Известия Гомельского государственного университета
имени Ф. Скорины, № 6(81), 2013

УДК 004.052

Автоматизация измерения пропускной способности ЛВС методом пакетной пары

В.Н. Кулинченко

В статье рассмотрены особенности реализации метода пакетной пары для определения пропускной способности ЛВС. Являясь одним из эффективных методов зондирования, он позволяет определить пропускную способность сети.
Ключевые слова: метод пакетной пары, пропускная способность, протоколы передачи, сегмент ЛВС, каналы передачи данных.

The article describes the features of the algorithm implementation methods for probing packet segments of a LAN. Software subsystem for determination the capacity of the network was created. The created software subsystem allows determining the network bandwidth.
Keywords: packet sensing, bandwidth, protocols, LAN segments, data channels.

При настройке сети необходимо различать номинальную и эффективную пропускные способности протокола. Под номинальной пропускной способностью обычно понимается битовая скорость передачи данных, поддерживаемая на интервале передачи одного пакета. Эффективная пропускная способность протокола – это средняя скорость передачи пользовательских данных, то есть данных, содержащихся в поле данных каждого пакета. В общем случае эффективная пропускная способность протокола будет ниже номинальной из-за наличия в пакете служебной информации, а также из-за пауз между передачей отдельных пакетов.

В сущности, пропускная способность представляет собой качество соединения, и очевиден факт, что чем лучше качество соединения, тем более высокая будет производительность.

Пропускная способность глобальной сети постоянно варьируется, и измерить ее точно довольно сложно. Это, хотя и в меньшей степени, характерно также для локальных сетей. Измерение пропускной способности аппаратно-программного обеспечения сетей может осуществляться с использованием протоколов ICMP, SNMP, UDP, TCP.

При измерении пропускной способности с помощью протокола ICMP необходимо учитывать время, требуемое для обработки его на удаленном компьютере, поскольку пропускная способность существенно зависит от размера пакета и максимальное значение достигается на пакетах размером 65535 байт. Здесь также сказываются затраты на переключения контекста, поскольку фрагментация происходит на уровне IP, работающем в режиме ядра. При измерении пропускной способности на базе TCP/IP необходимо учитывать гораздо больше факторов, нежели при использовании других протоколов.

Между операциями записи и посылаемыми TCP сегментами нет взаимнооднозначного соответствия. Как именно соотносятся обращения к операции записи с протоколом TCP, зависит от системы, но все же спецификации протокола достаточно определенны, и можно сделать некоторые выводы при знакомстве с конкретной реализацией: по отношению к TCP-соединениям операцию записи лучше представлять себе как копирование в очередь для передачи, сопровождаемое извещением TCP о появлении новых данных. Понятно, какую работу TCP произведет дальше, но эти действия будут асинхронны по отношению к самой записи.

Как отмечалось выше, операция записи отвечает лишь за копирование данных из буфера приложения в память ядра и за уведомление TCP о том, что появились новые данные для передачи.

Прежде всего очень важно не допускать перегрузки сети. Если TCP неожиданно пошлет в сеть большое число сегментов, может исчерпаться память маршрутизатора, что повлечет за собой отбрасывание датаграмм. Это может стать причиной повторных передач,

что способно еще больше загрузить сеть. В худшем случае она будет загружена настолько, что датаграммы вообще нельзя будет доставить. Это называется затором (congestion collapse). Чтобы избежать перегрузки, TCP не посылает по простаивающему соединению все сегменты сразу. Сначала он посылает один сегмент и постепенно увеличивает число неподтвержденных сегментов в сети, пока не будет достигнуто равновесие.

Для предотвращения перегрузки TCP применяет два алгоритма, в которых используется еще одно окно, называемое окном перегрузки. Максимальное число байтов, которое TCP может послать в любой момент, – это минимальная из двух величин: размер окна передачи и размер окна перегрузки. Эти окна отвечают за разные аспекты управления потоком. Окно передачи, декларируемое TCP на другом конце, предохраняет от переполнения его буферов. Окно перегрузки, отслеживаемое TCP на вашем конце, не дает превысить пропускную способность сети. Ограничив объем передачи минимальным из этих двух окон, вы удовлетворяете обоим требованиям управления потоком.

Еще один фактор, влияющий на стратегию отправки TCP, – алгоритм Нейгла. Этот алгоритм впервые предложен в RFC 896. Он требует, чтобы никогда не было более одного неподтвержденного маленького сегмента, то есть сегмента размером менее MSS. Цель алгоритма Нейгла – не дать TCP переполнить сеть последовательностью мелких сегментов. Вместо этого TCP сохраняет в своих буферах небольшие блоки данных, пока не получит подтверждение на предыдущий маленький сегмент, после чего посылает сразу все накопившиеся данные.

Алгоритм Нейгла – это один из двух алгоритмов, позволяющих избежать т.н. синдрома безумного окна (SWS – silly window syndrome). Смысл этой тактики в том, чтобы не допустить отправки небольших объемов данных. Синдром SWS и его отрицательное влияние на производительность обсуждаются в RFC 813. Алгоритм Нейгла пытается избежать синдрома SWS со стороны отправителя. Но требуются и усилия со стороны получателя, который не должен декларировать слишком маленькие окна.

На основе этой информации можно сформулировать стратегию отправки, принятую в реализациях TCP, производных от BSD. В других реализациях стратегия может быть несколько иной, но основные принципы сохраняются.

При каждом вызове процедуры вывода TCP вычисляет объем данных, которые можно послать. Это минимальное значение количества данных в буфере передачи, размера окон передачи и перегрузки и MSS. Данные отправляются при выполнении хотя бы одного из следующих условий:

– можно послать полный сегмент размером MSS;
– соединение простаивает, и можно опустошить буфер передачи;
– алгоритм Нейгла отключен, и можно опустошить буфер передачи;
– есть срочные данные для отправки;
– есть маленький сегмент, но его отправка уже задержана на достаточно длительное время;
– окно приема, объявленное хостом на другом конце, открыто не менее чем наполовину;
– необходимо повторно передать сегмент;
– требуется послать АСК на принятые данные;
– нужно объявить об обновлении окна.

Вот почему важно при использовании протокола TCP/IP считать именно полученные данные, нежели отправленные.

TCP – это сложный протокол, обеспечивающий базовую службу доставки IP-датаграмм надежностью и управлением потоком. В то же время UDP добавляет лишь контрольную сумму. Поэтому может показаться, что UDP должен быть на порядок быстрее TCP. Исходя из этого предположения, многие программисты полагают, что с помощью UDP можно достичь максимальной производительности.

Действительно, есть ситуации, когда UDP существенно быстрее TCP. Но иногда использование TCP оказывается эффективнее, чем применение UDP.

В сетевом программировании производительность любого протокола зависит от сети, приложения, нагрузки и других факторов, из которых не последнюю роль играет качество реализации. Единственный надежный способ узнать, какой протокол и алгоритм работают оптимально, – это протестировать их в предполагаемых условиях работы приложения. На практике это, конечно, не всегда осуществимо, но часто удается получить представление о производительности.

В типичной ситуации большая часть времени процессора в обоих протоколах тратится на копирование данных и вычисление контрольных сумм, поэтому здесь нет большой разницы.

Нужно отметить, что для обеспечения надежности TCP должен посылать подтверждения (АСК-сегменты) на каждый принятый пакет. Это несколько увеличивает объем обработки на обоих концах. Во-первых, принимающая сторона может включить АСК в состав данных, которые она посылает отправителю. В действительности во многих реализациях TCP отправка АСК задерживается на несколько миллисекунд: предполагается, что приложение-получатель вскоре сгенерирует ответ на пришедший сегмент. Во-вторых, TCP не обязан подтверждать каждый сегмент. В большинстве случаев реализации при нормальных условиях АСК посылается только на каждый второй сегмент.

Еще одно принципиальное отличие между TCP и UDP в том, что TCP требует логического соединения, и, значит, необходимо заботиться об его установлении и разрыве. Для установления соединения обычно требуется обменяться тремя сегментами. Для разрыва соединения нужно четыре сегмента, которые, кроме последнего, часто можно скомбинировать с сегментами, содержащими данные.

При длительном соединении между клиентом и сервером (например, клиент и сервер обмениваются большим объемом данных) период логического соединения «размазывается» между всеми передачами данных, так что существенного влияния на производительность это не оказывает. Однако если речь идет о простой транзакции, в течение которой клиент посылает запрос, получает ответ и разрывает соединение, то время инициализации составляет заметную часть от времени всей транзакции. Таким образом, следует ожидать, что UDP намного превосходит TCP по производительности именно тогда, когда приложение организует короткие сеансы связи. И, наоборот, TCP работает быстрее, когда соединение поддерживается в течение длительного времени при передаче больших объемов данных.

Необходимо отметить, что на хосте localhost TCP работает примерно в два раза быстрее, чем UDP. Это связано с тем, что TCP способен объединять несколько блоков по 1440 байт в один сегмент, тогда как UDP посылает отдельно каждую датаграмму длиной 1440 байт.

В локальной сети UDP примерно на 20% быстрее TCP, но потеря датаграмм значительнее. Потери имеют место даже тогда, когда и сервер, и клиент работают на одной машине; связаны они с исчерпанием буферов. Хотя передача 5000 датаграмм на максимально возможной скорости – это скорее отклонение, чем нормальный режим работы, но следует иметь ввиду возможность такого результата. Это означает, что UDP не дает никакой гарантии относительно доставки данной датаграммы, даже если оба приложения работают на одной машине.

Из этих примеров следует важный вывод: нельзя строить априорные предположения о сравнительной производительности TCP и UDP. При изменении условий, даже очень незначительном, показатели производительности могут очень резко поменяться.

Если говорить о практической стороне вопроса, то современные реализации TCP достаточно эффективны. Реально продемонстрировано, что TCP может работать со скоростью аппаратуры на стомегабитных сетях FDDI.

Измерение пропускной способности при помощи протокола UDP аналогично измерению при помощи протокола ICMP, разница заключается в программной реализации. Тогда встает вопрос, какой же протокол наиболее предпочтителен и наиболее точно отражает пропускную способность сети. На данный вопрос нельзя дать однозначный ответ, поскольку условия эксперимента могут сильно меняться. Поэтому в локальной сети желательно использовать для измерения все протоколы.

Знание общей пропускной способности между двумя узлами не может дать полной информации о возможных путях ее повышения, так как из общей цифры нельзя понять, какой из промежуточных этапов обработки пакетов в наибольшей степени тормозит работу сети. Поэтому могут быть полезны данные о пропускной способности отдельных элементов сети для принятия решения о способах ее оптимизации.

Существует несколько методов измерения пропускной способности ЛВС. Одним из достаточно эффективных способов является метод пакетной пары. Он предназначен для определения номинальной пропускной способности канала, то есть максимально возможной пропускной способности незагруженного канала. Он основан на использовании пакетной пары – двух пакетов, выпущенных в канал непосредственно друг за другом. Отправленные таким образом пакеты, могут быть приняты с временным интервалом $\Delta = \frac{S}{C}$, где S – размер пакета, C – номинальная пропускная способность интерфейса, через который пакеты попали в канал. На выходе канала, состоящего из нескольких линий, интервал пакетной пары будет соответствовать самому узкому месту в канале.

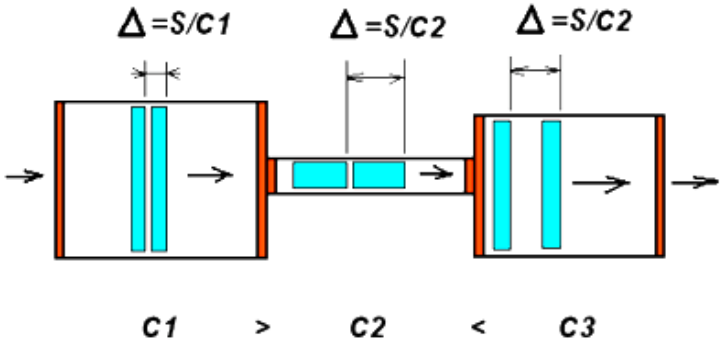

Рисунок 1 – Прохождение пакетной пары по каналу
(ширина каждой линии связи соответствует ее пропускной способности)

При наличии загрузки канала другими потоками данных пакеты, распространяющиеся в канале, начинают взаимодействовать, а точнее перемешиваться с пакетной парой. Какой-нибудь побочный пакет может войти между пакетами нашей пары, и тогда интервал времени увеличится. Если непосредственно перед пакетной парой промежуточный маршрутизатор примет большой побочный пакет, то интервал времени пакетной пары может уменьшиться, так как первый пакет пары не сможет отправиться, а будет стоять в очереди, ожидая передачи побочного пакета, и таким образом, дождется окончания приема второго пакета.

Литература

1. Семенов, Ю.А. Телекоммуникационные технологии. Учебное пособие / Ю.А. Семенов. – М. : ГНЦ ИТЭФ, 2010. – 600 с.

2. Закер, К. Компьютерные сети. Модернизация и поиск неисправностей: Пер. с англ. / К. Закер. – СПб. : БХВ-Петербург, 2004. – 1008 с.

3. Таненбаум, Э. Компьютерные сети / Э. Таненбаум. – СПб. : Питер, 2008. – 960 с.

4. Олифер, Н. Компьютерные сети. Принципы, технологии, протоколы / Н. Олифер, В. Олифер. – СПб. : Питер, 2011. – 944 с.

5. Бертсекас, Д. Сети передачи данных / Д. Бертсекас, Р. Галлагер. – М. : Мир, 1989. – 544 с.

6. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер. – М. : Вильямс, 2007 – 544 с.

Гомельский государственный
университет им. Ф. Скорины

Поступила в редакцию 31.10.2013