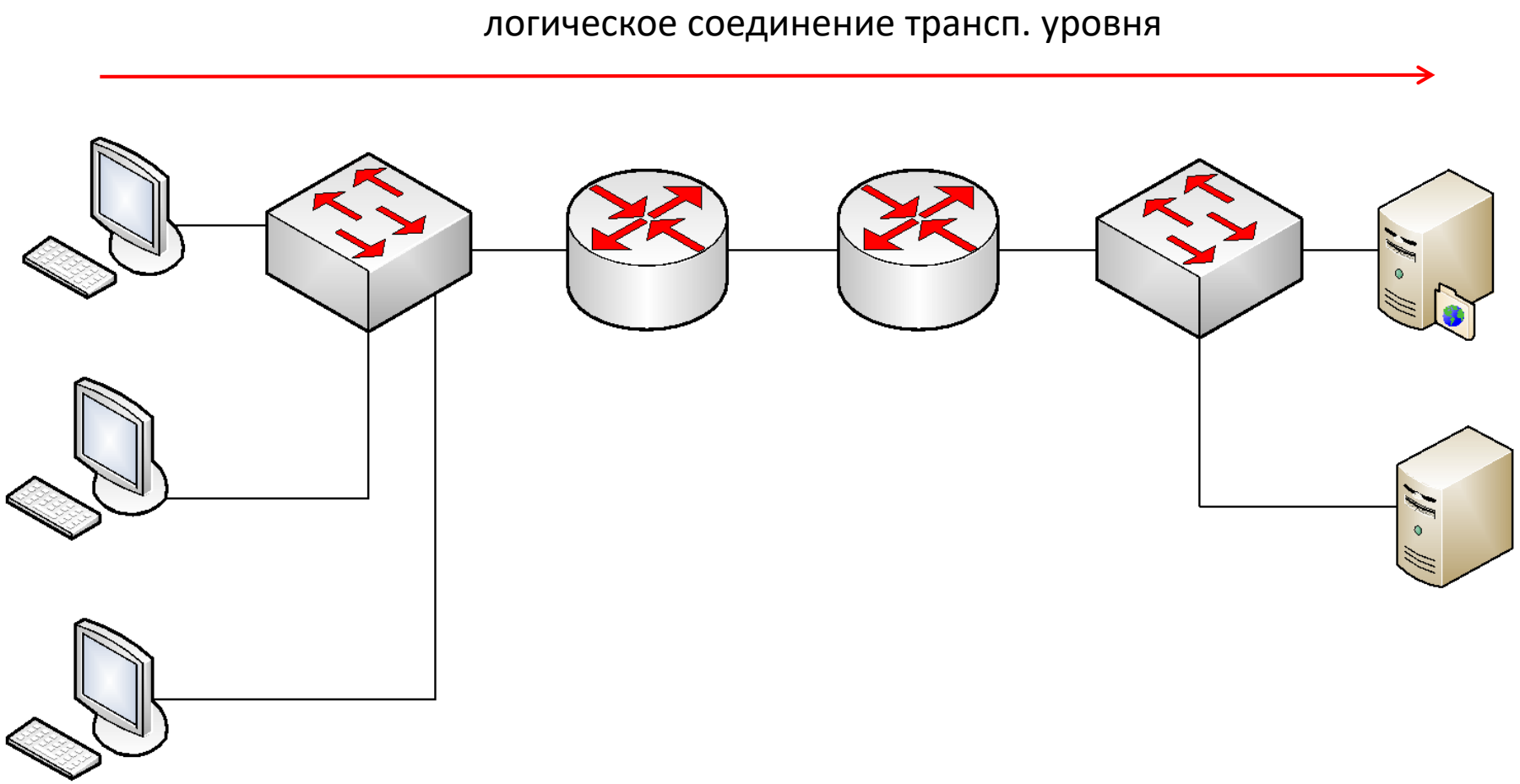
Лекция VII. Протоколы транспортного уровня

Рогозин Н.О., кафедра ИУ-7

Введение

- Транспортный уровень – **центральная часть** сетевой архитектуры
- Связывает службы доставки сетевого уровня, работающие между двумя конечными системами — с одной стороны, и службы доставки, действующими между двумя процессами прикладного уровня, запущенными на конечных системах — с другой.
- Определяет адресацию физических устройств (систем, их частей) в сети.
- Гарантирует доставку блоков информации адресатам и управляет этой доставкой.
- Когда в процессе обработки находится более одного пакета, транспортный уровень контролирует очередность прохождения пакетов. Если проходит дубликат принятого ранее сообщения, то данный уровень опознает это и игнорирует сообщение.

Введение



Сегмент

- Пакет транспортного уровня, используемый для передачи информации
- Получается разбиением сообщения прикладного уровня на фрагменты с последующим добавлением заголовка (TCP/UDP)
- Далее инкапсулируется в пакет сетевого уровня (дейтаграмму) и отсылается
- На принимающей стороне сетевой уровень извлекает сегмент и передает транспортному
- Транспортный уровень выполняет обработку

Сегмент транспортного уровня

Номер порта отправителя	Номер порта получателя
Другие поля заголовка	
Прикладные данные (сообщение/payload)	

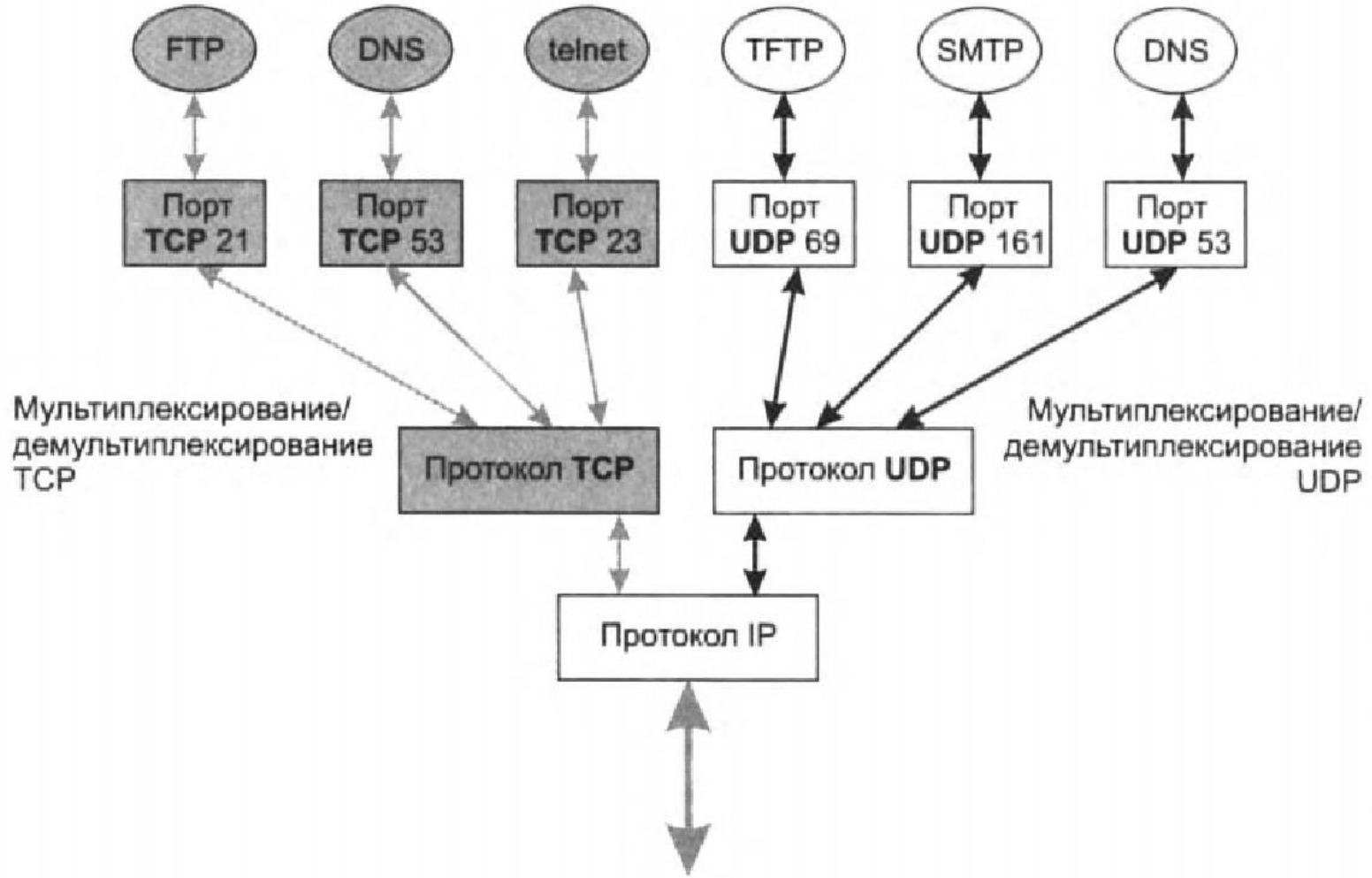
Мультиплексирование/ демультиплексирование

- Работа по доставке данных сегмента транспортного уровня нужному, соответствующему сокету называется **демультиплексированием**.
- Сбор фрагментов данных, поступающих на транспортный уровень хоста-отправителя из различных сокетов, создание сегментов путем присоединения заголовка (который используется при демультиплексировании) к каждому фрагменту и передача сегментов сетевому уровню называется **мультиплексированием**

Мультиплексирование/ демультиплексирование

- 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 и называются портами, причем входная и выходная очереди одного приложения рассматриваются как один порт.
- Для идентификации портов им присваивают номера.

Мультиплексирование/ демультиплексирование



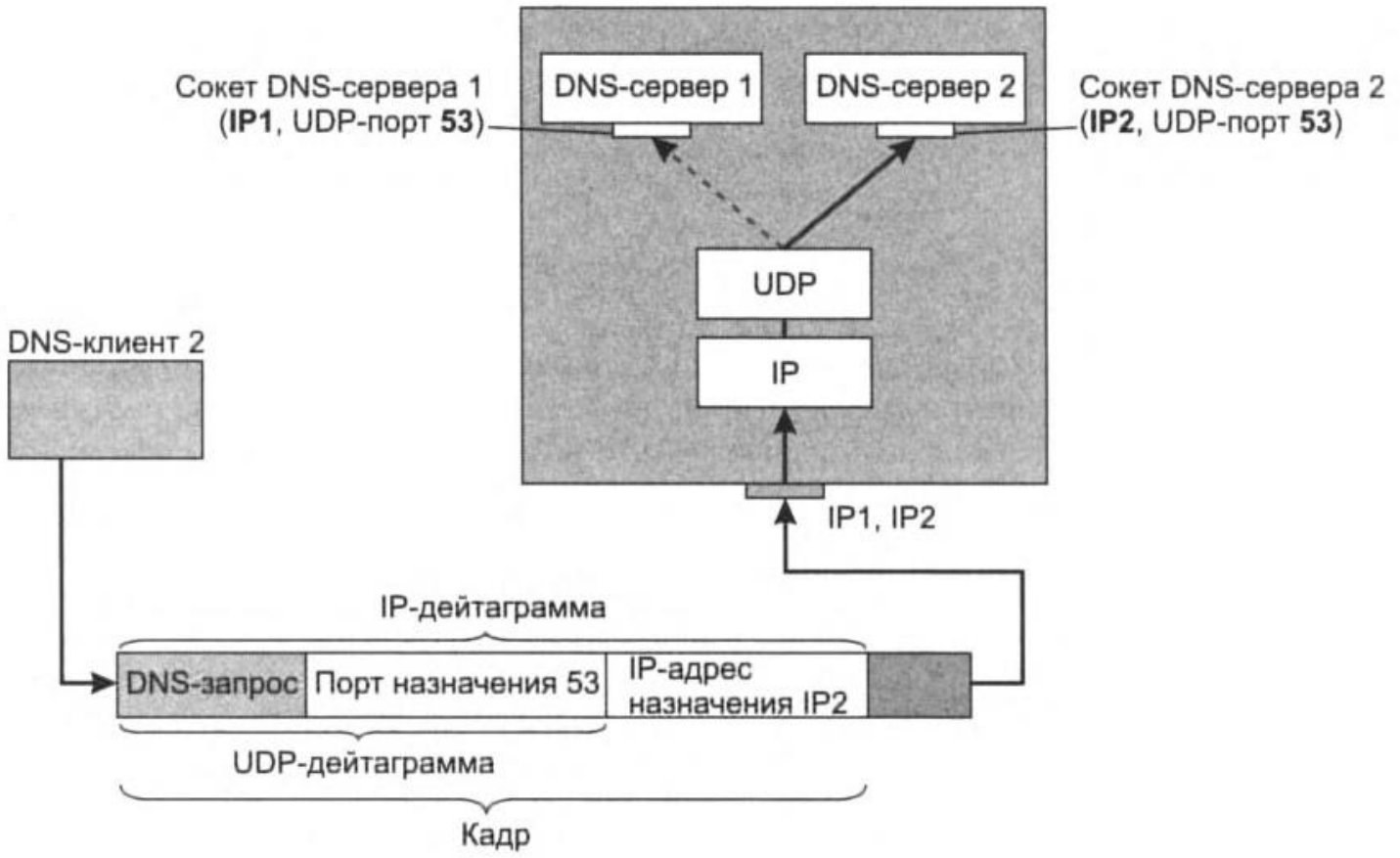
Номера портов

- **Хорошо известные** - для популярных системных служб, таких как FTP, telnet, HTTP, TFTP, DNS (1 - 1024)
- **Зарегистрированные** - предоставляются IANA (1025 - 49151)
- **Динамические** - назначаются локально разработчиками этих приложений или ОС в ответ на поступление запроса от приложения.
- На каждом компьютере ОС ведет список занятых и свободных номеров портов. При поступлении запроса от приложения, выполняемого на данном компьютере, ОС выделяет ему первый свободный номер.

UDP и TCP порты

- Нет никакой зависимости между назначением номеров портов для приложений, использующих протокол TCP, и приложений, работающих с протоколом UDP. Приложения, передающие данные на уровень IP по протоколу UDP, получают номера, называемые UDP-портами.
- Аналогично приложениям, обращающимся к протоколу TCP, выделяются TCP-порты.
- В том и 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.

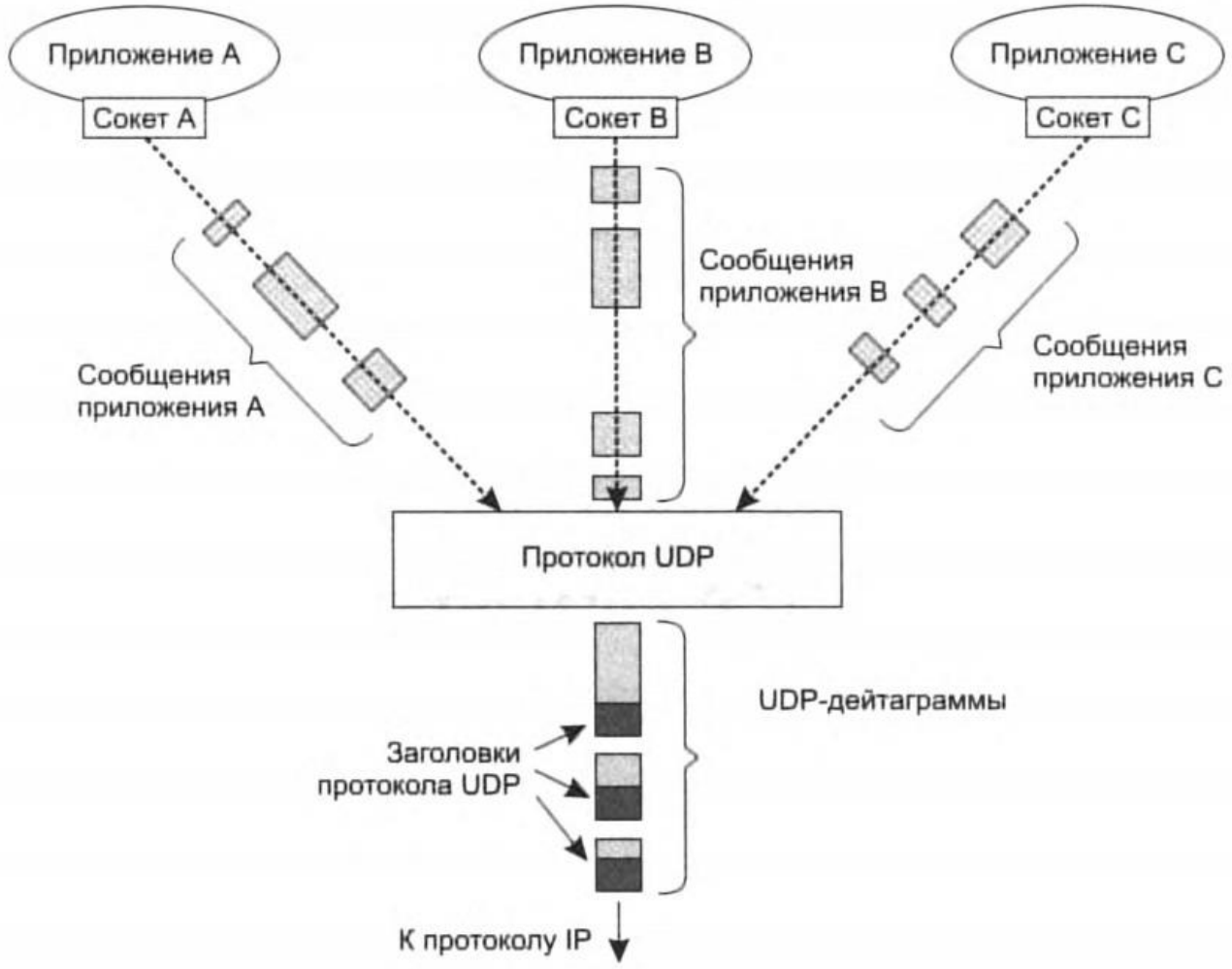
Демультиплексирование UDP по порту



Ассоциации IP-порт

- Чтобы снять неоднозначность в идентификации приложений, разные копии связываются с разными IP-адресами.
- Для этого сетевой интерфейс компьютера, на котором выполняется несколько копий приложения, должен иметь соответствующее число IP-адресов — на рисунке это IP1 и IP2. Во всех IP-пакетах, направляемых DNS-серверу 1, в качестве IP-адреса указывается IP1, а DNS-серверу 2 — адрес IP2.
- Поэтому показанный на рисунке пакет, в поле данных которого содержится UDP-дейтаграмма с указанным номером порта 53, а в поле заголовка задан адрес IP2, однозначно будет направлен заданному адресату — DNS-серверу 2.

UDP(RFC 0768)


Приложение A
Сокет A
Приложение B
Сокет B
Приложение C
Сокет C
Сообщения приложения B
Сообщения приложения A
Сообщения приложения C
Протокол UDP
UDP-дейтаграммы
Заголовки протокола UDP
К протоколу IP

UDP

- Поля портов состоят из 16 -битных целых чисел, представляющих номера портов приложений.
- Поле «порт источника» содержит номер порта, которым пользуется приложение - источник данных.
- Поле «порт - получатель» соответственно указывает на номер порта приложения - получателя данных.
- Поле «длина сообщения» определяет длину (в байтах) UDP - дейтаграммы, включая UDP - заголовок.
- Поле «контрольная сумма», в отличие от контрольной суммы IP - заголовка, содержит результат суммирования всей UDP -дейтаграммы, включая ее данные, область которых начинается сразу после заголовка.
- Модуль UDP отслеживает появление вновь прибывших дейтаграмм, сортирует их и распределяет в соответствии с портами назначения.

Расчет контрольной суммы

- Если длина UDP-сообщения в байтах нечётна, то UDP-сообщение дополняется в конце нулевым байтом (псевдозаголовок и добавочный нулевой байт не отправляются вместе с сообщением, они используются только при расчёте контрольной суммы).
- Поле контрольной суммы в UDP-заголовке во время расчёта контрольной суммы принимается нулевым.
- Псевдозаголовок и UDP-сообщение разбивается на двухбайтные слова. Затем рассчитывается сумма всех слов в арифметике обратного кода (т. е. кода, в котором отрицательное число получается из положительного инверсией всех разрядов числа и существует два нуля: 0x0000 (обозначается +0) и 0xffff(обозначается -0)).
- Результат записывается в соответствующее поле в UDP-заголовке.

Расчет контрольной суммы

- Значение контрольной суммы, равное 0x0000 (+0 в обратном коде), зарезервировано и означает, что для посылки контрольная сумма не вычислялась. В случае, если контрольная сумма вычислялась и получилась равной 0x0000, то в поле контрольной суммы заносят значение 0xffff(-0 в обратном коде).
- При получении сообщения получатель считает контрольную сумму заново (уже учитывая поле контрольной суммы), и, если в результате получится -0 (то есть 0xffff), то контрольная сумма считается сошедшейся. Если сумма не сходится (данные были повреждены при передаче, либо контрольная сумма неверно посчитана на передающей стороне), то решение о дальнейших действиях принимает принимающая сторона.
- Как правило, в большинстве современных устройств, работающих с UDP/IP-пакетами имеются настройки, позволяющие либо игнорировать такие пакеты, либо пропускать их на дальнейшую обработку, невзирая на неправильность контрольной суммы.

Особенности UDP

- ***Ненадёжный*** — когда сообщение посылается, неизвестно, достигнет ли оно своего назначения — оно может потеряться по пути. Нет таких понятий, как подтверждение, повторная передача, тайм-аут.
- ***Неупорядоченность*** — если два сообщения отправлены одному получателю, то порядок их достижения цели не может быть предугадан.
- ***Легковесность*** — никакого упорядочивания сообщений, никакого отслеживания соединений и т. д. Это небольшой транспортный уровень, разработанный на IP.
- ***Датаграммы*** — пакеты посылаются по отдельности и проверяются на целостность только если они прибыли. Пакеты имеют определенные границы, которые соблюдаются после получения, то есть операция чтения на сокете-получателе выдаст сообщение таким, каким оно было изначально послано.
- ***Нет контроля перегрузок*** — UDP сам по себе не избегает перегрузок. Для приложений с большой пропускной способностью возможно вызвать коллапс перегрузок, если только они не реализуют меры контроля на прикладном уровне.

Особенности UDP

- Из-за недостатка надёжности приложения UDP должны быть готовы к некоторым потерям, ошибкам и дублированиям. Некоторые из них (например, TFTP) могут при необходимости добавить элементарные механизмы обеспечения надёжности на прикладном уровне.
- В отличие от TCP, основанные на UDP приложения не обязательно имеют хорошие механизмы контроля и избегания перегрузок.
- Чувствительные к перегрузкам UDP-приложения, которые потребляют значительную часть доступной пропускной способности, могут поставить под угрозу стабильность в Интернете.

Приложения UDP

- Многочисленные ключевые Интернет-приложения используют UDP, в их числе — **DNS** (где запросы должны быть быстрыми и состоять только из одного запроса, за которым следует один пакет ответа), **Простой Протокол Управления Сетями (SNMP)**, **Протокол Маршрутной Информации (RIP)**, **Протокол Динамической Конфигурации Узла (DHCP)**.
- **Голосовой и видеотрафик.** Протоколы потокового видео в реальном времени и аудио разработаны для обработки случайных потерь пакетов так, что качество лишь незначительно уменьшается вместо больших задержек при повторной передаче потерянных пакетов. Поскольку и TCP, и UDP работают с одной и той же сетью, многие компании замечают, что недавнее увеличение UDP-трафика из-за этих приложений реального времени мешает производительности TCP-приложений вроде систем БД.

TCP

- Служит для передачи данных между сетевыми и прикладными уровнями сетевой модели.
- Для обеспечения надежной доставки и правильной последовательности данных в общем потоке, протокол TCP использует подтверждения. Каждый раз при передаче сообщения модуль TCP запускает таймер. По истечении заданного в нем времени и не получения подтверждения, протокол TCP повторяет попытку передать свое сообщение.

Логическое соединение

- 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.
- **Логическое соединение** дает возможность участникам обмена следить за тем, чтобы данные не были потеряны, искажены или продублированы, а также чтобы они пришли к получателю в том порядке, в котором были отправлены.
- Протокол TCP устанавливает логические соединения между прикладными процессами, причем в каждом соединении участвуют только два процесса. TCP-соединение является дуплексным, то есть каждый из участников этого соединения может одновременно получать и отправлять данные.

Параметры логического соединения

Каждая сторона сообщает:

- максимальный размер сегмента, который она готова принимать;
- максимальный объем данных (возможно несколько сегментов), 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 рамках данного соединения.

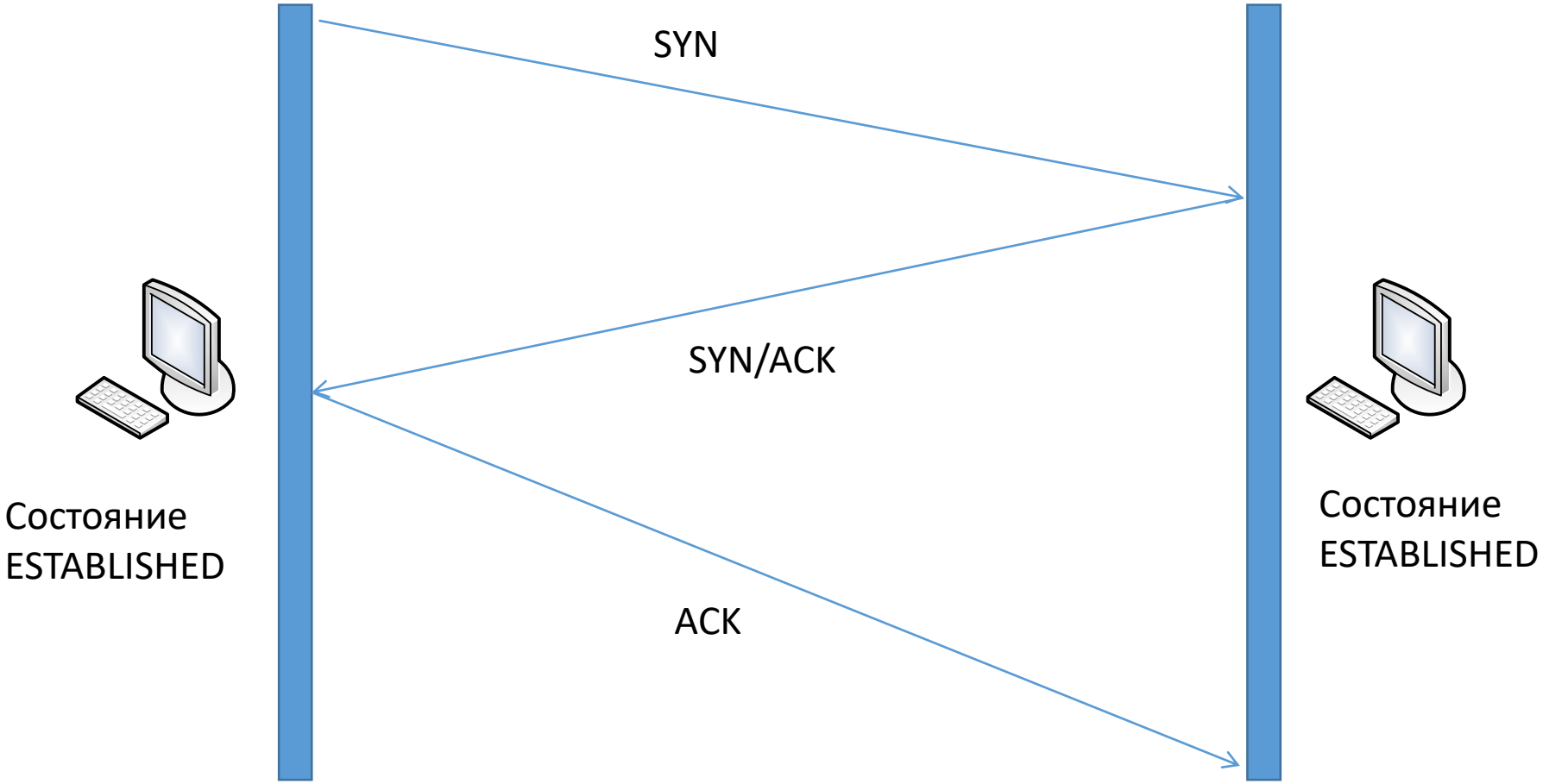
Особенности TCP

- ***Надёжность*** — TCP управляет подтверждением, повторной передачей и тайм-аутом сообщений. Производятся многочисленные попытки доставить сообщение. Если оно потеряется на пути, сервер вновь запросит потерянную часть. В TCP нет ни пропавших данных, ни (в случае многочисленных тайм-аутов) разорванных соединений.
- ***Упорядоченность*** — если два сообщения последовательно отправлены, первое сообщение достигнет приложения-получателя первым. Если участки данных прибывают в неверном порядке, TCP отправляет неупорядоченные данные в буфер до тех пор, пока все данные не могут быть упорядочены и переданы приложению.
- ***Тяжеловесность*** — TCP необходимо три пакета для установки сокет-соединения перед тем, как отправить данные. TCP следит за надёжностью и перегрузками.
- ***Потоковость*** — данные читаются как поток байтов, не передается никаких особых обозначений для границ сообщения или сегментов.

Установление сеанса

- Процесс установления сеанса TCP между клиентом и сервером (называемый также трехэтапным квитированием) происходит следующим образом. Клиент инициирует сеанс, передавая сегмент с установленным битом синхронизации (SYNchronization — SYN).
- Этот сегмент содержит данные о размере окна клиента и его текущем порядковом номере. Сервер отвечает на запрос SYN клиента сегментом АСК и также включает в передаваемый им сегмент бит SYN, данные о размере окна и начальном порядковом номере.
- Наконец, клиент отвечает на сегмент SYN сервера подтверждением АСК.

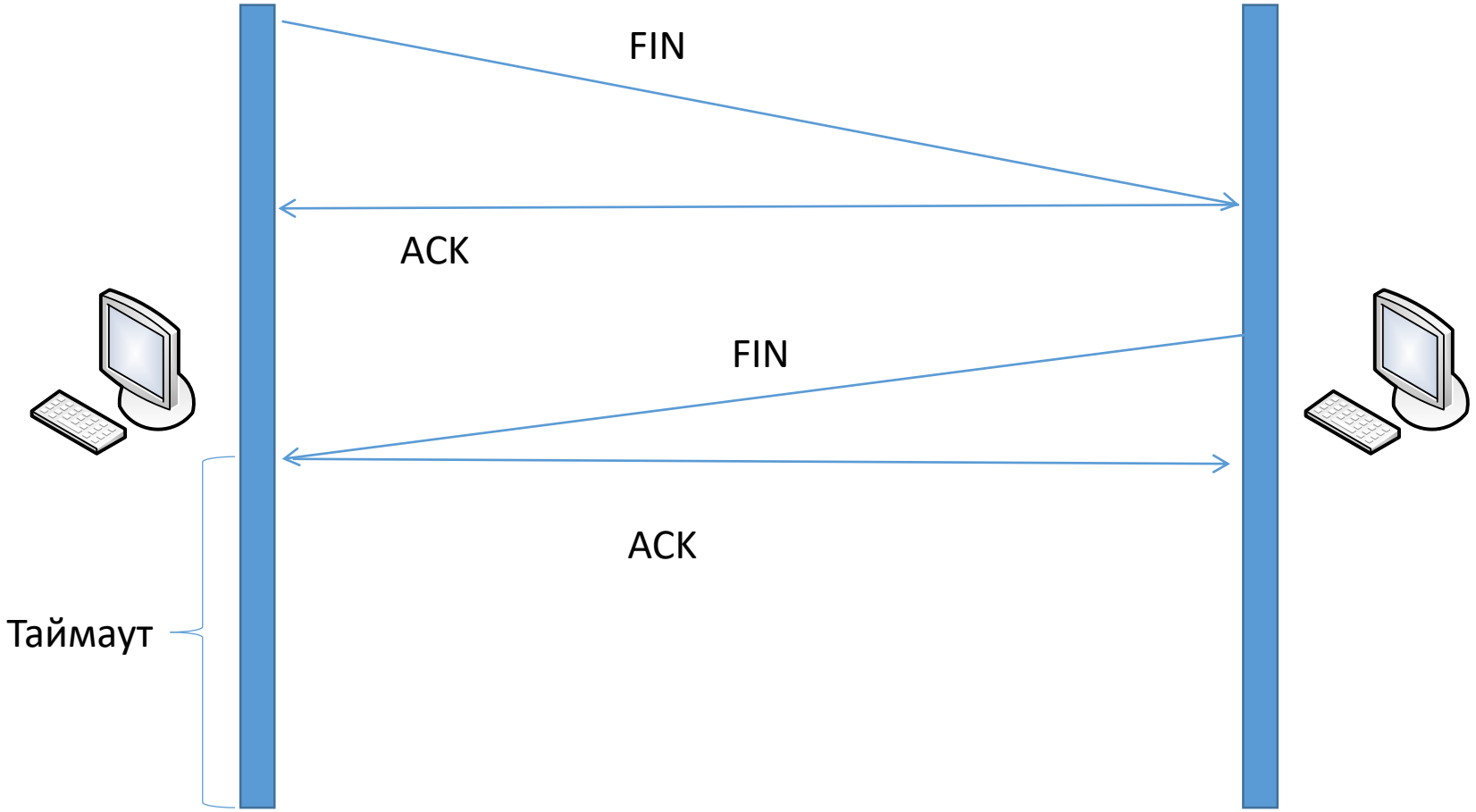
Установление сеанса



Разрыв сеанса

- аналогичен процессу его установления. Вначале участник соединения, желающий закрыть сеанс (предположим, что в данном примере это — клиент), инициализирует процесс завершения сеанса, отправляя сегмент с установленным битом завершения (FINish — FIN).
- Сервер отвечает, передавая в ответ на сегмент FIN клиента подтверждение ACK. Затем сервер передает собственный сегмент с установленным битом FIN. После этого клиент передает в ответ на сегмент FIN сервера подтверждение ACK и сеанс закрывается.
- Каждый FIN/ACK закрывает одно направление передачи

Разрыв сеанса



ARQ

- Должен ли отправитель ждать, пока не придет квитанция на отправленный пакет, прежде чем отсылать следующий?
- Должна ли принимающая сторона подтверждать приход каждого или сразу нескольких пакетов?
- Какое время ожидания квитанции источником является предельно допустимым?
- Что, если квитанция потеряется и отправитель еще раз пошлет тот же пакет? Каким образом приемник должен распознавать дубликаты пакетов, а источник — квитанций?

Методы решений

- методы простоя источника (Stop-and-Wait);
- методы скользящего окна, которые свою очередь тоже подразделяются на два класса:
  - методы, использующие **окно передачи** — к ним, в частности, относится **метод передачи с возвращением на N пакетов (Go-Back-N)**;
  - методы, использующие окно передачи и окно приема — примером является **метод передачи с выборочным повторением (Selective Repeat)**.

Общие условия для всех методов

- Отправитель и получатель, 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 используется таймер, тайм-аут которого устанавливается равным предельному времени ожидания квитанции.

Окно перегрузки

- Принцип скользящего окна позволяет послать несколько сообщений и только потом ожидать подтверждения. Модуль TCP регулирует полосу пропускания сети, договариваясь с удаленным узлом о параметрах потока данных.
- При этом процесс регулировки происходит на протяжении всего соединения TCP. Регулировка заключается в изменении размеров скользящего окна.
- При низкой загрузке сети, размер скользящего окна увеличивается, скорость выдачи данных на канале связи увеличивается. Если загрузка сети достаточна велика, модуль TCP уменьшает размер скользящего окна.

Окно перегрузки

- Метод передачи с применением окон, предусмотренный в протоколе TCP, не только представляет собой динамический процесс, но и осуществляется в дуплексном режиме. Это означает, что на каждом хосте предусмотрена возможность и передавать, и принимать данные о размере и положении окна.
- Каждое окно применяется независимо от другого и может увеличиваться или уменьшаться с учетом условий передачи в соответствующем направлении.

Окно управления

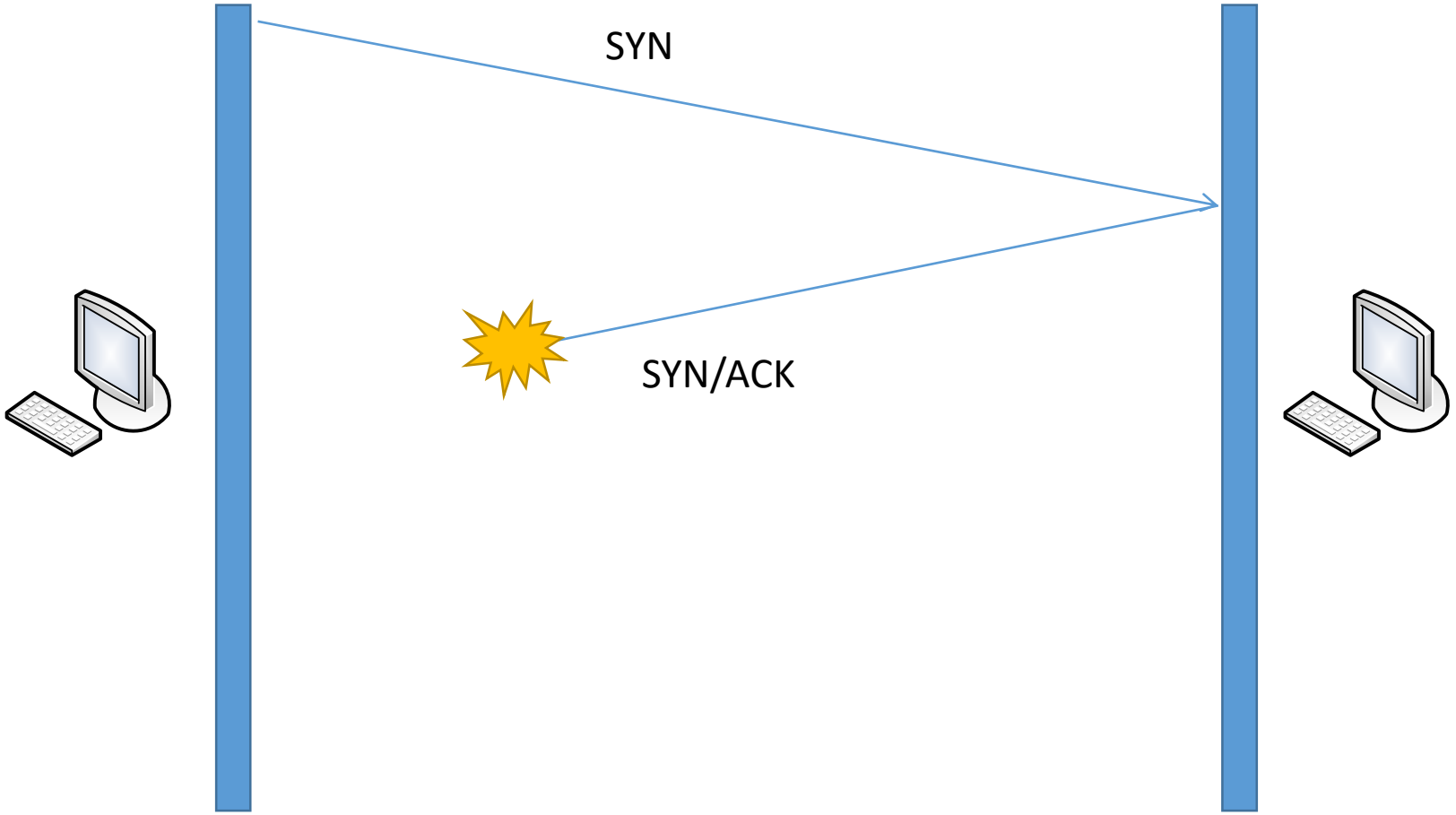
- Для идентификации пакетам присваиваются уникальные последовательные номера, размещаемые в заголовках пакетов.
- Разрядность поля «номер пакета» определяет диапазон возможных номеров. Когда этот диапазон исчерпывается, нумерация пакетов снова начинается с нуля.
- Таким образом, нельзя абсолютно исключить ситуацию, когда в сети существуют пакеты с одинаковыми номерами. Окно определяется на последовательности пронумерованных пакетов.
- 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.

Скользящее окно

Скользящее окно

- В идеальном случае пакеты и квитанции не теряются и приходят в том же порядке, в котором были отправлены.
- При этом интервалы между пакетами и квитанциями являются неравномерными.
- Движение окна определяется, во-первых, поступлением квитанций —подтверждение успешного приема очередного пакета позволяет переместить окно вперед, во-вторых, исчерпанием окна — окно приостанавливается, когда отправитель передал все пакеты из окна, но не получил ни на один из них квитанции.

Дублирование сообщения из-за потери

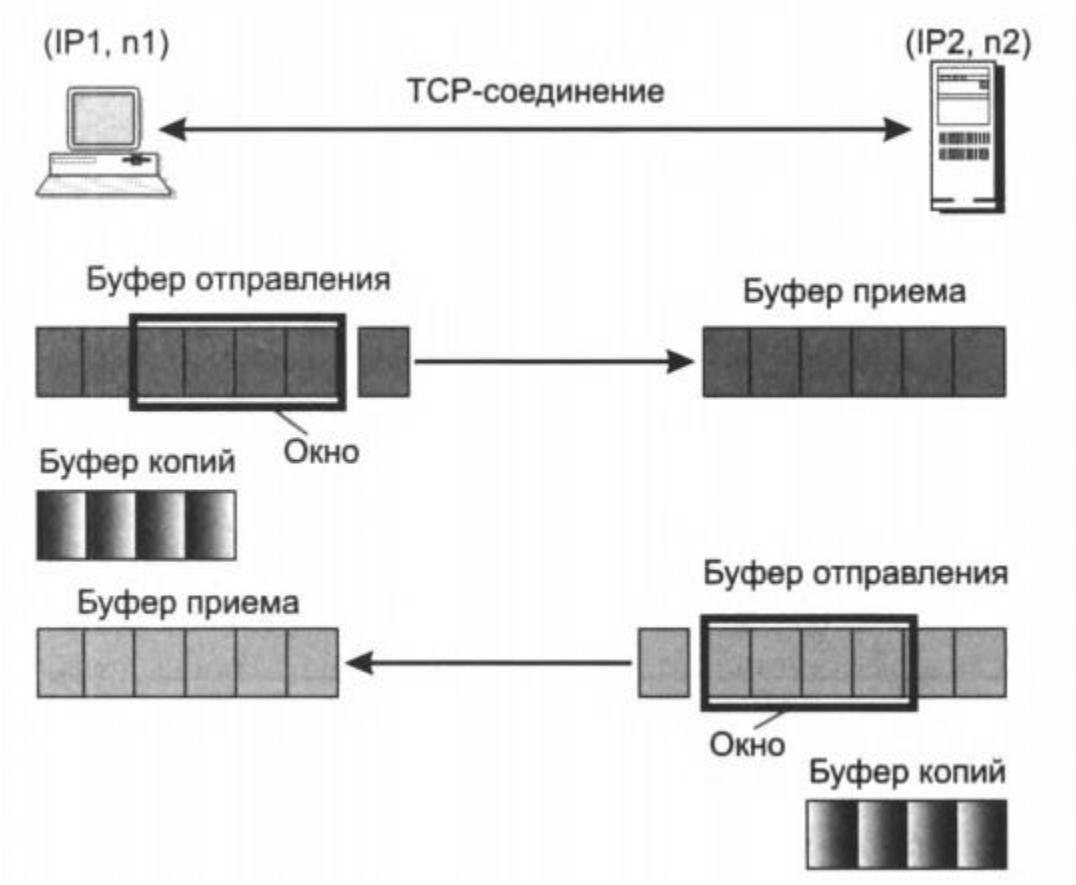


Сегменты и поток байтов

- Одним из параметров, регулируемых в процессе обмена данными 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 осуществляется, начиная от заголовка
- Когда отправитель посылает TCP-сегмент, он помещает в поле последовательного номера номер первого байта данного сегмента, который служит идентификатором сегмента.

Сегменты и поток байтов

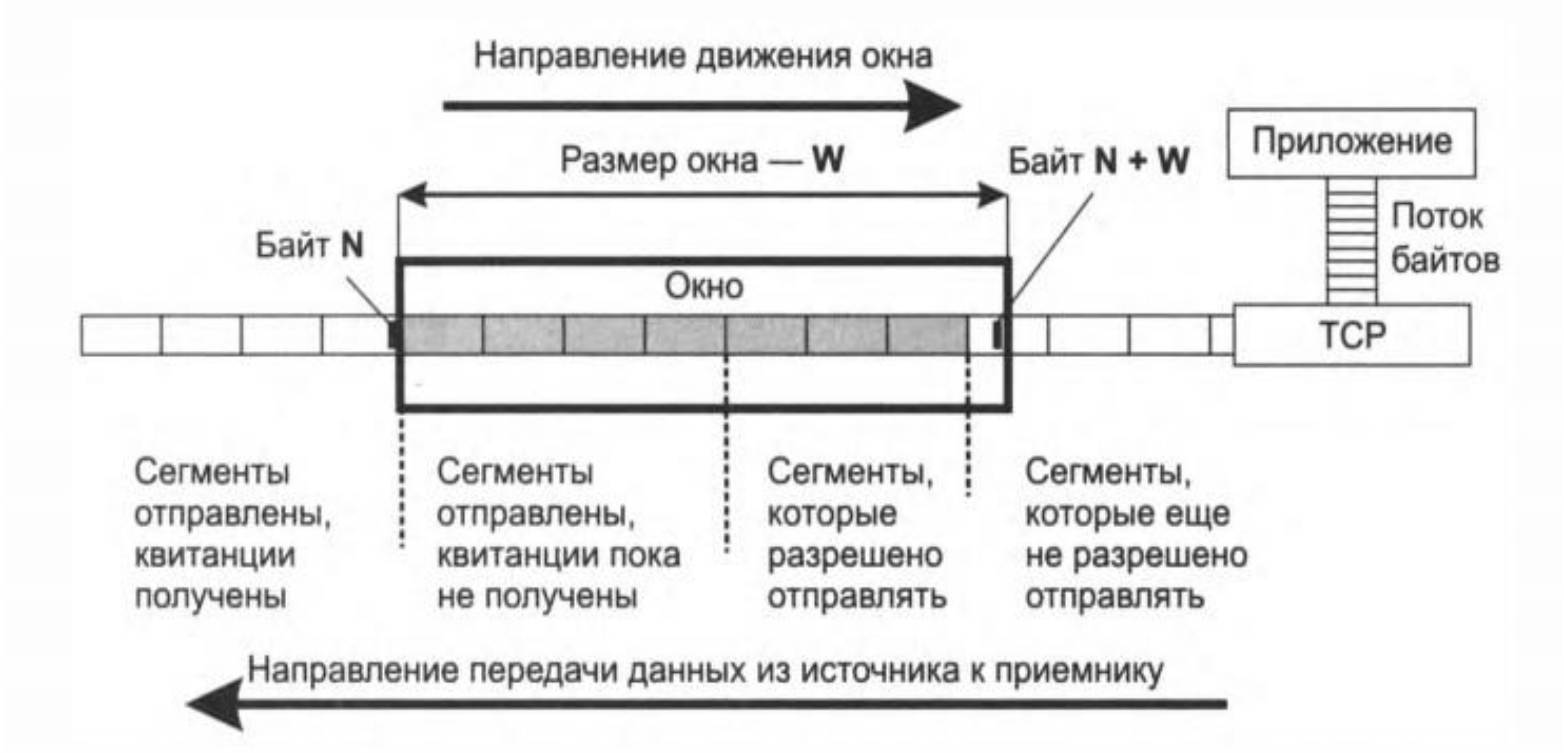
Система окон в TCP



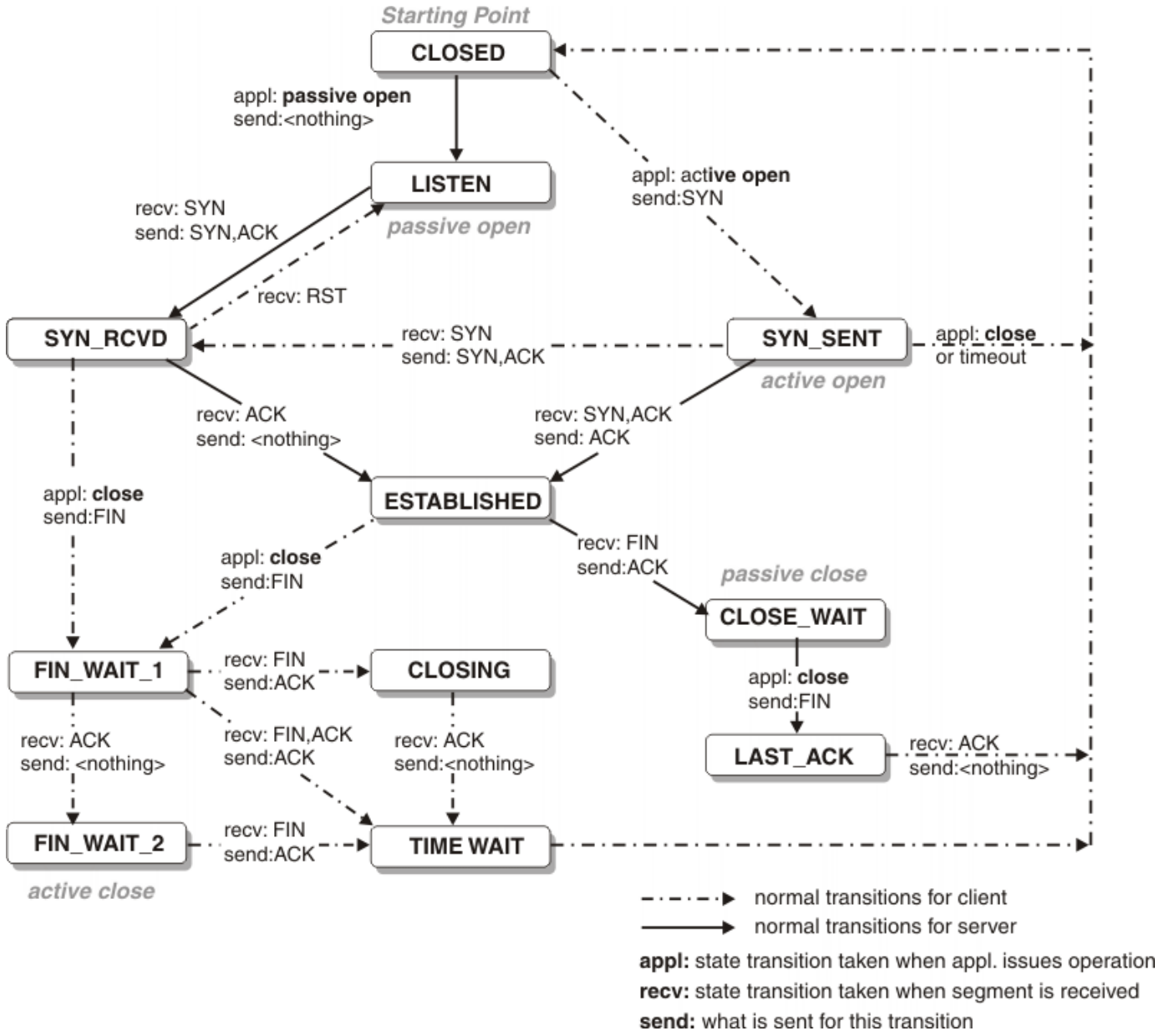
Система окон в TCP

- 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. Часть байтов, входящих в окно, составляют сегменты, которые также уже отправлены, но квитанции на которые пока не получены.
- 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.

Система окон в TCP



TCP FSM


Starting Point
CLOSED
appl: passive open
send:<nothing>
LISTEN
passive open
recv: SYN
send: SYN,ACK
recv: RST
appl: active open
send:SYN
SYN_RCVD
recv: SYN
send: SYN,ACK
SYN_SENT
appl: close
or timeout
active open
recv: ACK
send: <nothing>
recv: SYN,ACK
send: ACK
appl: close
send:FIN
ESTABLISHED
appl: close
send:FIN
recv: FIN
send:ACK
passive close
CLOSE_WAIT
FIN_WAIT_1
recv: FIN
send:ACK
CLOSING
appl: close
send:FIN
recv: ACK
send: <nothing>
recv: FIN,ACK
send:ACK
recv: ACK
send:<nothing>
LAST_ACK
recv: ACK
send:<nothing>
FIN_WAIT_2
recv: FIN
send:ACK
TIME WAIT
active close
normal transitions for client
normal transitions for server
appl: state transition taken when appl. issues operation
recv: state transition taken when segment is received
send: what is sent for this transition

TCP, формат сегмента

- Состоит из поля данных и нескольких полей заголовка.
- *Поле данных* содержит фрагмент данных, передаваемых между процессами.
- Размер поля данных ограничивается величиной **MSS** (максимальный размер сегмента). Большой файл, как правило, разбивается на фрагменты размером MSS (кроме последнего фрагмента, который обычно имеет меньший размер).
- Интерактивные приложения, напротив, часто обмениваются данными, объем которых значительно меньше MSS. Например, приложения удаленного доступа к сети, подобные Telnet, могут передать транспортному уровню 1 байт данных.
- Поскольку обычно длина заголовка TCP-сегмента составляет 20 байт (что на 12 байт больше, чем в UDP), полный размер сегмента в этом случае равен 21 байт.

TCP, формат сегмента

Порт источника (sours port)								**Порт приемника** (destination port)
Последовательный номер (sequence number) — номер первого байта данных в сегменте, определяет смещение сегмента относительно потока отправляемых данных								
Подтвержденный номер (acknowledgement number) — максимальный номер байта в полученном сегменте, увеличенный на единицу								
Длина заголовка (hlen)	**Резерв** (reserved)	URG	ACK	PSH	RST	SYN	FIN	**Окно** (window) — количество байтов данных, ожидаемых отправителем данного сегмента, начиная с байта, номер которого указан в поле подтвержденного номера
Контрольная сумма (checksum)								**Указатель срочности** (urgent pointer) — указывает на конец данных, которые необходимо срочно принять, несмотря на переполнение буфера
Параметры (options) — это поле имеет переменную длину и может вообще отсутствовать, используется для решения вспомогательных задач, например для согласования максимального размера сегмента								
Заполнитель (paddind) — это фиктивное поле может иметь переменную длину, используется для доведения размера заголовков до целого числа 32-битовых слов								

Сегмент TCP, структура

- **Порт источника и порт получателя.** 16 - битные поля источника и получателя однозначно определяют посылающие и принимающие данные приложения или прикладные протоколы (программы).
- Номера портов источника и получателя в совокупности с IP - адресами сетевых компьютеров (в IP - заголовке) однозначно идентифицируют любое TCP - соединение и представляют сокет.

Сегмент TCP, структура

- **Номер последовательности.** 32 - битное поле номера последовательности обозначает первый байт данных из области данных сегмента TCP.
- Оно соответствует смещению этого байта относительно начала потока данных.
- Каждый байт в потоке данных может быть идентифицирован при помощи номера последовательности.

Сегмент TCP, структура

- Флаги. *Заголовок TCP содержит шесть однобитных полей флагов.*
- **Флаг UR** сообщает принимающему модулю TCP о наличии данных, требующих немедленной обработки
- **Флаг ACK** подтверждает правильность номера подтверждения в заголовке TCP
- **Флаг PSH** требует от принимающего модуля немедленно передать принятый сегмент данных приложению -получателю
- **Флаг RST** запрашивает у принимающего модуля TCP сброс соединения (прекращения работы приложений)
- **Флаг SYN** указывает принимающему модулю TCP необходимость синхронизации последовательности
- **Флаг FIN** сообщает принимающему модулю TCP о том, что источник закончил передачу данных.

Сегмент TCP, структура

- **Размер окна.** 16 - битное поле «размер окна» сообщает принимающему модулю TCP количество байтов, которое собирается принять передатчик.
- Значение данного поля определяет размер скользящего окна. Как правило, оно равняется нескольким тысячам байтов.
- **Контрольная сумма TCP.** 16 - битное поле контрольной суммы TCP содержит сумму, вычисленную по всему сегменту TCP, включая данные. Протокол TCP требует от передатчика, чтобы он включил вычисленную контрольную сумму в поле, а от приемника - чтобы он вычислил ее повторно и сравнил с принятой.

Мультиплексирование с установлением соединения

- Когда TCP- сегмент поступает из сети на хост, тот использует IP-адрес отправителя, номер порта отправителя, IP-адрес получателя и номер порта получателя, чтобы направить (демультиплексировать) сегмент в соответствующий сокет.
- Два полученных TCP сегмента будут иметь различные IP-адреса отправителя или номера портов отправителя, (исключая случай, когда TCP-сегмент содержит первичный запрос на установление соединения) и окажутся направлены в два разных сокета

Пример работы TCP приложения

- На серверном приложении запущен “впускающий сокет”, ожидающий запрос на соединение
- TCP-клиент создает сокет и отправляет сегмент с запросом на создание соединения, для этого используются функции socket и connect
- Запрос на установление логического соединения – TCP-сегмент с портом назначения и комбинацией битов в заголовке TCP-сегмента, создающего соединение.

Пример работы TCP приложения

- Когда операционная система на хосте с запущенным серверным процессом получает входящий сегмент, который содержит запрос соединения с номером порта получателя 12 000, она направляет сегмент серверному процессу, ожидающему его, чтобы принять соединение на порт с номером 12 000.
- Затем серверный процесс создает новый сокет, с исп. функции accept.
- При обработке сегмента с запросом на установление логического соединения сервер использует четыре параметра: номер порта отправителя сегмента, IP-адрес хоста отправителя, номер порта получателя сегмента и собственный IP-адрес.

Соотношение протоколов

Приложение	Прикладной протокол	Транспортный протокол
Электронная почта	SMTP	TCP
Доступ с удаленного терминала	Telnet	TCP
Web	HTTP	TCP
Передача файлов	FTP	TCP
Удаленный файловый сервер	NFS	UDP или TCP
Потоковое мультимедиа	Нестандартный	UDP или TCP
Интернет-телефония	Нестандартный	Как правило, UDP
Сетевое администрирование	SNMP	Как правило, UDP
Протокол маршрутизации	RIP	Как правило, UDP
Трансляция имен	DNS	Как правило, UDP

SCTP

- первый стандарт - RFC 2960, затем дополнен в RFC 4960
- надежный транспортный протокол, который обеспечивает стабильную, упорядоченную (с сохранением порядка следования пакетов) передачу данных между двумя конечными точками (подобно TCP).
- Кроме того, протокол обеспечивает сохранение границ отдельных сообщений (подобно UDP).
- Однако в отличие от протоколов TCP и UDP протокол SCTP имеет дополнительные преимущества, такие как поддержка множественной адресации (multihoming) и многопоточности (multi-streaming) - каждая из этих возможностей увеличивает доступность узла передачи данных.

Основные функции

- unicast-протокол, который обеспечивает обмен данными между двумя конечными точками.
- каждая из этих точек может быть представлена более чем 1 адресом IP.
- обеспечивает гарантию доставки, детектирование случаев отбрасывания данных, изменения порядка доставки, повреждения или дублирования данных, а также повторную передачу пакетов при возникновении необходимости.
- Обмен данными по протоколу SCTP происходит в полнодуплексном режиме.

Основные функции

- Работа SCTP основана на обмене сообщениями и протокол поддерживает кадрирование границ отдельных сообщений. Протокол TCP ориентирован на обмен байтами и не хранит никаких неявных структур в передаваемых потоках данных.
- Скорость передачи по протоколу SCTP может адаптироваться подобно тому, как это происходит в TCP, – в зависимости от загрузки сети.
- может использоваться одновременно с TCP на общих каналах передачи данных.

Поддержка многопоточности

- Поддержка множества одновременных потоков позволяет распределить между этими потоками передаваемую информацию так, чтобы каждый из потоков обеспечивал независимую упорядоченную доставку данных. При потере сообщения в любом из потоков это оказывает влияние лишь на данный поток, не затрагивая работу других потоков данных.
- Протокол TCP работает с одним потоком данных и обеспечивает для такого потока сохранение порядка доставки байтов из потока. Такой подход удобен для доставки файлов или записей, но он может приводить к дополнительным задержкам при потере информации в сети или нарушении порядка доставки пакетов. При возникновении таких ситуаций протокол TCP должен дожидаться доставки всех данных, требуемых для восстановления порядка.

Поддержка многопоточности

- Для многих приложений строгое сохранение порядка доставки не является обязательным.
- В системах передачи телефонной сигнализации достаточно поддерживать порядок передачи для последовательности сообщений, которые воздействуют на один ресурс (например, для одного вызова или одного канала).
- Остальные сообщения слабо коррелируют между собой и могут доставляться с нарушением порядка.

Поддержка многопоточности

- Другим примером является возможность использования множества потоков для доставки multimedia-документов (например, web- страниц) в рамках одной сессии.
- Поскольку такие документы состоят из разнотипных объектов разных размеров, многопотоковая доставка таких компонент не требует строгой упорядоченности. Однако использование множества потоков при доставке существенно повышает для потребителей привлекательность сервиса.

Поддержка многопоточности

- Многопоточная передача поддерживается засчет устранения зависимости между передачей и доставкой данных. В частности, каждый блок полезной информации типа DATA (данные) использует два набора порядковых номеров. Номер TSN управляет передачей сообщений и детектированием их потери, а пара “идентификатор потока Stream ID – номер SSN ” используется для управления порядком доставки потребителю полученных данных.
- Такая независимость механизмов нумерации позволяет получателю незамедлительно обнаруживать пропуски данных (например, в результате потери сообщения), а также видеть влияние потерянных данных на поток.
- Утрата сообщения вызывает появление пропуска в порядковых номерах SSN для потока, на который это сообщение оказывает влияние и не вызывает такого пропуска для других потоков. Следовательно, получатель может продолжать доставку незатронутых потоков, не дожидаясь повтора передачи утраченного сообщения

Многодомность

- механизм предназначен для того, чтобы увеличить уровень устойчивости сети к выходам из строя интерфейсов на хосте и ускорить восстановление в случае сбоя в сети.
- Однако эффективность этого механизма падает, когда путь взаимодействия внутри ассоциации проходит через единую точку сбоя сети, к примеру, единственный канал или маршрутизатор, через которые должен проходить весь трафик ассоциации, или хост, обладающий всего одним интерфейсом.

Механизм Cookie

- обеспечивает защиту от атак вслепую, когда атакующий генерирует множество блоков INIT с целью перегрузки сервера SCTP за счет расхода памяти и иных ресурсов, требуемых для обработки новых запросов INIT.
- Взамен выделения памяти для TCB (Transport Control Block) сервер создает параметр Cookie с информацией TCB, корректным временем жизни и сигнатурой для аутентификации. Этот параметр передается клиенту в сообщении INIT ACK. Поскольку сообщения INIT ACK всегда возвращаются по адресу отправителя запросов INIT, атакующий вслепую не будет получать Cookie. Легитимный клиент SCTP получит Cookie и вернет серверу блок COOKIE ECHO, по которому сервер SCTP может проверить корректность Cookie и использовать это значение для создания TCB.
- Поскольку параметр Cookie создается сервером и на сервере же проверяется, формат и секретный ключ Cookie знает только сервер и не возникает необходимости в передаче их через сеть клиенту.

INIT Collision Resolution

- Поддержка многодомных хостов может приводить к изменению порядка доставки сообщений, для которых использовались разные пути.
- Эта проблема возникает на этапе создания ассоциации, когда работа без процедуры разрешения конфликтов может с легкостью привести к созданию множества параллельных ассоциаций между парой конечных точек.
- Для предотвращения этого SCTP включает множество процедур связывания параллельных попыток создания ассоциации с одной ассоциацией SCTP.

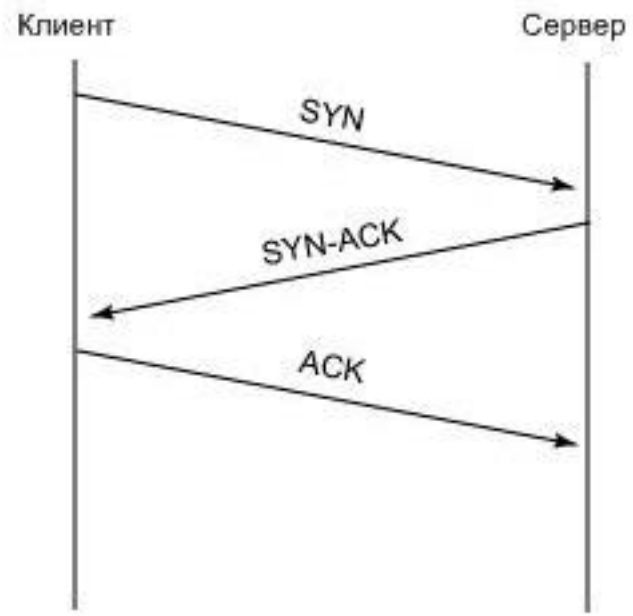
Инициация

- Прежде, чем начнется обмен данными, два SCTP-хоста должны передать друг другу информацию о состоянии соединений (в том числе задействованные IP-адреса) с помощью четырехэтапной процедуры установки соединения
- Процедура, предусмотренная протоколом SCTP, позволяет защититься от DoS-атак. Получателю сообщения о намерении установить контакт INIT в четырехэтапной процедуре установки соединения не требуется сохранять никакую информацию о состоянии или резервировать какие-либо ресурсы.
- Вместо этого он посылает в ответ сообщение INIT-ACK, которое включает в себя специальную запись (cookie) состояния, содержащую всю информацию необходимую отправителю INIT-ACK для того, чтобы сформировать свое состояние.

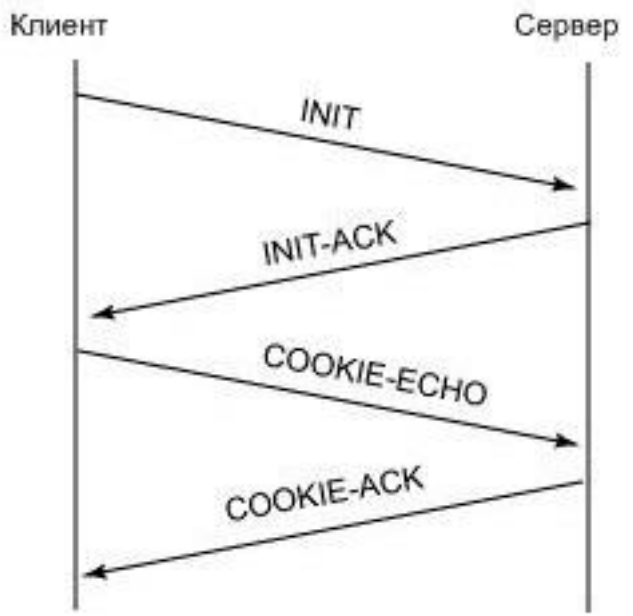
Инициация

- INIT и INIT-ACK содержат несколько параметров:
  - список всех IP-адресов, которые станут частью ассоциации;
  - номер транспортной последовательности, который используется для надежной передачи данных;
  - тег инициации, который должен быть включен в каждый входящий пакет SCTP;
  - число выходящих потоков, запрашиваемых каждой из сторон;
  - число входящих потоков, которые способна поддерживать каждая из сторон.

Инициация



Процедура установления соединения (трехэтапное квитирование) в протоколе TCP



Процедура установления соединения (четырехэтапное квитирование) в протоколе SCTP

Структура пакета

Биты	Биты 0-7	8-15	16-23	24-31
+0	Порт источника		Порт назначения	
32	Тег проверки			
64	Контрольная сумма			
96	Тип 1 блока	Флаги 1 блока	Длина 1 блока	
128	Данные 1 блока			
…	…			
…	Тип N блока	Флаги N блока	Длина N блока	
…	Данные N блока			

Структура пакета

- Любой пакет SCTP в ассоциации, не содержащий специальный тег, при получении будет удален. Тег проверки защищает от старых, неактуальных пакетов, «отставших» от предыдущей ассоциации, а также от различных вторжений, позволяет избежать характерного для TCP состояния ожидания, при котором расходуются ресурсы и ограничивается общее число соединений, которые может поддерживать хост.
- Каждый из типов фрагмента включает в себя информацию заголовка TLV, который содержит тип фрагмента, флаги обработки доставки и длину поля.

Структура пакета

- Кроме того, перед фрагментом DATA будет размещаться пользовательская информация о полезной нагрузке, состоящей из номера транспортной последовательности (TSN — transport sequence number), идентификатора потока, номера последовательности потока (SSN — stream sequence number).
- TSN и SSN — два разных номера последовательности для каждого фрагмента DATA. TSN используется для обеспечения надежности каждой ассоциации, а SSN для упорядочивания по потокам. Идентификатор потока отмечает отдельные сообщения в каждом потоке.

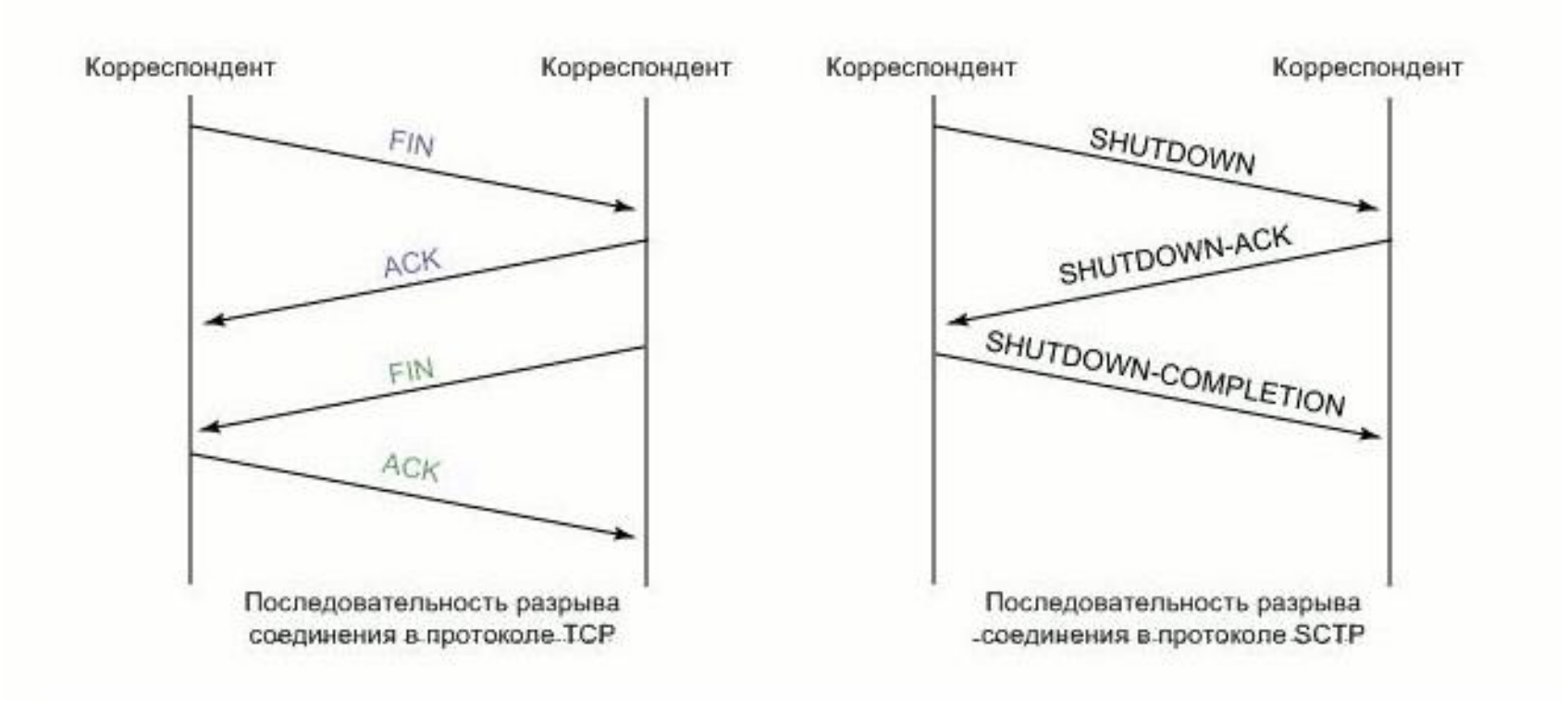
Передача данных

- Хост SCTP посылает избранные подтверждения (фрагменты SACK) в ответ на каждый пакет SCTP, сопровождающий фрагменты DATA.
- Сообщение SACK полностью описывает состояние получателя так, что отправитель может принимать решение о повторной передаче в зависимости от того, что ему уже удалось получить.
- SCTP поддерживает алгоритмы быстрой повторной передачи и повторной передачи с тайм-аутом, аналогичные тем, которые применяются в TCP.

Завершение

- SCTP использует трехэтапную процедуру установки соединения, которая имеет важное отличие от процедуры, применяемой в TCP: конечная точка TCP может инициировать процедуру отключения, сохраняя открытым соединение и получая новые данные от другого хоста.
- SCTP не поддерживает такого «наполовину закрытого» состояния, т. е. обе стороны не могут передавать новые данные на свой более высокий уровень, если инициирована последовательность постепенного отключения.

Завершение соединения



Источники

- В. Олифер, Н. Олифер “Компьютерные сети. Принципы, технологии, протоколы”
- Куроуз, Росс “Компьютерные сети. Нисходящий подход”
- Richard Stevens “TCP/IP Illustrated, vol. I”

Спасибо за внимание!