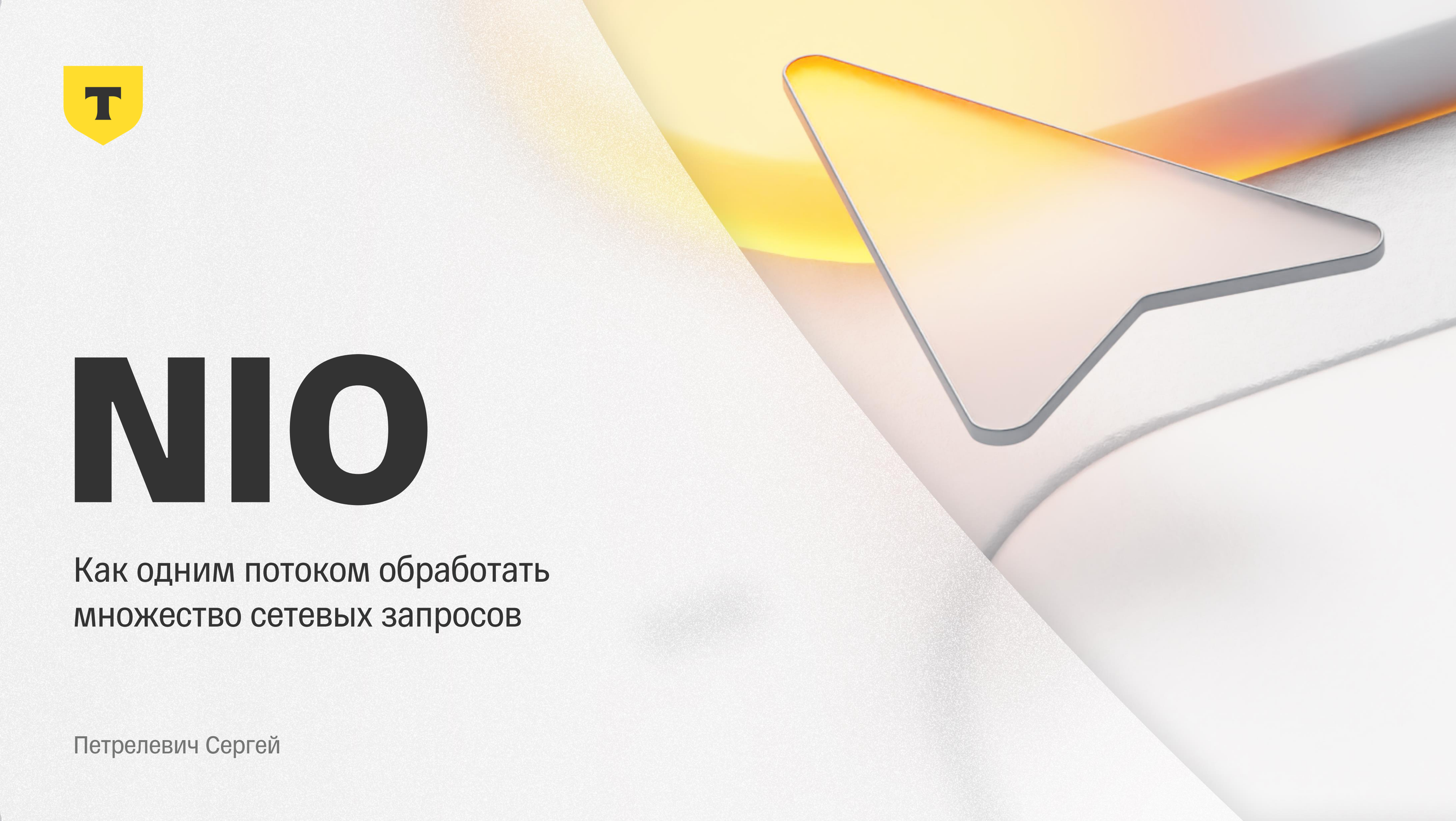


NIO

Как одним потоком обработать
множество сетевых запросов

Петрелевич Сергей

Петрелевич Сергей

Java – разработчик

Telegram @petrelevich

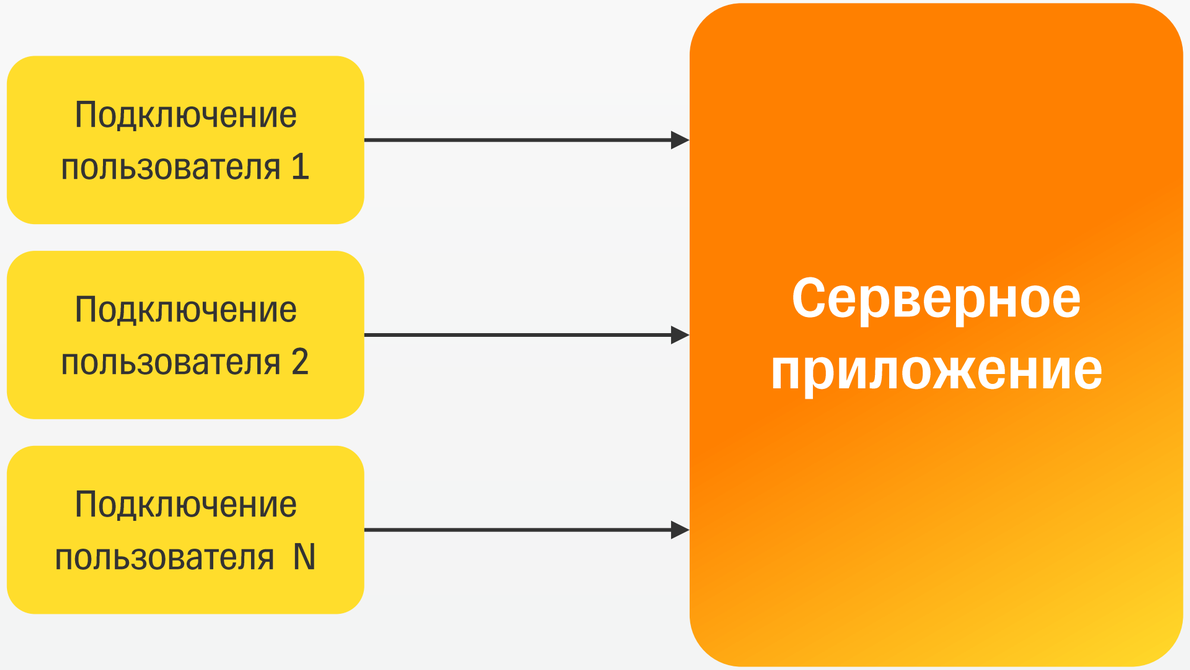
Создадим серверное приложение для обработки входящих запросов

- Приложение должно принимать
  ~~REST~~ TCP запросы и отдавать ответы
- Используем только JDK,
  без сторонних библиотек

Что получилось?

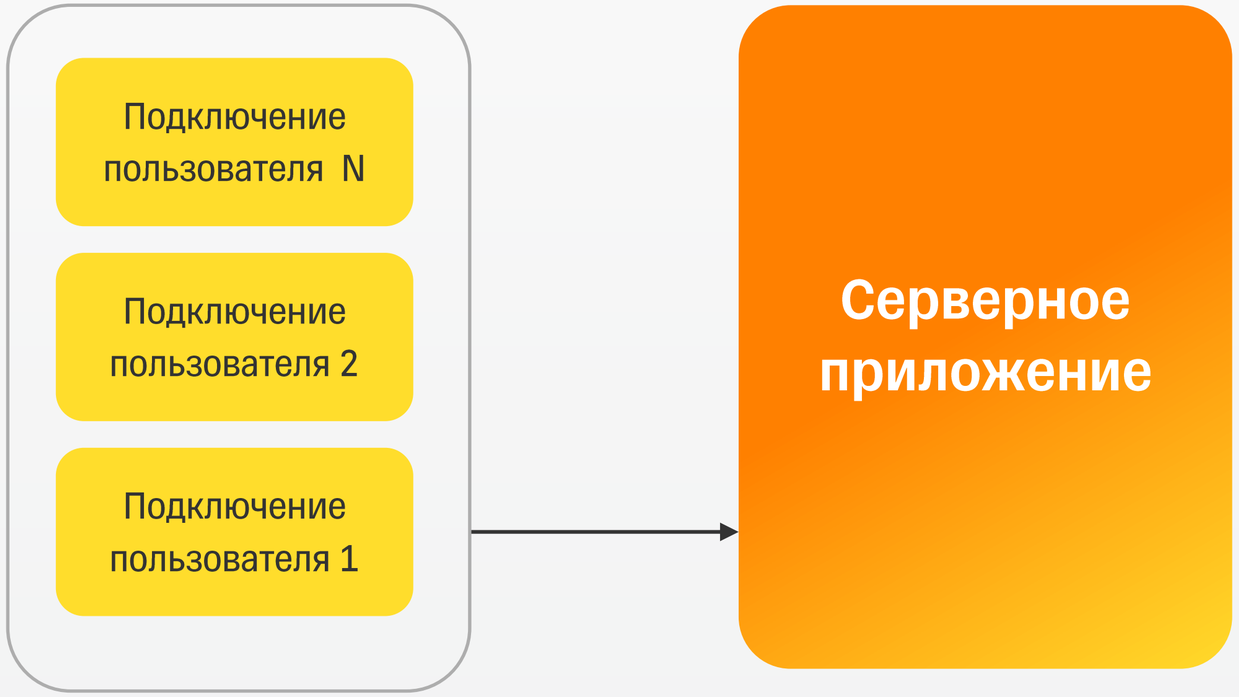
Что хотели?

Одновременная обработка нескольких подключений



Что получили?

К серверу выстроилась очередь на обработку



Что используем для диагностики?

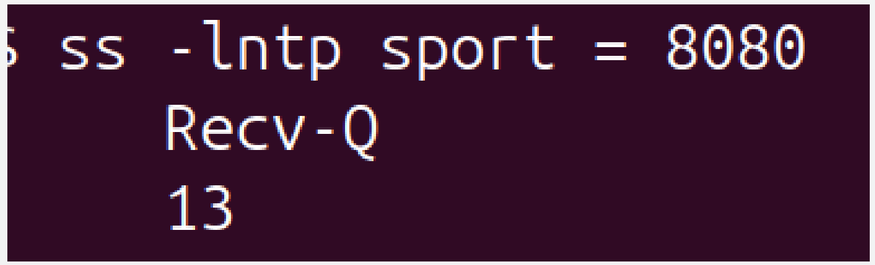
ss — another utility to investigate sockets

ss is used to dump socket statistics. It allows showing information similar to netstat.

It can display more TCP and state information than other tools.

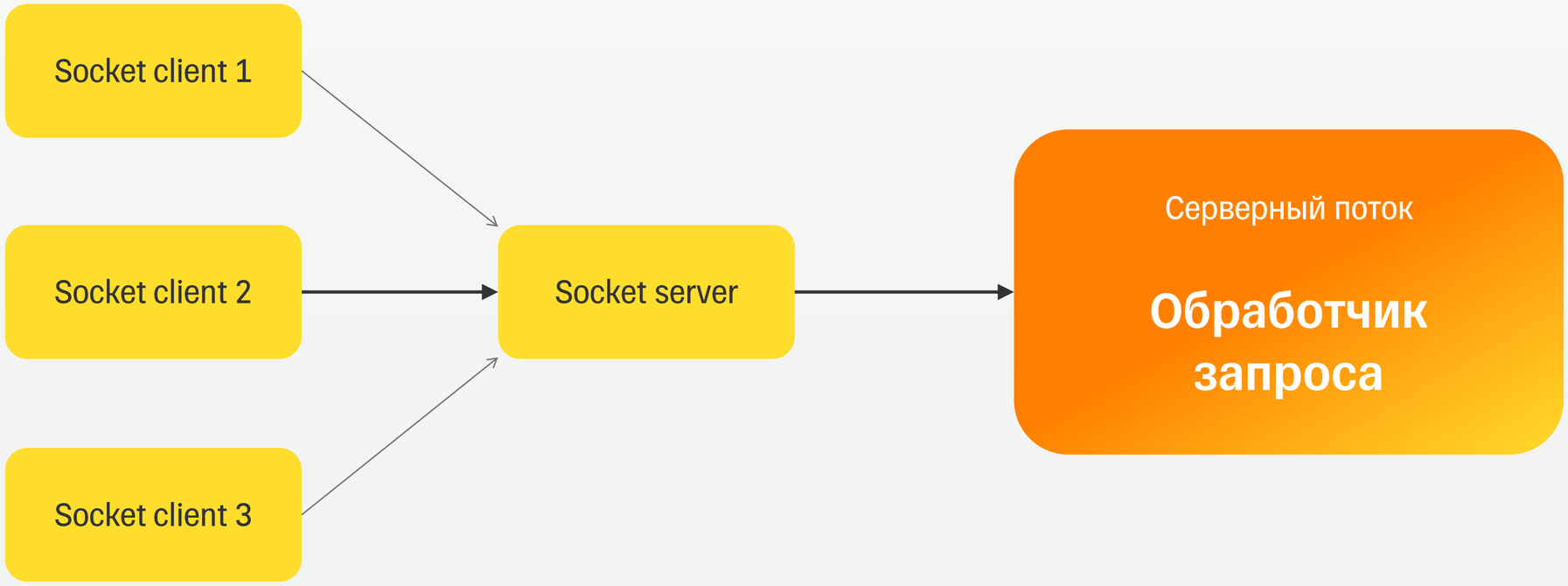
Пример запуска:

```
sergey@sergeyPc: $ ss -lntp sport = 8080
State                Recv-Q               Send-Q                             Local Address:Port                             Peer Address:Port              Process
LISTEN               13                   1000                           [::ffff:127.0.0.1]:8080                                 *:*                   users:(("java",pid=32709,fd=8))
```



Как так получилось?

К серверу выстроилась очередь на обработку



Что хотим получить?

Большой RPS

Что хотим получить?

Большой RPS

1000 RPS это:

1000 пользователей = 1000 запросов

1 сек/запрос

VS

1000 RPS это:

1 пользователь = 1000 запросов

0.001 с/запрос (1 мс/запрос)

Что хотим получить?

Большой RPS

1000 RPS это:

1000 пользователей = 1000 запросов

1 сек/запрос

VS

1000 RPS это:

1 пользователь = 1000 запросов

0.001 с/запрос (1 мс/запрос)

Что делать?

Не хватает одного пока –
возьми два…

Подобный подход, но с процессами,
мы уже видели в Apache Web-server.
«The Number One HTTP Server On The Internet»

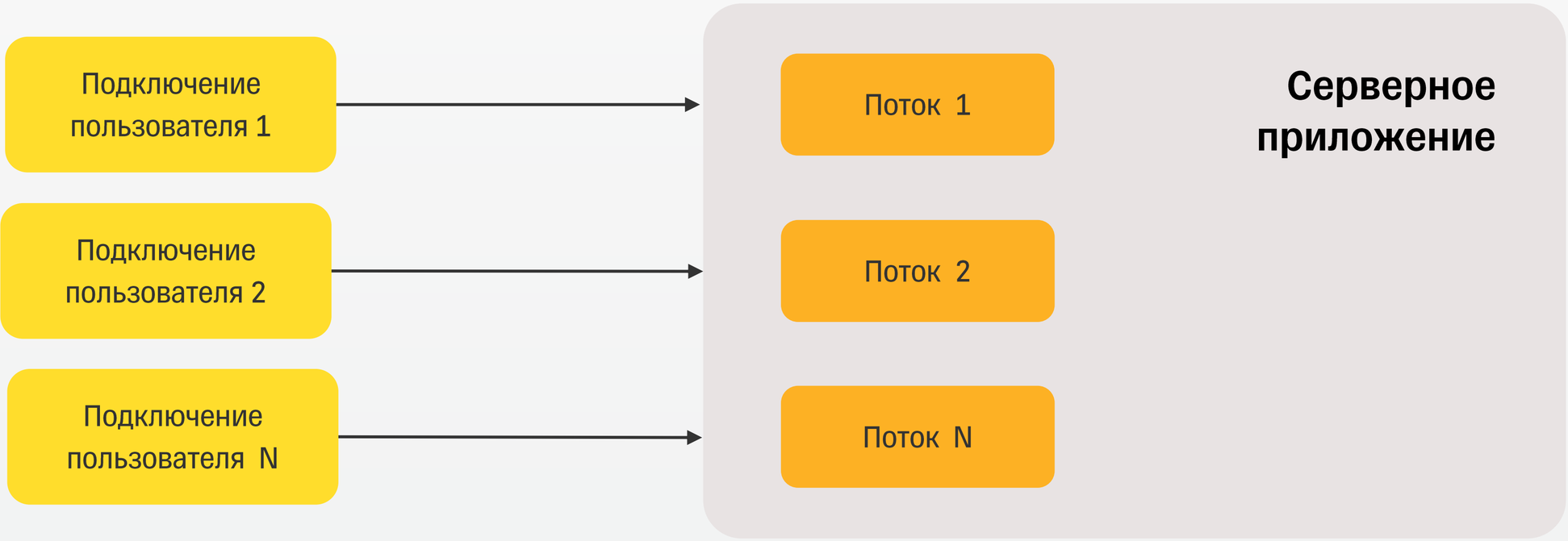
https://httpd.apache.org/docs/current/mod/prefork.html

Этот подход реализован в TomCat

https://tomcat.apache.org/tomcat-11.0-doc/config/executor.html

Что получилось?

Каждое подключение обрабатывается в своем потоке



А если параллельных подключений будет очень-очень много?

Требуется более масштабируемое решение.

Посмотрим на известный аналог.

Кто отлично умеет работать с сетью?

Nginx

nginx ("engine x") is an HTTP web server, reverse proxy, content cache, load balancer, TCP/UDP proxy server, and mail proxy server. Originally written by Igor Sysoev

Управление подключениями в Nginx

Nginx supports a variety of connection processing methods.

The availability of a particular method depends on the platform used.

https://nginx.org/en/docs/events.html

Зависимости от операционной системы

```
21      #if (NGX_FREEBSD)
22      #include <ngx_freebsd_config.h>
23
24
25      #elif (NGX_LINUX)
26      #include <ngx_linux_config.h>
27
28
29      #elif (NGX_SOLARIS)
30      #include <ngx_solaris_config.h>
31
32
33      #elif (NGX_DARWIN)
34      #include <ngx_darwin_config.h>
35
36
37      #elif (NGX_WIN32)
38      #include <ngx_win32_config.h>
39
40
41      #else /* POSIX */
42      #include <ngx_posix_config.h>
43
44      #endif
```

https://github.com/nginx/nginx/blob/master/src/core/ngx_config.h

Что там есть для Linux?

https://nginx.org/en/docs/events.html

Nginx supports a variety of connection processing methods.
The availability of a particular method depends on the platform used.

epoll — efficient method used on Linux 2.6+.

https://github.com/nginx/nginx/blob/master/src/event/modules/ngx_epoll_module.c

Что же внутри epoll модуля?

```
493         events = epoll_wait(ep, &ee, 1, 5000);
494
495         if (events == -1) {
496             ngx_log_error(NGX_LOG_ALERT, cycle->log, ngx_errno,
497                           "epoll_wait() failed");
498             goto failed;
499         }
500
501         if (events) {
502             ngx_use_epoll_rdhup = ee.events & EPOLLRDHUP;
503
504         } else {
505             ngx_log_error(NGX_LOG_ALERT, cycle->log, NGX_ETIMEDOUT,
506                           "epoll_wait() timed out");
507         }
508
509         ngx_log_debug1(NGX_LOG_DEBUG_EVENT, cycle->log, 0,
510                        "testing the EPOLLRDHUP flag: %s",
511                        ngx_use_epoll_rdhup ? "success" : "fail");
512
```

https://github.com/nginx/nginx/blob/master/src/event/modules/ngx_epoll_module.c

Можно ли как-то проще?

Можно, конечно

Полный пример:

https://github.com/petrelevich/tcp-server-client/blob/main/eventloop/server.c

```
// The event loop
while (1) {
  nfds = epoll_wait(epollfd, events, MAX_EVENTS, -1);
  for (int i = 0; i < nfds; i++) {
    if (events[i].events & EPOLLIN && events[i].data.fd == sock) {
      struct sockaddr client_addr;
      char buf[BUF_SIZE];
      int bytes_read = recvfrom(sock, buf, BUF_SIZE, 0, &client_addr, &addr_size);
      // Run the callback
      Request req = {client_addr, addr_size, sock, buf};
      server->cb_array[EventTypeMessage](&req);
    }
  }
}
```

Можно ли как-то проще?

```
// The event loop
while (1) {
  nfds = epoll_wait(epollfd, events, MAX_EVENTS, -1);
  for (int i = 0; i < nfds; i++) {
    if (events[i].events & EPOLLIN && events[i].data.fd == sock) {
      struct sockaddr client_addr;
      char buf[BUF_SIZE];
      int bytes_read = recvfrom(sock, buf, BUF_SIZE, 0, &client_addr, &addr_size);
      // Run the callback
      Request req = {client_addr, addr_size, sock, buf};
      server->cb_array[EventTypeMessage](&req);
    }
  }
}
```

Можно ли как-то проще?

```
// The event loop
while (1) {
  nfds = epoll_wait(epollfd, events, MAX_EVENTS, -1);
  for (int i = 0; i < nfds; i++) {
    if (events[i].events & EPOLLIN && events[i].data.fd == sock) {
      struct sockaddr client_addr;
      char buf[BUF_SIZE];
      int bytes_read = recvfrom(sock, buf, BUF_SIZE, 0, &client_addr, &addr_size);
      // Run the callback
      Request req = {client_addr, addr_size, sock, buf};
      server->cb_array[EventTypeMessage](&req);
    }
  }
}
```

Можно ли как-то проще?

```
// The event loop
while (1) {
  nfds = epoll_wait(epollfd, events, MAX_EVENTS, -1);
  for (int i = 0; i < nfds; i++) {
    if (events[i].events & EPOLLIN && events[i].data.fd == sock) {
      struct sockaddr client_addr;
      char buf[BUF_SIZE];
      int bytes_read = recvfrom(sock, buf, BUF_SIZE, 0, &client_addr, &addr_size);
      // Run the callback
      Request req = {client_addr, addr_size, sock, buf};
      server->cb_array[EventTypeMessage](&req);
    }
  }
}
```

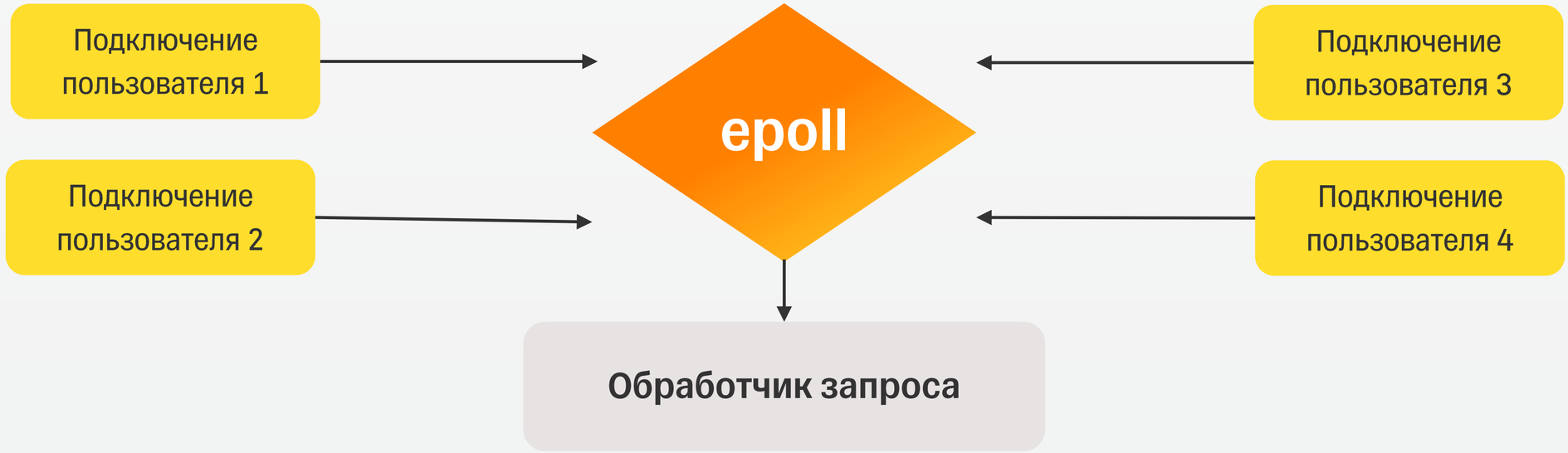
Что же такое epoll?

Это набор системных вызовов:

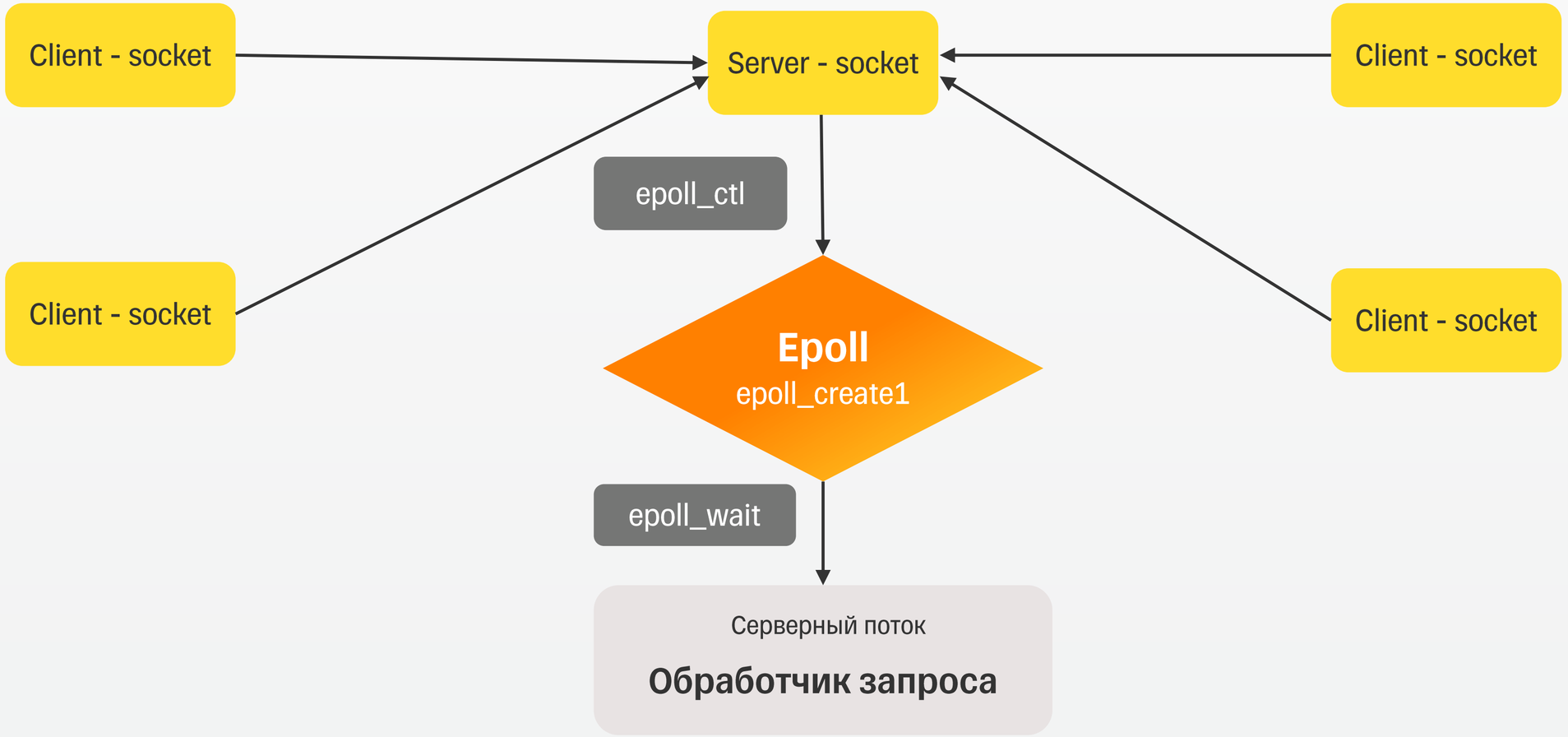
epoll_create1 – создает механизм для слежения за сетевыми подключениями.

epoll_ctl – связывает epoll с сетевым сокетом.

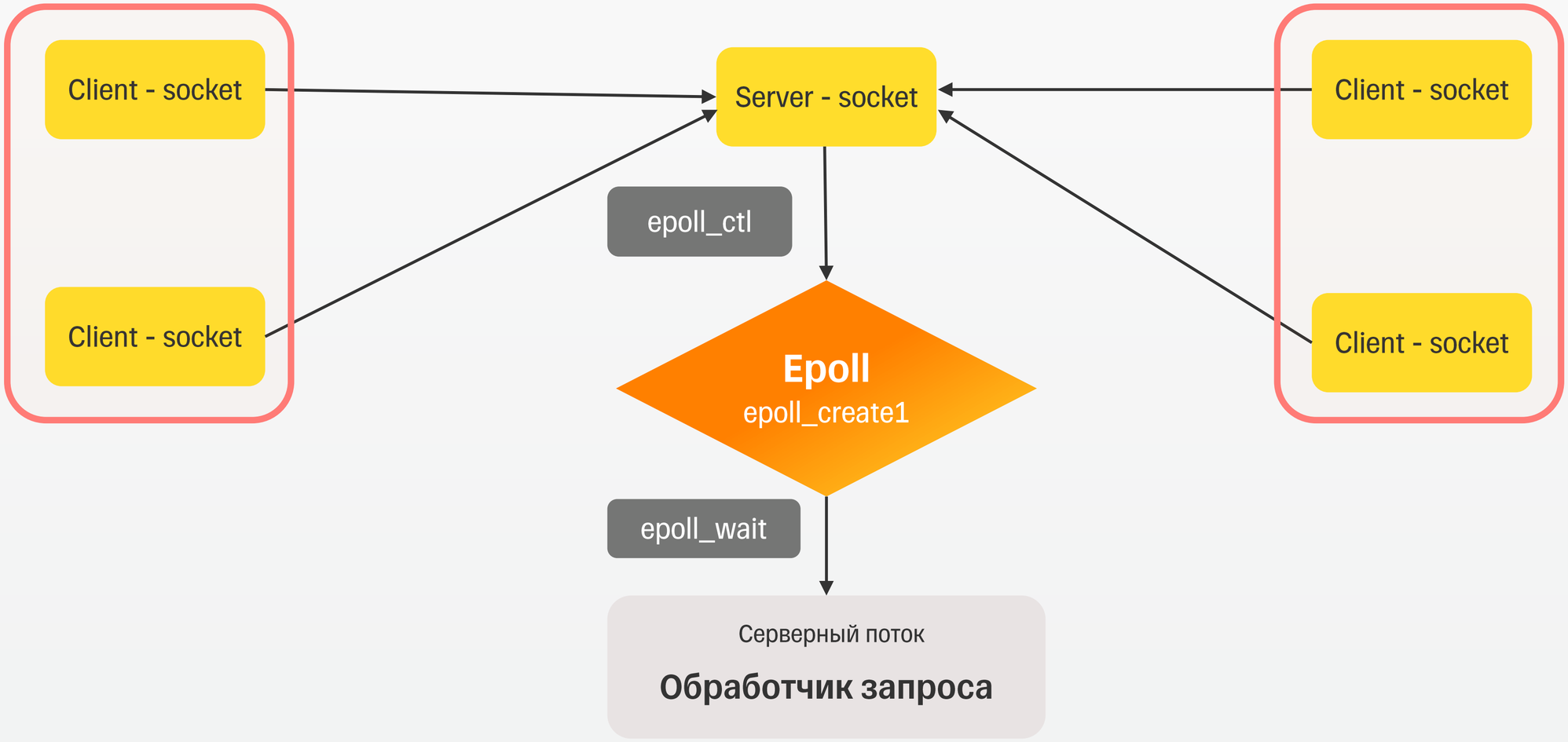
epoll_wait – ждем пока на дескрипторе epoll не появятся события.



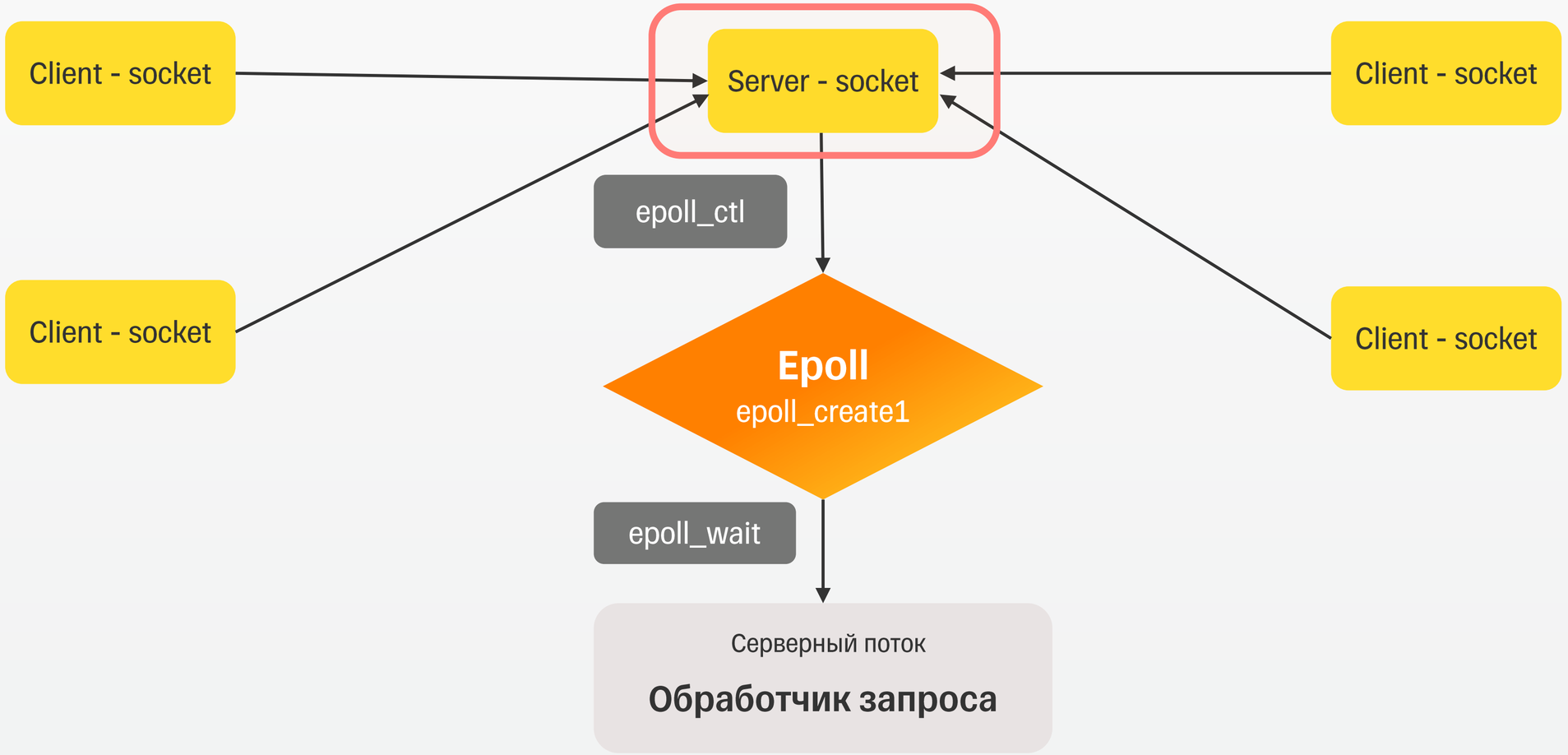
Как работает epoll?



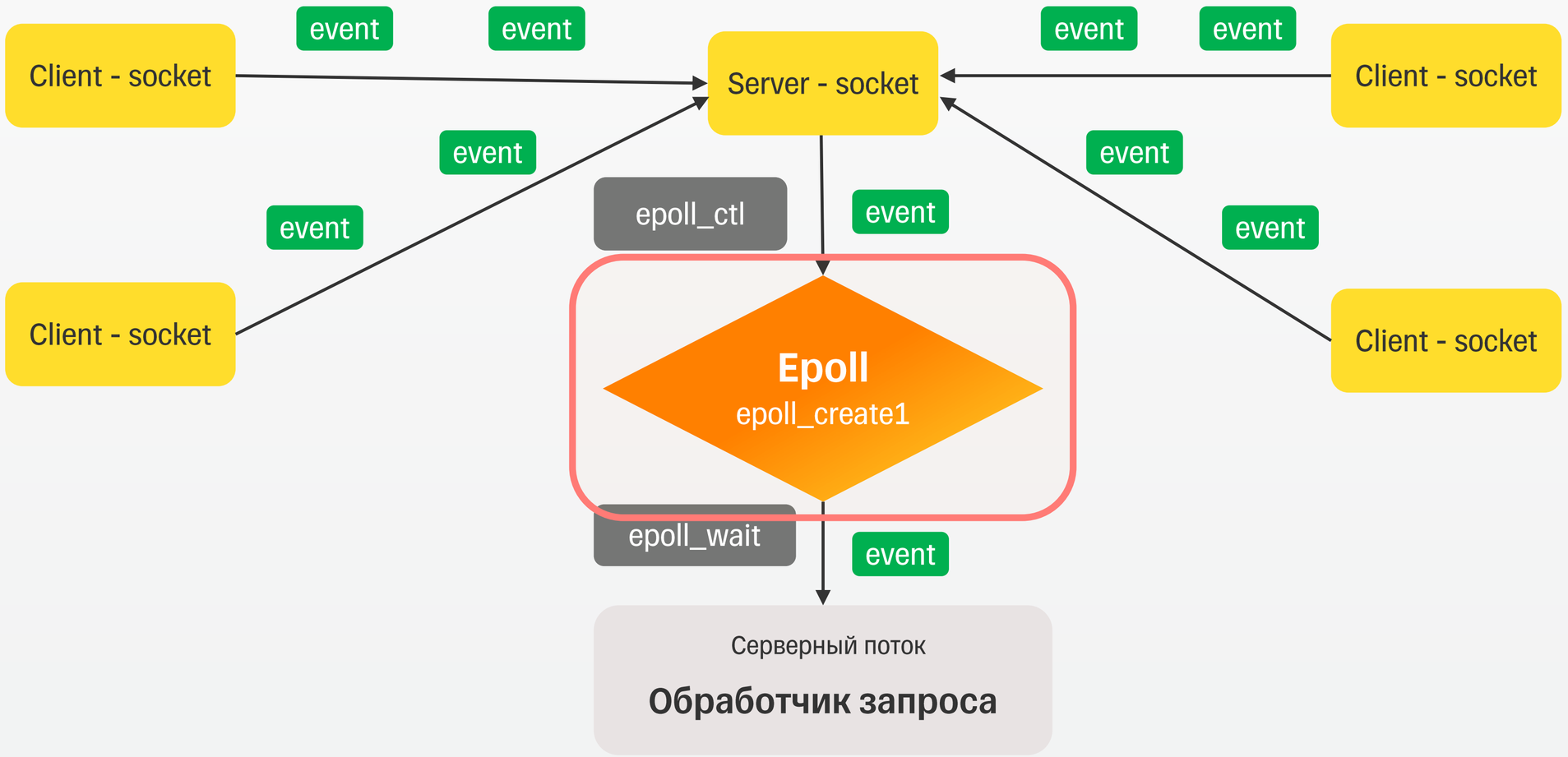
Как работает epoll?



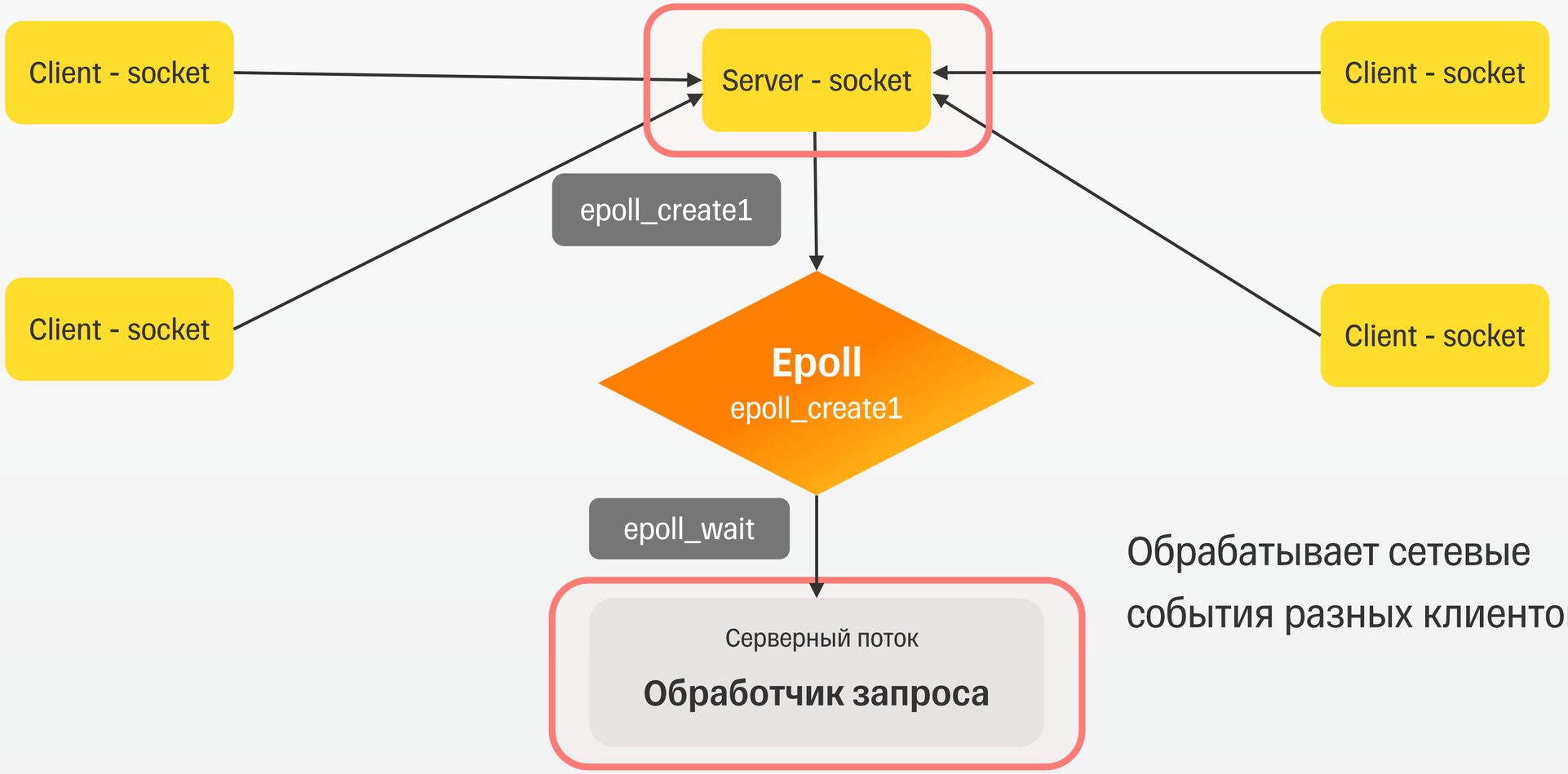
Как работает epoll?



Как работает epoll?



Как работает epoll?



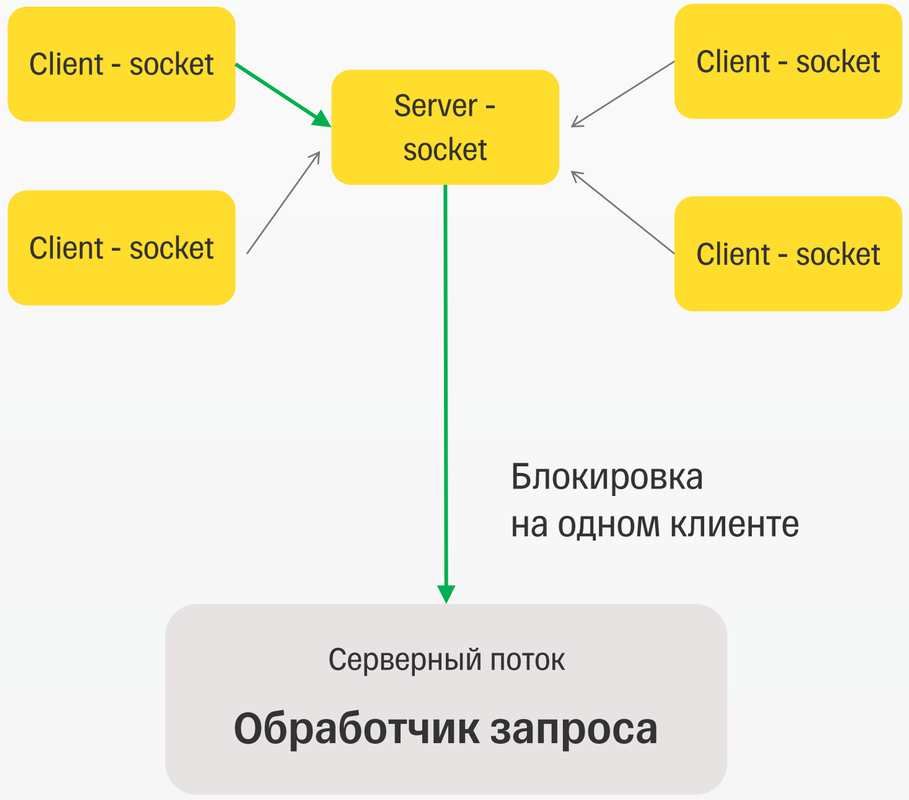
Обрабатывает сетевые события разных клиентов

Что делает epoll?

Принимает события
из сетевых интерфейсов

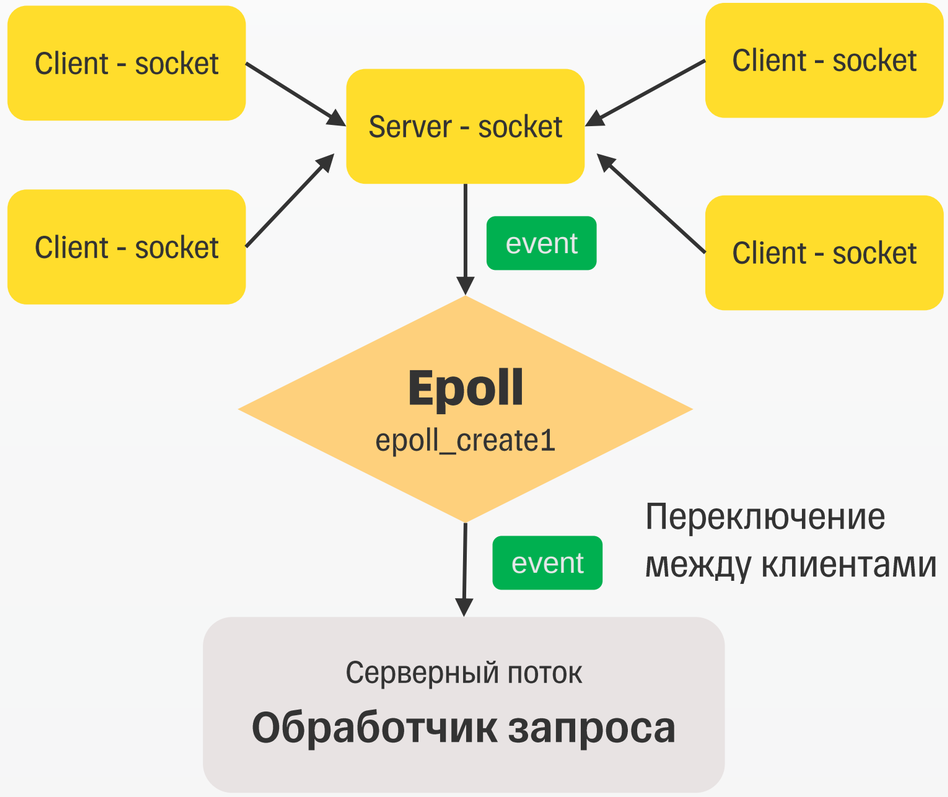
Сравним

Синхронное взаимодействие –
непрерывный поток



VS

Асинхронное взаимодействие –
набор событий



Как же все это применить в Java?

JSR 51: New I/O APIs for the JavaTM Platform

Final Release 09 May, 2002
Description:
APIs for scalable I/O, fast buffered binary and character I/O,
regular expressions, charset conversion, and an improved filesystem interface.

https://jcp.org/en/jsr/detail?id=51

Посмотрим, что получилось...

Что там внутри?

NIO – это абстракция.

В основе:package sun.nio.ch class EPoll

nio	epoll
static native int create() throws IOException;	epoll_create1
static native int ctl (int epfd, int opcode, int fd, int events);	epoll_ctl
static native int wait (int epfd, long pollAddress, int numfds, int timeout)	epoll_wait

А в Windows работать-то будет?

До java 17

https://github.com/openjdk/jdk/blob/master/src/java.base/windows/classes/sun/nio/ch/WindowsSelectorImpl.java

Использует Windows Sockets 2 https://learn.microsoft.com/en-us/windows/win32/api/_winsock/

Начиная с java 17

https://bugs.openjdk.org/browse/JDK-8266369

Add a new Selector implementation for the Windows platform based on the "wepoll" library.

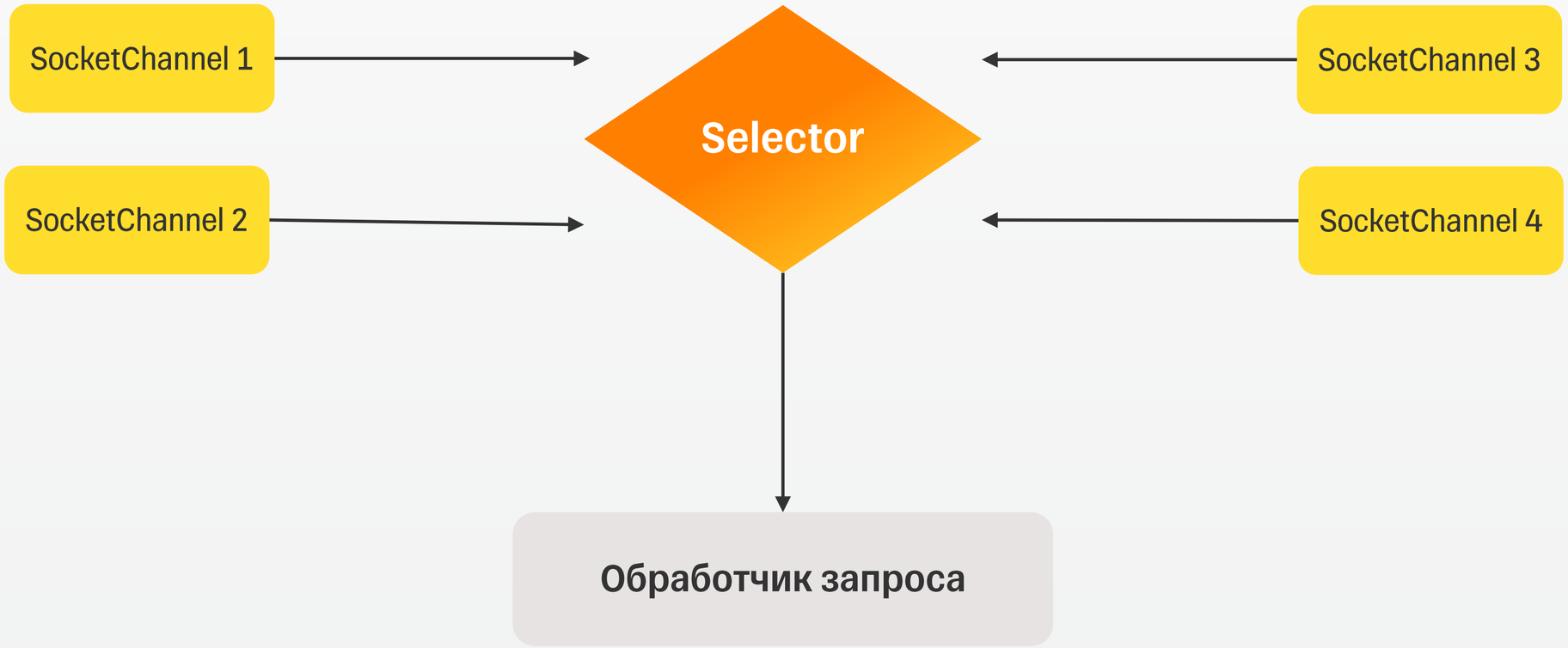
wepoll - epoll for Windows

https://github.com/piscisaureus/wepoll

Реализация в JDK:

https://github.com/openjdk/jdk/blob/master/src/java.base/windows/classes/sun/nio/ch/WEPoll.java

Как nio работает?

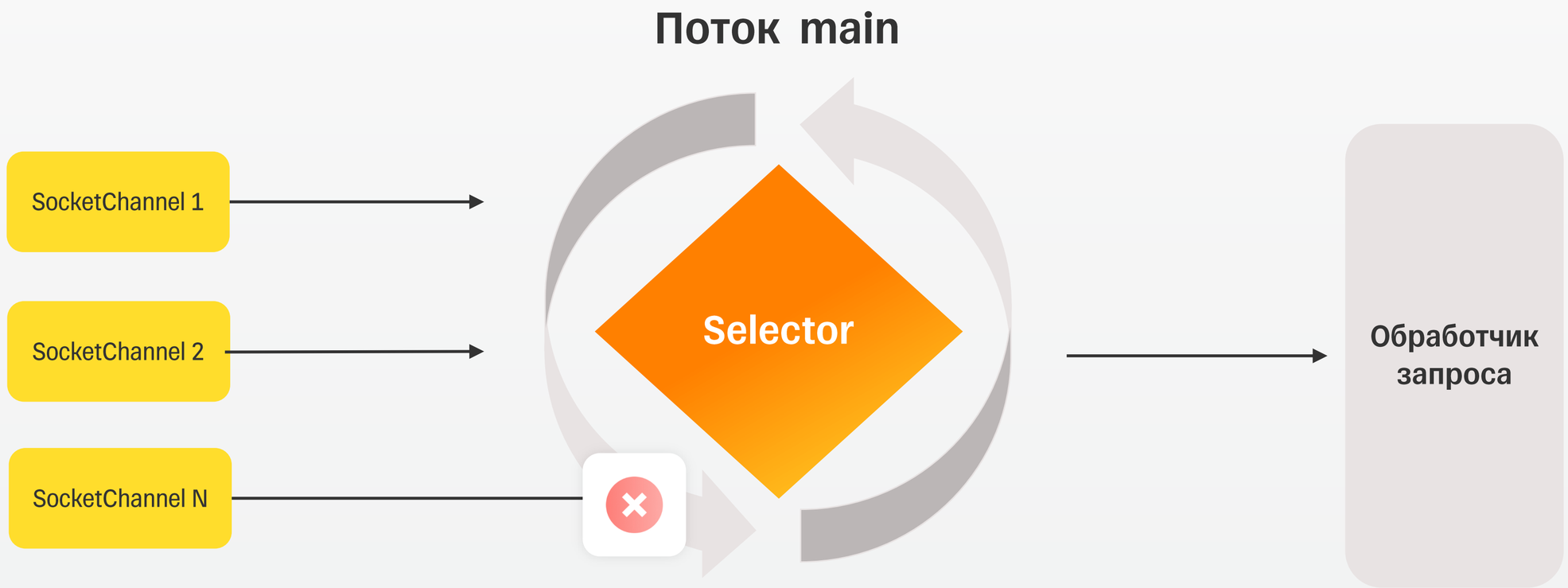


Вот оно идеальное решение?

NIO – универсальное решение для всех случаев?

Что сломалось?

```
while (!Thread.currentThread().isInterrupted()) {
    logger.info("waiting for client");
    selector.select(this::performIO, timeout: 10_000);
}
```



Запрос останавливает поток

Как чинить?

Не использовать nio, если требуется блокировка потока

Выполнять блокирующий код в отдельных потоках

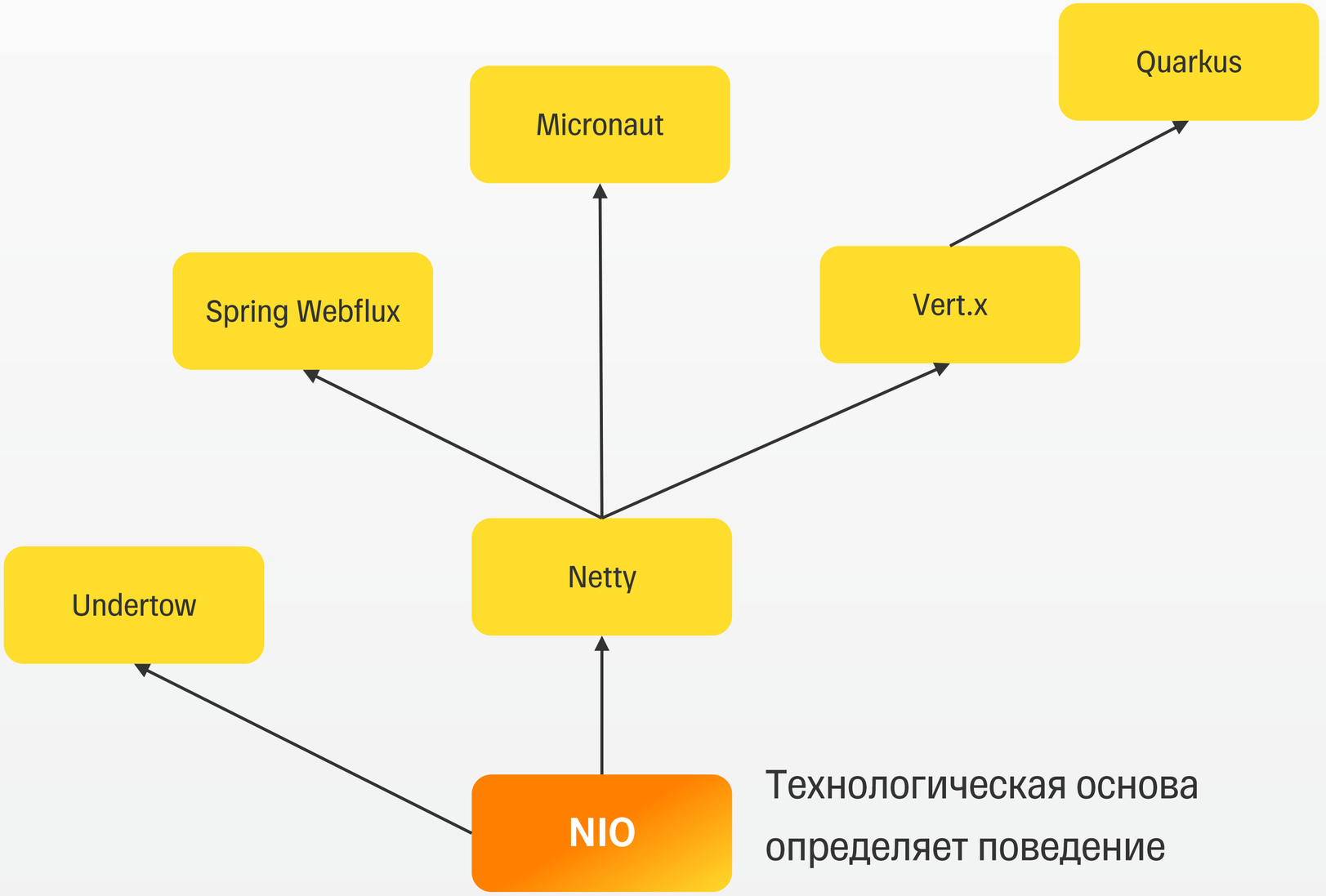
Virtual threads?

Заменят ли виртуальные потоки nio?

Наша цель: много запросов – один поток.

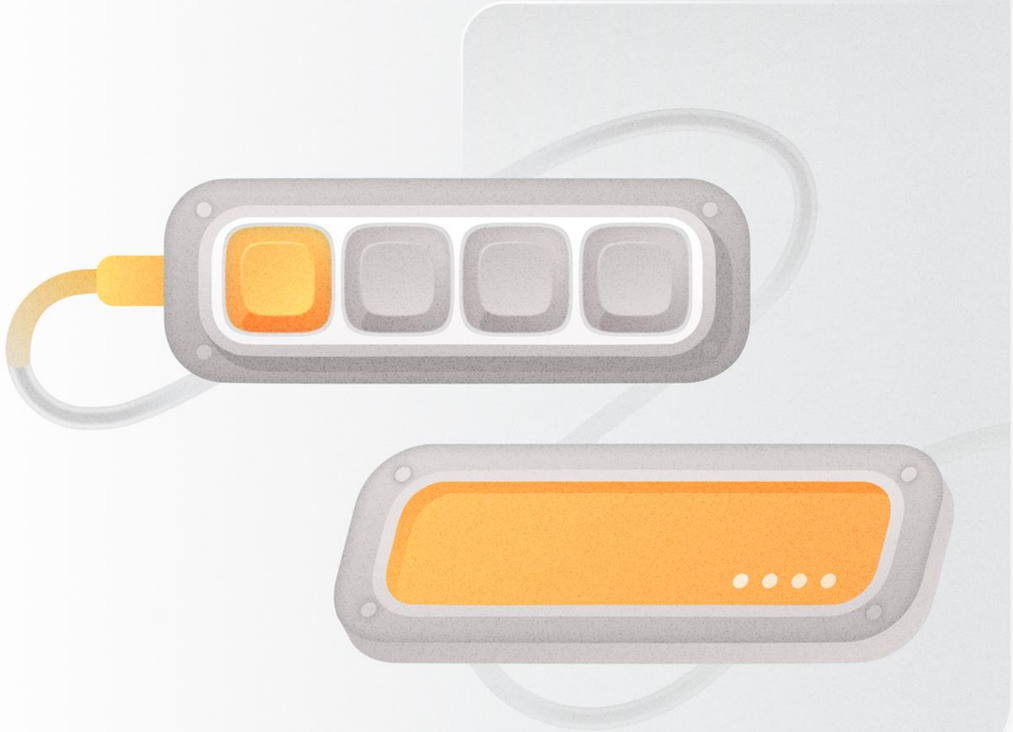
youtube:

NIO – одна из ключевых технологий Java



Технологическая основа определяет поведение потомков

Выводы



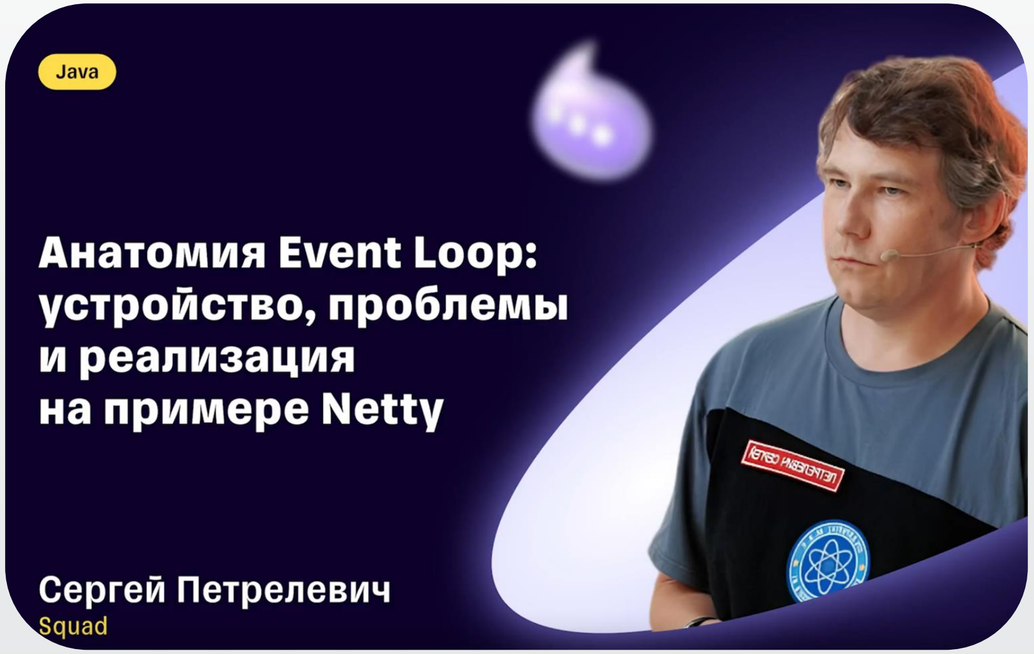
- Изучать принципы действия функционала
- Для небольших нагрузок блокирующий api – отличный вариант, не надо усложнять
- При больших нагрузках nio может отлично помочь
- Если есть блокирующий код, то его надо выполнять в отдельном потоке
- «Потомки» Nio наследуют ключевые свойства

Что дальше?

«Анатомия Event Loop: устройство, проблемы и реализация на примере Netty»

vk

youtube:



Петрелевич Сергей

@petrelevich

GitHub