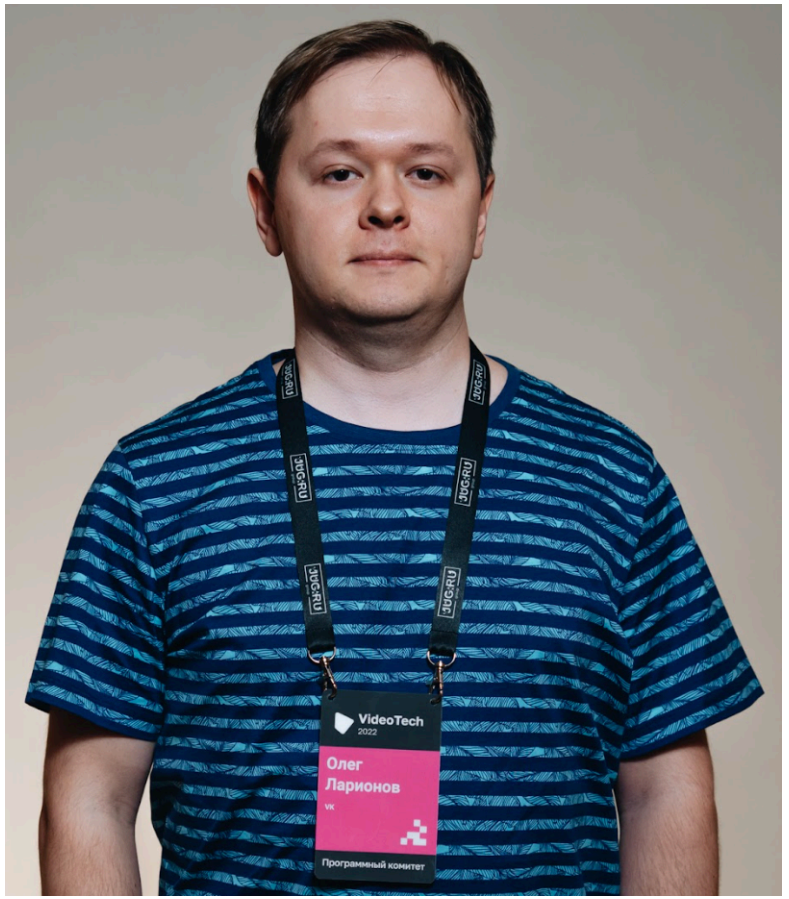
Эволюция раздачи видео VK

От 10 Гб/с по HTTP до 160 Гб/с по HTTPS с одной машины

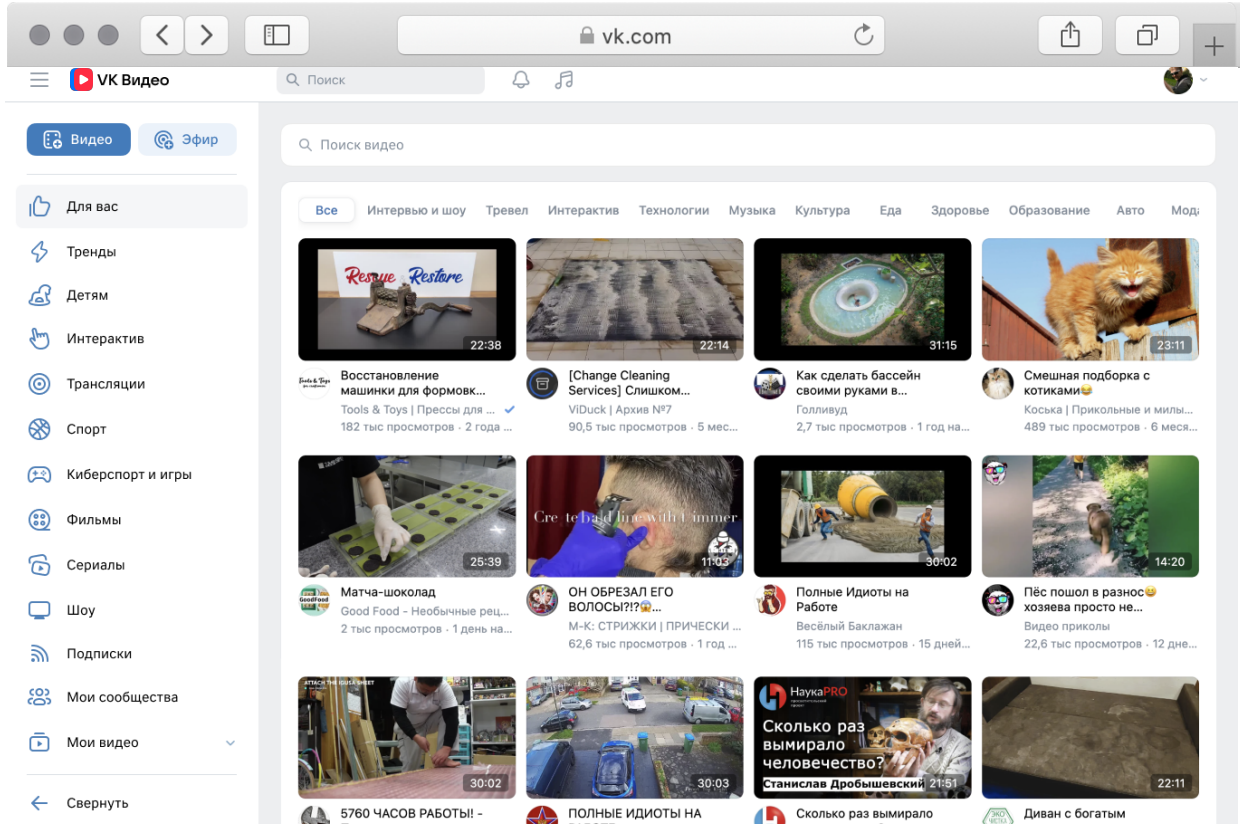
О себе

- Олег Ларионов
  руководитель группы Java разработки
- Java разработчик с 10-летним стажем
- OK -> Mail.ru -> VK Видео (ВКонтакте)



VK Видео сейчас

- 2.4 млрд просмотров в день
- 37 млн активная дневная аудитория
- 10 Тбит/с пиковый дневной трафик на видео
- 1 экзабайт (1024 * 1024 терабайт) дисков занято видеофайлами



Чем мы занимаемся

- Делаем видеосервисы на Java
  - Например, специализированный HTTP/1 сервер раздачи видео

Чем мы занимаемся

- Делаем видеосервисы на Java
  - Например, специализированный HTTP/1 сервер раздачи видео
- Пытаемся успеть за развитием железа и технологий

Поколение (Начало)	**G0 (2007)**	**G1 (2010)**	**G2 (2012)**	**G3 (2016)**	**G4 (2019)**	**G5 (2021)**
Процессор	Intel E5620	Intel E5630	Xeon 2620	Xeon 2660v4	Gold 62xx	Gold/ Platinum 63xx

Чем мы занимаемся

- Делаем видеосервисы на Java
  - Например, специализированный HTTP/1 сервер раздачи видео
- Пытаемся успеть за развитием железа и технологий
- Выжимаем из железа как можно больше
- Решаем проблемы при вертикальном масштабировании

Зачем мне слушать этот доклад?

- Путь боли

Зачем мне слушать этот доклад?

- Путь боли
- Новичкам: как сделать (http(s)) сервер обработки запросов на java

Зачем мне слушать этот доклад?

- Путь боли
- Новичкам: как сделать (http(s)) сервер обработки запросов на java
- Опытным: прочувствовать процесс разработки одного сервиса, узнать ручки «бесплатного» увеличения пропускной способности

План

- Реализуем простой http сервер на Java
- Перейдем на https
- Рассмотрим проблемы при вертикальном масштабировании
- Выжмем как можно больше из сервера на 2x Xeon Platinum 8380

Safe harbour

Все лгут

- Рассказываю собственный опыт
- Могу ошибаться (обычно неумышленно)
- Проверяйте перед апдейтом прода
- Не катите апдейт в пятницу

База video-download

Поднимаем свой сервер

1. База и запуск
2. Внедряем HTTPS
3. Прокачиваем до 100G
4. Последние достижения

HTTP/1 сервер на основе one-nio

- Простой, но мощный фреймворк
- Собственная разработка Одноклассников
- Java + JNI (+ Linux)

one-nio / src / one / nio /

Name
..
async
cluster
compiler
config
gen
http
lock
lz4
mem
mgt
net
os

HTTP/1 сервер на основе one-nio

- Простой, но мощный фреймворк
- Собственная разработка Одноклассников
- Java + JNI (+ Linux)
- https://github.com/odnoklassniki/one-nio
- https://youtu.be/gIh0X-RkftY

Quick start guide

```
public class MyServer extends HttpServer {

}
```

Quick start guide

```
public class MyServer extends HttpServer {



    @Path("/hello")
    public Response hello(Request request, HttpSession session) {
        return Response.ok( plainText: "Welcome to Joker 2024, " + session.getRemoteHost());
    }
}
```

Quick start guide

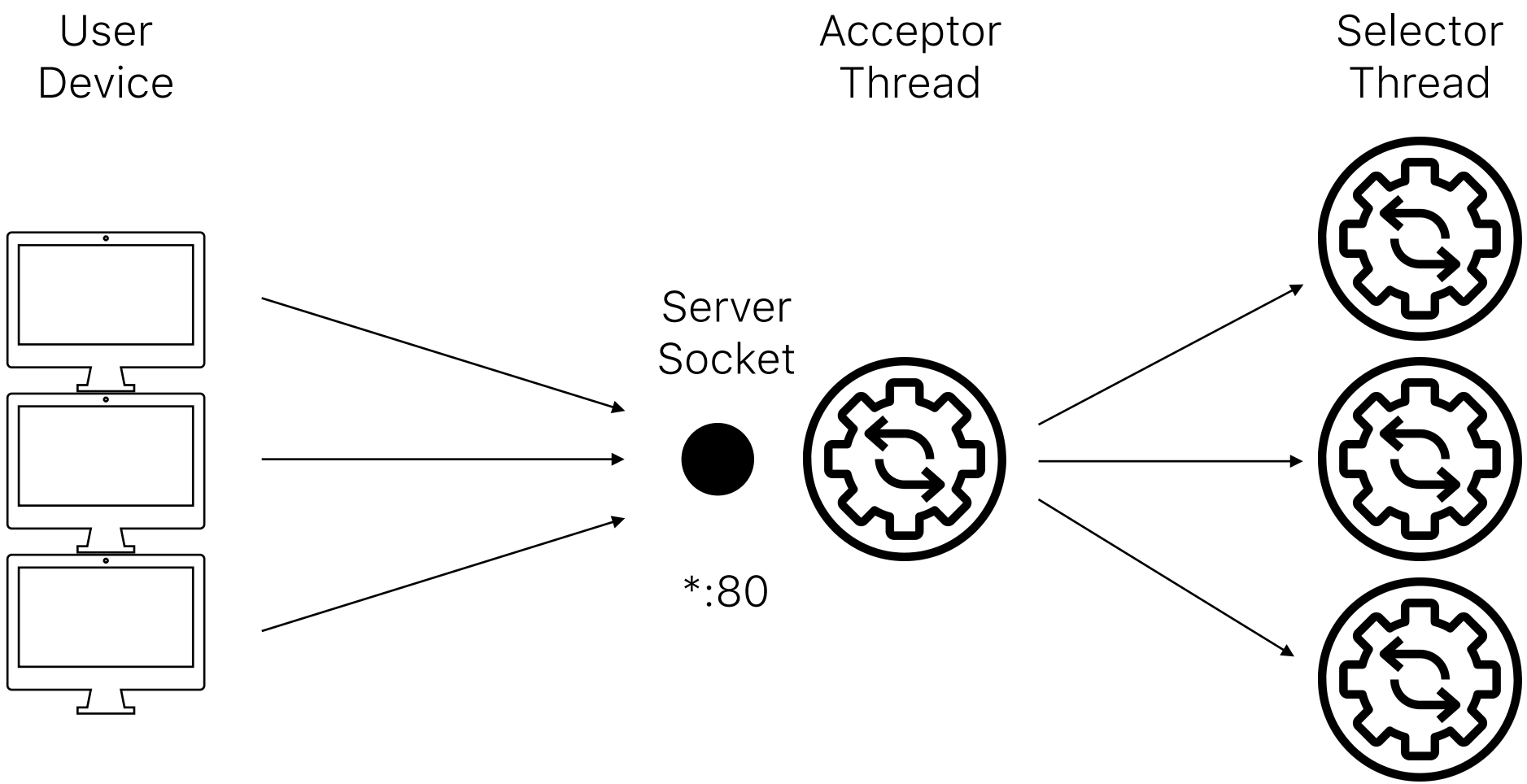
```
public class MyServer extends HttpServer {
    public MyServer(HttpServerConfig config) throws IOException {
        super(config);
    }

    @Path("/hello")
    public Response hello(Request request, HttpSession session) {
        return Response.ok( plainText: "Welcome to Joker 2024, " + session.getRemoteHost());
    }
}
```

Quick start guide

```java
@Config
public class ServerConfig {
    public AcceptorConfig[] acceptors;
    public boolean multiAcceptor;
    public int selectors;
    public boolean affinity;
    public int minWorkers;
    public int maxWorkers;
    @Converter(method = "time")
    public int queueTime;
    @Converter(method = "time")
    public int keepAlive;
    public int threadPriority = Thread.NORM_PRIORITY;
    public SchedulingPolicy schedulingPolicy;
    public boolean closeSessions;
    public boolean pinAcceptors;
```

```java
@Config
public class AcceptorConfig {
    public int threads = 1;
    public String address = "0.0.0.0";
    public int port;
    @Converter(method = "size")
    public int recvBuf;
    @Converter(method = "size")
    public int sendBuf;
    public int tos;
    public int backlog = 128;
    @Converter(method = "size")
    public int notsentLowat;
    public boolean keepAlive = true;
    public boolean noDelay = true;
    public boolean tcpFastOpen = true;
    public boolean deferAccept;
    public boolean reusePort;
    public boolean thinLto;
    public SslConfig ssl;
}
```

Event Loop модель



Event Loop модель

- На Linux: через JNI используем системные вызовы accept/send/recv/ epoll_ctl/epoll_wait/...

```
class NativeSocket extends Socket {
    ...
    final native void connect0(Object address, int port) throws IOException;
    final native void bind0(Object address, int port) throws IOException;
    final native int accept0(boolean nonblock) throws IOException;
    native long sendFile0(int sourceFD, long offset, long count) throws IOException;
    final native int sendTo0(byte[] data, int offset, int size, int flags, Object address, int port) throws IOException;
    final native int sendTo1(long buf, int size, int flags, Object address, int port) throws IOException;
    final native int recvFrom0(byte[] data, int offset, int maxSize, int flags, AddressHolder holder) throws IOException;
    final native int recvFrom1(long buf, int maxSize, int flags, AddressHolder holder) throws IOException;
    final native int sendMsg0(byte[] data, int cmsgType, int[] cmsgData, int flags) throws IOException;
    final native int recvMsg0(byte[] data, Msg msg, int flags) throws IOException;
}
```

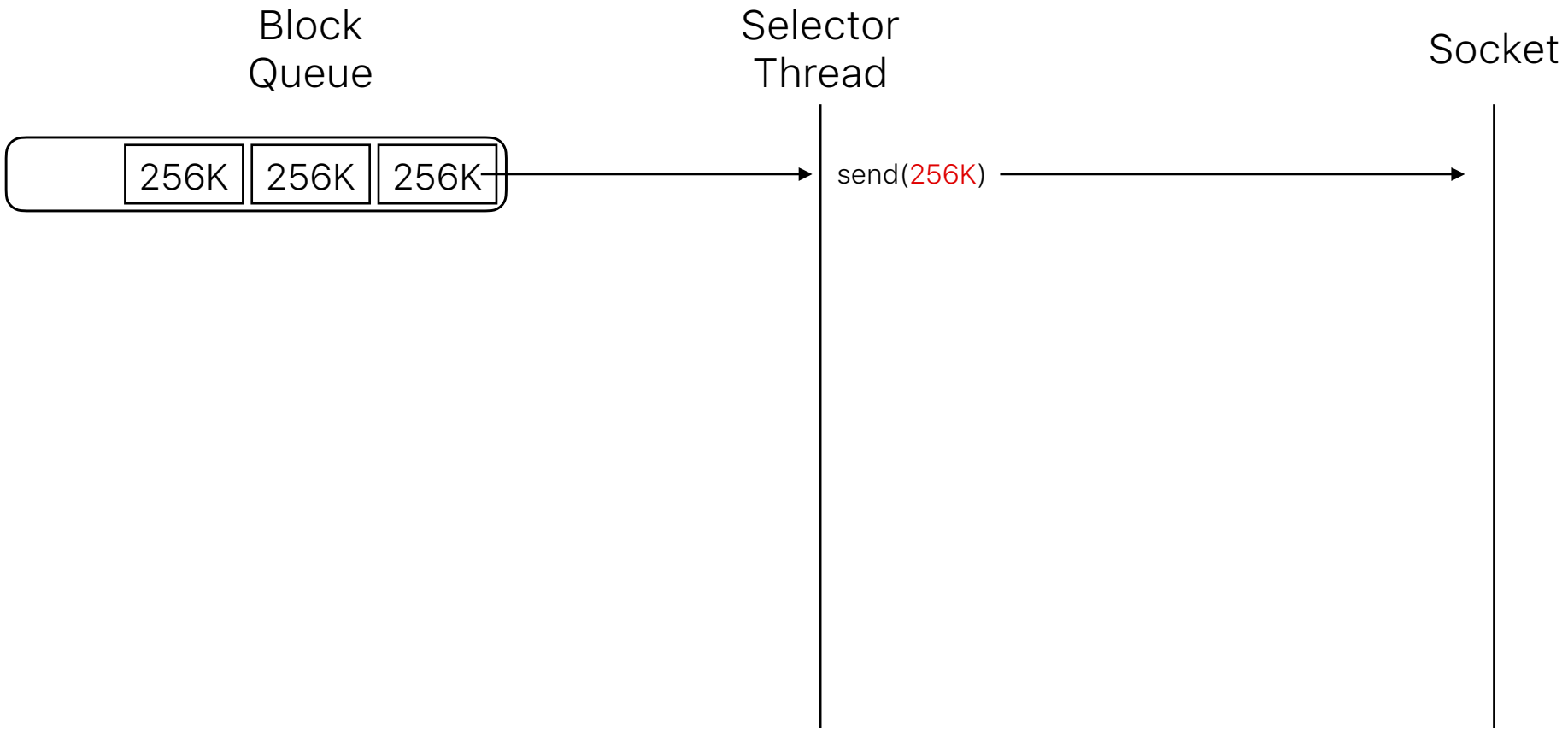
- Не на Linux: используется JavaSelector и JavaSocket из java.nio

Event Loop модель

- Читаем
  - ssize_t recv(int sockfd, void *buf, size_t len, int flags);
- Пишем
  - ssize_t send(int sockfd, const void *buf, size_t len, int flags);
  - ssize_t sendfile(int out_fd, int in_fd, off_t * offset, size_t count);
- Мультиплексируем соединения
  - API epoll: epoll_create/epoll_ctl/epoll_wait

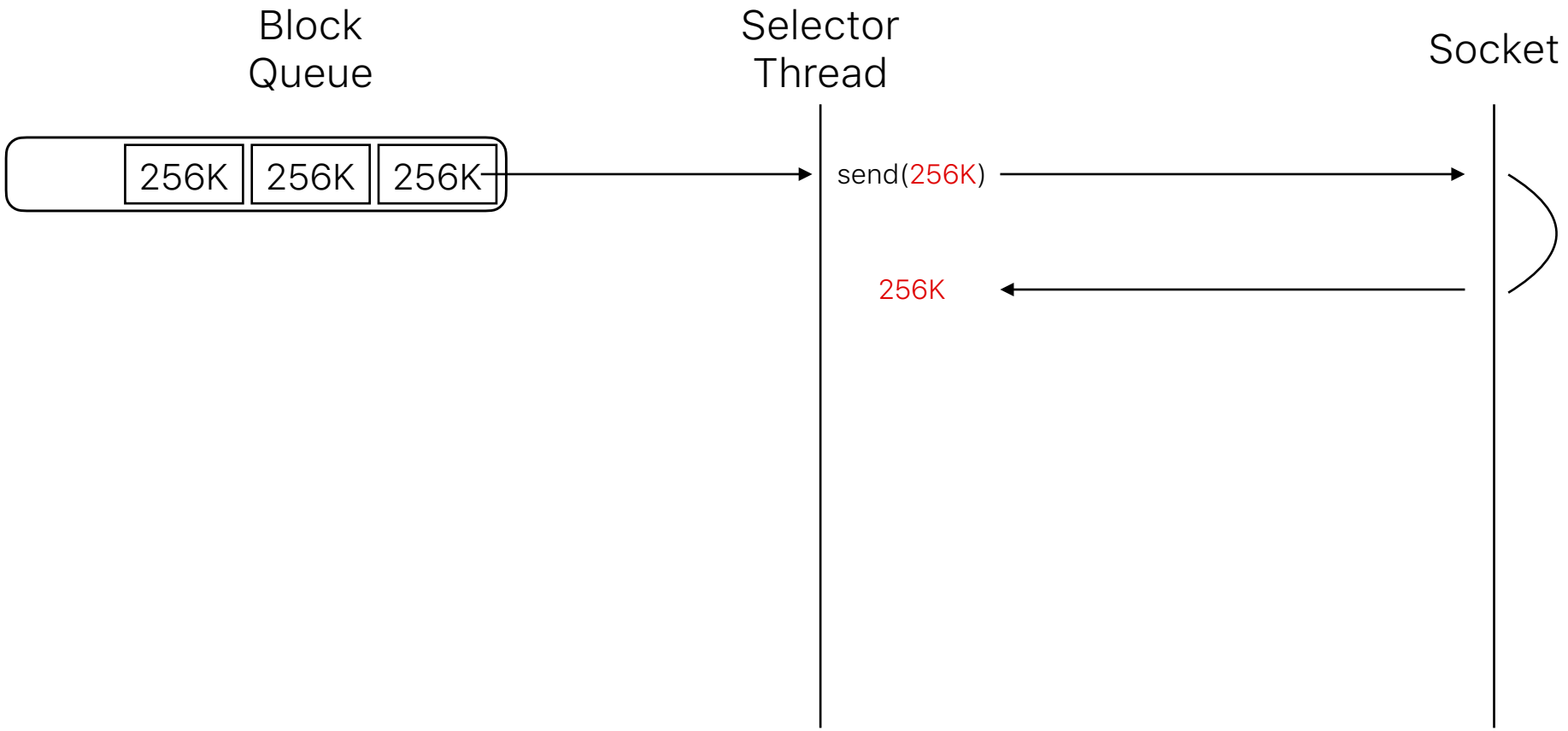
Event Loop модель

- Блокирующий режим



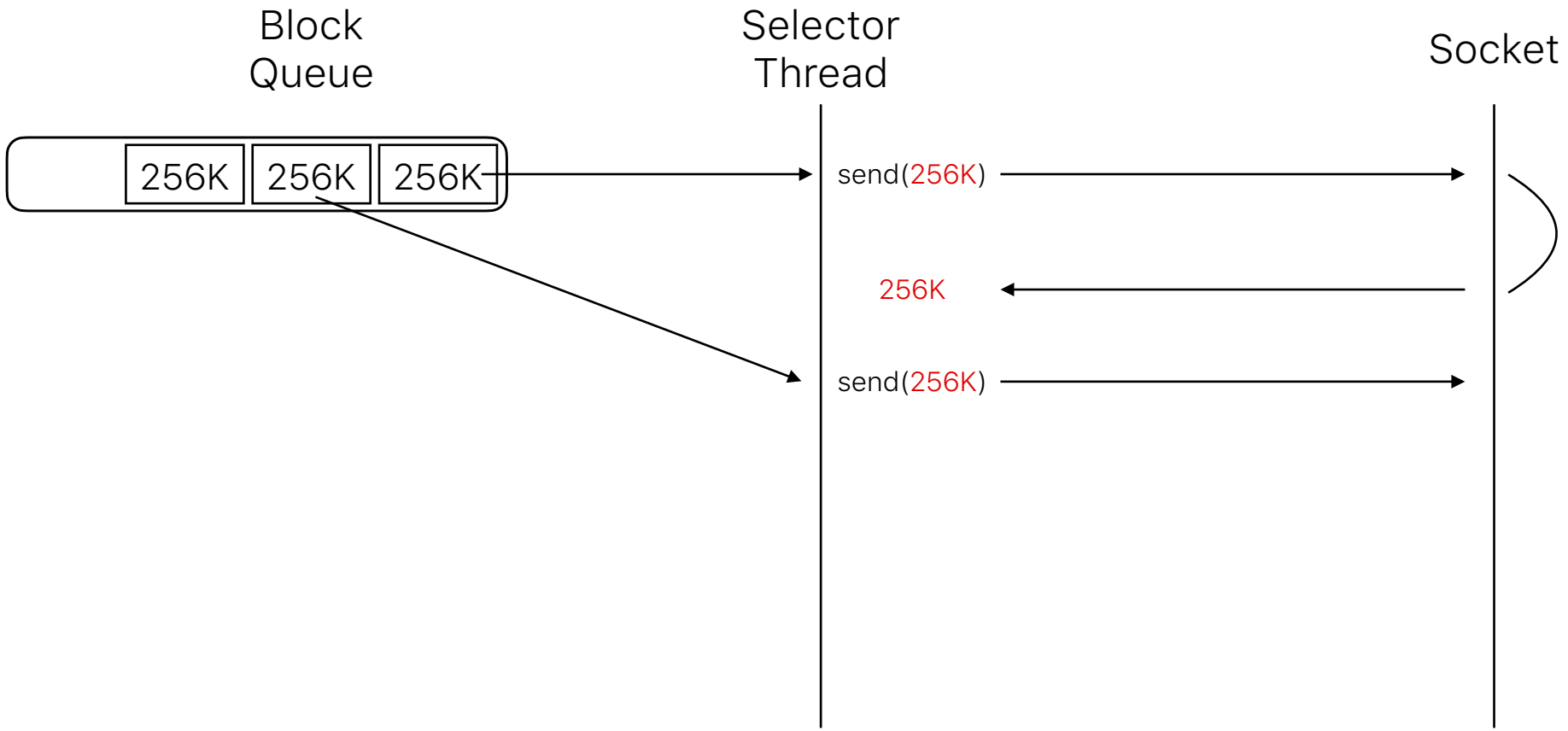
Event Loop модель

- Блокирующий режим



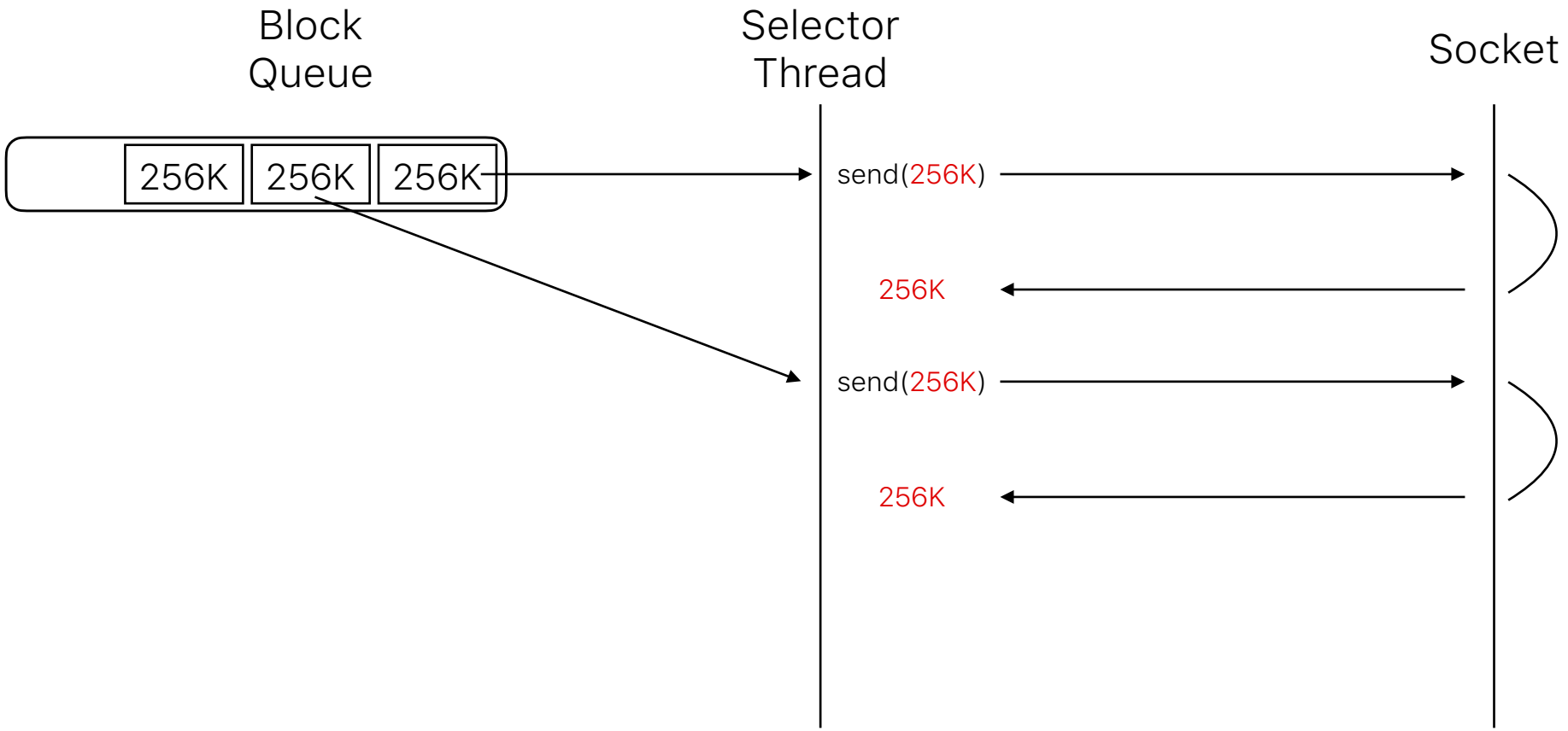
Event Loop модель

- Блокирующий режим



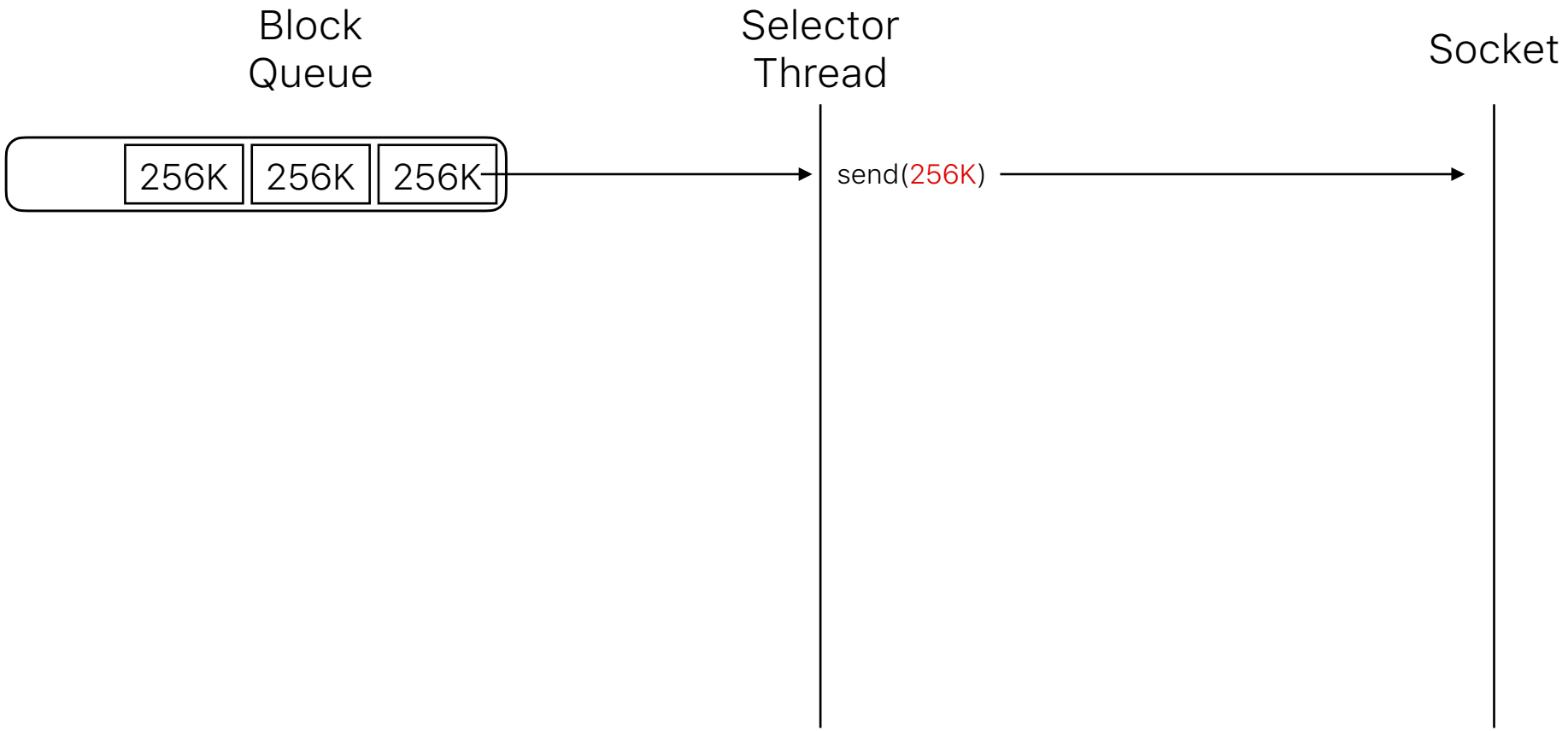
Event Loop модель

- Блокирующий режим



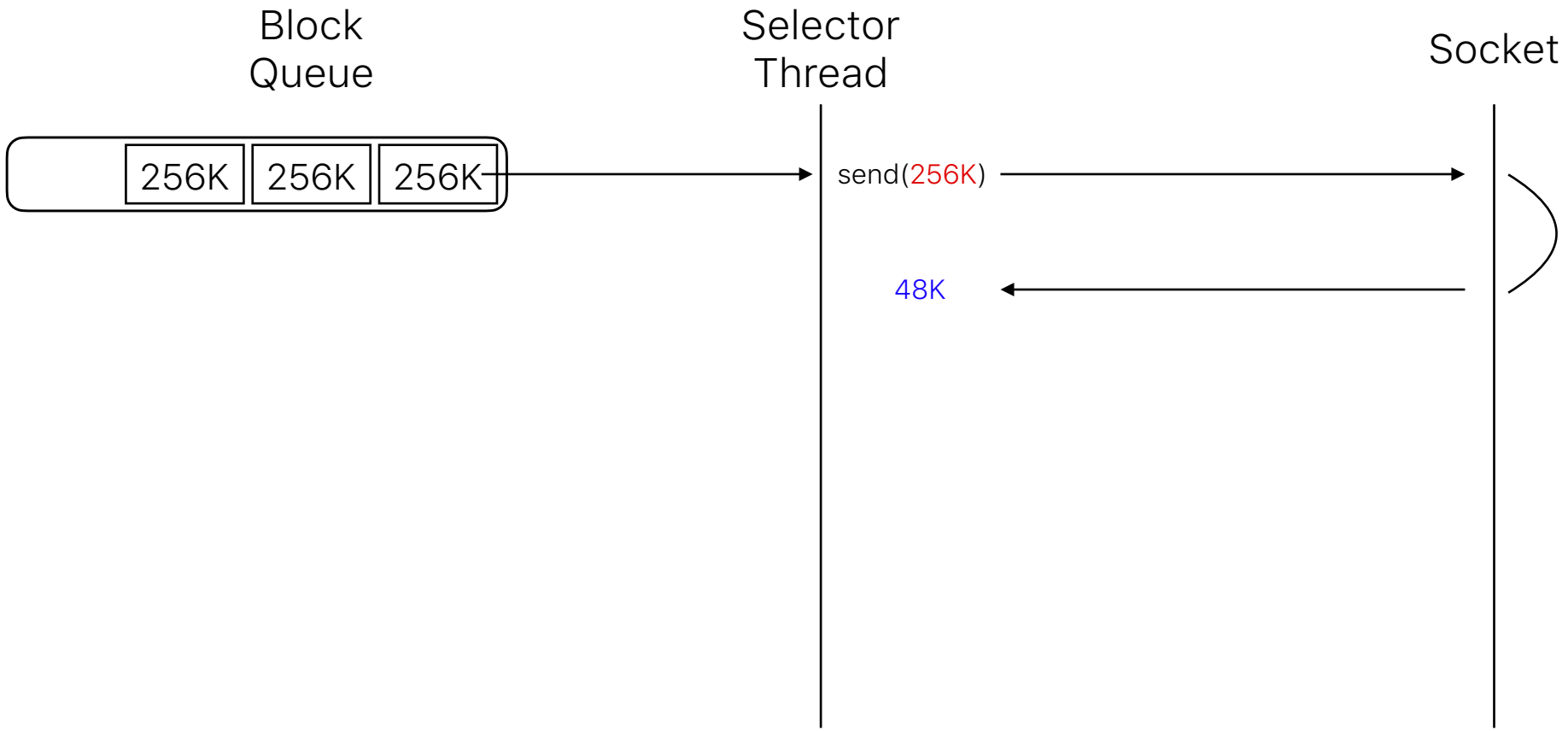
Event Loop модель

- Неблокирующий режим (флаг O_NONBLOCK)



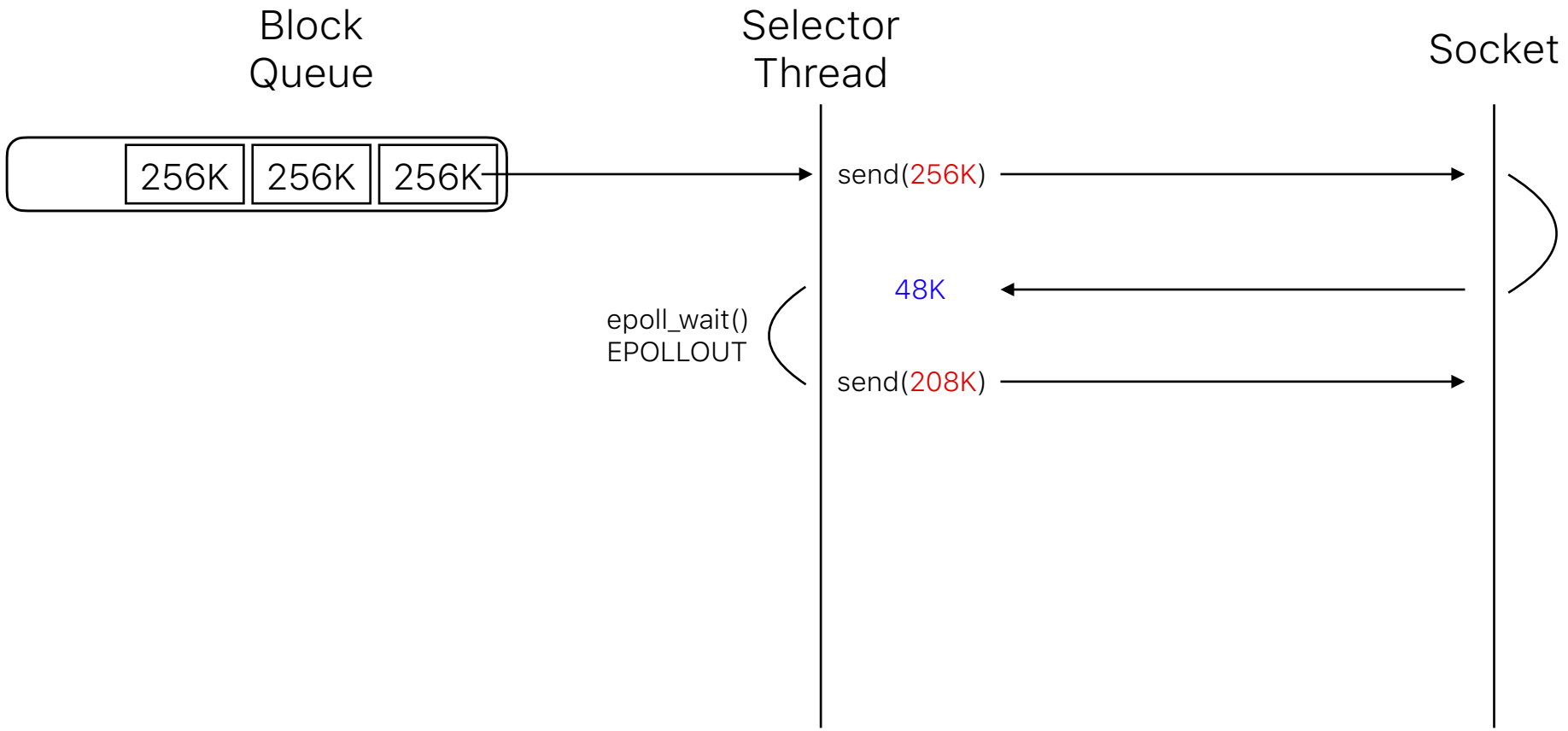
Event Loop модель

- Неблокирующий режим (флаг O_NONBLOCK)



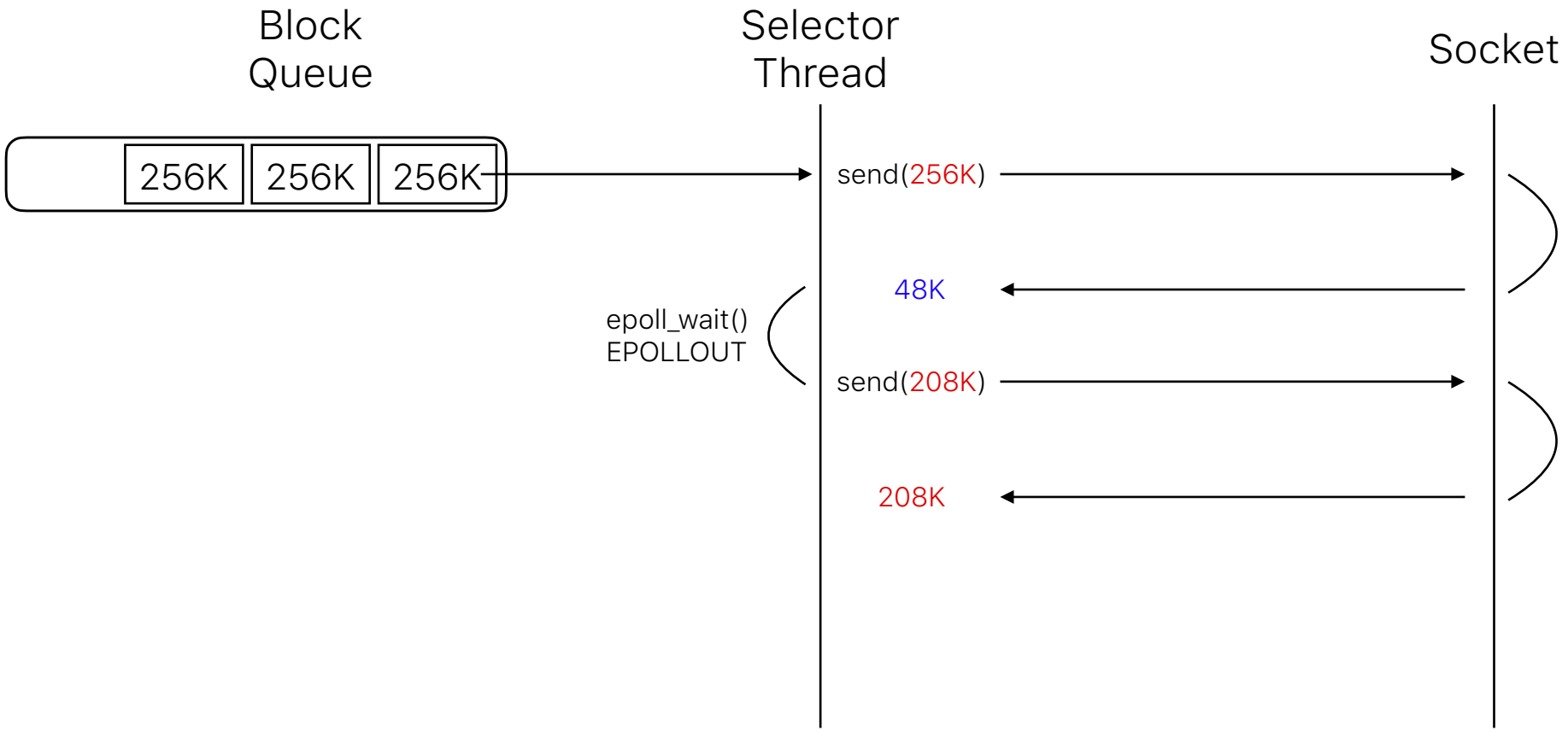
Event Loop модель

- Неблокирующий режим (флаг O_NONBLOCK)



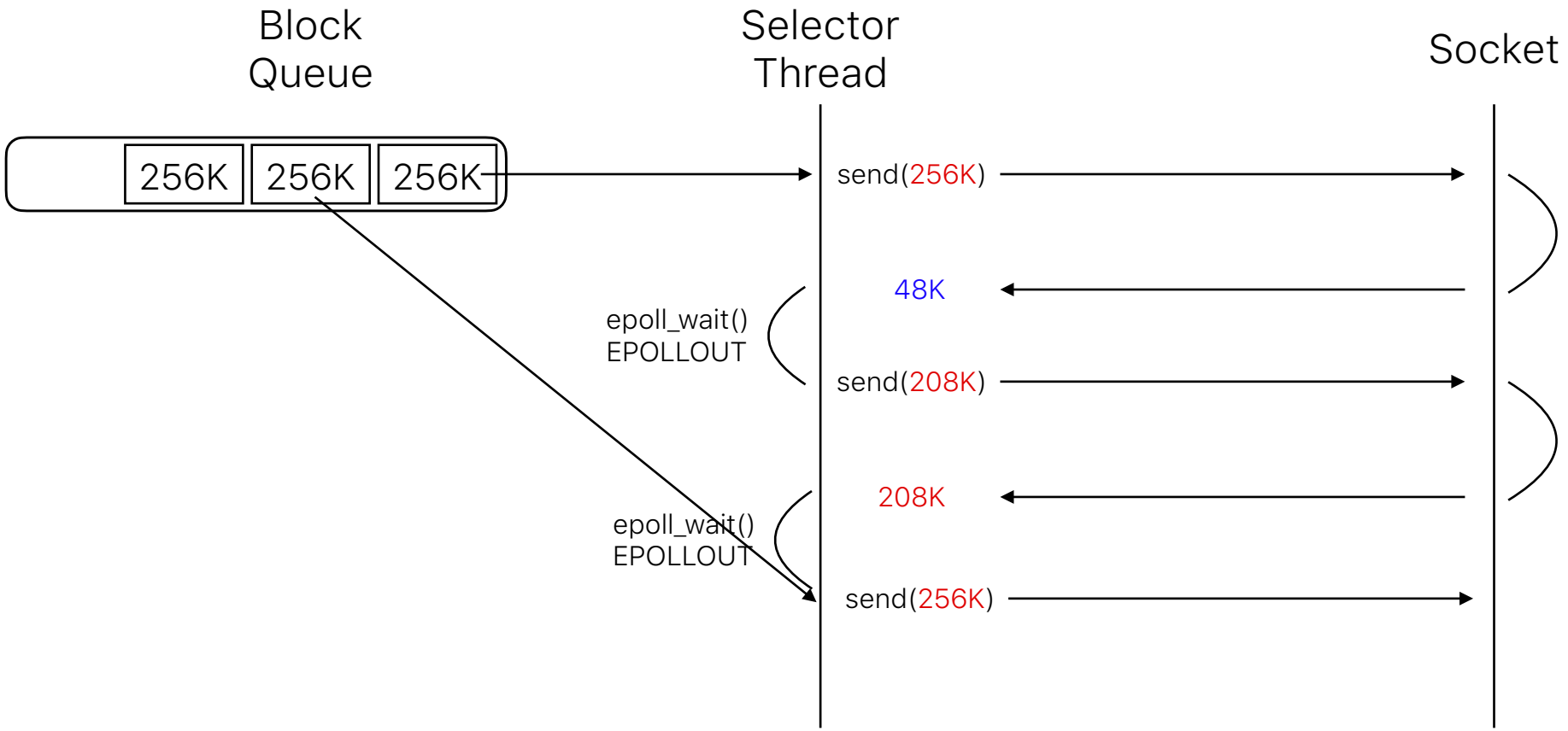
Event Loop модель

- Неблокирующий режим (флаг O_NONBLOCK)



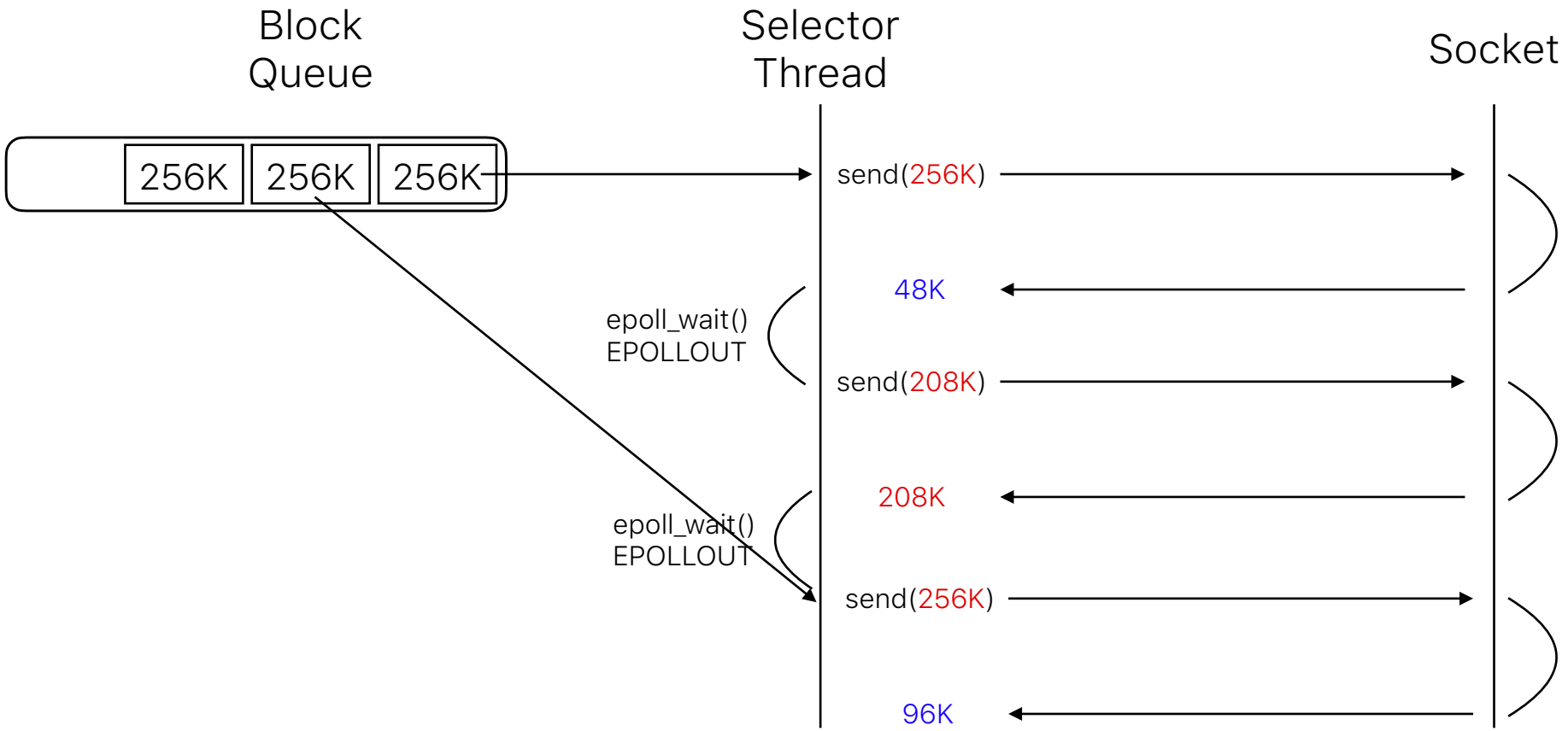
Event Loop модель

- Неблокирующий режим (флаг O_NONBLOCK)

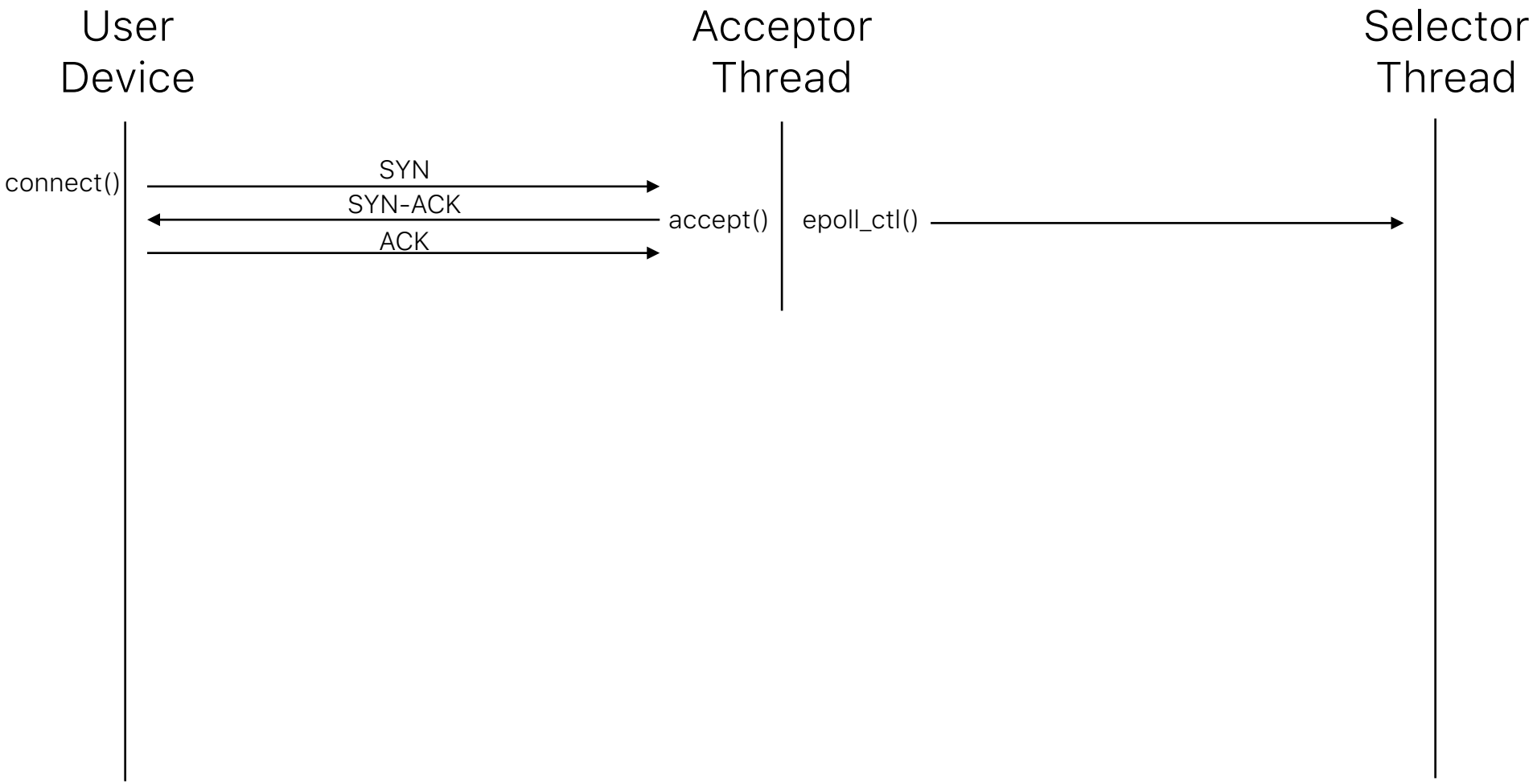


Event Loop модель

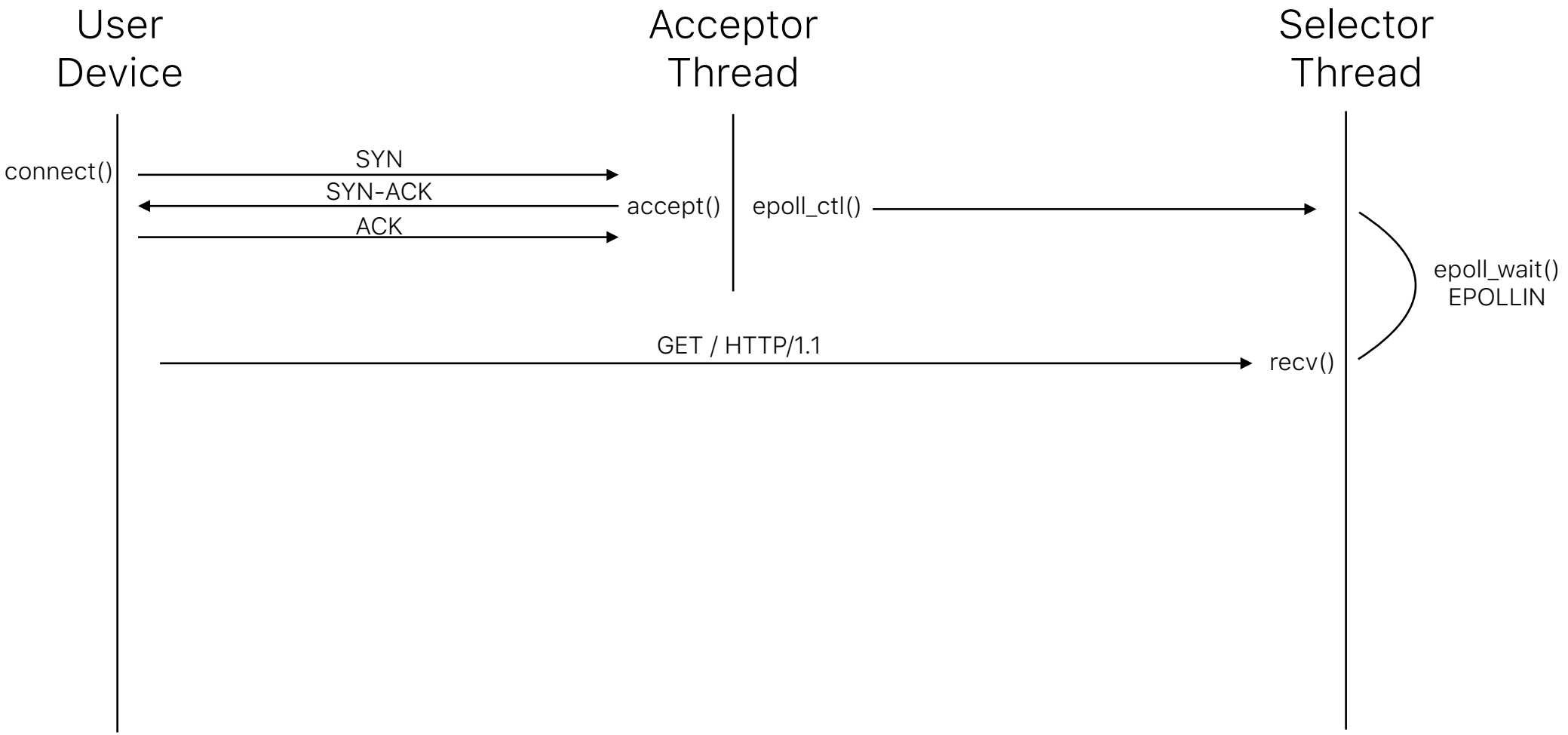
- Неблокирующий режим (флаг O_NONBLOCK)



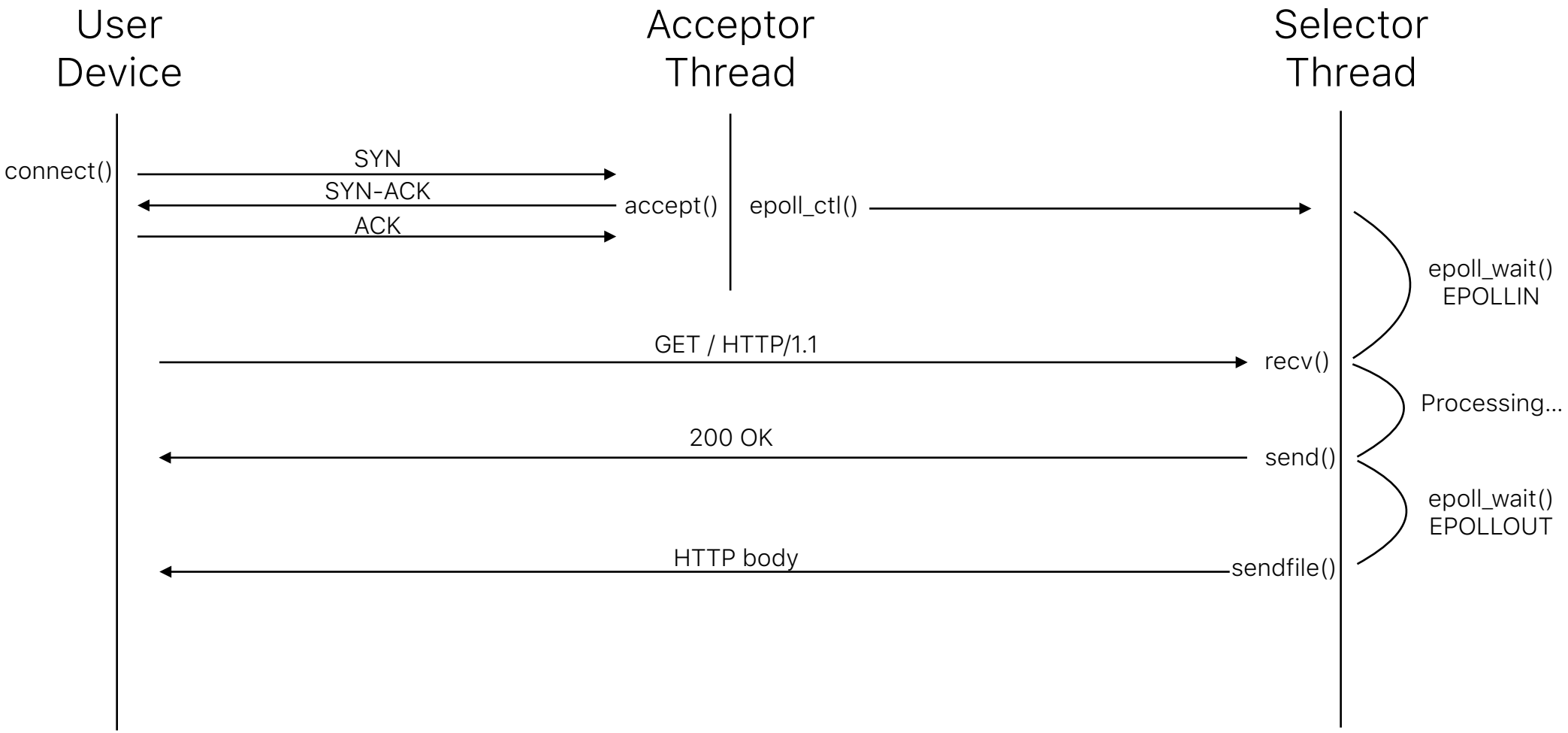
Event Loop модель



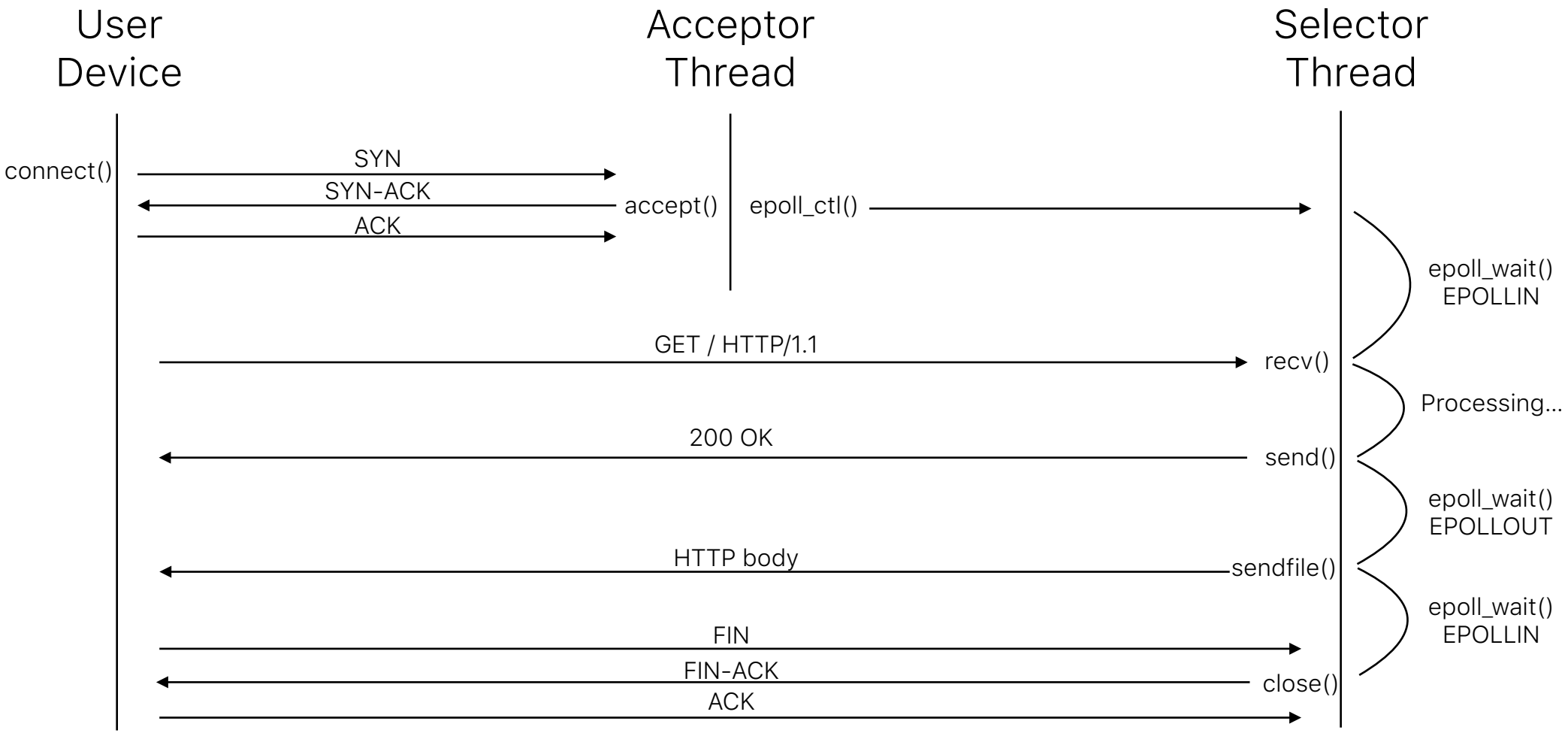
Event Loop модель



Event Loop модель



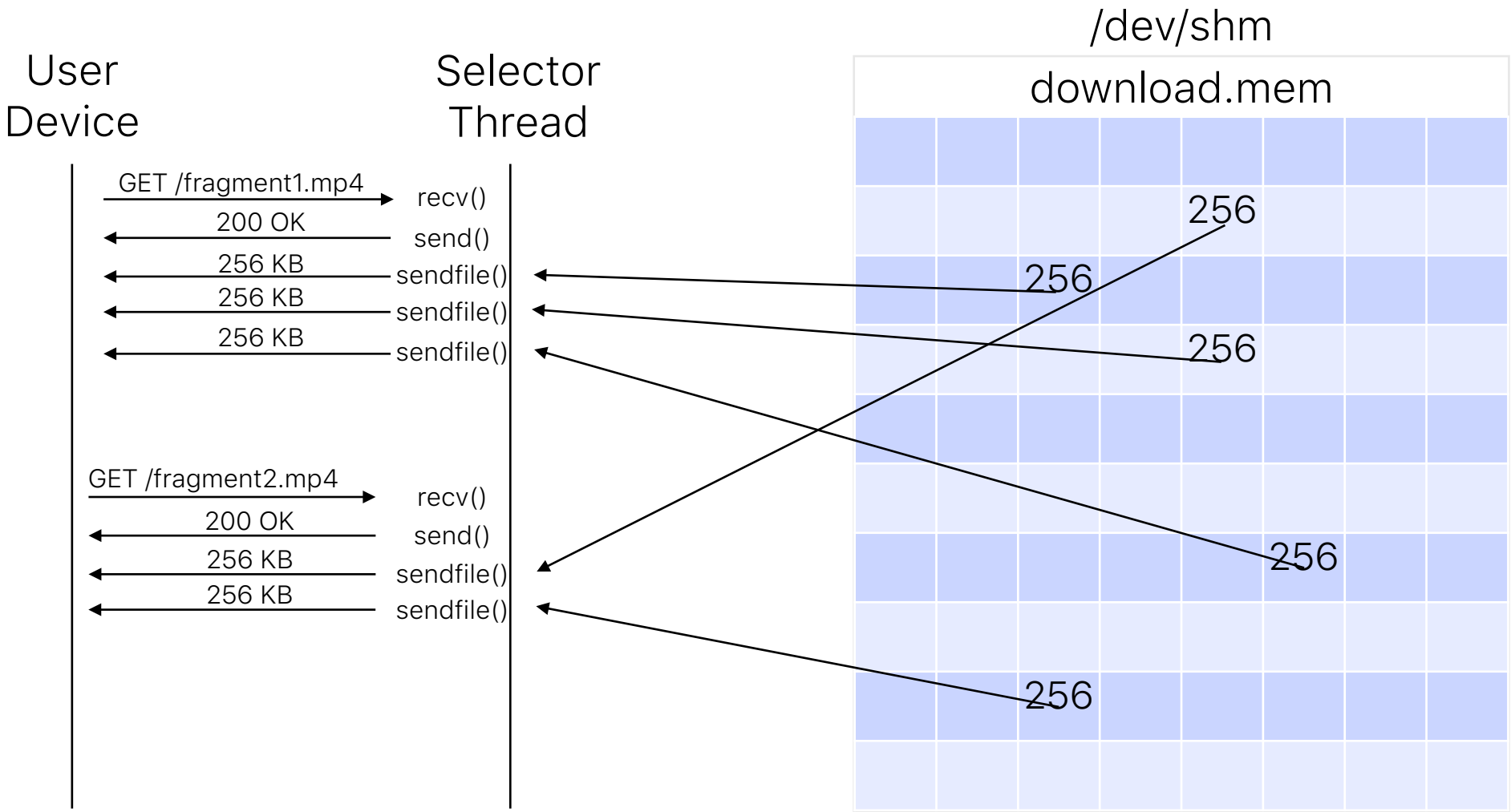
Event Loop модель



Два уровня кеширования

- offheap кеш в shared memory (LRU)
  https://youtu.be/h9AICWfaEVQ

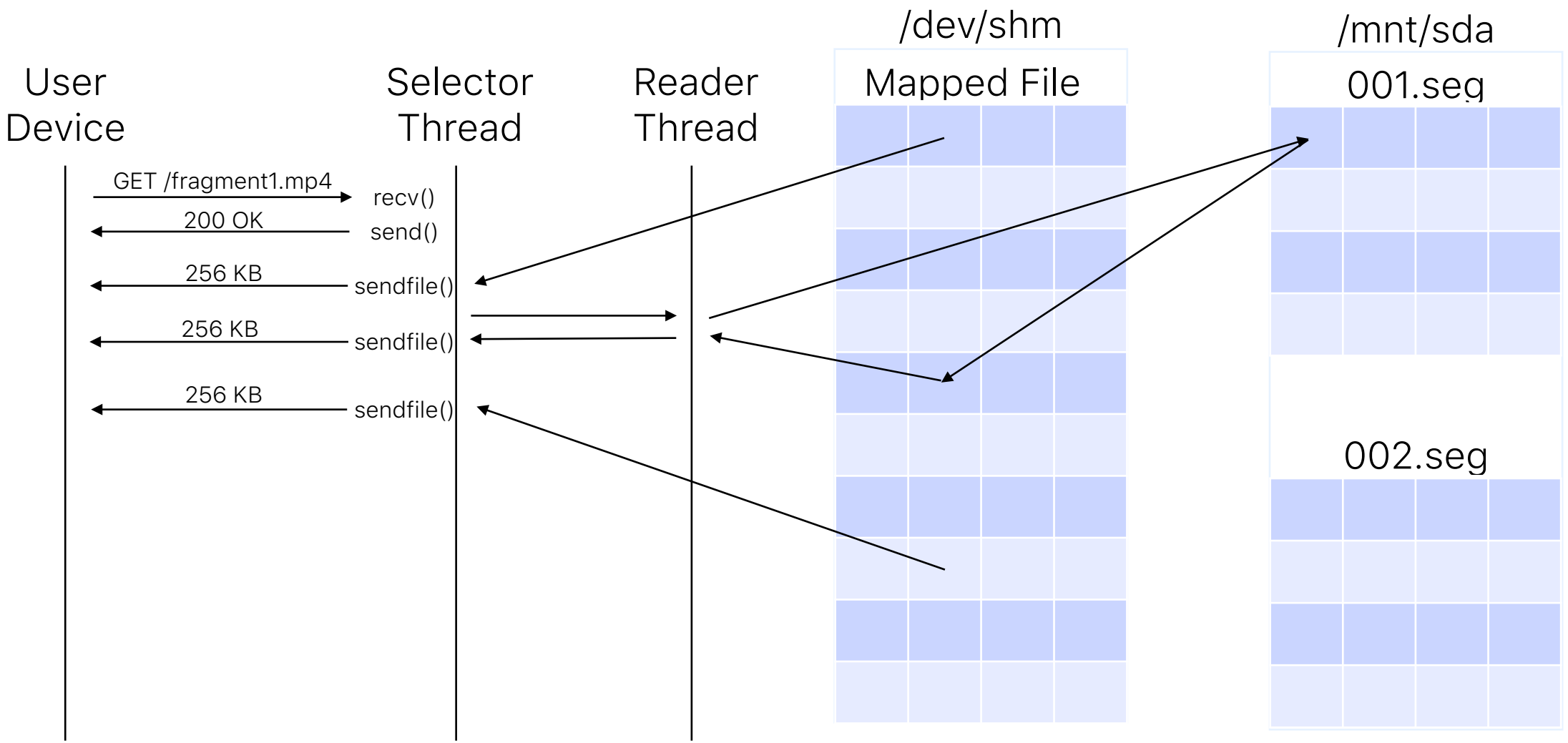
Два уровня кеширования



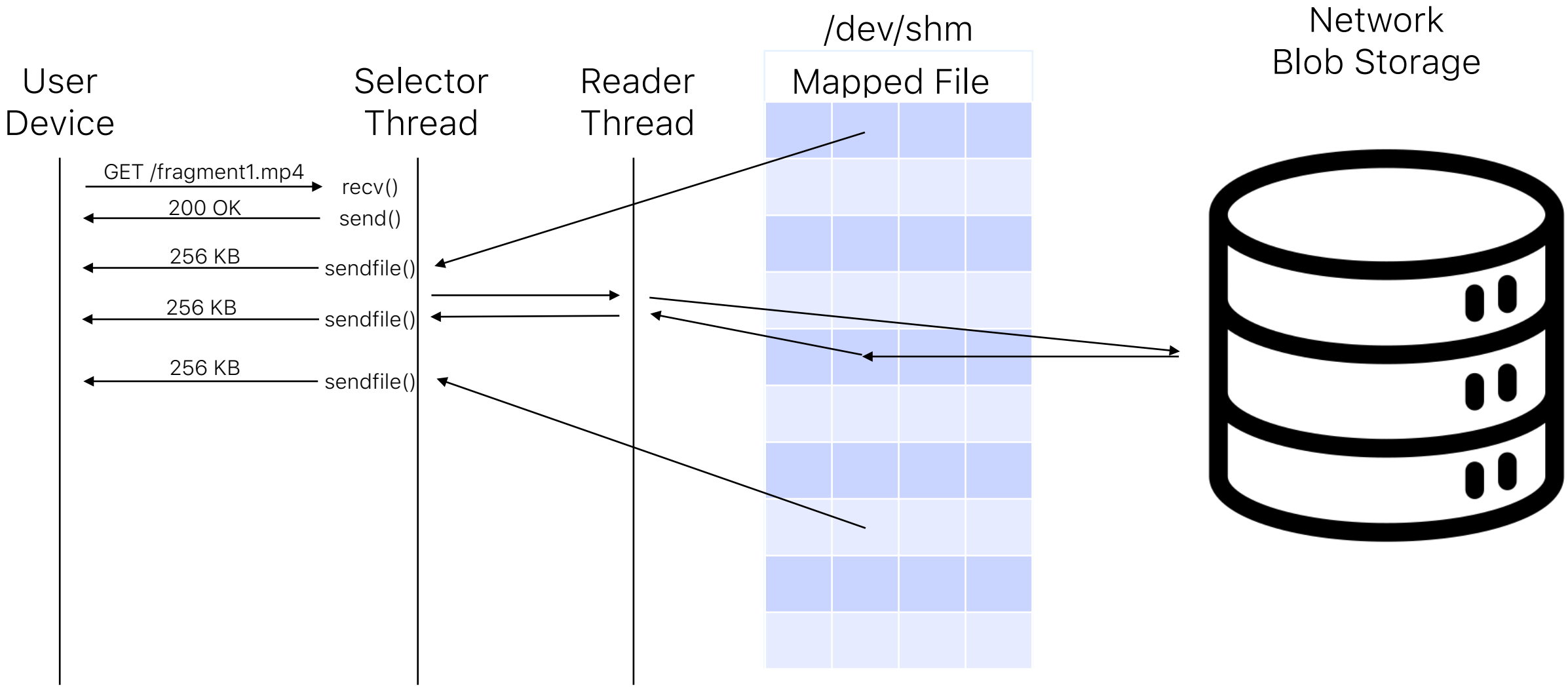
Два уровня кеширования

- дисковый кеш на hdd/ssd/nvme (FIFO)
  https://youtu.be/uuGbbJhS7o8

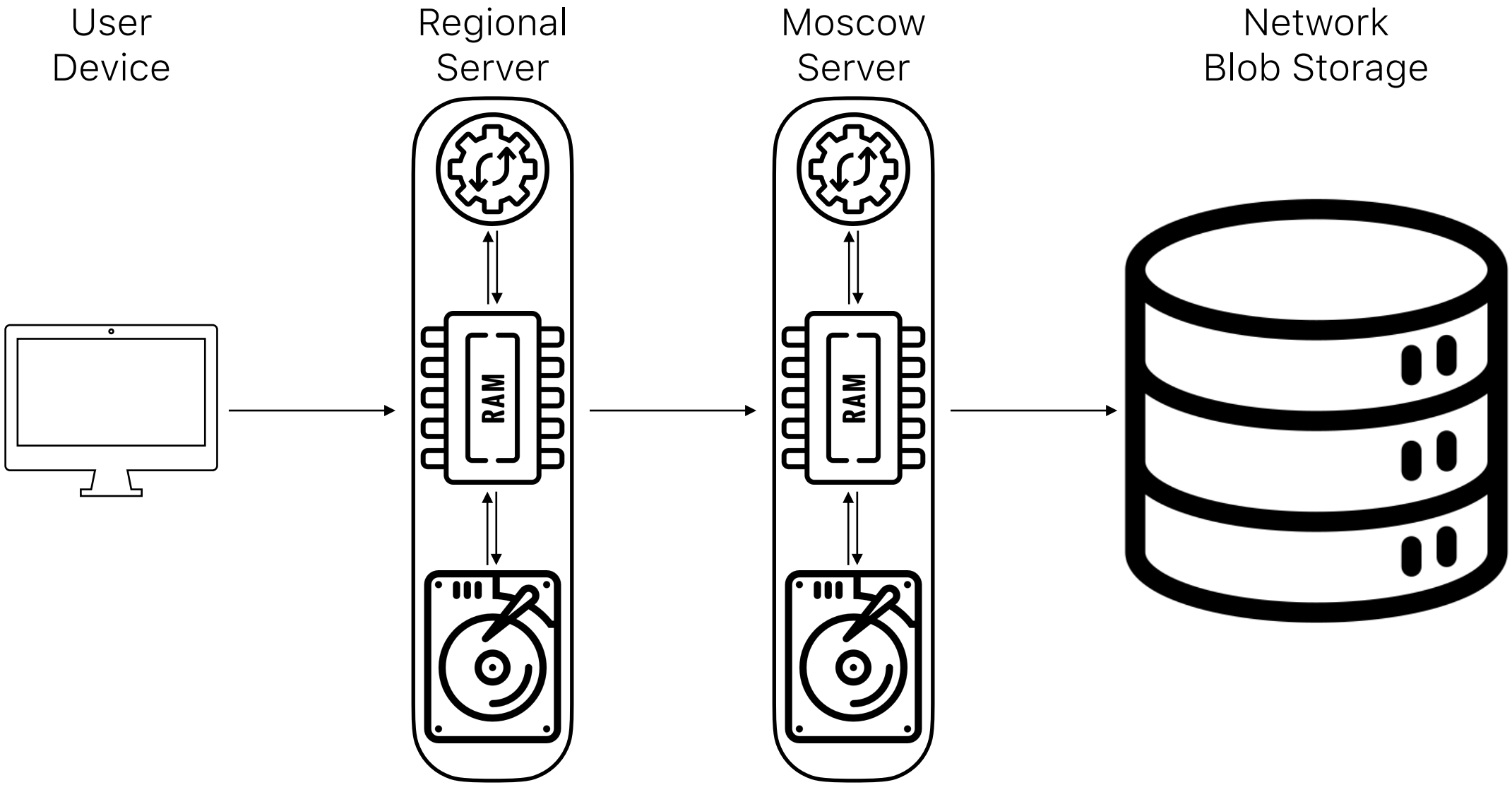
Два уровня кеширования



Два уровня кеширования

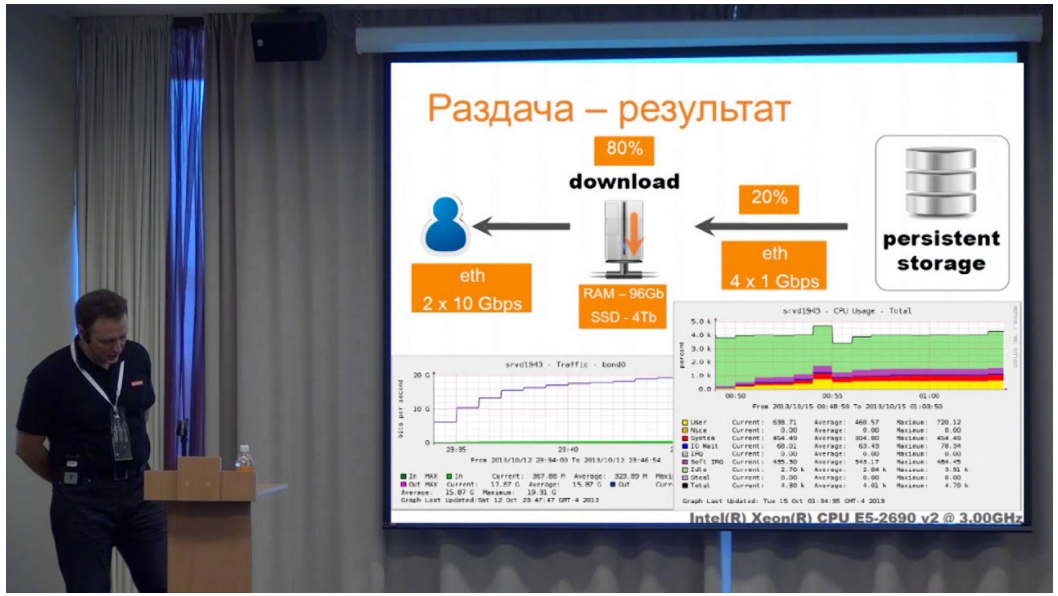


Взгляд сверху



Подробнее

- Платформа для Видео сроком в квартал
  https://youtu.be/2bDZj9_hpuI



HDCONF
HIGHLOAD DEV CONF

- Кадры решают всё
  https://youtu.be/my6h8dLvGks

«Кадры решают все, или стриминг видео в Одноклассниках»

Александр Тоболь

Запуск

2013 год

1. База и запуск
2. Внедряем HTTPS
3. Прокачиваем до 100G
4. Последние достижения

Конфигурация серверов

- Xeon 2620 (6C/12T)
- 128 GB RAM
- 20 Gbps NIC
- HDD
- ядро 2.6

Конфигурация серверов

Наш сервер

- Xeon 2620 (6C/12T)
- 128 GB RAM
- 20 Gbps NIC
- HDD
- ядро 2.6

Mac Studio (2023)

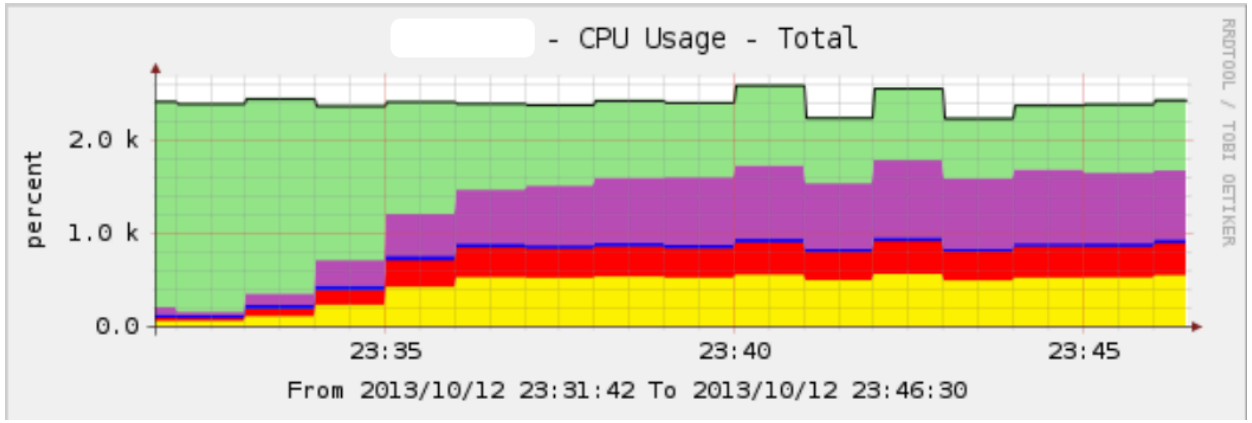
- M2 Ultra (24C)
- 192 GB RAM
- 10 Gbps NIC
- SSD

Конфигурация серверов

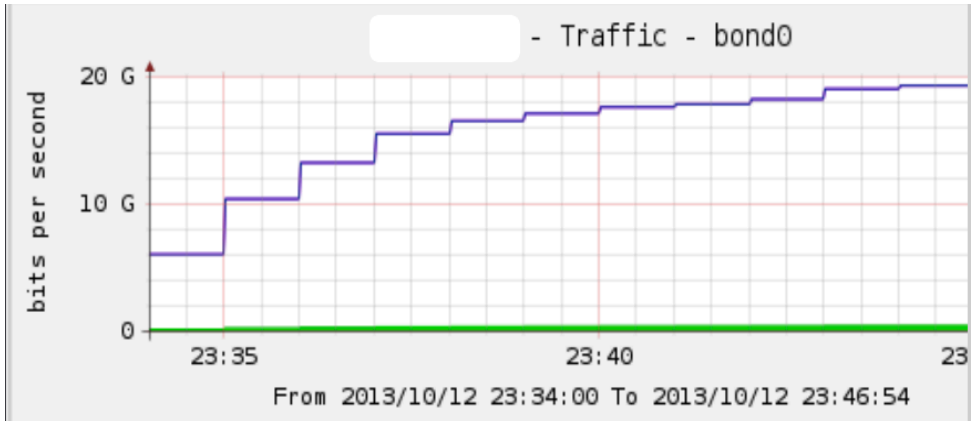
```
selectors: 24

acceptors:
  - port: 80
```

Ожидание и реальность

- Как должно быть



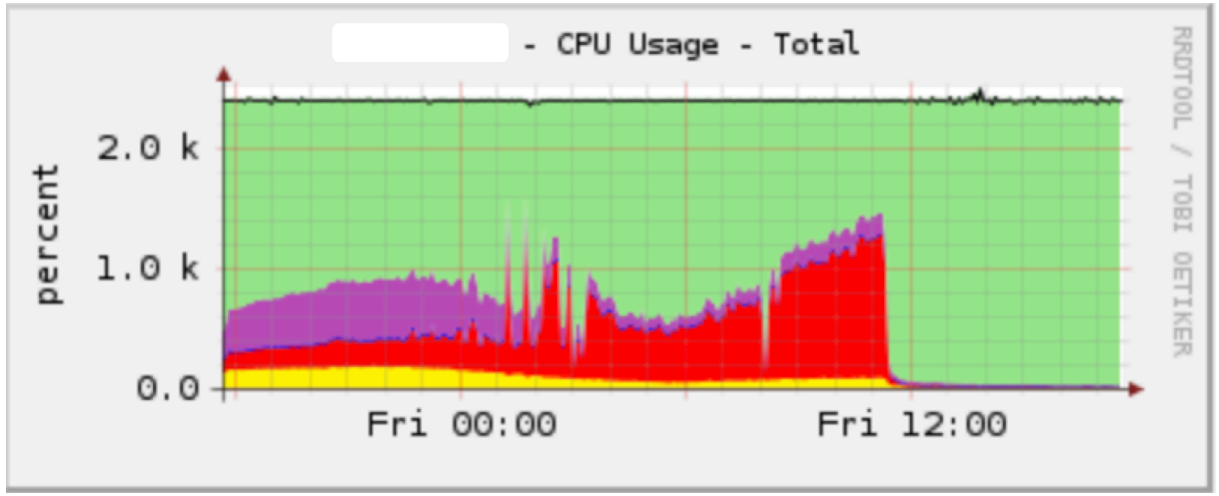
- Желтый — приложение (user)
- Красный — ядро (system)
- Фиолетовый — прерывания (softirq)
- Синий — диск (iowait)
- Зеленый — свободное (idle)



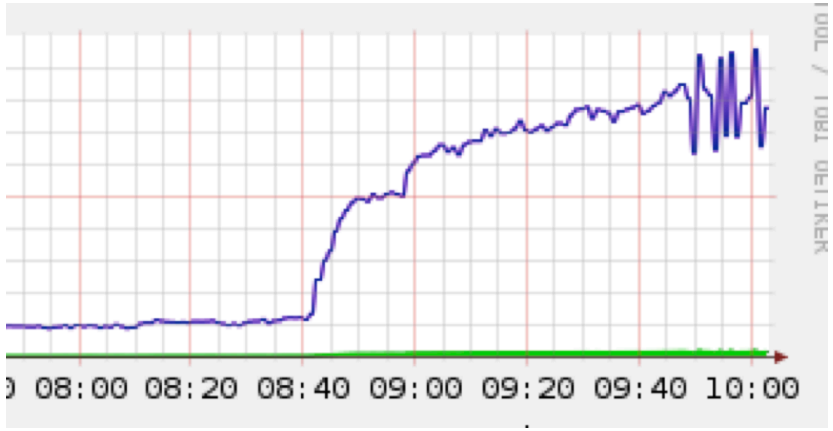
- Зеленый — к серверу (in)
- Синий — от сервера (out)

Ожидание и реальность

- Что-то пошло не так...



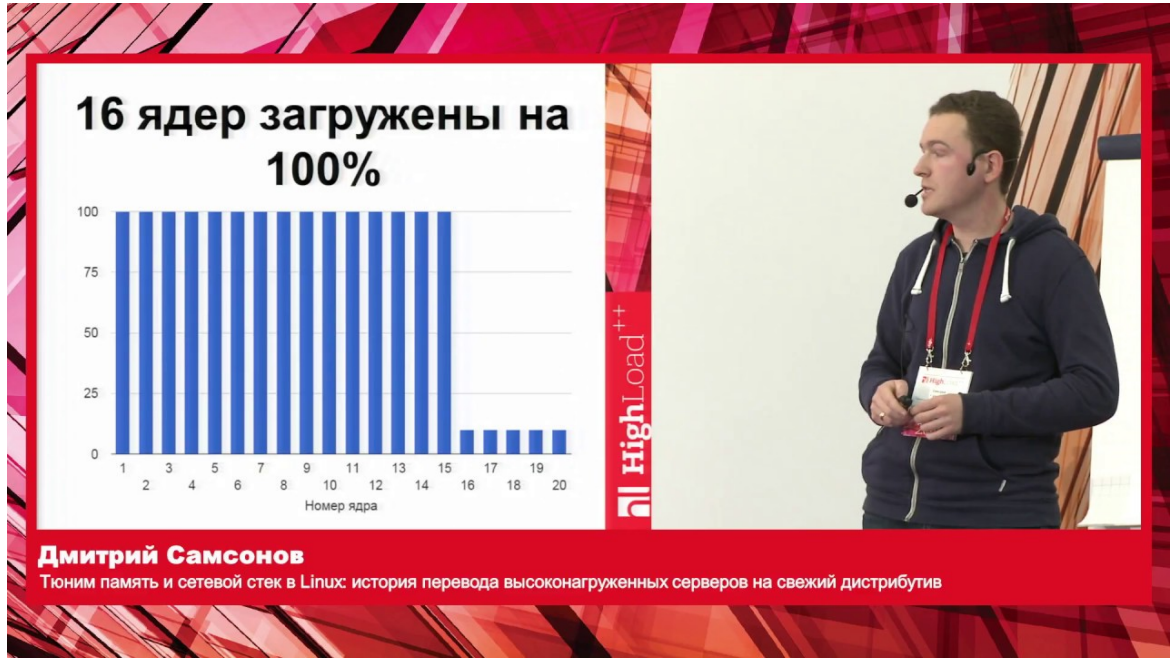
- Желтый — приложение (user)
- Красный — ядро (system)
- Фиолетовый — прерывания (softirq)
- Синий — диск (iowait)
- Зеленый — свободное (idle)



- Зеленый — к серверу (in)
- Синий — от сервера (out)

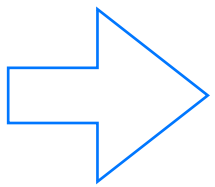
Первые проблемы

- Чиним силами админов:
  - оптимизируем фрагментацию памяти
  - включаем numa interleave
  - перераспределяем прерывания (RPS/RFS/RSS)
  - добавляем affinity
- Тюним память и сетевой стек
  https://youtu.be/iyTiMVTUhNM



Обновляем конфиг

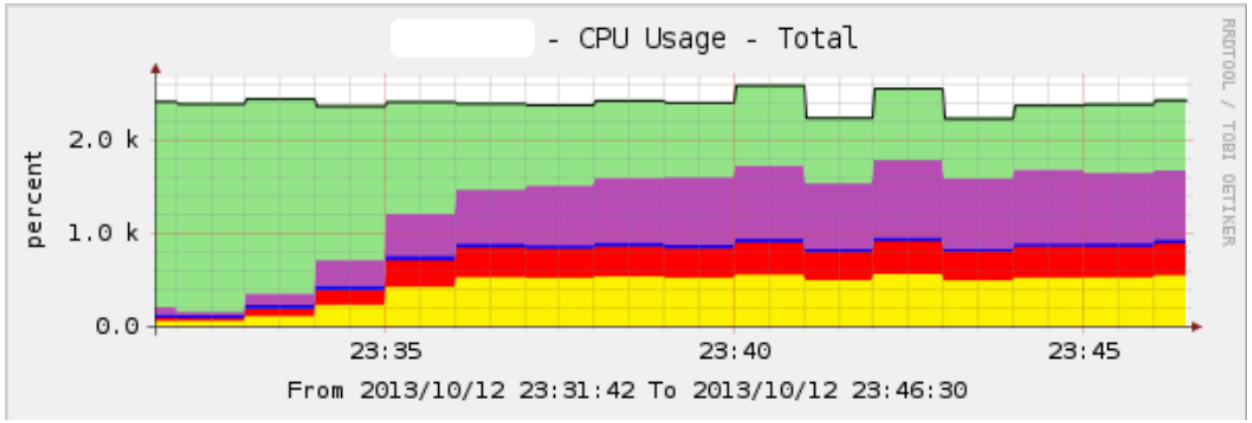
```
selectors: 16

acceptors:
 - port: 80
```



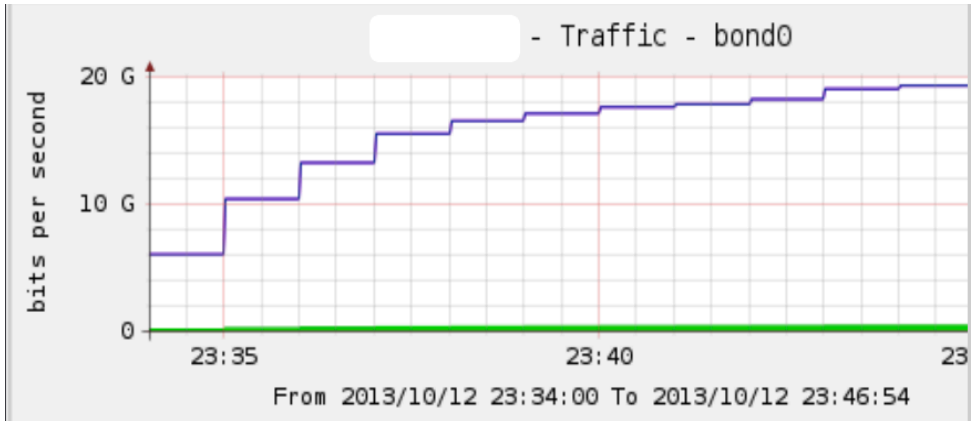
```
selectors: 24
affinity: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 256k
```

Результаты

- Держим до 20 Gbps



- Желтый — приложение (user)
- Красный — ядро (system)
- Фиолетовый — прерывания (softirq)
- Синий — диск (iowait)
- Зеленый — свободное (idle)



- Зеленый — к серверу (in)
- Синий — от сервера (out)

Переезд на https

2015 — 2018 гг

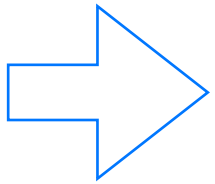
1. База и запуск
2. Внедряем HTTPS
3. Прокачиваем до 100G
4. Последние достижения

Интеграция в one-nio

- Используем OpenSSL через JNI
- Пишем слой совместимости для разных версий
  - 1.0.1 на Suse
  - 1.1.0 на CentOS
  - 1.1.1 собственной сборки

Новый пайплайн обработки

- accept()
- recv()
- send()/sendfile()
- close()



accept()

SSL_new()

SSL_set_fd()

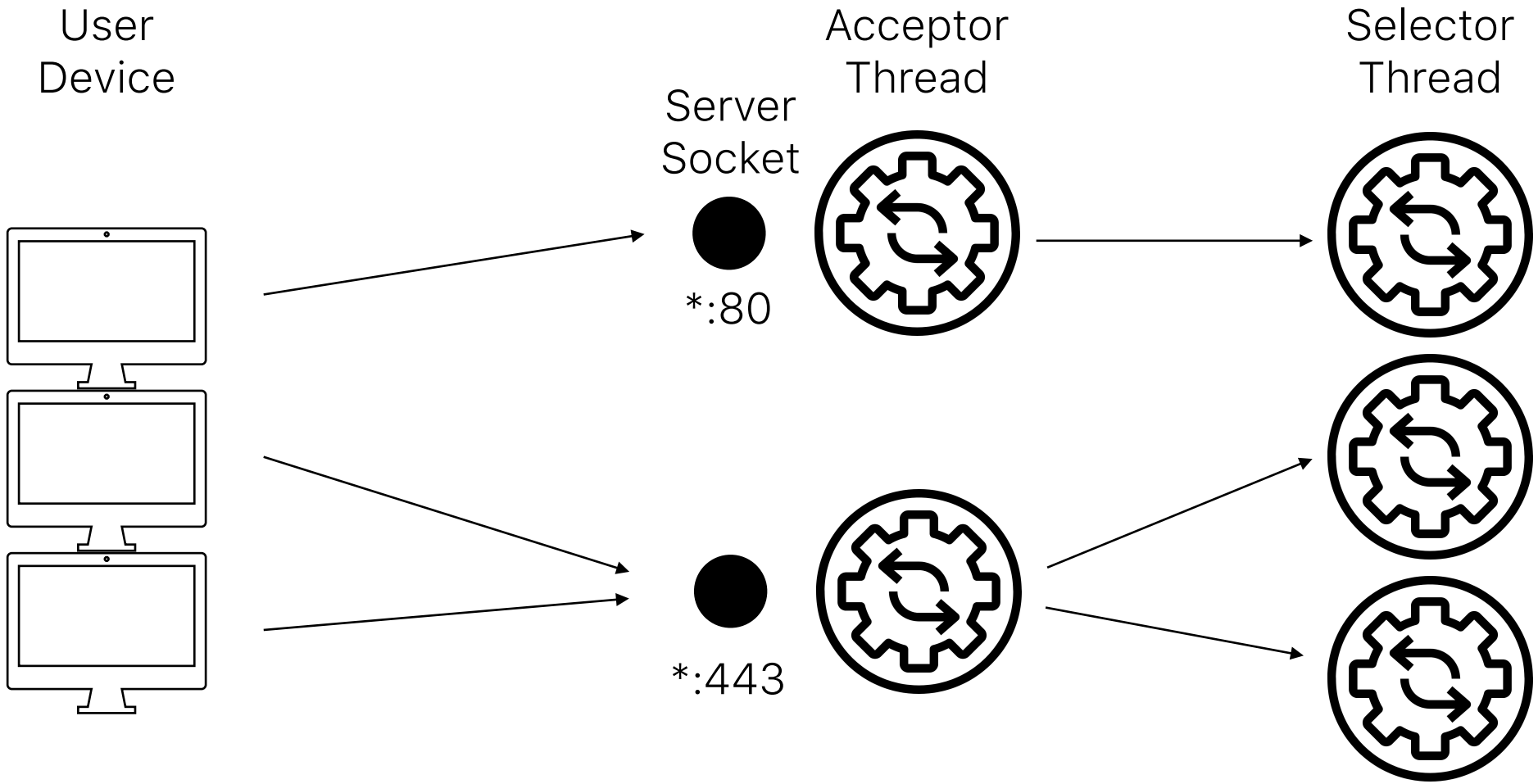
SSL_read()

SSL_write()

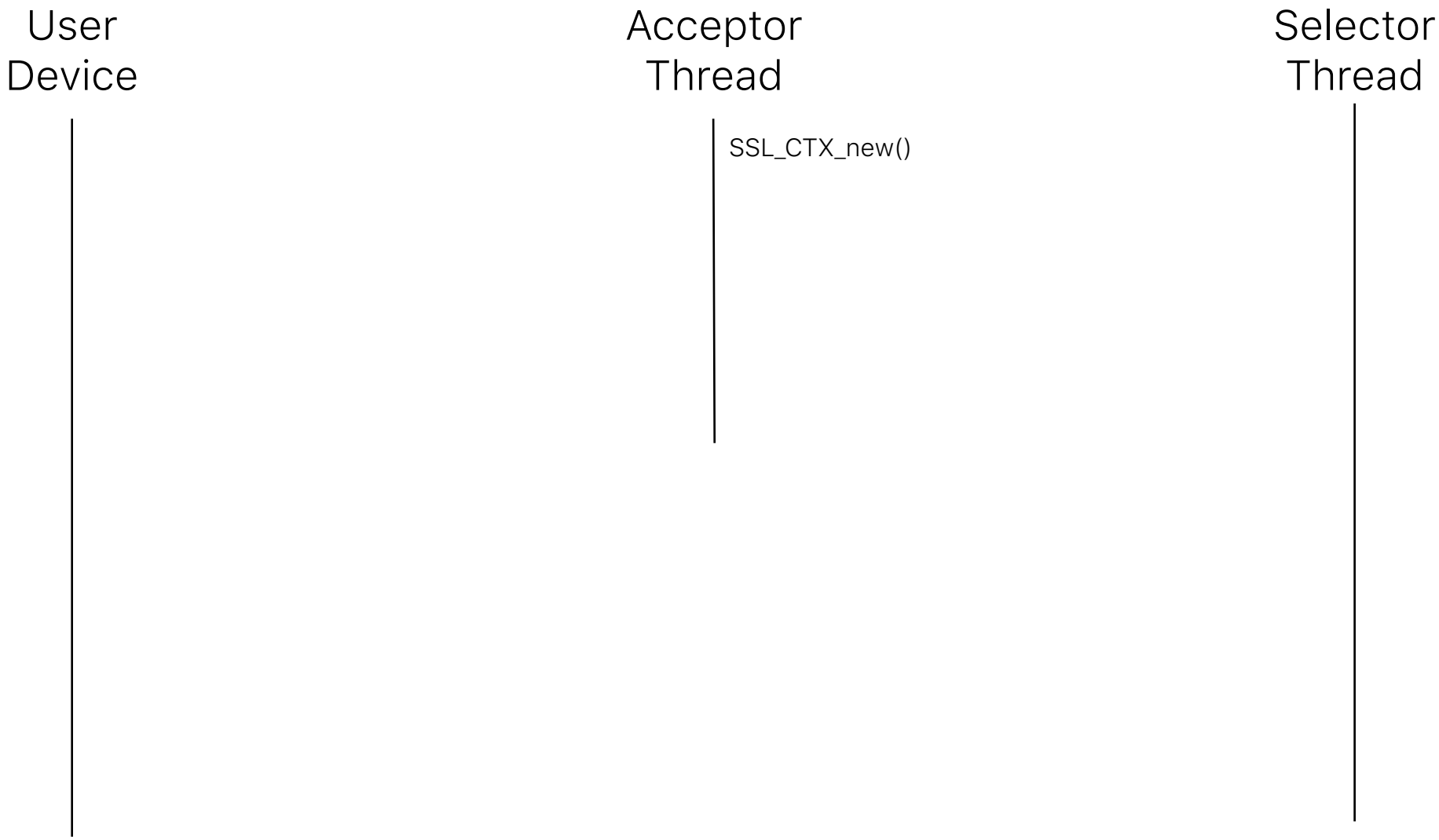
SSL_shutdown()

close()

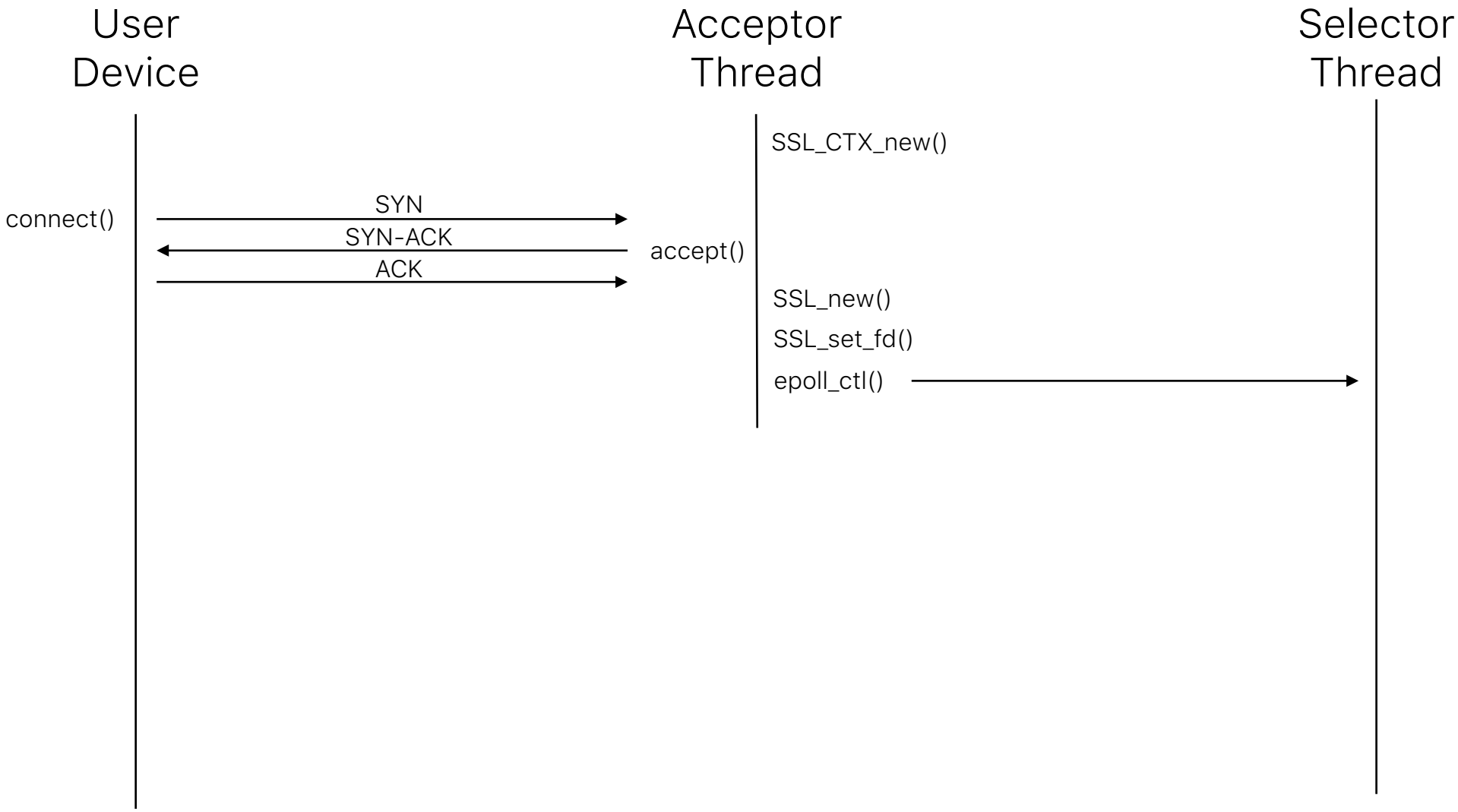
Открываем новый порт



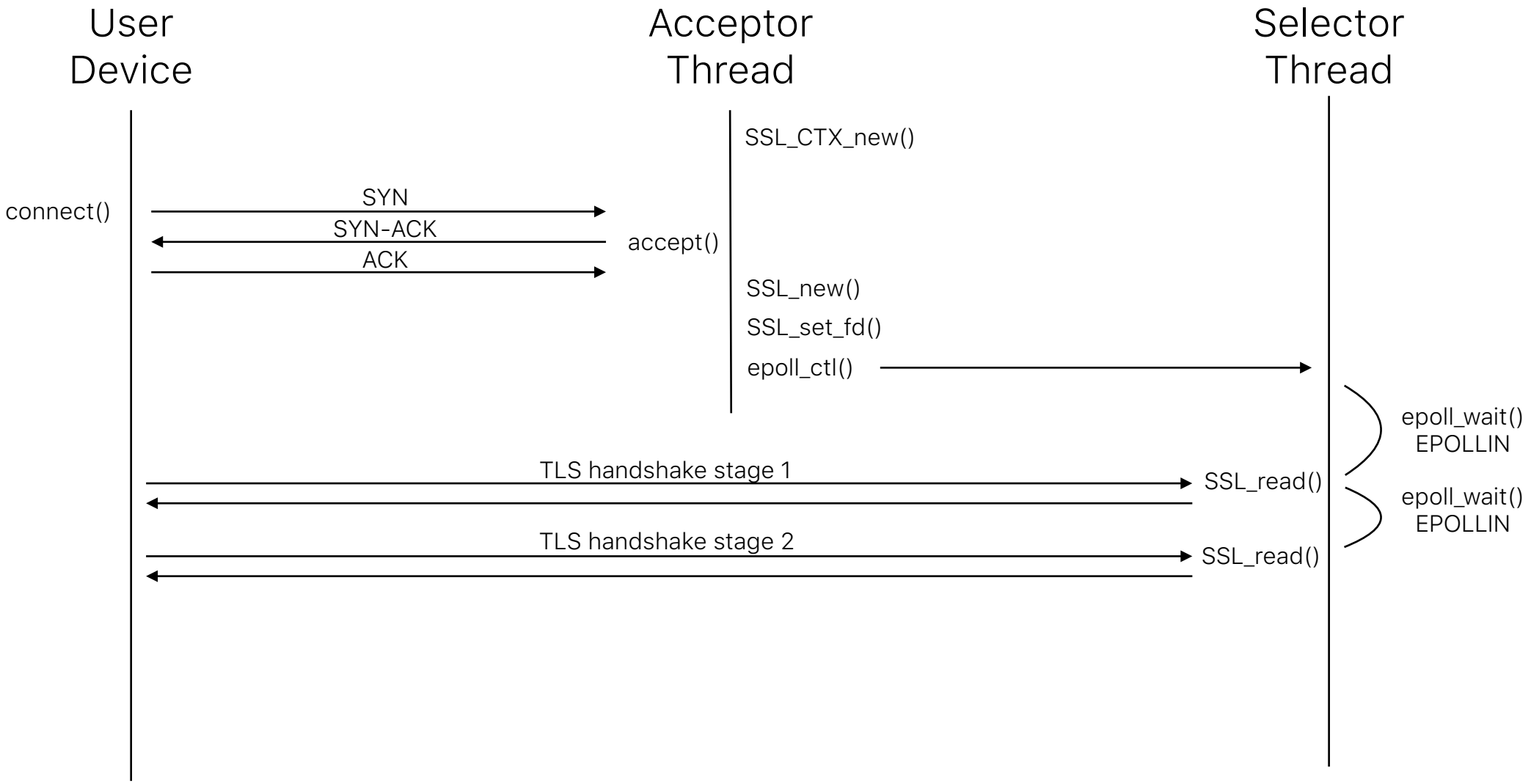
Новый пайплайн обработки



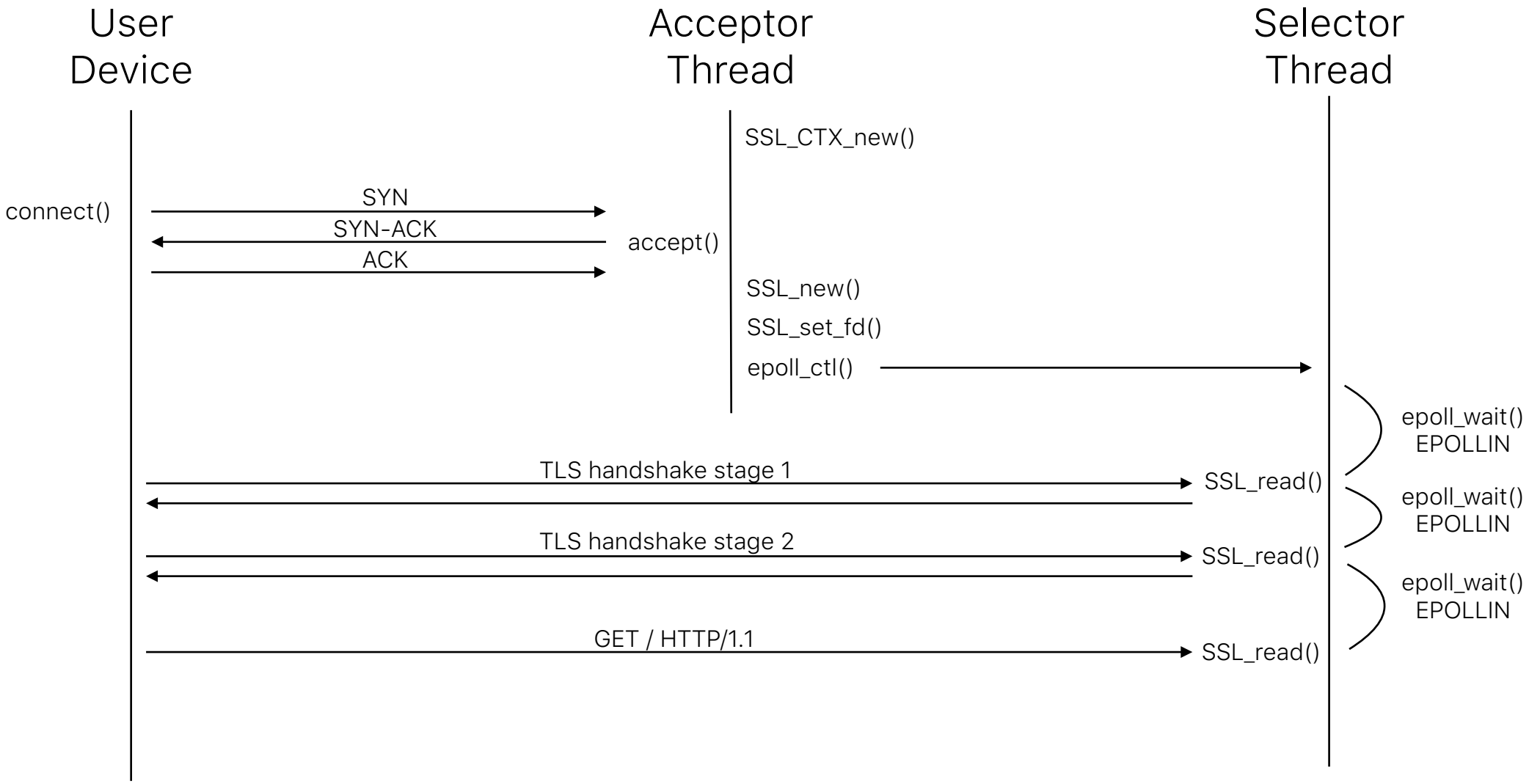
Новый пайплайн обработки



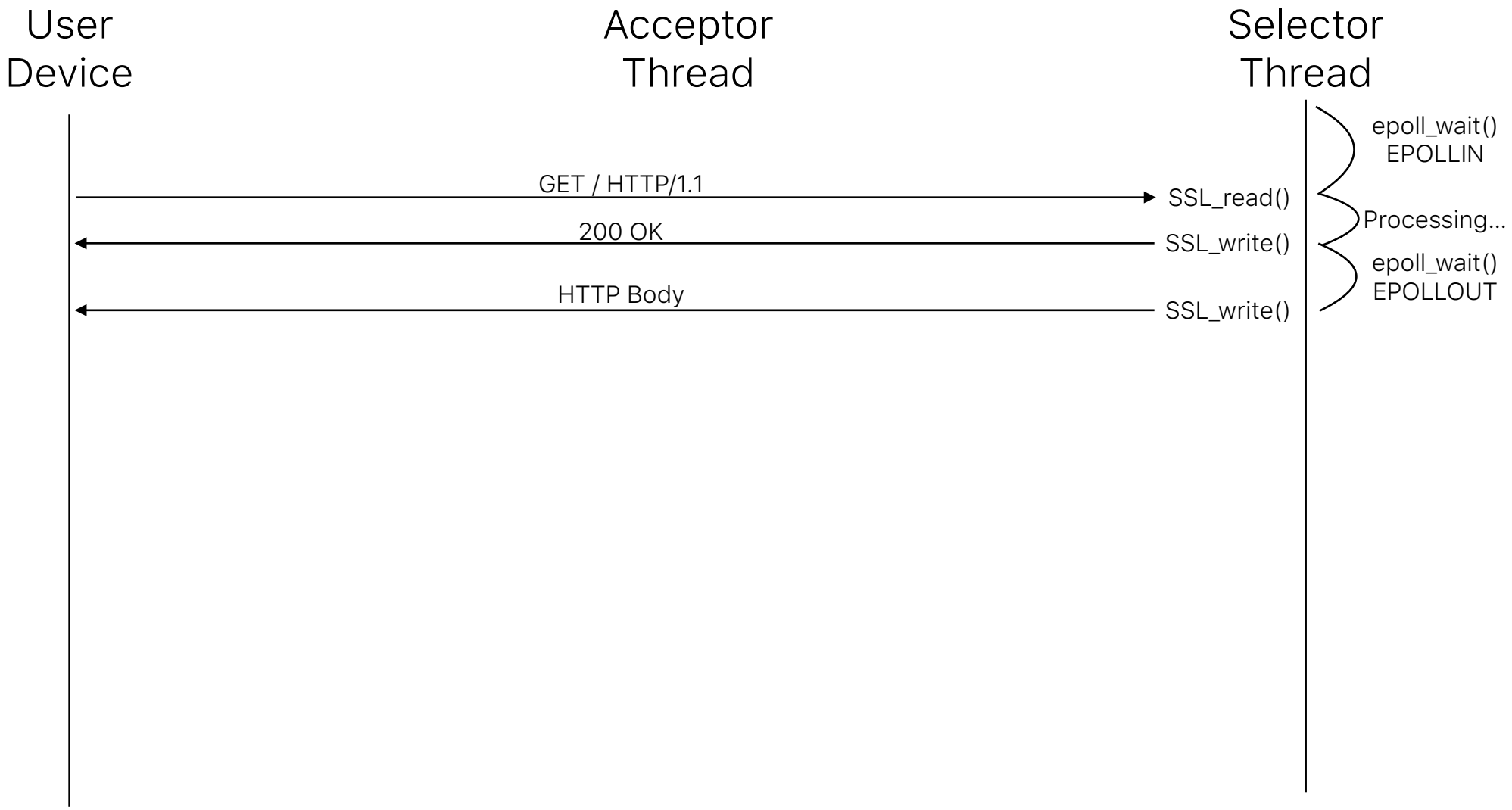
Новый пайплайн обработки



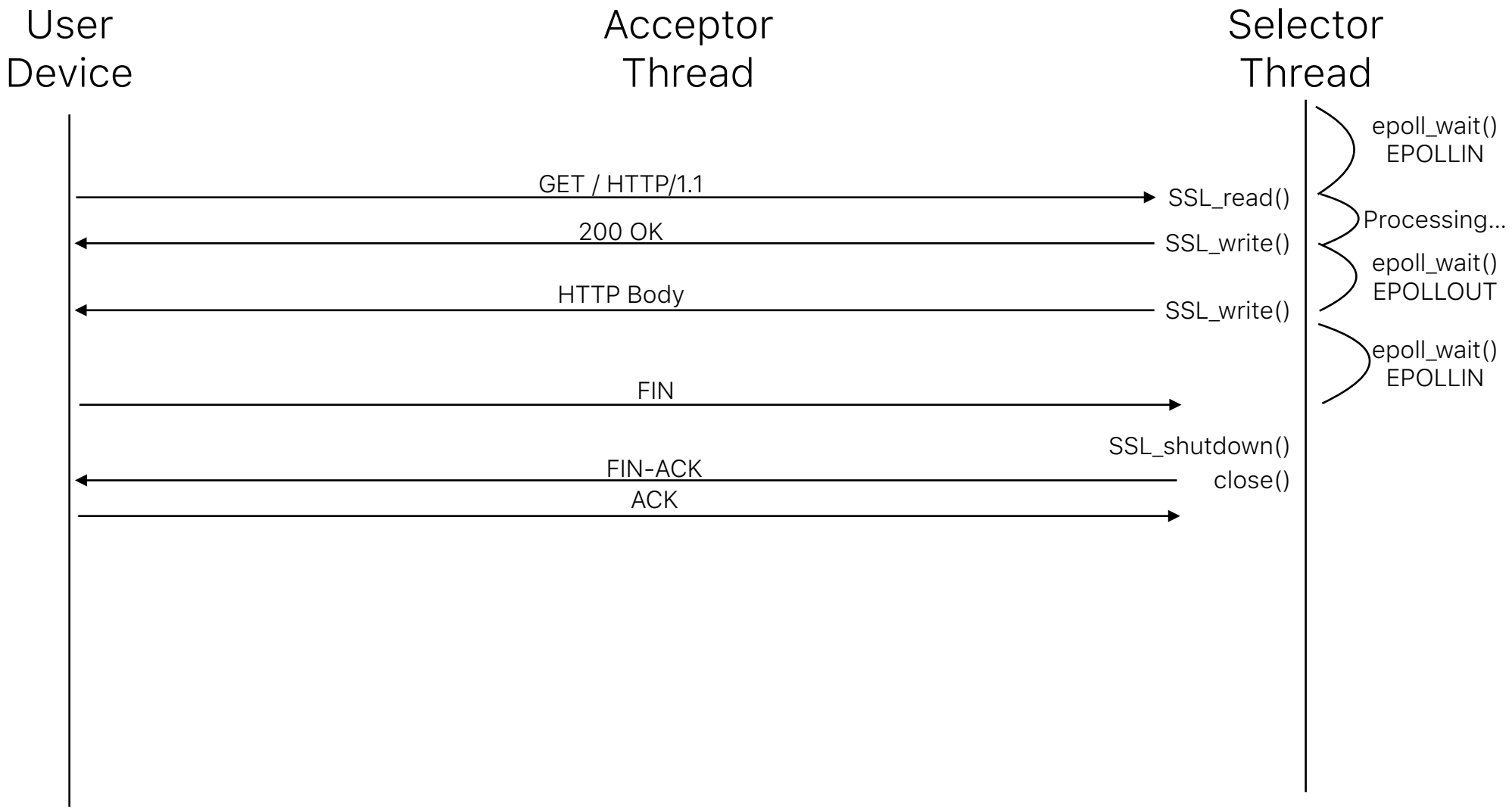
Новый пайплайн обработки



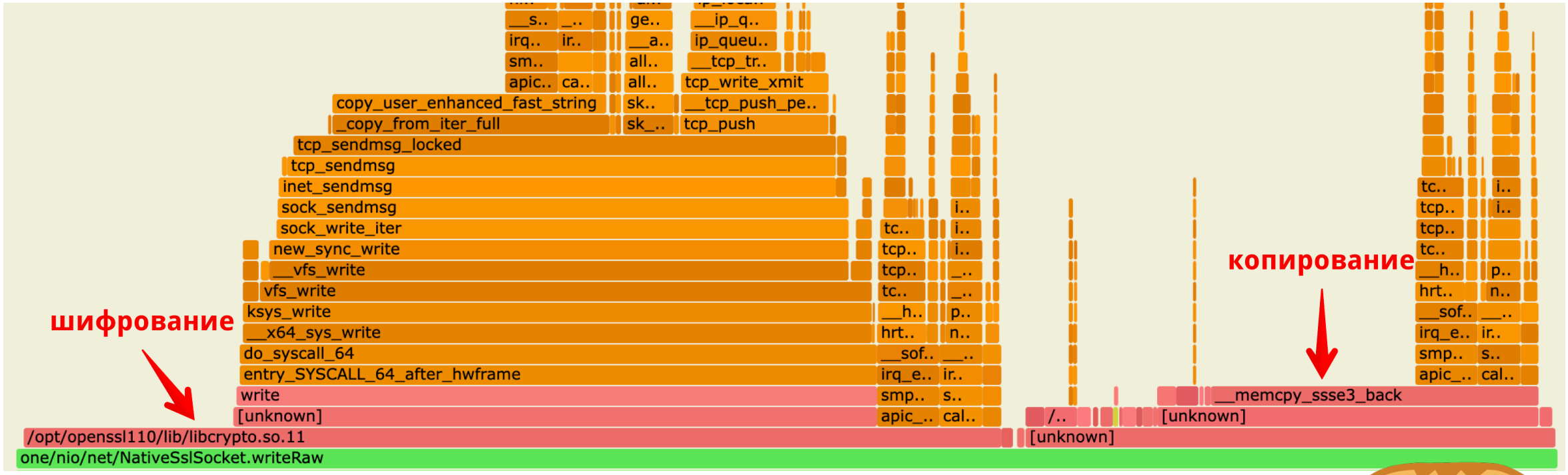
Новый пайплайн обработки



Новый пайплайн обработки



Новый стек вызовов



- Потребление cpu x2
- sendfile невозможен

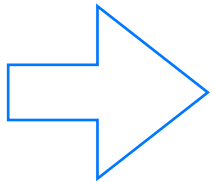
Новые испытания на https

2015 — 2018 гг

1. База и запуск
2. Внедряем HTTPS
3. Прокачиваем до 100G
4. Последние достижения

Конфигурация серверов

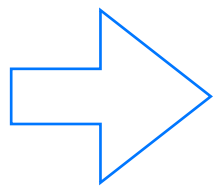
- Xeon 2620 (6C/12T)
- 128 GB RAM
- 20 Gbps NIC
- HDD
- ядро 2.6



- Xeon 2660v2 (10C/20T)
- 256 GB RAM
- 40 Gbps NIC
- HDD
- ядро 2.6

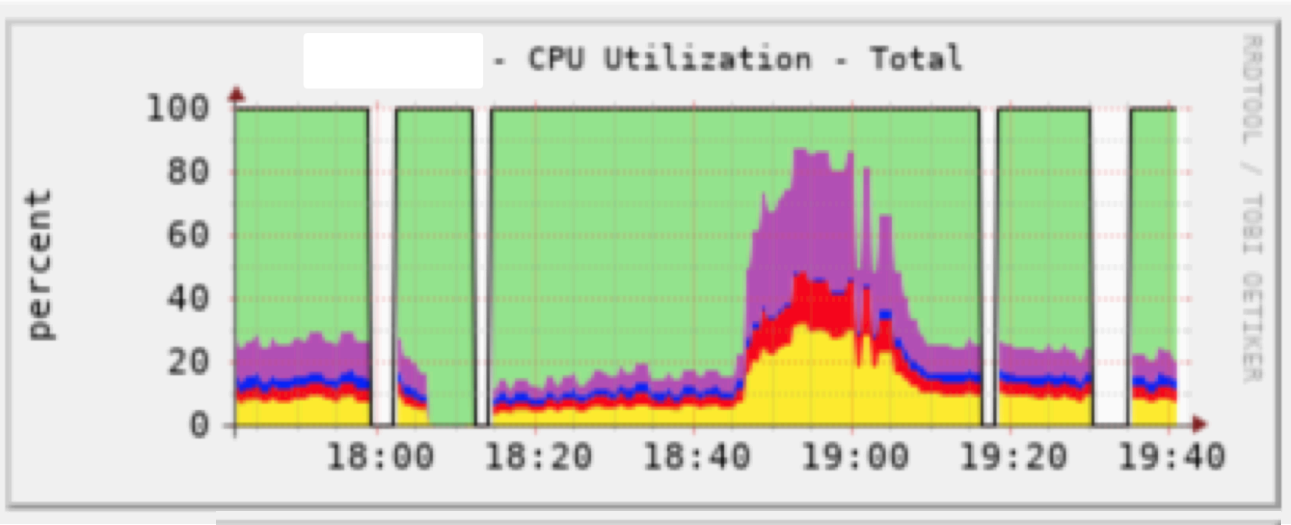
Обновляем конфиг

```
selectors: 24
affinity: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 256k
```

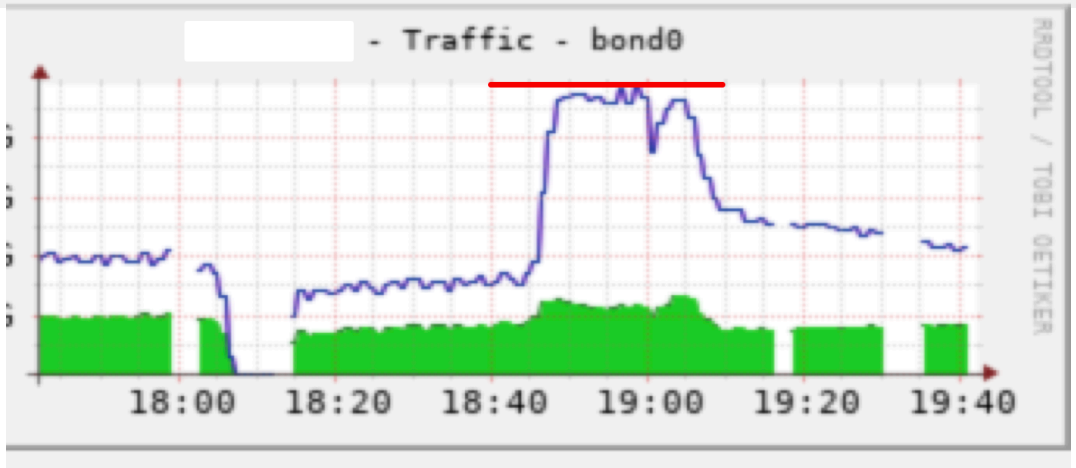


```
selectors: 40
affinity: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 256k
 - port: 443
   recvBuf: 64k
   sendBuf: 256k
   ssl:
    - protocols: TLSv1 TLSv1.1 TLSv1.2
      certFile: ./file.cert
      privateKeyFile: ./file.key
```

Проблемы под нагрузкой



- Желтый — приложение (user)
- Красный — ядро (system)
- Фиолетовый — прерывания (softirq)
- Синий — диск (iowait)
- Зеленый — свободное (idle)



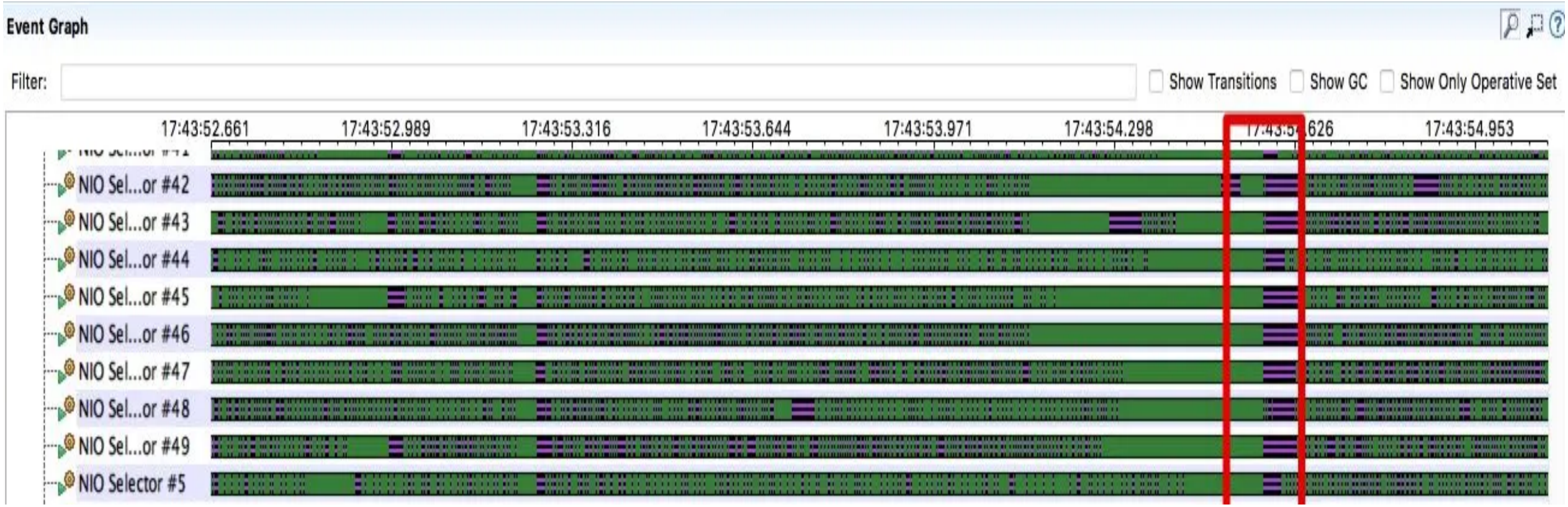
- Зеленый — к серверу (in)
- Синий — от сервера (out)

Подробнее про async-profiler

- https://youtu.be/WoomehXd4NY

Треды под микроскопом

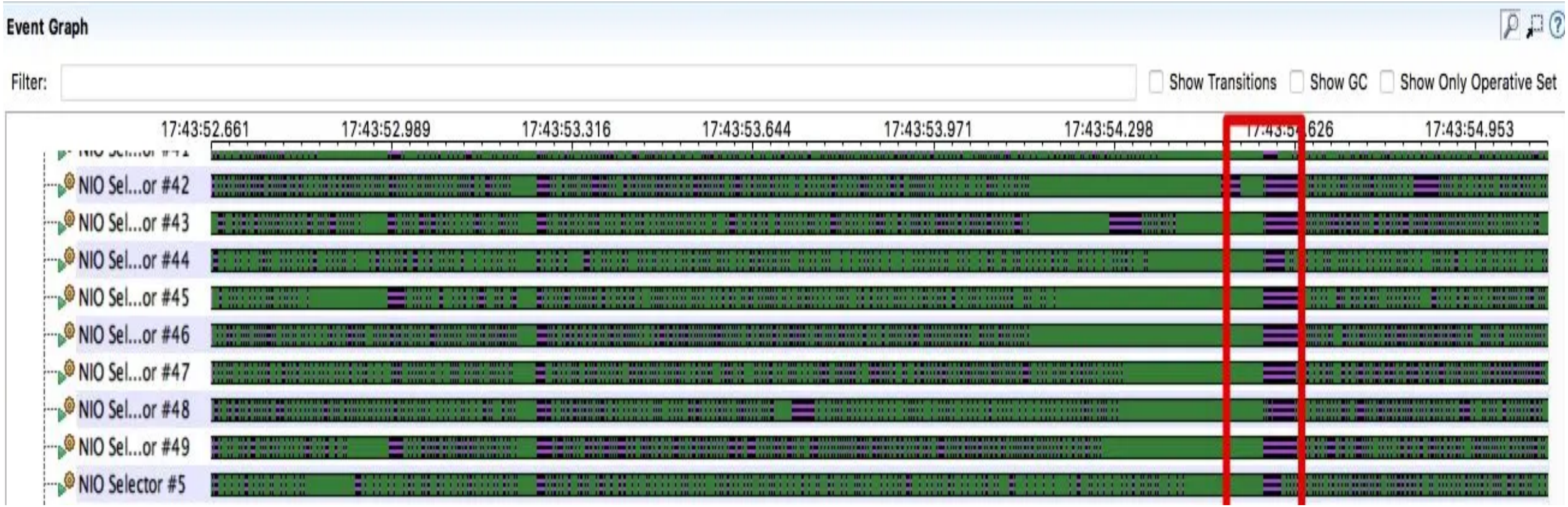
- profiler -e cpu -i 999us -o jfr -f profile.jfr



• Зеленый - тред ждет (спит) • Фиолетовый - работает

Треды под микроскопом

- profiler -e cpu -i 999us -o jfr -f profile.jfr



• Зеленый - тред ждет (спит) • Фиолетовый - работает

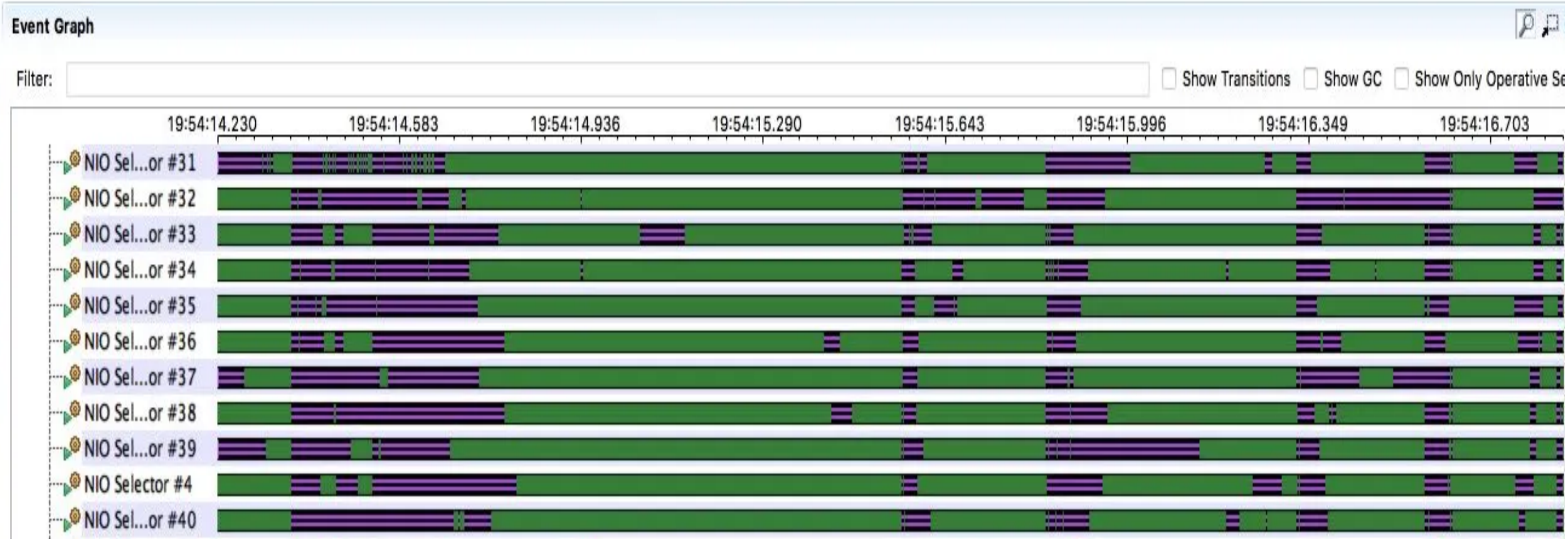
Треды под микроскопом

- profiler -e cpu -i 999us -o jfr -f profile.jfr

• Зеленый - тред ждет (спит) • Фиолетовый - работает

Треды под микроскопом

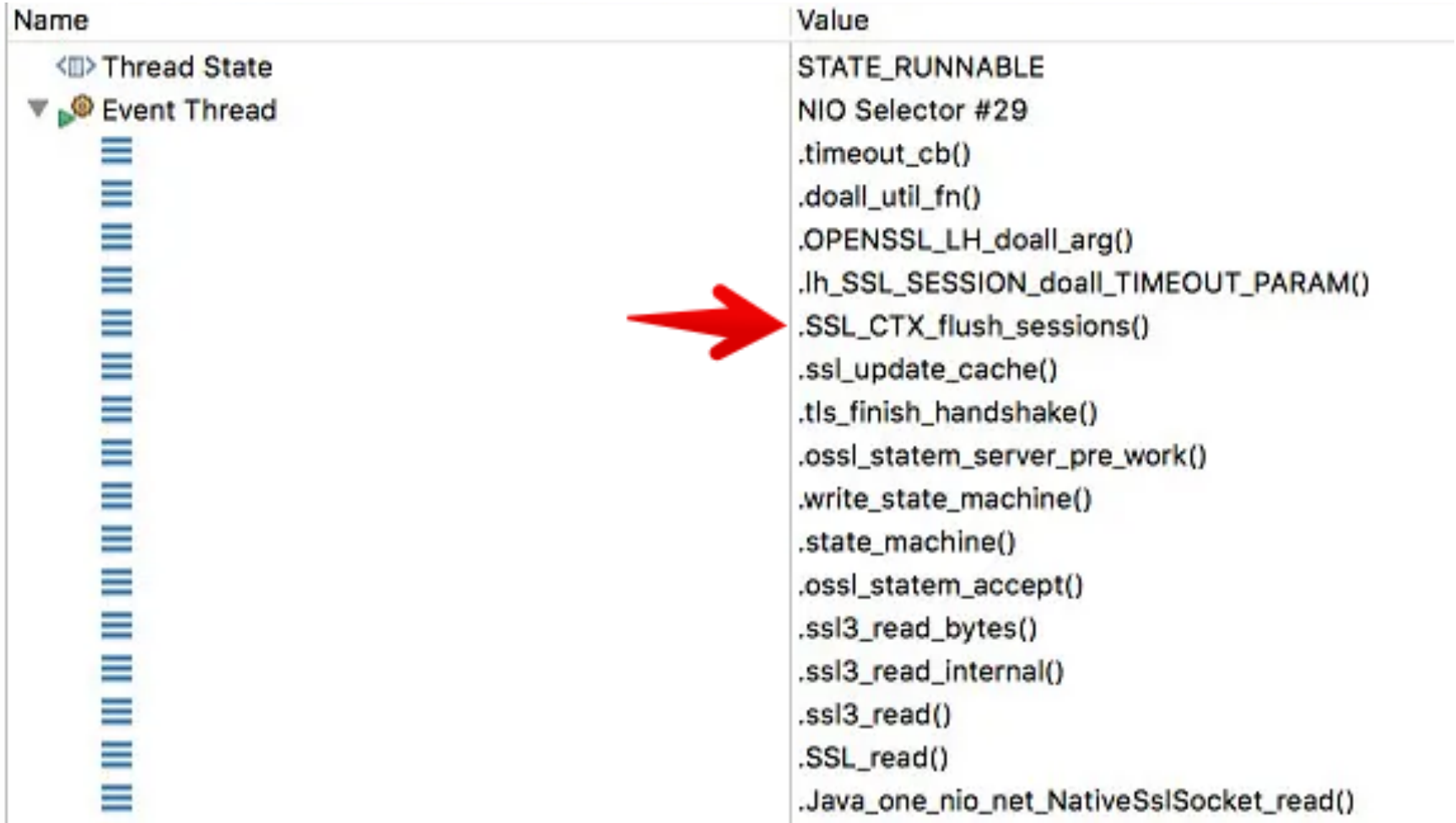
- profiler -e cpu -i 999us -o jfr -f profile.jfr



• Зеленый - тред ждет (спит) • Фиолетовый - работает

Блокировка на кеше ssl сессий

- profiler -e cpu -i 999us -o jfr -f profile.jfr



Блокировка на кеше ssl сессий

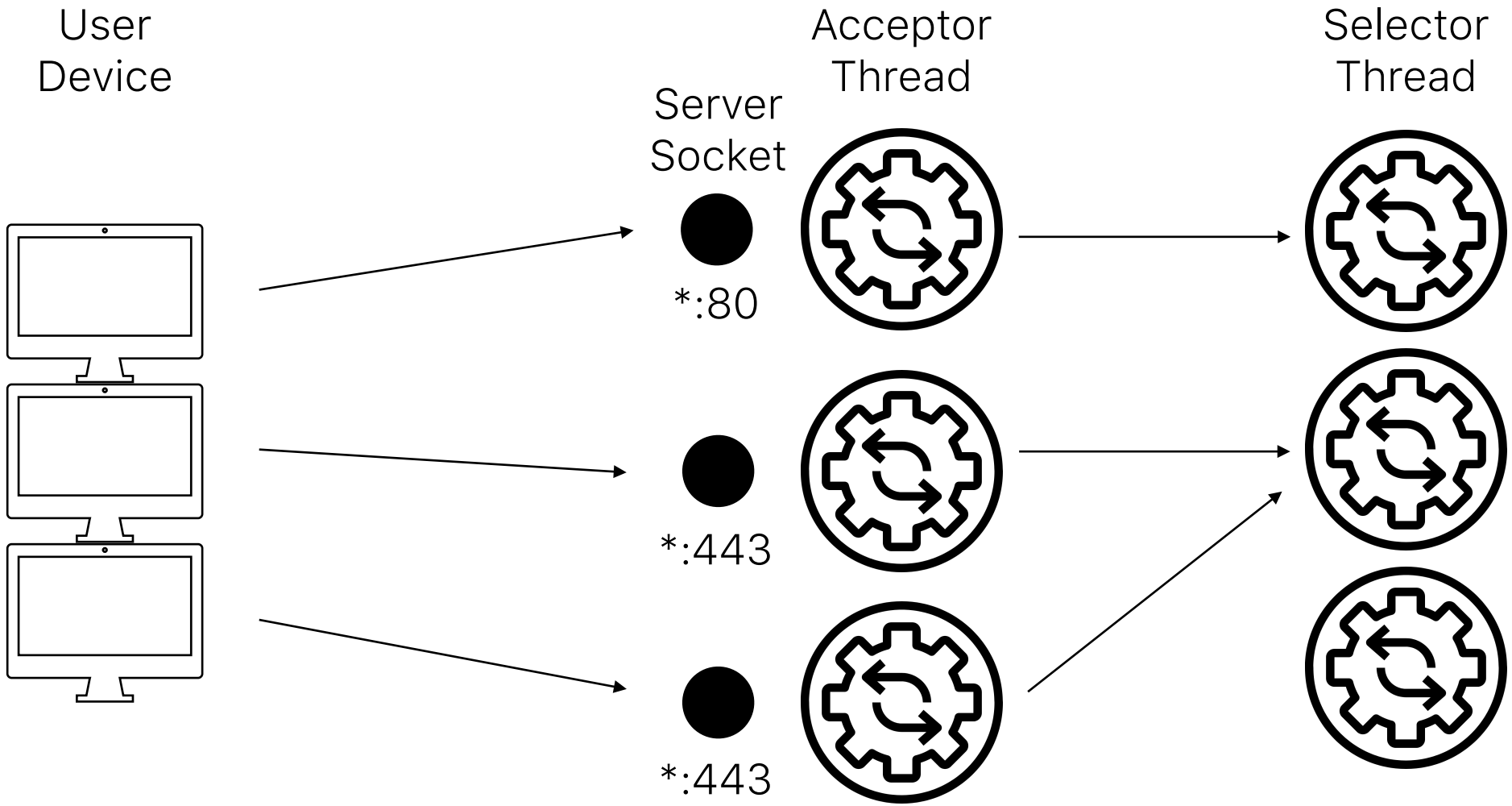
- https://github.com/openssl/openssl/blob/OpenSSL_1_1_0-stable/ssl/ssl_lib.c#L3072

openssl / ssl / **ssl_lib.c**

Code | Blame | 4485 lines (3876 loc) · 124 KB

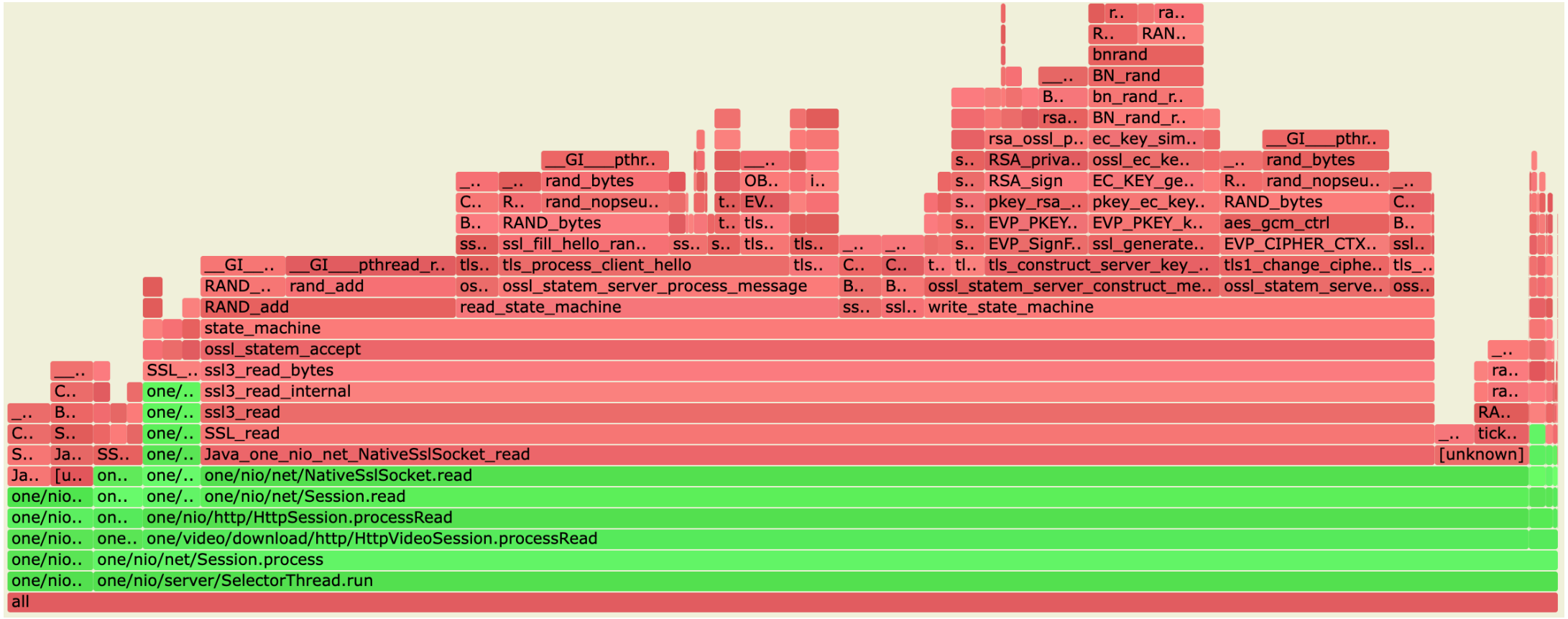
```
3067        /* auto flush every 255 connections */
3068        if ((!(i & SSL_SESS_CACHE_NO_AUTO_CLEAR)) && ((i & mode) == mode)) {
3069            if ((((mode & SSL_SESS_CACHE_CLIENT)
3070                  ? s->session_ctx->stats.sess_connect_good
3071                  : s->session_ctx->stats.sess_accept_good) & 0xff) == 0xff) {
3072                SSL_CTX_flush_sessions(s->session_ctx, (unsigned long)time(NULL));
3073            }
3074        }
3075    }
```

Нужно больше контекстов



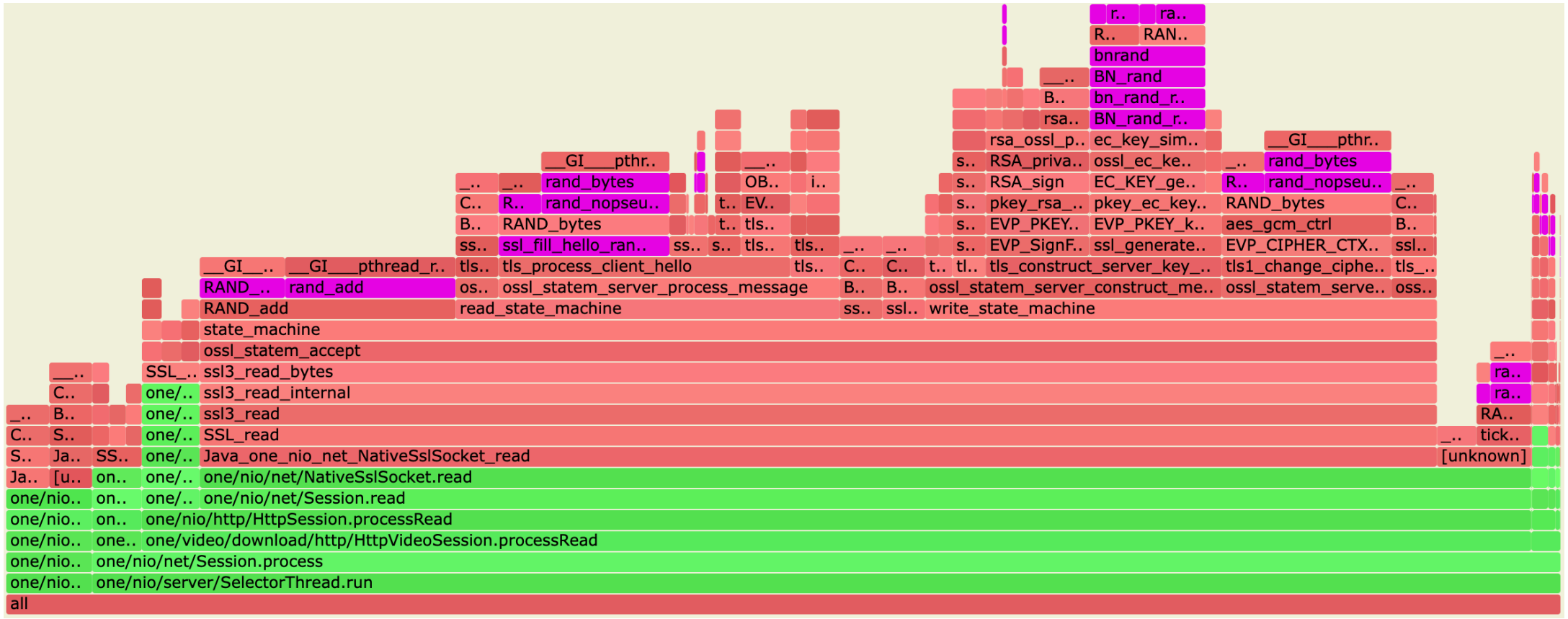
Ищем блокировки

- profiler -e pthread_rwlock_unlock -f profile.html



Блокировка на random

- profiler -e pthread_rwlock_unlock -f profile.html



Блокировка на random

- https://github.com/openssl/openssl/blob/OpenSSL_1_1_0-stable/crypto/rand/md_rand.c

openssl / crypto / rand / **md_rand.c**

Code | Blame | 665 lines (573 loc) · 18.5 KB

```
348
349         CRYPTO_THREAD_write_lock(rand_lock);
350         /*
351          * We could end up in an async engine while holding this lock so ensure
352          * we don't pause and cause a deadlock
353          */
354         ASYNC_block_pause();
355
356         /* prevent rand_bytes() from trying to obtain the lock again */
357         CRYPTO_THREAD_write_lock(rand_tmp_lock);
358         locking_threadid = CRYPTO_THREAD_get_current_id();
359         CRYPTO_THREAD_unlock(rand_tmp_lock);
360         crypto_lock_rand = 1;
361
362         if (!initialized) {
363             RAND_poll();
364             initialized = (entropy >= ENTROPY_NEEDED);
365         }
```

Блокировка на random

- https://github.com/openssl/openssl/blob/OpenSSL_1_1_0-stable/crypto/rand/rand_lib.c

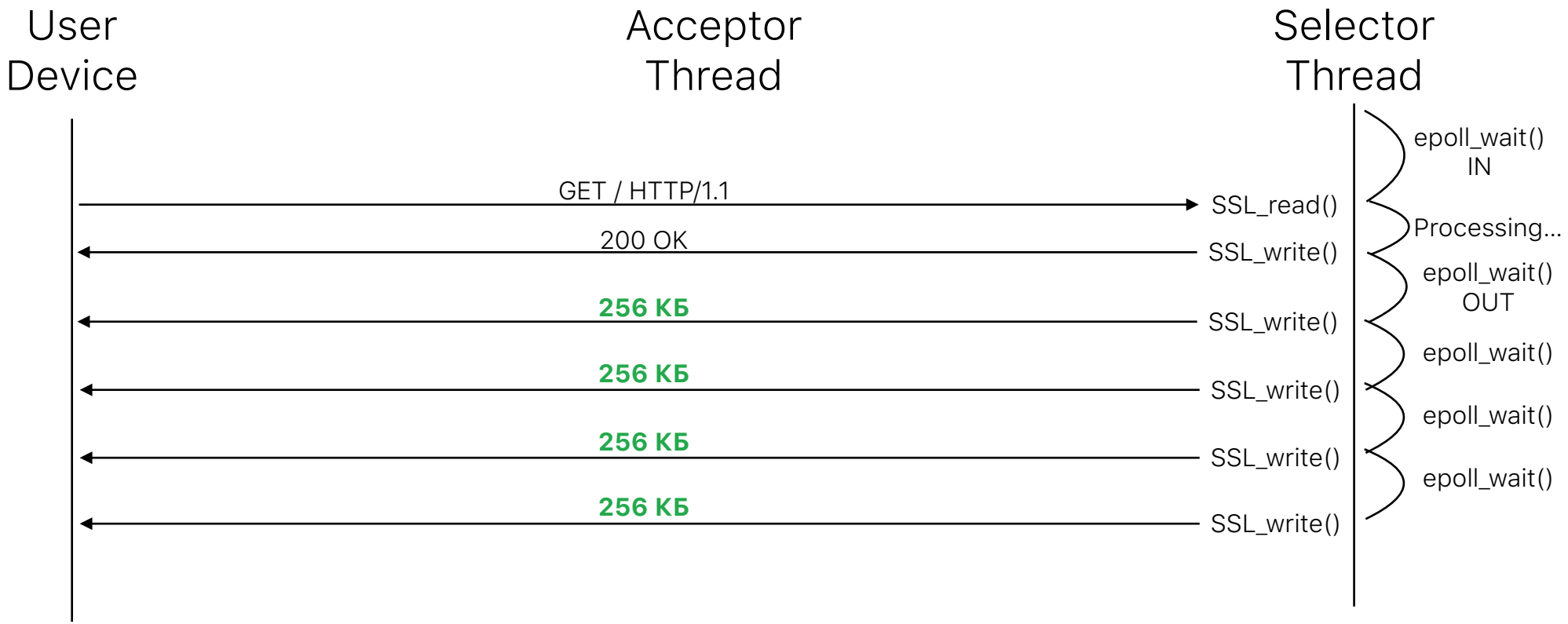
```
59  const RAND_METHOD *RAND_get_rand_method(void)
60  {
61      const RAND_METHOD *tmp_meth = NULL;
62
63      if (!RUN_ONCE(&rand_lock_init, do_rand_lock_init))
64          return NULL;
65
66      CRYPTO_THREAD_write_lock(rand_meth_lock);
67      if (!default_RAND_meth) {
68  #ifndef OPENSSL_NO_ENGINE
69          ENGINE *e = ENGINE_get_default_RAND();
70          if (e) {
71              default_RAND_meth = ENGINE_get_RAND(e);
72              if (default_RAND_meth == NULL) {
73                  ENGINE_finish(e);
74                  e = NULL;
75              }
76          }
77          if (e)
78              funct_ref = e;
79          else
80  #endif
81              default_RAND_meth = RAND_OpenSSL();
82      }
```

Используем RDRAND

- https://github.com/openssl/openssl/blob/OpenSSL_1_1_0-stable/crypto/engine/eng_rdrand.c

```
25  static int get_random_bytes(unsigned char *buf, int num)
26  {
27      size_t rnd;
28
29      while (num >= (int)sizeof(size_t)) {
30          if ((rnd = OPENSSL_ia32_rdrand()) == 0)
31              return 0;
32
33          *((size_t *)buf) = rnd;
34          buf += sizeof(size_t);
35          num -= sizeof(size_t);
36      }
37      if (num) {
38          if ((rnd = OPENSSL_ia32_rdrand()) == 0)
39              return 0;
40
41          memcpy(buf, &rnd, num);
42      }
43
44      return 1;
45  }
```

Рост числа epoll_wait/write



Рост числа epoll_wait/write

SSL ограничения на запись

- Опция SSL_MODE_ENABLE_PARTIAL_WRITE
  https://docs.openssl.org/1.1.1/man3/SSL_CTX_set_mode/

  - SSL_MODE_ENABLE_PARTIAL_WRITE

    Allow SSL_write_ex(..., n, &r) to return with 0 < r < n (i.e. report success when just a single record has been written). This works in a similar way for SSL_write(). When not set (the default), SSL_write_ex() or SSL_write() will only report success once the complete chunk was written. Once SSL_write_ex() or SSL_write() returns successful, **r** bytes have been written and the next call to SSL_write_ex() or SSL_write() must only send the n-r bytes left, imitating the behaviour of write().

SSL ограничения на запись

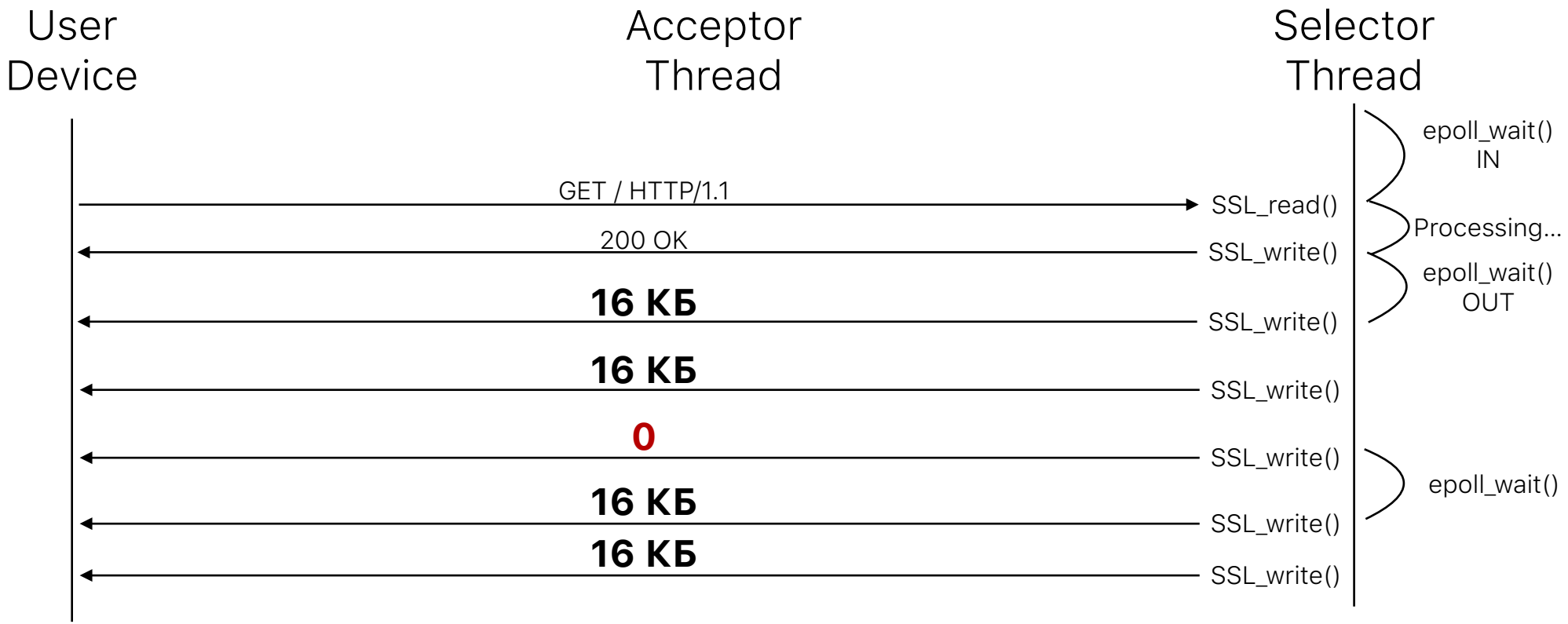
- Документация SSL_write
  https://docs.openssl.org/1.1.1/man3/SSL_write/#notes

> SSL_MODE_ENABLE_PARTIAL_WRITE option of SSL_CTX_set_mode(3). When this flag is set the write functions will also return with success when a partial write has been successfully completed. In this case the write function operation is considered completed. The bytes are sent and a new write call with a new buffer (with the already sent bytes removed) must be started. A partial write is performed with the size of a message block, which is 16kB.

- Спецификация SSL/TLS
  https://github.com/openssl/openssl/blob/OpenSSL_1_1_0-stable/include/openssl/ssl3.h#L163

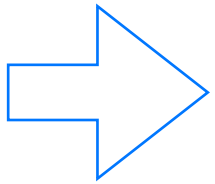
```
163     /* Maximum plaintext length: defined by SSL/TLS standards */
164     # define SSL3_RT_MAX_PLAIN_LENGTH                16384
```

Переделываем использование SSL_write



Обновляем конфиг

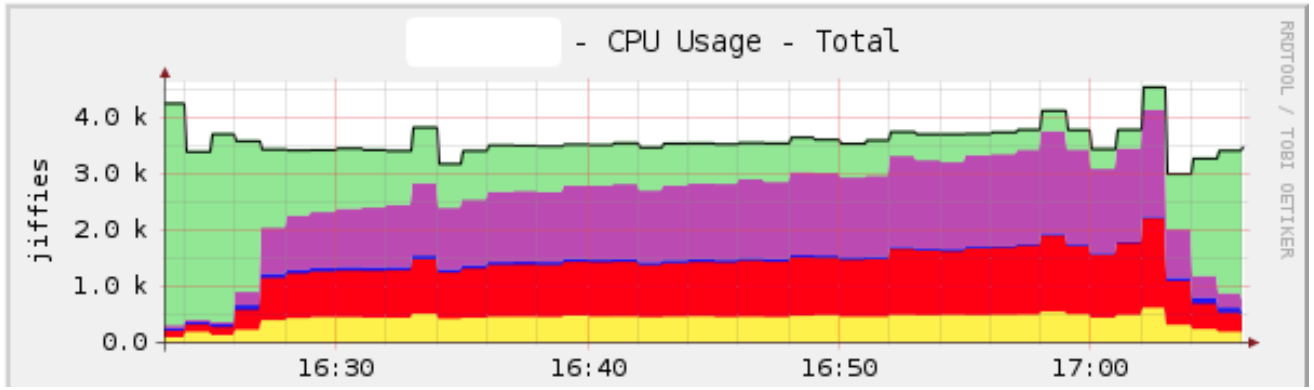
```
selectors: 40
affinity: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 256k
 - port: 443
   recvBuf: 64k
   sendBuf: 256k
   ssl:
    - protocols: TLSv1 TLSv1.1 TLSv1.2
      certFile: ./file.cert
      privateKeyFile: ./file.key
```



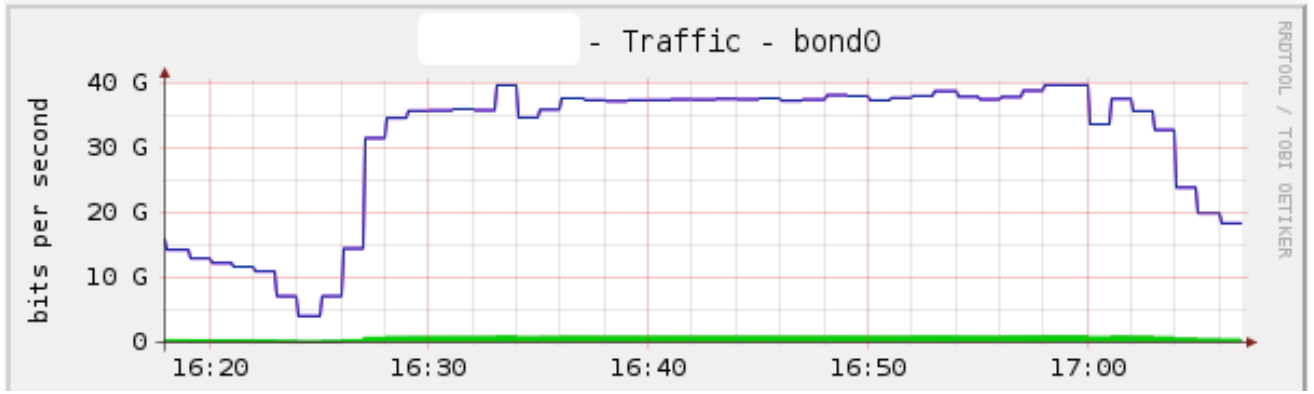
```
selectors: 40
affinity: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 256k
 - port: 443
   recvBuf: 64k
   sendBuf: 256k
   threads: 20
   reusePort: true
   ssl:
    - protocols: TLSv1 TLSv1.1 TLSv1.2
      certFile: ./file.cert
      privateKeyFile: ./file.key
      rdrand: true
```

Результаты

- Держим до ~~20~~ 40 Gbps



- Желтый — приложение (user)
- Красный — ядро (system)
- Фиолетовый — прерывания (softirq)
- Синий — диск (iowait)
- Зеленый — свободное (idle)



- Зеленый — к серверу (in)
- Синий — от сервера (out)

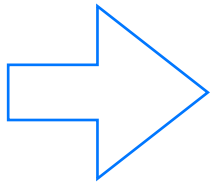
Новые сервера

2018 — 2022 гг

1. База и запуск
2. Внедряем HTTPS
3. Прокачиваем до 100G
4. Последние достижения

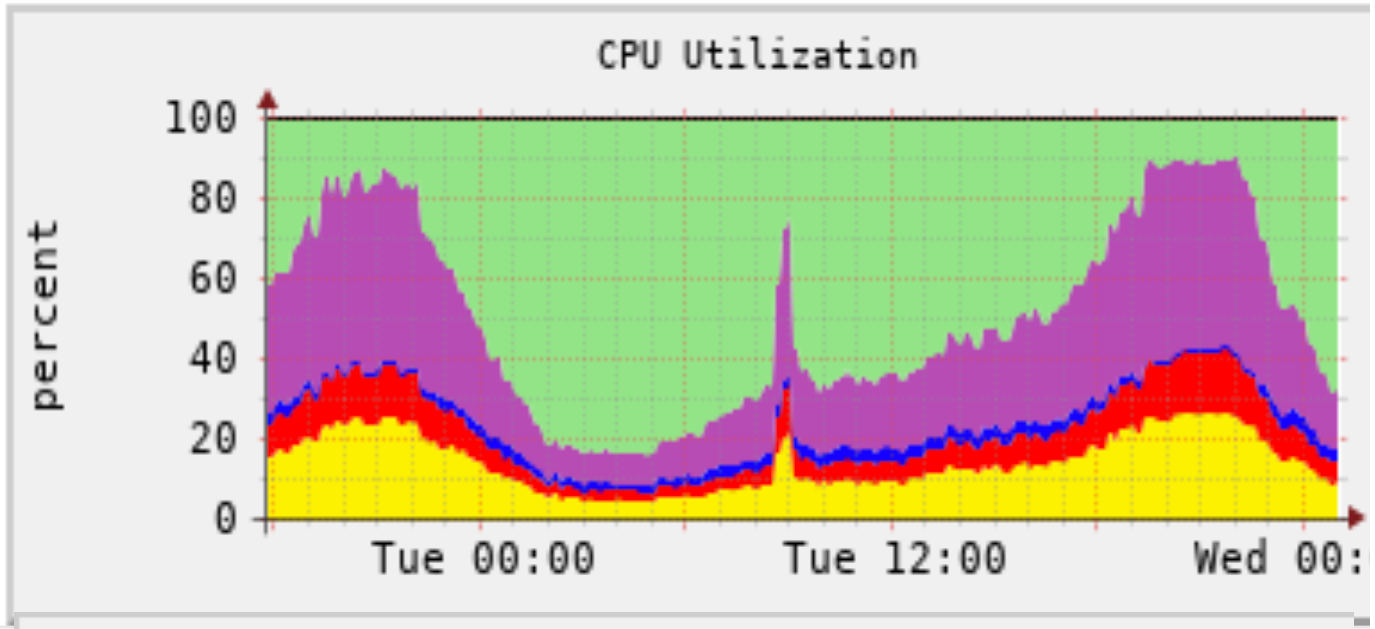
Свежая конфигурация

- Xeon 2660v2 (10C/20T)
- 256 GB RAM
- 40 Gbps NIC
- HDD
- ядро 2.6

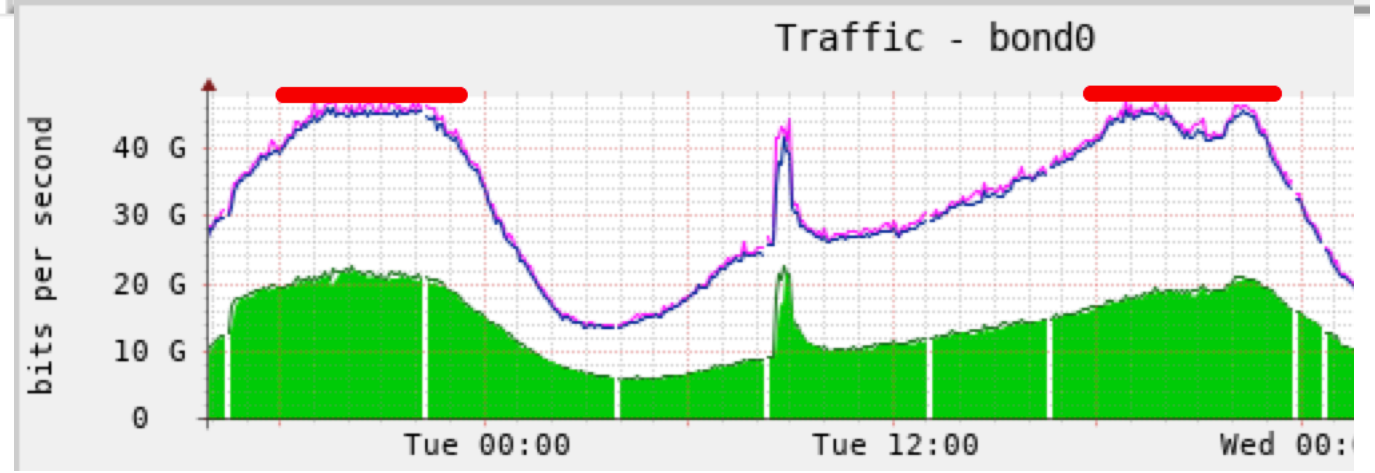


- Xeon Gold 62xx (22C/44T)
- 512 GB RAM
- 100 Gbps NIC
- SSD
- ядро 2.6

Опять не нагружается



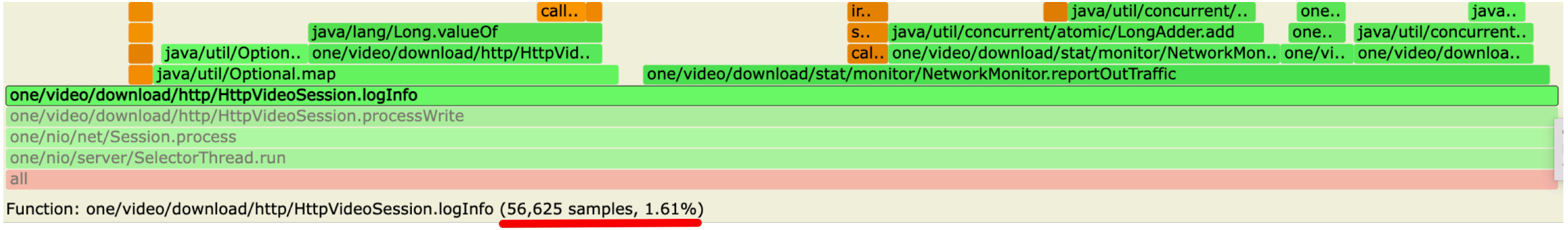
- Желтый — приложение (user)
- Красный — ядро (system)
- Фиолетовый — прерывания (softirq)
- Синий — диск (iowait)
- Зеленый — свободное (idle)



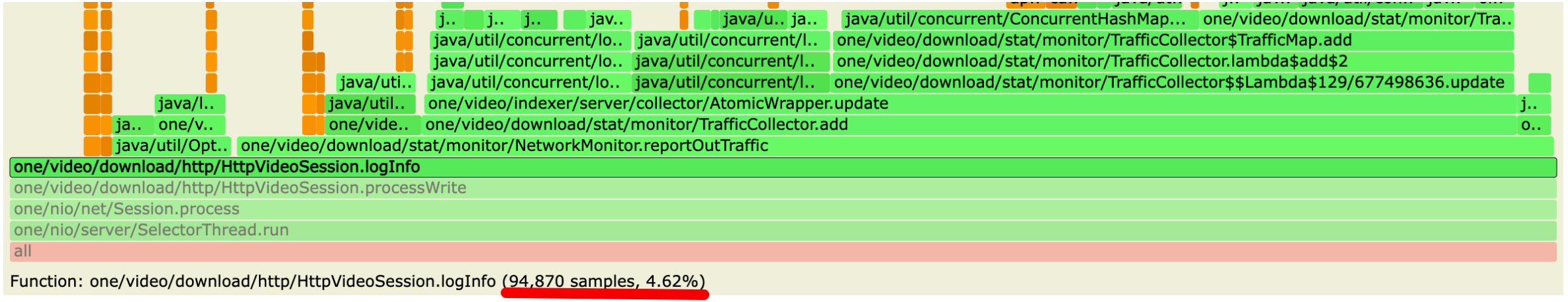
- Зеленый — к серверу (in)
- Синий — от сервера (out)

Излишнее потребление cpu

- Обычная нагрузка



- Повышенная нагрузка



Излишнее потребление cpu

- "Плохое" использование метода

Излишнее потребление cpu

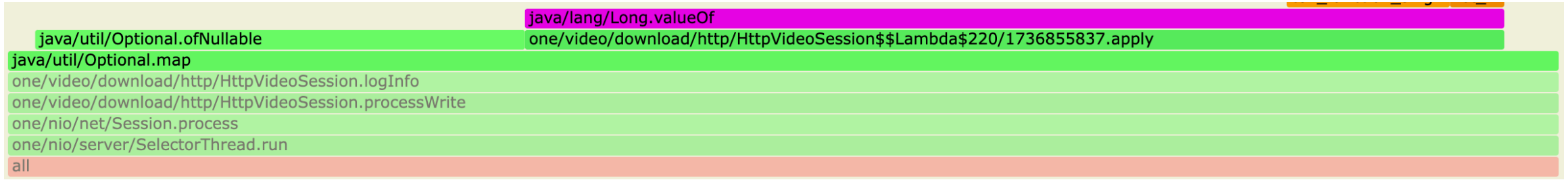
- "Плохое" использование методов

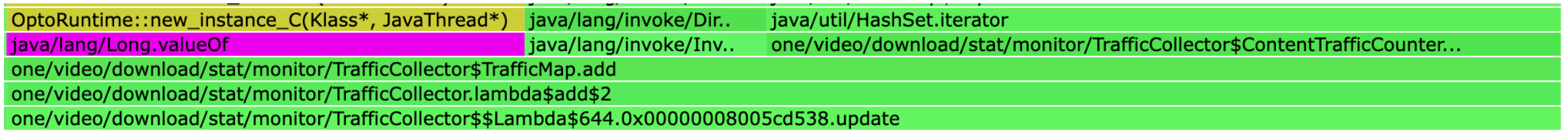
> Returns the number of elements in this queue. If this queue contains more than `Integer.MAX_VALUE` elements, returns `Integer.MAX_VALUE`.
>
> Beware that, unlike in most collections, this method is *NOT* a constant-time operation. Because of the asynchronous nature of these queues, determining the current number of elements requires an O(n) traversal. Additionally, if elements are added or removed during execution of this method, the returned result may be inaccurate. Thus, this method is typically not very useful in concurrent applications.
>
> Returns: the number of elements in this queue

```
public int size() {
    restartFromHead: for (;;) {
        int count = 0;
        for (Node<E> p = first(); p != null;) {
            if (p.item != null)
                if (++count == Integer.MAX_VALUE)
                    break;  // @see Collection.size()
            if (p == (p = p.next))
                continue restartFromHead;
        }
        return count;
    }
}
```

Излишнее потребление cpu+heap

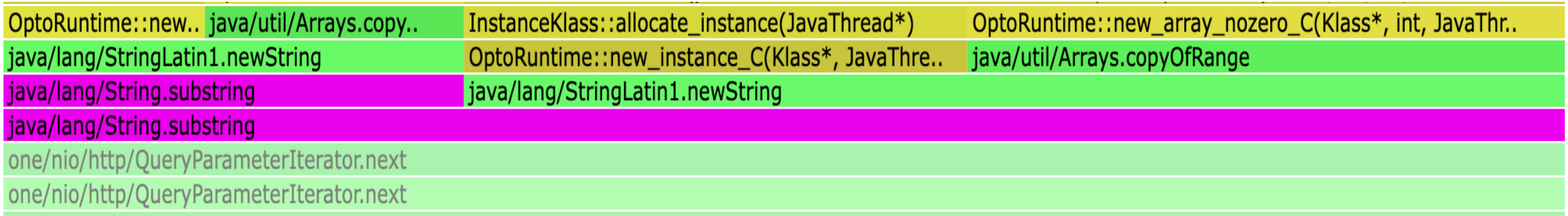
- "Лишний" боксинг примитивов





Излишнее потребление cpu+heap

- Массивное создание подстрок



Большой time-to-safepoint

- Total time for which application threads were stopped: **0.258** seconds, Stopping threads took: **0.201** seconds

Большой time-to-safepoint

- Total time for which application threads were stopped: **0.258** seconds, Stopping threads took: **0.201** seconds
- -XX:SafepointTimeout
  -XX:SafepointTimeoutDelay

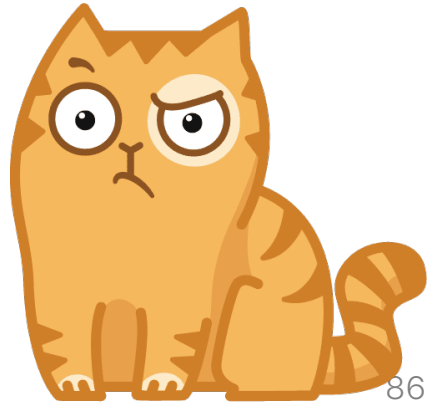
Большой time-to-safepoint, случай 1

- Список пуст

```
[warning][safepoint    ]
[warning][safepoint    ] # SafepointSynchronize::begin: Timeout detected:
[warning][safepoint    ] # SafepointSynchronize::begin: Timed out while spinning to reach a safepoint.
[warning][safepoint    ] # SafepointSynchronize::begin: Threads which did not reach the safepoint:
[warning][safepoint    ]
```

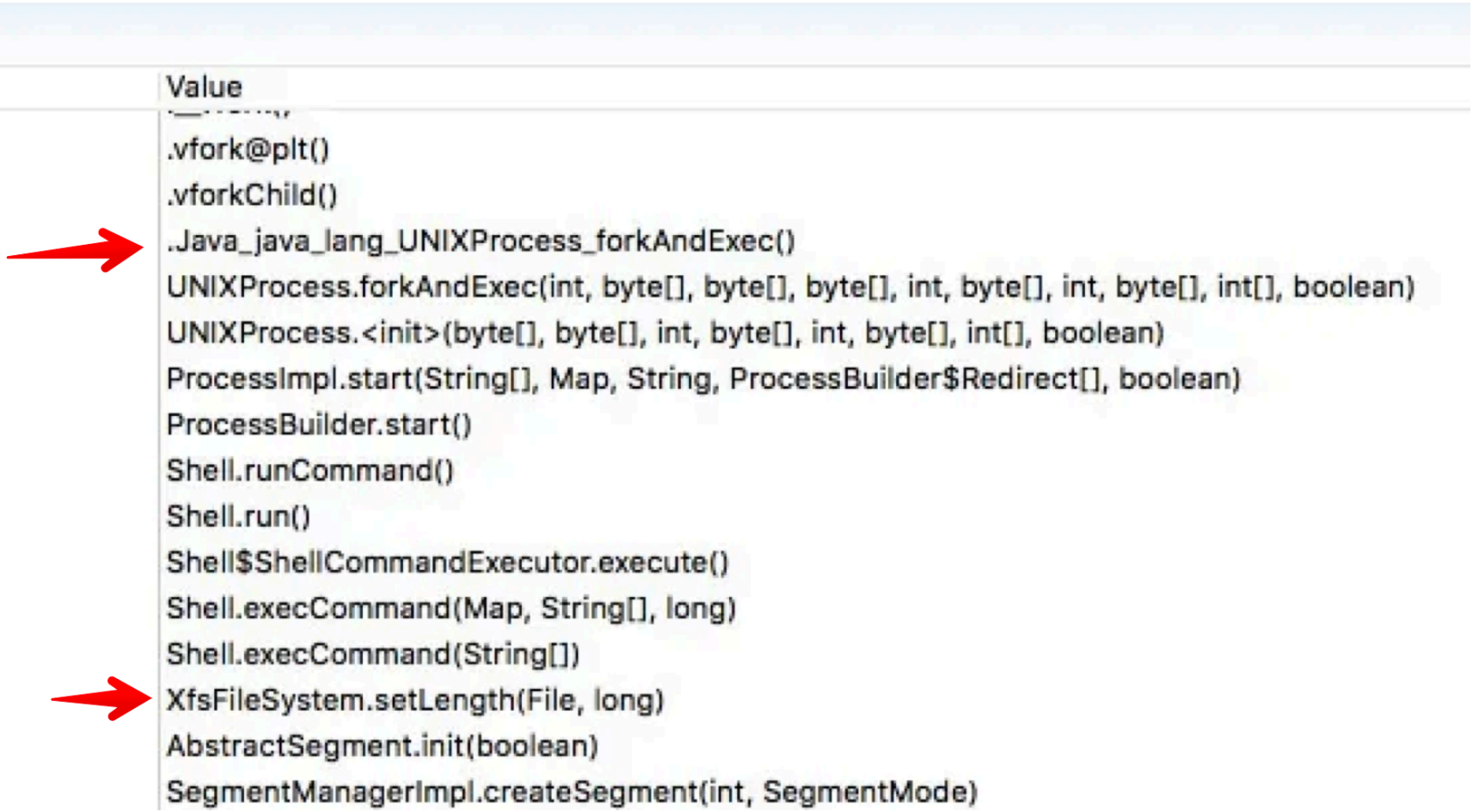
А ГДЕ?

```
[warning][safepoint    ]
[warning][safepoint    ] # SafepointSynchronize::begin: (End of list)
[info   ][gc,start     ] GC(120) Pause Final Update Refs
```



Большой time-to-safepoint, случай 1

- profiler -e wall -i 1ms -f profile.jfr --ttsp



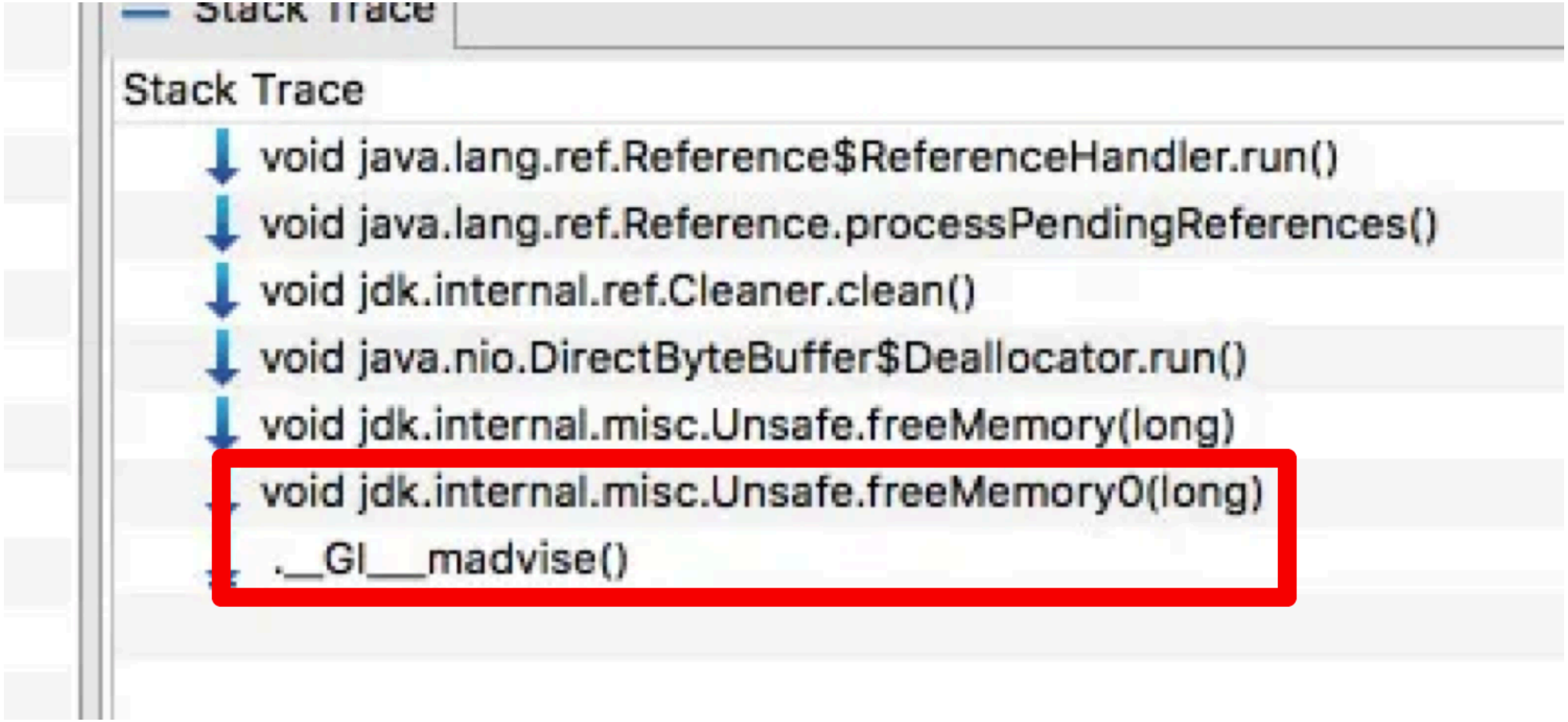
Большой time-to-safepoint, случай 2

- Тред есть, но причина неясна

```
[info   ][gc           ] GC(120) Concurrent update references 14.494ms
[warning][safepoint    ]
[warning][safepoint    ] # SafepointSynchronize::begin: Timeout detected:
[warning][safepoint    ] # SafepointSynchronize::begin: Timed out while spinning to reach a safepoint.
[warning][safepoint    ] # SafepointSynchronize::begin: Threads which did not reach the safepoint:
[warning][safepoint    ] # "Reference Handler" #2 daemon prio=10 os_prio=0 cpu=6222.71ms elapsed=4179.99s tid=0x00007f2e12424100 nid=0xf587

[warning][safepoint    ]    java.lang.Thread.State: RUNNABLE
[warning][safepoint    ]
[warning][safepoint    ] # SafepointSynchronize::begin: (End of list)
[info   ][gc,start     ] GC(120) Pause Final Update Refs
[info   ][gc,task      ] GC(120) Using 20 of 20 workers for final reference update
```

Большой time-to-safepoint, случай 2

- profiler -e wall -i 1ms -f profile.jfr --ttsp



Большой time-to-safepoint, случай 2

- Метод нативный, но не совсем

```
public native void freeMemory(long var1);
```

```
Thread 110175: (state = IN_VM)
 - sun.misc.Unsafe.freeMemory(long) @bci=0 (Compiled frame; information may be imprecise)
 - java.nio.DirectByteBuffer$Deallocator.run() @bci=17, line=94 (Compiled frame)
 - sun.misc.Cleaner.clean() @bci=12, line=143 (Compiled frame)
```

Большой time-to-safepoint, случай 2

- https://gist.github.com/apangin/bccb9297c6bbc91e39ae3b724e06c954

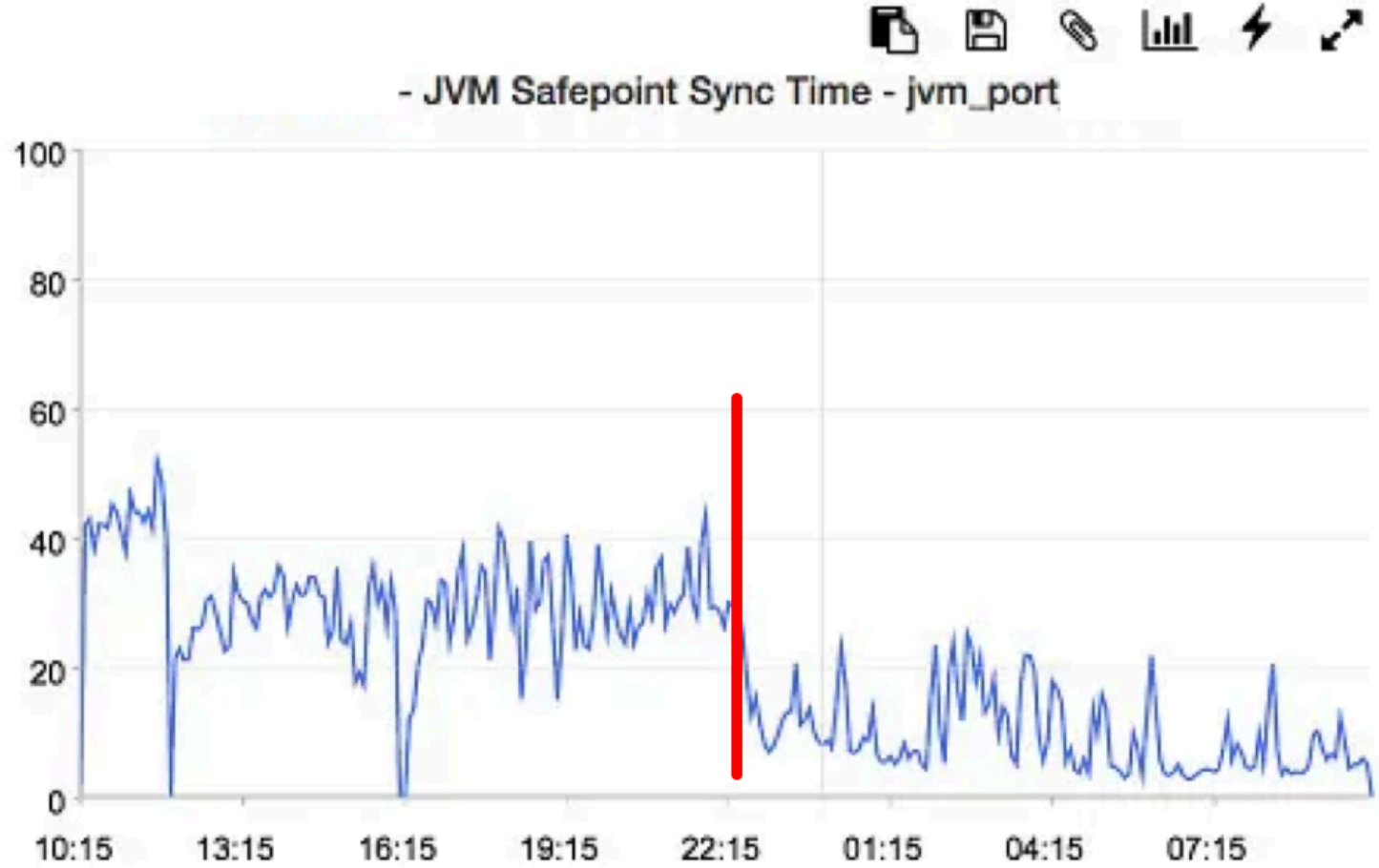
```
jlong JNICALL unmalloc_AllocateMemory(JNIEnv* env, jobject unsafe, jlong size) {
    return (intptr_t) malloc((size_t)size);
}

jlong JNICALL unmalloc_ReallocateMemory(JNIEnv* env, jobject unsafe, jlong addr, jlong size) {
    return (intptr_t) realloc((void*)(intptr_t)addr, (size_t)size);
}

void JNICALL unmalloc_FreeMemory(JNIEnv* env, jobject unsafe, jlong addr) {
    free((void*)(intptr_t)addr);
}


static const JNINativeMethod new_unsafe_methods[] = {
    {(char*)"allocateMemory0",   (char*)"(J)J",  (void*)unmalloc_AllocateMemory},
    {(char*)"reallocateMemory0", (char*)"(JJ)J", (void*)unmalloc_ReallocateMemory},
    {(char*)"freeMemory0",       (char*)"(J)V",  (void*)unmalloc_FreeMemory}
};
```

Большой time-to-safepoint, случай 2

- Компилируем
  - gcc -O3 -fno-omit-frame-pointer -fPIC -shared -olibunmalloc.so unmalloc.c
- На лету подгружаем агент
  - https://github.com/jattach/jattach
  - jattach <pid> load libunmalloc.so true

Большой time-to-safepoint, случай 2

- Получаем снижение ttsp



Обновляем jemalloc

- https://bugs.openjdk.org/browse/JDK-8215355

JDK / JDK-8215355

Object monitor deadlock with no threads holding the monitor (using jemalloc 5.1)

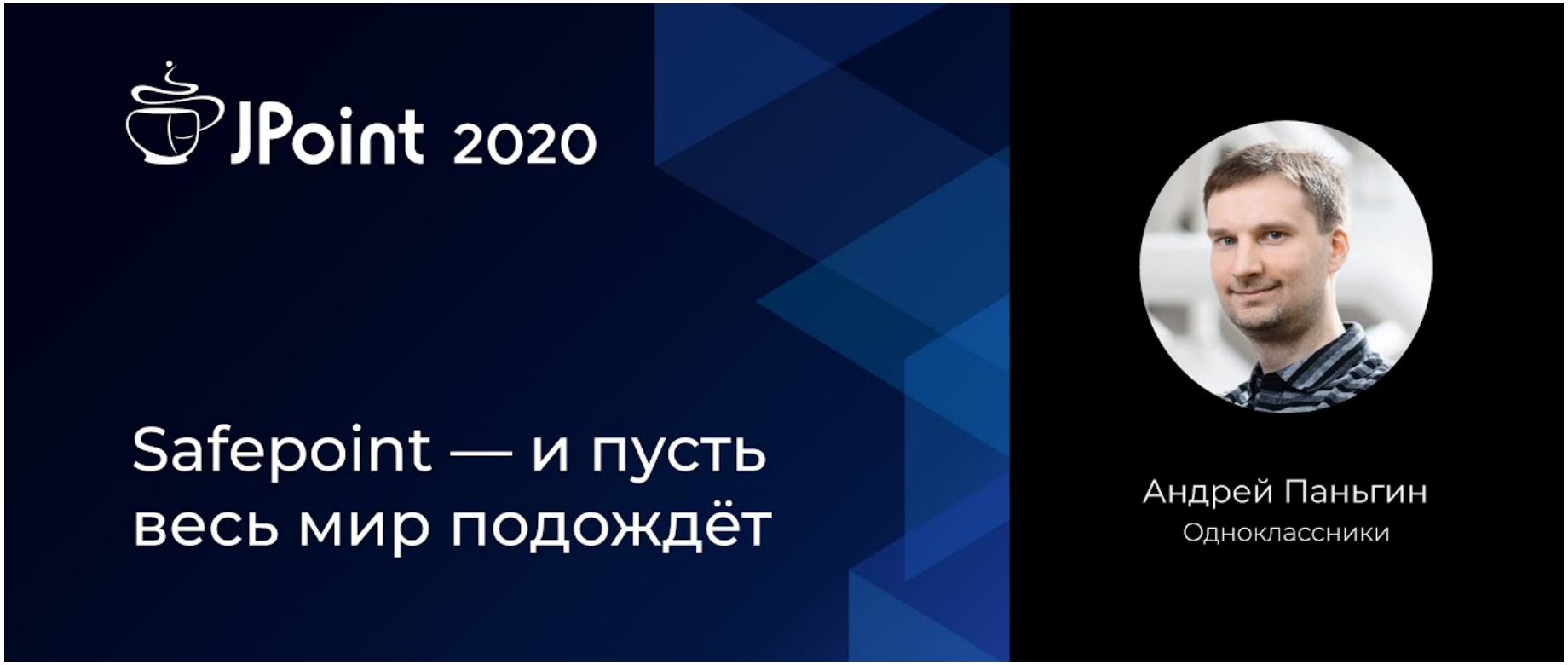
Resolved

Details

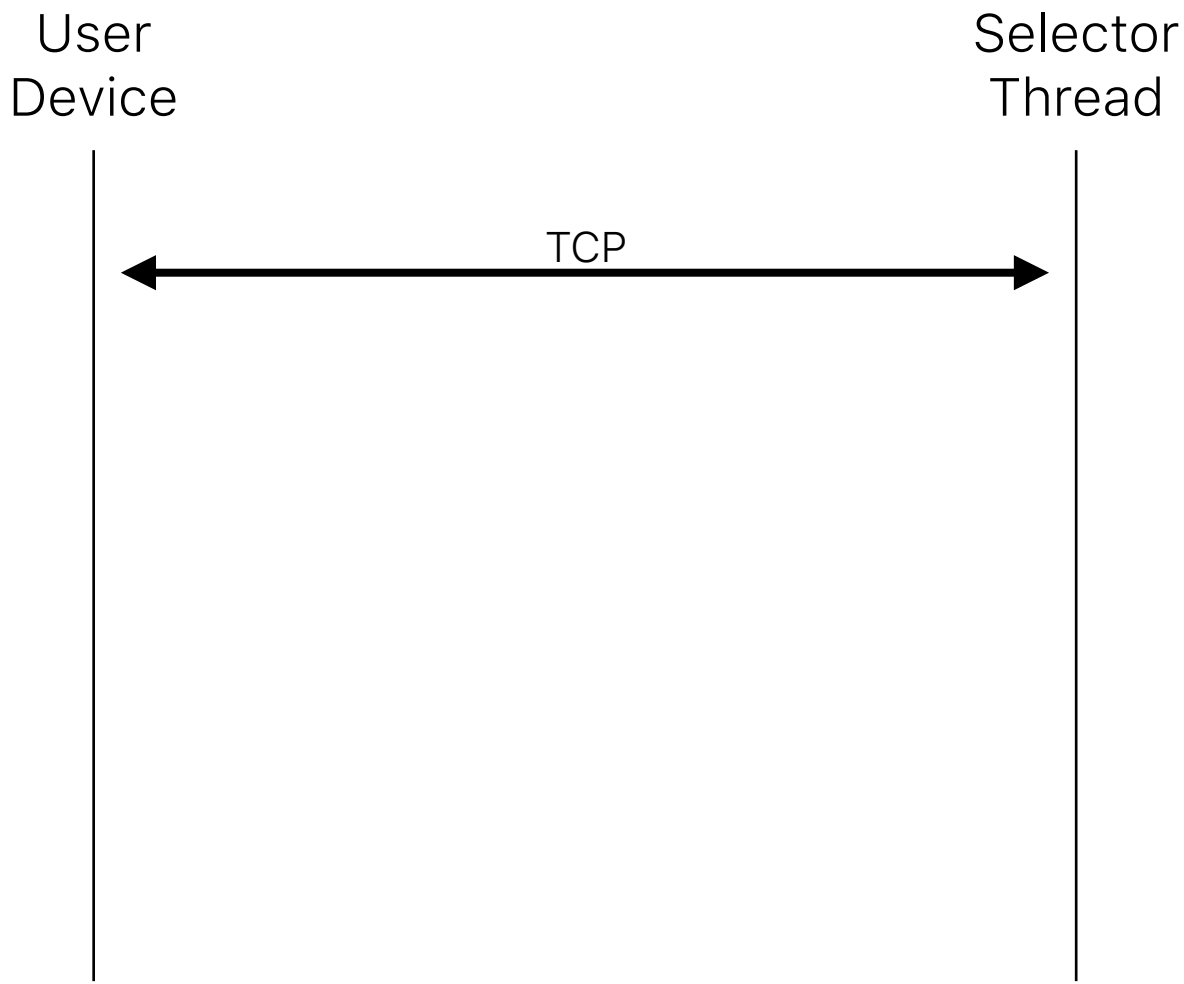
Type:	Bug	Resolution:	Fixed
Priority:	P3	Fix Version/s:	14
Affects Version/s:	7, 8, 11, 12		
Component/s:	hotspot		
Labels:	8-bp, additional-information-received, dcsfai, jdk11u-fix-request, jdk11u-fix-yes, jdk13u-fix-request, jdk13u-fix-yes, jdk8u-fix-request, jdk8u-fix-yes, noreg-hard, oracle-triage-14, redhat-interest, reproducer-yes, webbug		
Subcomponent:	runtime		
Resolved In Build:	b24		

Подробнее про safepoint

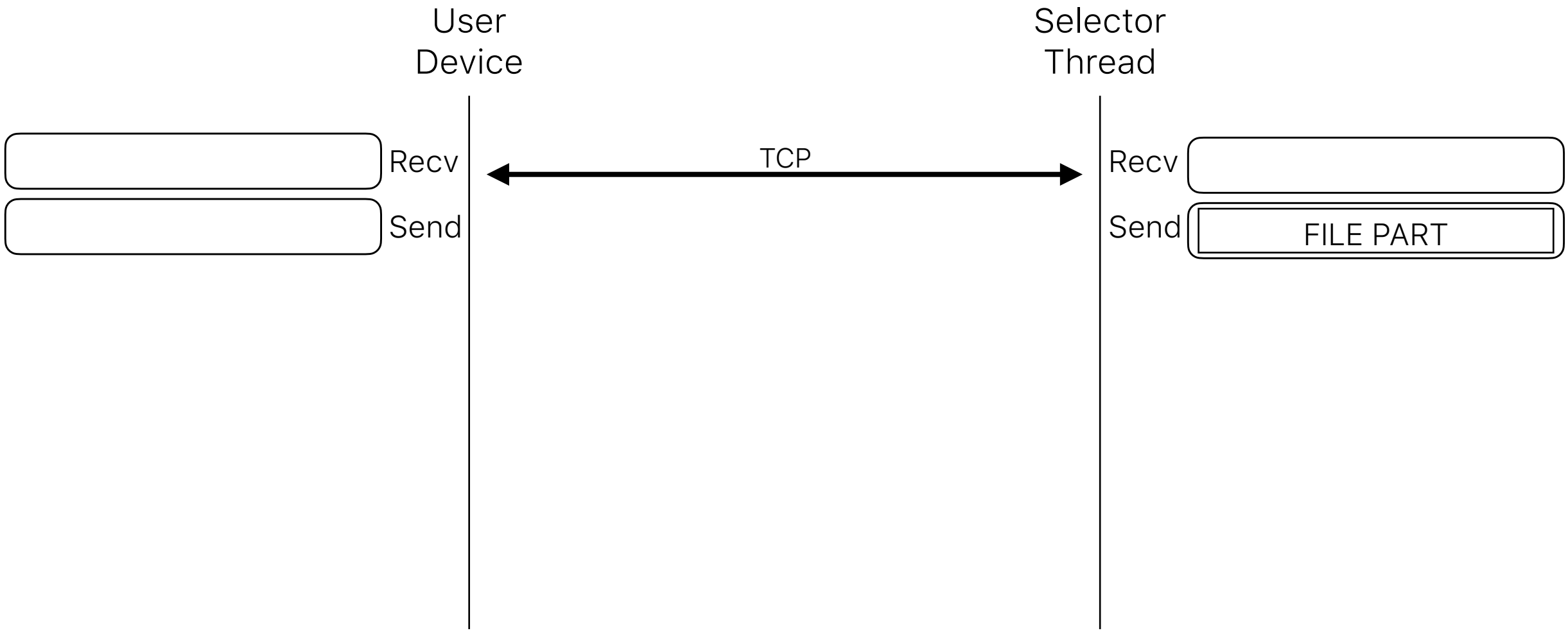
- https://youtu.be/rthWVvU9gWo



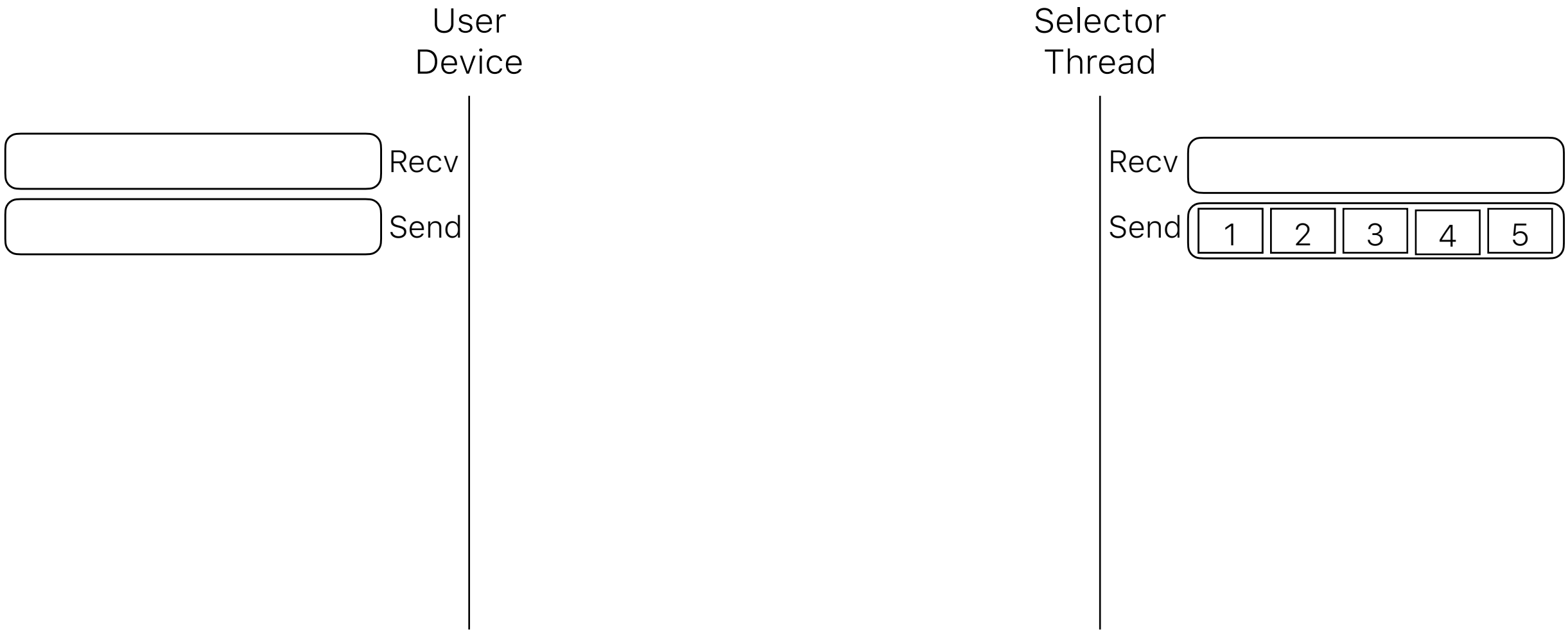
Как работает TCP



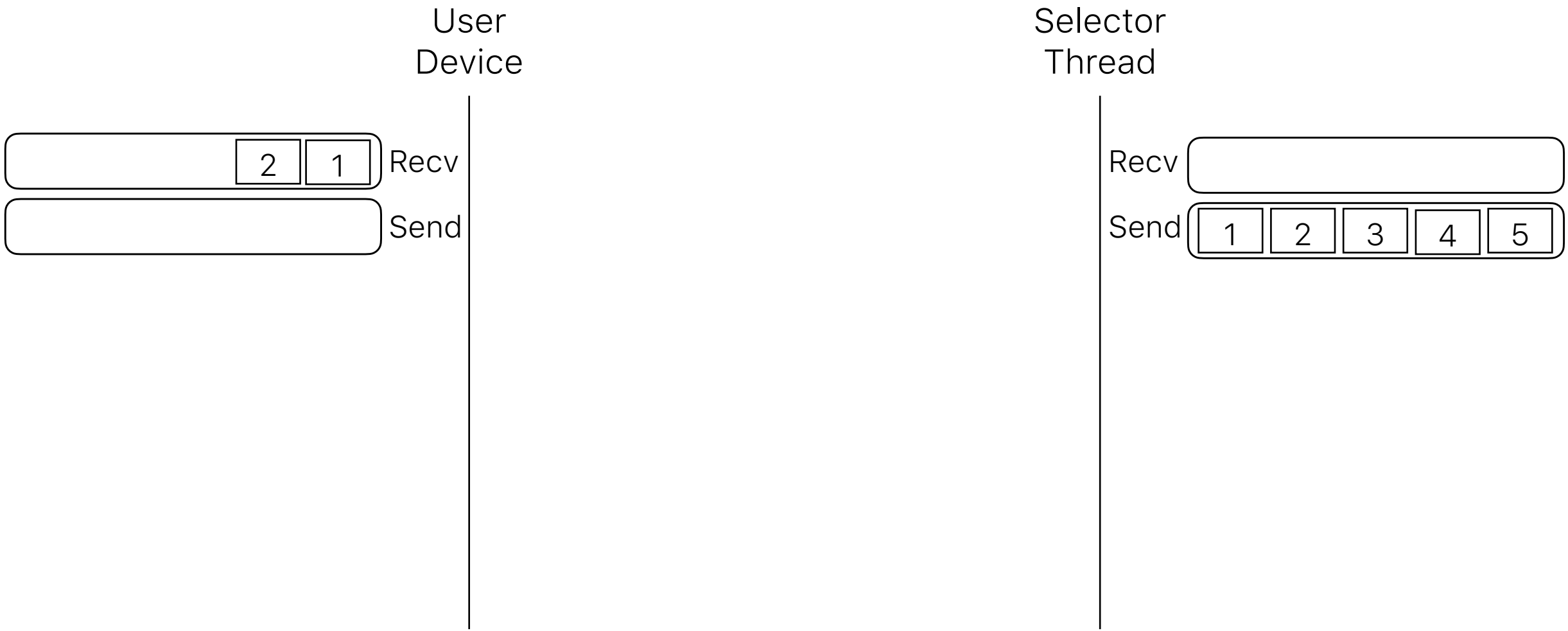
Как работает TCP



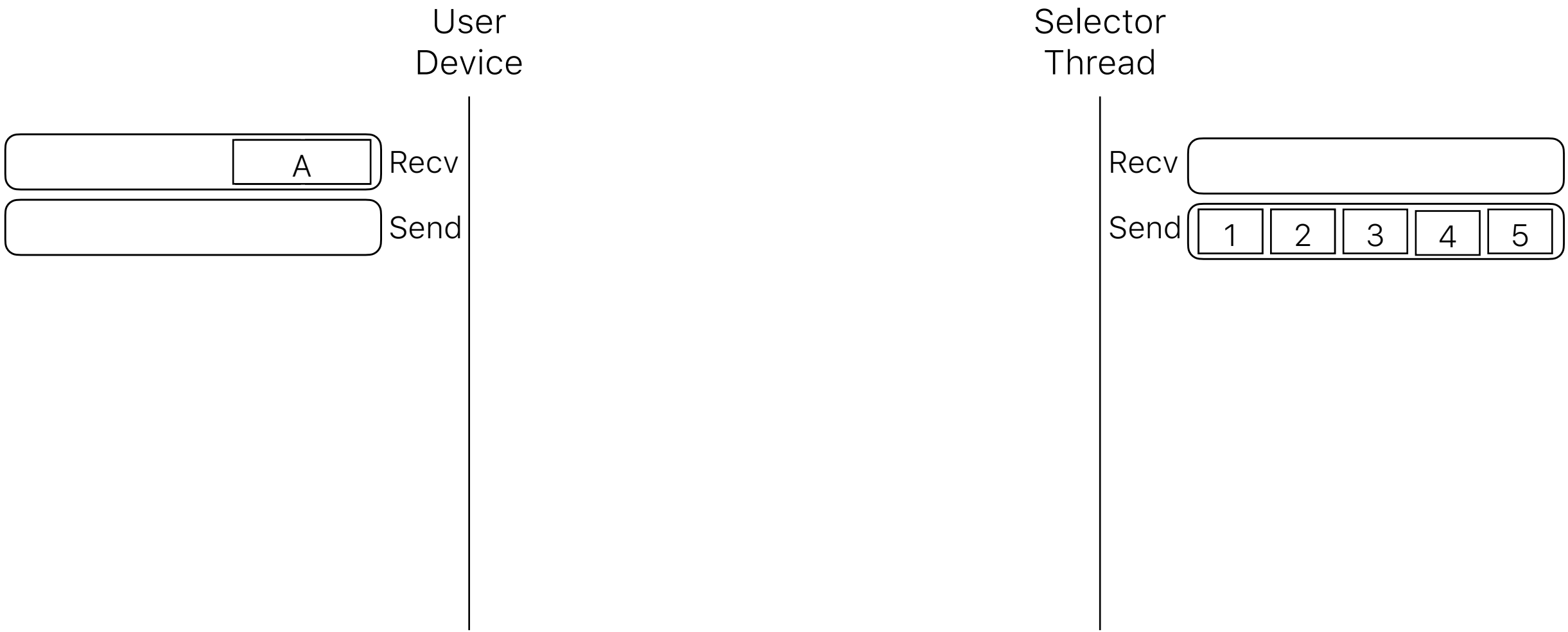
Как работает TCP



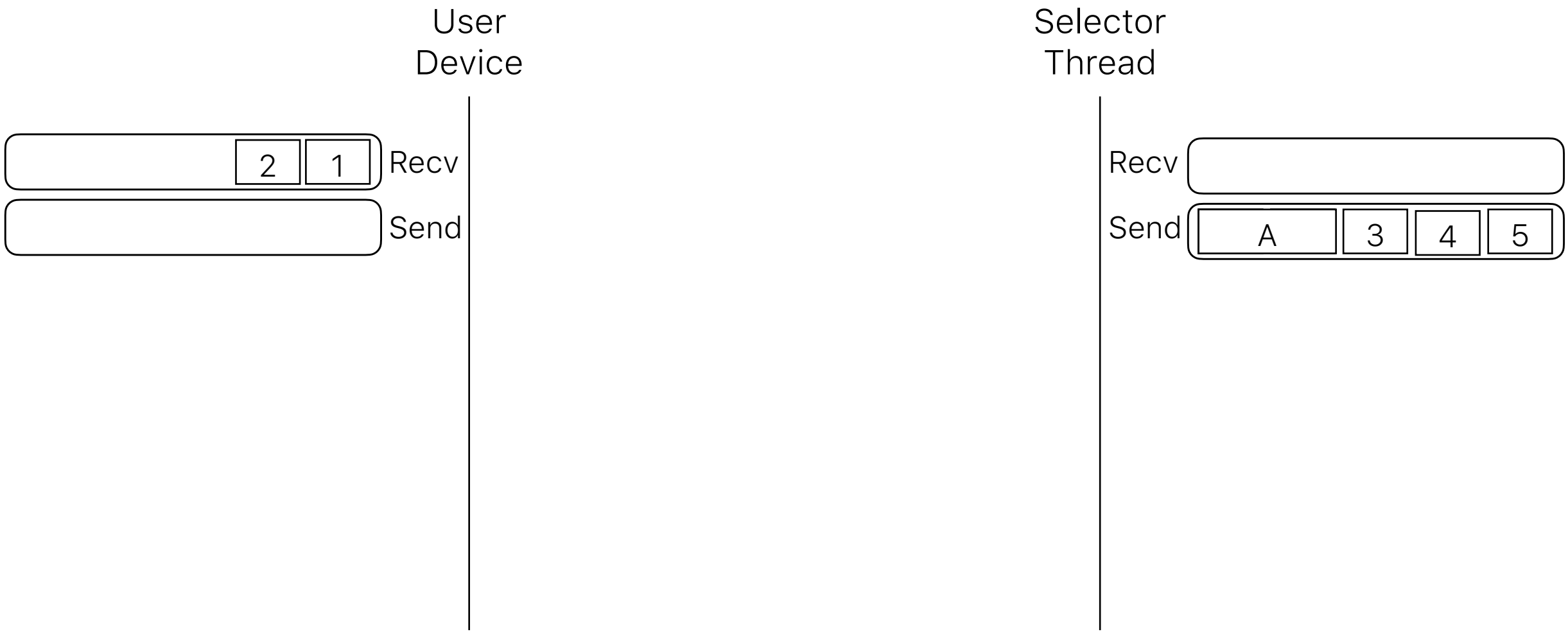
Как работает TCP



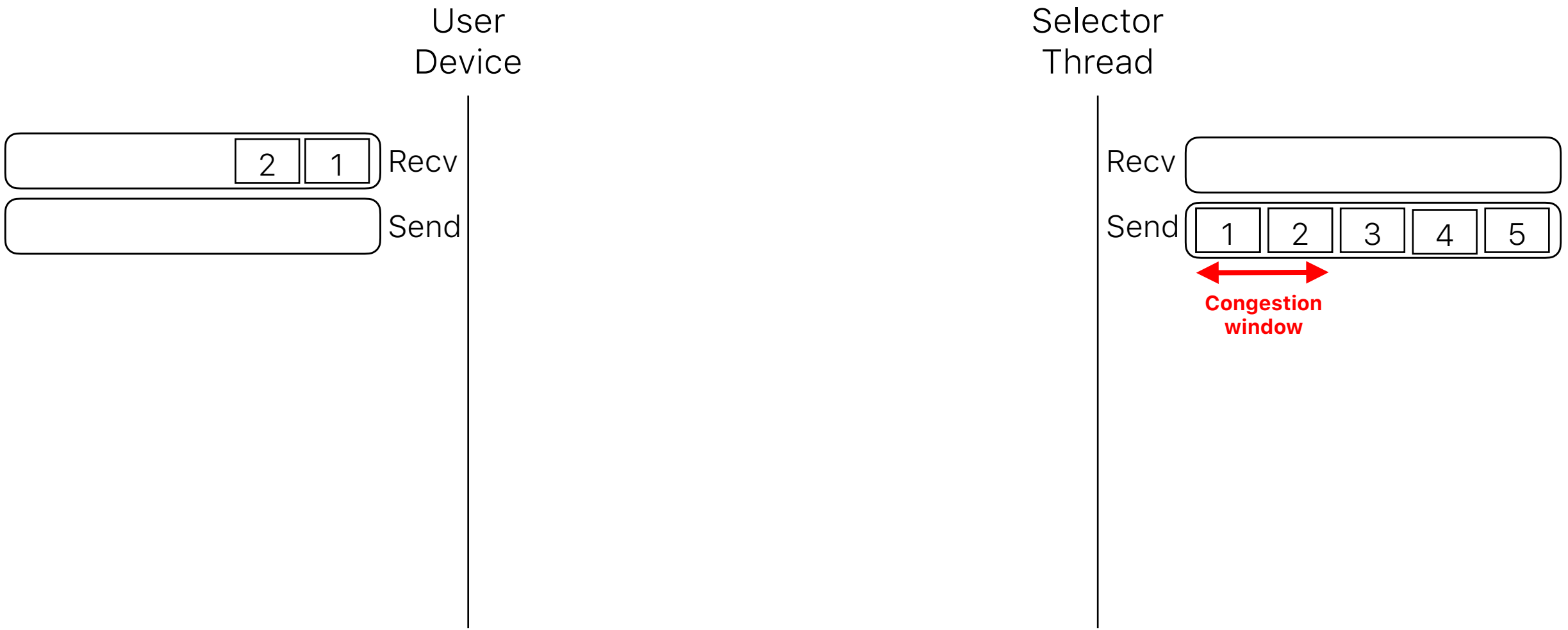
Как работает TCP



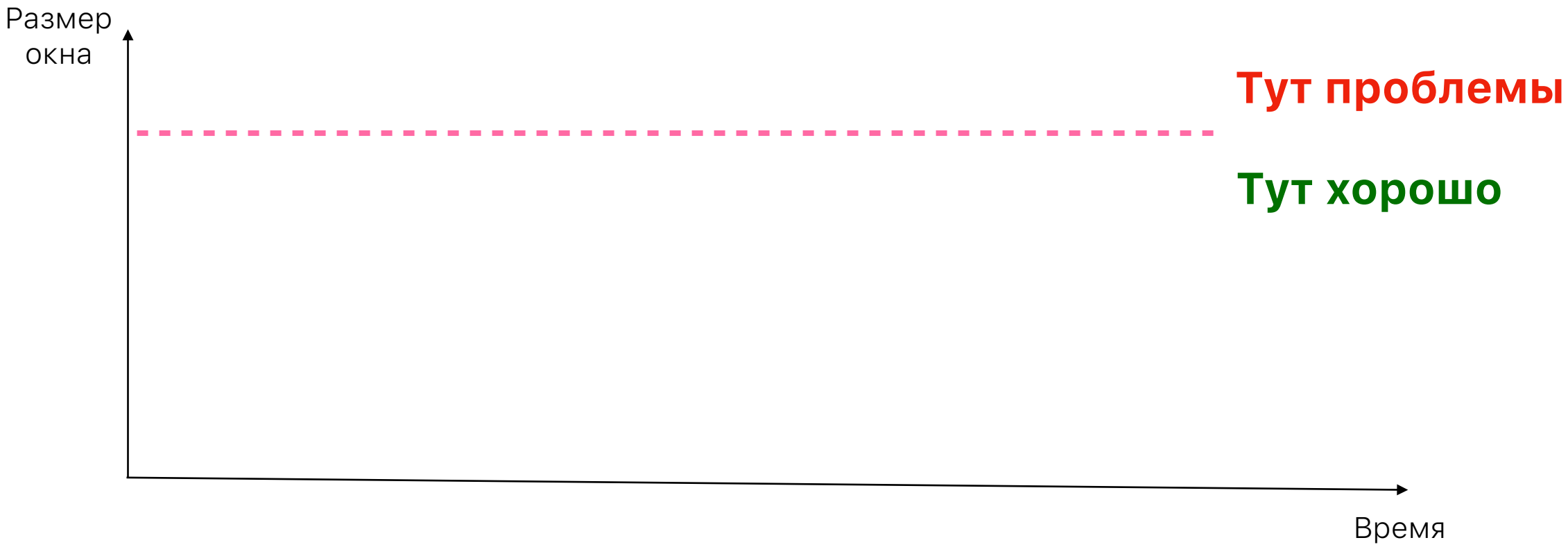
Как работает TCP



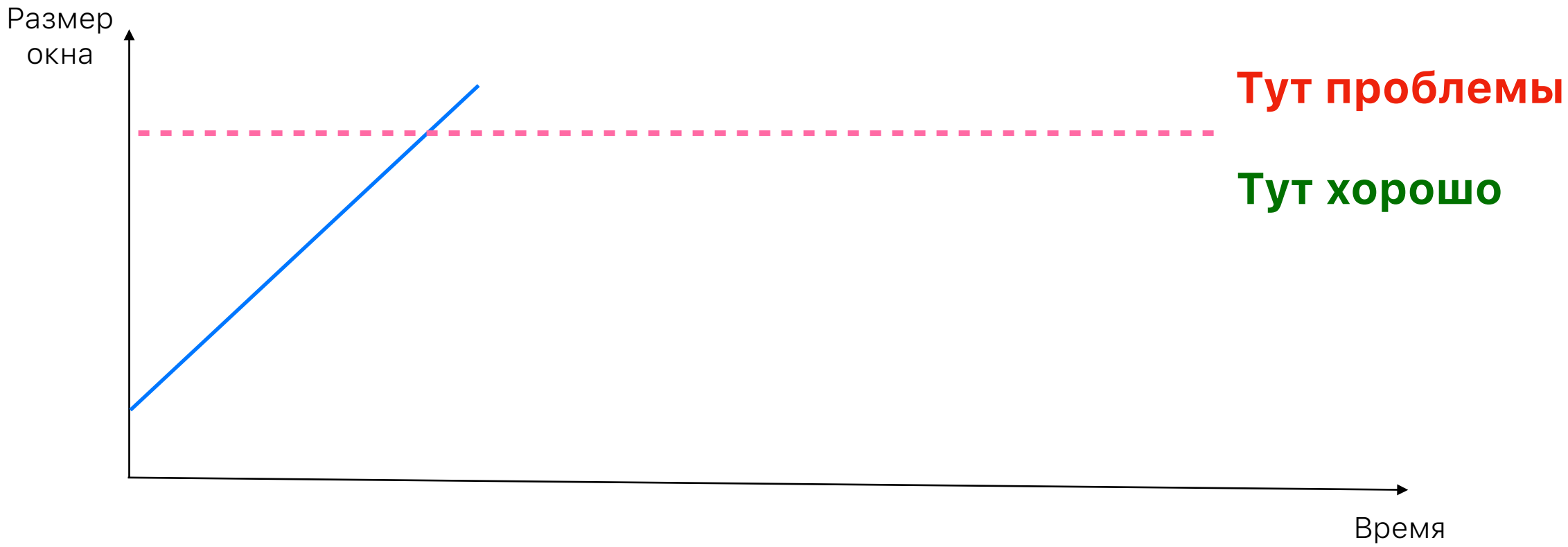
Congestion control



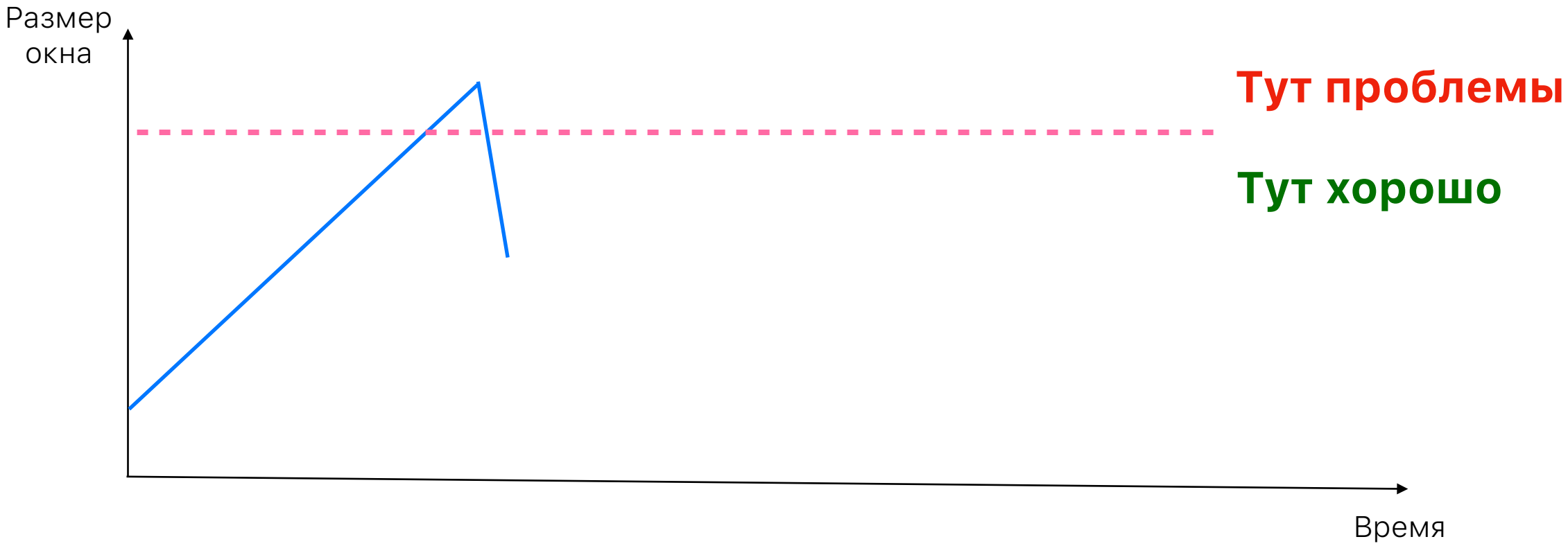
Congestion control



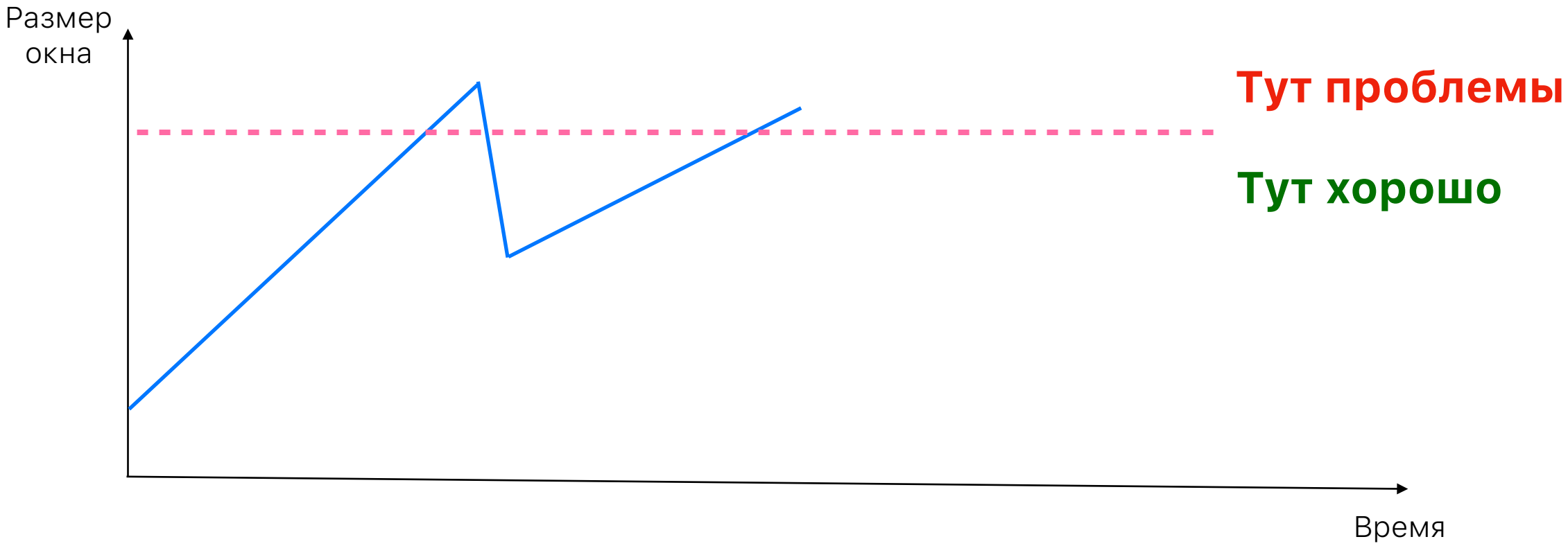
Congestion control



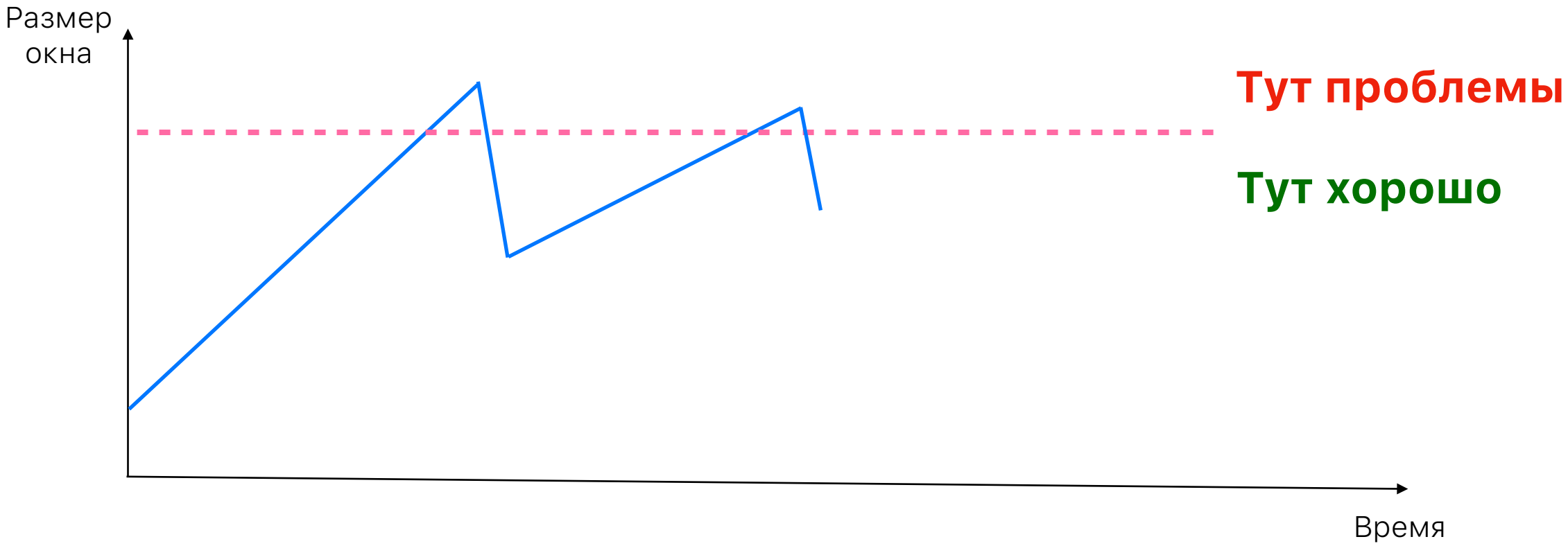
Congestion control



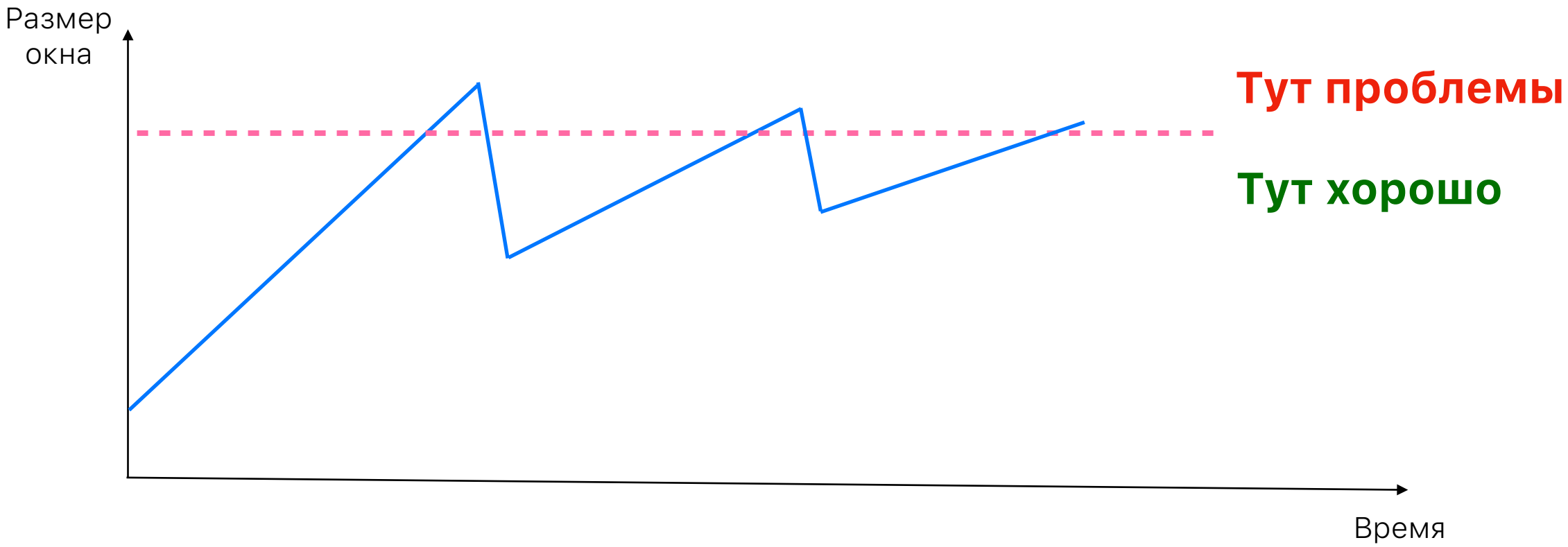
Congestion control



Congestion control



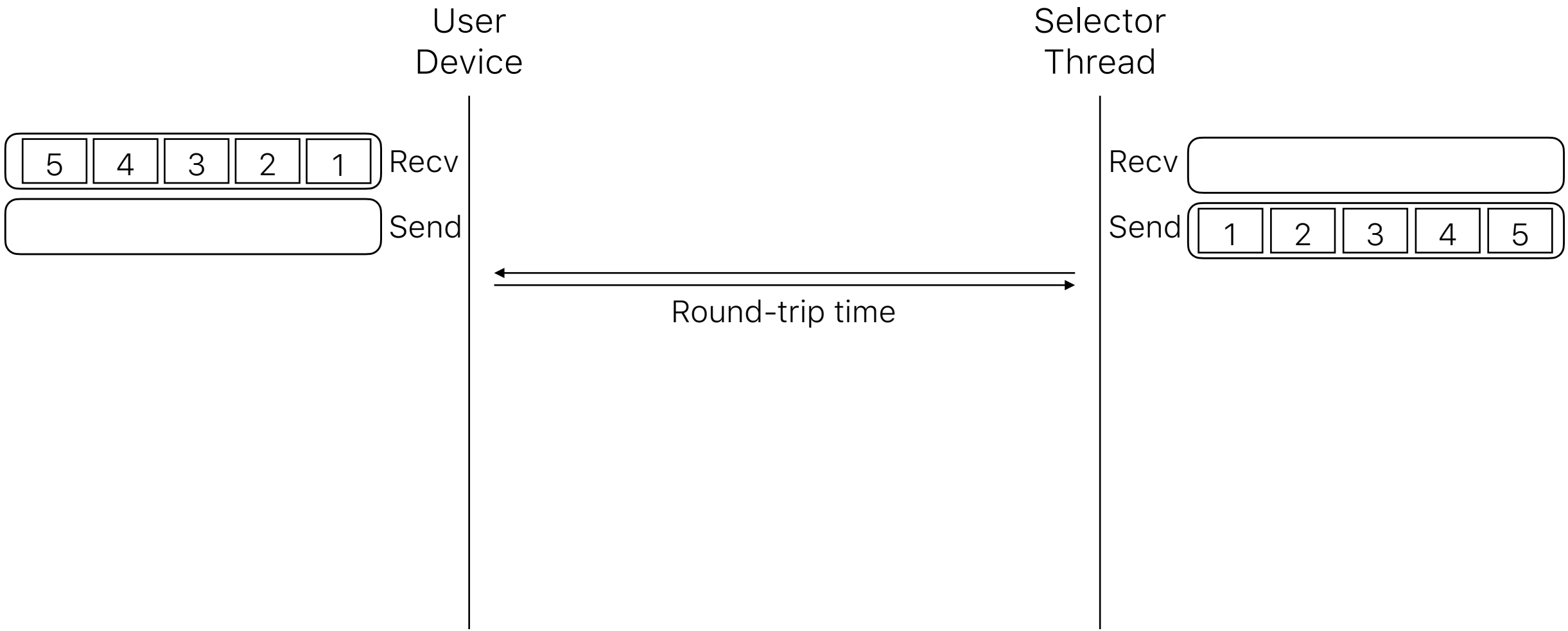
Congestion control



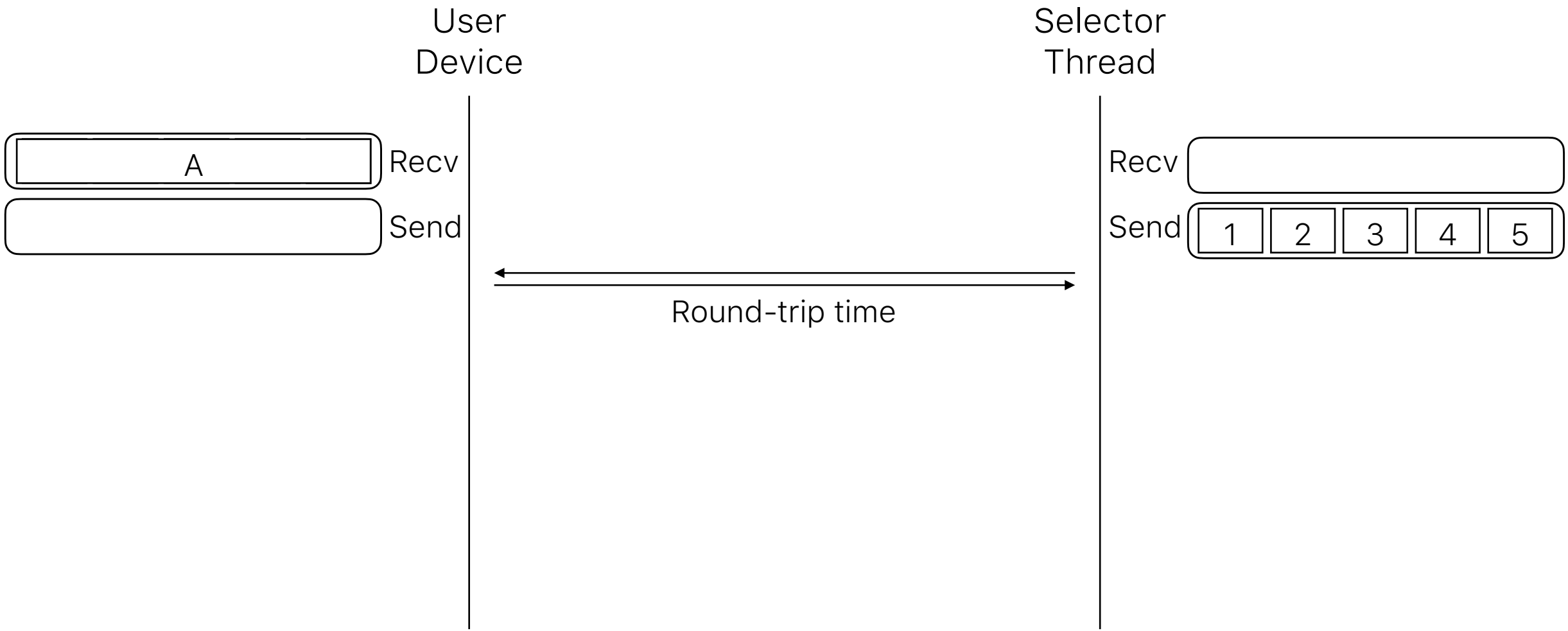
Выбираем алгоритм congestion control

- Выбираем congestion control алгоритм с другим типом "обратной связи"
  - был CUBIC (Loss-based feedback)
  - стал BBR (Delay-based feedback)
  - sysctl -w net.ipv4.tcp_congestion_control=bbr
- Переходим на ядро 5.9
  - из-за проблем связки ядро/драйвер NIC/QoS/cc алгоритм

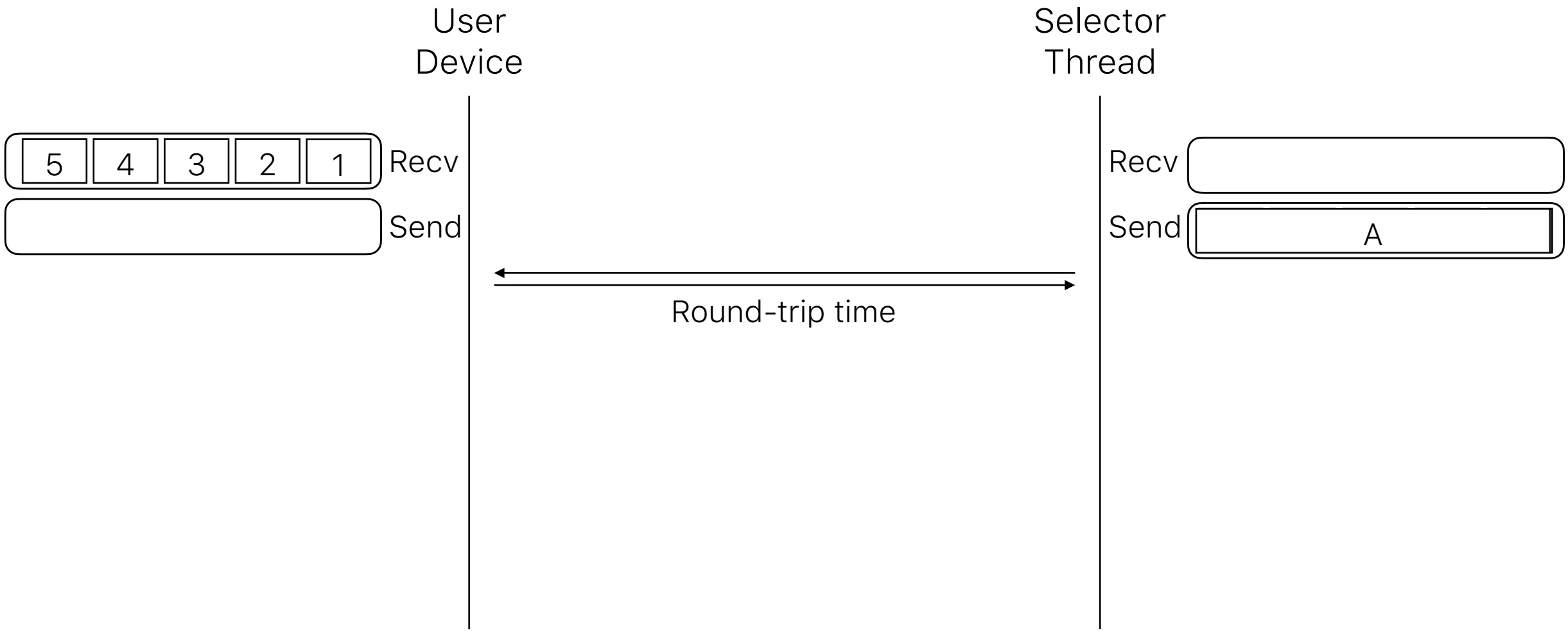
Считаем максимальную скорость



Считаем максимальную скорость



Считаем максимальную скорость



Считаем максимальную скорость

- Bandwidth-delay product (BDP) это теоретическая максимальная пропускная способность

$$BDP_{bytes} = BW_{bytes\ per\ second} \times RTT_{seconds}$$

$$BW = \frac{BDP}{RTT}$$

Считаем максимальную скорость

- Bandwidth-delay product (BDP) это теоретическая максимальная пропускная способность

$$BDP_{bytes} = BW_{bytes\ per\ second} \times RTT_{seconds}$$

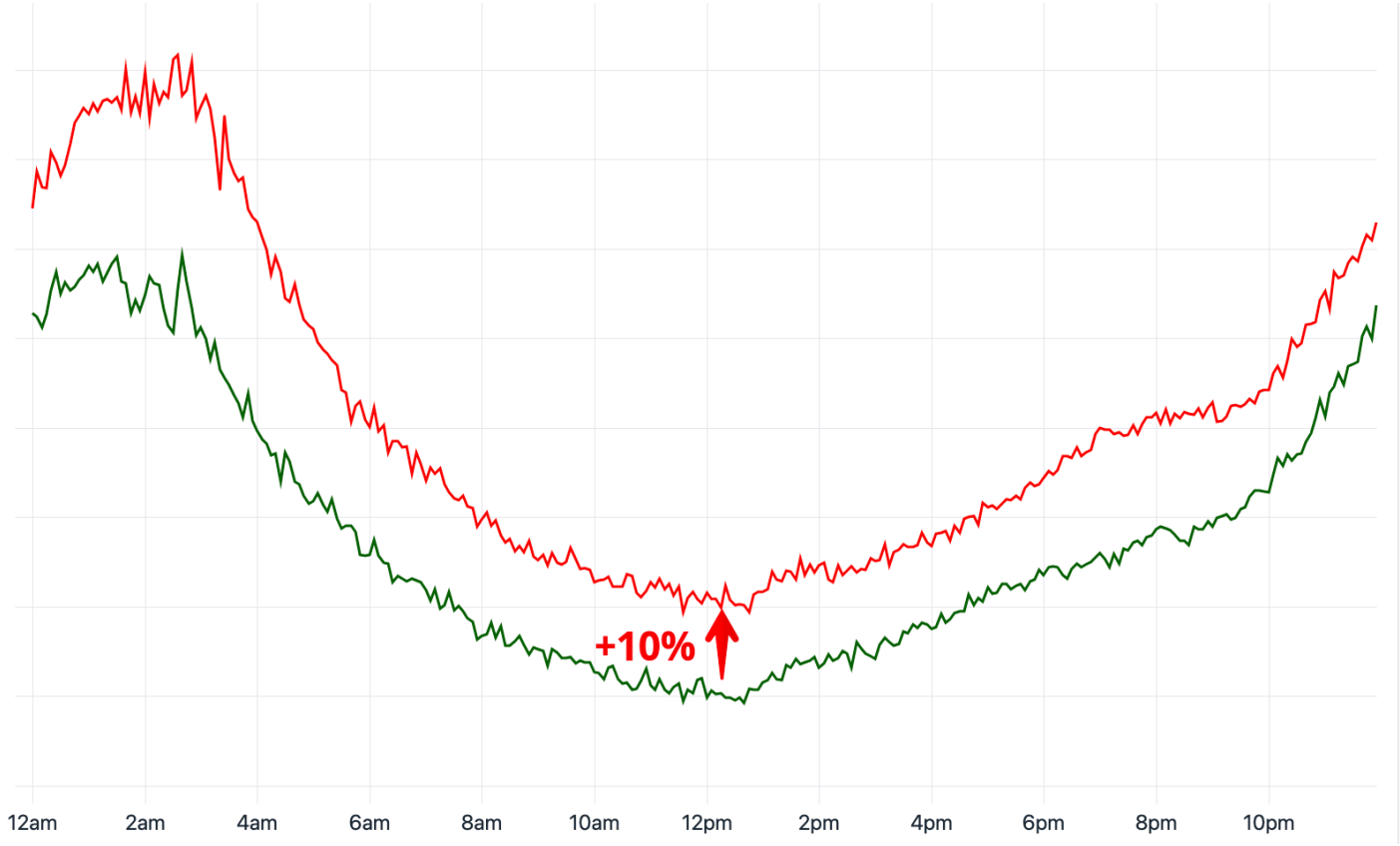
$$BW = \frac{BDP}{RTT}$$

$BW_{(\text{дальний восток, 256 КБ})} = 262144 * 8 / 0.5 =$ **4** Mbps

$BW_{(\text{дальний восток, 4 МБ})} = 4194304 * 8 / 0.5 =$ **32** Mbps

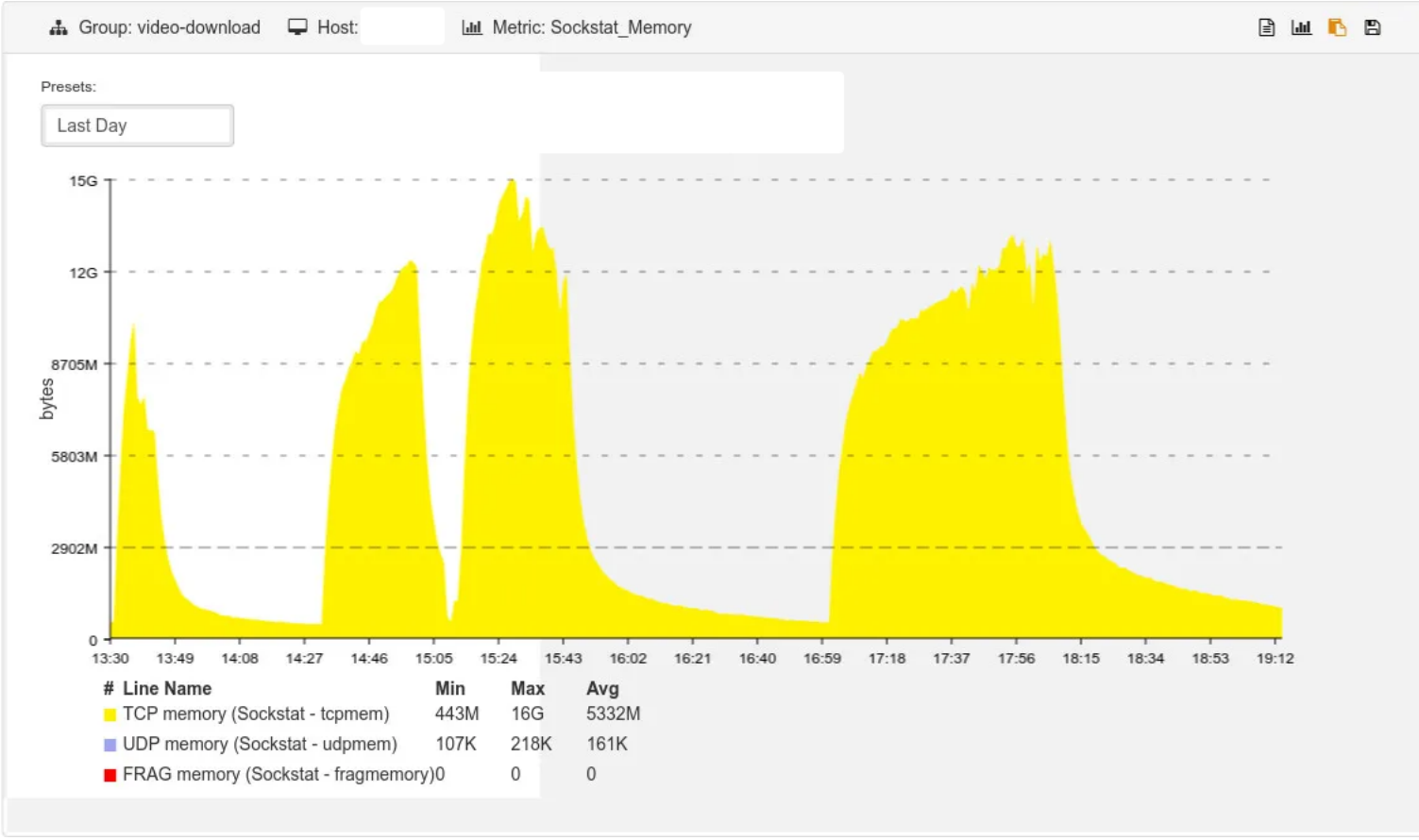
Увеличиваем буфер отправки

- До 4 мегабайт, получаем прирост клиентской скорости



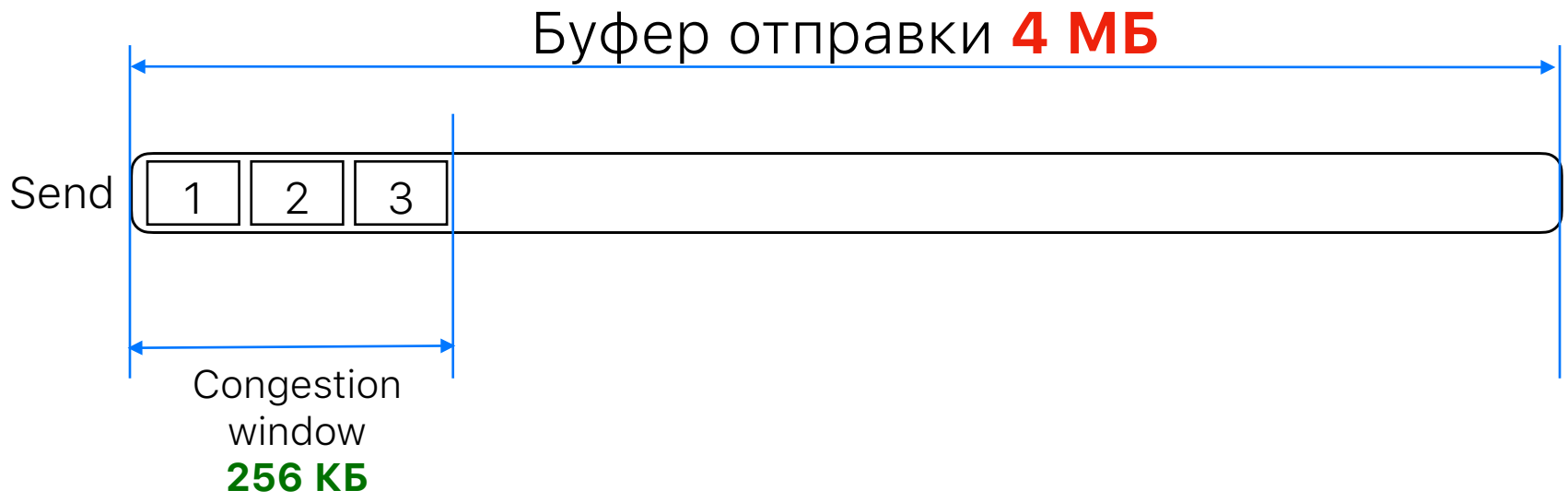
Уменьшаем потребление памяти

- Увеличиваем net.ipv4.tcp_mem
  sysctl -w net.ipv4.tcp_mem = 15000000 15000000 15000000



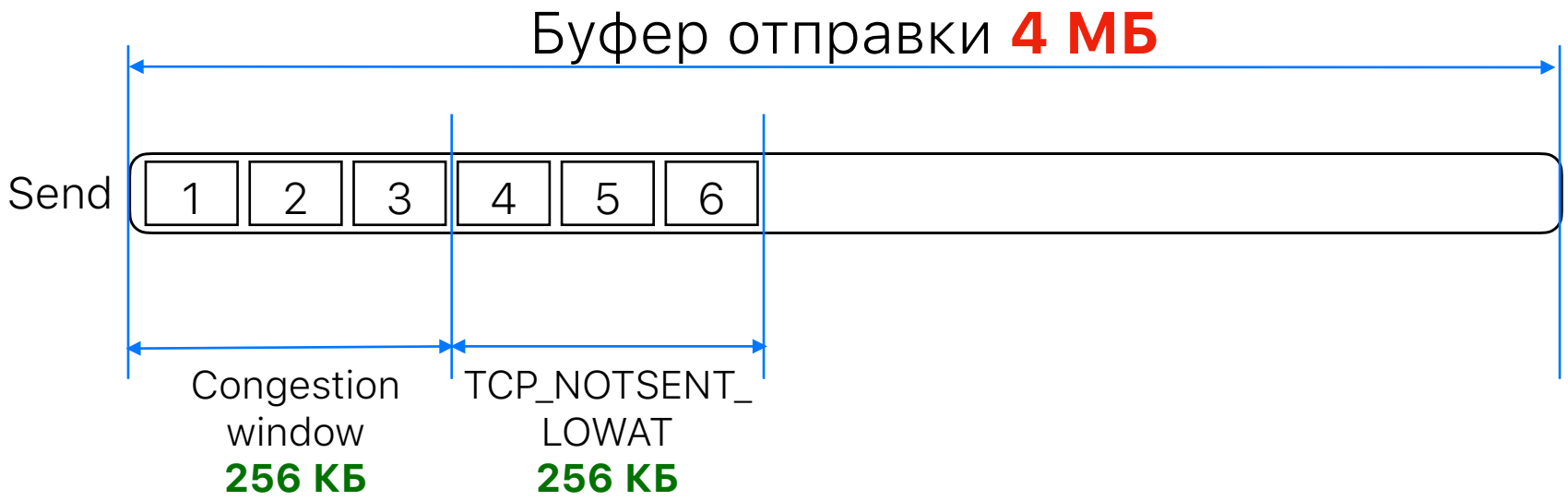
Уменьшаем потребление памяти

- Включаем TCP_NOTSENT_LOWAT
  sysctl -w net.ipv4.tcp_notsent_lowat=262144

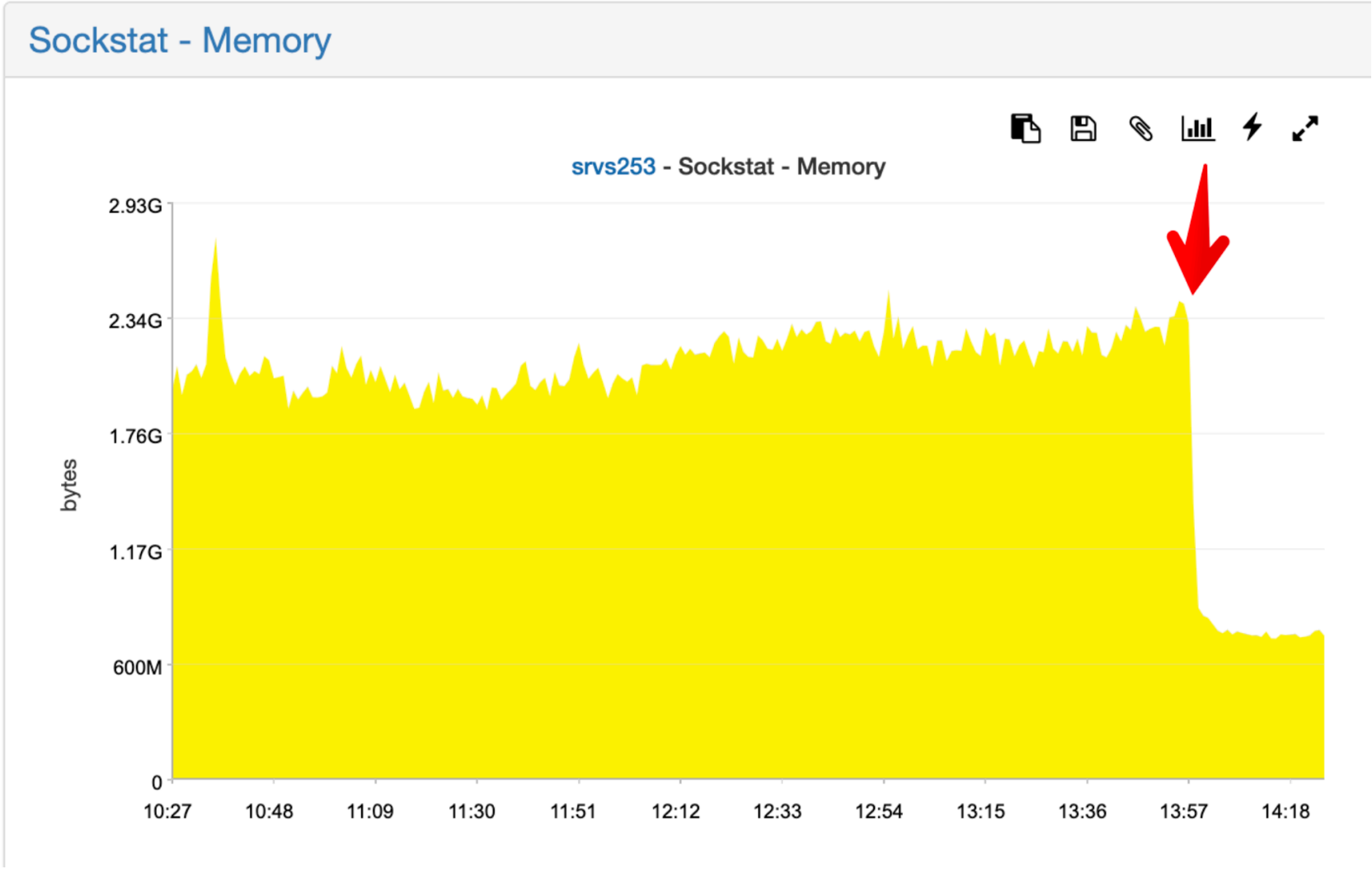


Уменьшаем потребление памяти

- Включаем TCP_NOTSENT_LOWAT
  sysctl -w net.ipv4.tcp_notsent_lowat=262144

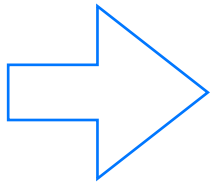


Уменьшаем потребление памяти



Обновляем конфиг

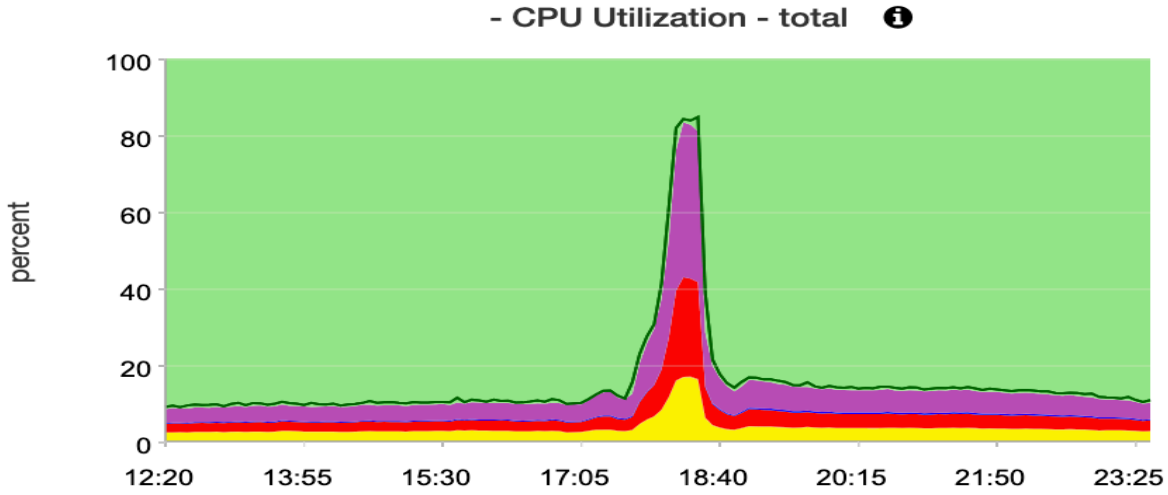
```
selectors: 40
affinity: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 256k
 - port: 443
   recvBuf: 64k
   sendBuf: 256k
   threads: 20
   reusePort: true
   ssl:
    - protocols: TLSv1 TLSv1.1 TLSv1.2
      certFile: ./file.cert
      privateKeyFile: ./file.key
      rdrand: true
```



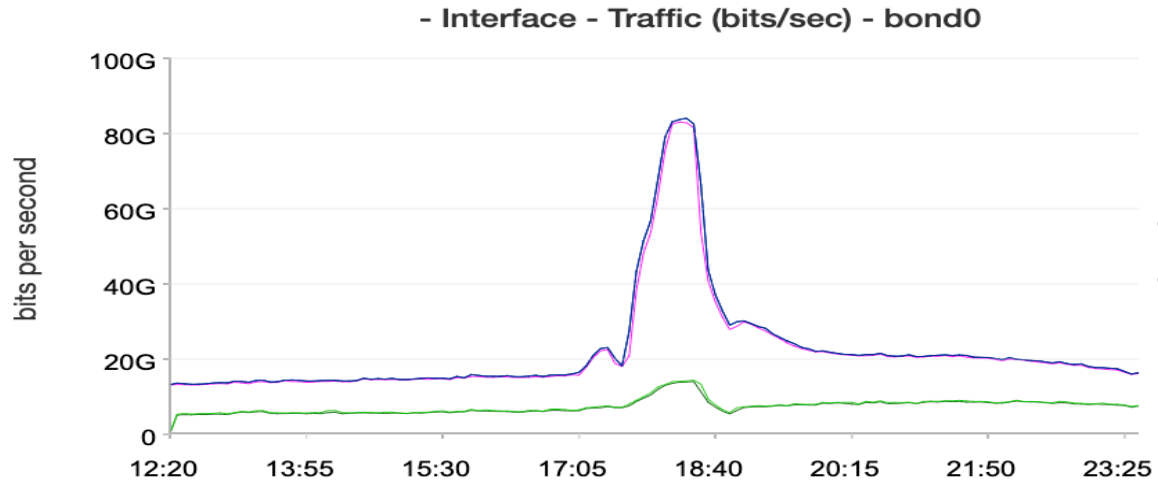
```
selectors: 88
affinity: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 4m
 - port: 443
   recvBuf: 64k
   sendBuf: 4m
   threads: 20
   reusePort: true
   notsentLowat: 512k
   ssl:
    - protocols: TLSv1 TLSv1.1 TLSv1.2
      certFile: ./file.cert
      privateKeyFile: ./file.key
      rdrand: true
```

Результаты

- Держим до ~~40~~ 100 Gbps



- Желтый — приложение (user)
- Красный — ядро (system)
- Фиолетовый — прерывания (softirq)
- Синий — диск (iowait)
- Зеленый — свободное (idle)



- Зеленый — к серверу (in)
- Синий — от сервера (out)

Последние достижения

2022 — настоящее время

1. База и запуск
2. Внедряем HTTPS
3. Прокачиваем до 100G
4. Последние достижения

Тестовый стенд

- Xeon Gold 62xx
- 512 GB RAM
- 100 Gbps NIC
- SSD
- ядро 5.9

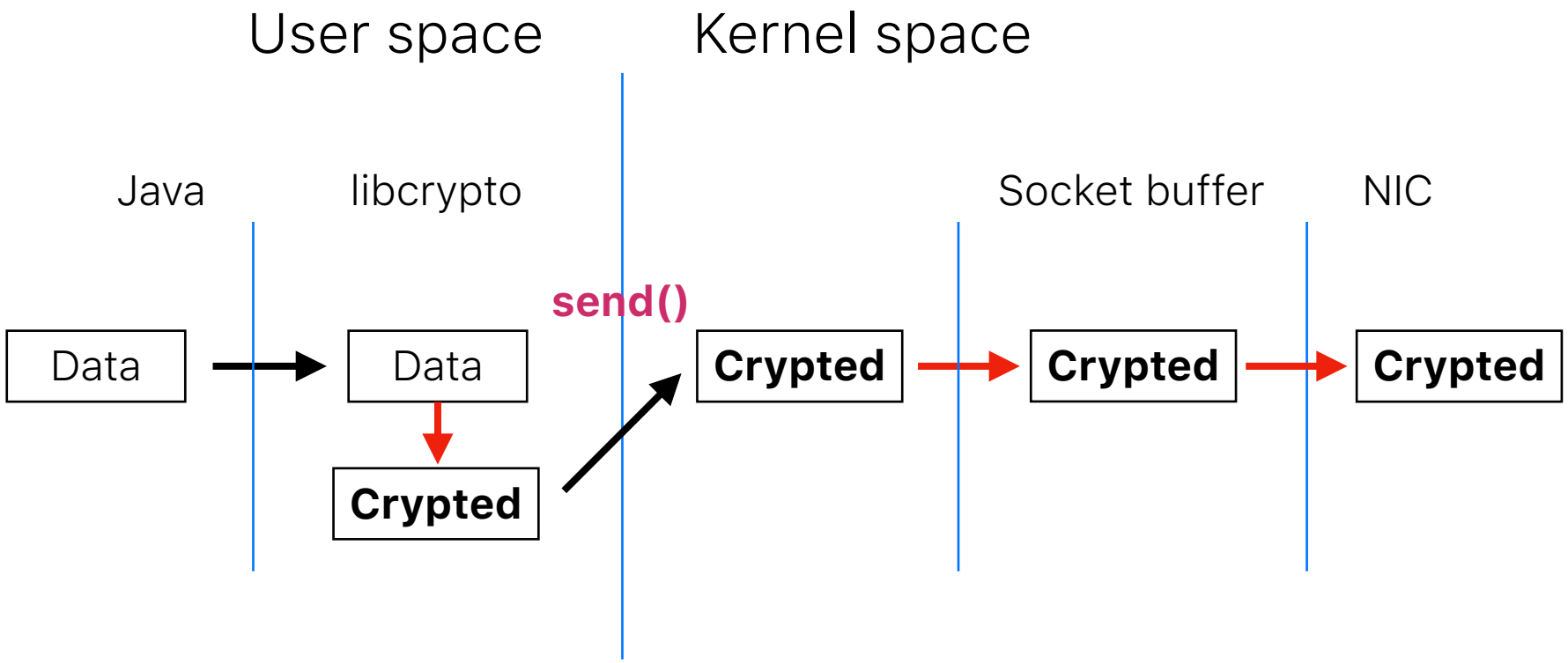
- Xeon Platinum 83xx
- 1 TB RAM
- 200 Gbps NIC
- NVME
- ядро 5.9

Kernel TLS (KTLS)

- Шифрование данных на уровне ядра или NIC
- Экономия сисколлов и копирования
- Реализация sendfile

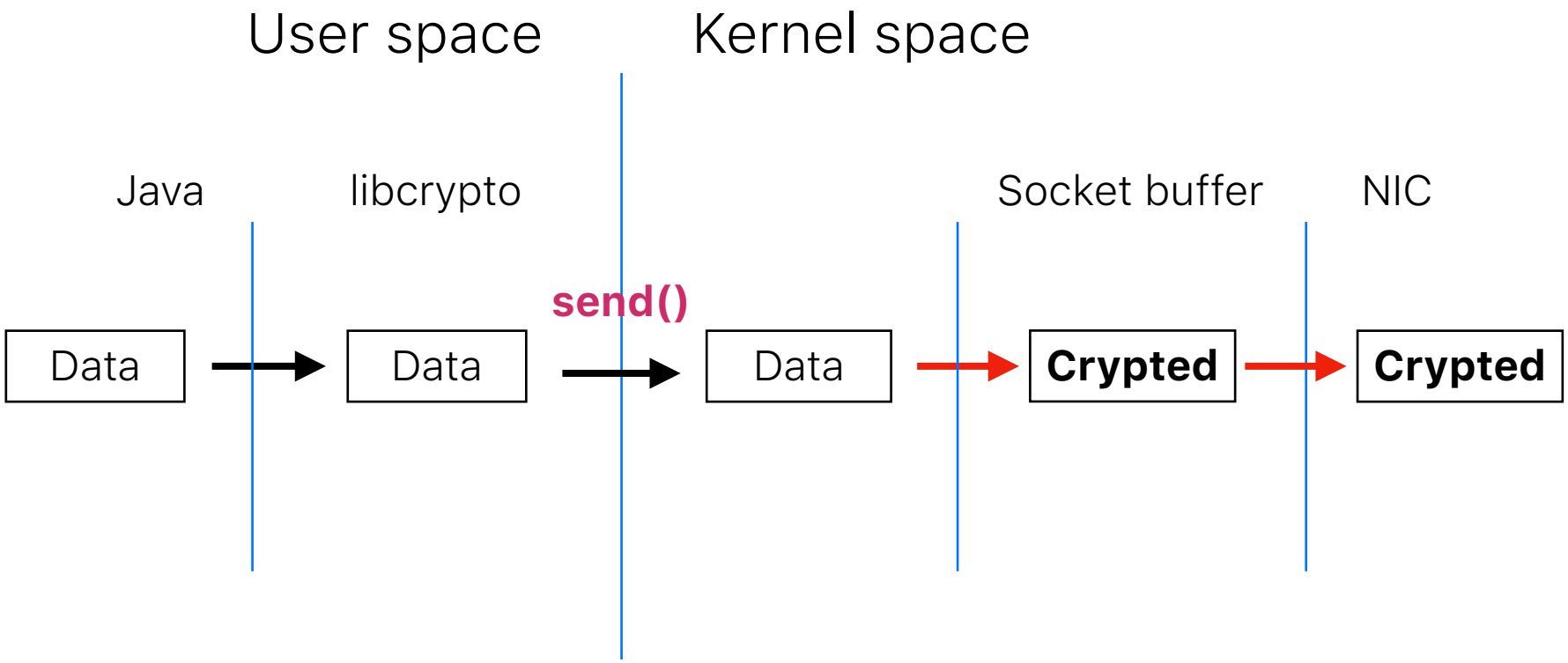
Без KTLS

- Три копирования
- Шифрование на cpu в user space



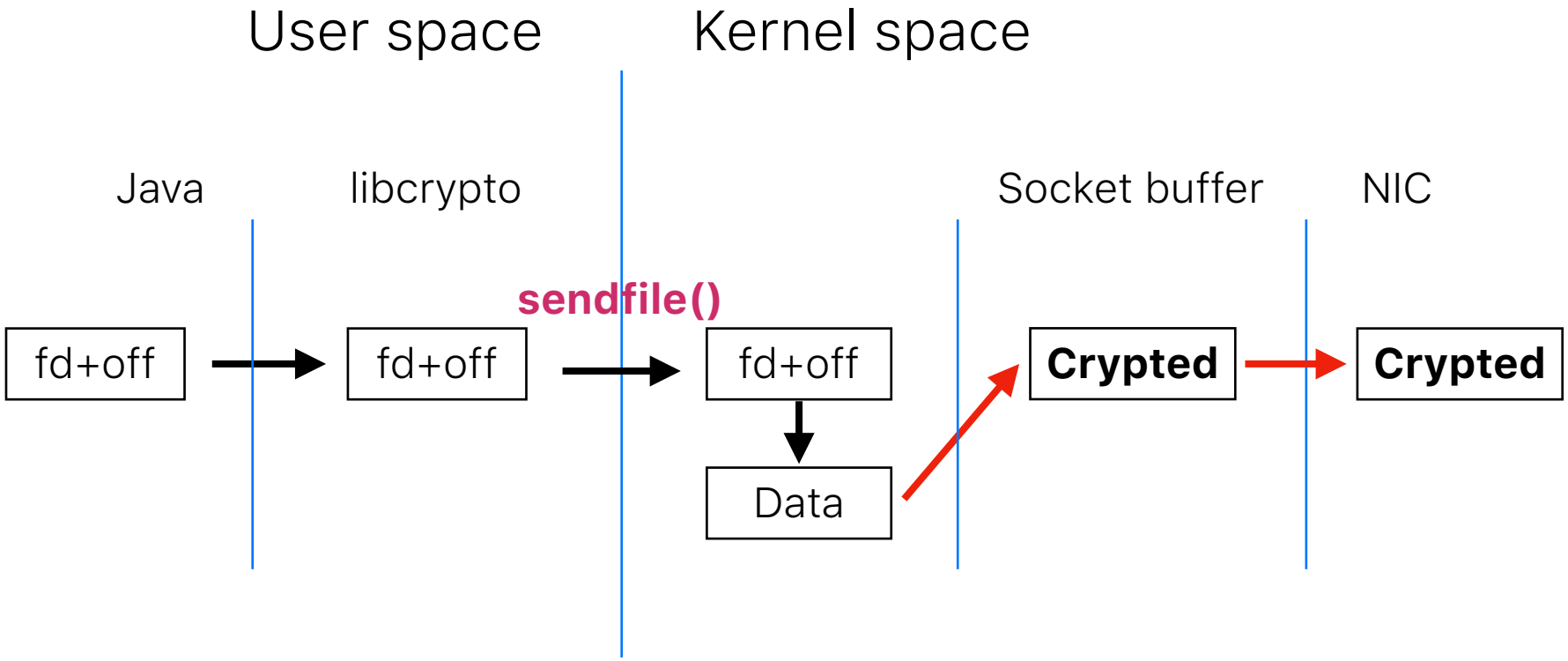
TLS_SW

- Два копирования
- Шифрование на cpu в kernel space



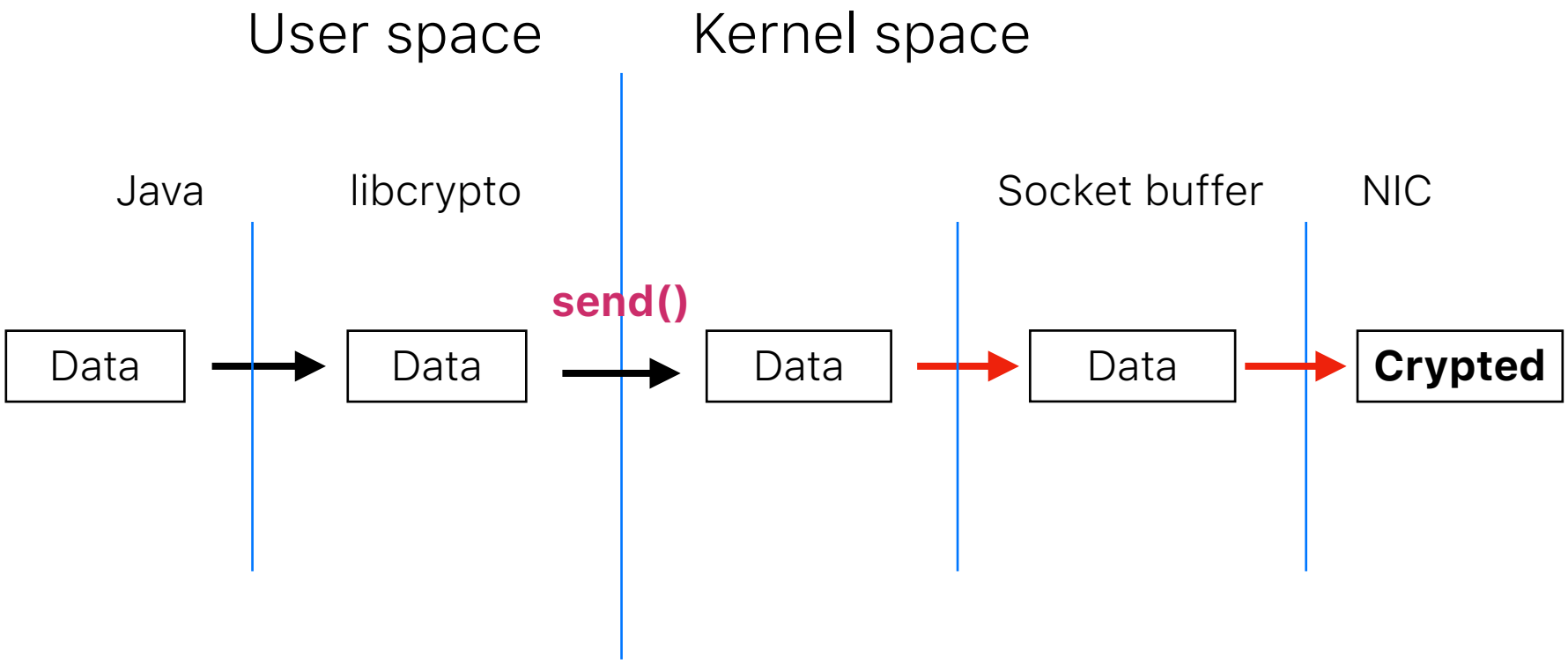
TLS_SW + sendfile

- Два копирования
- Шифрование на cpu в kernel space



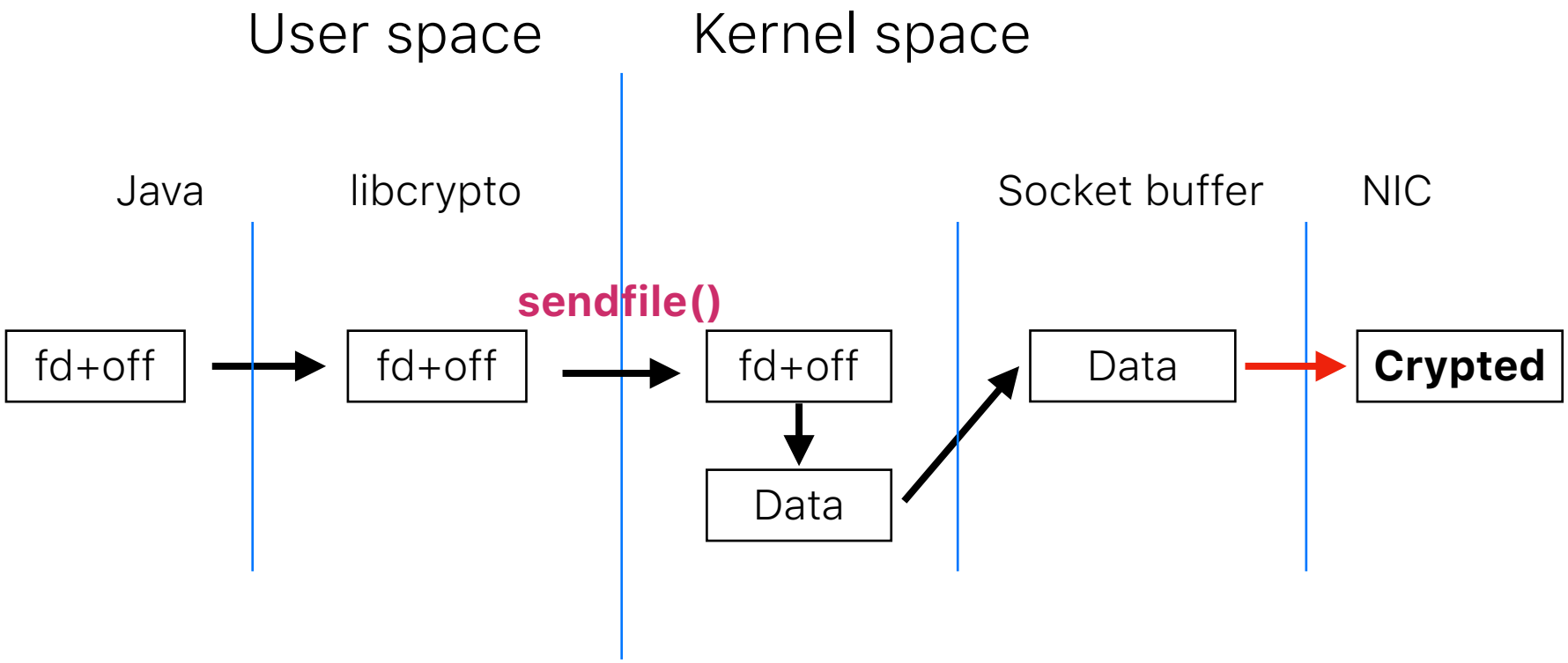
TLS_HW

- Два копирования
- Шифрование на специальном модуле



TLS_HW + sendfile

- Одно копирование
- Шифрование на специальном модуле



Как подключить

- Первая поддержка с версии ядра 4.13 (у нас 5.9 и 6.6)
  - Включить модуль ядра tls
- OpenSSL 3.0
  - Собрать с --enable-ktls
  - И на нужной версии ядра

Поддерживаемые протоколы

- https://delthas.fr/blog/2023/kernel-tls/

Only some encryption ciphers are supported by kernel TLS.

TLS version	Encryption cipher	Linux support	FreeBSD support
1.2	AES-GCM(128)	≥ 4.13	≥ 13.0
1.2	AES-GCM(256)	≥ 5.1	*?*
1.2	CHACHA20-POLY1305	≥ 5.11	*?*
1.2	ARIA-GCM(128)	≥ 6.1	*?*
1.2	ARIA-GCM(256)	≥ 6.1	*?*
1.3	AES-GCM(128)	≥ 5.1	*?*
1.3	AES-GCM(256)	≥ 5.1	*?*
1.3	CHACHA20-POLY1305	≥ 5.11	*?*
1.3	AES-CCM(128) *OpenSSL >= 3.2.0*	≥ 5.16	*?*

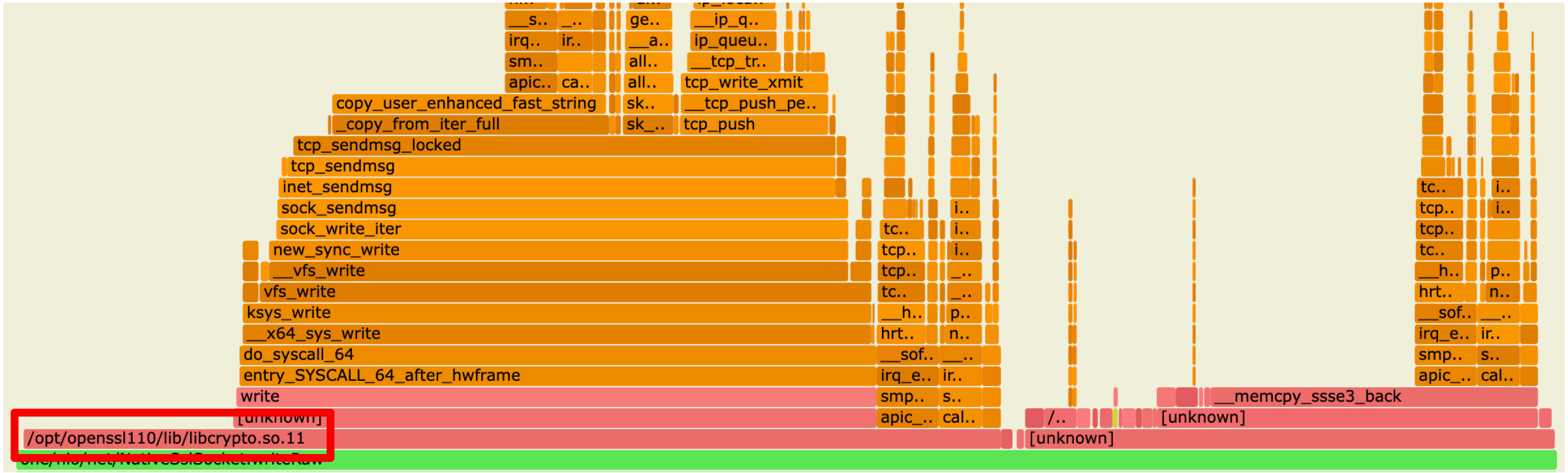
This translates to the following ciphers & ciphersuites.

Encryption cipher	TLS ≤ v1.2 ciphers	TLS ≥ v1.3 ciphersuites
AES-GCM(128)	ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 DHE-RSA-AES128-GCM-SHA256 AES128-GCM-SHA256	TLS_AES_128_GCM_SHA256
AES-GCM(256)	ECDHE-ECDSA-AES256-GCM-SHA384 ECDHE-RSA-AES256-GCM-SHA384 DHE-RSA-AES256-GCM-SHA384 AES256-GCM-SHA384	TLS_AES_256_GCM_SHA384
ARIA-GCM(128)	ECDHE-ECDSA-ARIA128-GCM-SHA256 ECDHE-ARIA128-GCM-SHA256 DHE-RSA-ARIA128-GCM-SHA256 ARIA128-GCM-SHA256	
ARIA-GCM(256)	ECDHE-ECDSA-ARIA256-GCM-SHA384 ECDHE-ARIA256-GCM-SHA384 DHE-RSA-ARIA256-GCM-SHA384 ARIA256-GCM-SHA384	
CHACHA20-POLY1305	ECDHE-ECDSA-CHACHA20-POLY1305 ECDHE-RSA-CHACHA20-POLY1305 DHE-RSA-CHACHA20-POLY1305	TLS_CHACHA20_POLY1305_SHA256
AES-CCM(128)		TLS_AES_128_CCM_SHA256

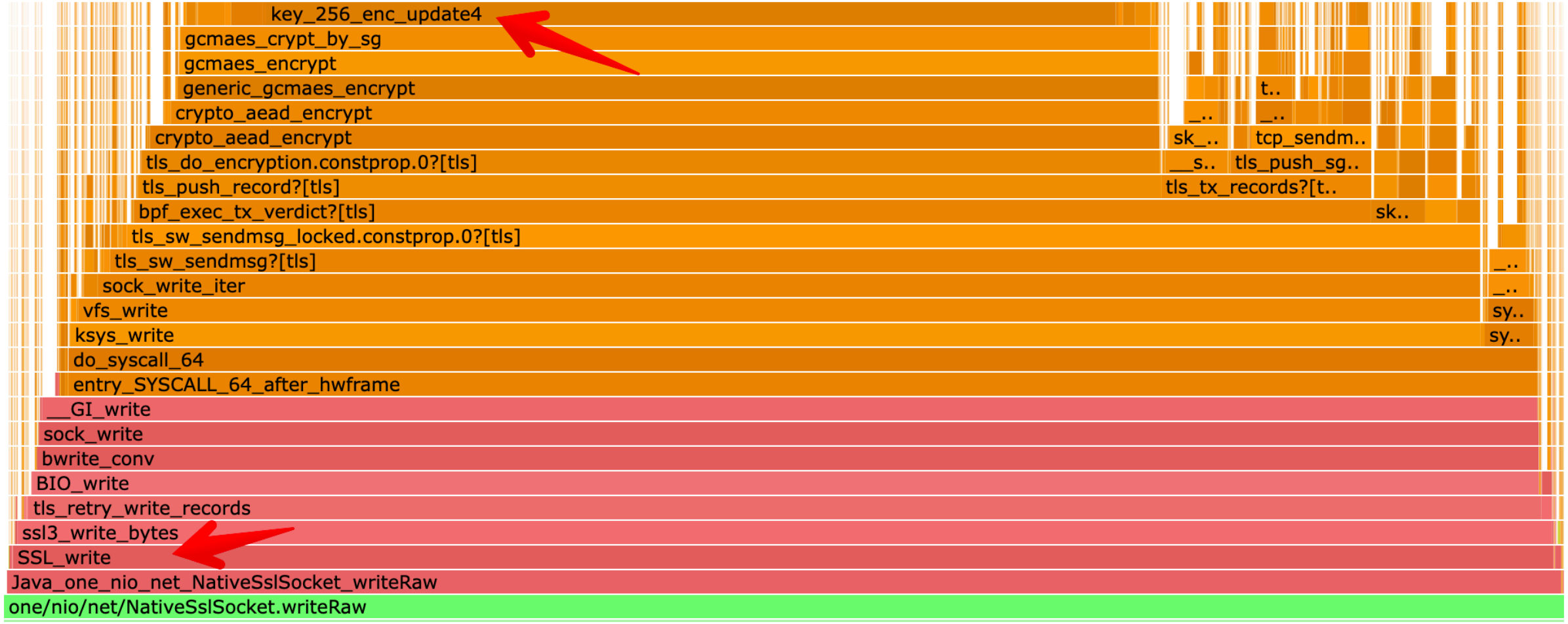
Как подключить

- SSL_CTX_set_options(ctx, SSL_OP_ENABLE_KTLS)
- SSL_sendfile(...)

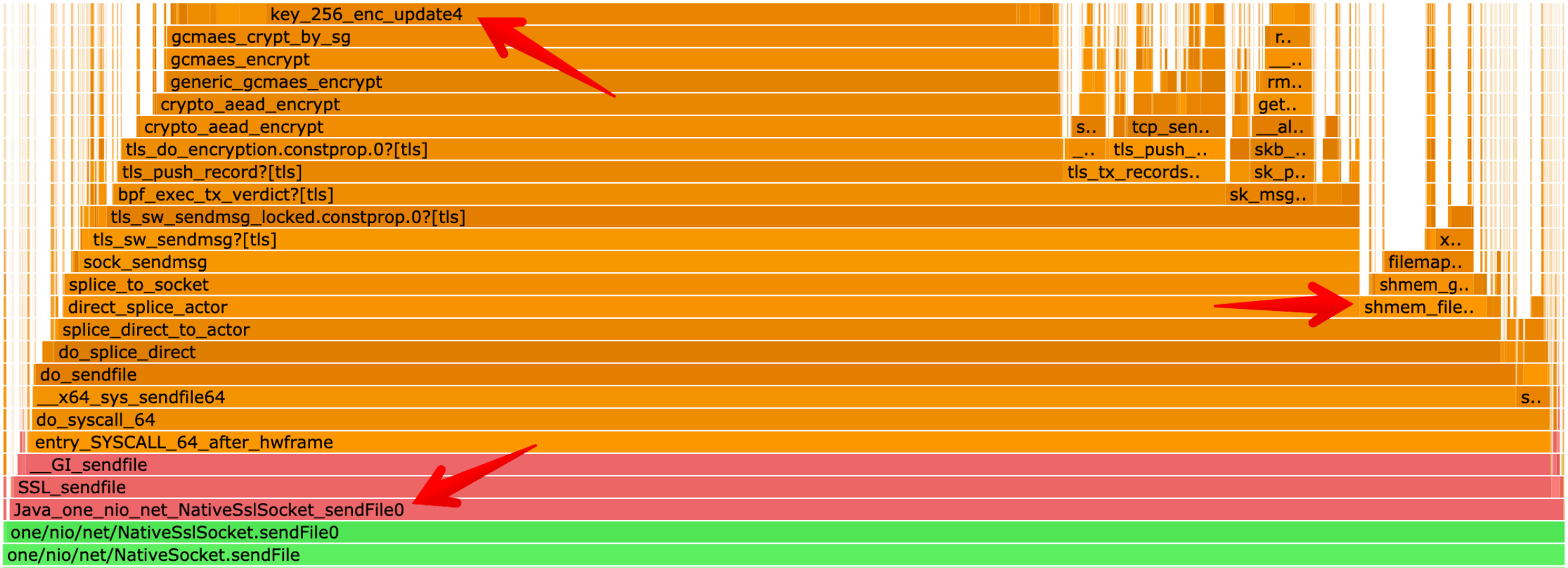
Без KTLS


__s..
_..
ge..
__ip_q..
irq..
ir..
__a..
ip_queu..
sm..
all..
__tcp_tr..
apic..
ca..
all..
tcp_write_xmit
copy_user_enhanced_fast_string
sk..
__tcp_push_pe..
_copy_from_iter_full
sk_..
tcp_push
tcp_sendmsg_locked
tcp_sendmsg
inet_sendmsg
sock_sendmsg
sock_write_iter
new_sync_write
__vfs_write
vfs_write
ksys_write
__x64_sys_write
do_syscall_64
entry_SYSCALL_64_after_hwframe
write
[unknown]
/opt/openssl110/lib/libcrypto.so.11
tc..
i..
tcp..
i..
tcp..
_..
tc..
_..
__h..
p..
hrt..
n..
__sof..
__..
irq_e..
ir..
smp..
s..
apic_..
cal..
/..
[unknown]
__memcpy_ssse3_back
[unknown]

Режим TLS_SW


key_256_enc_update4
gcmaes_crypt_by_sg
gcmaes_encrypt
generic_gcmaes_encrypt
crypto_aead_encrypt
crypto_aead_encrypt
tls_do_encryption.constprop.0?[tls]
tls_push_record?[tls]
bpf_exec_tx_verdict?[tls]
tls_sw_sendmsg_locked.constprop.0?[tls]
tls_sw_sendmsg?[tls]
sock_write_iter
vfs_write
ksys_write
do_syscall_64
entry_SYSCALL_64_after_hwframe
__GI_write
sock_write
bwrite_conv
BIO_write
tls_retry_write_records
ssl3_write_bytes
SSL_write
Java_one_nio_net_NativeSslSocket_writeRaw
one/nio/net/NativeSslSocket.writeRaw
t..
_..
_..
sk_..
tcp_sendm..
__s..
tls_push_sg..
tls_tx_records?[t..
sk..
_..
_..
sy..
sy..

Режим TLS_SW + sendfile



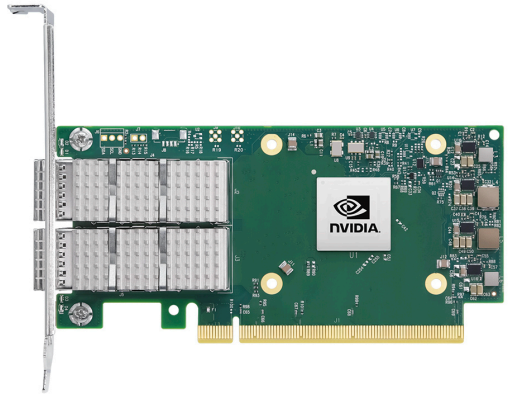
Режим TLS_HW

- Intel QuickAssist Adapter или Intel Xeon Scalable (64xx/84xx)
- NVIDIA ConnectX-6 Dx (с крипто-модулем)

Dual-Port 100GbE/Single-Port 200GbE SmartNIC

ConnectX-6 Dx SmartNIC is the industry's most secure and advanced cloud network interface card to accelerate mission-critical data-center applications, such as security, virtualization, SDN/NFV, big data, machine learning, and storage. The SmartNIC provides up to two ports of 100Gb/s or a single-port of 200Gb/s Ethernet connectivity and delivers the highest return on investment (ROI) of any smart network interface card. ConnectX-6 Dx is a member of NVIDIA's world-class, award-winning ConnectX series of network adapters powered by leading 50Gb/s (PAM4) and 25/10Gb/s (NRZ) SerDes technology and novel capabilities that accelerate cloud and data-center payloads.

LEARN MORE › ›



Highlights

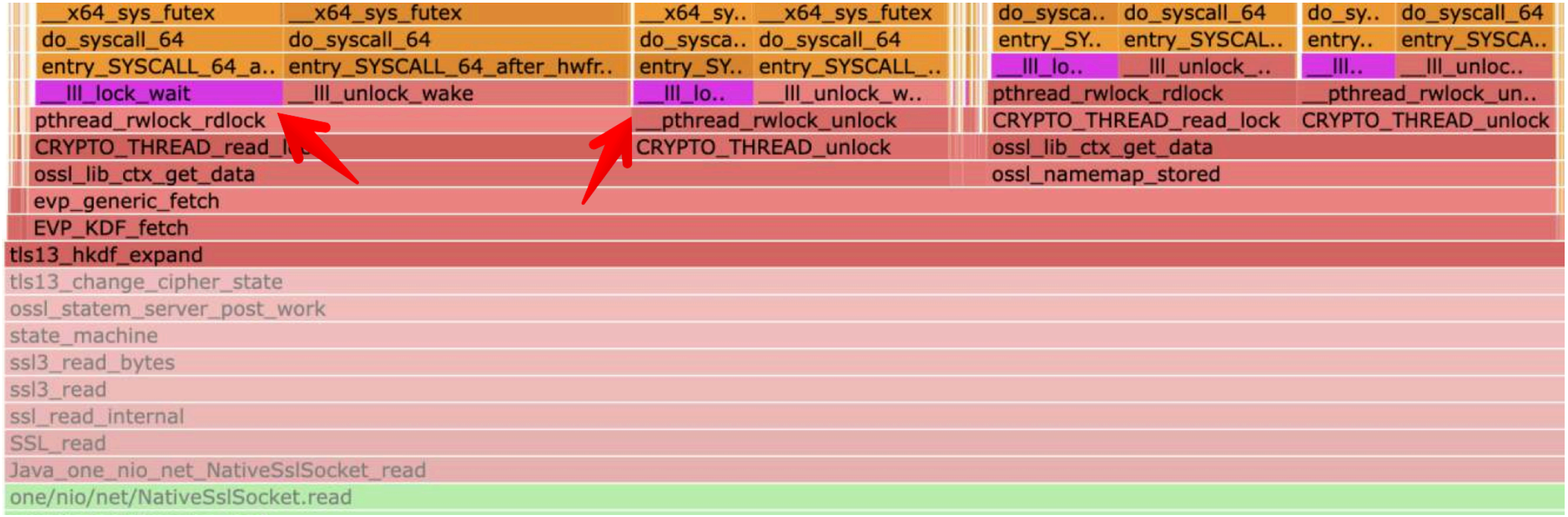
PORT SPEEDS	MESSAGE RATE (DPDK)	PCIe LANES	ENCRYPTION
2x100/1x200 Gb/s	**215** million msgs/sec	**16** Gen3/Gen4	**IPSec/TLS/AES-XTS**

Режим TLS_HW

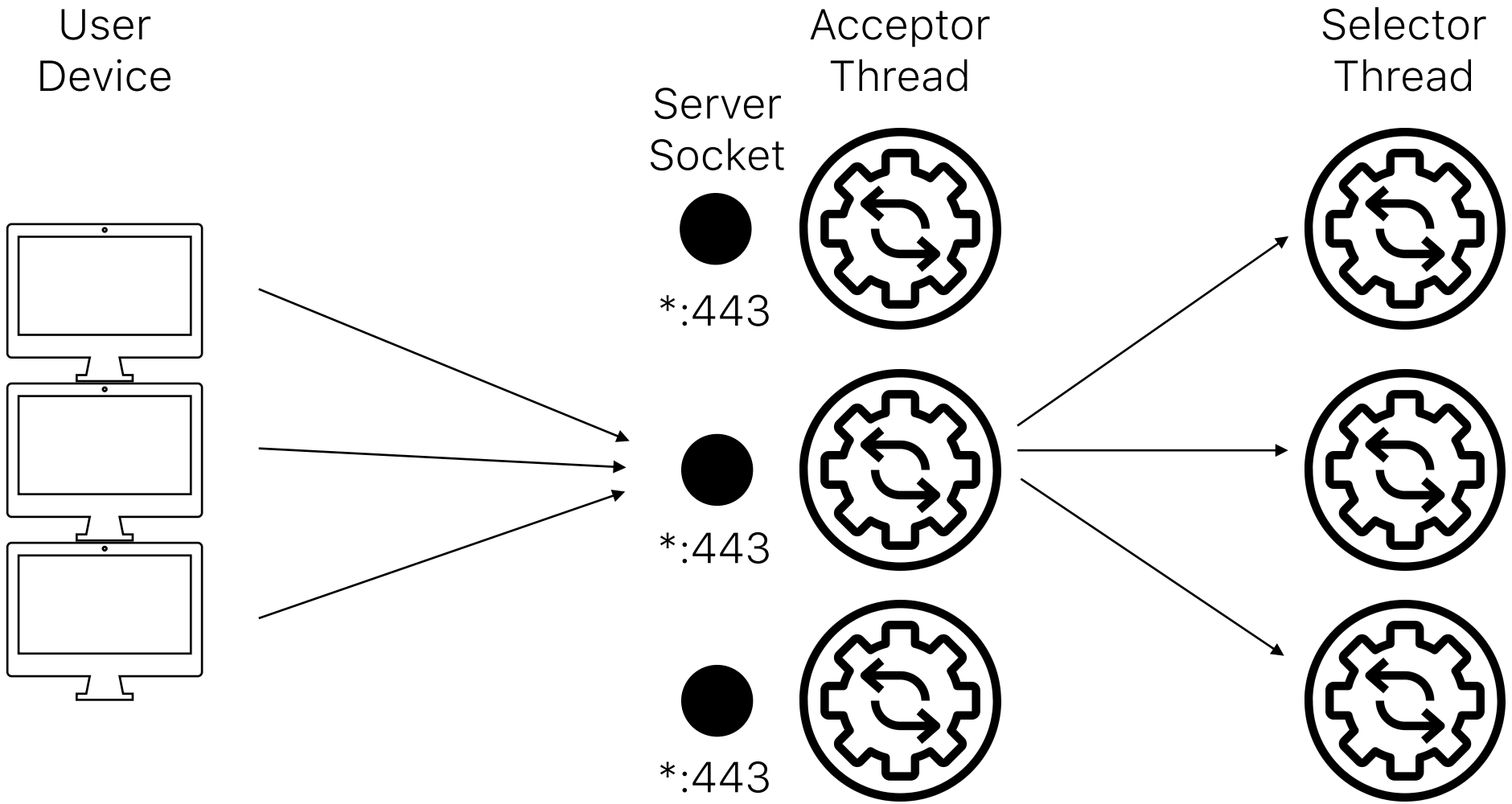
- NVIDIA ConnectX-6 Dx (с крипто-модулем)

SKU	OPN	Data Rate	Network Interface	Ports	PCIe	Technology	Crypto
900-9X6AG-0048-ST0	MCX623105AC-VDAT	200GbE	QSFP56	Single	PCIe 4.0 x16	Ethernet	Enabled
900-9X6AG-0016-ST0	MCX623105AN-CDAT	100GbE	QSFP56	Single	PCIe 4.0 x16	Ethernet	Disabled
900-9X6AG-0086-ST0	MCX623106AC-CDAT	100GbE	QSFP56	Dual	PCIe 4.0 x16	Ethernet	Enabled
900-9X6AG-0056-ST1	MCX623106AN-CDAT	100GbE	QSFP56	Dual	PCIe 4.0 x16	Ethernet	Disabled
900-9X6AG-0076-ST0	MCX623106AS-CDAT	100GbE	QSFP56	Dual	PCIe 4.0 x16	Ethernet	Disabled
900-9X658-0086-SB0	MCX623436AC-CDAB	100GbE	QSFP56	Dual	OCP 3.0 - PCIe 4.0 x16	Ethernet	Enabled
900-9X658-0056-SB1	MCX623436AN-CDAB	100GbE	QSFP56	Dual	OCP 3.0 - PCIe 4.0 x16	Ethernet	Disabled
900-9X6AG-0018-ST0	MCX623105AN-VDAT	200GbE	QSFP56	Single	PCIe 4.0 x16	Ethernet	Disabled
900-9X658-0046-SB0	MCX623435AC-CDAB	100GbE	QSFP56	Single	OCP 3.0 - PCIe 4.0 x16	Ethernet	Enabled
900-9X658-0016-SB0	MCX623435AN-CDAB	100GbE	QSFP56	Single	OCP 3.0 - PCIe 4.0 x16	Ethernet	Disabled
900-9X663-0083-SQ0	MCX621202AC-ADAT	25GbE	SFP28	Dual	PCIe 4.0 x8	Ethernet	Enabled
900-9X663-0073-SQ0	MCX621202AS-ADAT	25GbE	SFP28	Dual	PCIe 4.0 x8	Ethernet	Disabled

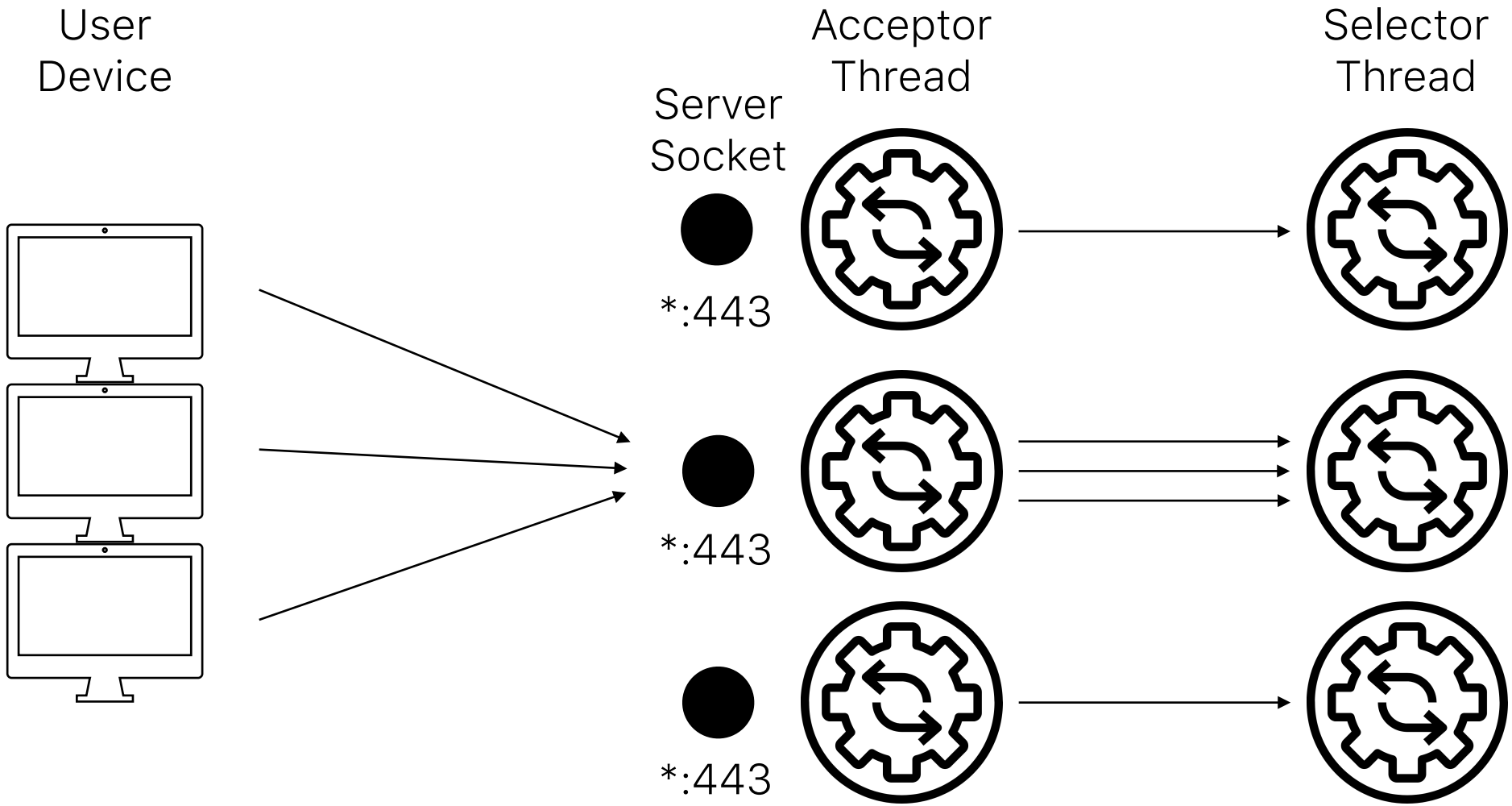
Чиним блокировки



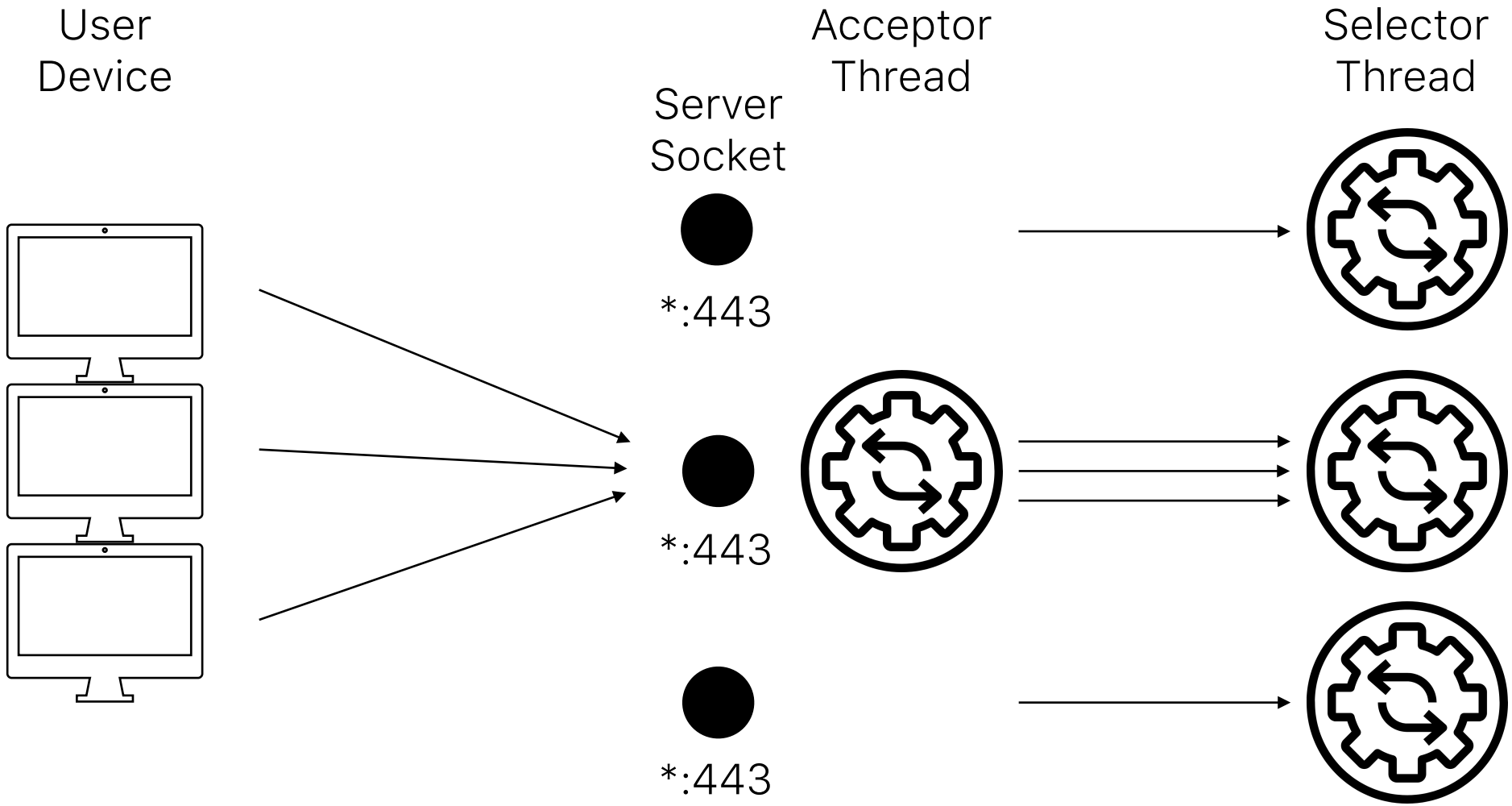
Как сейчас


User
Device
Server
Socket
Acceptor
Thread
Selector
Thread
*:443
*:443
*:443

Привязываем селекторы

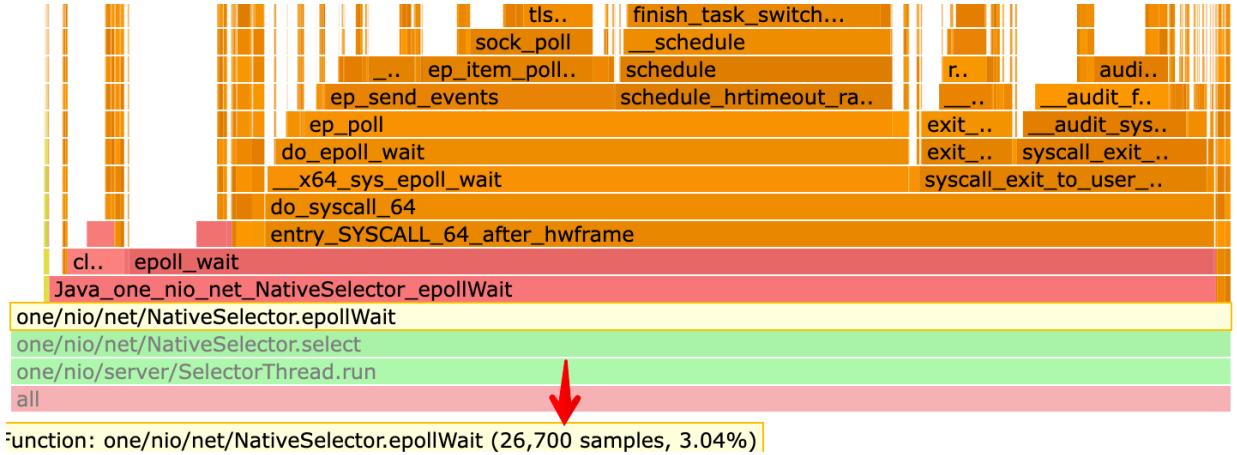


Делаем общий акцептор



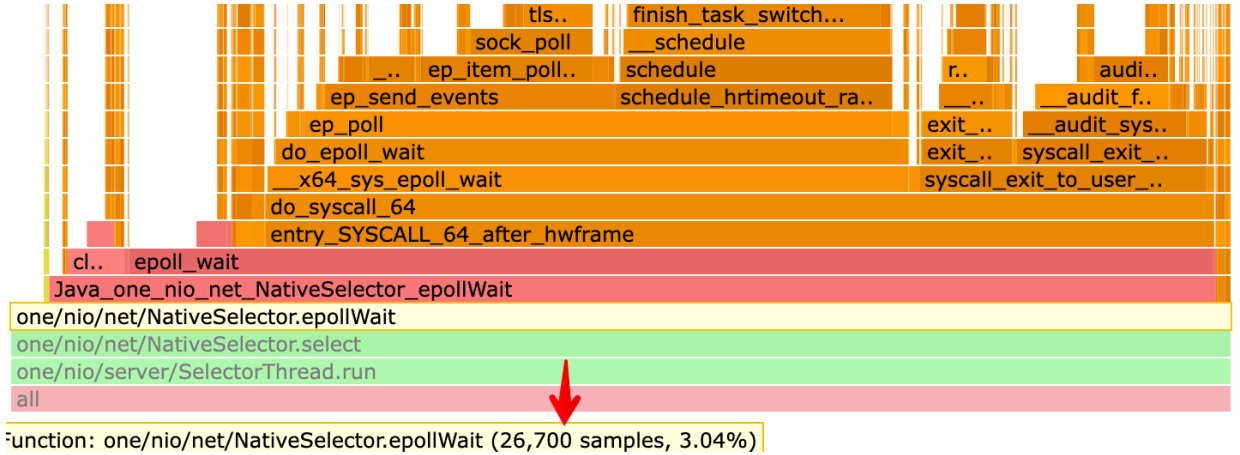
Рост утилизации cpu

- profiler -e cpu

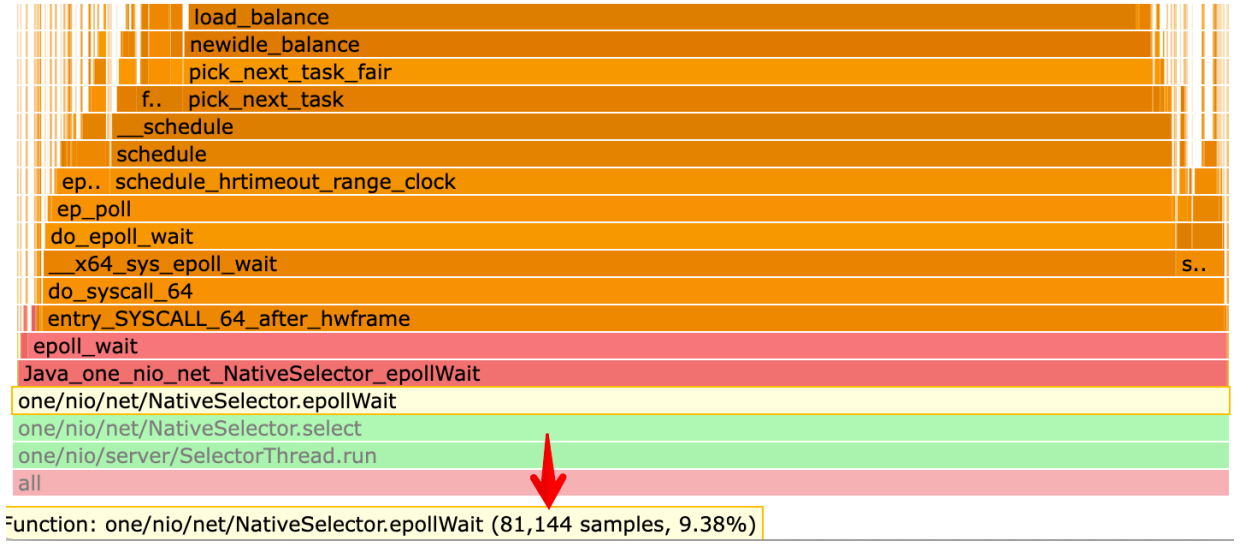


Рост утилизации cpu

- profiler -e cpu

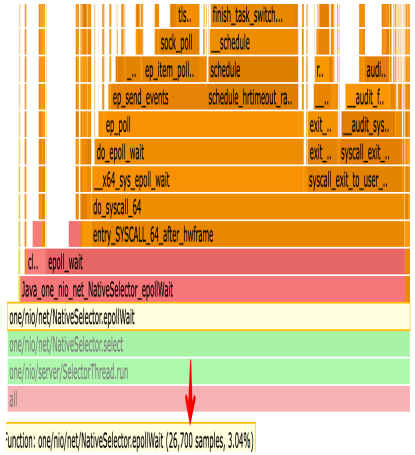


- profiler -e cycles

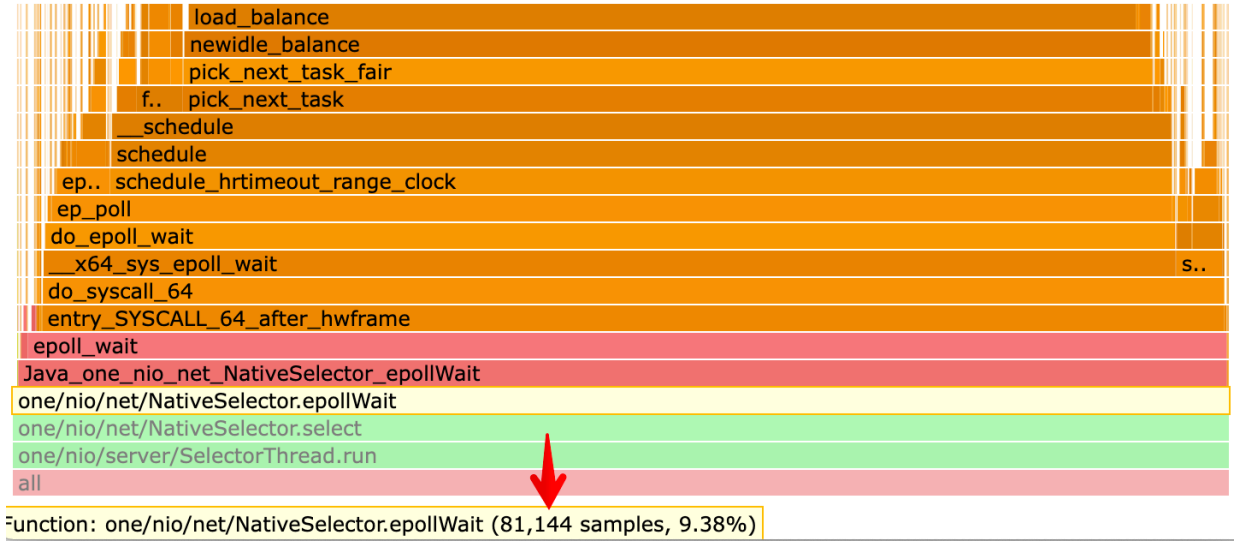


Рост утилизации cpu

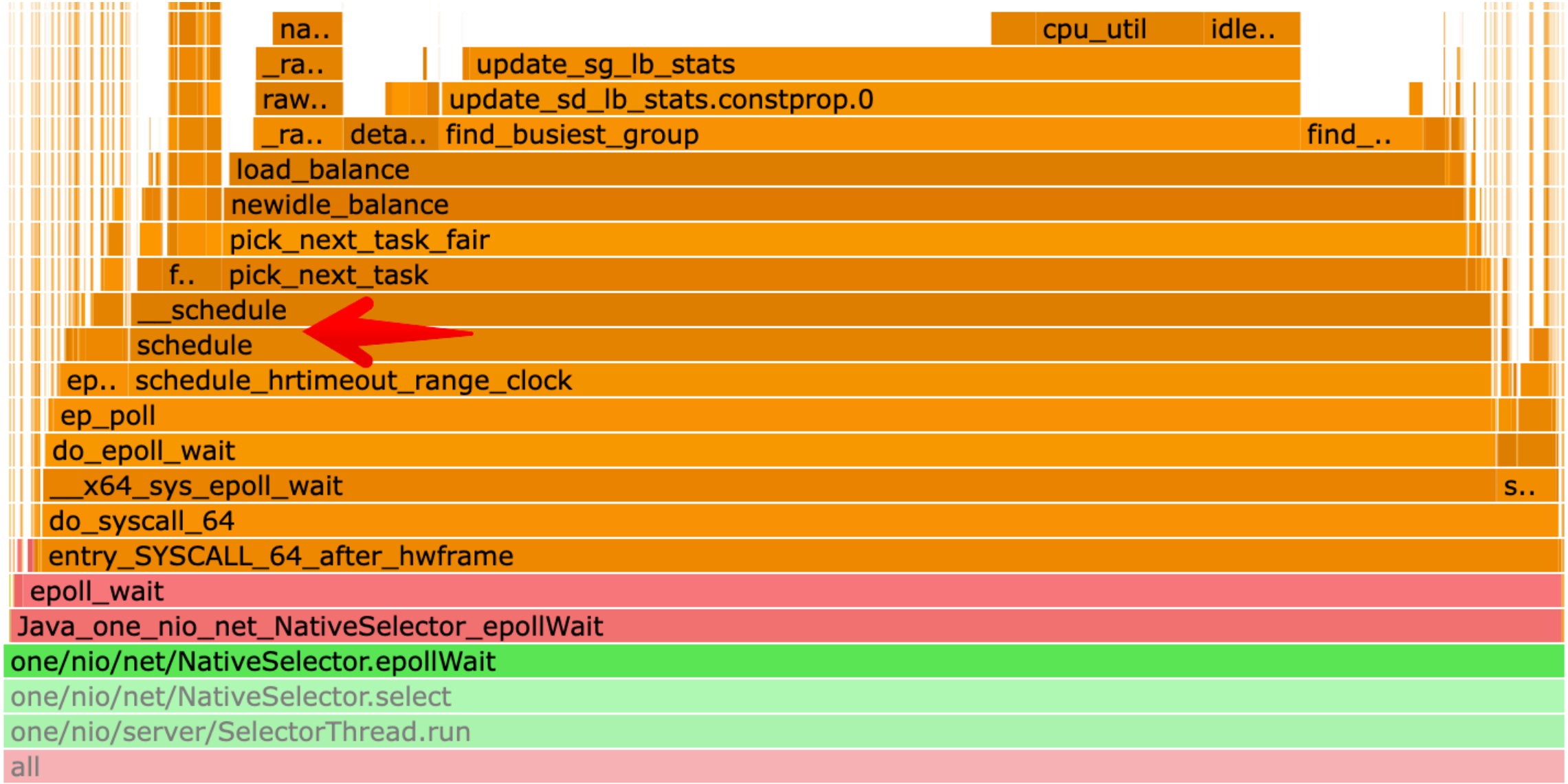
- profiler -e cpu



- profiler -e cycles



Cpu тратится в ожидании?



Cpu тратится в ожидании?

linux / kernel / sched / **fair.c**

Code | Blame | 13016 lines (10937 loc) · 346 KB | Code 55% faster with GitHub Copilo

```
11047    /*
11048     * Check this_cpu to ensure it is balanced within domain. Attempt to move
11049     * tasks if there is an imbalance.
11050     */
11051    static int load_balance(int this_cpu, struct rq *this_rq,
11052                            struct sched_domain *sd, enum cpu_idle_type idle,
11053                            int *continue_balancing)
11054    {
11055            int ld_moved, cur_ld_moved, active_balance = 0;
11056            struct sched_domain *sd_parent = sd->parent;
11057            struct sched_group *group;
11058            struct rq *busiest;
11059            struct rq_flags rf;
11060            struct cpumask *cpus = this_cpu_cpumask_var_ptr(load_balance_mask);
11061            struct lb_env env = {
```

Перераспределение прерываний

- An EEVDF CPU scheduler for Linux
  https://lwn.net/Articles/925371/

EEVDF Scheduler

The “Earliest Eligible Virtual Deadline First” (EEVDF) was first introduced in a scientific publication in 1995 [1]. The Linux kernel began transitioning to EEVDF in version 6.6 (as a new option in 2024), moving away from the earlier Completely Fair Scheduler (CFS) in favor of a version of EEVDF proposed by Peter Zijlstra in 2023 [2-4]. More information regarding CFS can be found in CFS Scheduler.

Ограничиваем миграцию прерываний

- Настройка cpuset.sched_relax_domain_level

As of this writing, when this capability was introduced in Linux 2.6.26, on certain popular architectures, the positive values of *sched_relax_domain_level* have the following meanings.

1 Perform immediate load balancing across Hyper-Thread siblings on the same core.

2 Perform immediate load balancing across other cores in the same package.

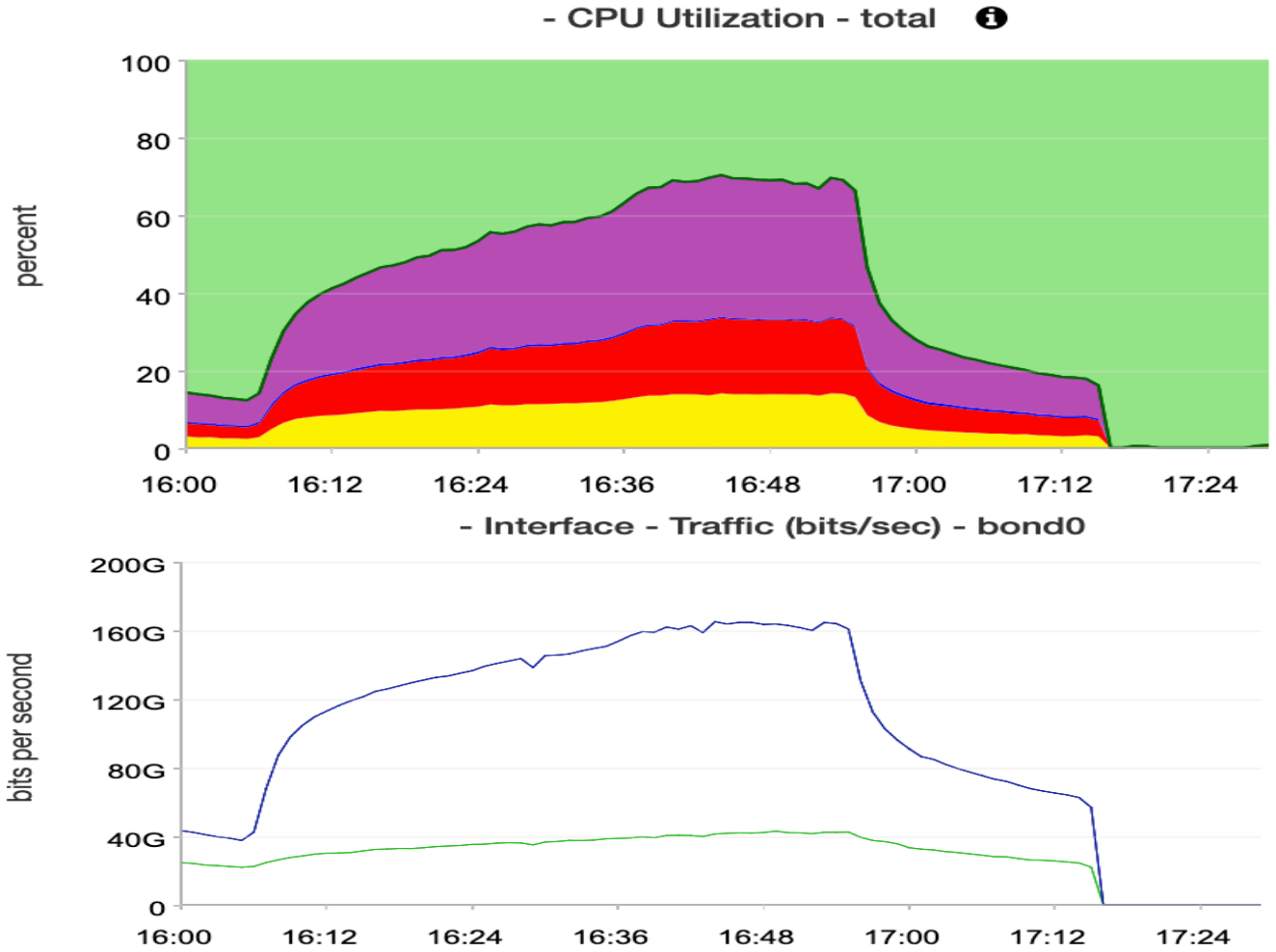
3 Perform immediate load balancing across other CPUs on the same node or blade.

4 Perform immediate load balancing across over several (implementation detail) nodes [On NUMA systems].

5 Perform immediate load balancing across over all CPUs in system [On NUMA systems].

Максимальный тест

- Держим до ~~100~~ 165 Gbps

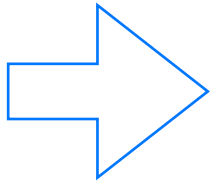


- Желтый — приложение (user)
- Красный — ядро (system)
- Фиолетовый — прерывания (softirq)
- Синий — диск (iowait)
- Зеленый — свободное (idle)

- Зеленый — к серверу (in)
- Синий — от сервера (out)

Обновляем конфиг

```
selectors: 88
affinity: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 4m
 - port: 443
   recvBuf: 64k
   sendBuf: 4m
   threads: 20
   reusePort: true
   notsentLowat: 512k
   ssl:
    - protocols: TLSv1 TLSv1.1 TLSv1.2
      certFile: ./file.cert
      privateKeyFile: ./file.key
      rdrand: true
```



```
selectors: 160
affinity: true
pinAcceptors: true
multiAcceptor: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 4m
 - port: 443
   recvBuf: 64k
   sendBuf: 4m
   threads: 160
   reusePort: true
   notsentLowat: 512k
   ssl:
    - protocols: TLSv1 TLSv1.1 TLSv1.2 TLSv1.3
      certFile: ./file.cert
      privateKeyFile: ./file.key
      rdrand: true
      ticketsDir: ./tickets
      kernelTlsEnabled: true
      maxEarlyDataSize: 4096
      antiReplayEnabled: false
      cacheMode: none
      compressionAlgorithms: brotli
```

Итоги

1. База и запуск
2. Внедряем HTTPS
3. Прокачиваем до 100G
4. Последние достижения

one-nio 2.0

- https://github.com/odnoklassniki/one-nio

Upgrade to OpenSSL 3 #85

Edit | Code

Merged **incubos** merged 1 commit into `odnoklassniki:master` from `Blooof:master`

Conversation 1 | Commits 1 | Checks 0 | Files changed 66 | +3,968 −385

Blooof commented • edited by incubos — Contributor

Breaking changes

- The native part of one-nio now links and works only with OpenSSL 3 (tested on OpenSSL versions 3.0 and 3.2). The OpenSSL library file must have name `libssl.so.3`.

Major changes

- Added support for Kernel TLS, including sendfile (https://docs.kernel.org/networking/tls-offload.html)
- Added support for TLS early data (TLS1.3 0-RTT, https://docs.openssl.org/3.2/man3/SSL_read_early_data/)
- Added ability to export encryption keys (SSL keylog, https://docs.openssl.org/3.2/man3/SSL_CTX_set_keylog_callback/)
- Added support for SSL certificate compression (RFC8879, https://docs.openssl.org/3.2/man3/SSL_CTX_set1_cert_comp_preference/)
- Added ability to use an external cache for SSL sessions (https://docs.openssl.org/3.2/man3/SSL_CTX_sess_set_get_cb/)
- Added ability to constrain selectors used by an acceptor (`one.nio.server.ServerConfig#pinAcceptors`)
- Added ability to use a single acceptor thread for all sockets (`one.nio.server.ServerConfig#multiAcceptor`)
- Added support for server-sent events in `one.nio.http.HttpClient` (https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events)

Also various fixes and optimizations.

Big thanks to:

Reviewers: incubos ✓

Assignees: incubos

Labels: None yet

Projects: None yet

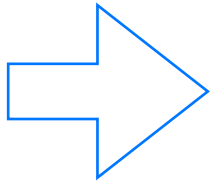
Milestone: No milestone

Development: Successfully merging this pull request may close these issues.

None yet

Наш путь

```
selectors: 24

acceptors:
 - port: 80
```



```
selectors: 160
affinity: true
pinAcceptors: true
multiAcceptor: true

acceptors:
 - port: 80
   recvBuf: 64k
   sendBuf: 4m
 - port: 443
   recvBuf: 64k
   sendBuf: 4m
   threads: 160
   reusePort: true
   notsentLowat: 512k
   ssl:
    - protocols: TLSv1 TLSv1.1 TLSv1.2 TLSv1.3
      certFile: ./file.cert
      privateKeyFile: ./file.key
      rdrand: true
      ticketsDir: ./tickets
      kernelTlsEnabled: true
      maxEarlyDataSize: 4096
      antiReplayEnabled: false
      cacheMode: none
      compressionAlgorithms: brotli
```

Выводы

- Приложение не живет в вакууме

Выводы

- Приложение не живет в вакууме
- При вертикальном масштабировании могут быть проблемы

Выводы

- Приложение не живет в вакууме
- При вертикальном масштабировании могут быть проблемы
- Нет серебряной пули, есть методичный поиск проблем

Выводы

- Приложение не живет в вакууме
- При вертикальном масштабировании могут быть проблемы
- Нет серебряной пули, есть методичный поиск проблем
- Обращайтесь за помощью к коллегам и учитесь сами

Выводы

- Приложение не живет в вакууме
- При вертикальном масштабировании могут быть проблемы
- Нет серебряной пули, есть методичный поиск проблем
- Обращайтесь за помощью к коллегам и учитесь сами
- Делайте доклады!

Спасибо

Олег Ларионов, руководитель группы разработки

Прошлые доклады

- Особенности разработки высоконагруженного сервера на Java https://youtu.be/gIh0X-RkftY
- Кеширование данных вне Java Heap https://youtu.be/h9AICWfaEVQ
- Дешевле, надежнее, проще https://youtu.be/uuGbbJhS7o8
- Платформа для Видео сроком в квартал https://youtu.be/2bDZj9_hpuI
- Кадры решают всё https://youtu.be/my6h8dLvGks
- Тюним память и сетевой стек https://youtu.be/iyTiMVTUhNM
- JVM профайлер с чувством такта https://youtu.be/WoomehXd4NY
- Safepoint — и пусть весь мир подождет https://youtu.be/rthWVvU9gWo