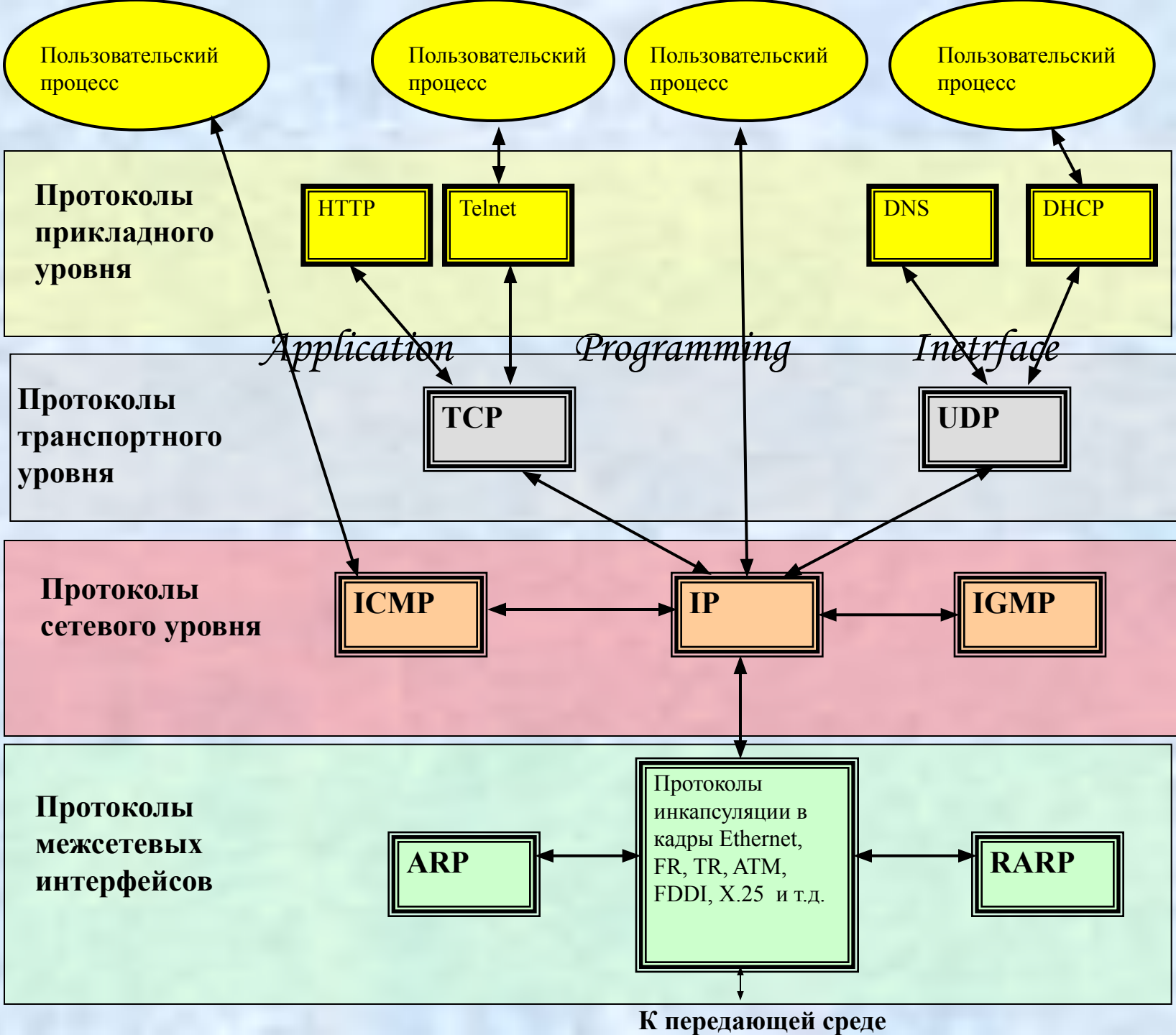
# Транспортный уровень стека протоколов TCP/IP


Пользовательский процесс
Пользовательский процесс
Пользовательский процесс
Пользовательский процесс
Протоколы прикладного уровня
HTTP
Telnet
DNS
DHCP
Application Programming Inetrface
Протоколы транспортного уровня
TCP
UDP
Протоколы сетевого уровня
ICMP
IP
IGMP
Протоколы межсетевых интерфейсов
ARP
Протоколы инкапсуляции в кадры Ethernet, FR, TR, ATM, FDDI, X.25 и т.д.
RARP
К передающей среде

# Дейтаграммный протокол UDP (RFC 768)

- Зарезервированные и доступные порты
- Мультиплексирование прикладных протоколов
- Формат дейтаграммы UDP

Задача протокола транспортного уровня
**UDP** (*User Datagram Protocol*) - передача данных

между прикладными процессами

без гарантий доставки

**Прикладной процесс в сети однозначно определяется**

- IP-адресом компьютера
- номером порта

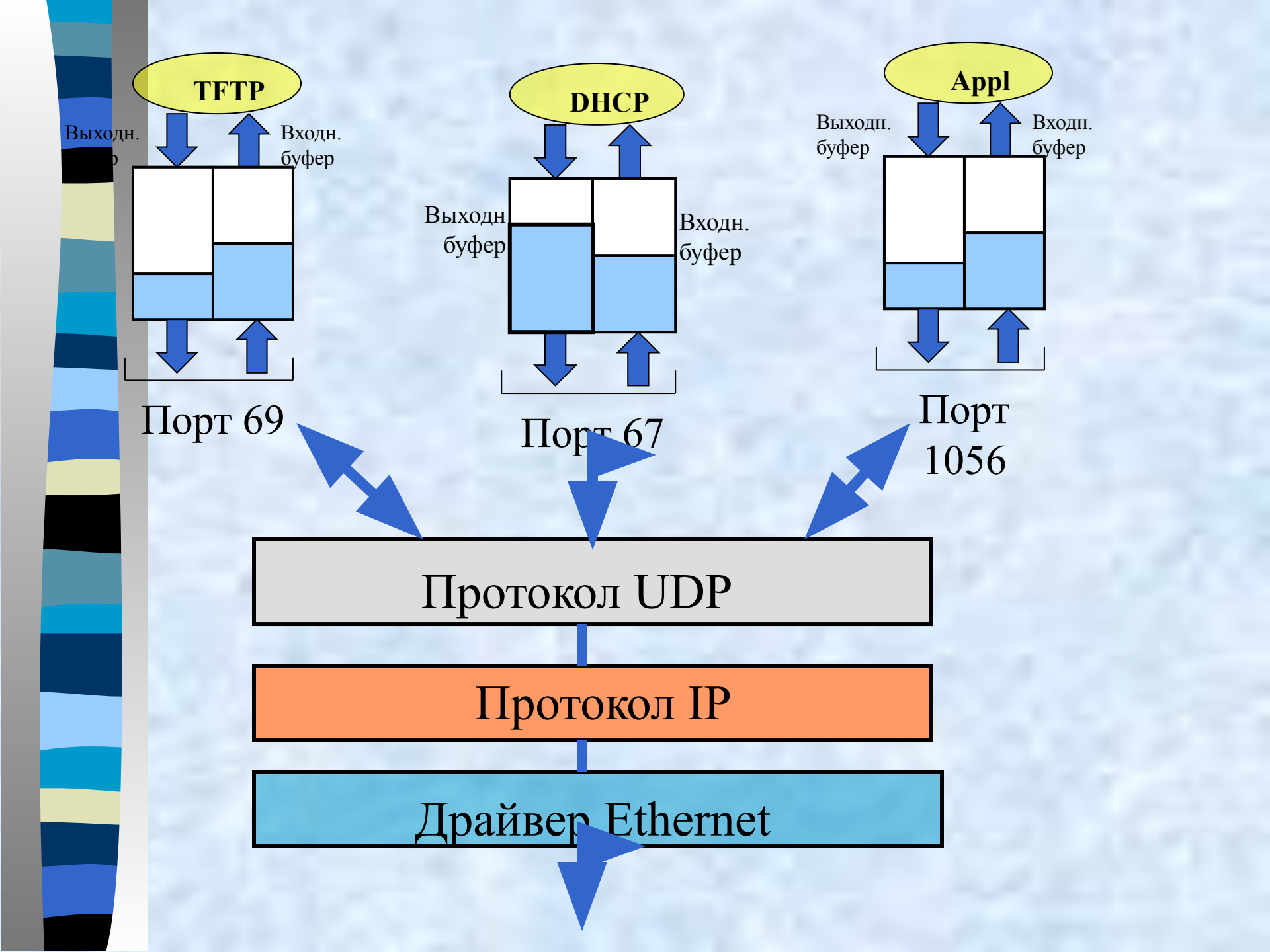
**Нет гарантий доставки** –

- дейтаграммный протокол,
- без установления соединений
- best effort

**Основная функция протокола UDP** – мультиплексирование и демультиплексирование процессов на основе портов

## Порт UDP

- идентификатор приложения
- определяет обменный буфер, создаваемый ОС в оперативной памяти
- если буфер переполняется, то сообщения отбрасываются


TFTP
Выходн.
Входн.
буфер
DHCP
Выходн
буфер
Входн.
буфер
Appl
Выходн.
буфер
Входн.
буфер
Порт 69
Порт 67
Порт
1056
Протокол UDP
Протокол IP
Драйвер Ethernet

# Назначение номеров портов прикладным процессам

1. **централизованное**

- для популярных сервисов - стандартные, зарезервированные номера в диапазоне 1-1023
- Internet Assigned Numbers Authority (IANA)
- Например: серверы TFTP - 69, DNS- 53, DHCP – 67, SNMP - 161
- Уникальны в пределах Internet

2. **локальное**

- для клиентских процессов
- выделяются операционной системой по запросу
- произвольные номера, обычно в диапазоне 1024-5000
- уникальны в пределах компьютера

Каждая дейтаграмма [illegible] переносит отдельное пользовательское сообщение

# Формат UDP-пакета (user datagram)

Длина заголовка 8 байт

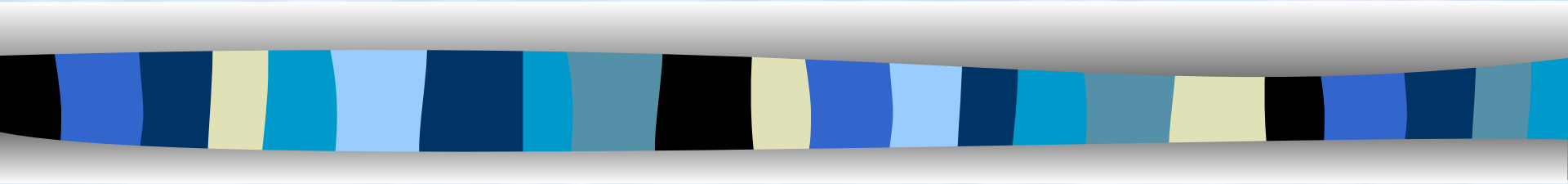
| 2 байта |
| --- |
| **UDP source port** - номер порта процесса-отправителя |
| **UDP destination port** - номер порта процесса-получателя |
| **UDP message length** - длина UDP-пакета в байтах |
| **UDP checksum -** контрольная сумма UDP-пакета |
| **Поле данных - IP-пакет** . . . |

Инкапсуляция дейтаграммы UDP


## Пример заголовка UDP с заполненными полями:

```
UDP: Source Port = 0x0035 (DNS)
UDP: Destination Port = 0x0411
UDP: Total length = 132 (0x84) bytes
UDP: UDP Checksum = 0x5333
UDP: Data: Number of data bytes remaining = 124
  (0x007C)
```

# Протокол надежной передачи данных TCP (RFC 793)



- Сравнение с UDP
- Порты,  сокеты, соединения
- Концепция скользящего окна
- Процедура установления соединения
- Процедура квитирования в TCP
- Адаптивный выбор тайм-аута
- Реакция на перегрузку

# Общая характеристика протокола TCP

*(Transmission Control Protocol)*

- транспортный уровень стека Internet
- обеспечивает надежную передачу
- основан на соединениях
- соединения не компьютеров, а **приложений**
- более медленный, чем UDP

Протокол TCP, в отличие от протокола UDP, не может быть использован для широковещательной и групповой передачи

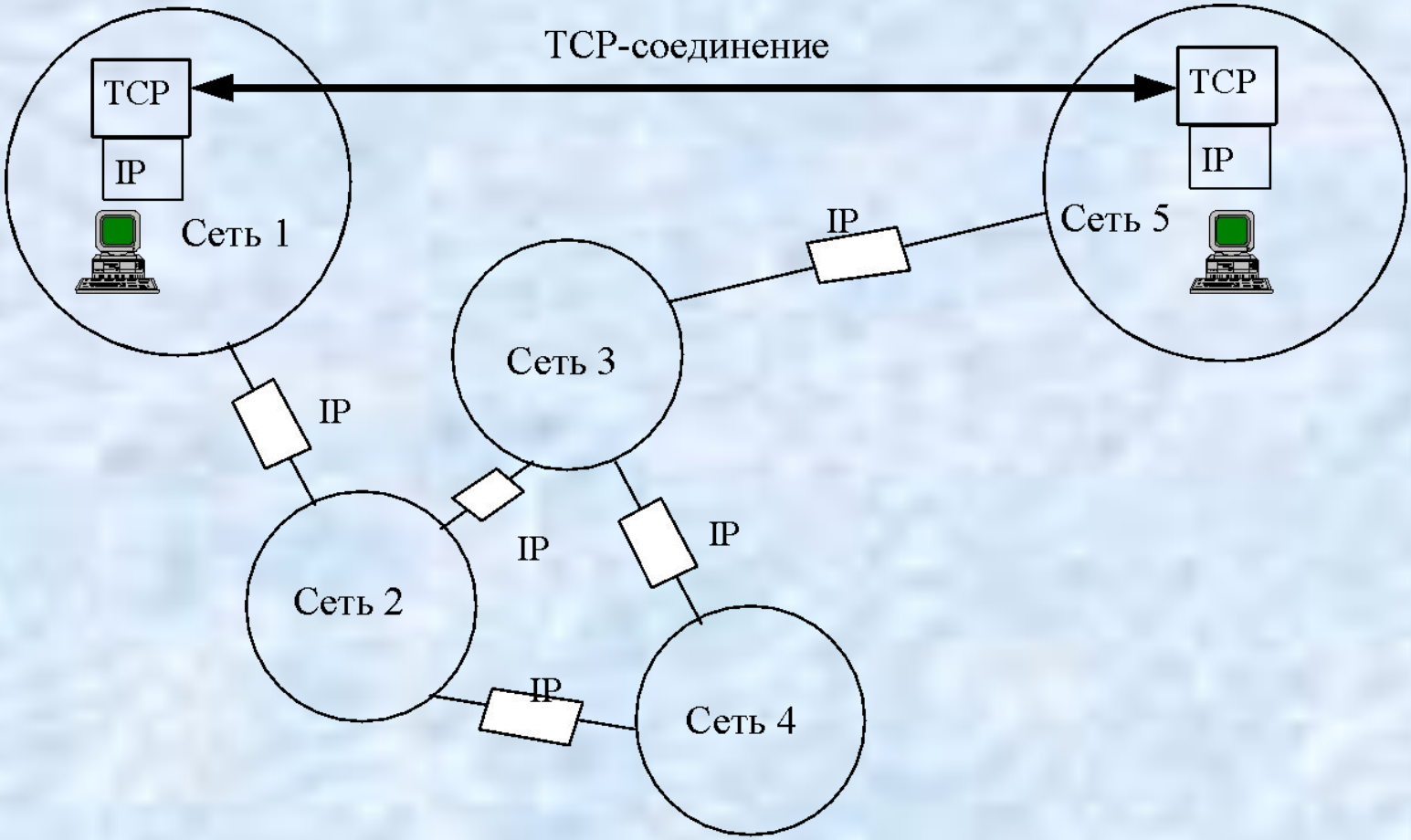
# Общая характеристика протокола TCP

*(Transmission Control Protocol)*

- транспортный уровень стека Internet
- обеспечивает надежную передачу
- основан на соединениях
- соединения не компьютеров, а **приложений**
- более медленный, чем UDP

Инкапсуляция сегмента TCP

# TCP-соединение создает надежный канал связи между конечными узлами



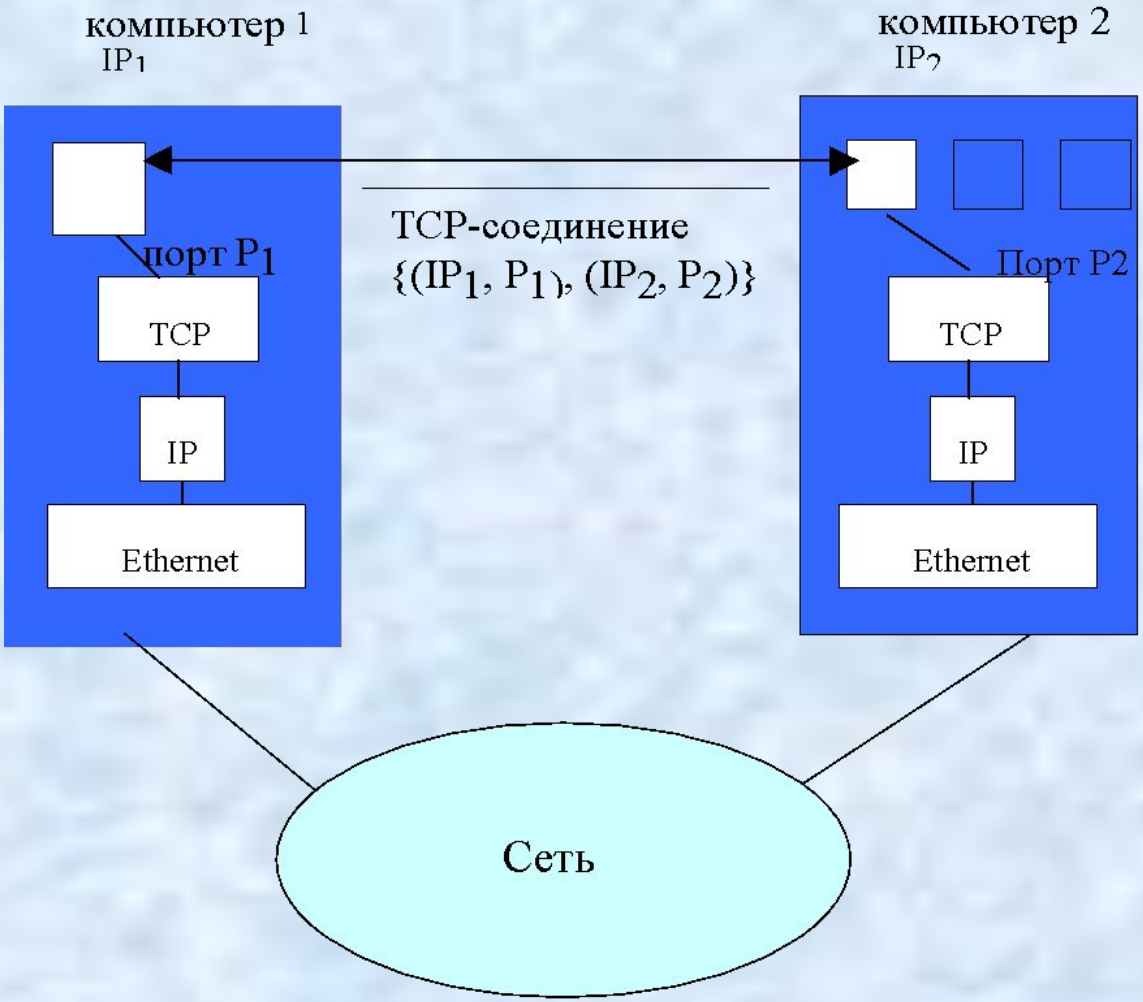
# Основные задачи протокола TCP

- передавать непрерывные потоки данных между клиентами в обоих направлениях
- обеспечивать защиту от разрушения данных, потери, дублирования и нарушения очередности получения - **нумерация, квитанции**
- управлять количеством данных, посылаемых ему отправителем - **окном**
- адресовать приложения - **номера портов**
- инициализировать и поддерживать определенную информацию о состоянии каждого потока данных - **соединениях**

# TCP-соединение

- между приложениями
- полнодуплексная передача
- согласованные параметры процедуры обмена:
  - номера портов
  - последовательные номера байт
  - размеры окон
- Соединение идентифицируется парой сокетов:
  - **Сокет** (IP, порт)
  - Соединение { (IP1, порт1) (IP2, порт2) }
  - Сокет может принимать участие во многих соединениях

# TCP-соединение



# Порты TCP

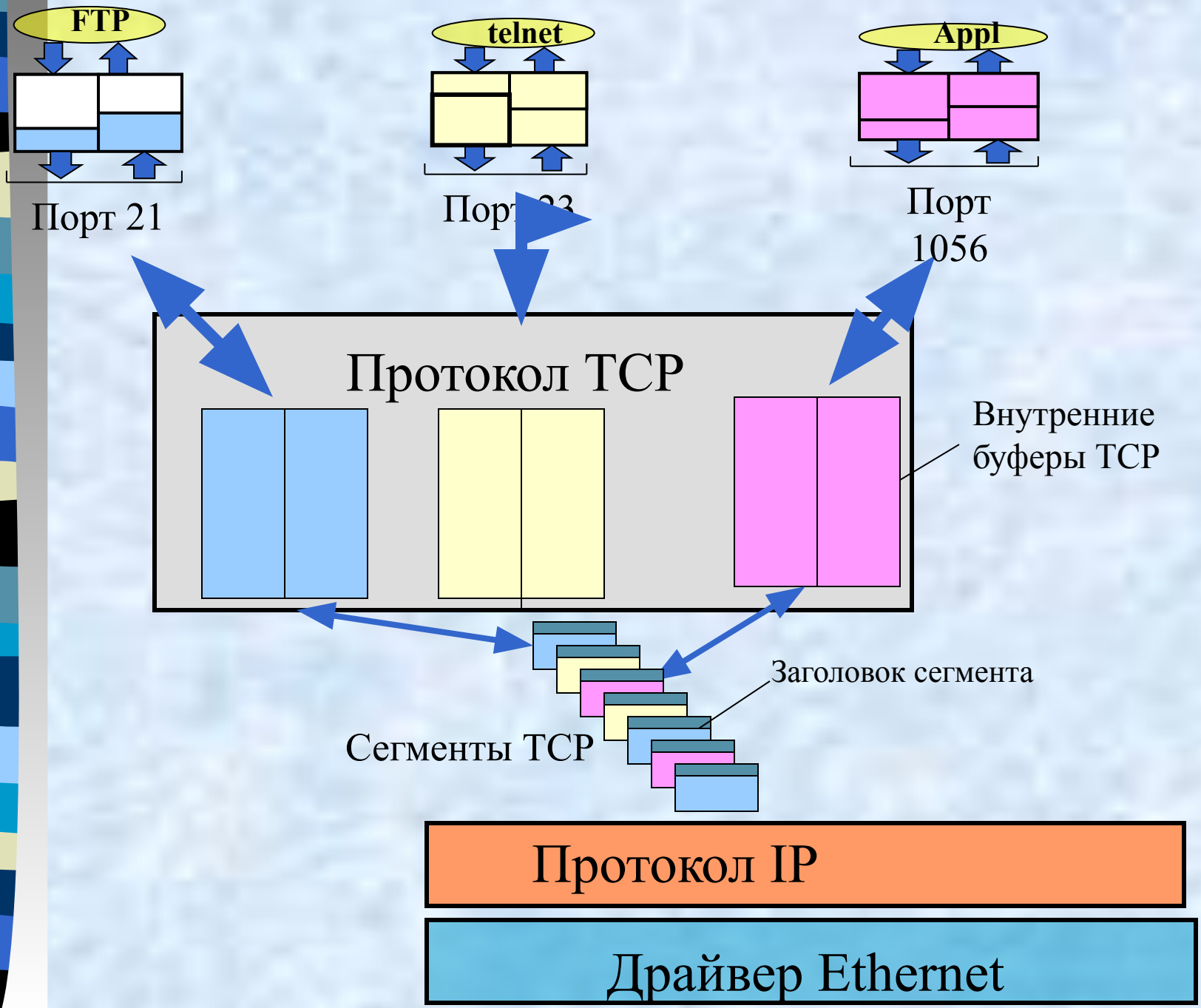
- **стандартные, зарезервированные номера** (до 1024, 21 – FTP, 23 – Telnet… ) ются Assigned Numbers Authority
- **произвольно выбранные локальные номера (1024 - 5000)**

# Сегменты TCP

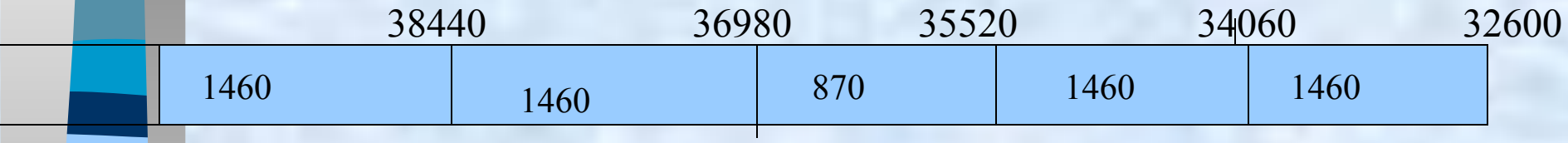
- На входе TCP - **поток** (неструктурированный поток байт от приложениий и протоколов более высокого уровня)
- На выходе TCP- **сегмент** (непрерывная часть потока)

## Максимальный размер сегмента

- ограничен стандартом
- должен быть определен при установлении соединения
- ограничен принятым в сети максимальным полем данных кадра (для исключения фрагментации на хосте)
- ограничен минимальным значением из множества MTU составной сети (для исключения фрагментации на шлюзах)


FTP
telnet
Appl
Порт 21
Порт
1056
Протокол TCP
Внутренние
буферы TCP
Заголовок сегмента
Сегменты TCP
Протокол IP
Драйвер Ethernet

Идентификатор сегмента – номер первого байта




- Протокол TCP может выжидать заполнения буфера перед отправкой сегмента.
- Приложение должно указать протоколу TCP, если требуется срочная передача – *параметр push*
- Приложение-отправитель должно указать протоколу TCP, если какие-то данные необходимо переслать приложению-получателю вне очереди – параметр *urgent data*

## Функция прот алкивания

- запрос на отправку сообщения без ожидания заполнения буфера
- поле *PSH* сегмента TCP = 1
- проталкивание не привязано к границам сегмента

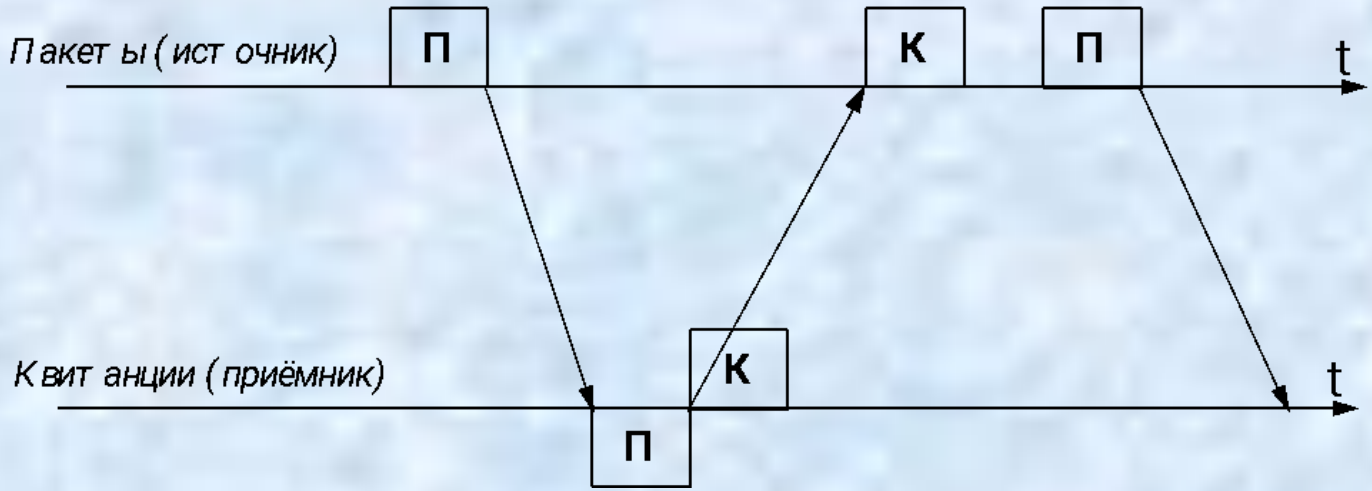
## Срочная передача

- Признак URG
- Указатель срочности Urgent Pointer
- Передача вне потока

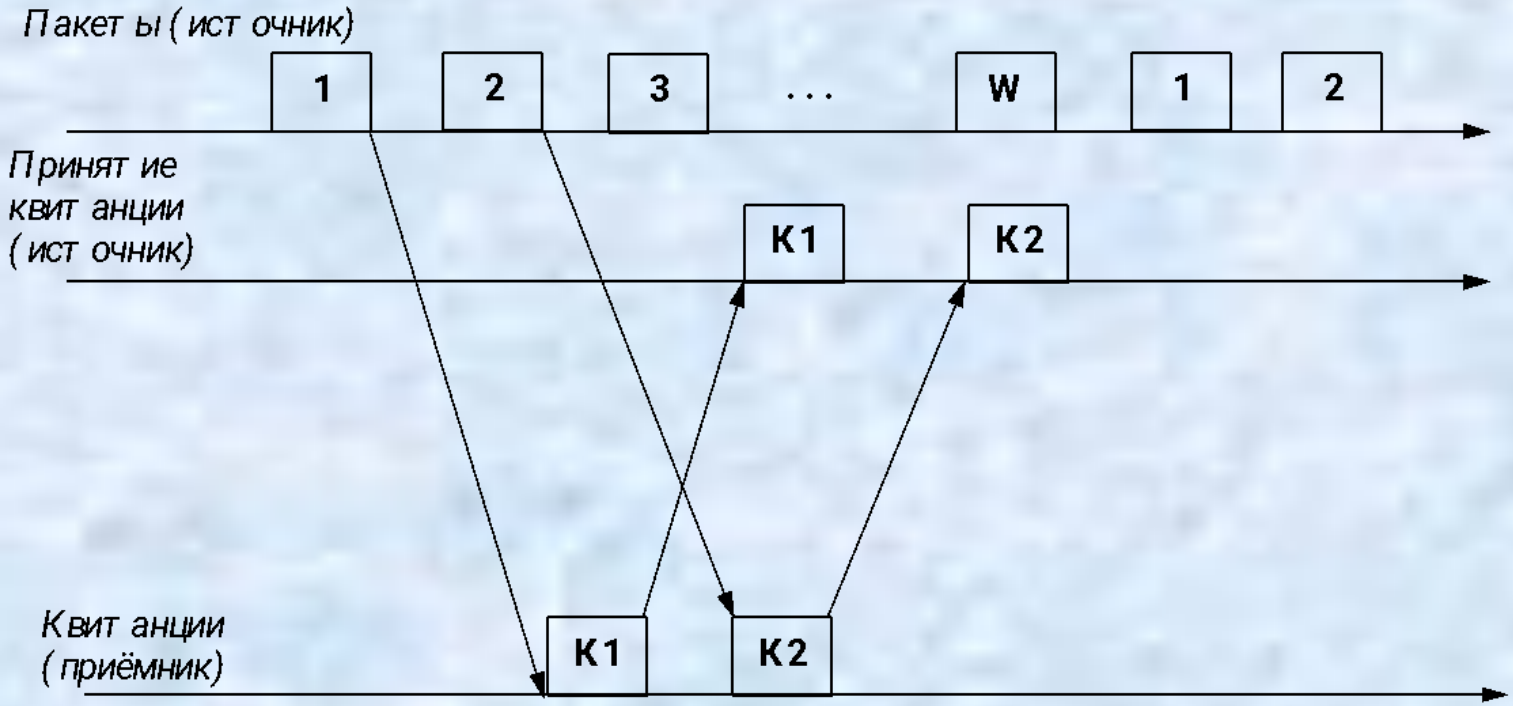
# Концепция квитирования

**В рамках соединения правильность передачи каждого сегмента должна подтверждаться квитанцией получателя (положительной или отрицательной)**

## Метод подтверждения корректности передачи с простоями источника

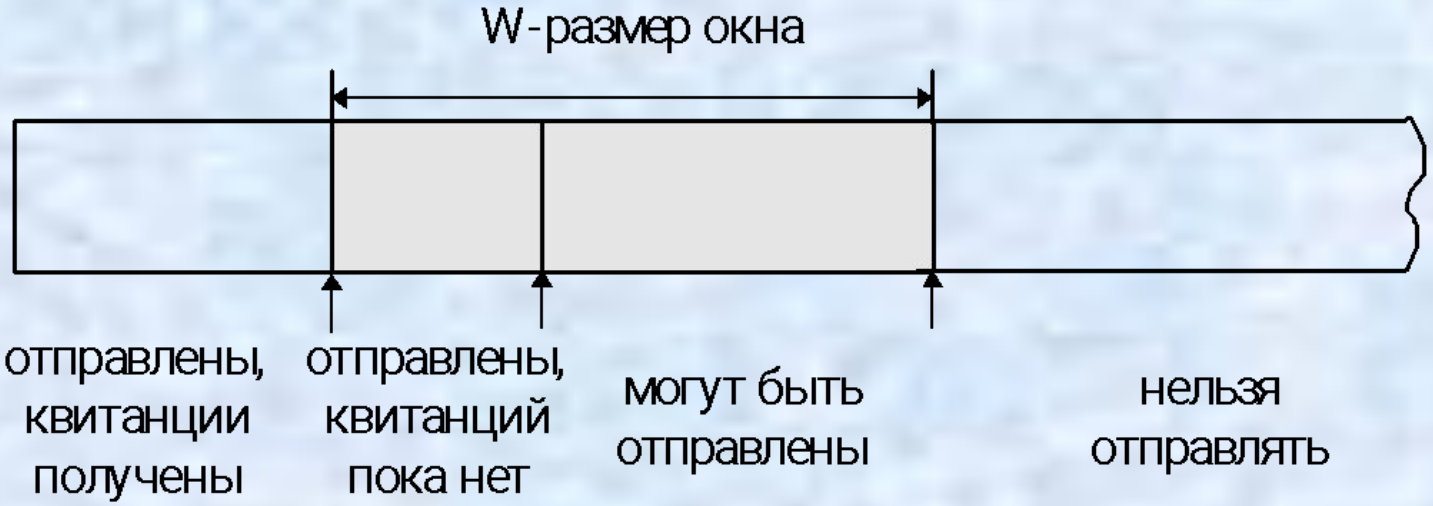


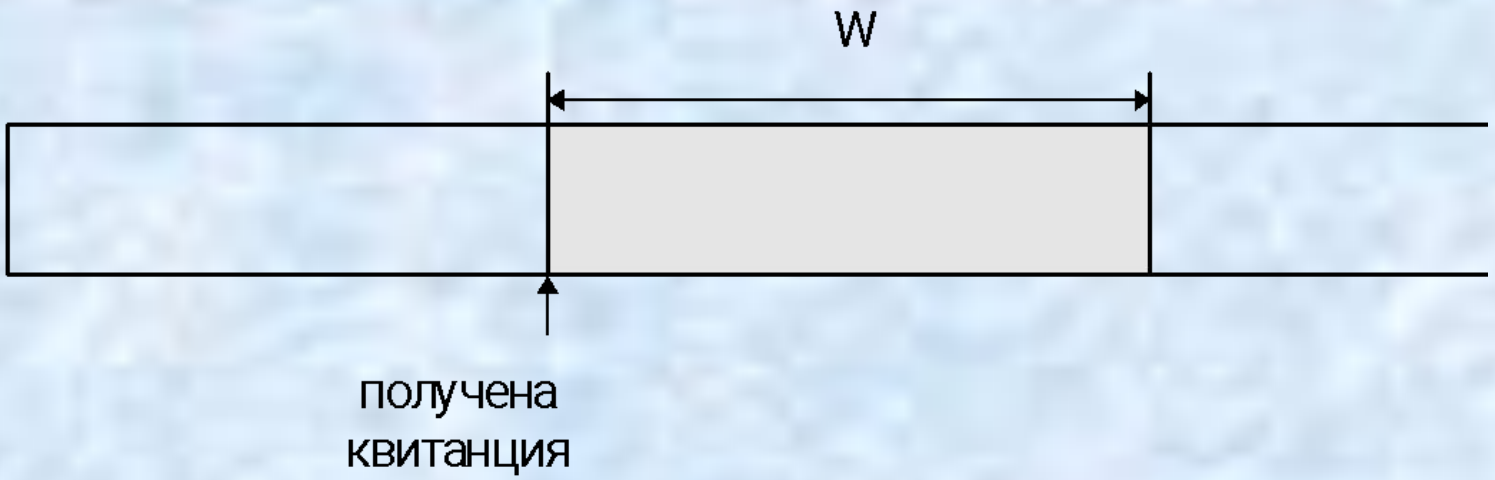
# Метод "скользящего окна"



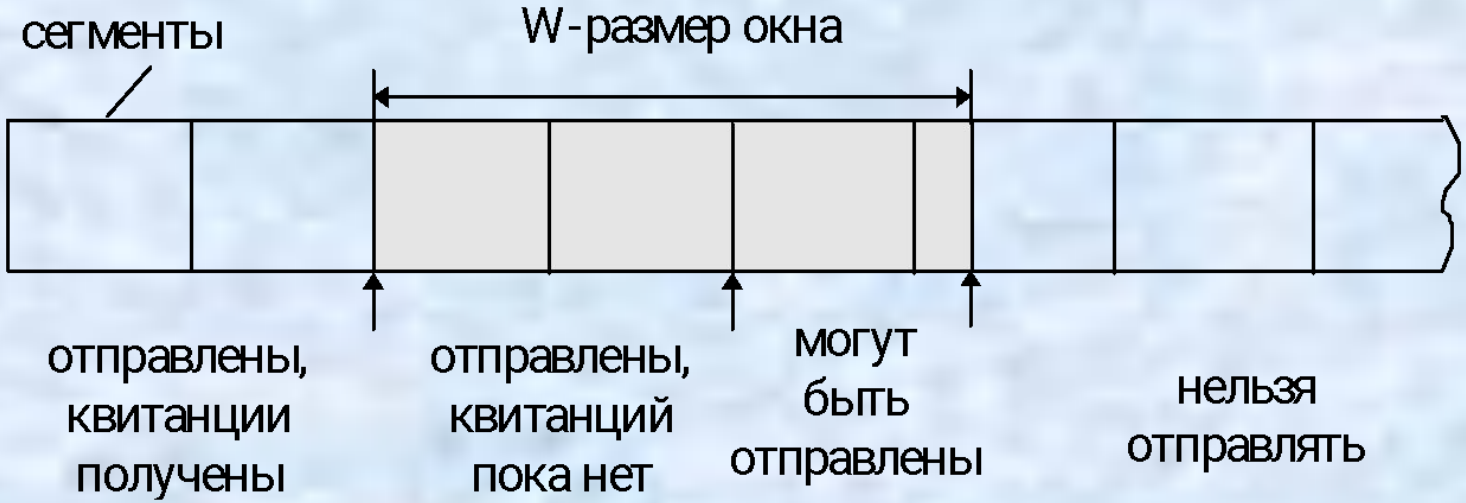

**W** - размер окна - количество кадров, которые разрешается передавать без получения квитанции

## Квитанция на каждый байт



## Квитинция на каждый сегмент, окно - в байтах



# Параметры управления потоком

- В каждом отправляемом сегменте каждая сторона обмена сообщает другой стороне размер своего окна

**ОКНО (win)** - количество байтов (начиная с номера подтверждения), которое программа TCP готова в настоящий момент принять

- При получении сегмента соответствующая сторона отсылает квитанцию - сегмент с подтверждением

**ПОДТВЕРЖДЕНИЕ (ack)**- число, на единицу превышающее максимальный номер байта в полученном сегменте

- В каждом сегменте отправитель помещает номер первого байта из отправляемых данный

**НОМЕР ОЧЕРЕДИ (seq)** - номер байта, который определяет смещение сегмента относительно потока отправляемых данных

# Фрагмент TCP-сеанса

```
len:    0, seq:    726000-727399, ack:2340403658,
win: 8760, src: 1044  dst:   20

len: 1460, seq:2340403658-2340405117, ack:
727400, win:17520, src:   20  dst: 1044

TCP: .A...., len:    0, seq:    727400-728799,
ack:2340405118, win: 8760, src: 1044  dst:   20
```

# Механизм подтверждения

- копия отправленного сегмента ставится в очередь повторной передачи и запускает таймер
- когда приходит подтверждение - сегмент удаляется из очереди
- если подтверждение не приходит до истечения срока, то сегмент посылается повторно
- механизм подтверждения является накопительным - подтверждение с номерами N<X номера X означает, что все байты уже получены
- возможно появление дубликатов в условиях повторной передачи

# Проблема совпадений номеров очереди

- при многократном повторном использовании одного и того же соединения
- количество номеров для очереди ограничено от 0 до $2^{32}$-1
  - генератор первоначальных 32 битных номеров очереди меняет значения каждые 4 микросекунды
  - полный цикл генератора составляет примерно 4.55 часа
  - при скорости 100 Мбит/с один цикл использования всех номеров очереди составляет 5.4 минуты
- Необходимость периода "молчания" после сбоя
- Первоначальный номер в очереди выясняется во время установления соединения

# Формат заголовка сегмента TCP

- *Порт источника* 2 байта, идентифицирует процесс-отправитель
- *Порт назначения* 2 байта, идентифицирует процесс-получатель
- *Последовательный номер* 4 байта, определяет смещение сегмента относительно потока отправляемых данных
- *Подтвержденный номер* 4 байта, максимальный номер байта в полученном сегменте, увеличенный на единицу
- *Длина заголовка* 4 бита, измеренная в 32-битовых словах
- *Резерв* 6 битов, поле зарезервировано
- *Кодовые биты* 6 битов
- *Окно* 2 байта, размер окна в байтах
- *Контрольная сумма* 2 байта
- *Указатель срочности* 2 байта, используется совместно с кодовым битом URG
- *Опции*
- *Заполнитель (PADDING)* фиктивное поле, используемое для доведения заголовка до целого числа 32-битовых слов

## Кодовые биты (CODE BITS)

1. **URG** - срочное сообщение

2. **ACK** - квитанция на принятый сегмент

3. **PSH** - запрос на отправку без ожидания заполнения

4. **RST** - запрос на восстановление соединения

5. **SYN** - сообщение используемое для синхронизации счетчиков переданных данных при установлении соединения

6. **FIN** - признак достижения передающей стороной последнего байта в потоке передаваемых данных

# Пример заголовка TCP-сегмента

```
TCP: Source Port = 0x0412
          TCP: Destination Port = FTP [control]
          TCP: Sequence Number = 695034 (0xA9AFA)
          TCP: Acknowledgement Number = 0 (0x0)
          TCP: Data Offset = 24 (0x18)
          TCP: Reserved = 0 (0x0000)
          TCP: Flags = 0x02 : ....S.
     TCP: ..0..... = No urgent data
     TCP: ...0.... = Acknowledgement field not significant
     TCP: ....0... = No Push function
     TCP: .....0.. = No Reset
     TCP: ......1. = Synchronize sequence numbers
     TCP: .......0 = No Fin
          TCP: Window = 8192 (0x2000)
          TCP: Checksum = 0x45C2
          TCP: Urgent Pointer = 0 (0x0)
          TCP: Options
        TCP: Option Kind (Maximum Segment Size) = 2 (0x2)
        TCP: Option Length = 4 (0x4)
        TCP: Option Value = 1460 (0x5B4)
```

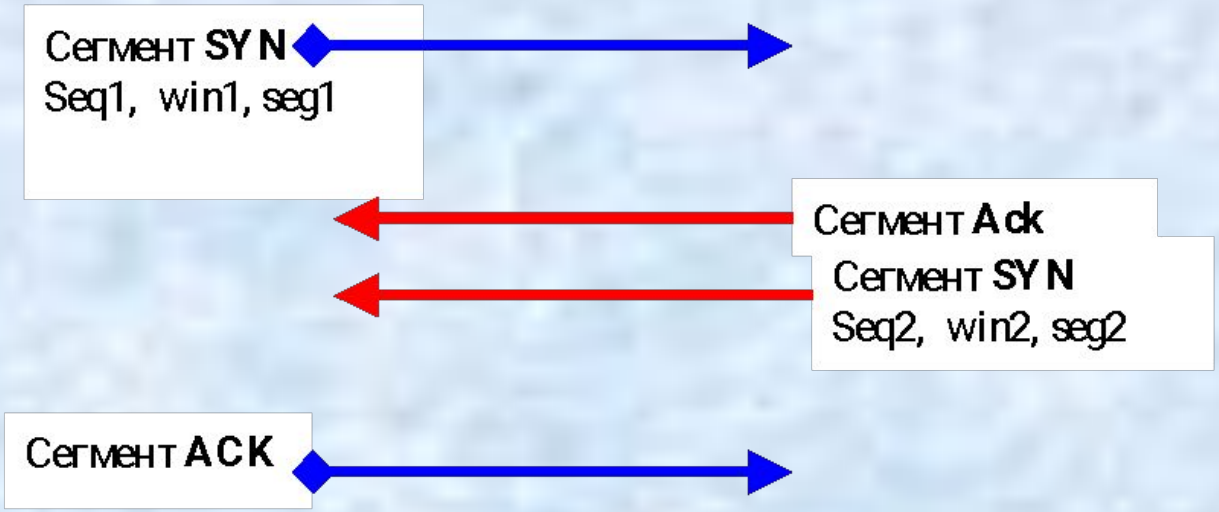
# Процедура установления соединения

- порт - ресурс: необходимо получить номер порта
- открытие соединения - команда OPEN (аргумент чужой сокет)
- ответ - имя локального (короткого) соединения
- запрос на пассивное открытие соединения - команда OPEN (аргумент неопределенный сокет) - процесс ждет получения запросов на соединение
- Отмена соединения - обмен сегментами с флагом FIN

# Процедура трехкратного подтверждения

Каждая сторона посылает:

- начальное значение номера очереди отправления seq
- подтверждение начального номера очереди напарника ack
- первоначальный размер окна win
- максимальный размер сегмента seg



# Алгоритм определения тайм-аута для повторной посылки сегментов (RFC793)

1. Измерение времени обращения RTT (*Round Trip Time*)

2. Вычисление усредненного времени обращения SRTT (*Smoothed Round Trip Time*):

$$SRTT = a * SRTT + (1-a) * RTT$$

3. Определение контрольного времени повторной посылки RTO (*Retransmission Timeout*):

$$RTO = \min[\ ubound, \max\{lbound, (b * SRTT)\ \}]$$

где

*ubound* - верхний предел контрольного времени (например, 1 мин),

*lbound* - нижний предел (например, 1 сек).

*a* - фактор сглаживания (например, от 0.8 до 0.9),

*b* - ...

# Управление окном. Реакция на перегрузку

99% потерь пакетов в Internet вызвано перегрузками и 1% - искажениями данных

***Приемы оптимизации:***

- при установлении соединения заявляется большое окно, но впоследствии его размер существенно уменьшается
- окно нужно уменьшать, когда свободный объем в буфере снижается 20-40% от максимально возможного объема
- отправителю не стоит спешить с посылкой данных, пока окно не станет достаточно большим
- при переполнение буферов в маршрутизаторах - централизованное изменение окна дифференцированно для всех конечных узлов
- при переполнении буфера конечного узла задается нулевое окно. В этом случае никакие сегменты приниматься не будут за исключением сегментов с флагами ACK , RST , URG

# Алгоритмы управления потоком данных в протоколе TCP

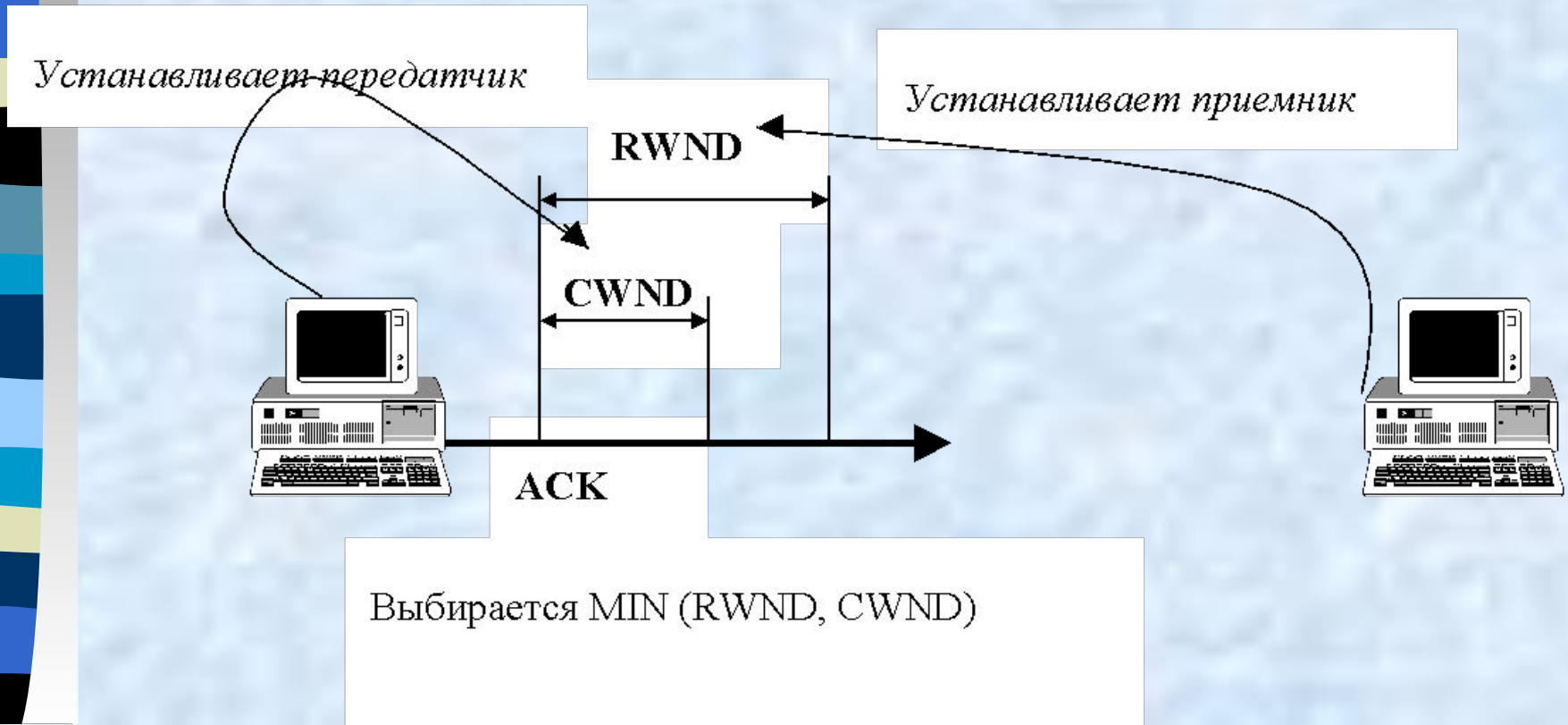
4 основных алгоритма для управления потоком данных при перегрузках в Internet (RFC 2581 – Proposed Standard):

1) Slow Start —" едленный старт"
2) Congestion Avoidance — "
3) Fast Retransmit — "
4) Fast Recovery — "

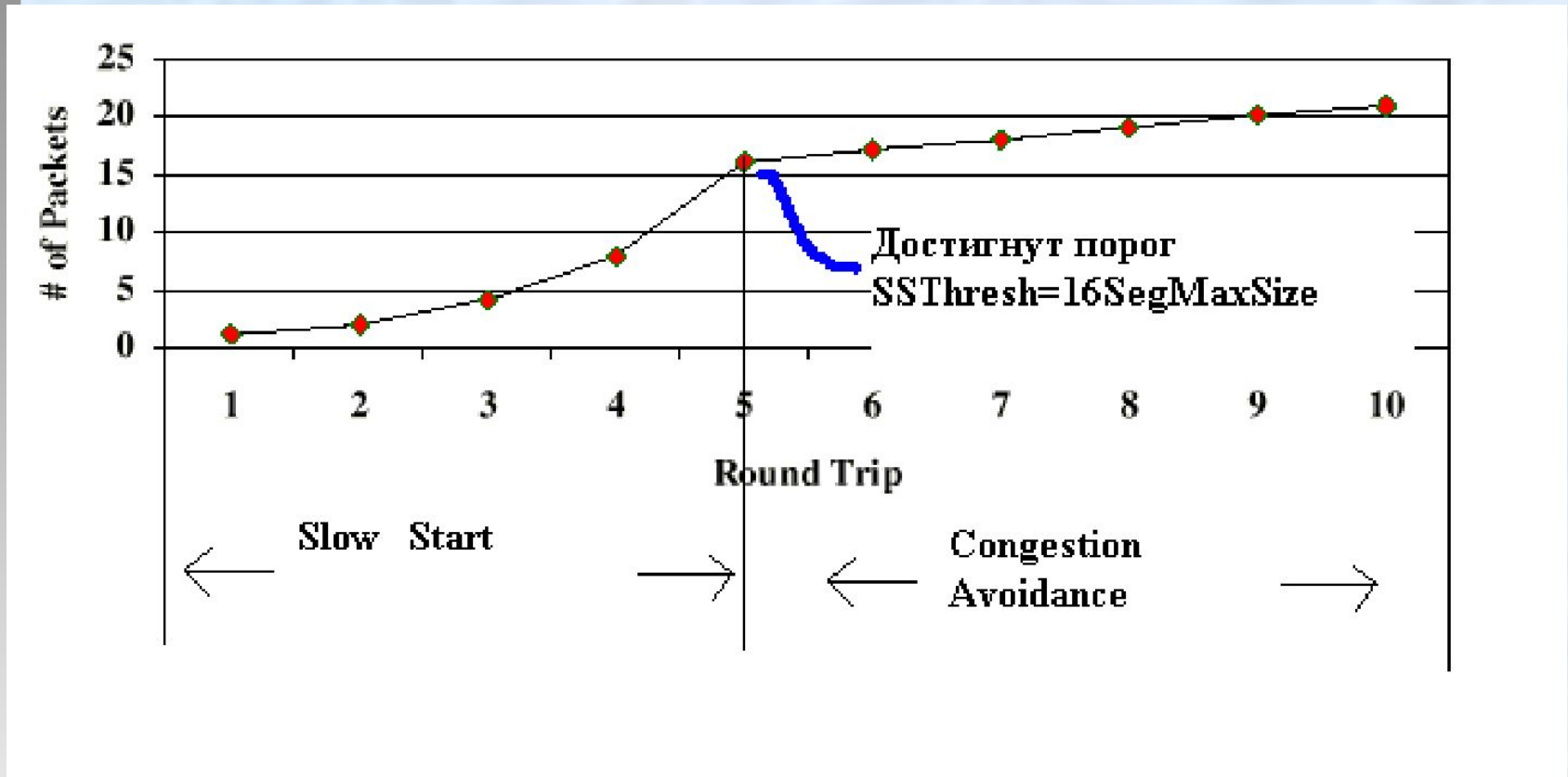
предупреждение перегрузок"
быстрый повтор"
быстрое восстановление"

Во многих ситуациях эти алгоритмы используются совместно

- Congestion Window, CWND — "окно перегрузок", устанавливаемое передатчиком
- Receiver Window, RWND – окно, объявляемое приемником



Slow Start —" едленный старт"
м
Congestion Avoidance — "
предупреждение перегрузок"



# Медленный старт

Применяется:

- в начале сессии TCP, когда условия загруженности сети еще не известны
- после потери пакета — по истечению таймера повторной передачи
- после длительного периода "молчания" сессии

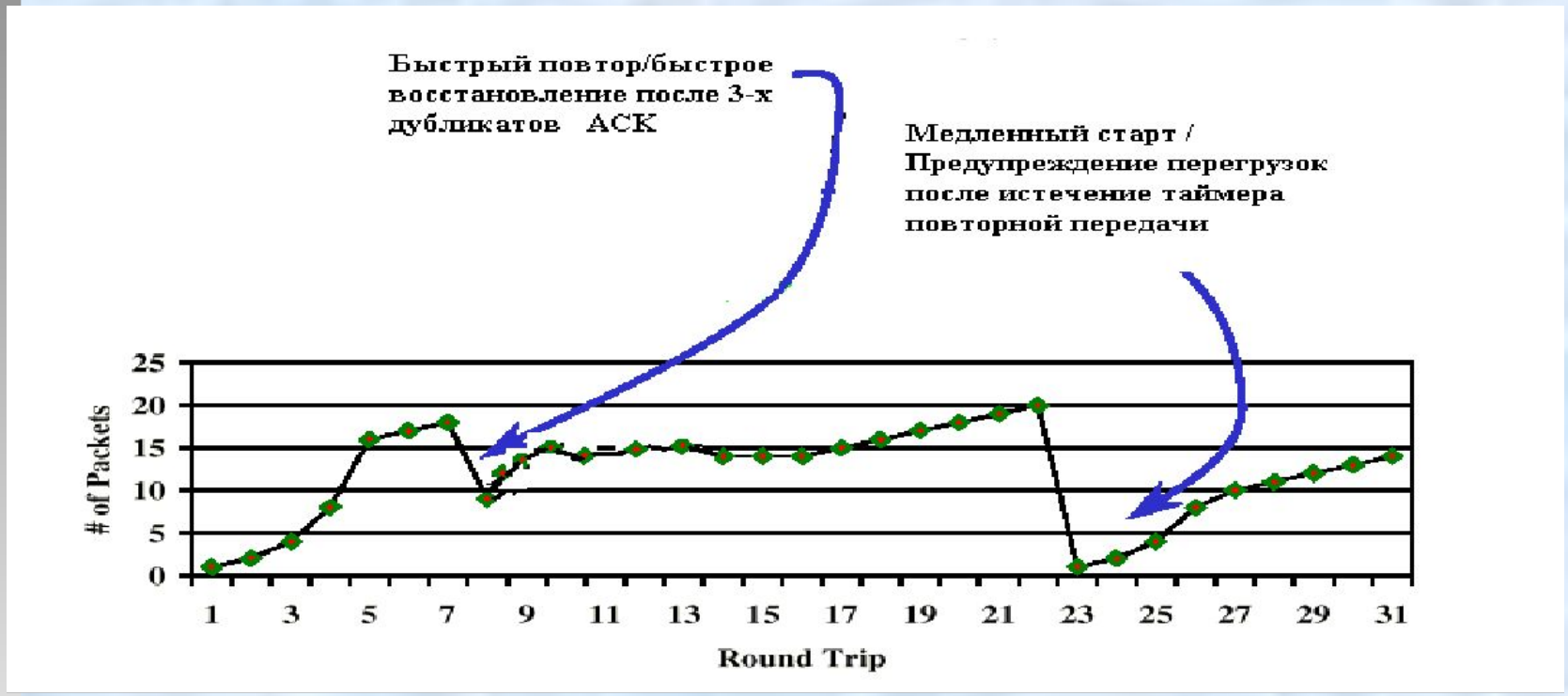
# Медленный старт

- Передающий узел использует окно CWND = 1 SegMaxSize и отправляет один сегмент в SegMaxSize байт
- При приходе очередной квитанции ACK окно CWND наращивается на SegMaxSize и отправляется столько сегментов, сколько разрешает CWND (но не больше RWND)
- Наращивание размера окна CWND происходит до порога SSThrash (Slow Start Thrashold) 65535 — дальше действие алгоритма Slow Start прекращается
- В результате размер окна и скорость передачи данных растут в геометрической прогрессии: 1 – 2 – 4 – 8 – 16 …

## Предупреждение перегрузок

- Применяется для осторожного повышения скорости передачи при выходе сессии в устойчивый режим
- Условия применения — размер окна CWND превышает порог SSTrash
- Передающий узел наращивает окно на 1 SegMaxSize каждые RRT секунд

" / "
"
"Быстрый повтор" Быстрое восстановление при получении 3 дубликатов ACK
старт" / "Медленный перегрузок" Предупреждение по истечение таймера повторной передачи



## " Медленный старт" при истечении таймера

1. Порог перехода на "Предупреждение перегрузок" уменьшается в 2 раза:

SSThresh = CWND/2

2. Окно CWND устанавливается в 1 SegMaxSize (как при старте сессии)

# Алгоритмы "Быстрый повтор" /"Быстрое восстановление"

Реакция передатчика на получение дубликатов квитанций ACK — потеря пакета только подозревается

1. При получении 4-х ACK подряд передатчик **не ждет** истечения таймера и посылает повторно сегмент, который ждет передатчик (возможно утерянный)

2. Порог SSThresh уменьшается до CWND/2

3. Окно CWND уменьшается только до SSThresh + 3*SegMaxSize -> окно учитывает, что 3 пакета уже получены приемником

4. CWND
Передается столько пакетов, сколько разрешает окно

5. При получении первого "нового" ACK окно уменьшается до SSThrash