

# **Протоколы транспортного уровня TCP и UDP**

# Мультиплексирование и демультиплексирование приложений. Порты.

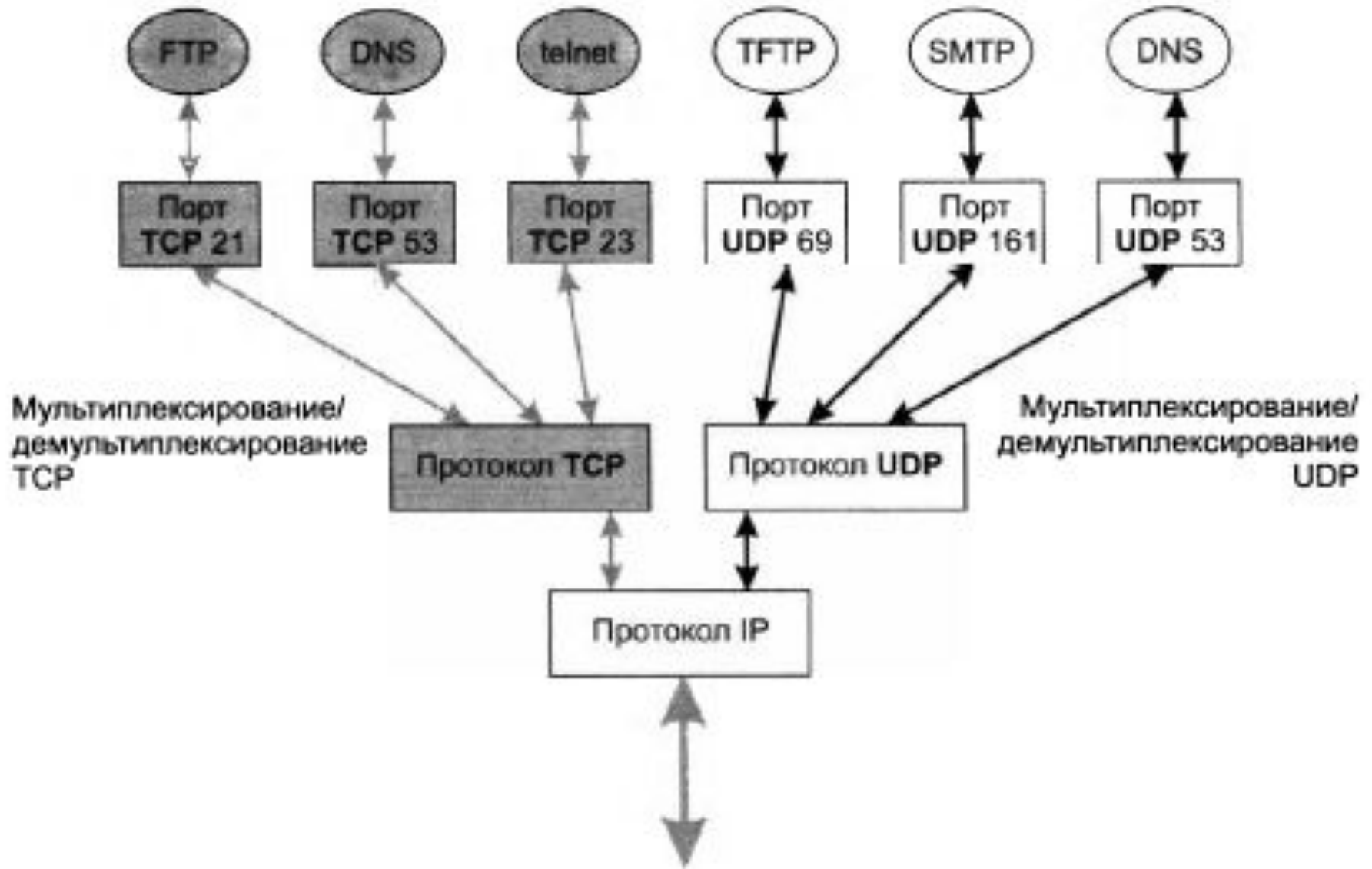


Рис. 16.1. Мультиплексирование и демультиплексирование на транспортном уровне

Протоколы TCP и UDP ведут для каждого приложения две системные очереди: очередь данных, поступающих к приложению из сети, и очередь данных, отправляемых этим приложением в сеть. **Такие системные очереди называются портами, причем входная и выходная очереди одного приложения рассматриваются как один порт. Для идентификации портов им присваивают номера.**

Приложения, которые передают данные на уровень IP по протоколу UDP, **получают номера, называемые UDP-портами.**

Аналогично, приложениям, обращающимся к протоколу TCP, **выделяются TCP-порты.**

В том и в другом случаях это могут быть как назначенные, так и динамические номера.

**Диапазоны чисел, из которых выделяются номера TCP- и UDP-портов, совпадают: от 0 до 1023 для назначенных и от 1024 до 65 535 для динамических.**

Стандартные назначенные номера портов уникально идентифицируют тип приложения (FTP, или HTTP, или DNS и т. д.), однако они не могут использоваться для однозначной идентификации прикладных процессов, связанных с каждым из этих типов приложений.

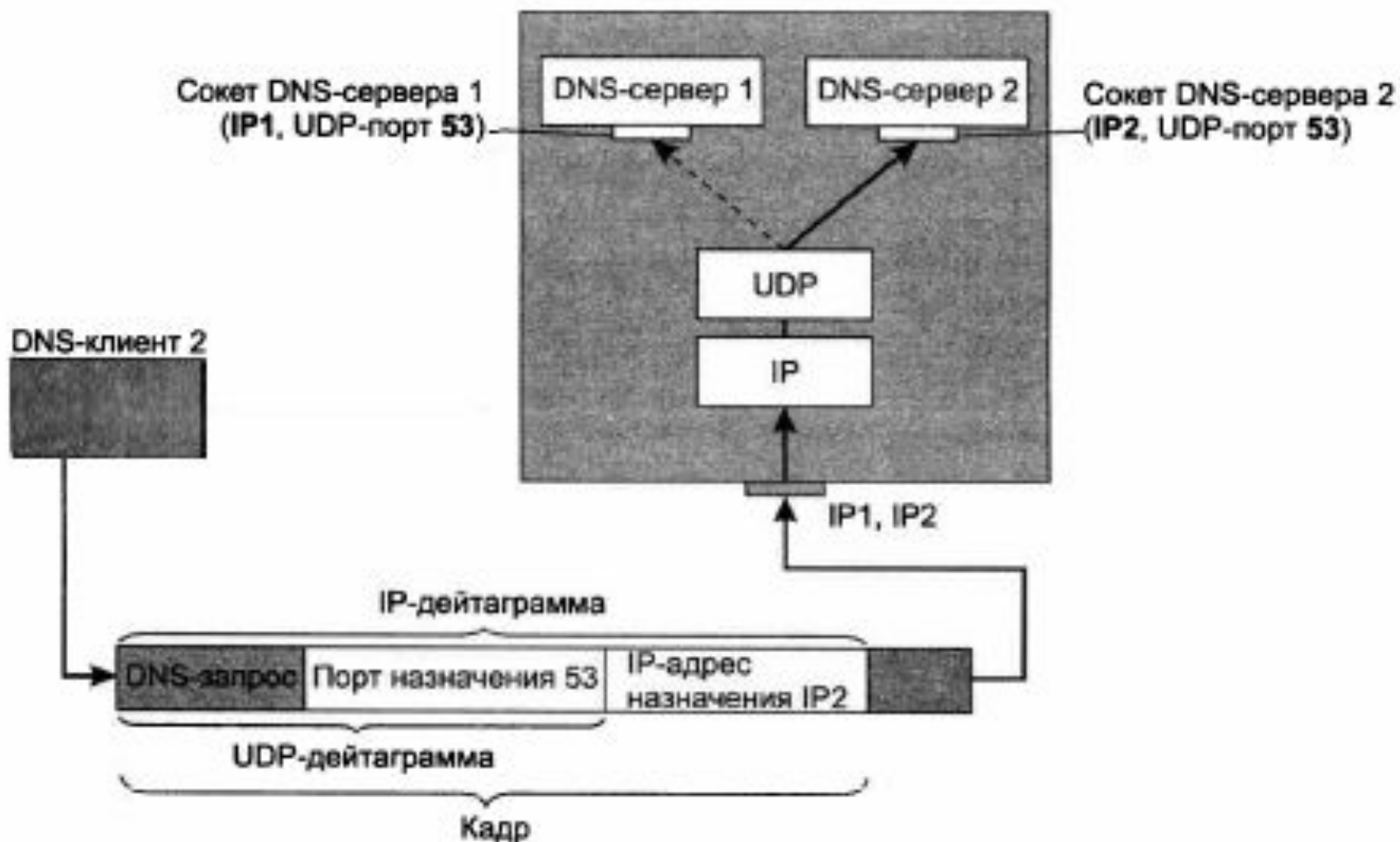


Рис. 16.2. Демультимплексирование протокола UDP на основе сокетов

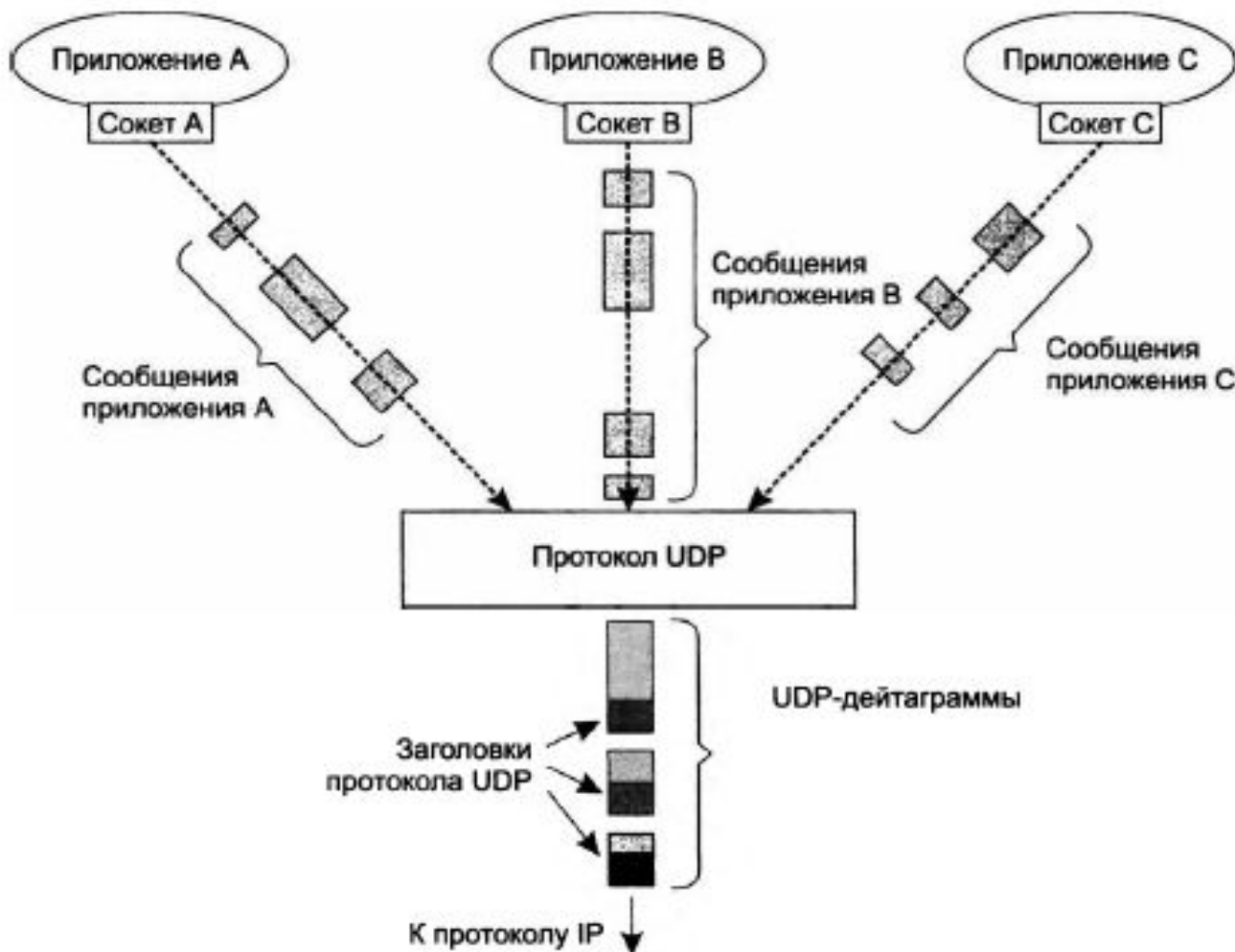
# Сокеты

Прикладной процесс однозначно определяется в пределах сети и в пределах отдельного компьютера **парой** (IP-адрес, номер порта), называемой **сокетом (socket)**. Сокет, определенный IP-адресом и номером UDP-порта, называется **UDP-сокетом**, а IP-адресом и номером TCP-порта — **TCP-сокетом**.

После получения IP-пакета от протокола канального уровня протокол IP анализирует содержимое заголовка этого пакета, после чего заголовок отбрасывается, и «наверх» передается содержимое поля данных IP-

**Протокол UDP является дейтаграммным протоколом, реализующим ненадежный сервис по возможности, который не гарантирует доставку сообщений адресату.**

При работе на хосте-отправителе данные от приложений поступают протоколу UDP через порт в виде сообщений. Протокол UDP добавляет к каждому отдельному сообщению свой 8-байтный заголовок, формируя **UDP-дейтаграммы**, и передает их ниже



**Рис. 16.3.** Работа протокола UDP на хосте-отправителе

Каждая дейтаграмма переносит отдельное пользовательское сообщение. Сообщения могут иметь разную длину, не превышающую, однако, длину поля данных протокола IP, которое, в свою очередь, ограничено размером кадра технологии нижнего уровня. **Поэтому если буфер UDP переполняется, то сообщение приложения отбрасывается.**

Заголовок UDP состоит из четырех 2-байтных полей:

- **номер UDP-порта отправителя;**
- **номер UDP-порта получателя;**
- **контрольная сумма;**
- **длина дейтаграммы**



**Функции протокола UDP сводятся к простой передаче данных между прикладным и сетевым уровнями, а также к примитивному контролю искажений в передаваемых данных. При контроле искажений протокол UDP только диагностирует, но не исправляет ошибку. Работая на хосте-получателе, протокол UDP принимает от протокола IP извлеченные из пакетов UDP-дейтаграммы. Протокол UDP освобождает дейтаграмму от UDP-заголовка. Полученное в результате сообщение он передает приложению на соответствующий UDP-сокеты. Таким образом, протокол UDP выполняет демультимплексирование на**

# Протокол TCP и TCP-сегменты

Протокол TCP предназначен для передачи данных между приложениями. Этот протокол основан на логическом соединении, что позволяет ему обеспечивать гарантированную доставку данных, используя в качестве инструмента

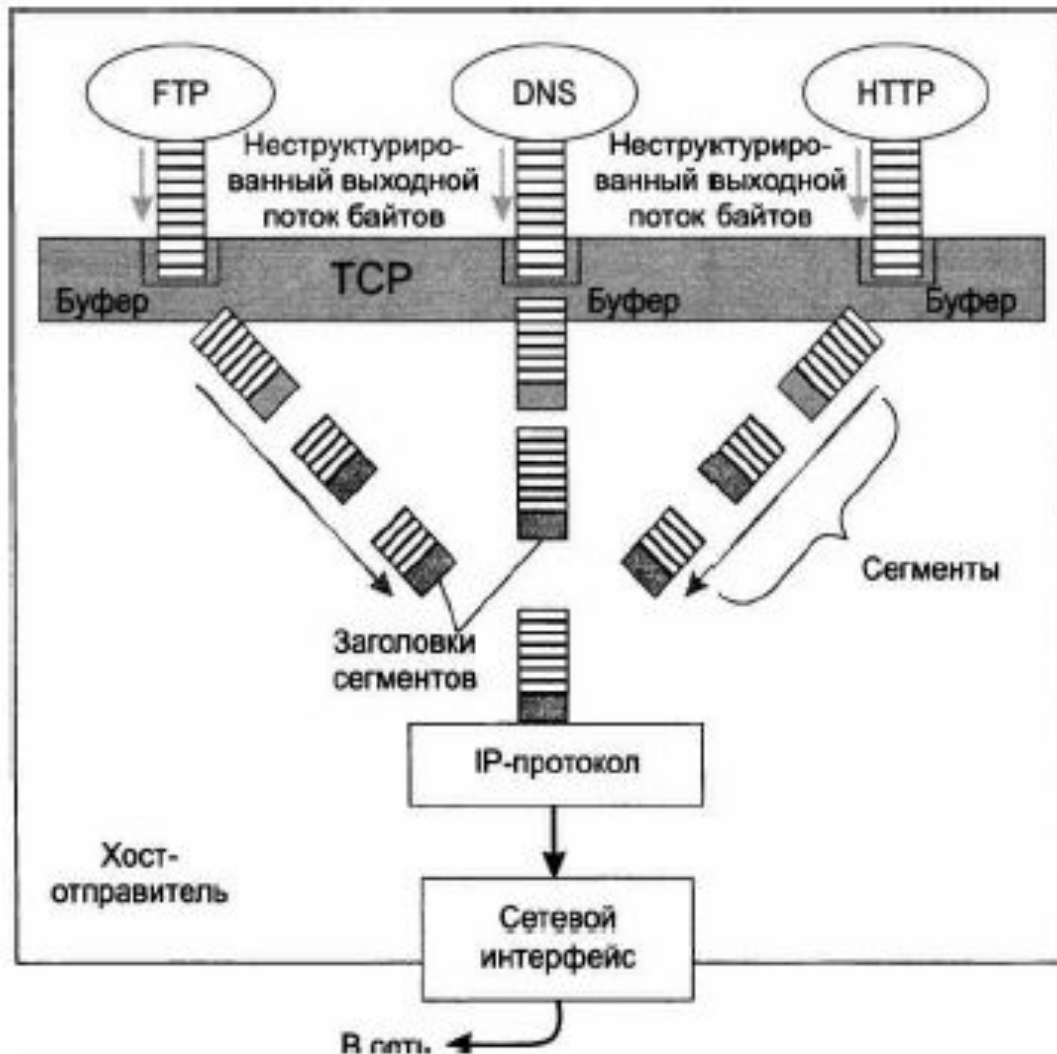
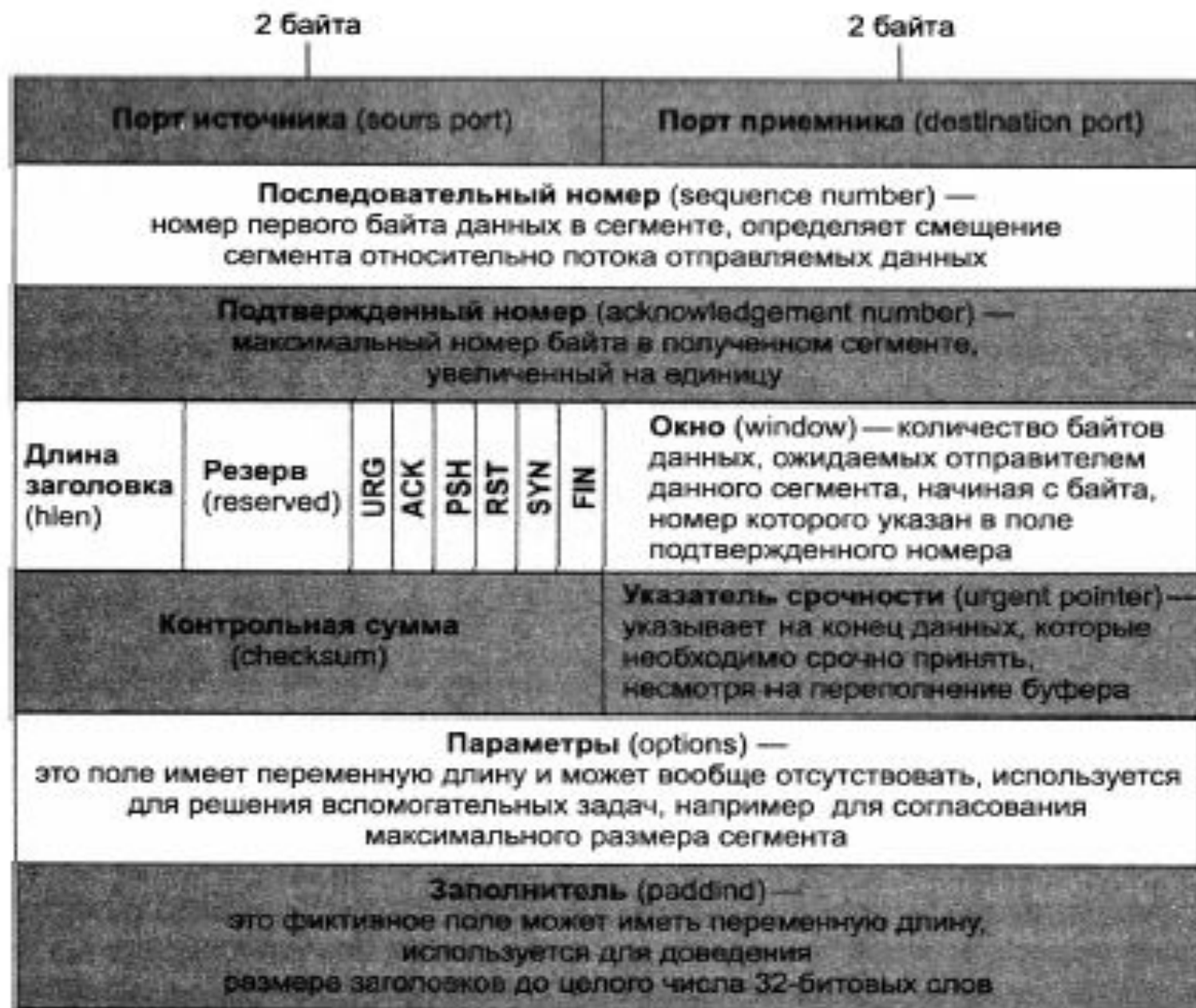


Рис. 16.4. Формирование TCP-сегментов из потока байтов



**Рис. 16.5.** Формат заголовка TCP-сегмента

Коротко поясним значение однобитных полей, называемых флагами, или **кодовыми битами** (code bits). Они расположены сразу за резервным полем и содержат служебную информацию о типе данного сегмента. Положительное значение сигнализируется установкой этих битов в единицу:

- **URG** — срочное сообщение;
- **ACK** — квитанция на принятый сегмент;
- **SH** — запрос на отправку сообщения без ожидания заполнения буфера;
- **RST** — запрос на восстановление соединения;
- **SYN** — сообщение, используемое для синхронизации счетчиков переданных данных при установлении соединения;
- **FIN** — признак достижения передающей стороной последнего байта в потоке передаваемых данных

## Логические соединения — основа надежности TCP

Основным отличием TCP от UDP является то, что на протокол TCP возложена дополнительная задача — обеспечить надежную доставку сообщений, используя в качестве основы ненадежный дейтаграммный протокол IP.

Протокол TCP устанавливает логические соединения между прикладными процессами, причем в каждом соединении участвуют только два процесса. TCP-соединение является дуплексным, то есть

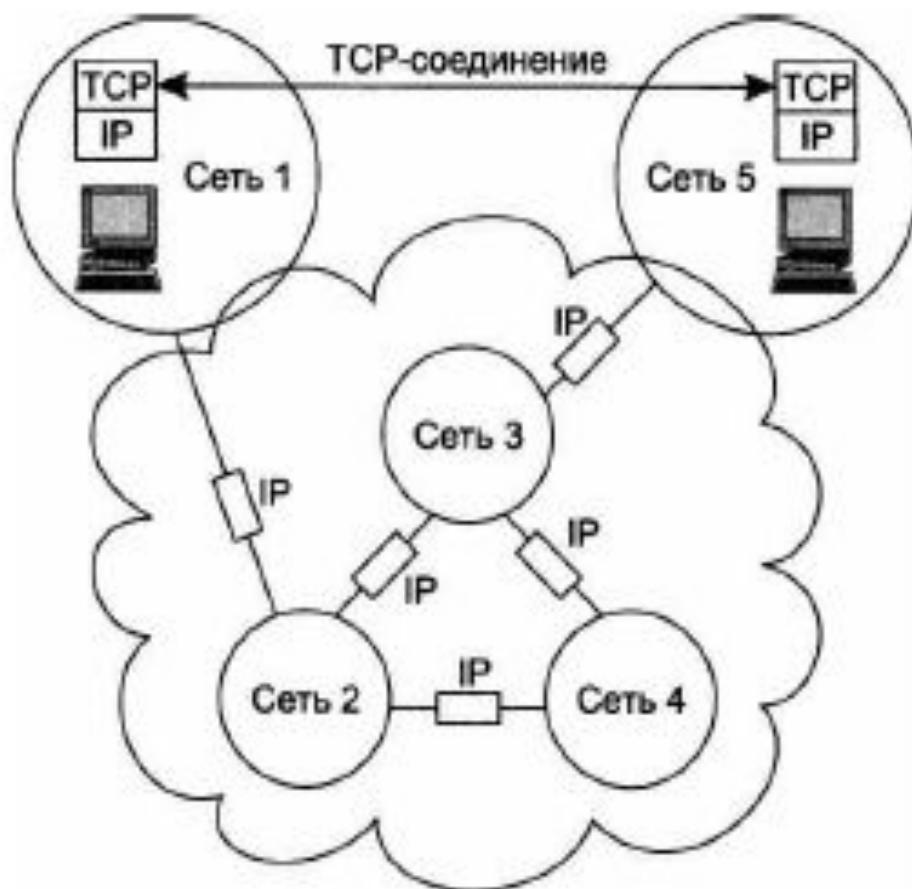


Рис. 16.6. TCP-соединение создает надежный логический канал между конечными узлами

При установлении логического соединения модули ТСР договариваются между собой о параметрах процедуры обмена данными. В протоколе ТСР каждая сторона соединения посылает противоположной стороне следующие параметры:

- **максимальный размер сегмента**, который она готова принимать;
- **максимальный объем данных**, которые она разрешает другой стороне передавать в свою сторону, даже если та еще не получила квитанцию на предыдущую порцию данных (размер окна);
- **начальный порядковый номер байта**, с которого она начинает отсчет потока данных в

Соединение устанавливается по инициативе клиентской части приложения. При необходимости выполнить обмен данными с серверной частью приложение-клиент обращается к нижележащему протоколу TCP, который в ответ на это обращение посылает сегментзапрос на установление соединения протоколу TCP, работающему на стороне СЕ

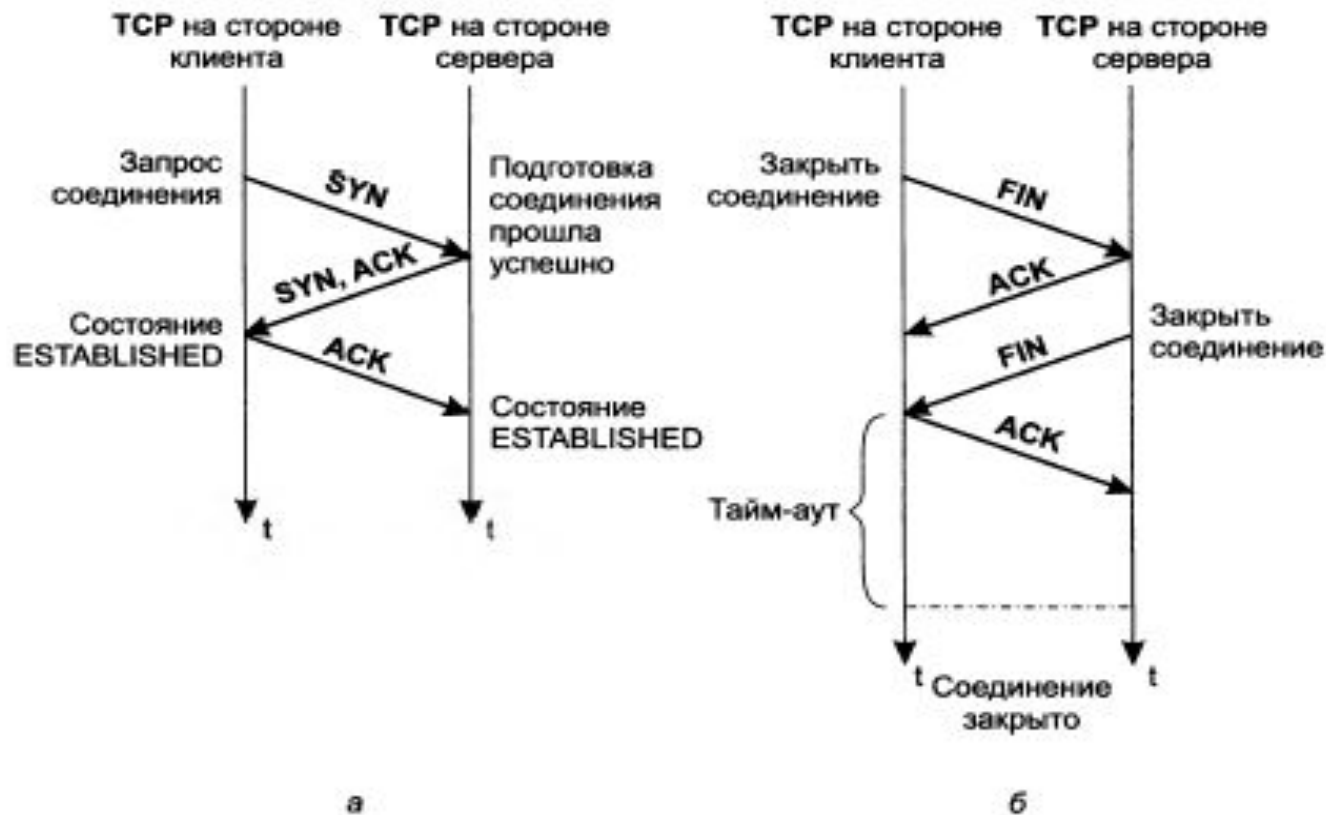


Рис. 16.7. Процедура установления и разрыва логического соединения при нормальном течении процесса

Сокет одновременно может участвовать в нескольких соединениях.

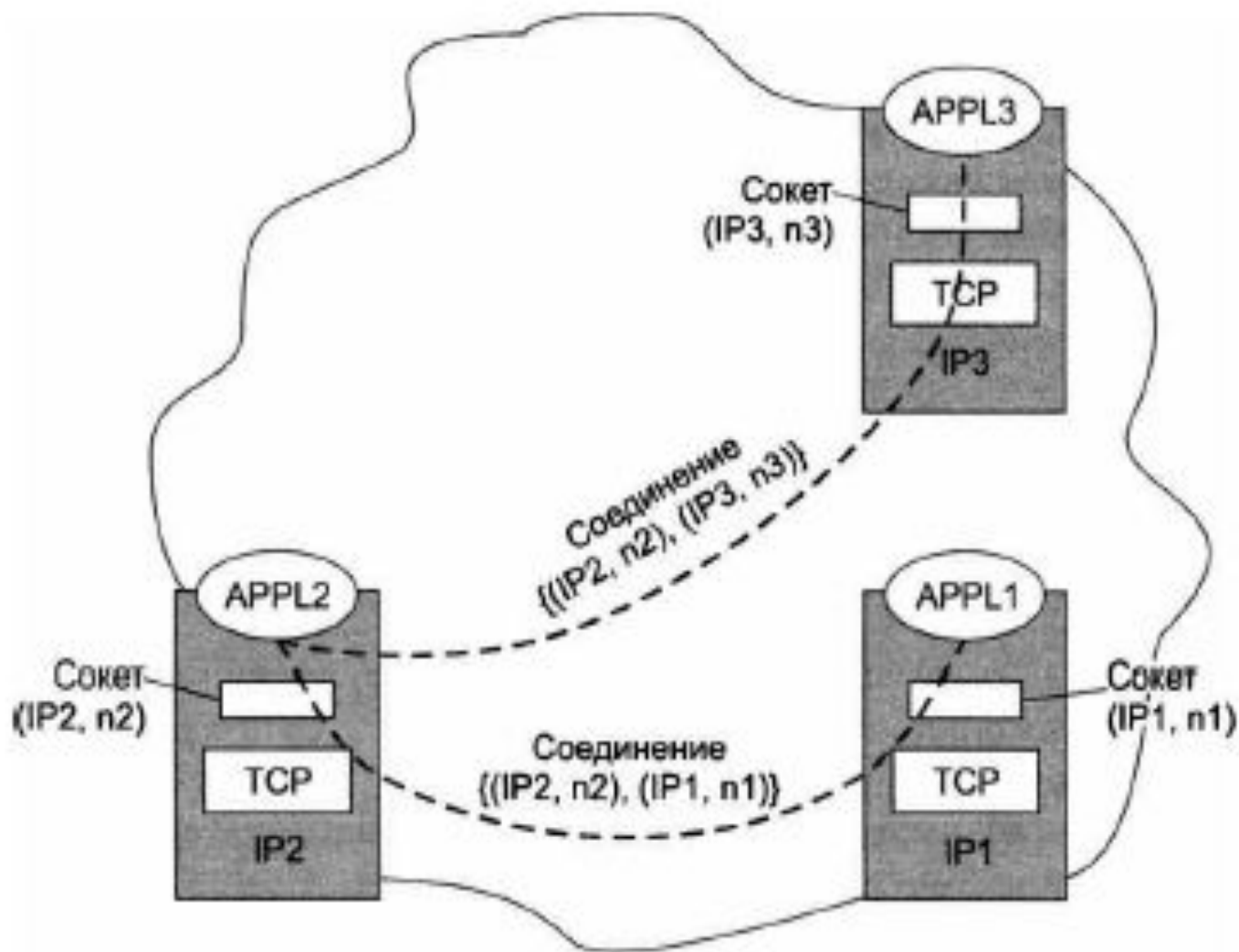
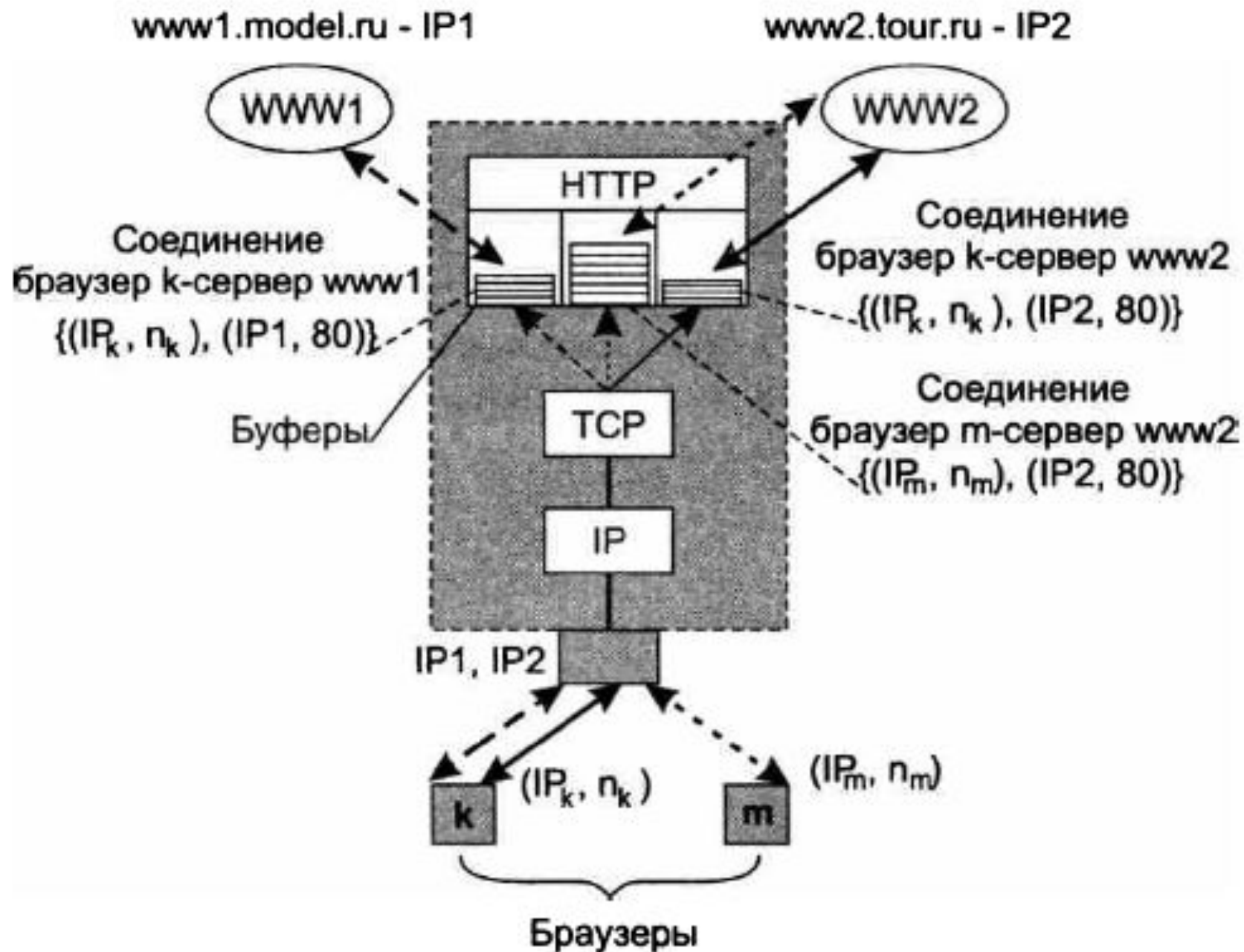


Рис. 16.В. Один сокет может участвовать в нескольких соединениях





**Рис. 16.9.** Демультимплексирование протокола TCP на основе соединений

## Методы квитирования

Один из наиболее естественных приемов, используемых для организации надежного обмена данными, — **это передача с квитированием, суть которой состоит в следующем:**

**Отправитель отсылает данные, а получатель подтверждает их получение квитанциями. Если отправитель вовремя не получает квитанции на переданные данные, то он передает их повторно.**

Любая попытка реализации запроса повторной передачи (**Automatic Repeat reQuest, ARQ**) «обрастает» множеством деталей и вопросов. Все решения могут быть разделены на два класса:

- **методы простоя источника (Stop-and-Wait);**
- **методы скользящего окна.**

В свою очередь, методы скользящего окна тоже делятся на два класса:

- **методы, использующие окно передачи, — к ним, в частности, относится метод передачи с возвратом на N пакетов (Go-Back-N);**
- **методы, использующие окно передачи и окно приема, — примером является метод передачи с выборочным повторением.**

## **Основные черты методов:**

- **Отправитель (источник) и получатель (приемник), в общем случае работающие асинхронно, осуществляют передачу пакетов по ненадежной линии связи, в которой возможны искажения, большие задержки и потери пакетов.**
- **Отправитель принимает данные от протокола верхнего уровня (приложения), получатель передает полученные данные на верхний уровень (приложению).**
- **Получатель располагает механизмом определения искаженных пакетов, например по контрольной сумме.**
- **После успешного получения пакета получатель посылает отправителю квитанции (acknowledgment, АСК).**
- **Для отслеживания задержек пакетов**

## Метод простоя источника

В методе простоя источника отправитель передает последовательность ненумерованных пакетов. **Метод требует, чтобы отправитель дожидался от получателя квитанции и только после этого посылал следующий пакет.**

С переданным пакетом отправитель связывает таймер. **Если в течение тайм-аута квитанция не пришла, то пакет считается утерянным или искаженным и его передача повторяется.**

Если первая квитанция не была утеряна, а просто шла к отправителю слишком долго, то возможна коллизия с приходом в источник двух квитанций на пакет-оригинал и пакетдубль.

Вторую квитанцию отправитель может интерпретировать как квитанцию на получение следующего по порядку пакета.

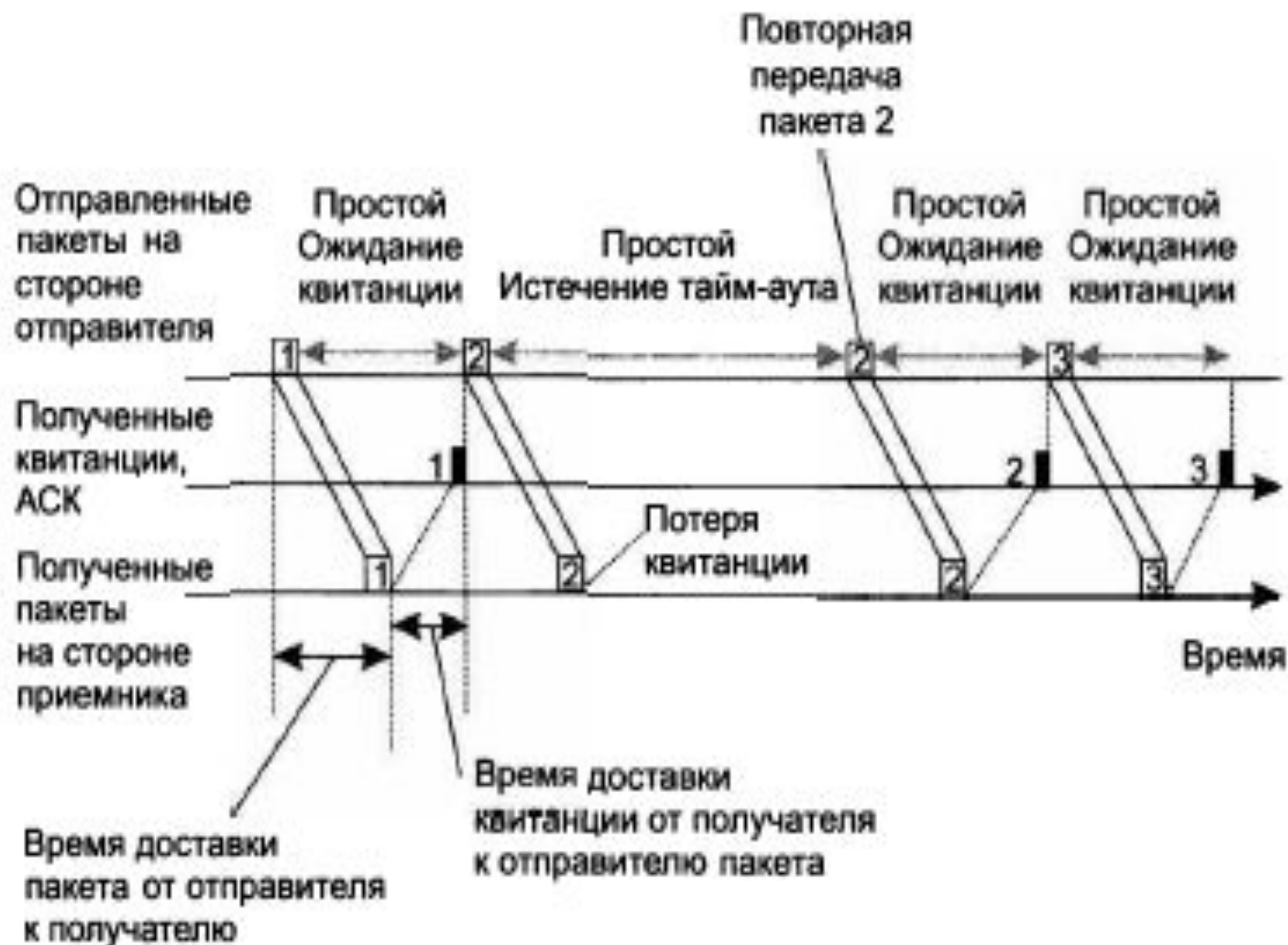


Рис. 16.10. Метод простоя источника

**Концепция скользящего окна (sliding window) заключается в том, что для повышения скорости передачи данных отправителю разрешается передать некоторое количество пакетов, не дожидаясь прихода на эти пакеты квитанций.**

**Для идентификации пакетам присваиваются уникальные последовательные номера, которые размещаются в заголовках пакетов. Разрядность поля «номер пакета» определяет диапазон возможных номеров. Когда этот диапазон исчерпывается,**

Окно определяется на последовательности пронумерованных пакетов (рис. 16.11). **Окно всегда имеет нижнюю границу, называемую также базой окна, и верхнюю границу. Количество номеров, попадающих в пределы окна, называют размером окна.** Очевидно, что окно всегда перемещается в сторону больших номеров.

Окно, показанное на рисунке, регулирует процесс передачи — оно ограничивает количество пакетов, которые отправитель может передать до получения квитанции. Окно может быть определено не только для передачи, но и для приема пакетов - в таком случае ограничивается количество пакетов, которые

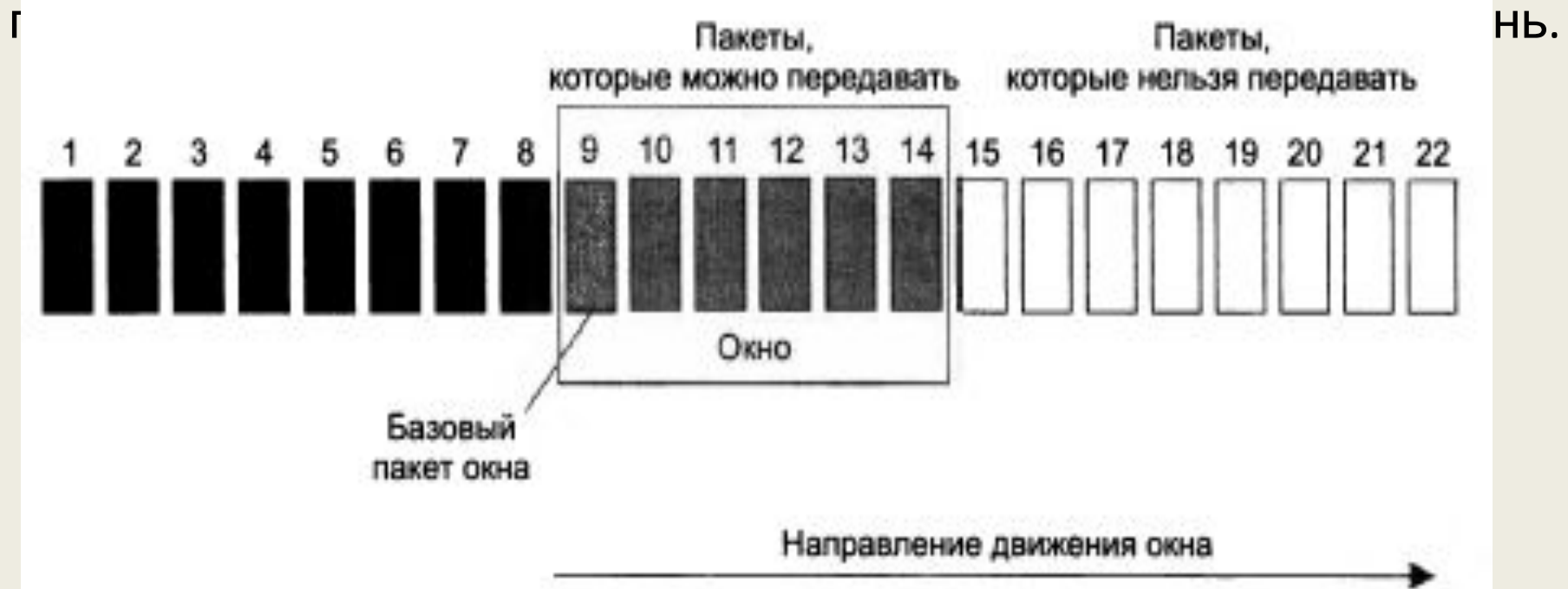


Рис. 16.11. Окно передачи

Диапазон номеров пакетов, разрешенных к отправке

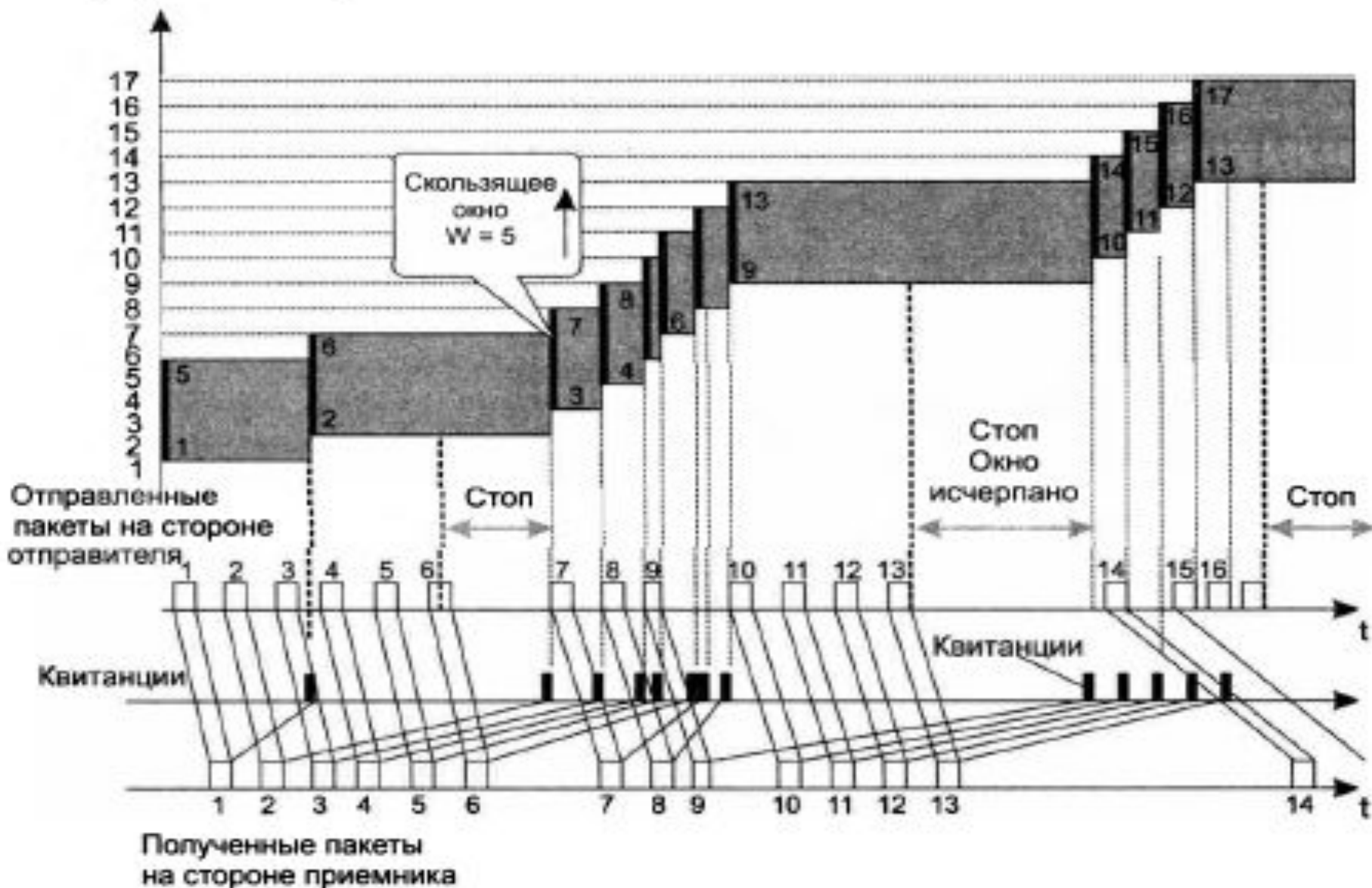


Рис. 16.12. Метод скользящего окна



## Передача с возвратом на N пакетов

В данном методе ставится задача обеспечить более эффективное использование линии связи, чем в методе с простым источником. **Для этого отправителю разрешается отправлять следующие пакеты, не дожидаясь подтверждения получателем их успешного приема.** Количество переданных таким образом пакетов ограничивается окном.

Поскольку отправитель должен иметь возможность при необходимости повторить передачу любого из переданных пакетов, их необходимо буферизовать. Кроме того, отправитель

Д  
П



Рис. 16.13. Окно передачи в методе с возвратом на N пакетов

**При поступлении нового пакета получатель всегда выполняет одни и те же действия, проверяя:**

- является ли пакет неискаженным;**
- является ли он следующим по порядку в последовательности уже полученных пакетов.**

Если оба условия выполнены, то пакет принимается и передается на более высокий уровень, а отправителю посылается квитанция, в которой указывается номер успешно принятого пакета.

Если хотя бы одно из этих условий не выполнено, то пакет отбрасывается, а отправителю снова посылается квитанция с номером последнего до

На процесс передачи пакетов влияют следующие события:

- **Исчерпание окна** — ситуация, когда все пакеты из окна отправлены, но не подтверждены. **В этом случае передача останавливается.**
- **Истечение тайм-аута** — это событие интерпретируется отправителем как потеря пакета или квитанции, а значит, **выполняется повторная передача базового пакета** и для него заново устанавливается таймер.
- **Поступление квитанции.** Соответствующий пакет и все пакеты в пределах окна с меньшими номерами считаются успешно принятыми. **Окно сдвигается, фиксируется новый базовый пакет, переустанавливается таймер.** Если квитанция на базовый пакет пришла после истечения его тайм-аута, а значит, и после его повторной передачи, эта квитанция «засчитывается», и выполняются все действия, определенные для этого случая.

Эффективность данного метода выше по сравнению с методом простоя источника за счет передачи в линию связи сразу нескольких пакетов. **Однако для него характерна избыточность: во-первых, получатель отбрасывает не только искаженный, но и корректно принятый пакет, если его номер выбивается из последовательности, во-вторых, отправитель повторно передает не только потерянный или искаженный пакет, но и все пакеты, которые были отправлены после него.**

## **Передача с выборочным повторением**

**В данном методе получатель может выборочно запросить повторную передачу отдельного пакета, а не всей последовательности переданных пакетов.**

**Чтобы избежать избыточных повторных передач, принимающей стороне запрещено отбрасывать правильно принятый пакет лишь потому, что его номер выбивается из последовательности.**

**В данном методе концепция скользящего окна используется как для передачи, так и для приема пакетов. Оба окна определены на одной и той же последовательности**

Отправитель и получатель работают только с пакетами, находящимися в пределах окна передачи и окна приема соответственно.

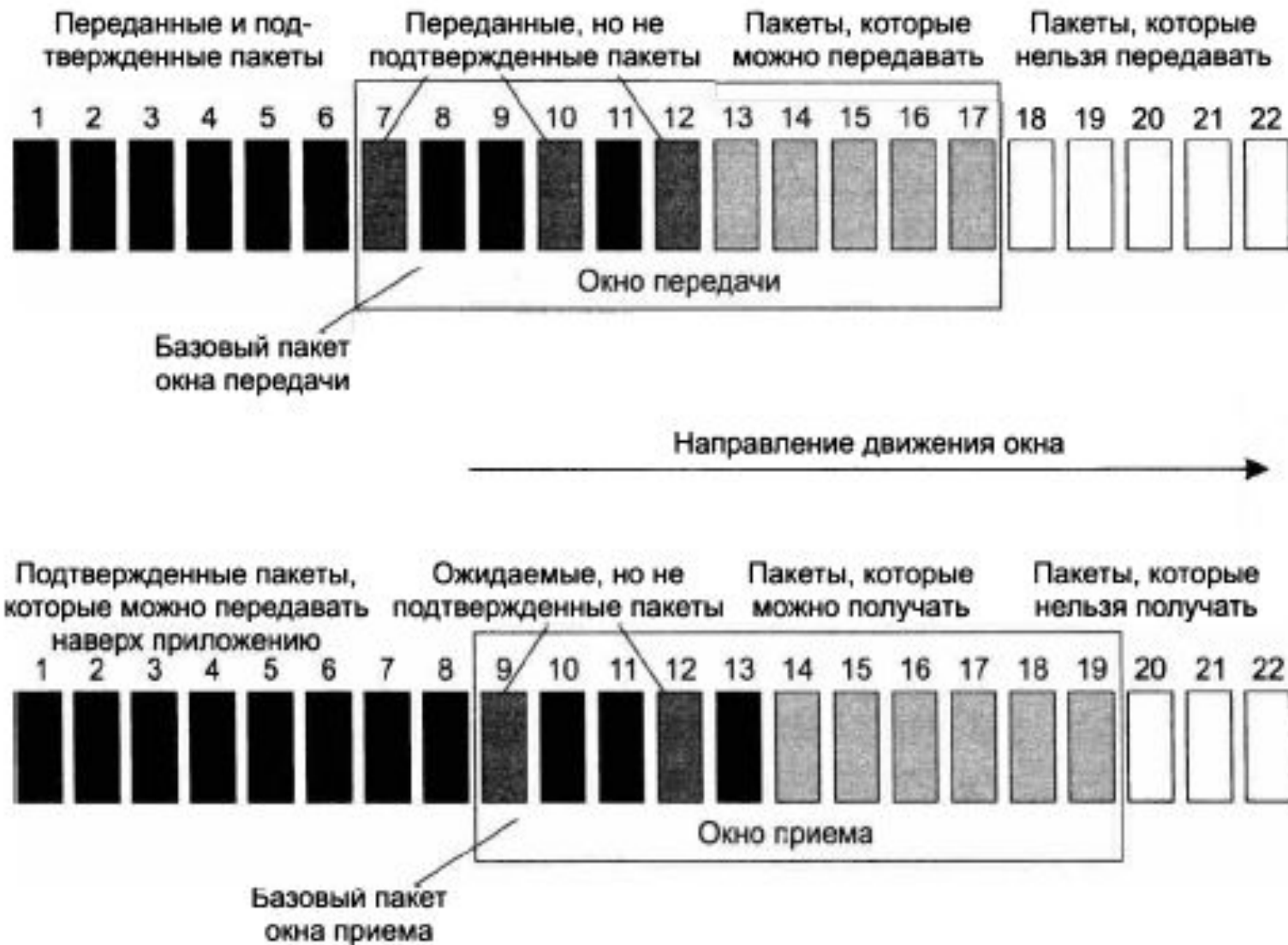


Рис. 16.14. Окна передачи и приема при выборочном повторении

Номера из окна приема в общем случае могут соответствовать:

- **принятым и подтвержденным пакетам (10, 11, 13),** которые не могут быть переданы из буфера верхнему уровню, так как в их последовательность «вклинились» номера некоторых отсутствующих пакетов (9, 12);
- **ожидаемым, еще не полученным пакетам (9,12);**
- **пакетам, которые получателю разрешено принимать (14-19),** так как их номера попадают в окно приема.

Получатель принимает, размещает в буфере и подтверждает квитанцией любой принятый пакет при условии, что он не искажен и его номер попадает в окно приема. Если принят базовый пакет, то левая граница окна приема сдвигается до первого ожидаемого, но еще не полученного пакета.

На работе отправителя сказываются следующие события:

- **Исчерпание окна.** Отправитель последовательно посылает пакеты до тех пор, пока не исчерпается окно передачи.
- **Приход квитанции.** Отправитель, получив квитанцию, присваивает пакету статус успешно переданного. Если это был базовый пакет (например, пакет 7 на рисунке), то **окно смещается вправо до первого по порядку принятого, но не подтвержденного пакета, который становится базой** (на рисунке пакет 10).
- **Истечение тайм-аута.** Таймер устанавливается для каждого пакета отдельно, по истечении тайм-аута соответствующий пакет повторяют. Таким образом, пакет повторяют, только если он был потерян или искажен.



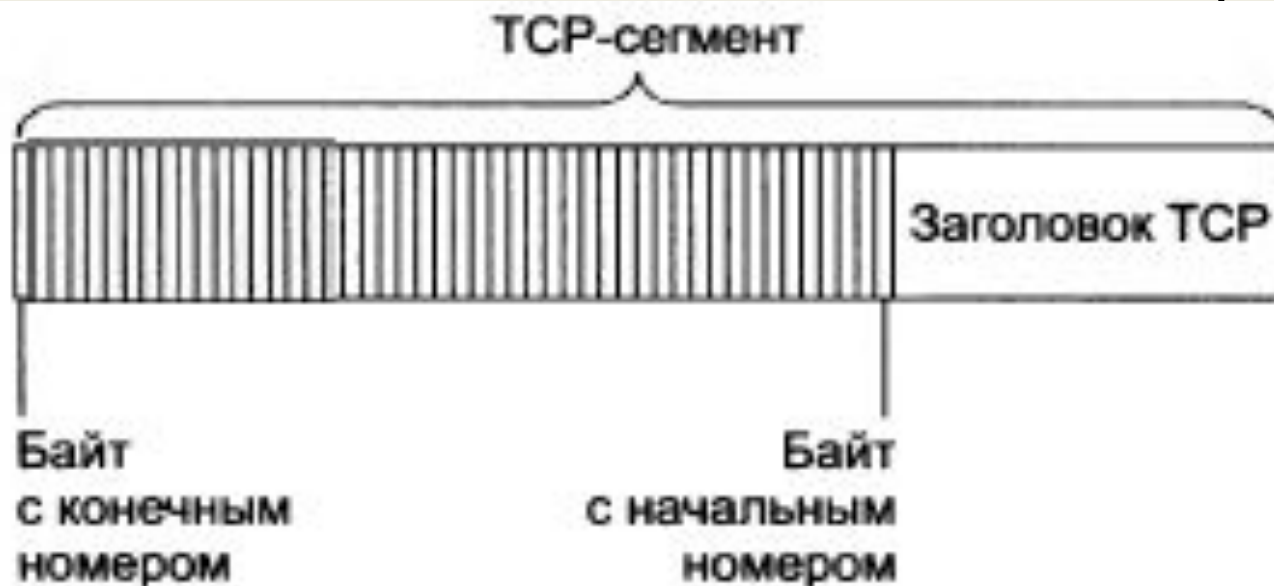
# Реализация метода скользящего окна в протоколе TCP

## Сегменты и поток байтов

Алгоритм скользящего окна в протоколе TCP имеет некоторые существенные особенности. В частности, в рассмотренном обобщенном алгоритме скользящего окна единицей передаваемых данных является пакет, и размер окна также определяется в кадрах, в то время как в протоколе TCP дело обстоит совсем по-другому.

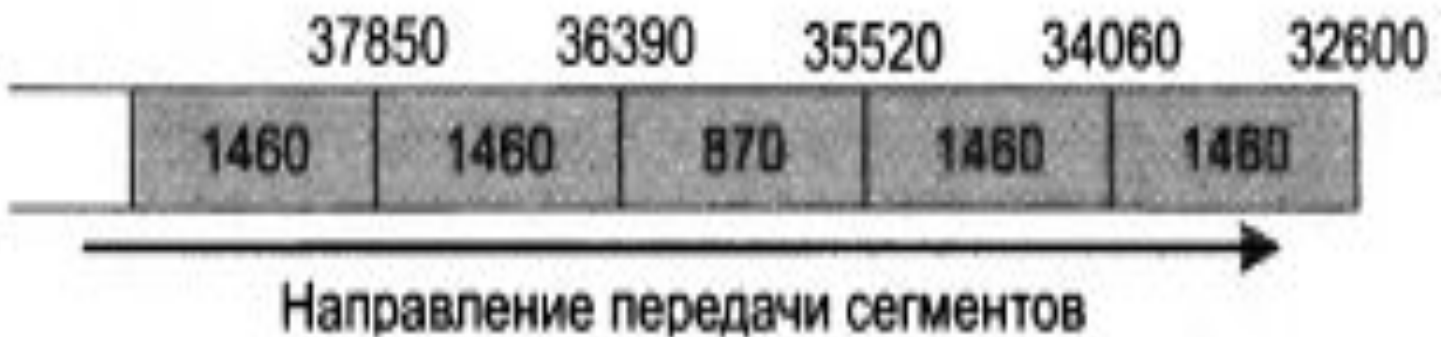
В ходе переговорного процесса модули ТСП обеих участвующих в обмене сторон договариваются между собой о параметрах процедуры обмена данными. Одним из таких параметров является начальный номер байта, с которого будет вестись отсчет в течение всего функционирования данного соединения. У каждой стороны свой начальный номер. Нумерация байтов в пределах сегмента осу

16.1



**Рис. 16.15.** Нумерация байтов в TCP-сегменте

Когда отправитель посылает ТСР-сегмент, он помещает в поле последовательного номера **номер первого байта данного сегмента, который служит идентификатором сегмента.** На основании этих номеров получатель ТСР-сегмента не только отличает данный сегмент от других, но и позиционирует полученный фрагмент относительно общего потока байтов. Кроме того, он может сделать вывод, например, что полученный сегмент является дубликатом или что между двумя полученными сегментами пропущены данные.



**Рис. 16.16.** Порядковый номер и номер квитанции

**В качестве квитанции получатель сегмента отсылает ответное сообщение (сегмент), в поле подтвержденного номера которого он помещает число, на единицу превышающее максимальный номер байта в полученном сегменте.**

Подтвержденный номер часто интерпретируют не только как оповещение о благополучной доставке, но и как номер следующего ожидаемого байта данных.

**Квитанция в протоколе TCP посылается только в случае правильного приема данных.** Таким образом, отсутствие квитанции означает либо потерю сегмента, либо потерю квитанции, либо прием

# Система буферов при дуплексной передаче

Поскольку протокол TCP является дуплексным, каждая сторона одновременно выступает и как отправитель, и как получатель. У каждой стороны есть пара буферов: один — для хранения принятых сегментов, другой — для сегментов, которые только еще предстоит отправить.

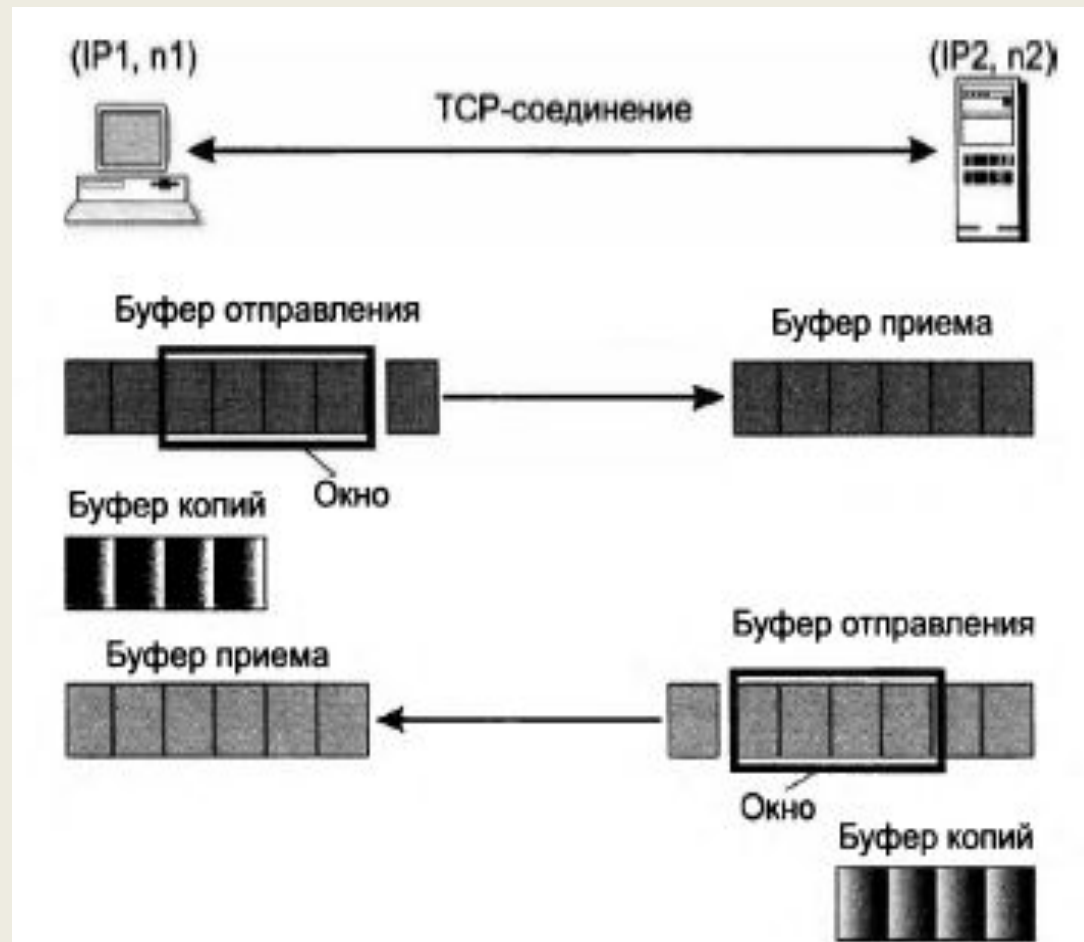


Рис. 16.17. Система буферов TCP-соединения

Из потока байтов модуль TCP «нарезает» последовательность сегментов и поочередно отправляет их приложению-получателю. В этом потоке можно указать несколько логических границ:

- **Первая граница отделяет сегменты, которые уже были отправлены и на которые уже пришли квитанции.** Последняя квитанция пришла на байт с номером  $N$ .
- По другую сторону этой границы располагается **окно размером  $W$  байт.**
- Оставшаяся часть окна — это **сегменты, которые пока не отправлены, но могут быть отправлены, так как входят в пределы окна.**
- И наконец, **последняя граница указывает на начало последовательности сегментов, ни один из которых не может быть отправлен до тех пор, пока не придет очередная квитанция и окно не будет сдвинуто вправо.**

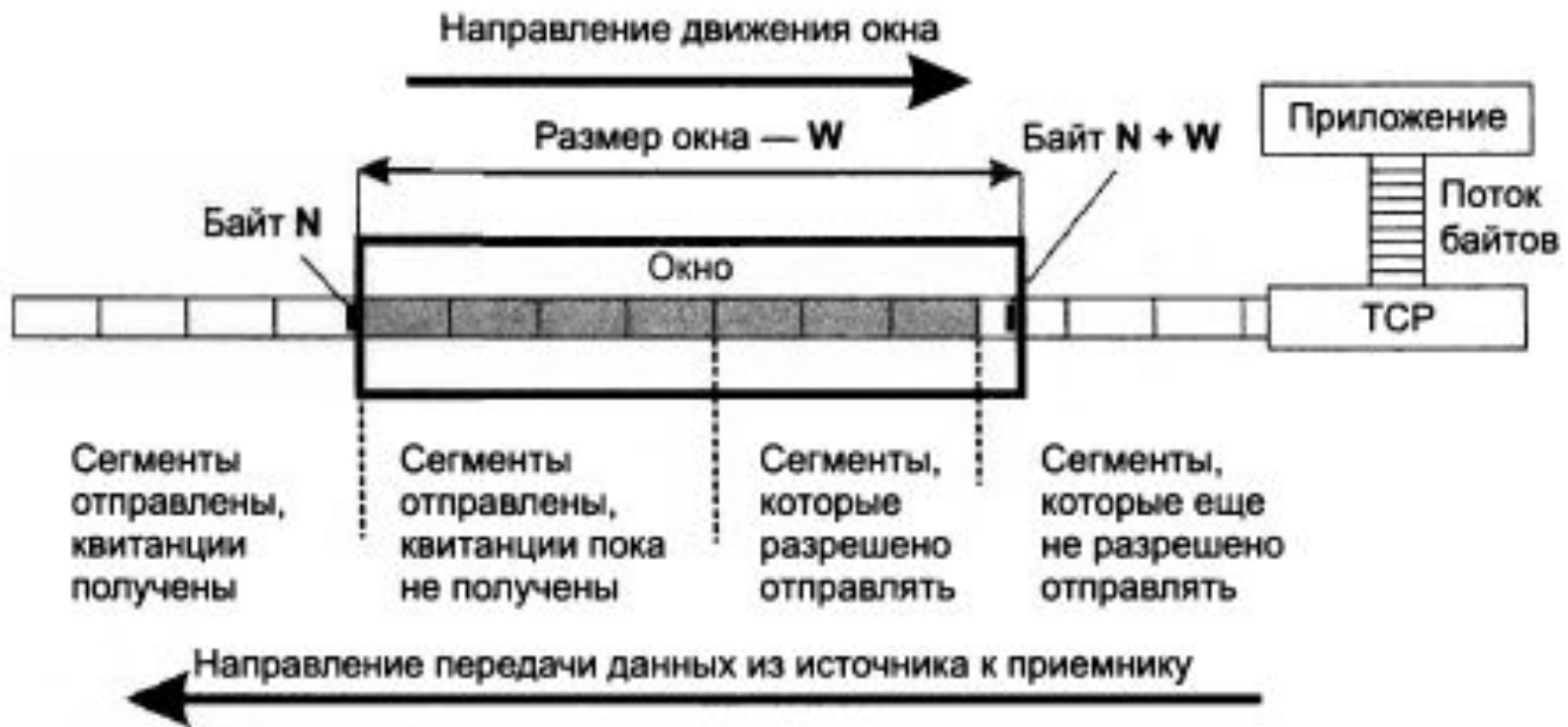
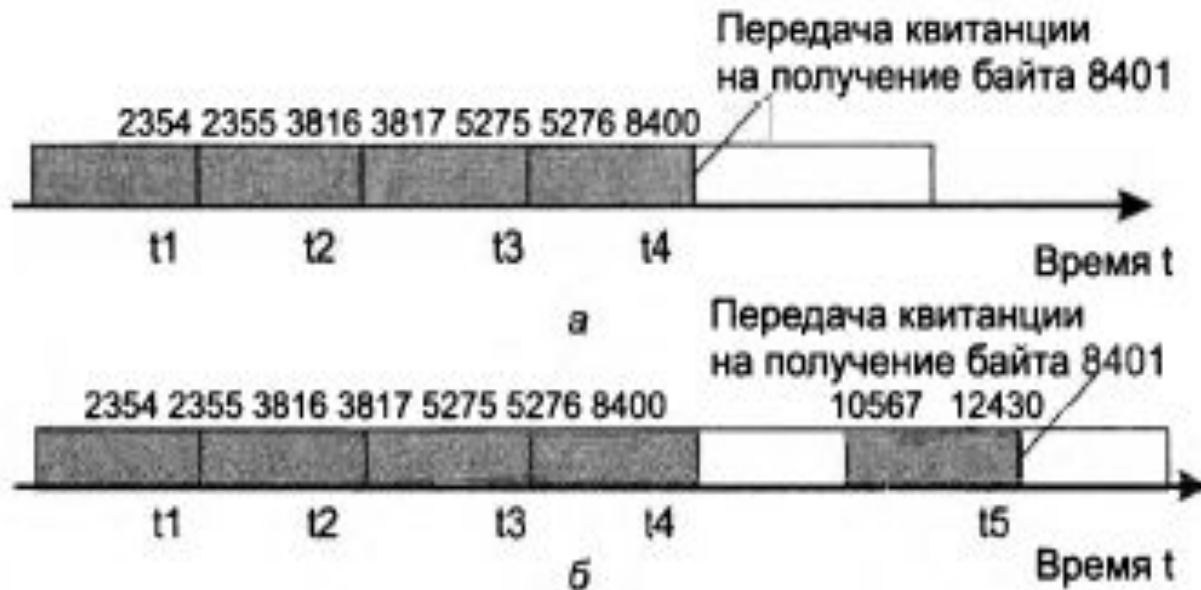


Рис. 16.18. Особенности реализации алгоритма скользящего окна в протоколе TCP

## Накопительный принцип квитирования

Если размер окна равен  $W$ , а последняя по времени квитанция содержала значение  $N$ , то отправитель может посылать новые сегменты до тех пор, пока в очередной сегмент не попадет байт с номером  $N + W$ . Этот сегмент выходит за рамки окна, и передачу в таком случае необходимо приостановить до прихода следующей квитанции.

Получатель может послать квитанцию, подтверждающую получение сразу нескольких сегментов, если они образуют непрерывный поток байтов.



**Рис. 16.19.** Накопительный принцип квитирования: а — плотное заполнение буфера (в момент  $t_4$  передается квитанция на байт 8401), б — неплотное заполнение буфера (в момент  $t_5$  снова передается квитанция на байт 8401)

Вполне возможны ситуации, когда сегменты приходят к получателю не в том порядке, в каком были посланы, то есть в приемном буфере может образоваться «прогалина» (рис. 16.19, б).

Когда протокол TCP передает в сеть сегмент, он «на всякий случай» помещает его копию в буфер, называемый также очередью повторной передачи, и запускает таймер. Когда приходит квитанция на этот сегмент, соответствующая копия удаляется из очереди. Если же квитанция не приходит до истечения срока, то сегмент, вернее его копия, посылается повторно.



# Параметры управления потоком в ТСР

В протоколе ТСР тайм-аут определяется с помощью достаточно сложного адаптивного алгоритма, идея которого состоит в следующем. При каждой передаче засекается время от момента отправки сегмента до прихода квитанции о его приеме (время оборота). Получаемые значения времени оборота усредняются с весовыми коэффициентами, возрастающими от предыдущего замера к последующему. **В качестве тайм-аута выбирается среднее время оборота, умноженное на некоторый коэффициент. Практика показывает, что значение этого коэффициента должно**

**Признаком перегрузки ТСР-соединения является возникновение очередей на промежуточных узлах (маршрутизаторах) и на конечных узлах (компьютерах). При переполнении приемного буфера конечного узла «перегруженный» модуль ТСР, отправляя квитанцию, помещает в нее новый уменьшенный размер окна. Если он совсем отказывается от приема, то в квитанции указывается окно нулевого размера. После приема квитанции с нулевым значением окна протокол-отправитель время от времени делает контрольные попытки продолжить обмен данными. Если протокол-приемник уже готов принимать информацию, то в ответ на контрольный запрос он посылает квитанцию с указанием ненулевого размера окна.**



**Спасибо за внимание!**