Слайд 1

Титульник

Слайд 2

Содержание

Слайд 3 (место протокола в модели OSI)

Согласно модели OSI (Open System Interconnection - Сетевая модель взаимодействия открытых систем) все программное обеспечение системы делится на 7 уровней, причем принято начинать отсчет с нижнего:

7. Прикладной уровень (application layer) **связь пользовательских приложений с сетью**

6. Представительский уровень или уровень представления (presentation layer) **преобразует данные в соответствующий формат**

5. Сеансовый уровень (session layer) **организует сеанс связи между компьютерами**

4. Транспортный уровень (transport layer) **обеспечивает надёжность передачи данных от отправителя к получателю**

3. Сетевой уровень (network layer) **определяет путь, по которому данные будут переданы**

2. Канальный уровень (data link layer) **нужен для взаимодействия сетей на физическом уровне**

1. Физический уровень (physical layer) **самый нижний уровень, непосредственно осуществляющий передачу потока данных**

Сетевая модель – это модель взаимодействия сетевых протоколов (стандартов), на каждом уровне и присутствуют свои протоколы.

Transmission Control Protocol (TCP, протокол управления передачей) — один из основных протоколов передачи данных интернета, предназначенный для управления передачей данных, перед передачей устанавливается соединение, подтверждается доставка данных, при необходимости делается повтор, гарантируется целостность и правильная последовательность загружаемых данных.

Сети и подсети, в которых совместно используются протоколы TCP и IP, называются сетями TCP/IP. В стеке протоколов IP TCP выполняет функции протокола транспортного уровня модели OSI.

Слайд 4 (как используют)

Протокол TCP предназначен для надежной и гарантированной доставки данных между хостами в компьютерных сетях с коммутацией пакетов и между такими сетями через промежуточные системы. Он освобождает прикладные процессы от необходимости использовать таймауты и повторные передачи. Для обеспечения такого сервиса на базе менее надежных коммуникационных систем Internet требуется поддержка следующих функций:

- базовый обмен данными (Basic Data Transfer);
- надежность (Reliability);
- управление потоком данных (Flow Control);
- мультиплексирование (Multiplexing);
- поддержка соединений (Connections);
- предпочтения и безопасность (Precedence and Security)

TCP используется в таких протоколах, как FTP (использует разные сетевые соединения для передачи команд и данных между клиентом и сервером - для загрузки сетевых страниц и других документов с частного устройства разработки на открытые сервера хостинга), SSH (позволяет производить удалённое управление операционной системой и туннелирование TCP-соединений), Telnet (для реализации текстового интерфейса по сети), POP3 (используется клиентами электронной почты для получения почты с удалённого сервера по TCP-соединению), SMTP (предназначенный для передачи электронной почты в сетях TCP/IP), SIP (описывает способ установления и завершения пользовательского интернет-сеанса, включающего обмен мультимедийным содержимым), DNS (компьютерная распределённая система для получения информации о доменах), SSL (стандартная интернет технология безопасности, которая используется, чтобы обеспечить зашифрованное соединение между веб-сервером (сайтом) и браузером), HTTP (доступ к веб-ресурсам (Ваш браузер, получая данные с сервера, знает, как их требуется обработать, и успешно обрабатывает их, показывая Вам запрашиваемую информацию)), HTTPS (расширение протокола HTTP для поддержки шифрования в целях повышения безопасности), NTP (для синхронизации серверов точного времени), BGP (динамический протокол маршрутизации), SNMP (интернет-протокол для управления устройствами в IP-сетях).

Наиболее типичными протоколами использующими TCP, являются FTP (File Transfer Protocol - протокол передачи файлов) и TELNET. Кроме того, TCP используют система X-Window, rcp (remote copy - удаленное копирование) и другие "r-команды". Большие возможности TCP даются не бесплатно. Реализация TCP требует большой производительности процессора и большой пропускной способности сети. Внутренняя структура модуля TCP гораздо сложнее структуры модуля UDP.

Слайд 5 (что может)

При стадии передачи данных главной функцией этого протокола является обеспечение надежности доставки сообщения. TCP обеспечивает её с помощью механизма подтверждения принятия сообщения:

1. Отправитель посылает сообщение получателю.
2. Отправитель устанавливает таймер, который фиксирует максимальный промежуток времени, в течение которого отправитель должен получить подтверждение от получателя об успешной доставке.
3. Если подтверждение за отведенный промежуток времени получено не было, выполняется повторная передача, иначе передается следующее сообщение.

Слайд 6 (взаимодействие с другими протоколами)

Протокол TCP был спроектирован в качестве связующего протокола для обеспечения интерактивной работы между компьютерами. TCP обеспечивает надежность и достоверность обмена данными между процессами на компьютерах, входящих в общую сеть TCP, с одной стороны, взаимодействует с прикладным протоколом пользовательского приложения, а с другой, с протоколом, обеспечивающим "низкоуровневые" функции: маршрутизацию и адресацию пакетов, которые выполняет IP.

Протокол IP занимается пересылкой дейтаграмм по сети, никак не гарантируя доставку, целостность, порядок прибытия информации и готовность получателя к приему данных; все эти задачи возложены на протокол TCP.

При получении дейтаграммы, в поле Protocol которой указан код протокола TCP , модуль IP передает данные этой дейтаграммы модулю TCP. Эти данные представляют собой TCP-сегмент, содержащий TCP-заголовок и данные пользователя (прикладного процесса). Модуль TCP анализирует служебную информацию заголовка, определяет, какому именно процессу предназначены данные пользователя, проверяет целостность и порядок прихода данных и подтверждает их прием другой стороне. По мере получения правильной последовательности неискаженных данных пользователя они передаются прикладному процесс

В операционной системе реализация TCP представляет собой отдельный системный модуль (драйвер), через который, как правило, проходят все вызовы функций протокола. Интерфейс между прикладным процессом и TCP представляет собой библиотеку вызовов, такую же как библиотека системных вызовов, например, для работы с файлами. Соединение открывается и данные могут быть отправлены или приняты по открытому соединению аналогично операциям чтения и записи в файл, затем соединение должно быть закрыто. Вызовы TCP могут работать с приложением в асинхронном режиме. Реализация TCP в каждой системе может предложить много собственных функций, однако любая из этих реализаций должна обеспечивать необходимый минимум функциональности, предусмотренный стандартами TCP.

Схема работы пользовательского приложения с TCP в общих чертах состоит в следующем. Для передачи данных пользовательскому процессу надо вызвать соответствующую функцию TCP, с указанием на буфер передаваемых данных. TCP упаковывает эти данные в сегменты своего стека и вызывает функцию передачи протокола нижнего уровня, например IP.

На другом конце, получатель TCP группирует поступившие от протокола нижнего уровня данные в принимающие сегменты своего буфера, проверяет целостность данных, передает данные пользовательскому процессу и уведомляет отправителя об их получении.

Слайд 7 (история протокола)

Джон Постел из университета Южной Калифорнии в 1981 определил протокол TCP в RFC-793.

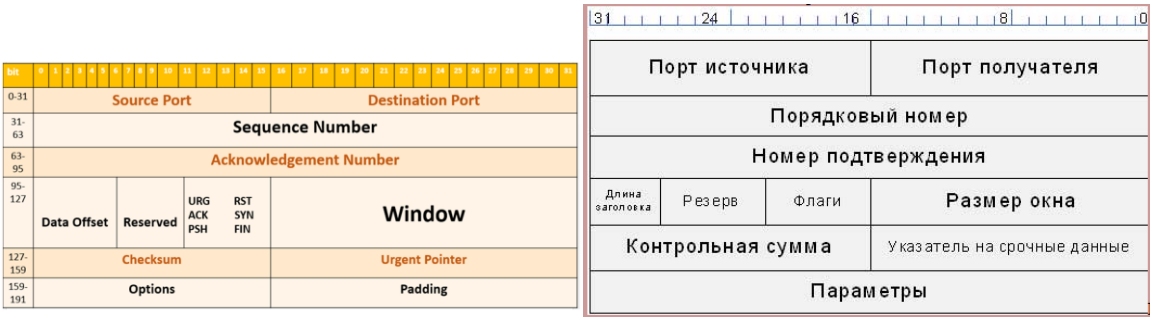
Стандарт Министерства обороны США для протокола управления передачей TCP (Transmission Control Protocol). До этого было выпущено девять предварительных редакций спецификации ARPA TCP, на которых основан данный стандарт, и текст документа тесно связан с предварительными спецификациями.

Слайд 8-11 (формат заголовка и назначение полей)

Протокол управления передачей TCP является обязательным стандартом TCP/IP и предоставляет надежную службу доставки пакетов, ориентированную на установление соединения. Протокол TCP:

- гарантирует доставку IP-датаграмм;
- выполняет разбиение на сегменты и сборку больших блоков данных, отправляемых программами;
- обеспечивает доставку сегментов данных в нужном порядке;
- выполняет проверку целостности переданных данных с помощью контрольной суммы;
- посылает положительные подтверждения, если данные получены успешно. Используя избирательные подтверждения, можно также посылать отрицательные подтверждения для данных, которые не были получены;
- предлагает предпочтительный транспорт для программ, которым требуется надежная передача данных с установлением сеанса связи, например для баз данных «клиент-сервер» и программ электронной почты.

TCP основан на связи «точка-точка» между двумя узлами сети. TCP получает данные от программ и обрабатывает их как поток байтов. Байты группируются в сегменты, которым TCP присваивает последовательные номера, необходимые для правильной сборки сегментов на узле-приемнике. Синхронизируются номера последовательности и передается управляющая информация, необходимая для установления виртуального соединения между узлами. TCP-сегменты инкапсулируются и передаются в IP-датаграммах, как показано на рисунке.



Номер порта отправителя – Source Port(16 бит) – содержит номер порта, с которого был отправлен пакет, когда это имеет значение (например, отправитель ожидает ответа). Если это поле не используется, оно заполняется нулями.

Номер порта назначения – Destination Port(16 бит) – содержит номер порта, на который будет доставлен пакет.

Порядковый номер – Sequence number(32 бита) – значение, присвоенное пакету TCP, определяющее номер стартового байта пакета, если не установлен бит SYN. Если установлен указанный бит, то порядковый номер является начальным порядковым номером (ISN) и первый байт данных равен ISN+ 1;

Номер подтверждения – Acknowledgment Number(32 бита) – значение, отсылаемое принимающей станцией отправителю, подтверждающее прием переданного ранее пакета (пакетов). Оно задает следующий порядковый номер, который целевая станция ожидает получить при установленном бите ACK. При установленном соединении подтверждение отправляется всегда.

Смещение данных – Data Offset(4 бита) – задает длину заголовка TCP(количество 32-битовых слов в заголовкеTCP);

Резервное поле – Reserved(6 бит) – зарезервировано.

Окно – Window(16 бит) – содержит количество байт данных, которое отправитель данного сегмента может принять, отсчитанное от номера байта, указанного в поле Acknowledgment Number.

Поле контрольной суммы – Checksum(16 бит) – представляет собой побитное дополнение 16-битной суммы 16-битных слов заголовка и данных дополненных нулевым байтом, если сегмент содержит нечетное число байт заголовка и данных. При вычислении контрольной суммы поле контрольной суммы полагается равным нулю.

Указатель срочных данных – Urgent Pointer(16 бит) – содержит значение счетчика байтов, начиная с которого следуют данные повышенной срочности. Данное поле интерпретируется только в пакетах с установленным флагом URG;

Опции – Options– имеет переменную длину и содержит дополнительные параметры;

Заполнение – Padding– имеет переменную длину и используется для выравнивания заголовка по 32-битному слову нулевыми значениями.

Слайд 12 (флаги управления)

Поле флаги (Code Bits) занимает 6 бит и содержит шесть 1-битовых флагов:

- *URG* – флаг срочности, применяется при посылке сообщения получателю, ожидающему приема экстренной информации (устанавливается в 1 в случае использования поля указатель на срочные данные)
- *ACK* – флаг пакета, содержащего подтверждение получения (устанавливается в 1 в случае, если поле номер подтверждения содержит данные. В противном случае это поле игнорируется)
- *PSH* – флаг выталкивания, немедленная отсылка данных после считывания данных этого пакета (означает, что принимающий стек TCP должен немедленно информировать приложение о поступивших данных, а не ждать пока буфер заполнится)
- *RST* – флаг переустановки соединения (используется для отмены соединения: из-за ошибки приложения, отказа от неверного сегмента, попытки создать соединение при отсутствии затребованного сервиса)
- *SYN* – флаг синхронизации чисел последовательности (устанавливается при инициировании соединения и синхронизации порядкового номера)
- *FIN* – флаг окончания передачи со стороны отправителя (используется для разрыва соединения. Он указывает, что отправитель закончил передачу данных)

Слайд 13 (порты TCP протокола)

В современных компьютерных сетях из стека сетевых протоколов TCP/IP на транспортном уровне чаще всего используются TCP и UDP. Две конечные точки (хосты) при установке соединения по этим протоколам идентифицируются согласно номерам портов. Номера портов, используемые для конкретных специфических целей, выделяет и регистрирует IANA (Internet Assigned Numbers Authority), однако на практике часто встречаются случаи их неофициального применения.

Количество портов ограничено с учётом 16-битной адресации (2^{16}=65536, начало — «0»). Все порты разделены на три диапазона — общеизвестные (или системные, 0—1023), зарегистрированные (или пользовательские, 1024—49151) и динамические (или частные, 49152—65535).

Первоначально номера портов использовались в ARPANET протоколом NCP. Передача велась в полудуплексном режиме, и для соединения требовалось два порта. С принятием протоколов TCP и UDP необходимым остался только один порт, и чётные номера не применялись — этим объясняется отсутствие регистрации некоторых портов из диапазона общеизвестных.

Номера портов TCP и UDP используются также протоколами SCTP и DCCP. Службы в SCTP и DCCP обычно используют номера, соответствующие их реализациям в TCP и UDP (при наличии).

Слайд 14 (порты)

Основные, наиболее часто используемые, порты:

21 — FTP (**File Transfer Protocol** — *протокол передачи файлов*) — протокол, предназначенный для передачи файлов в компьютерных сетях. FTP позволяет подключаться к серверам, просматривать содержимое каталогов и загружать файлы с сервера или на сервер.

22 — SSH (**Secure Shell** — безопасная оболочка) — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов).

23 — TELNET (**TELecommunication NETwork**) — сетевой протокол для реализации текстового интерфейса по сети (при помощи транспорта TCP). Позволяет управлять Операционной Системой удаленно.

25 — SMTP (**Simple Mail Transfer Protocol** — простой протокол передачи почты) — это сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP.

53 — DNS (**Domain Name System** — система доменных имён) — распределённая система, способная по запросу, содержащему доменное имя хоста (компьютера или другого сетевого устройства), сообщить IP адрес или (в зависимости от запроса) другую информацию. DNS работает в сетях TCP/IP.

79 — Finger (Фингер) — сетевой протокол, предназначенный для предоставления информации о пользователях удалённого компьютера.

80 — WWW (web-сервер) — показывает присутствует ли web-сервер на машине.

110 — POP3 (**Post Office Protocol Version 3** — протокол почтового отделения, версия 3) используется почтовым клиентом для получения сообщений электронной почты с сервера.

111 — Sun RPC. Система удаленного вызова процедур.

119 — (**Network News Transfer Protocol**) — сетевой протокол, используемый для обмена сообщениями в группах новостей.

139 — NetBIOS (**Network Basic Input/Output System**) — протокол для работы в локальных сетях на персональных ЭВМ типа IBM/PC, разработан в виде интерфейса, который не зависит от фирмы-производителя.

443 — HTTPS (**Hypertext Transfer Protocol Secure**) — расширение протокола HTTP, поддерживающее шифрование.

513 — rLogin (**Remote LOGIN** — удаленный вход в систему) позволяет пользователям UNIX подключаться к системам UNIX на других машинах через сеть Internet и работать так же, как при прямом подключении терминала к машине. Этот протокол используется аналогично протоколу TELNET.

Слайд 15-16 (порядковый номер - подтверждение порядкового номера)

Одним из фундаментальных аспектов TCP является нумерация данных — каждый октет, передаваемый через соединение TCP имеет свой порядковый номер. Поскольку каждый октет пронумерован, для любого из октетов может быть передано подтверждение (acknowledgment). Механизм подтверждений является кумулятивным (накопительным), поэтому подтверждение для порядкового номера X показывает, что все октеты до X (но не включая сам октет с номером X) были получены. Этот механизм позволяет обнаруживать дубликаты данных при использовании повторной передачи. Нумерация октетов в сегменте начинается от заголовка, т. е. октет, следующий сразу после заголовка, имеет наименьший порядковый номер, а номера следующих октетов последовательно возрастают.

Важно помнить, что реальное пространство порядковых номеров имеет ограниченные размеры, хотя и достаточно велико (от 0 до 2^32 - 1). Поскольку число порядковых номеров конечно, все арифметические операции с порядковыми номерами выполняются по модулю 2^32. Такая беззнаковая арифметика сохраняет соотношения между порядковыми номерами при переходе номера от 2^32 - 1 к нулю. В такой арифметике с использованием модуля существуют некоторые тонкости, которые следует принимать во внимание при разработке программ, использующих сравнение значений. Символ =< означает "меньше или равно" (модуль 2^32).

Типичные операции сравнения порядковых номеров, используемые TCP, включают:

- Проверка того, что подтверждение указывает на некоторый порядковый номер для посланных, но еще не подтвержденных данных.
- Проверка того, что все порядковые номера, занимаемые сегментом, имеют подтверждение (например, для удаления сегмента из очереди повторной передачи).
- Проверка того, что входящий сегмент содержит ожидаемые порядковые номера (например, чтобы убедиться в том, что сегмент "вписывается" в окно приема).

Слайд 17 (состояния сеанса TCP)

Упрощённая диаграмма состояний TCP. Более подробно в (на английском языке)

Состояния сеанса TCP

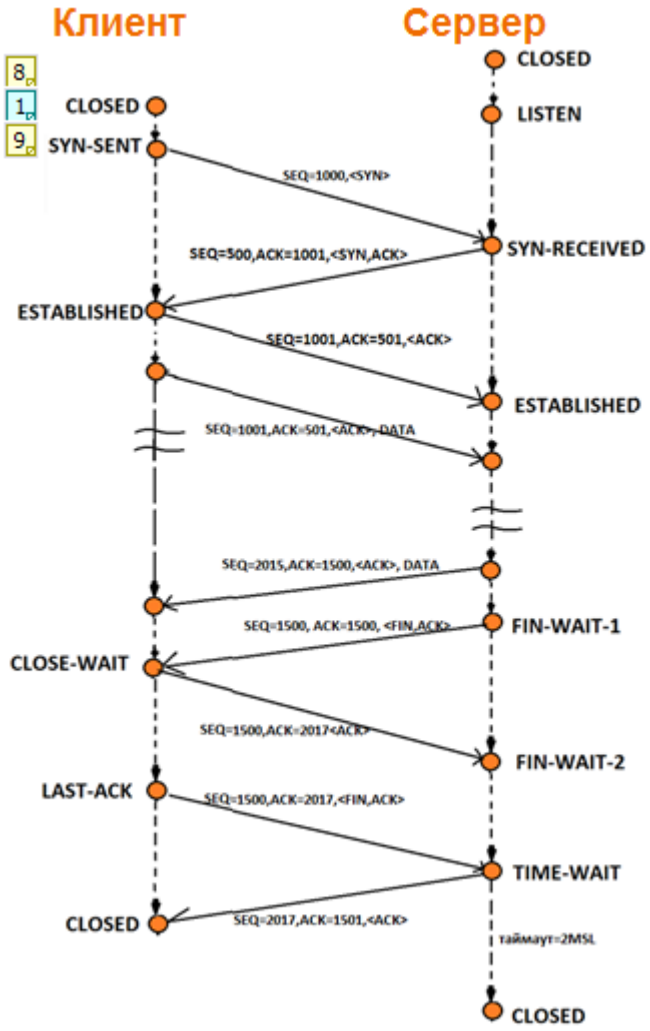
CLOSED	Начальное состояние узла. Фактически фиктивное
LISTEN	Сервер ожидает запросов установления соединения от клиента
SYN-SENT	Клиент отправил запрос серверу на установление соединения и ожидает ответа
SYN-RECEIVED	Сервер получил запрос на соединение, отправил ответный запрос и ожидает подтверждения
ESTABLISHED	Соединение установлено, идёт передача данных
FIN-WAIT-1	Одна из сторон (назовём её узел-1) завершает соединение, отправив сегмент с флагом FIN
CLOSE-WAIT	Другая сторона (узел-2) переходит в это состояние, отправив, в свою очередь сегмент ACK и продолжает одностороннюю передачу
FIN-WAIT-2	Узел-1 получает ACK, продолжает чтение и ждёт получения сегмента с флагом FIN
LAST-ACK	Узел-2 заканчивает передачу и отправляет сегмент с флагом FIN
TIME-WAIT	Узел-1 получил сегмент с флагом FIN, отправил сегмент с флагом ACK и ждёт 2*MSL секунд, перед окончательным закрытием соединения
CLOSING	Обе стороны инициировали закрытие соединения одновременно: после отправки сегмента с флагом FIN узел-1 также получает сегмент FIN, отправляет ACK и находится в ожидании сегмента ACK (подтверждения на свой запрос о разъединении)

Слайд 18 (установление связи TCP сеанса - сценарий MCS)

Начальная фаза сеанса передачи получила название **"тройное рукопожатие"** (three-way handshake), [несмотря на то что возможен процесс установления соединения с использованием четырёх сегментов (SYN в сторону сервера, ACK в сторону клиента, SYN в сторону клиента, ACK в сторону сервера), на практике для экономии времени используется три сегмента] которое достаточно точно отражает процесс обмена служебными сегментами между сторонами (Клиент-Сервер).

Клиентом, инициирует начало сеанса, посылая другой стороне — серверу сегмент SYN.

Как правило этот сегмент является числом служебным, т.е. не содержит полезных данных, его заголовок определяет номер порта и начальный порядковый номер потока клиент-сервер. Если сервер готов принять данные от клиента, он создает логический канал (размещая соответствующие структуры данных) и отправляет клиенту сегмент с установленным начальным порядковым номером потока сервер-клиент и флагами SYN и ACK, подтверждающий получение сегмента SYN и выражающего готовность сервера к получению данных. Наконец, и это третье рукопожатие, клиент отвечает сегментом с установленным флагом ACK, подтверждающим получение ответа от сервера и тем самым завершающим фазу создания TCP-канала.



После этого обе стороны начинают передачу TCP-сегментов, каждый из которых содержит подтверждение полученных данных и новое значение окна. Начиная с подтвержденного октета, источник может передать, не дожидаясь подтверждения, количество данных, определенных значением окна. Если отправитель не получает подтверждения на посланные данные в течение определенного промежутка времени, он полагает, что данные утеряны, и их передача повторяется, начиная с последнего подтвержденного октета. Поскольку надежность передачи гарантируется протоколом, для данных приложения, переданных, но не подтвержденных, протокол хранит копию, которая уничтожается после получения подтверждения или вновь передается при отсутствии такового. Получение дублированных данных также подтверждается, хотя сами данные уничтожаются, поскольку дублирование могло быть вызвано неполучением подтверждения. Если одна из сторон получает неупорядоченные данные, они, как правило, сохраняются до получения недостающих последовательных сегментов.

Разумеется, получение таких неупорядоченных данных не подтверждается, поскольку подтверждение отправляется только на полученный непрерывный последовательный поток октетов.

Завершение сеанса в TCP происходит в несколько этапов. Любая из сторон может завершить передачу данных, отправив сегмент с установленным флагом FIN. Получение такого сегмента подтверждается другой стороной и эквивалентно достижению конца файла при его чтении. Однако другая сторона может продолжать передавать данные, также впоследствии завершив передачу сегментом FIN. Подтверждение этого сегмента полностью разрушает канал и завершает сеанс. Для того чтобы гарантировать синхронизацию завершения сеанса, сторона, отправившая подтверждение на последний сегмент FIN, должна поддерживать сеанс достаточно долго, чтобы иметь возможность вновь подтвердить повторные сегменты FIN.

Также проиллюстрированы **состояния коммуникационных узлов TCP-канала.** Как видно из рисунка, начальное состояние узла (сервера или клиента) — состояние CLOSED. Готовность сервера к обработке инициирующих запросов от клиента определяется переходом его в состояние LISTEN. С этого момента сервер может принимать и обрабатывать инициирующие сеанс сегменты SYN. При отправлении такого сегмента клиент переходит в состояние SYN-SENT и ожидает ответного запроса от сервера. Сервер при получении сегмента также отправляет сегмент SYN с подтверждением ACK и переходит в состояние SYN-RECEIVED. Подтверждение от клиента завершает "рукопожатие" и сеанс переходит в состояние ESTABLISHED. После завершения обмена данными одна из сторон (например, клиент) отправляет сегмент FIN, переходя при этом в состояние FIN-WAIT-1. Приняв этот сегмент другая сторона (например, сервер) отправляет подтверждение ACK и переходит в состояние CLOSE-WAIT, при этом канал становится симплексным — передача данных возможна только в направлении от сервера к клиенту. Когда клиент получает подтверждение он переходит в состояние FIN-WAIT-2, в котором находится до получения сегмента FIN. После подтверждения получения этого сегмента канал окончательно разрушается. Для обеспечения правильной обработки данных для каждого логического TCP-канала хранится полная информация о его состоянии, различных таймерах и о текущих порядковых номерах переданных и принятых октетов. Это необходимо, например, для корректной обработки служебных сегментов SYN и FIN.

Процесс называется «трёхэтапным согласованием» (*three way handshake*), так как несмотря на то что возможен процесс установления соединения с использованием четырёх сегментов (SYN в сторону сервера, ACK в сторону клиента, SYN в сторону клиента, ACK в сторону сервера), на практике для экономии времени используется три сегмента.

Пример базового 3-этапного согласования:

```
TCP A                                                TCP B
   1.  CLOSED                                           LISTEN
   2.  SYN-SENT    --> <SEQ=100><CTL=SYN>               --> SYN-RECEIVED
   3.  ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK>  <-- SYN-RECEIVED
   4.  ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK>      --> ESTABLISHED
   5.  ESTABLISHED <-- <SEQ=301><ACK=101><CTL=ACK>      <-- ESTABLISHED
```

В строке 2 TCP A начинает передачу сегмента SYN, говорящего об использовании номеров последовательности, начиная со 100. В строке 3 TCP B передает SYN и

подтверждение для принятого SYN в адрес TCP A. Надо отметить, что поле подтверждения показывает ожидание TCP B приёма номера последовательности 101, подтверждающего SYN с номером 100.

В строке 4 TCP A отвечает пустым сегментом с подтверждением ACK для сегмента SYN от TCP B; в строке 5 TCP B передает некоторые данные. Отметим, что номер подтверждения сегмента в строке 5 (ACK=101) совпадает с номером последовательности в строке 4 (SEQ=101), поскольку ACK не занимает пространства номеров последовательности (если это сделать, придется подтверждать подтверждения — ACK для ACK).

Слайд 19 (механизм действия протокола)

В отличие от традиционной альтернативы — UDP, который может сразу же начать передачу пакетов, TCP устанавливает соединения, которые должны быть созданы перед передачей данных. TCP соединение можно разделить на 3 стадии:

- Установка соединения
- Передача данных
- Завершение соединения

Более подробно эти стадии были рассмотрены в пункте **сценарии MSC**.

Слайд 20-21 (установка соединения)

Процесс начала сеанса TCP (также называемый «рукопожатие» (*handshake*)), состоит из трёх шагов.

1. Клиент, который намеревается установить соединение, посылает серверу сегмент с номером последовательности и флагом SYN.

    - Сервер получает сегмент, запоминает номер последовательности и пытается создать сокет (буферы и управляющие структуры памяти) для обслуживания нового клиента.
        - В случае успеха сервер посылает клиенту сегмент с номером последовательности и флагами SYN и ACK, и переходит в состояние SYN-RECEIVED.
        - В случае неудачи сервер посылает клиенту сегмент с флагом RST.

2. Если клиент получает сегмент с флагом SYN, то он запоминает номер последовательности и посылает сегмент с флагом ACK.

    - Если клиент одновременно получает и флаг ACK (что обычно и происходит), то он переходит в состояние ESTABLISHED.
    - Если клиент получает сегмент с флагом RST, то он прекращает попытки соединиться.
    - Если клиент не получает ответа в течение 10 секунд, то он повторяет процесс соединения заново.

3. Если сервер в состоянии SYN-RECEIVED получает сегмент с флагом ACK, то он переходит в состояние ESTABLISHED.

- В противном случае после тайм-аута он закрывает сокет и переходит в состояние CLOSED.

Слайд 25 (передача данных)

При обмене данными приёмник использует номер последовательности, содержащийся в получаемых сегментах, для восстановления их исходного порядка. Приёмник уведомляет передающую сторону о номере последовательности, до которой он успешно получил данные, включая его в поле «номер подтверждения». Все получаемые данные, относящиеся к промежутку подтвержденных последовательностей, игнорируются. Если полученный сегмент содержит номер последовательности больший, чем ожидаемый, то данные из сегмента буферизируются, но номер подтвержденной последовательности не изменяется. Если впоследствии будет принят сегмент, относящийся к ожидаемому номеру последовательности, то порядок данных будет автоматически восстановлен исходя из номеров последовательностей в сегментах.

Для того, чтобы передающая сторона не отправляла данные интенсивнее, чем их может обработать приёмник, TCP содержит средства управления потоком. Для этого используется поле «окно». В сегментах, направляемых от приёмника передающей стороне, в поле «окно» указывается текущий размер приёмного буфера. Передающая сторона сохраняет размер окна и отправляет данных не более, чем указал приёмник. Если приёмник указал нулевой размер окна, то передача данных в направлении этого узла не происходит, пока приёмник не сообщит о большем размере окна.

В некоторых случаях передающее приложение может явно затребовать протолкнуть данные до некоторой последовательности принимающему приложению, не буферизируя их. Для этого используется флаг PSH. Если в полученном сегменте обнаруживается флаг PSH, то реализация TCP отдает все буферизированные на текущий момент данные принимающему приложению. «Проталкивание» используется, например, в интерактивных приложениях. В сетевых терминалах нет смысла ожидать ввода пользователя после того, как он закончил набирать команду. Поэтому последний сегмент, содержащий команду, обязан содержать флаг PSH, чтобы приложение на принимающей стороне смогло начать её выполнение.

Слайд 26-27 (завершение соединения)

Завершение соединения можно рассмотреть в три этапа:

1. Посылка серверу от клиента флага FIN на завершение соединения.
2. Сервер посылает клиенту флаги ответа ACK , FIN, что соединение закрыто.
3. После получения этих флагов клиент закрывает соединение и в подтверждение отправляет серверу ACK , что соединение закрыто.

Слайд 31 (уязвимость протокола)

Уязвимости протоколов, входящих в стек TCP/IP обусловлены, как правило, слабой аутентификацией, ограничением размера буфера, отсутствием проверки корректности служебной информации и т.д..

Протокол TCP наиболее уязвим при отсутствии механизма проверки корректности заполнения служебных заголовков пакета, что вызывает существенное снижение скорости обмена и даже полный разрыв произвольных соединений по протоколу.

Слайд 32 (основные типу угроз связанные с протоколом)

1. Подмена одного из субъектов TCP-соединения в сети Internet

Для идентификации TCP-пакета в TCP-заголовке существуют два 32-разрядных идентификатора, которые также играют роль счетчика пакетов. Их названия - **Sequence Number** (номер последовательности) и **Acknowledgment Number** (номер подтверждения).

Для формирования ложного TCP-пакета атакующему необходимо знать текущие идентификаторы для данного соединения. Это значит, что ему достаточно, подобрав соответствующие текущие значения идентификаторов TCP-пакета для данного TCP-соединения послать пакет с любого хоста в Сети от имени одного из участников данного соединения, и данный пакет будет воспринят как верный.

При нахождении взломщика и объекта атаки в одном сегменте, задача получения значений идентификаторов решается анализом сетевого трафика. Если же они находятся в разных сегментах, приходится пользоваться математическим предсказанием начального значения идентификатора экстраполяцией его предыдущих значений.

Для защиты от таких атак необходимо использовать ОС, в которых начальное значение идентификатора генерируется действительно случайным образом. Также необходимо использовать защищённые протоколы типа SSL, S-HTTP, Kerberos и т.д.

2. Направленный шторм ложных TCP-запросов на создание соединения

На каждый полученный TCP-запрос на создание соединения операционная система должна сгенерировать начальное значение идентификатора ISN и отослать его в ответ на запросивший хост. При этом, так как в сети Internet (стандарта IPv4) не предусмотрен контроль за IP-адресом отправителя сообщения, то невозможно отследить истинный маршрут, пройденный IP-пакетом, и, следовательно, у конечных абонентов сети нет возможности ограничить число возможных запросов, принимаемых в единицу времени от одного хоста. Поэтому возможно осуществление типовой атаки "Отказ в обслуживании", которая будет заключаться в передаче на атакуемый хост как можно большего числа ложных TCP-запросов на создание соединения от имени любого хоста в сети. При этом атакуемая сетевая ОС в зависимости от вычислительной мощности компьютера либо – в худшем случае – практически зависает, либо – в лучшем случае – перестает реагировать на легальные запросы на подключение (отказ в обслуживании).

Это происходит из-за того, что для всей массы полученных ложных запросов система должна, во-первых, сохранить в памяти полученную в каждом запросе информацию и, во-вторых, выработать и отослать ответ на каждый запрос. Таким образом, все ресурсы системы "съедаются" ложными запросами: переполняется очередь запросов, и система занимается только их обработкой.

Недавно в Сети был отмечен новый тип атак. Вместо типичных атак Denial of Service хакеры переполняют буфер пакетов корпоративных роутеров не с единичных машин, а с целых тысяч компьютеров-зомби.

Такие атаки способны блокировать каналы мощностью вплоть до T3 (44.736 Мбит/с) и уже отмечено несколько таких случаев. Опасность атаки становится тем важнее, чем больше бизнесов используют частные сети типа VPN и другие Интернет-технологии. Ведь отказ канала у публичного провайдера приведет в этом случае не просто к отключению отдельных пользователей, а к остановке работы огромных корпораций.

В этом случае существуют трудности в определении источника атаки - ложные пакеты идут с различных неповторяющихся IP-адресов. "Зомби-атаку" называют самой сложной из известных. На одинокую жертву нападает целая армия, и каждый зомби бьет только один раз.

Приемлемых способов защиты от подобных атак в сети стандарта IPv4 нет, так как невозможен контроль за маршрутом сообщений. Для повышения надёжности работы системы можно использовать по возможности более мощные компьютеры, способные выдержать направленный шторм ложных запросов на создание соединения.

3. WinNuke – атака Windows-систем передачей пакетов TCP/IP с флагом Out Of Band (OOB) на открытый TCP-порт. На сегодняшний день эта атака устарела. Ранние версии Windows95/NT зависали.

Слайд 33 (защита)

Итак, для защиты от атак необходимо использовать комплекс средств безопасности, реализующий основные защитные механизмы и состоящий из следующих компонентов:

• Межсетевые экраны, являющиеся первой линией обороны и реализующие комплекс защитных механизмов, называемый защитой периметра.

• Средства анализа защищённости, позволяющие оценить эффективность работы средств защиты и обнаружить уязвимости узлов, протоколов, служб.

• Средства обнаружения атак, осуществляющие мониторинг в реальном режиме времени.