Network Working Group
Request for Comments: 2001
Category: Standards Track

W. Stevens
NOAO

January 1997

Алгоритмы Slow Start, Congestion Avoidance, Fast Retransmit и Fast Recovery для протокола TCP

TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms

Статус документа

Этот документ содержит спецификацию стандарта, предложенного сообществу Internet, и служит приглашением к дискуссии по затронутым вопросам. Текущее состояние стандартизации[1] для данной спецификации вы можете найти в документе «Internet Official Protocol Standards» (STD 1). Данная спецификация может распространяться без ограничений.

Тезисы

Современные реализации протокола TCP используют 4 алгоритма, которые не описаны ни в одном из стандартов Internet: **slow start**[2], **congestion avoidance**[3], **fast retransmit**[4] и **fast recovery**[5]. В работах [2] и [3] описаны некоторые детали этих алгоритмов, в работе [4] приводятся примеры использования, а работа [5] содержит исходные тексты реализации алгоритмов для 4.4BSD. Стандарт RFC 1122 требует от реализаций протокола TCP поддержки алгоритмов **slow start** и **congestion avoidance** (параграф 4.2.2.15 [1]), ссылаясь на работу [2]. Однако алгоритмы **fast retransmit** и **fast recovery** были реализованы после выхода RFC 1122. Целью данного документа является описание всех 4 алгоритмов для Internet.

Благодарности

Значительная часть этого документа заимствована из книг «TCP/IP Illustrated, Volume 1: The Protocols» (Addison-Wesley, 1994) и «TCP/IP Illustrated, Volume 2: The Implementation» (Addison-Wesley, 1995), написанных Гэри Райтом (Gary R. Wright) и Ричардом Стивенсом (W. Richard Stevens). Материалы использованы с разрешения издательства Addison-Wesley. Описываемые алгоритмы были разработаны Ван Якобсоном (Van Jacobson).

Замедленный старт

Старые реализации TCP начинают соединение с передачи отправителем в сеть множества сегментов (вплоть до размера окна, анонсируемого принимающей стороной). Такое поведение нормально для случаев, когда оба хоста находятся в одной ЛВС, а при наличии между ними маршрутизаторов и медленных каналов могут возникать проблемы. Некоторые промежуточные маршрутизаторы вынуждены буферизовать пакеты и в таких случаях может возникнуть нехватка буферного пространства. В работе [2] показано, как это может снизить пропускную способность соединений TCP.

Алгоритм предотвращения таких ситуаций носит название **slow start**. Работа алгоритма основана на опытных данных о том, что скорость передачи пакетов в сеть должна совпадать со скоростью возврата подтверждений удаленной стороной.

Алгоритм **slow start** добавляет на стороне отправителя TCP дополнительное окно - окно насыщения, обозначаемое термином **cwnd**. При организации нового соединения с хостом другой сети окно насыщения инициализируется с размером в один сегмент[6]. Каждый раз при получении подтверждения ACK окно насыщения увеличивается на размер одного сегмента. Отправитель может передать в сеть количество пакетов, ограниченное меньшим из двух окон - окно насыщения и анонсируемое[7] окно. Таким образом обеспечивается управление потоком передаваемых данных - окно насыщения управляет со стороны получателя, анонсируемое окно - со стороны отправителя. Первое основано на возможностях получателя и пропускной способности сети, а второе связано с доступным получателю размером буферного пространства.

Отправитель начинает с передачи одного сегмента и ждет подтверждения ACK. При получении пакета ACK размер окна насыщения увеличивается на 1 и могут быть переданы два сегмента. Когда каждый из этих сегментов будет подтвержден, размер окна насыщения будет увеличен до 4. В идеальном случае обеспечивается экспоненциальный рост размера окна, однако задержка передачи пакетов ACK получателем снижает скорость роста размера окна. Обычно получатель передает одно подтверждение ACK для каждой пары принятых сегментов.

---

[1]Этот документ признан устаревшим и заменен RFC 2581, который, в свою очередь, был заменен RFC 5681. Переводы этих документов доступны на сайте www.protocols.ru. *Прим. перев.*
[2]Замедленный старт.
[3]Предотвращение насыщения.
[4]Ускоренный повтор передачи.
[5]Ускоренное восстановление.
[6]Т. е., окно имеет размер сегмента, анонсируемый удаленной стороной, или используемое по умолчанию значение размера сегмента, которое обычно составляет 536 или 512 октетов.
[7]Получателем. *Прим. перев.*

В какой-то момент будет достигнуто насыщение канала между хостами и промежуточный маршрутизатор начнет отбрасывать пакеты. Это скажет отправителю о том, что размер окна насыщения слишком велик.

Ранние реализации использовали алгоритм **slow start** только при передаче сегментов между сетями. Современные реализации протокола используют slow start во всех случаях.

Предотвращение перегрузки

Насыщение может возникать в тех случаях, когда данные поступают из «толстой трубы» (например, скоростная ЛВС) и должны передаваться в узкополосный канал (медленная сеть WAN). Может возникать насыщение и в тех случаях, когда множество потоков приходят на маршрутизатор одновременно и производительности этого маршрутизатора не хватает для обслуживания всех потоков. Алгоритм предотвращения перегрузки (Congestion avoidance) используется при потере пакетов. Описание алгоритма приведено в работе [2].

Алгоритм основан на допущении, что потери пакетов, связанные с их повреждением при передаче, весьма малы (менее 1%). Следовательно, потери пакетов сигнализируют о насыщении где-то на пути между отправителем и получателем. Существует два индикатора потери пакетов - возникновение тайм-аутов и появление дубликатов ACK.

Алгоритмы **congestion avoidance** и **slow start** независимы один от другого и имеют разные цели. Однако при возникновении насыщения протокол TCP должен снизить скорость передачи пакетов в сеть и даже подключить алгоритм **slow start** для снижения размера окна насыщения. На практике эти алгоритмы обычно используются совместно.

Алгоритмы **congestion avoidance** и **slow start** требуют поддержки для каждого соединения двух переменных - **cwnd** (размер окна насыщения) и **ssthresh** (порог slow start). Комбинация этих алгоритмов работает следующим образом:

1) При инициализации соединения задается **cwnd = 1** (сегмент) и **ssthresh = 65535** (байтов).

2) Модуль передачи протокола TCP никогда не передает в сеть число сегментов, превышающее минимальное из двух значений - **cwnd** и анонсируемый получателем размер окна.

3) При возникновении насыщения (тайм-аут или появление дубликатов ACK) для переменной **ssthresh** устанавливается значение, соответствующее половине размера меньшего из окон (**cwnd** и анонсируемый получателем размер окна), но не менее размера одного сегмента. Кроме того, при наличии тайм-аутов для переменной **cwnd** устанавливается значение 1 (т. е., повторяется процедура **slow start**).

4) При получении подтверждений от удаленной стороны значение **cwnd** увеличивается. Способ увеличения зависит от того, какой алгоритм будет использовать модуль TCP - **slow start** или **congestion avoidance**.

   Если **cwnd** <= **ssthresh**, TCP будет использовать алгоритм **slow start**, в остальных случаях - **congestion avoidance**. Процедура **slow start** продолжается, пока TCP находится в первой половине пути к насыщению (поскольку, на этапе 3 был установлен порог в половину размера окна, при котором возникло насыщение), а дальше начинает использоваться алгоритм предотвращения насыщения.

   Алгоритм **slow start** использует начальное значение **cwnd = 1** и увеличивает размер окна насыщения на 1 при получении каждого сегмента ACK. Как было отмечено выше, это ведет к экспоненциальному росту размера окна - передается сначала 1 сегмент, затем 2, 4 и т. д. Алгоритм **congestion avoidance** задает для окна cwnd увеличение на **segsize*segsize/cwnd** при получении каждого сегмента ACK (**segsize** - размер сегмента, а значение **cwnd** пересчитано в байты).

   Такая процедура обеспечивает линейный рост **cwnd** в отличие от экспоненциального роста в **slow start**. Размер **cwnd** должен увеличиваться хотя бы на 1 сегмент в течение каждого периода кругового обхода RTT[1], независимо от числа полученных в течение периода RTT подтверждений ACK, тогда как **slow start** увеличивает размер **cwnd** на количество принятых подтверждений.

Многие реализации некорректно используют увеличение размера окна насыщения на часть размера сегмента (обычно, на 1/8) в течение периода предотвращения пергрузки (**congestion avoidance**). Это неправильно и такое поведение не будет эмулироваться в будущих версиях.

Ускоренный повтор передачи

Этот алгоритм является модификацией алгоритма **congestion avoidance**, предложенной в 1990 году [3]. Прежде, чем рассматривать различия этих алгоритмов, отметим, что протокол TCP может генерировать дубликаты ACK при нарушении порядка доставки пакетов (см. параграф 4.2.2.21 документа [1][2]). Эти дубликаты ACK не следует задерживать - цель их состоит в том, чтобы передать другой стороне информацию о нарушении порядка доставки сегментов и сообщить порядковый номер ожидаемого сегмента.

Поскольку TCP не знает о причине появления дубликатов ACK (потеря сегментов или нарушение порядка доставки), протокол вынужден ожидать получения некоторого количества дубликатов ACK. Предполагается, что в случае нарушения порядка доставки будет приходить один или два дубликата ACK до начала обработки полученного с нарушением порядка сегмента, при котором будет генерироваться новый пакет ACK. Получение же подряд трех или более дубликатов ACK с высокой вероятностью говорит о потере сегмента. В этом случае TCP повторяет передачу потерянного сегмента без ожидания таймера повторной передачи.

Ускоренное восстановление

После выполнения ускоренного повтора (fast retransmit), связанного с потерей сегмента, выполняются операции **congestion avoidance**, а не **slow start**. Такая процедура используется, как алгоритм восстановления. Это повышает

[1]Round-trip time.
[2]Экспериментальный алгоритм Fast Retransmit отмечен в этом документе, как одна из причин такого поведения. *Прим. перев.*

эффективность работы протокола, поскольку обеспечивает высокую пропускную способность при условиях незначительного насыщения. Особенно эффективно такое решение при большом размере окон.

Причина отказа от процедуры **slow start** в данном случае связана с тем, что прием дубликатов ACK говорит модулю TCP больше, нежели просто о потере пакета. Поскольку принимающая сторона может генерировать дубликат ACK только после получения другого сегмента, этот сегмент уже покинул сеть и находится в буфере принимающей стороны (т. е., существует поток данных между обеими сторонами соединения и TCP нет смысла уменьшать этот поток с помощью процедуры **slow start**).

Алгоритмы ускоренного повтора и восстановления обычно используются совместно следующим образом:

1) При получении подряд третьего дубликата ACK для переменной **ssthresh** устанавливается значение в половину размера текущего окна насыщения **cwnd**, но не менее двух сегментов и повторяется передача потерянного сегмента. Далее для переменной **cwnd** устанавливается значение **ssthresh** + размер 3 сегментов. Это увеличивает окно насыщения на количество сегментов, которые покинули сеть и находятся в кэше удаленной стороны.

2) При получении каждого следующего дубликата ACK значение **cwnd** увеличивается на размер сегмента. Это увеличивает окно насыщения с учетом дополнительных пакетов, принятых удаленной стороной из сети. Пакет передается в сеть, если новое значение **cwnd** допускает это.

3) При получении пакета ACK, подтверждающего доставку новых данных для переменной **cwnd** устанавливается значение **ssthresh** (с округлением до целого числа). Этот пакет ACK должен быть подтверждением повторной передачи на этапе 1 в течение одного периода RTT. В дополнение к этому данный пакет ACK подтверждает доставку всех промежуточных сегментов, переданных в интервале между потерей пакета и получением первого дубликата ACK. Этот этап является процедурой предотвращения перегрузки, поскольку TCP снижает скорость передачи до половины значения скорости в момент потери пакета.

Алгоритм ускоренного повтора был впервые реализован в 4.3BSD Tahoe и после него использовалась процедура **slow start**. Алгоритм ускоренного восстановления был реализован в 4.3BSD Reno.

Вопросы безопасности

Рассматриваемые в этом документе вопросы не связаны с безопасностью.

Литература

[1] B. Braden, ed., "Requirements for Internet Hosts -- Communication Layers," RFC 1122[1], Oct. 1989.

[2] V. Jacobson, "Congestion Avoidance and Control," Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988. ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

[3] V. Jacobson, "Modified TCP Congestion Avoidance Algorithm," end2end-interest mailing list, April 30, 1990. ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail.

[4] W. R. Stevens, "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley, 1994.

[5] G. R. Wright, W. R. Stevens, "TCP/IP Illustrated, Volume 2: The Implementation", Addison-Wesley, 1995.

Адрес автора

W. Richard Stevens

1202 E. Paseo del Zorro

Tucson, AZ 85718

телефон: 520-297-9416

электронная почта: rstevens@noao.edu

сайт: http://www.noao.edu/~rstevens

Перевод на русский язык

Николай Малых

электронная почта: nmalykh@protocols.ru

сайт: http://www.nmalykh.org

[1]На сайте www.protocols.ru имеется перевод этой спецификации на русский язык. *Прим. перев.*