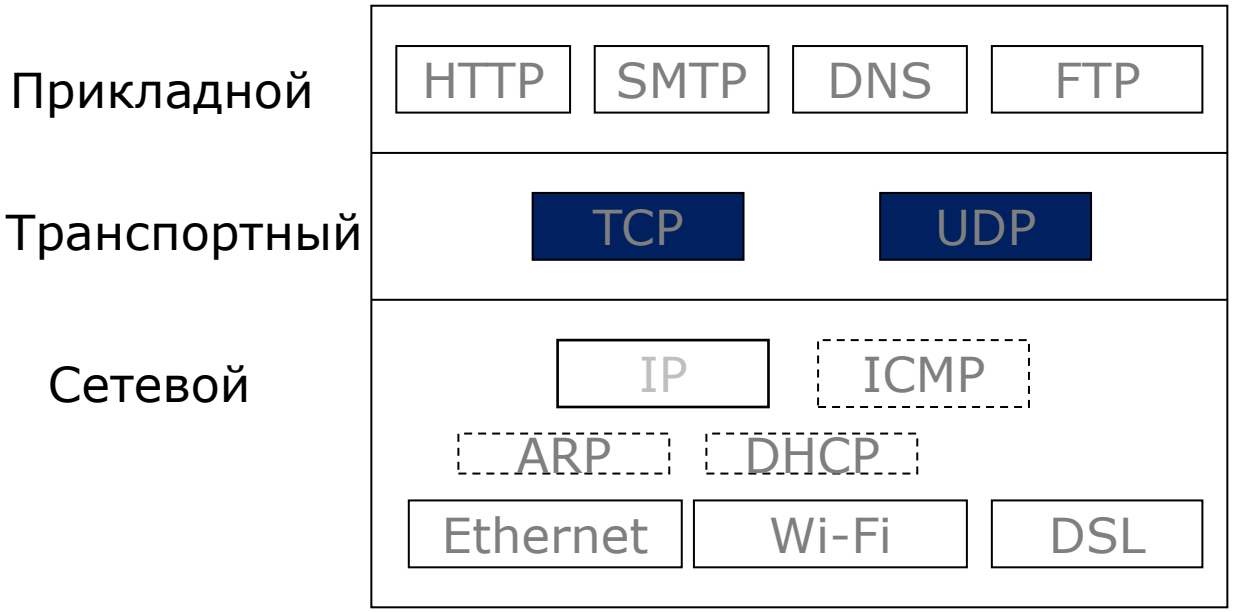
ETU "LETI"
SAINT PETERSBURG ELECTROTECHNICAL UNIVERSITY

ОСНОВЫ ПОСТРОЕНИЯ ЗАЩИЩЕННЫХ КОМПЬЮТЕРНЫХ СЕТЕЙ

Сетевые протоколы передачи информации. Протокол TCP

Модель TCP/IP

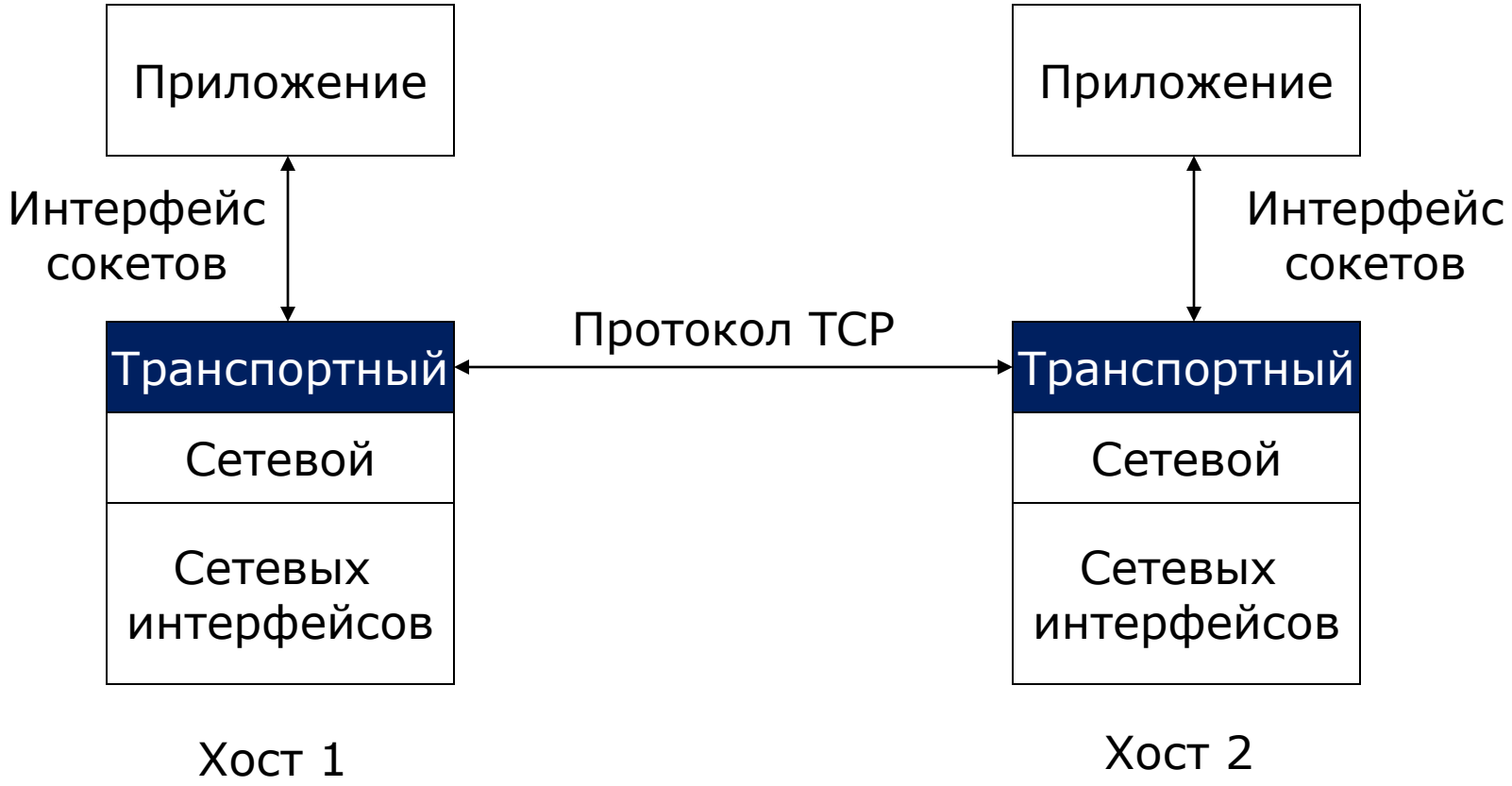


Задачи TCP

- Предварительная установка соединения
- Гарантия доставки данных за счет подтверждение получения
- Повторная передача

Модель TCP/IP

Сокеты, TCP и UDP относятся к транспортному уровню
Сокеты – интерфейс транспортного уровня
TCP и UDP – протоколы транспортного уровня



Модель TCP/IP. Сокеты

Сокеты впервые появились в ОС Berkeley UNIX 4.2 BSD (1983 г.)

Сокеты – де-факто стандарт интерфейсов для транспортной подсистемы.

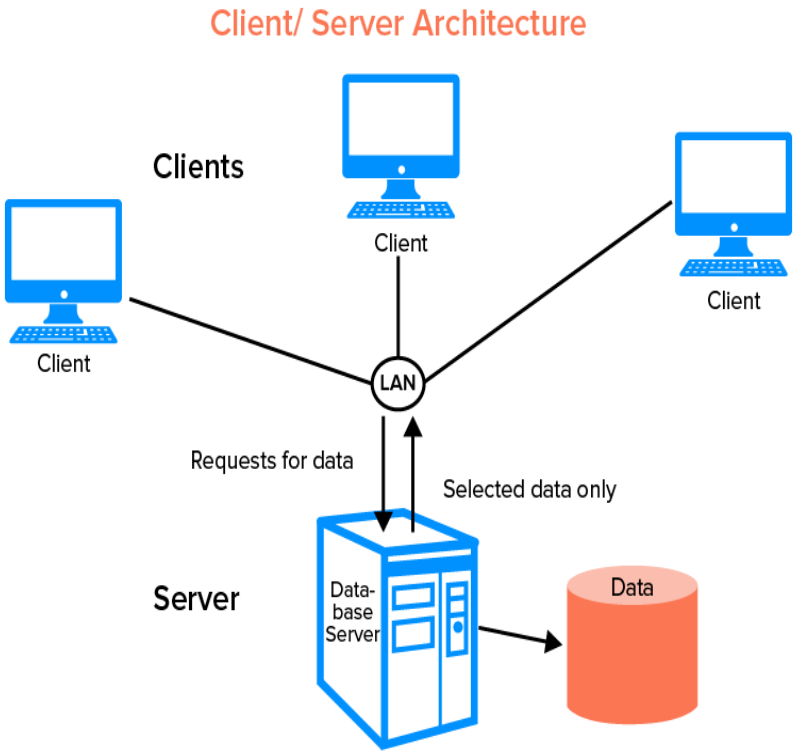
Адрес задаётся комбинацией IP-адреса и 16-битного номера порта.

Операции чтения и записи в файл сокета передают данные по сети

- Отправитель записывает данные в файл сокета
- Получатель читает данные их файла сокета
- Передача данных по сети скрыта от программиста

Клиент- серверное соединение

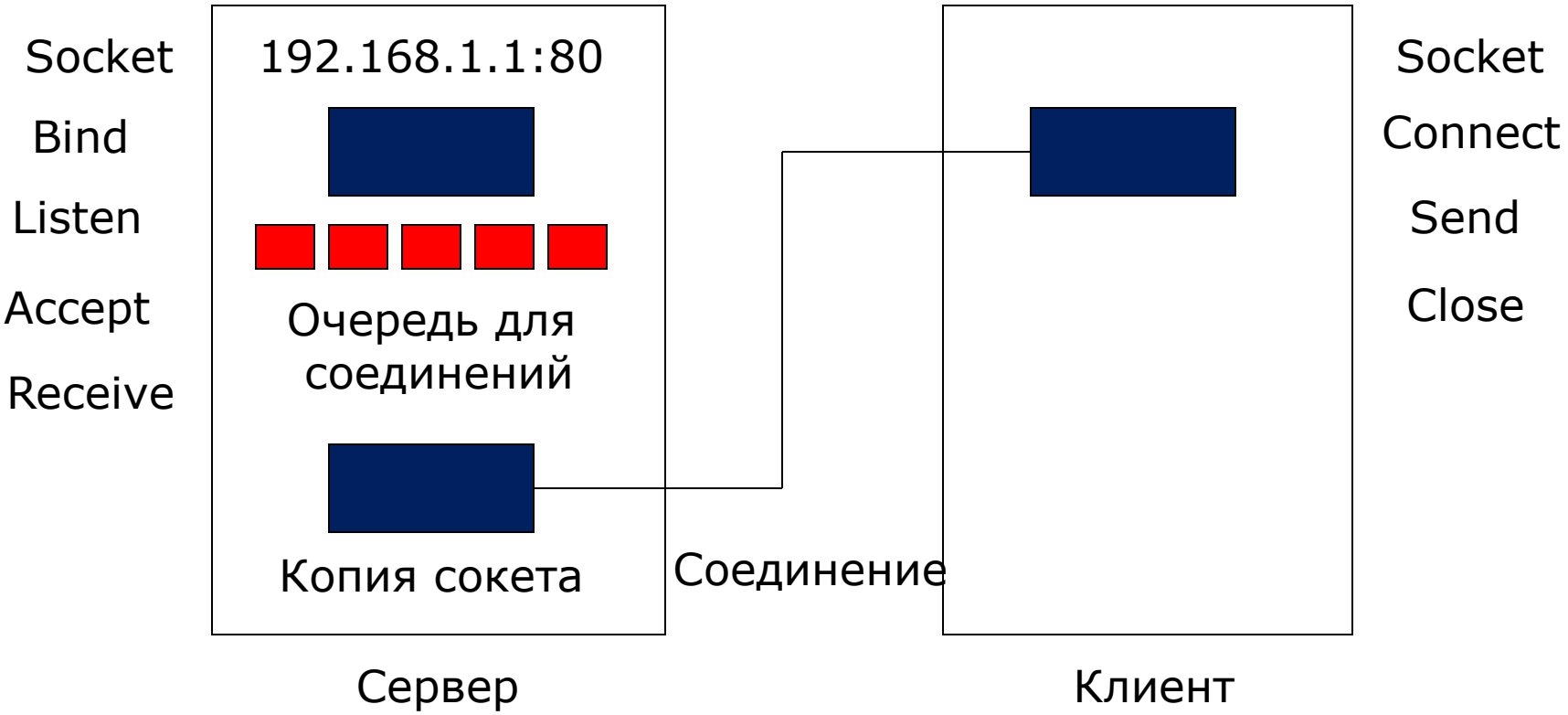
- Передача данных возможна, только если установлено соединение клиент-сервер.
- Сервер – работает (слушает) на известном IP-адресе и пассивно ждет запросов на соединение
- Клиент – активно устанавливает соединение с сервером на заданном IP + порт.
- При получении запроса от клиента создается копия сокета.
- Соединение устанавливается с копией, оригинальный сокет продолжает ждать запросы от других клиентов
- Такой сокет не может принимать и передавать данные.



Операции сокетов Беркли

Типы Операций	Назначение
Socket	Создать новый сокет (файла)
Bind	Связать сокет с IP-адресом и портом
Listen	Объявить о желании принимать соединения
Connect	Установить соединение
Accept	Принять запрос на установку соединения
Send	Отправить данные по сети
Receive	Получить данные из сети
Close	Закрыть соединение

Сокеты Беркли



Пример реализации сокета на Python

1. Серверный» сокет
2. Клиентский сокет
3. Передача данных

- Протоколы сетевого уровня
  - socket.AF_INET – IPv4
  - socket.AF_INET6 – IPv6

- Протоколы транспортного уровня
  - socket.SOCK_STREAM – TCP
  - socket.SOCK_DGRAM – UDP

Пример реализации сокета на Python

1. Создание сокета:

```
serversocket = socket.socket( socket.AF_INET,
    socket.SOCK_STREAM)
```

2. Привязка к IP-адресу:

```
serversocket.bind(('192.168.0.1', 80))
```

3. «Прослушивание»:

```
serversocket.listen(5)
while 1:
    #Принимаем соединение – создаем клиентский сокет
    (clientsocket, address) = serversocket.accept()

    #Создаем поток для обслуживания запроса клиента
    ct = client_thread(clientsocket)
    ct.run()
```

Пример реализации клиентский сокета на Python

1. Создание сокета:

```
clientsocket = socket.socket(socket.AF_INET,
  socket.SOCK_STREAM)
```

2. Установка соединения:

```
clientsocket.connect(('192.168.0.1', 80))
```

3. Отправка данных:

```
socket.send('Hello, world!')
```

4. Получение данных:

```
socket.recv(1024)
# 1024 – размер буфера
```

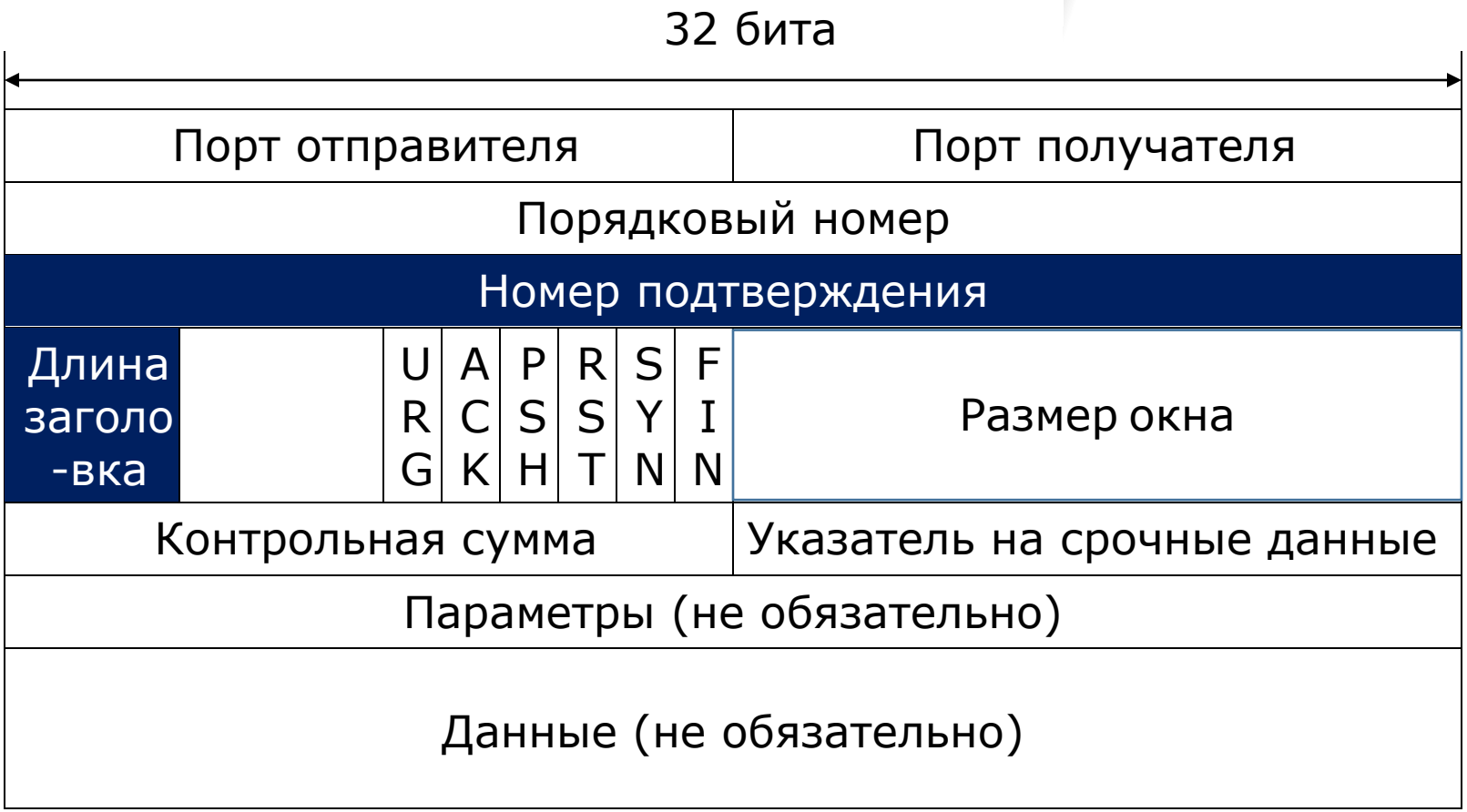
5. Закрытие сокета:

```
socket.close
```

Сокеты. Безопасность соединения

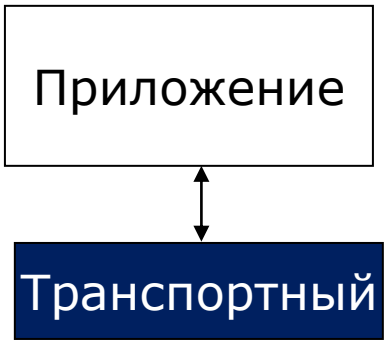
1. Возможность установить соединение между доменами, не спрашивая разрешения у пользователя.
2. Запрос посылается без уведомления пользователя с помощью некоторого скипта.
3. Не все прокси-серверы корректно понимают протокол веб-сокетов, атакующий может заставить веб-прокси кэшировать навязанные данные.
4. Возможность реализации атаки через посредника.

Безопасность TCP. Заголовок TCP



Соединение TCP. Поток байт

- Транспортная подсистема получает от приложения данные в виде потока байт
- Поток разбивается на отдельные части – сегменты
- Протокол TCP нумерует байты в потоке
  - Сегменты не нумеруются



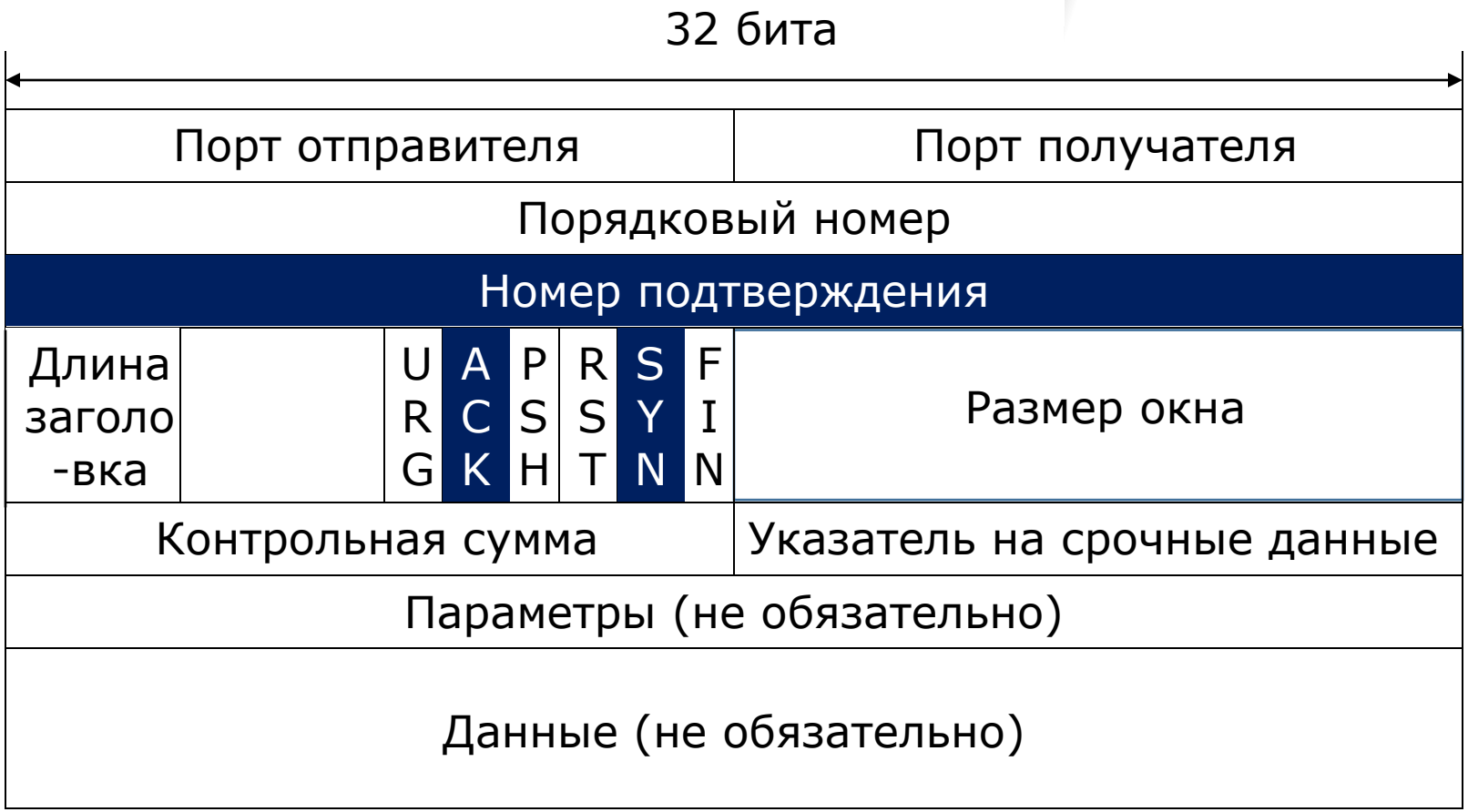
Поток байт от приложения

Сегмент	Сегмент	Сегмент	Сегмент
Байт 0	Байт 1024	Байт 2048	Байт 3072

Гарантия доставки

- Возможные проблемы при доставке:
  - Потеря сегментов
  - Изменение порядка доставки сегментов
  - Повторная доставка сегментов
- Сервис TCP:
  - Гарантия доставки
  - Гарантия сохранения порядка следования сообщений
- Механизмы реализации:
  - Нумерация сообщений
  - Подтверждение получения сообщения
  - Повторная отправка при отсутствии подтверждения

Безопасность TCP. Заголовок TCP



Установка соединения TCP

- Соединение – договоренность между отправителем и получателем о передаче данных
- Соединение задает:
  - Начальные номера для нумерации данных отправителя и получателя
  - Параметры передачи: максимальный размер сегмента и т.п.
  - Объем данных, которые готов принять получатель
- Соединение в TCP дуплексное
  - Данные могут передаваться в обе стороны
  - Подтверждение получения и данные в одном сегменте.

Установка соединения TCP

- TCP использует схему «Трехкратного рукопожатия»
- Флаг SYN – признак установки соединения
  - SYN = 1, ACK = 0 – запрос установки соединения (CONNECTION REQUEST)
  - SYN = 1, ACK = 1 – подтверждение установки соединения (CONNECTION ACCEPT)
  - SYN = 0, ACK = 1 – завершение установки соединения
- ACK – флаг подтверждения
  - Если флаг ACK установлен, значит поле «Номер подтверждения» содержит осмысленные данные

Трехкратное рукопожатие

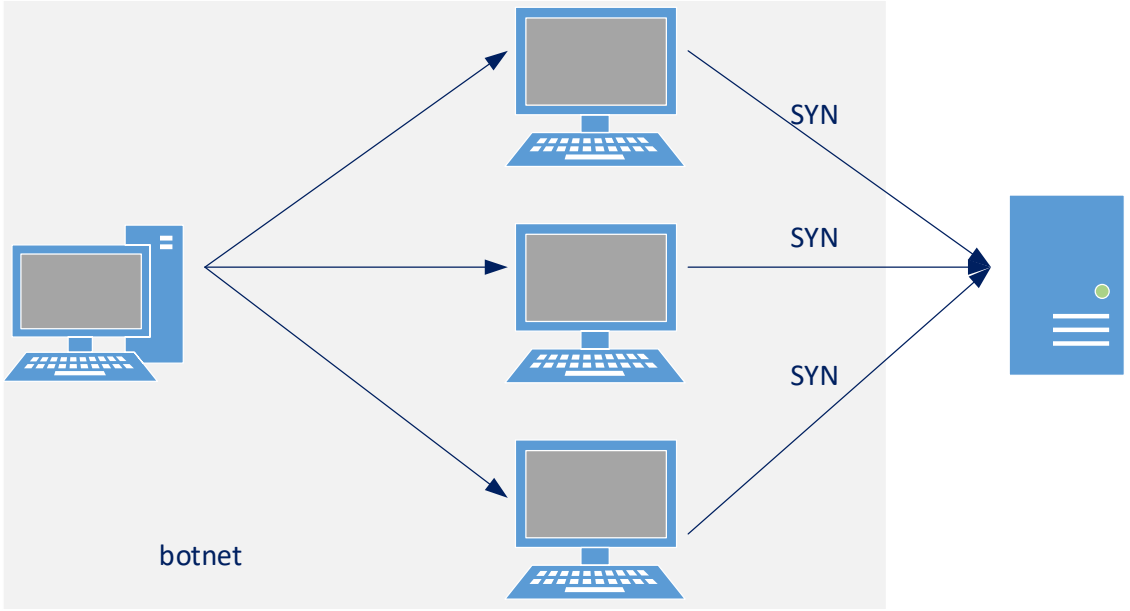
Разрыв соединения TCP

- Соединение в TCP дуплексное
  - Данные могут передаваться в обе стороны
- Схема разрыва соединения
  - Одновременное (обе стороны разорвали соединение)
  - Одностороннее (сторона прекращает передавать данные, но может принимать)
- Флаг FIN – одностороннее закрытие соединение
  - Соединение закрывается, когда обе стороны отправят сегмент с установленным флагом FIN и подтверждение
- Флаг RST – разрыв соединения из-за критической ситуации
  - Одновременный разрыв соединения обеими сторонами

Проблема безопасности TCP

1. Затопление с Ddos
2. Затопление при перегрузке

3. Сканирование с установлением флагаFIN, SYN, RST, PSH,URG



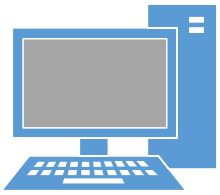
Безопасность TCP. Таймеры TCP

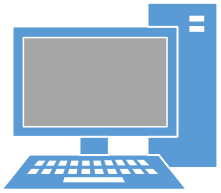
- Таймер повторной передачи:
  - Время ожидания подтверждения получения сегмента.
  - Если подтверждения нет, сегмент отправляется вновь.
- Таймер проверки активности:
  - Используется при длительном простое соединения.
  - Задает время, через которое должна выполниться проверка работоспособности соединения.
- Таймер закрытия соединения:
  - Задает ожидание, равное двойному времени жизни сегмента.
  - За это время все сегменты соединения должны уйти из сети.

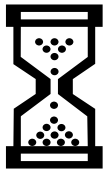
Безопасность TCP. Таймеры TCP

Отправитель

Получатель





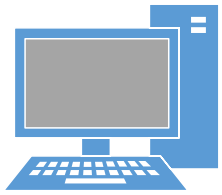


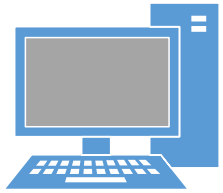
Буфер

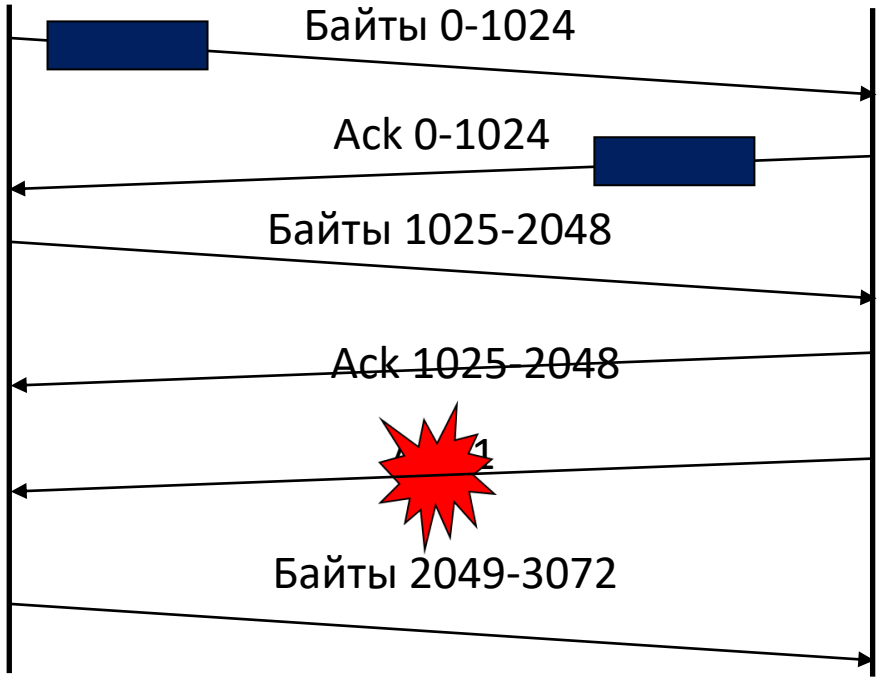
Безопасность TCP. Таймеры TCP

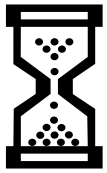
Отправитель

Получатель









Буфер

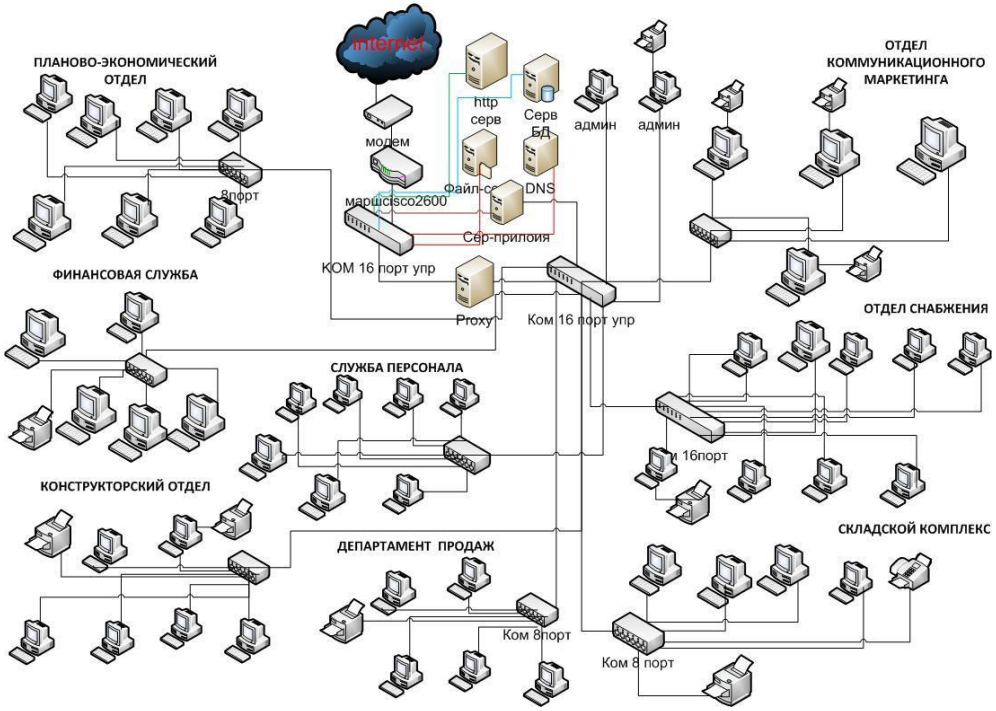
Безопасность TCP. Затопление сети

Сообщение передается не мгновенно:

- Время передачи короткое, но не нулевое

В среде может «находиться» некоторый объем данных:

- Скорость × Задержка
- Небольшой объем для локальных сетей
- Большой объем для широких территориально-протяженных каналов. Ожидание подтверждения приводит к снижению производительности

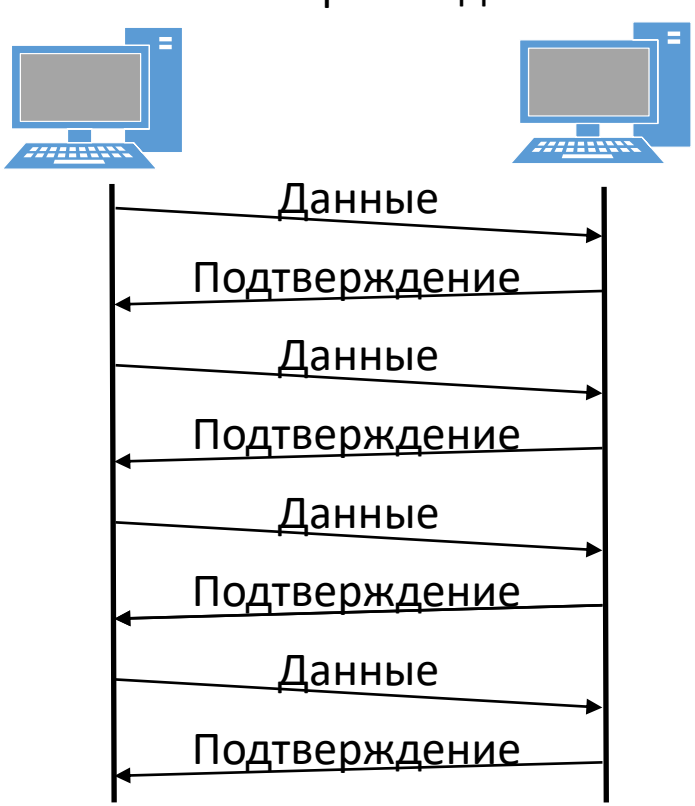


Подтверждение о доставке

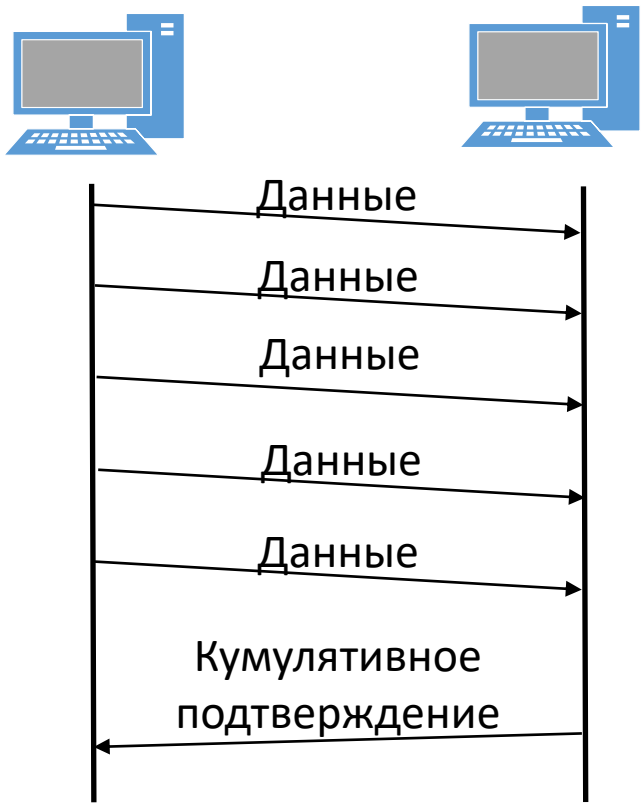
TCP должен обеспечивать

- Гарантированную доставку данных
- Эффективное использование канала связи

- TCP должен хорошо работать как на медленных каналах связи с ошибками, так и на быстрых надежных каналах

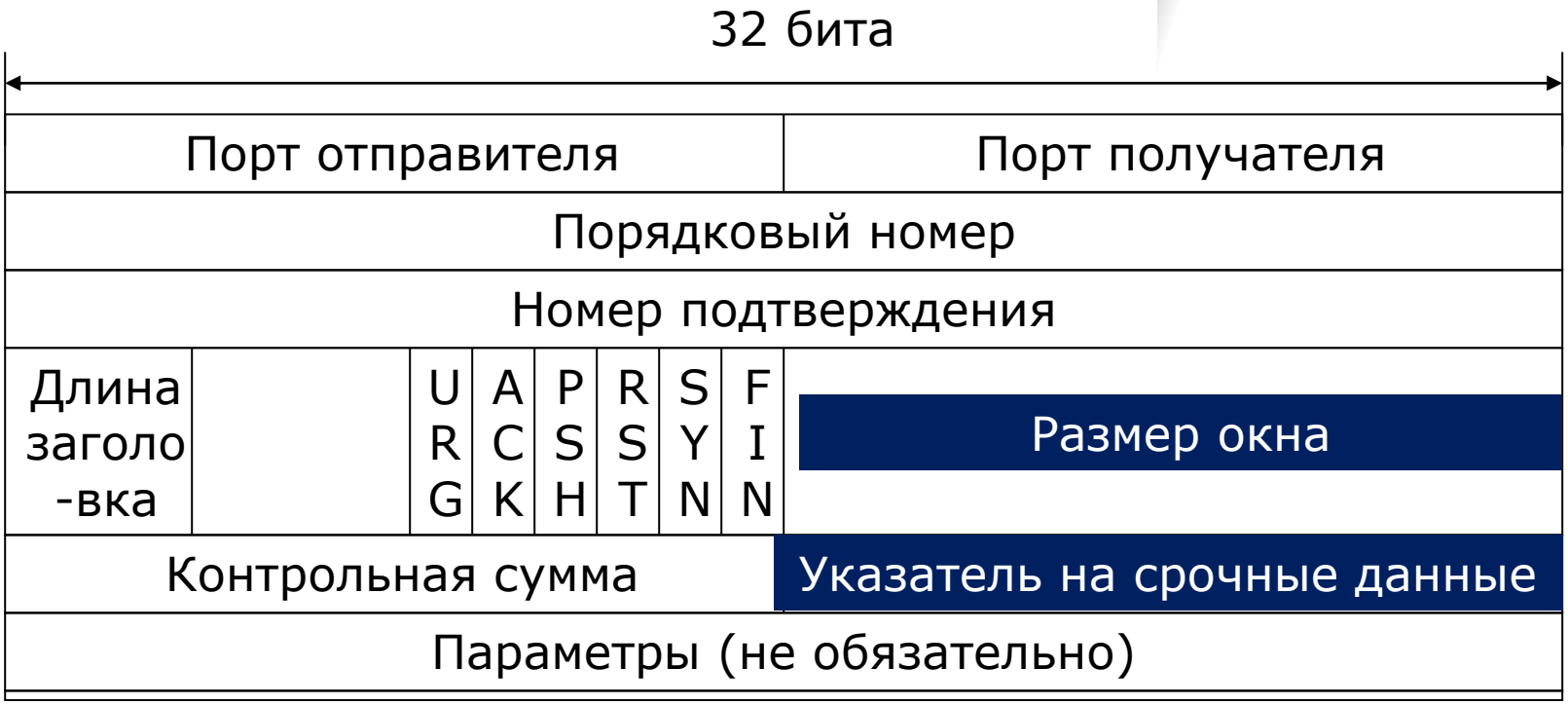


Остановка и ожидание



Скользящее окно

Подтверждение о доставке

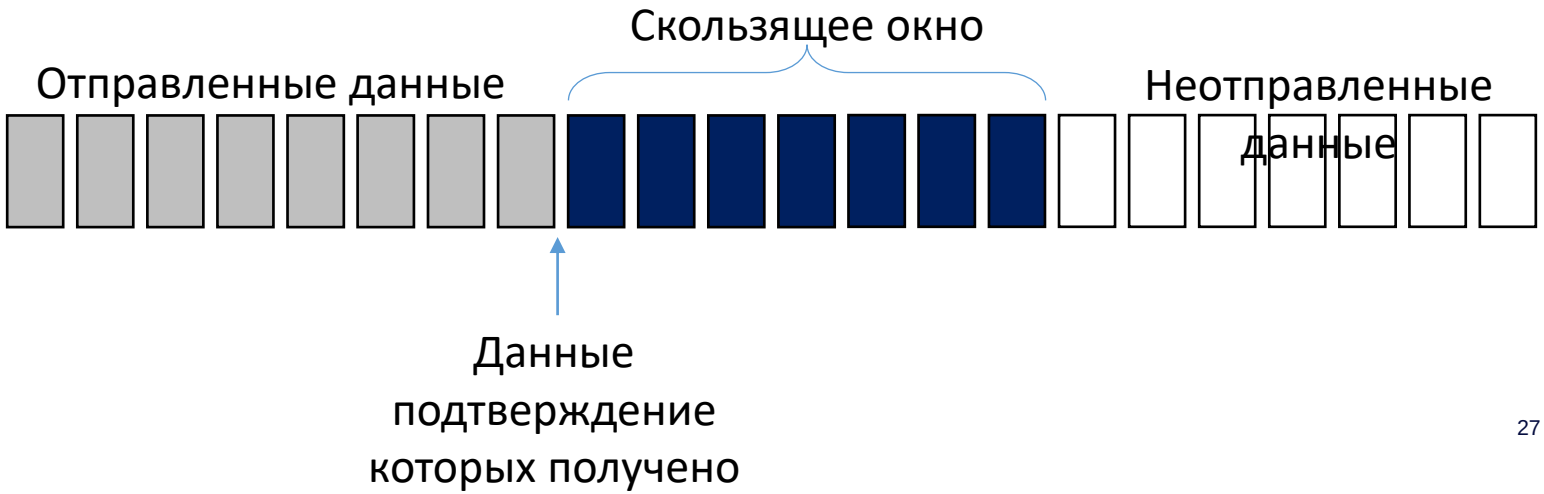


Изменение размера скользящего окна – основной метод регулирования скорости в TCP

- Традиционный подход – фиксированный размер окна 8 сегментов TCP
- Современный подход – динамический размер окна в зависимости от требований приложения и загрузки сети

Подтверждение о доставке

- Разные варианты подтверждений:
  - **Остановка и ожидание** – передача данных после получения подтверждения каждого сообщения (Wi-Fi, канальный уровень)
  - **Скользящее окно** – передача заданного количества сообщений без ожидания подтверждения (TCP, транспортный уровень)
- Размер окна – количество байтов данных, которые могут быть переданы без получения подтверждения
- Кумулятивное подтверждение – подтверждение приема указанного байта данных и всех предыдущих



Указатель на срочность

- Флаг URG – флаг наличия в сегменте срочных данных
  - Используется совместно с полем «Указатель на срочные данные»
  - Позволяет передавать сигналы от отправителя к получателю (прерывания)

- PSH – флаг выталкивания (PUSH)
  - Просит получателя сразу отправлять данные приложению, без буферизации
  - Указатель на срочные данные – смещение от текущего порядкового номера байта до срочных данных в сегменте

Безопасность скользящего окна

- Предотвращение «затопления» быстрым отправителем медленного получателя
  - Сеть может быть свободна, но приложение не готово получить данные
- Транспортная подсистема работает с приложениями:
  - Приложение не обязано забирать данные, как только они появились
  - Транспортная подсистема не обязана передавать данные приложению или в сеть, как только она их получила

Безопасность скользящего окна

- Атака на размер окна
- Атака на срочность данных

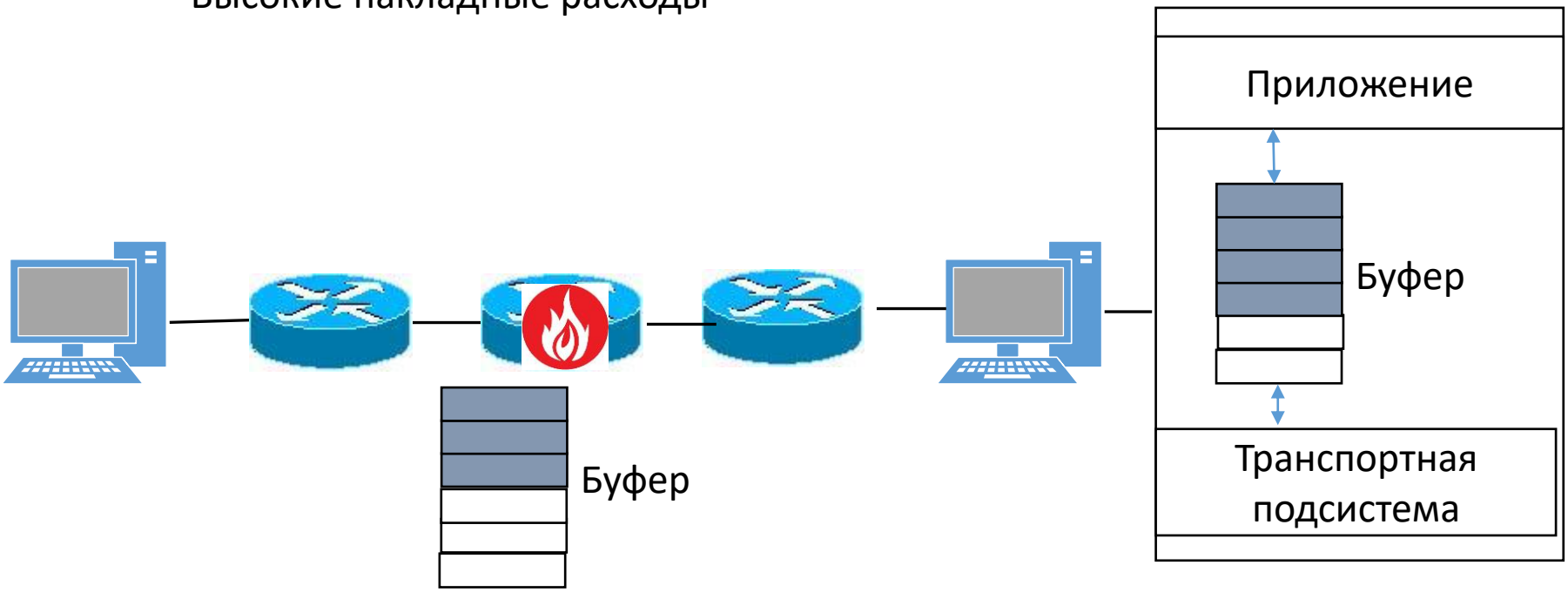
- Приложение может быть готово принять данные, но сеть перегружена
  - Отправляется большая порция данных
  - Многие сегменты будут отброшены сетью

- Перегрузка (congestion) – состояние, при котором в сеть поступает больше пакетов, чем она способна передать

Безопасность скользящего окна

Некоторые приложения читают и пишут данные маленькими порциями

- Эмуляторы терминала telnet или ssh
  - При нажатии каждой клавиши данные передаются на сервер – 1 байт данных
  - Для передачи 1 байта данных требуется передать IP-пакет длиной 41 байт (20 байт заголовок IP, 20 байт заголовок TCP, 1 байт данных)
  - Высокие накладные расходы



Безопасность скользящего окна

- Учет загрузки сети при формировании размера скользящего окна
  - Традиционный подход – фиксированный размер 8 сегментов TCP
  - Предложенный подход – динамический размер окна в зависимости от нагрузки на сеть
- Окно управления потоком:
  - Задается получателем (поле «Размер окна» в заголовке TCP)
  - Размер определяется возможностями приложения читать данные из буфера
- Окно перегрузки:
  - Задается отправителем
  - Размер определяется загрузкой сети
- Размер скользящего окна определяется меньшим из окон перегрузки или управления потоком
- Балансировка нагрузки
  - Сеть должна быть максимально загружена
  - Все хосты в сети получают примерно одинаковую часть от пропускной способности сети

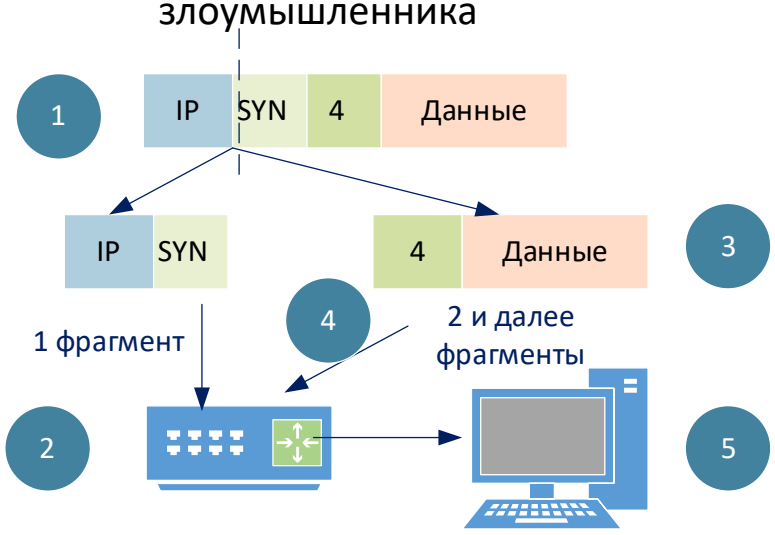
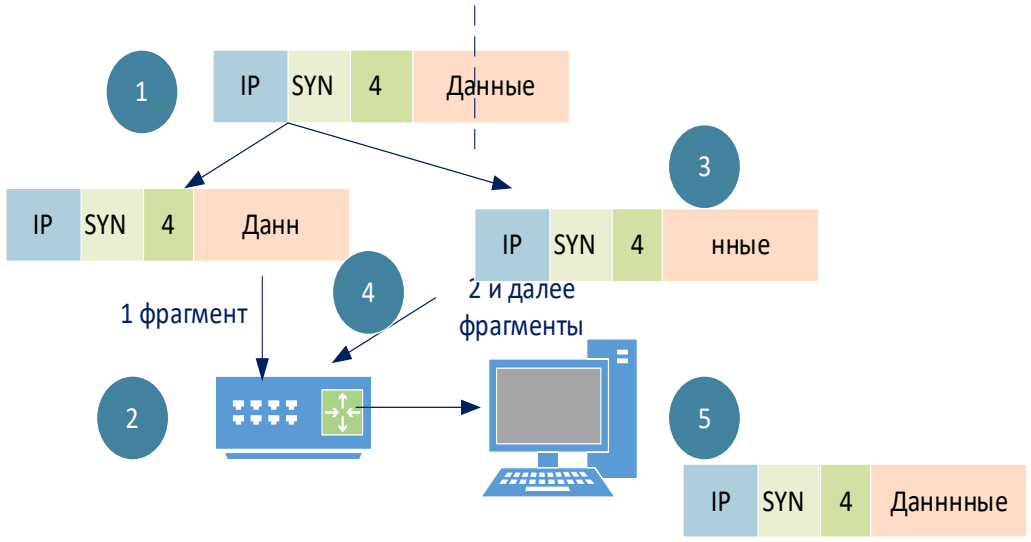
Параметры

Параметры в заголовке TCP являются необязательными, но некоторые используются широко.

Примеры параметров:

- Максимальный размер сегмента (Maximum Segment Size, MSS)
- Масштаб окна - позволяет увеличить размер окна до 1 ГБ, что эффективно для быстрых каналов
- Метки времени
- Выборочное подтверждение (Selective Acknowledgment, SACK) – подтверждение диапазонов принятых байт.

<table>
<tr><td colspan="8">Порядковый номер</td></tr>
<tr><td colspan="8">Номер подтверждения</td></tr>
<tr><td>Длина заголо-вка</td><td></td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Размер окна</td></tr>
<tr><td colspan="6">Контрольная сумма</td><td colspan="3">Указатель на срочные данные</td></tr>
<tr><td colspan="9">Параметры (не обязательно)</td></tr>
</table>

Атака крошечными и накладывающимися фрагментами

1. Злоумышленник X формирует датаграмму TCP-сегмента, состоящую из 2 фрагментов: в первом фрагменте датаграммы равной 8 октетам, находятся номера портов отправителя и получателя и поле SYN, однако значения флагов не попадает.
2. Маршрутизатор, используя правила фильтрации проверяет первый фрагмент датаграммы, на наличие флага SYN, так как он отсутствует, то дейтограмма пропускается в сеть.
3. X формирует 2-ой фрагмент дейтограммы, помещая оставшуюся часть заголовка TCP-сегмента, с установленным флагом SYN
4. Так как первый фрагмент дейтограммы проверен, остальные фрагменты маршрутизатором проверяться не будут, и дайтограмма будет пропущена в сеть.
5. Происходит сборка дейтограммы на узле получателе и соединение с компьютером злоумышленника