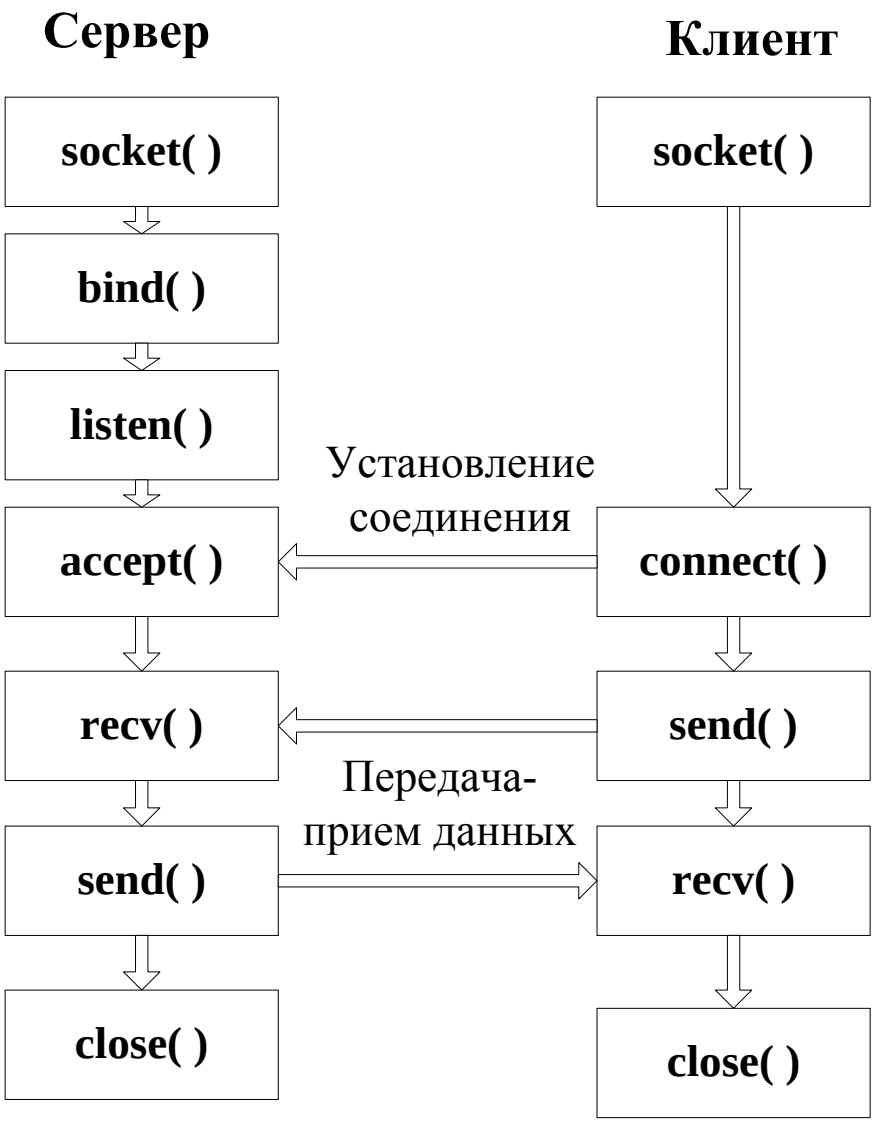
Программа типа клиент-сервер для TCP



Применение процессов для обеспечения параллельной работы сервера

```
signal( SIGCHLD, reaper ) ;
    for( ; ; ) {  ….
     }
void reaper( int sig ){   int status;
 while( wait3( &status, WNOHANG,
(struct rusage *) 0 )  >=  0 ) ;
```

Родительский процесс

```
...
for(;;){
sn=accept(s,….);
f=fork();
If(f==0){
…
}
If(f>0){
Close(sn);
... }
...}
```

Можно узнать о наличии зомби-процесса командой: `>ps –aux`

Child for client1

```
…
for(;;){
sn=accept(s,….);
f=fork();
If(f==0){
Close(s);
Send(sn,…);
Recv(sn,…);
…
Close(sn);
Exit(0);
}
If(f>0){
}
… }
```

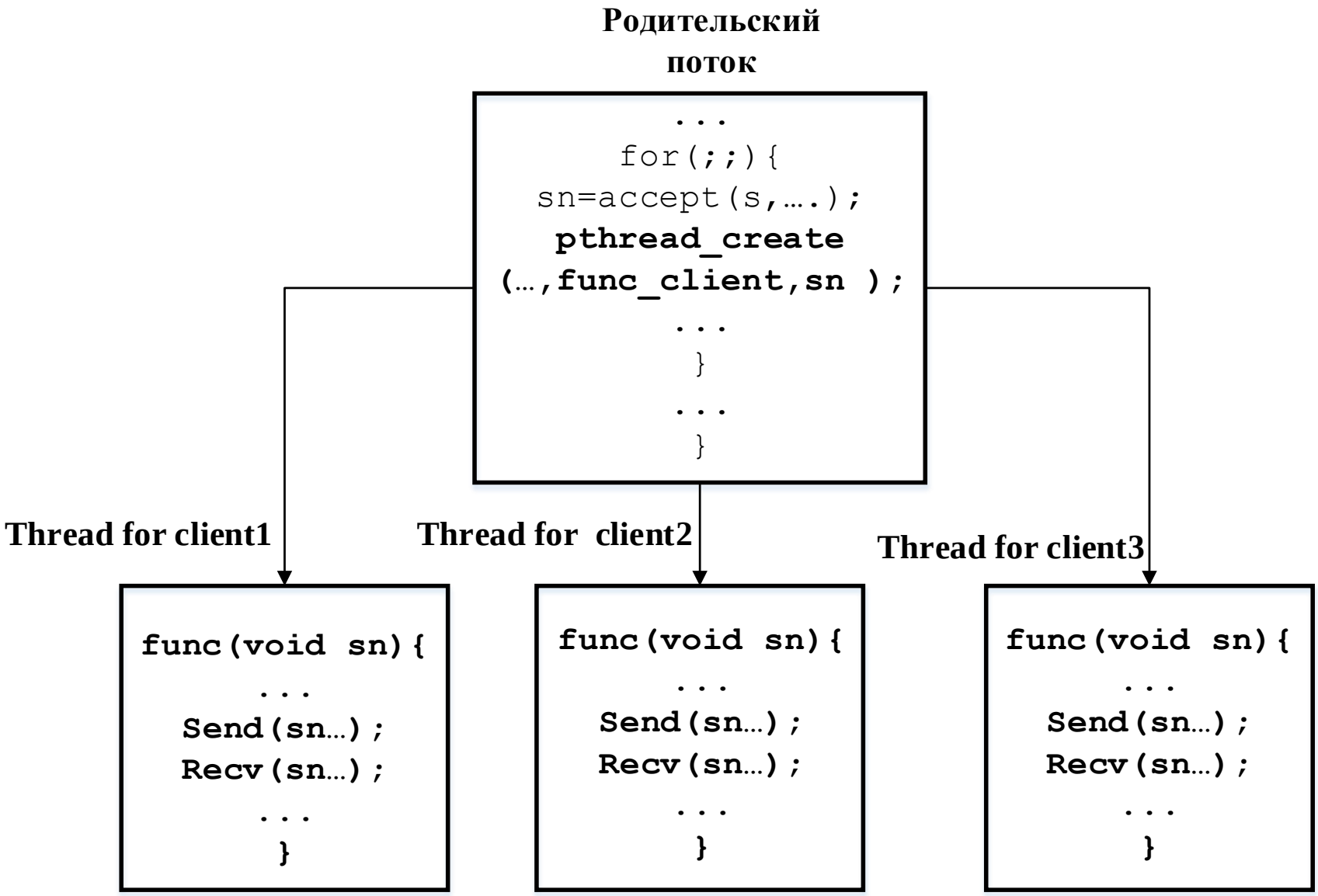
Child for client2

```
…
for(;;){
sn=accept(s,….);
f=fork();
If(f==0){
Close(s);
Send(sn,…);
Recv(sn,…);
…
Close(sn);
Exit(0);
}
If(f>0){
}
… }
```

Child for client3

```
…
for(;;){
sn=accept(s,….);
f=fork();
If(f==0){
Close(s);
Send(sn,…);
Recv(sn,…);
…
Close(sn);
Exit(0);
}
If(f>0){
}
… }
```

Применение потоков для обеспечения параллельной работы сервера



Протокол POP3 (RFC-1939)

- Перед работой через протокол **POP3** сервер прослушивает **порт 110**. Когда клиент хочет использовать этот протокол, он должен создать TCP соединение с сервером. Когда соединение установлено, сервер отправляет приглашение. Затем клиент и POP3 сервер обмениваются информацией пока соединение не будет закрыто или прервано.

- **Команды POP3 состоят из ключевых слов**, за некоторыми следует один или более аргументов. Все команды заканчиваются парой CRLF. Ключевые слова и аргументы состоят из печатаемых ASCII символов. Ключевое слово и аргументы разделены одиночным пробелом. **Ключевое слово состоит от 3-х до 4-х символов**, а аргумент может быть длиной до 40-ка символов.

- Ответы в POP3 состоят из индикатора состояния и ключевого слова, за которым может следовать дополнительная информация. Ответ заканчивается парой CRLF. Существует только два индикатора состояния: "**+OK**" - положительный и "**-ERR**" - отрицательный.
- Ответы на некоторые команды могут состоять из нескольких строк. В этих случаях каждая строка разделена парой CRLF, а конец ответа заканчивается ASCII символом 46 (".") и парой CRLF.
- POP3 сессия состоит из нескольких режимов. Как только соединение с сервером было установлено и сервер отправил приглашение, то сессия переходит в режим **AUTHORIZATION (Авторизация)**. После успешной идентификации сессия переходит в режим **TRANSACTION (Передача).** В этом режиме клиент запрашивает сервер выполнить определённые команды. Когда клиент отправляет команду QUIT, сессия переходит в режим UPDATE. В этом режиме POP3 сервер освобождает все занятые ресурсы и завершает работу. После этого TCP соединение закрывается.

Протокол POP3. Авторизация

Примеры

- C: USER Bob
- S: -ERR sorry, no mailbox for frated here

или

- C: USER Bob
- S: +OK Bob is a real hoopy frood..

- C: PASS mypass
- S: +OK Bob's maildrop has 3 messages (350 octets)

или

- C: PASS mymail
- S: -ERR maildrop already locked

- C: QUIT: +OK dewey POP3 server signing off

Протокол POP3. Транзакция

Примеры

C: **STAT**

S: +OK 2 350

Описание: В ответ на вызов команды сервер выдаёт положительный ответ "+OK", за которым следует количество сообщений в почтовом ящике и их общий размер в символах. Сообщения, которые помечены для удаления, не учитываются в ответе сервера.

****

C: **LIST**

S: +OK 2 messages (350 octets)

S: 1 120

S: 2 230

C: **LIST 2**

S: +OK 2 230

C: **LIST 3**

S: -ERR no such message, only 2 messages in maildrop

Описание: Если был передан аргумент, то сервер выдаёт информацию об указанном сообщении. Если аргумент не был передан, то сервер выдаёт информацию о всех сообщениях, находящихся в почтовом ящике. Сообщения, помеченные для удаления не перечисляются.

****

C: **RETR** 1

S: +OK 120 octets

S:

S: .

Описание: После положительного ответа сервер передаёт содержание сообщения

Протокол POP3. Транзакция

Примеры

C: **DELE** 1

S: +OK message 1 deleted...

C: **DELE** 1

S: -ERR message 1 already deleted

Описание: POP3 сервер помечает указанное сообщение как удалённое, но не удаляет его, пока сессия не перейдёт в режим UPDATE

****

C: **NOOP**

S: +OK

Описание: POP3 сервер ничего не делает и всегда отвечает положительно

****

C: **RSET**

S: +OK maildrop has 2 messages (350 octets)

Описание: Если какие - то сообщения были помечены для удаления, то с них снимается эта метка

****

Обновление

Когда клиент передаёт команду **QUIT** в режиме TRANSACTION, то сессия переходит в режим UPDATE. В этом режиме сервер удаляет все сообщения, помеченные для удаления. После этого TCP соединение закрывается.

Протокол POP3.
Дополнительные POP3 команды

Примеры:

C: **TOP** 1 10

S: +OK

S: *<здесь POP3 сервер передаёт заголовки первого сообщения и первые 10-ть строк из тела сообщения.>*

S: . ...

C**: TOP** 100 3

S: -ERR no such message

Описание. Команда: **TOP** [сообщение] [n] Аргументы: [сообщение] - номер сообщения [n] - положительное число (обязательный аргумент). Если ответ сервера положительный, то после него он передаёт заголовки сообщения и указанное количество строк из тела сообщения. Возможные ответы: +OK top of message follows –ERR no such message.

Протокол POP3. Пример сессии

S: <создаём новое TCP соединение с POP3 сервером через порт 110>
S: +OK POP3 server ready
C: **USER** Bob
S: +OK User Bob is exists
C: **PASS** mymail
S: +OK Bob's maildrop has 2 messages (350 octets)
C: **STAT**
S: +OK 2 350
C: **LIST**
S: +OK 2 messages (350 octets)
S: 1 120
S: 2 230
S: .
C: **RETR** 1
S: +OK 120 octets
S: *<После положительного ответа сервер передаёт содержание сообщения>*
S: .
C: **DELE** 1
S: +OK message 1 deleted
C: **QUIT**
S: +OK dewey POP3 server signing off (maildrop empty)
C: <закрываем соединение>

Протокол SMTP (RFC-5321)

- **Основная задача** протокола **SMTP** (Simple Mail Transfer Protocol) заключается в том, чтобы **обеспечивать передачу электронных сообщений** (почту). Для работы через протокол SMTP клиент создаёт TCP соединение с сервером через **порт 25**. Затем клиент и SMTP сервер обмениваются информацией пока соединение не будет закрыто или прервано. **Основной процедурой в SMTP является передача почты.** Далее идут процедуры перенаправления почты, проверка имён почтового ящика и вывод списков почтовых групп. Самой первой процедурой является открытие канала передачи, а последней - его закрытие.
- **Команды SMTP** указывают серверу, какую операцию хочет произвести клиент. Команды состоят из ключевых слов, за которыми следует один или более параметров. Ключевое слово состоит **из 4-х символов** и разделено от аргумента одним или несколькими пробелами. Каждая командная строка заканчивается символами **CRLF**.

Протокол SMTP. Пример сессии

- **Отправка почты**
- Первым делом **подключаемся к SMTP серверу через порт 25**. Теперь надо передать серверу команду HELO и наш IP адрес:
- C: HELO 195.161.101.33
- S: 250 smtp.mail.ru is ready *При отправке почты передаём некоторые нужные данные (отправитель, получатель и само письмо*):
- C: MAIL FROM:<frol> *указываем отправителя*
- S: 250 OK
- C: RCPT TO:<bob@mail.ru> *указываем получателя*
- S: 250 OK *указываем серверу, что будем передавать содержание письма (заголовок и тело письма)*
- C: DATA
- S: 354 Start mail input; end with <CRLF>.<CRLF> *передачу письма необходимо завершить символами CRLF.CRLF*
- S: 250 OK
- C: From: Frol frol@mail.ru
- C: To: Bob bob@mail.ru
- C: Subject: Hello *между заголовком письма и его текстом не одна пара CRLF, а две.*
- C: Hello Bob!
- C: You will be happy on next week! *заканчиваем передачу символами CRLF.CRLF*
- S: 250 OK *Теперь завершаем работу, отправляем команду QUIT:*
- S: QUIT
- C: 221 smtp.mail.ru is closing transmission channel

Протокол SMTP

Другие команды

- **SEND** - используется вместо команды MAIL и указывает, что почта должна быть доставлена на терминал пользователя.
- **SOML, SAML** - комбинации команд SEND или MAIL, SEND и MAIL соответственно.
- **RSET** - указывает серверу прервать выполнение текущего процесса. Все сохранённые данные (отправитель, получатель и др.) удаляются. Сервер должен отправить положительный ответ.
- **VRFY** - просит сервер проверить, является ли переданный аргумент именем пользователя. В случае успеха сервер возвращает полное имя пользователя.
- **EXPN** - просит сервер подтвердить, что переданный аргумент - это список почтовой группы, и если так, то сервер выводит членов этой группы.
- **HELP** - запрашивает у сервера полезную помощь о переданной в качестве аргумента команде.
- **NOOP** - на вызов этой команды сервер должен положительно ответить. NOOP ничего не делает и никак не влияет на указанные до этого данные.

Протокол TFTP

Простейший протокол пересылки файлов (**Trivial File Transfer Protocol — TFTP**) используется как очень полезное средство копирования файлов между компьютерами. TFTP передает данные **в датаграммах UDP** (при реализации в другом стеке протоколов TFTP должен запускаться поверх службы пакетной доставки данных). Для этого не потребуется слишком сложное программное обеспечение — достаточно только IP и UDP. Особенно полезен TFTP для инициализации сетевых устройств (маршрутизаторов, мостов или концентраторов). Протокол FTP определен в RFC 959, а TFTP - в RFC 1350.

Сеанс TFTP начинается запросами ***Read Request*** (запрос чтения) или ***Write Request*** (запрос записи). Клиент TFTP начинает работу после получения порта, посылая *Read Request* или *Write Request* на **порт 69** сервера.

Один из партнеров по TFTP пересылает нумерованные блоки данных одинакового размера, другой партнер подтверждает их прибытие сигналом АСК Отправитель ожидает АСК для посланного блока до того, как пошлет следующий блок.

Характеристики TFTP

- Пересылка блоков данных размером в 512 октетов (за исключением последнего блока).
- Указание для каждого блока простого 4-октетного заголовка.
- Нумерация блоков от 1.
- Поддержка пересылки двоичных и ASCII октетов.
- Возможность чтения и записи удаленных файлов.
- Отсутствие ограничений по аутентификации пользователей.

Протокол TFTP (RFC-1350)

Каждый блок (за исключением последнего) должен иметь размер в **512 октетов** данных и завершаться **EOF (коней файла)**. Если длина файла кратна 512, то заключительный блок содержит только заголовок и не имеет никаких данных. Блоки данных нумеруются от единицы. **Каждый АСК содержит номер блока данных,** получение которого он подтверждает.

Элементы данных протокола TFTP

- Read Request (RRQ, запрос чтения)
- Write Request (WRQ, запрос записи)
- Data (DATA, данные)
- Acknowledgment (АСК, подтверждение)
- Error (ERROR, ошибка)

- Сообщение об ошибке указывает на события, подобные таким: "файл не найден" или "для записи файла на диске нет места".
- Каждый заголовок TFTP начинается операционным кодом, идентифицирующим тип элемента данных протокола (Protocol Data Unit — PDU). Форматы PDU показаны на рисунке следующего слайда.
- Отметим, что длина Read Request и Write Request меняется в зависимости от длины имени файла и полей режима, каждое из которых представляет собой текстовую строку ASCII, завершенную нулевым байтом. В поле режима могут присутствовать netascii (сетевой ASCII) или octet (октет).

Форматы элементов данных протокола TFTP

Read Request

2 байта	Строка	1 байт	Строка	1 байт
Операционный код = 1	**Имя Файла**	**0**	**Режим**	**0**

Write Request

2 байта	Строка	1 байт	Строка	1 байт
Операционный код = 2	**Имя Файла**	**0**	**Режим**	**0**

Data

2 байта	2 байта	
Операционный код = 3	**Номер блока**	**Данные**

Acknowledgment

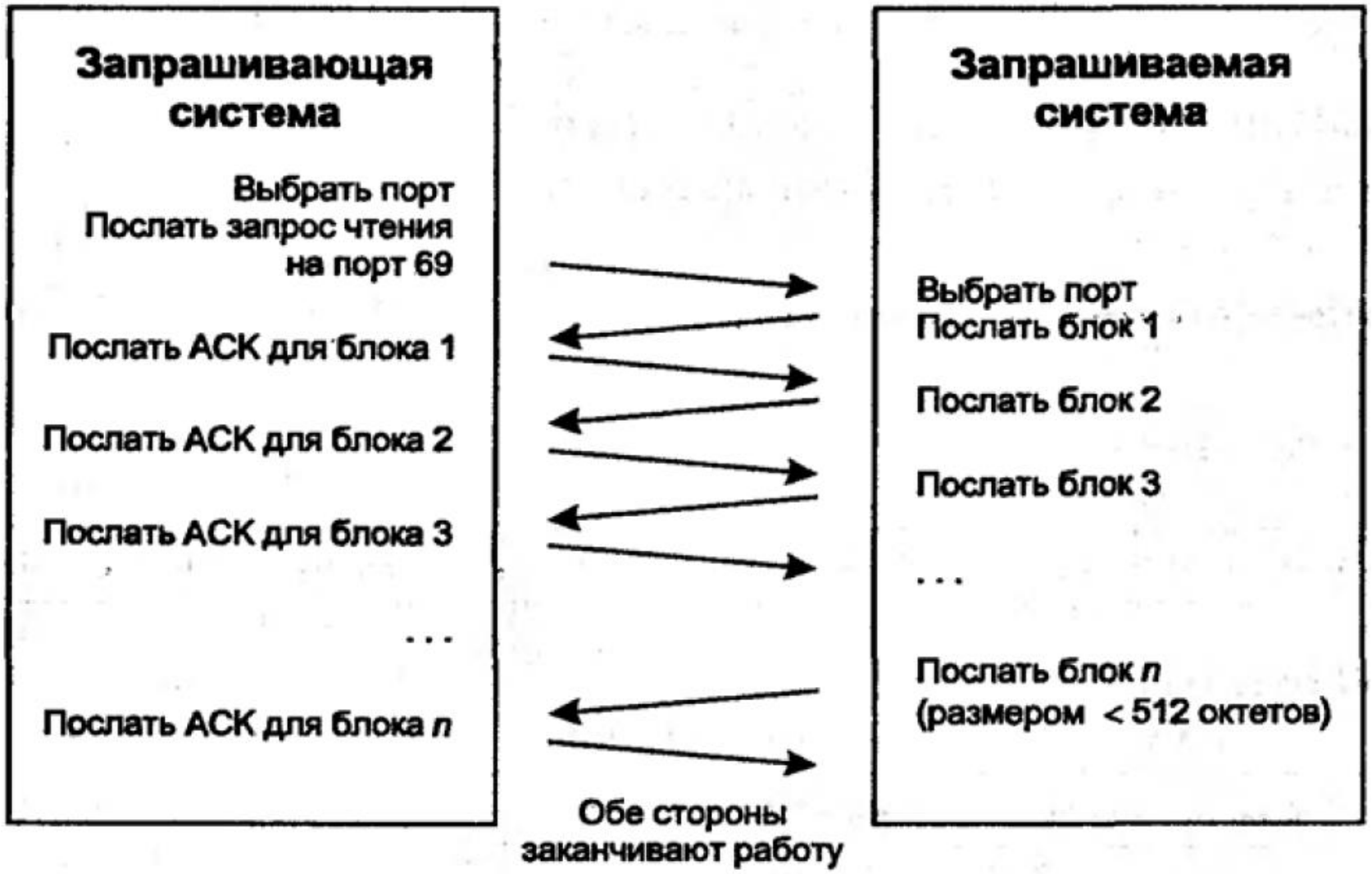
2 байта	2 байта
Операционный код = 4	**Номер блока**

Error

2 байта	1 байт	Строка	1 байт
Операционный код = 5	**Код ошибки**	**Сообщение об ошибке**	**0**

Протокол TFTP
Сценарий работы

После отправки запрашиваемой стороной блока данных она переходит в режим ожидания ACK на посланный блок и, только получив этот ACK, посылает следующий блок данных.



СПАСИБО ЗА ВНИМАНИЕ