Лабораторная работа № 6

Изучение методов анализа и исследования сетевого трафика на примере работы протоколов транспортного уровня TCP и UDP

Цель работы. Анализ и исследование TCP и UDP трафика на основе наблюдений за обменом датаграммами и изучение сеанса передачи данных в рамках TCP сессии.

Теоретические сведения

Протокол TCP (Transmission Control Protocol)

TCP — это протокол обеспечения надежности прямых соединений, созданный для многоуровневой иерархии протоколов, поддерживающих межсетевые приложения. Протокол TCP обеспечивает надежность коммуникаций между парами процессов на узлах, включенных в различные компьютерные коммуникационные сети, которые объединены в единую систему.

TCP занимает в многоуровневой архитектуре протоколов нишу непосредственно над IP, который позволяет протоколу TCP отправлять и получать сегменты информации переменной длины, заключенные в оболочку IP датаграмм. IP датаграмма предоставляет средства для адресации отправителя и получателя сегментов TCP в различных сетях. IP также осуществляет любую фрагментацию и сборку сегментов TCP, необходимую для осуществления передачи и доставки через множество сетей и промежуточных шлюзов. IP также обрабатывает информацию о приоритете, классификации безопасности, а также осуществляет разграничение TCP сегментов. Так что информация с помощью TCP может быть передана напрямую через множество сетей.

Протокол TCP взаимодействует с одной стороны с пользователем или прикладной программой, а с другой — с протоколом более низкого уровня, таким как IP.

Взаимодействие между протоколом TCP и протоколами более низкого уровня формализовано достаточно слабо, за исключением того, что должен существовать механизм, с помощью которого эти два уровня могут асинхронно обмениваться информацией друг с другом. Обычно полагают, что протокол нижнего уровня задает это взаимодействие. Протокол TCP спроектирован так, чтобы работать с разнообразной средой объединенных компьютерных сетей.

Протокол TCP способен передавать непрерывные потоки октетов между своими клиентами в обоих направлениях, пакуя некое количество октетов в сегменты для передачи через сеть. В общем случае протокол TCP решает по своему усмотрению, когда производить блокировку и передачу данных.

Иногда пользователям бывает необходимо убедиться в том, что все данные, переданные ими протоколу TCP, уже отправлены. Для этой цели определена функция проталкивания (PUSH). Чтобы убедиться в том, что данные, отправленные протоколу TCP, действительно переданы, отправитель указывает, что их следует протолкнуть к получателю.

Проталкивание приводит к тому, что программы протокола TCP сразу осуществляют отправление и, соответственно, получение остающихся данных. Правильно осуществленное проталкивание может быть невидимо для получателя, а сама функция проталкивания может не иметь маркера границы записи.

Достоверность

Протокол TCP должен иметь защиту от разрушения данных, потери, дублирования и нарушения очередности получения, вызываемых коммуникационной средой. Это

достигается присвоением очередного номера каждому передаваемому октету, а также требованием подтверждения (ACK) от программы TCP, принимающей данные. Если подтверждения не получено в течении контрольного интервала времени, то данные посылаются повторно. Со стороны получателя номера очереди используются для восстановления очередности сегментов, которые могут быть получены в неправильном порядке, а также для ограничения возможности появления дубликатов.

Повреждения фиксируются посредством добавления к каждому передаваемому сегменту контрольной суммы, проверки ее при получении и последующей ликвидации дефектных сегментов.

До тех пор, пока программы протокола TCP продолжают функционировать корректно, а сеть не распалась полностью на составные части, ошибки пересылки не будут влиять на правильное получение данных. Протокол TCP защищает от ошибок коммуникационной среды.

Управление потоком

Протокол TCP дает средства получателю управлять количеством данных, посылаемых ему отправителем. Это достигается возвратом так называемого «окна» (window) вместе с каждым подтверждением, которое указывает диапазон приемлемых номеров, следующих за номером последнего успешно принятого сегмента. Окно определяет количество октетов, которое отправитель может послать до получения дальнейших указаний.

Разделение каналов

Чтобы позволить на отдельно взятом узле многим процессам одновременно использовать коммуникационные возможности уровня TCP, протокол TCP предоставляет на каждом узле набор адресов или портов. Вместе с адресами сетей и узлов на коммуникационном уровне они образуют сокет (socket).

Каждое соединение уникальным образом идентифицируется парой сокетов. Таким образом, любой сокет может одновременно использоваться во многих соединениях.

Соотнесение портов и процессов осуществляется каждым узлом самостоятельно. Однако, для часто используемых процессов, таких как HTTP серверы или серверы электронной почты, используются фиксированные документированные порты.

Работа с соединениями

Механизмы управления потоком и обеспечения достоверности, описанные выше, требуют, чтобы программы протокола TCP инициализировали и поддерживали определенную информацию о состоянии каждого потока данных. Набор такой информации, включающий сокеты, номера очереди, размеры окон, называется соединением. Каждое соединение уникальным образом идентифицируется парой сокетов на двух концах.

Если два процесса желают обмениваться информацией, соответствующие процессы протокола TCP должны сперва установить соединение, то есть инициализировать информацию о статусе на каждой стороне. По завершении обмена информацией соединение должно быть расторгнуто или закрыто, чтобы освободить ресурсы для предоставления другим пользователям.

Поскольку соединения должны устанавливаться между ненадежными узлами и через ненадежную коммуникационную среду, то во избежание ошибочной инициализации соединений используется механизм подтверждения связи с хронометрированными номерами очереди.

Общие принципы работы протокола TCP

TCP может применяться как в локальных так и в больших сетях, но в любом случае они должны основываться на технологии коммутации пакетов. Реальными агентами, создающими и потребляющими сообщения, циркулирующие в сети, являются процессы.

Согласно наиболее общему определению процессов как исполняющихся программ, они рассматриваются как активные элементы на узлах. Говоря о TCP, любые коммуникации рассматриваются как коммуникации между процессами.

Поскольку процесс может контролировать несколько коммуникационных потоков, ведущих от него к другому процессу или процессам, каждый процесс может иметь набор портов, через которые он общается с портами других процессов.

Процесс пересылает данные, вызывая программу протокола TCP и передавая ей в качестве аргументов буферы с данными. Протокол TCP пакует данные из этих буферов в сегменты, а затем вызывает модуль IP для передачи каждого сегмента программе протокола TCP, являющейся адресатом. Этот адресат в свою очередь помещает данные из сегмента в буферы получателя и затем оповещает своего клиента о прибытии предназначенных ему данных. Программы протокола TCP помещают в сегменты контрольную информацию, которая затем используется ими для проверки очередности передачи данных.

Модель Internet коммуникаций состоит в том, что с каждой программой протокола TCP связан модуль IP, обеспечивающий ей интерфейс с локальной сетью. Данный модуль IP помещает сегменты TCP в IP датаграммы, а затем направляет их на другой IP модуль или на промежуточный шлюз. Для передачи датаграммы по локальной сети она в свою очередь помещается в пакет соответствующего типа.

Коммутаторы пакетов могут осуществлять дальнейшую упаковку, фрагментацию или другие операции с тем, чтобы в локальной сети осуществить передачу пакетов по назначению на модуль IP.

На шлюзах между локальными сетями датаграмма Internet освобождается от пакета локальной сети и исследуется с тем, чтобы определить, по какой сети она должна в дальнейшем идти. Затем IP датаграмма упаковывается в пакет, соответствующий выбранной локальной сети, и посылается на следующий шлюз или же прямо к конечному получателю.

Шлюз имеет возможность разбивать IP датаграмму на более мелкие датаграммы — фрагменты, если это необходимо для передачи по очередной локальной сети. Чтобы осуществить это, шлюз сам создает набор IP датаграмм, помещая в каждую по одному фрагменты. В дальнейшем фрагменты могут быть снова разбиты следующими шлюзами на еще более мелкие части. Формат фрагмента IP датаграммы спроектирован так, чтобы адресат — модуль IP смог собрать фрагменты снова в исходные IP датаграммы.

IP модуль, являющийся адресатом, выделяет сегмент из датаграммы, предварительно собрав ее в случае необходимости, и затем передает его по назначению на программу протокола TCP.

Обеспечение надежной доставки данных

Поток данных, посылаемый на TCP соединение, принимается получателем гарантированно и в соответствующей очередности.

Гарантированная передача осуществляется благодаря использованию подтверждений и номеров очереди. Концептуально каждому октету данных присваивается номер очереди. Номер очереди для первого октета данных в сегменте передается вместе с этим сегментом и называется номером очереди для сегмента. Сегменты также несут номер подтверждения, который является номером для следующего ожидаемого октета данных, передаваемого в обратном направлении. Когда протокол TCP передает сегмент с данными, он помещает его копию в очередь повторной передачи и запускает таймер. Когда приходит подтверждение для этих данных, соответствующий сегмент удаляется из

очереди. Если подтверждение не приходит до истечения срока, то сегмент посылается повторно.

Подтверждение протокола TCP не гарантирует, что данные достигли конечного получателя, оно гарантирует лишь, что программа протокола TCP на узле-получателе берет на себя ответственность за это.

Для направления потока данных между программами протоколов TCP используется механизм управления потоками. Получающая программа протокола TCP сообщает «окно» посылающей программе. Данное окно указывает количество октетов, начиная с номера подтверждения, которое принимающая программа TCP готова в настоящий момент принять.

Установка соединения и его отмена

Чтобы идентифицировать отдельные потоки данных, поддерживаемые протоколом TCP, последний определяет идентификаторы портов. Поскольку идентификаторы портов выбираются каждой программой протокола TCP независимо, то они не уникальны. Чтобы обеспечить уникальность адресов для каждой программы протокола TCP, объединяют идентифицирующий эту программу IP адрес и идентификатор порта. В результате получается сокет, который будет уникален во всех локальных сетях, объединенных в единое целое.

Соединение полностью определяется парой сокетов на своих концах. Отдельный сокет может принимать участие во многих соединениях с различными удаленными сокетами. Соединение можно использовать для передачи данных в обоих направлениях, иными словами, оно является полнодуплексным (full duplex).

Протокол TCP волен произвольным образом связывать порты с процессами. Однако все реализации протокола придерживаются нескольких основополагающих концепций:

- Должны присутствовать общеизвестные порты, которые протокол TCP ассоциирует исключительно с соответствующими им типами обслуживающих процессов
- Процессы могут резервировать порты и могут инициировать соединения только с зарезервированных ими портов

Процедура соединения начинается после того, как для сокета был выполнен запрос на открытие.

Существуют два вида открытия соединения: пассивное и активное.

Запрос на пассивное открытие соединения означает, что процесс ждет получения извне запросов на соединение, вместо того, чтобы пытаться самому установить его. Часто процесс, сделавший запрос на пассивное открытие, будет принимать запросы на соединение от любого другого процесса. В этом случае чужой сокет указывается как состоящий целиком из нулей, что означает неопределенность. Неопределенные чужие сокеты могут присутствовать лишь в командах пассивного открытия.

Сервисный процесс, желающий обслужить другие, неизвестные ему процессы, должен осуществить запрос на пассивное открытие с указанием неопределенного сокета. В этом случае соединение может быть установлено с любым процессом, запросившим соединения с этим локальным сокетом. Именно так работает большинство стандартных TCP сервисов.

Общеизвестные порты представляют собой удобный механизм априорного привязывания адреса сокета к какиму-либо стандартному сервису. Например, процесс «Telnet сервер» жестко связан с портом 23.

Процессы могут осуществлять пассивные открытия соединений и ждать, пока от других процессов придут соответствующие запросы на активное открытие, а протокол TCP проинформирует их об установлении соединения. Два процесса, сделавшие друг другу одновременно запросы на активное открытие, получат корректное соединение.

Гибкость такого подхода становится критичной при поддержке распределенных вычислений, когда компоненты системы взаимодействуют друг с другом асинхронным образом.

Когда осуществляется подбор сокетов для локального запроса пассивного открытия и чужого запроса на активное открытие, то принципиальное значение имеют два случая. В первом случае местное пассивное открытие полностью определяет чужой сокет. При этом подбор должен осуществляться очень аккуратно. Во втором случае во время местного пассивного открытия чужой сокет не указывается. Тогда в принципе может быть установлено соединение с любых чужих сокетов. Во всех остальных случаях подбор сокетов имеет частичные ограничения.

Если на один и тот же местный сокет осуществлено несколько ждущих пассивных запросов на открытие, записанных в TCB, и осуществляется извне активный запрос на открытие, то чужой активный сокет будет связываться с тем блоком TCB, где было указание именно на этот запросивший соединения сокет. И только если такого блока TCB не существует, выбор партнера осуществляется среди блоков TCB с неопределенным чужим сокетом.

Для установки соединения используется процедура трехвариантного подтверждения. Процедура установки соединения использует флаг управления синхронизацией (SYN) и трижды обменивается сообщениями. Эта процедура обычно инициируется программой протокола TCP в ответ на запрос другой программы TCP. Данная процедура также работает, если две программы TCP инициируют ее одновременно. Когда попытка инициализации осуществляется с обоих концов одновременно, каждая программа протокола TCP получает SYN-сегмент, который не несет подтверждения для уже отправленного ею SYN. Конечно, прибытие старых дубликатов SYN-сегмента может произвести создать у получателя ложное впечатление, будто осуществляется одновременное открытие соединения. Корректно обрабатывать такие ситуации помогает использование RST-сегментов.

Соединение инициируется при встрече пришедшего сегмента, несущего флаг синхронизации (SYN), и ждущей его записи в TCB. И сегмент и запись создаются пришедшими от пользователей запросами на открытие. Соответствие местного и чужого сокетов устанавливается при инициализации соединения. Соединение признается установленным, когда номера очередей синхронизированы в обоих направлениях между сокетами.

Ниже приведен пример простейшей процедуры трехвариантного подтверждения. Процедура именно такого вида в подавляющем большинстве случаев используется на практике.

Состояние стороны А	TCP сегмент					Состояние стороны В
	Направление пересылки	Номер очереди	Номер подтверждения	Флаги	Содержит ли данные	
CLOSED	Состояние до обмена данными					LISTEN
SYN-SENT	от А к В	100	нет	SYN	нет	SYN-RECEIVED
ESTABLISHED	от В к А	300	101	SYN, ACK	нет	SYN-RECEIVED
ESTABLISHED	от А к В	101	301	ACK	нет	ESTABLISHED
ESTABLISHED	от А к В	101	301	ACK	да	ESTABLISHED

В приведенном примере на строке 2 программа сторона А начинает с посылки сигнала SYN, показывая тем самым, что она будет использовать номера очереди, начиная с номера 100. На строке 3 сторона В посылает сигнал SYN, а также подтверждение о том, что сигнал SYN от стороны А получен. Поле подтверждения информирует о том, что сторона В в данный момент ожидает получение номера 101.

На строке 4 для отправленного стороной В в строке 3 сигнала SYN сторона А дает ответ с помощью пустого сегмента, содержащего сигнал ACK. В строке 5 сторона А передает по сети уже некую порцию данных. Заметим, что сегмент в строке 5 имеет тот же номер очереди, что был у сегмента в строке 4, поскольку сигнал ACK в очереди места

не занимает, если бы это было не так, необходимо бы было подтверждение ACK для самого подтверждения.

В тех случаях, когда при установке соединения некорректная передача сегментов по сети приводит к получению одной из сторон сегмента с флагом, не соответствующим ее текущему состоянию, эта сторона посытает RST-сегмент, что приводит к переходу второй стороны в состояние LISTEN или CLOSED.

Отмена соединения также включает обмен сегментами, несущими на этот раз управляющий флаг FIN.

Закрытие соединения

Состояние CLOSED означает «я не имею данных для передачи». Конечно, закрытие полнодуплексного соединения является предметом множества интерпретаций, поскольку не очевидно, как интерпретировать в соединении сторону, получающую информацию. В случае же TCP клиент, находящийся в состоянии CLOSED, еще может получать информацию до тех пор, пока другая сторона также не сообщит, что переходит в состояние CLOSED. Каждая сторона гарантированно получит все буферы с информацией, отправленные до того, как соединение было закрыто. Поэтому клиенту, не ждущему информации с соединения, следует лишь ждать сообщения об успешном закрытии этого соединения, что означает, что все данные получены стороной, принимающей информацию. Клиенты должны сохранять уже закрытые ими для чтения информации соединения до тех пор, пока программа протокола TCP не сообщит им, что такой информации больше нет.

Существует три основных сценария закрытия соединения:

1. Местный клиент инициирует закрытие соединения

В этом случае создается FIN-сегмент и помещается в очередь сегментов, ждущих отправления. После этого программа TCP уже не будет принимать от этого клиента каких-либо команд на отправку данных по закрытому соединению, а сама переходит в состояние FIN-WAIT-1. Тем не менее, в этом состоянии еще возможно получение клиентом данных с этого соединения. Все сегменты, стоящие в очереди, и сам сегмент с сигналом FIN будут в случае необходимости посылаться другой стороне вновь и вновь, пока не получат своего подтверждения.

Когда программа TCP партнера подтвердит получение сигнала FIN, и сама отправит в ответ свой сигнал FIN, местная программа может подтвердить получение последнего. Программа TCP, получающая сигнал FIN, будет подтверждать его, но не будет посылать своего собственного сигнала FIN до тех пор, пока ее клиент тоже не закроет соединения.

2. От удаленной стороны приходит FIN-сегмент

Когда от удаленной стороны приходит FIN-сегмент, принимающая его программа TCP может подтвердить получение такого сигнала и оповестить своего клиента о том, что соединение закрыто. Клиент ответит командой CLOSE, по факту которой программа TCP может после пересылки оставшихся данных послать удаленной стороне свой FIN-сегмент. После этого программа TCP ждет, пока не придет подтверждение на отправленный ею сигнал FIN, после чего она ликвидирует соединение. Если подтверждения не было по истечении отведенного времени, то соединение ликвидируется в принудительном порядке, о чем сообщается клиенту.

3. Обе стороны закрывают соединение одновременно

Одновременное закрытие соединения клиентами на обоих концах приводит к обмену FIN-сегментами. Когда все сегменты, стоящие в очереди перед сегментом с FIN, будут переданы и получат подтверждение, каждая сторона может послать подтверждение на полученный ею сигнал FIN. Обе программы по получении этих подтверждений ликвидируют соединение.

Номер очереди

Основополагающей идеей в проектировании протокола является то, что каждый октет данных, посылаемый на TCP соединение, имеет номер очереди. Поскольку каждый октет пронумерован, то каждый из них может быть опознан. Приемлемый механизм опознания является накопительным, так что опознание номера X означает, что все октеты с предыдущими номерами уже получены. Этот механизм позволяет регистрировать появление дубликатов в условиях повторной передачи. Нумерация октетов в пределах сегмента осуществляется так, чтобы первый октет данных сразу вслед за заголовком имел наименьший номер, а следующие за ним октеты имели номера по возрастающей.

Протокол TCP должен осуществлять следующие типы сравнения для номеров очереди:

- Является ли номер в подтверждении номером очереди для октетов, уже отправленных, но еще не получивших подтверждения
- Получили ли все октеты в сегменте подтверждение своих номеров (т.е. следует ли удалить данный сегмент из очереди на повторную посылку)
- Содержит ли пришедший сегмент ожидаемые нами номера (т.е. «перекрывает» ли этот сегмент окно получателя)

Во время установления или инициализации какого-либо соединения обе программы протокола TCP должны синхронизировать друг с другом первоначальные номера очередей. Это осуществляется посредством обмена сегментами, устанавливающими соединения, несущими контрольный бит SYN (for synchronize — для синхронизации), несущими исходные номера для очередей. Для краткости, сегменты, несущие бит SYN, также называются SYN сегментами. Следовательно, решение проблемы требует приемлемого механизма для подбора первоначального номера очереди и немногочисленных сигналов подтверждения при обмене номерами ISN.

Синхронизация требует, чтобы каждая сторона, участвующая в соединении, посылала свой собственный первоначальный номер очереди, а также получала подтверждение на это от напарника. Каждая сторона должна также получить первоначальный номер очереди от напарника и послать подтверждение.

- сигнал SYN от A к B: мой номер очереди X
- сигнал ACK от B к A: ваш номер очереди X
- сигнал SYN от B к A: мой номер очереди Y
- сигнал ACK от A к B: ваш номер очереди Y

Поскольку шаги 2 и 3 можно объединить в одно сообщение, такая процедура называется трехвариантным подтверждением.

Обмен данными

Набор данных, передаваемых по соединению, можно рассматривать как поток октетов. Пользователь, отправляющий данные, указывает при запросе на посылку, следует ли данные, отправляемые при этом запросе, немедленно доставлять получателю немедленно. Указание осуществляется установкой флага PUSH (проталкивание).

Реализация TCP может собирать данные, принимаемые от пользователя, а затем передавать их в сеть по своему усмотрению в виде сегментов. Если же выставлен запрос на немедленную доставку, то протокол должен передать все не отправленные ранее данные. Когда программа протокола TCP, принимающая данные, сталкивается с флагом немедленной доставки, ей не следует ожидать получения новых данных по сети до тех пор, пока уже имеющиеся данные не будут переданы ждущему их местному процессу.

Нет нужды привязывать функции немедленной доставки к границам сегмента. Данные, содержащиеся в каком-либо сегменте, могут быть результатом одного или нескольких запросов на посылку. Или же один запрос может породить несколько сегментов.

Существует связь между функцией немедленной доставки и использованием буферов данных в конкретной реализации TCP. Каждый раз, когда в буфер получателя приходят данные с флагом PUSH, содержимое этого буфера передается пользователю на обработку, даже если буфер и не был заполнен. Если приходящие данные заполняют буфер пользователя до того, как получена команда срочной доставки, пользователю отправляется блок данных, соответствующий размеру буфера. Протокол TCP имеет также средства для сообщения получателю, что с некоторого момента он имеет дело со срочными данными. Протокол TCP не пытается определить, что именно пользователь делает со ждущими обработки срочными данными. Однако обычно предполагается, что получающий данные процесс будет предпринимать усилия для быстрой обработки таких данных.

После того, как соединение было установлено, передача данных осуществляется с помощью обмена сегментами. Так как сегменты могут быть потеряны в результате ошибок или перегрузки сети, стороны использует механизм повторной посылки по истечении определенного времени с тем, чтобы убедиться в получении каждого сегмента.

Отправитель данных отслеживает следующий номер в очереди, подлежащий отправке. Получатель данных отслеживает следующий номер, прибытие которого он ожидает, а так же значение самого старого номера, который был отправлен, но еще не получил подтверждения. Когда поток данных иссякает, а все отправленные данные получают подтверждение, эти переменные имеют одинаковое значение.

Поскольку соединения находятся в состоянии ESTABLISHED, все сегменты, в дополнение к данным, несут информацию о подтверждении ранее отправленных сегментов.

Передача срочной информации

Механизм срочной передачи протокола TCP предназначен для того, чтобы клиент, отправляющий данные, мог побудить получателя принять некую срочную информацию, а также позволить программе TCP, принимающей данные, информировать своего клиента, когда вся имеющаяся на настоящий момент информация будет получена.

Данный механизм позволяет помечать некую точку в потоке данных как конец блока срочной информации. Когда в программе TCP, принимающей данные, данная точка окажется впереди индикатора номера в очереди получения, эта программа TCP должна дать команду своему клиенту перейти в «срочный режим». Когда номер в очереди получения догонит срочный указатель в потоке данных, программа TCP должна дать команду клиенту прийти в «нормальный режим». Если срочный указатель сменит свое положение, когда клиент находится в «срочном режиме», последний не узнает об этом.

Данный метод использует поле флага срочности, который присутствует во всех передаваемых сегментах. Единица в поле контрольного флага URG означает, что задействовано поле срочности. Чтобы получить указатель этого поля в потоке данных, необходимо дополнить его номером рассматриваемого сегмента в очереди. Отсутствие флага URG означает отсутствие у отправителя не посланных срочных данных.

При указании срочности клиент должен также послать по крайней мере один октет данных. Если клиент, помещающий данные, дополнительно закажет функцию проталкивания (флаг PSH), то передача срочной информации ждущему ее процессу многократно ускорится.

Управление окном

Окно, посылаемое с каждым сегментом, указывает диапазон номеров очереди, которые отправитель окна, он же получатель данных, готов принять в настоящее время. Предполагается, что такой механизм связан с наличием в данный момент места в буфере данных.

Указание окна большого размера стимулирует передачу. Но если пришло большее количество данных, чем может быть принято программой TCP, данные будут отброшены. Это приведет к излишним пересылкам информации и ненужному увеличению нагрузки на сеть. Указание окна малого размера может ограничить передачу данных скоростью, которая определяется временем путешествия по сети после каждого посылаемого сегмента.

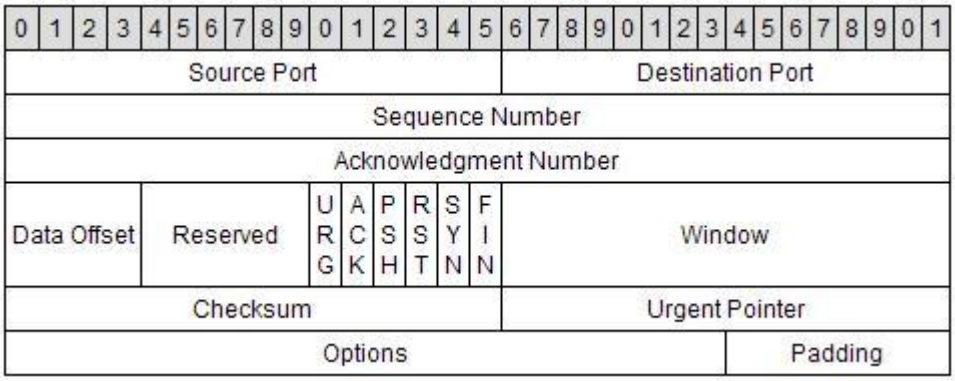
TCB — блок управления передачей (Transmission Control Block).

Термин TCB веден в спецификации TCP и представляет собой структуру, содержащую следующие данные:

- идентификаторы местного и удаленного сокетов
- флаги безопасности и приоритета для данного соединения
- указатели буферов посылки и получения
- указатели текущего сегмента и очереди повторной посылки
- несколько дополнительных переменных, имеющих отношение к номерам очередей отправителя и получателя

Формат заголовка TCP сегмента

Передача TCP сегментов осуществляется в виде IP датаграмм. Заголовок датаграммы в IP имеет несколько информационных полей, включая адреса отправляющего и принимающего узлов. Заголовок TCP следует за IP заголовком и дополняет его информацией, специфичной для TCP. Такое деление допускает использование на уровне узлов протоколов, иных нежели TCP.



1. Source Port (порт отправителя) 16 бит.

2. Destination Port (порт получателя) 16 бит.

3. Sequence Number (номер очереди) 32 бита.

Номер очереди для первого октета данных в данном сегменте (за исключением тех случаев, когда присутствует флаг синхронизации SYN). Если же флаг SYN присутствует, то номер очереди является инициализационным (ISN), а номер первого октета данных — ISN + 1.

4. Acknowledgment Number (номер подтверждения) 32 бита.

Если установлен контрольный бит ACK, то это поле содержит следующий номер очереди, который отправитель данной датаграммы желает получить в обратном направлении. Номера подтверждения посылаются постоянно, как только соединение будет установлено.

5. Data Offset (смещение данных) 4 бита.

Количество 32-битных слов в TCP заголовке. Указывает на начало поля данных. TCP заголовок всегда кончается на 32-битной границе слова, даже если он содержит опции.

6. Reserved (зарезервировано) 6 бит.

Это резервное поле, должно быть заполнено нулями.

7. Control Bits (контрольные биты) 6 бит.

Биты этого поля слева направо:

- URG: поле срочного указателя задействовано
- ACK: поле подтверждения задействовано
- PSH: функция проталкивания
- RST: перезагрузка данного соединения
- SYN: синхронизация номеров очереди
- FIN: нет больше данных для передачи

8. Window (окно) 16 бит.

Количество октетов данных, начиная с октета, чей номер указан в поле подтверждения. Количество октетов, получения которых ждет отправитель настоящего сегмента.

9. Checksum (контрольная сумма) 16 бит.

Контрольная сумма, помимо всего прочего, учитывает 96 бит псевдозаголовка, который для внутреннего употребления ставится перед TCP заголовком. Этот псевдозаголовок содержит адрес отправителя, адрес получателя, протокол и длину TCP сегмента. Такой подход обеспечивает защиту протокола TCP от ошибшихся в маршруте сегментов. Эту информацию обрабатывает IP.

10. Urgent Pointer (срочный указатель) 16 бит.

Это поле сообщает текущее значение срочного указателя. Последний является положительной величиной — смещением относительно номера очереди данного сегмента. Срочный указатель сообщает номер очереди для октета, следующего за срочными данными. Это поле интерпретируется только в том случае, когда в сегменте выставлен контрольный бит URG.

11. Options (опции) поле переменной длины.

Опции могут располагаться в конце TCP заголовка, а их длина кратна 8 бит. Все опции учитываются при расчете контрольной суммы.

Битовая последовательность 00000000 определяет окончание поля опций. Окончание поля опций может не совпадать с концом TCP заголовка, указанным в поле Data Offset. В этом случае, оставшееся место в заголовке должно быть заполнено нулями.

Порядок выполнения лабораторной работы

Задание 1.

1. На основе отчета об обмене узлов датаграммами, полученного в предыдущей работе, произвести анализ первого TCP пакета.
2. Произвести краткий анализ 10 TCP сегментов с тем, чтобы восстановить ход TCP сессии.
3. Разбить сессию на три логические части: установка соединения, обмен данными, завершение соединения.
4. Составьте три группы из датаграмм:

- Датаграммы, использовавшиеся при установке соединения
- Датаграммы с данными
- Датаграммы, участвовавшие в процедуре завершения соединения

5. Заполнить бланк аналитического отчета.

Задание 2.

1. Запустите анализатор и подключитесь к серверу, согласно варианту заданий.
2. Перехватите и сохраните в файл 5 датаграмм. Выдели из IP-датаграмм TCP сегменты.
3. Проанализируйте TCP сегмент. Необходимо вывести в отчет информацию о сегменте TCP:
    - заголовок TCP в шестнадцатеричном виде;

- порты отправителя и получателя;
- размер заголовка и размер данных в сегменте (TCP payload);
- значения поля порядкового номера (sequence number, seq) и номера подтверждения (acknowledgement number, ack);
- если пакет содержит пользовательские данные, то необходимо вывести их в шестнадцатеричном виде.

4. Согласно описанию формата заголовка TCP сегмента заполните таблицу в бланке отчета. Обратите внимание, что наличие или отсутствие поля «Опции» в TCP заголовке, а так же его длину, можно выяснить, лишь проанализировав значения других полей.

Задание 3.

1. Запустите анализатор и подключитесь к серверу, согласно варианту заданий.
2. Перехватите и сохраните в файл 5 UDP сегментов при обращении к сайту согласно варианту заданий.
5. Проанализируйте UDP сегменты. Проанализируйте сегмент. Необходимо вывести в отчет информацию о сегменте:
   - заголовок в шестнадцатеричном виде;
   - порты отправителя и получателя;
   - другие поля в сегменте;
   - если пакет содержит пользовательские данные, то необходимо вывести их в шестнадцатеричном виде.
3.
4. Согласно описанию формата заголовка UDP сегмента заполните таблицу в бланке отчета.

Пояснения для заполнения таблиц бланка отчета

1. Стороны, между которыми происходил обмен данными, должны быть указаны в виде X.X.X.X:Y, где X.X.X.X - IP-адрес, а Y - номер TCP-порта. То есть в данном случае сторона соответствует сокету.
2. Каждый TCP сегмент, полученный из отчета об обмене узлов IP-датаграммами должен быть представлен строкой в одной из трех таблиц.
3. Таблицы заполняются данными в хронологическом порядке.
4. Под состоянием стороны понимается состояние возникающее сразу после получения и обработки сегмента, либо после отправки TCP сегмента и истечения отведенного времени на его получение стороной-получателем.
5. В некоторых случаях возможна ситуация, при которой последний пакет в таблице будет одновременно первым в следующей.
6. Последняя запись в первой таблице должна соответствовать датаграмме, после которой на обеих сторонах устанавливается состояние ESTABLISHED.
7. Поле «Заголовок» должно содержать TCP заголовок датаграммы в виде октетов.
8. В поле «Пояснения» должно быть указано, с какой целью сторона-отправитель послала данный сегмент. Так же должно быть дано объяснение всем флагам, которые в нем используются. Для первой и последней таблиц в этом поле нужно указать, содержит ли сегмент данные.

Защита работа

Для защиты данной работы, необходимо:

1. Предоставить отчеты об обмене узлов TCP и UDP сегментами с распечатками полученных текстовых файлов и пояснениями, что в них содержится.

2. Предоставить заполненные бланки отчета с пояснениями по полям заголовка TCP и UDP сегментов в таблице.

3. Продемонстрировать результат преподавателю и ответить на контрольные вопросы.

Содержание отчета

1. Протоколы выполнения работы в виде сохраненных файлов захвата пакетов с текстовым пояснением их содержания.
2. Распечатки скриншотов с результатами выполнения заданий работы.
3. Результаты анализа сегментов, описание служб, протоколов, формата сегмента.