namespace

Component::Component(const components::ComponentConfig& config,
                     const components::ComponentContext& context)
    : components::LoggableComponentBase(config, context),
      config_(
          context.FindComponent<components::DynamicConfig>()
              .GetSource()  // getting "client" from a component
      ) {
  // ...
}

int Component::DoSomething() const {
  // Getting a snapshot of dynamic config.
  const auto runtime_config = config_.GetSnapshot();
  return runtime_config[kMyConfig];
}
```

gRPC

Проблемы HTTP

Проблемы HTTP

В HTTP есть недостатки для «удалённого вызова процедур»:

Проблемы HTTP

В HTTP есть недостатки для «удалённого вызова процедур»:

- Нет встроенных парсеров

Проблемы HTTP

В HTTP есть недостатки для «удалённого вызова процедур»:

- Нет встроенных парсеров
- Нет встроенных сериализаторов

Проблемы HTTP

В HTTP есть недостатки для «удалённого вызова процедур»:

- Нет встроенных парсеров
- Нет встроенных сериализаторов
- Сторонние решения разрозненны, и не всегда хорошо поддерживаются

gRPC

gRPC

gRPC — добавляем к HTTP парсеры и сериализаторы, скрываем что это HTTP.

gRPC

gRPC — добавляем к HTTP парсеры и сериализаторы, скрываем что это HTTP.

- Вводится специальный язык proto3, описывающий RPC

gRPC

gRPC — добавляем к HTTP парсеры и сериализаторы, скрываем что это HTTP.

- Вводится специальный язык proto3, описывающий RPC
- Компилятор, который генерирует RPC код для всех современных языков программирования

gRPC

gRPC — добавляем к HTTP парсеры и сериализаторы, скрываем что это HTTP.

- Вводится специальный язык proto3, описывающий RPC
- Компилятор, который генерирует RPC код для всех современных языков программирования
- Реализация, расширяемая под различные фреймворки

gRPC – proto3

gRPC

```
syntax = "proto3";

package samples.api;

service GreeterService {
  rpc SayHello(GreetingRequest) returns(GreetingResponse) {}
}

message GreetingRequest {
  string name = 1;
}

message GreetingResponse {
  string greeting = 1;
}
```

gRPC

```
syntax = "proto3";

package samples.api;

service GreeterService {
  rpc SayHello(GreetingRequest) returns(GreetingResponse) {}
}

message GreetingRequest {
  string name = 1;
}

message GreetingResponse {
  string greeting = 1;
}
```

gRPC

```
syntax = "proto3";

package samples.api;

service GreeterService {
  rpc SayHello(GreetingRequest) returns(GreetingResponse) {}
}

message GreetingRequest {
  string name = 1;
}

message GreetingResponse {
  string greeting = 1;
}
```

gRPC

```
syntax = "proto3";

package samples.api;

service GreeterService {
  rpc SayHello(GreetingRequest) returns(GreetingResponse) {}
}

message GreetingRequest {
  string name = 1;
}

message GreetingResponse {
  string greeting = 1;
}
```

gRPC

```
syntax = "proto3";

package samples.api;

service GreeterService {
  rpc SayHello(GreetingRequest) returns(GreetingResponse) {}
}

message GreetingRequest {
  string name = 1;
}

message GreetingResponse {
  string greeting = 1;
}
```

gRPC

```
syntax = "proto3";

package samples.api;

service GreeterService {
  rpc SayHello(GreetingRequest) returns(GreetingResponse) {}
}

message GreetingRequest {
  string name = 1;
}

message GreetingResponse {
  string greeting = 1;
}
```

gRPC

```
syntax = "proto3";

package samples.api;

service GreeterService {
  rpc SayHello(GreetingRequest) returns(GreetingResponse) {}
}

message GreetingRequest {
  string name = 1;
}

message GreetingResponse {
  string greeting = 1;
}
```

gRPC

```
class GreeterServiceComponent final
    : public api::GreeterServiceBase::Component {
 public:

  static constexpr std::string_view kName = "greeter-service";

  GreeterServiceComponent(const components::ComponentConfig& config,
                          const components::ComponentContext& context)
      : api::GreeterServiceBase::Component(config, context),
        prefix_(config["greeting-prefix"].As<std::string>()) {}

  void SayHello(SayHelloCall& call, api::GreetingRequest&& request) override;

 private:
  const std::string prefix_;
};
```

gRPC

```
class GreeterServiceComponent final
    : public api::GreeterServiceBase::Component {
 public:

  static constexpr std::string_view kName = "greeter-service";

  GreeterServiceComponent(const components::ComponentConfig& config,
                          const components::ComponentContext& context)
      : api::GreeterServiceBase::Component(config, context),
        prefix_(config["greeting-prefix"].As<std::string>()) {}

  void SayHello(SayHelloCall& call, api::GreetingRequest&& request) override;

 private:
  const std::string prefix_;
};
```

gRPC

```
class GreeterServiceComponent final
    : public api::GreeterServiceBase::Component {
 public:

  static constexpr std::string_view kName = "greeter-service";

  GreeterServiceComponent(const components::ComponentConfig& config,
                          const components::ComponentContext& context)
      : api::GreeterServiceBase::Component(config, context),
        prefix_(config["greeting-prefix"].As<std::string>()) {}

  void SayHello(SayHelloCall& call, api::GreetingRequest&& request) override;

 private:
  const std::string prefix_;
};
```

gRPC

```
class GreeterServiceComponent final
    : public api::GreeterServiceBase::Component {
 public:

  static constexpr std::string_view kName = "greeter-service";

  GreeterServiceComponent(const components::ComponentConfig& config,
                          const components::ComponentContext& context)
      : api::GreeterServiceBase::Component(config, context),
        prefix_(config["greeting-prefix"].As<std::string>()) {}

  void SayHello(SayHelloCall& call, api::GreetingRequest&& request) override;

 private:
  const std::string prefix_;
};
```

gRPC

```
class GreeterServiceComponent final
    : public api::GreeterServiceBase::Component {
 public:

  static constexpr std::string_view kName = "greeter-service";

  GreeterServiceComponent(const components::ComponentConfig& config,
                          const components::ComponentContext& context)
      : api::GreeterServiceBase::Component(config, context),
        prefix_(config["greeting-prefix"].As<std::string>()) {}

  void SayHello(SayHelloCall& call, api::GreetingRequest&& request) override;

 private:
  const std::string prefix_;
};
```

gRPC

```
class GreeterServiceComponent final
    : public api::GreeterServiceBase::Component {
 public:

  static constexpr std::string_view kName = "greeter-service";

  GreeterServiceComponent(const components::ComponentConfig& config,
                          const components::ComponentContext& context)
      : api::GreeterServiceBase::Component(config, context),
        prefix_(config["greeting-prefix"].As<std::string>()) {}

  void SayHello(SayHelloCall& call, api::GreetingRequest&& request) override;

 private:
  const std::string prefix_;
};
```

gRPC

```
class GreeterServiceComponent final
    : public api::GreeterServiceBase::Component {
 public:

  static constexpr std::string_view kName = "greeter-service";

  GreeterServiceComponent(const components::ComponentConfig& config,
                          const components::ComponentContext& context)
      : api::GreeterServiceBase::Component(config, context),
        prefix_(config["greeting-prefix"].As<std::string>()) {}

  void SayHello(SayHelloCall& call, api::GreetingRequest&& request) override;

 private:
  const std::string prefix_;
};
```

gRPC

```
void GreeterServiceComponent::SayHello(
    api::GreeterServiceBase::SayHelloCall& call,
    api::GreetingRequest&& request) {

  api::GreetingResponse response;

  response.set_greeting(fmt::format("{}, {}!", prefix_, request.name()));

  call.Finish(response);
}
```

gRPC

```
void GreeterServiceComponent::SayHello(
    api::GreeterServiceBase::SayHelloCall& call,
    api::GreetingRequest&& request) {

  api::GreetingResponse response;

  response.set_greeting(fmt::format("{}, {}!", prefix_, request.name()));

  call.Finish(response);
}
```

gRPC

```
void GreeterServiceComponent::SayHello(
    api::GreeterServiceBase::SayHelloCall& call,
    api::GreetingRequest&& request) {

  api::GreetingResponse response;

  response.set_greeting(fmt::format("{}, {}!", prefix_, request.name()));

  call.Finish(response);
}
```

gRPC

```
void GreeterServiceComponent::SayHello(
    api::GreeterServiceBase::SayHelloCall& call,
    api::GreetingRequest&& request) {

  api::GreetingResponse response;

  response.set_greeting(fmt::format("{}, {}!", prefix_, request.name()));

  call.Finish(response);
}
```

gRPC

```yaml
components_manager:
    components:
        # ...

        grpc-server:
            port: 8091

        greeter-service:
            task-processor: main-task-processor
            greeting-prefix: Hello
```

gRPC

```yaml
components_manager:
    components:
        # ...

        grpc-server:
            port: 8091

        greeter-service:
            task-processor: main-task-processor
            greeting-prefix: Hello
```

gRPC

```yaml
components_manager:
    components:
        # ...

        grpc-server:
            port: 8091

        greeter-service:
            task-processor: main-task-processor
            greeting-prefix: Hello
```

gRPC

```
class GreeterServiceComponent final
    : public api::GreeterServiceBase::Component {
 public:

  static constexpr std::string_view kName = "greeter-service";

  GreeterServiceComponent(const components::ComponentConfig& config,
                          const components::ComponentContext& context)
      : api::GreeterServiceBase::Component(config, context),
        prefix_(config["greeting-prefix"].As<std::string>()) {}

  void SayHello(SayHelloCall& call, api::GreetingRequest&& request) override;

 private:
  const std::string prefix_;
};
```

gRPC клиент

gRPC

```
class GreeterClient final : public components::LoggableComponentBase {
 public:

  static constexpr std::string_view kName = "greeter-client";

  GreeterClient(const components::ComponentConfig& config,
                const components::ComponentContext& context)
      : LoggableComponentBase(config, context),
        client_factory_(
            context.FindComponent<ugrpc::client::ClientFactoryComponent>()
                .GetFactory()),
        client_(client_factory_.MakeClient<api::GreeterServiceClient>(
            config["endpoint"].As<std::string>())) {}

  std::string SayHello(std::string name);

 private:
  ugrpc::client::ClientFactory& client_factory_;
  api::GreeterServiceClient client_;
};
```

gRPC

```
class GreeterClient final : public components::LoggableComponentBase {
 public:

  static constexpr std::string_view kName = "greeter-client";

  GreeterClient(const components::ComponentConfig& config,
                const components::ComponentContext& context)
      : LoggableComponentBase(config, context),
        client_factory_(
            context.FindComponent<ugrpc::client::ClientFactoryComponent>()
                .GetFactory()),
        client_(client_factory_.MakeClient<api::GreeterServiceClient>(
            config["endpoint"].As<std::string>())) {}

  std::string SayHello(std::string name);

 private:
  ugrpc::client::ClientFactory& client_factory_;
  api::GreeterServiceClient client_;
};
```

gRPC

```
class GreeterClient final : public components::LoggableComponentBase {
 public:

  static constexpr std::string_view kName = "greeter-client";

  GreeterClient(const components::ComponentConfig& config,
                const components::ComponentContext& context)
      : LoggableComponentBase(config, context),
        client_factory_(
            context.FindComponent<ugrpc::client::ClientFactoryComponent>()
                .GetFactory()),
        client_(client_factory_.MakeClient<api::GreeterServiceClient>(
            config["endpoint"].As<std::string>())) {}

  std::string SayHello(std::string name);

 private:
  ugrpc::client::ClientFactory& client_factory_;
  api::GreeterServiceClient client_;
};
```

gRPC

```
class GreeterClient final : public components::LoggableComponentBase {
 public:

  static constexpr std::string_view kName = "greeter-client";

  GreeterClient(const components::ComponentConfig& config,
                const components::ComponentContext& context)
      : LoggableComponentBase(config, context),
        client_factory_(
            context.FindComponent<ugrpc::client::ClientFactoryComponent>()
                .GetFactory()),
        client_(client_factory_.MakeClient<api::GreeterServiceClient>(
            config["endpoint"].As<std::string>())) {}

  std::string SayHello(std::string name);

 private:
  ugrpc::client::ClientFactory& client_factory_;
  api::GreeterServiceClient client_;
};
```

gRPC

```
std::string GreeterClient::SayHello(std::string name) {
  api::GreetingRequest request;
  request.set_name(std::move(name));

  // Deadline must be set manually for each RPC
  auto context = std::make_unique<grpc::ClientContext>();
  context->set_deadline(
      engine::Deadline::FromDuration(std::chrono::seconds{20}));

  auto stream = client_.SayHello(request, std::move(context));

  api::GreetingResponse response = stream.Finish();
  return std::move(*response.mutable_greeting());
}
```

gRPC

```
std::string GreeterClient::SayHello(std::string name) {
  api::GreetingRequest request;
  request.set_name(std::move(name));

  // Deadline must be set manually for each RPC
  auto context = std::make_unique<grpc::ClientContext>();
  context->set_deadline(
      engine::Deadline::FromDuration(std::chrono::seconds{20}));

  auto stream = client_.SayHello(request, std::move(context));

  api::GreetingResponse response = stream.Finish();
  return std::move(*response.mutable_greeting());
}
```

gRPC

```
std::string GreeterClient::SayHello(std::string name) {
  api::GreetingRequest request;
  request.set_name(std::move(name));

  // Deadline must be set manually for each RPC
  auto context = std::make_unique<grpc::ClientContext>();
  context->set_deadline(
      engine::Deadline::FromDuration(std::chrono::seconds{20}));

  auto stream = client_.SayHello(request, std::move(context));

  api::GreetingResponse response = stream.Finish();
  return std::move(*response.mutable_greeting());
}
```

gRPC

```
std::string GreeterClient::SayHello(std::string name) {
  api::GreetingRequest request;
  request.set_name(std::move(name));

  // Deadline must be set manually for each RPC
  auto context = std::make_unique<grpc::ClientContext>();
  context->set_deadline(
      engine::Deadline::FromDuration(std::chrono::seconds{20}));

  auto stream = client_.SayHello(request, std::move(context));

  api::GreetingResponse response = stream.Finish();
  return std::move(*response.mutable_greeting());
}
```

gRPC

```
std::string GreeterClient::SayHello(std::string name) {
  api::GreetingRequest request;
  request.set_name(std::move(name));

  // Deadline must be set manually for each RPC
  auto context = std::make_unique<grpc::ClientContext>();
  context->set_deadline(
      engine::Deadline::FromDuration(std::chrono::seconds{20}));

  auto stream = client_.SayHello(request, std::move(context));

  api::GreetingResponse response = stream.Finish();
  return std::move(*response.mutable_greeting());
}
```

gRPC

```
int main(int argc, const char* const argv[]) {
  const auto component_list =
      components::MinimalServerComponentList()
          .Append<ugrpc::client::ClientFactoryComponent>()
          .Append<ugrpc::server::ServerComponent>()
          .Append<samples::GreeterClient>()
          .Append<samples::GreeterServiceComponent>();

  return utils::DaemonMain(argc, argv, component_list);
}
```

gRPC

```yaml
components_manager:
    components:
        # ...
        grpc-client-factory:
            task-processor: grpc-blocking-task-processor
            channel-args: {}

        greeter-client:
            endpoint: '[::1]:8091'
```

gRPC

```
components_manager:
    components:
        # ...
        grpc-client-factory:
            task-processor: grpc-blocking-task-processor
            channel-args: {}

        greeter-client:
            endpoint: '[::1]:8091'
```

gRPC

```yaml
components_manager:
    components:
        # ...
        grpc-client-factory:
            task-processor: grpc-blocking-task-processor
            channel-args: {}

        greeter-client:
            endpoint: '[::1]:8091'
```

Фичи userver

Фичи

- PostgreSQL

Фичи

- PostgreSQL
- Mongo

Фичи

- PostgreSQL
- Mongo
- Clickhouse

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing
- Functional Testing (Testuite)

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing
- Functional Testing (Testuite)
- Formats (JSON, YAML, BSON, …)

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing
- Functional Testing (Testuite)
- Formats (JSON, YAML, BSON, …)
- Tracing, Metrics

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing
- Functional Testing (Testuite)
- Formats (JSON, YAML, BSON, …)
- Tracing, Metrics
- Subprocesses

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing
- Functional Testing (Testuite)
- Formats (JSON, YAML, BSON, …)
- Tracing, Metrics
- Subprocesses
- Rcu, Queue, Subscriptions, MutexSet

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing
- Functional Testing (Testuite)
- Formats (JSON, YAML, BSON, …)
- Tracing, Metrics
- Subprocesses
- Rcu, Queue, Subscriptions, MutexSet
- Deadlines / Timeouts

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing
- Functional Testing (Testuite)
- Formats (JSON, YAML, BSON, …)
- Tracing, Metrics
- Subprocesses
- Rcu, Queue, Subscriptions, MutexSet
- Deadlines / Timeouts
- Caches and cache dumps

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing
- Functional Testing (Testuite)
- Formats (JSON, YAML, BSON, …)
- Tracing, Metrics
- Subprocesses
- Rcu, Queue, Subscriptions, MutexSet
- Deadlines / Timeouts
- Caches and cache dumps
- Decimal, FastPimpl, Containers

Фичи

- PostgreSQL
- Mongo
- Clickhouse
- Redis
- Socket, TLS
- Mutex/ConditionVariable/Semaphore…
- Logs
- Dynamic Configs
- HTTP server/client
- gRPC
- Periodic Tasks
- DistLocks
- Unit testing
- Functional Testing (Testuite)
- Formats (JSON, YAML, BSON, …)
- Tracing, Metrics
- Subprocesses
- Rcu, Queue, Subscriptions, MutexSet
- Deadlines / Timeouts
- Caches and cache dumps
- Decimal, FastPimpl, Containers
- Stacktraces, PFR

Итоги

userver

userver

Высокие нагрузки требуют особых подходов

userver

Высокие нагрузки требуют особых подходов

- C++

userver

Высокие нагрузки требуют особых подходов

- C++

userver

Высокие нагрузки требуют особых подходов

- C++
- Асинхронная работа

userver

Высокие нагрузки требуют особых подходов

- C++
- Асинхронная работа
- Корутины

userver

Высокие нагрузки требуют особых подходов

- C++
- Асинхронная работа
- Корутины

Простои в работе недопустимы

userver

Высокие нагрузки требуют особых подходов

- C++
- Асинхронная работа
- Корутины

Простои в работе недопустимы

- Динамические конфиги

userver

Высокие нагрузки требуют особых подходов

- C++
- Асинхронная работа
- Корутины

Простои в работе недопустимы

- Динамические конфиги
- Надо думать об отказоустойчивости

userver

Высокие нагрузки требуют особых подходов

- C++
- Асинхронная работа
- Корутины

Простои в работе недопустимы

- Динамические конфиги
- Надо думать об отказоустойчивости
- Надо тестировать!

userver

Высокие нагрузки требуют особых подходов

- C++
- Асинхронная работа
- Корутины

Простои в работе недопустимы

- Динамические конфиги
- Надо думать об отказоустойчивости
- Надо тестировать!

Важна скорость разработки

userver

Высокие нагрузки требуют особых подходов

- C++
- Асинхронная работа
- Корутины

Простои в работе недопустимы

- Динамические конфиги
- Надо думать об отказоустойчивости
- Надо тестировать!

Важна скорость разработки

- Компоненты

userver

Высокие нагрузки требуют особых подходов

- C++
- Асинхронная работа
- Корутины

Простои в работе недопустимы

- Динамические конфиги
- Надо думать об отказоустойчивости
- Надо тестировать!

Важна скорость разработки

- Компоненты, много компонентов

Спасибо

Полухин Антон

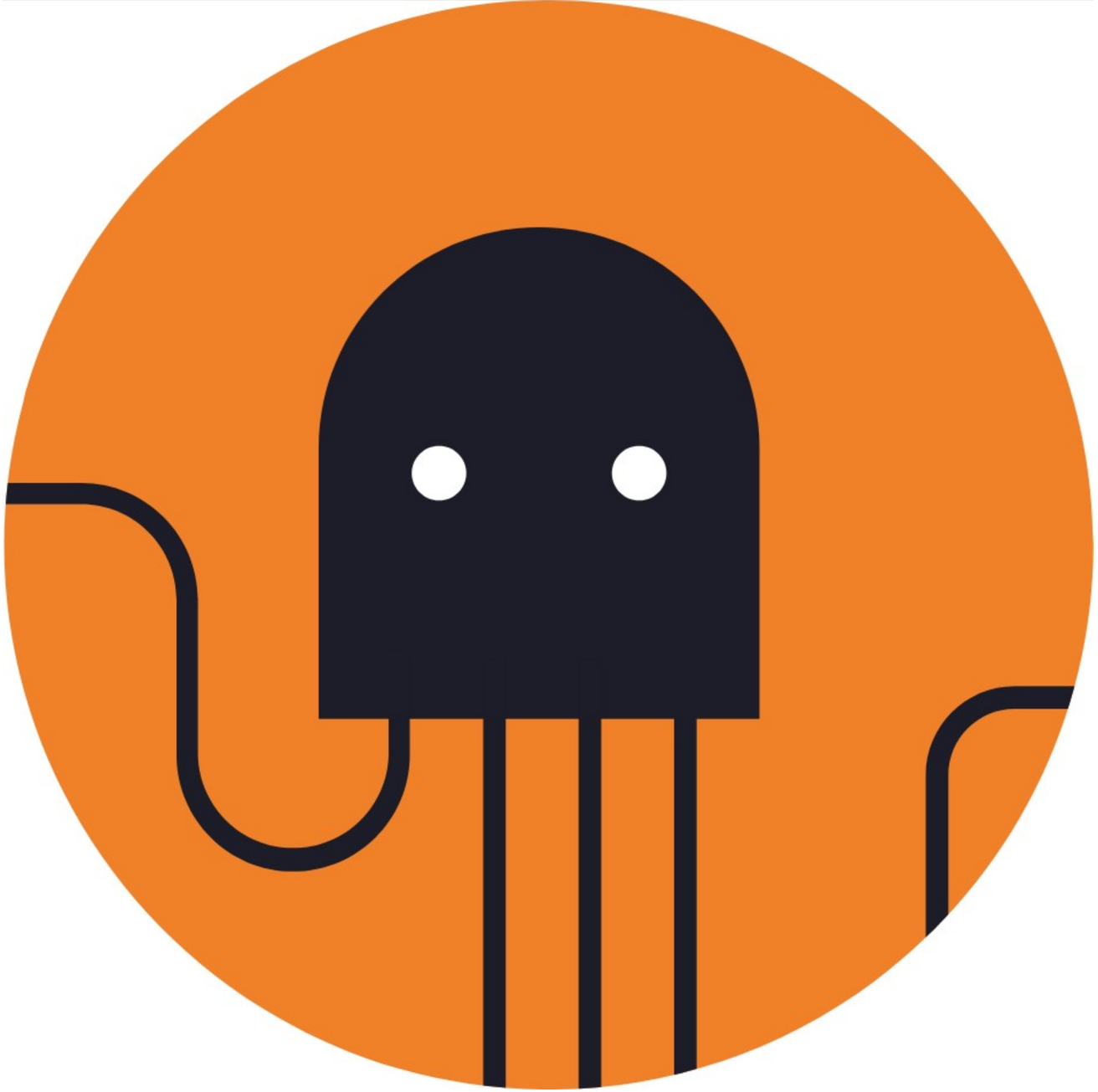
Эксперт-разработчик C++

antoshkka@gmail.com

antoshkka@yandex-team.ru

https://github.com/apolukhin

C++ РГ21 C++ РОССИЯ https://stdcpp.ru/



https://github.com/userver-framework

https://userver.tech/