Network Working Group
Request for Comments: 5681
Obsoletes: 2581
Category: Standards Track

M. Allman
V. Paxson
ICSI
E. Blanton
Purdue University

September 2009

Контроль насыщения в TCP

TCP Congestion Control

Аннотация

В этом документе определены 4 связанных между собой алгоритма контроля насыщения[1] для протокола TCP - slow start[2], congestion avoidance[3], fast retransmit[4] и fast recovery[5]. Кроме того, в документе указано, как протоколу TCP следует начинать передачу после сравнительно долгого периода бездействия, а также рассмотрены различные методы генерации подтверждений. Данный документ служит заменой RFC 2581.

Статус документа

Документ содержит спецификацию стандартного протокола для сообщества Internet и является приглашением к дискуссии в целях развития и совершенствования протокола. Сведения о стандартизации и состоянии данного протокола можно найти в документе Internet Official Protocol Standards (STD 1). Допускается свободное распространение данного документа.

Оглавление

1. Введение

В этом документе даны спецификации четырёх алгоритмов контроля насыщения для протокола TCP [RFC793]: slow start, congestion avoidance, fast retransmit и fast recovery. Эти алгоритмы были опубликованы в документах [Jac88] и [Jac90]. Использование алгоритмов с протоколом TCP стандартизовано в [RFC1122].

Отметим, что в документе [Ste94] приводятся примеры реального использования описанных здесь алгоритмов, а [WS95] содержит пояснения к исходному коду реализации этих алгоритмов в BSD.

---

[1] При переводе термина “congestion” термин “перегрузка” использовался наряду с термином “насыщение”. *Прим. перев.*
[2] Замедленный старт.
[3] Предотвращение насыщения.
[4] Ускорение повторной передачи.
[5] Ускоренное восстановление.

В дополнение к спецификации алгоритмов контроля насыщения этот документ задаёт, как следует вести себя соединениям TCP после сравнительно долгого периода бездействия, а также задаются и разъясняются некоторые вопросы, связанные с генерацией подтверждений TCP ACK.

Этот документ отменяет действие [RFC2581], который, в свою очередь, отменил [RFC2001].

Глава данного документа 2 содержит определения используемых в документе терминов. В главе 3 приведены спецификации алгоритмов контроля насыщения. Глава 4 посвящена ситуациям, связанным с алгоритмами контроля насыщения, а в главе 5 обсуждаются вопросы безопасности.

Ключевые слова "MUST" (**необходимо**), "MUST NOT" (**недопустимо**), "REQUIRED" (**требуется**), "SHALL" (**следует**), "SHALL NOT" (**не следует**), "SHOULD" (**следует**), "SHOULD NOT" (**не следует**), "RECOMMENDED" (**рекомендуется**), "MAY" (**возможно**) и "OPTIONAL" (**необязательно**) в данном документе трактуются в соответствии с [Bra97].

2. Определения

В этой главе приводятся определения некоторых терминов, которые будут использованы в документе.

SEGMENT - сегмент
Сегментом является **любой** пакет данных или подтверждение TCP/IP.

SENDER MAXIMUM SEGMENT SIZE (SMSS) - максимальный размер сегмента для отправителя
SMSS представляет собой размер самого большого сегмента, который может быть передан отправителем. Это значение может определяться на основе максимального блока данных, передаваемого через сеть, алгоритма определения Path MTU [RFC1191, RFC4821], RMSS (см. следующее определение) и других факторов. Размер не включает заголовков и опций TCP/IP.

RECEIVER MAXIMUM SEGMENT SIZE (RMSS) - максимальный размер сегмента для получателя
RMSS - размер максимального сегмента, который желает принимать получатель. Это значение задается в опции MSS, передаваемой хостом при организации соединения. Если опция MSS при организации соединения не была задана, используется значение 536 байтов [RFC1122]. Размер не включает заголовков и опций TCP/IP.

FULL-SIZED SEGMENT - полноразмерный сегмент
Сегмент, содержащий максимально допустимое количество данных (т. е., SMSS байтов).

RECEIVER WINDOW (rwnd) - размер окна принимающей стороны
Последнее анонсированное значение размера окна принимающей стороны.

CONGESTION WINDOW (cwnd) - размер окна насыщения
Переменная состояния TCP, которая ограничивает количество данных, разрешённых протоколу для передачи. В любой момент для TCP **недопустима** передача данных с порядковыми номерами, превышающими значение суммы наибольшего из подтверждённых порядковых номеров и меньшего из двух значений cwnd и rwnd.

INITIAL WINDOW (IW) - начальный размер окна
Начальным размером окна является размер окна насыщения отправителя после завершения трехэтапного согласования параметров.

LOSS WINDOW (LW) - размер окна потерь
Размер окна насыщения после того, как отправитель TCP обнаружит потерю, используя свой таймер повтора передачи.

RESTART WINDOW (RW) - размер окна перезапуска
Размер окна насыщения после того, как TCP возобновит передачу из состояния бездействия (для случая использования алгоритма slow start см. параграф 4.1).

FLIGHT SIZE - размер звена
Количество данных, которые уже переданы, но ещё не подтверждены кумулятивно.

DUPLICATE ACKNOWLEDGMENT - дубликат подтверждения
Подтверждение считается «дубликатом» в описанных здесь алгоритмах, если (a) получатель ACK имеет остающиеся для передачи данные, (b) входящее подтверждение не содержит каких-либо данных, (c) оба флага SYN и FIN сброшены, (d) номер подтверждения равен или превышает значение наибольшего номера подтверждения, полученного в данном соединении (TCP.UNA из [RFC793]) и (e) анонсируемое во входящем подтверждении окно равно окну, анонсированному в последнем входящем подтверждении.
В дополнение к этому TCP, использующий селективные подтверждения (SACK[1]) [RFC2018, RFC2883], может применять информацию SACK для определения дубликатов подтверждений (содержит ли ACK ранее неизвестную информацию SACK).

3. Алгоритмы контроля насыщения

В этой главе определены четыре алгоритма контроля насыщения - slow start, congestion avoidance, fast retransmit и fast recovery, разработанные в [Jac88] и [Jac90]. В некоторых случаях для отправителя TCP предпочтительно быть более консервативным, нежели позволяют алгоритмы, но для TCP **недопустимо** быть более агрессивным, чем позволяют алгоритмы (т. е., **недопустимо** передавать данные, когда рассчитанное с помощью алгоритмов значение cwnd не разрешает передачу).

Отметим также, что описанные в этом документе алгоритмы рассчитаны на использование потери пакетов в качестве индикации перегрузки (насыщения). Можно также использовать механизм явных уведомлений о насыщении (ECN[2]), описанный в [RFC3168].

3.1 Алгоритмы Slow Start и Congestion Avoidance

Алгоритмы замедленного старта (slow start) и предотвращения перегрузки (congestion avoidance) **должны** использоваться отправителем TCP для контроля за передачей в сеть остающихся неотправленными данных. Для реализации этих алгоритмов в состояние соединения TCP добавлены две переменных - размер окна насыщения (cwnd) - задаваемый на стороне отправителя предел для количества данных, которые отправитель может передать в сеть до получения подтверждения (ACK), и анонсируемое получателем окно (rwnd), которое определяет

[1]Selective acknowledgment.
[2]Explicit Congestion Notification.

установленный на приёмной стороне предел размера остающихся данных. Передачей управляет меньшее из двух значений cwnd и rwnd.

Ещё одна переменная состояния ssthresh[1] используется для определения момента, когда следует использовать алгоритм замедленного старта или предотвращения перегрузки в соответствии с приведёнными ниже описаниями.

Начало передачи в сеть с неизвестными условиями требует от TCP достаточно медленной проверки сети с целью определения доступной «ёмкости» для того, чтобы избежать насыщения сети избыточным потоком данных. Алгоритм slow start используется для решения этой задачи на начальном этапе передачи или после восстановления в результате потери пакетов, обнаруженной с помощью таймера повторной передачи. Этот алгоритм служит также для запуска процесса ACK-синхронизации, который используется отправителем TCP для отправки данных в сети при использовании алгоритмов замедленного старта, предотвращения перегрузки и восстановления после потерь.

IW - начальное значение cwnd - **должно** устанавливаться с использованием приведённых рекомендаций для максимального значение:

```
Если SMSS > 2190 байтов
    IW = 2 * SMSS байтов; недопустимо превышать размер 2 сегментов
Если (SMSS > 1095 байтов) и (SMSS <= 2190 байтов)
    IW = 3 * SMSS байтов; недопустимо превышать размер 3 сегментов
Если SMSS <= 1095 байтов
    IW = 4 * SMSS байтов; недопустимо превышать размер 4 сегментов
```

Как сказано в [RFC3390], на основании SYN/ACK и подтверждения SYN/ACK **недопустимо** увеличивать размер окна насыщения. Более того, даже при потере SYN или SYN/ACK начальный размер окна, используемый отправителем после корректно переданного SYN **должен** быть равен одному сегменту, содержащему не более SMSS байтов.

Детальное обоснование и обсуждение установки значения IW приведено в работе [RFC3390][2].

Когда начальное окно размером более 1 сегмента реализуется вместе с Path MTU Discovery [RFC1191] и обнаружено, что используемое значение MSS слишком велико, значение окна насыщения cwnd **следует** уменьшить для предотвращения «выброса» большого числа мелких сегментов. Конкретно значение cwnd **следует** уменьшать SHOULD на отношение старого размера сегмента к новому размеру сегмента.

Начальное значение ssthresh **следует** устанавливать сколь угодно высоким (например, некоторые реализации используют в качестве порога размер анонсируемого окна), но значение порога **должно** быть уменьшено при возникновении насыщения. Установка для ssthresh как можно более высокого значения позволяет определять скорость передачи в соответствии с условиями в сети, а не в соответствии с ограничениями некого конкретного хоста. В ситуациях, когда оконечные системы хорошо понимают путь через сеть, имеет смысл более аккуратно устанавливать значение ssthresh (например, так, чтобы конечные хосты не создавали насыщения на пути через сеть).

Алгоритм замедленного старта используется в тех случаях, когда **cwnd** < **ssthresh**, а при **cwnd** > **ssthresh** применяется алгоритм предотвращения перегрузки. Если cwnd = ssthresh отправитель может использовать любой из этих алгоритмов.

При замедленном старте TCP увеличивает размер окна cwnd не более, чем на SMSS байтов для каждого пакета ACK, кумулятивно подтверждающего доставку новой порции данных. Замедленный старт завершается, когда размер окна насыщения cwnd превышает порог ssthresh (или становится равным этому порогу), а также при возникновении перегрузки. Хотя традиционные реализации TCP увеличивают cwnd ровно на SMSS байтов при получении ACK, подтверждающего новые данные, **рекомендуется** увеличивать значение cwnd, как показано в уравнении

```
cwnd += min (N, SMSS)                    (2)
```

где N - число ранее не подтверждённых байтов, подтверждаемых входящим ACK. Это уточнение является частью процедуры Appropriate Byte Counting[3] [RFC3465] и обеспечивает устойчивость к некорректному поведению получателей, которые могут пытаться спровоцировать отправителя на искусственное повышение cwnd с использованием механизма ACK Division [SCWA99]. Этот способ воздействия заключается в том, что получатель передаёт для одного сегмента данных TCP множество сегментов ACK, каждый из которых подтверждает только часть данных. В этом случае реализация TCP, увеличивающая cwnd на SMSS для каждого такого ACK будет недопустимо повышать объем данных, отправляемых в сеть.

В процессе предотвращения перегрузки размер окна cwnd увеличивается на 1 полноразмерный сегмент за каждый период кругового обхода RTT[4]. Предотвращение насыщения продолжается до тех пор, пока насыщение наблюдается. Основными рекомендациями для увеличения cwnd при предотвращении перегрузки являются следующие:

- **можно** увеличивать cwnd на SMSS байтов;
- **следует** увеличивать cwnd в соответствии с уравнением (2) один раз за период RTT;
- **недопустимо** увеличивать cwnd более, чем на SMSS байтов.

Отметим, что [RFC3465] разрешает увеличивать cwnd более, чем на SMSS байтов для входящих подтверждений в процессе замедленного старта в качестве экспериментов. Однако такое поведение недопустимо в качестве стандартного.

Рекомендуется увеличивать cwnd во время предотвращения перегрузки на число байтов, которые были подтверждены сегментами ACK для новых данных (недостатком этой рекомендации является необходимость поддержки дополнительной переменной состояния). Когда число подтверждённых данных достигает значения cwnd, последнее может быть увеличено на значение до SMSS байтов. Отметим, что в процесс предотвращения перегрузки **недопустимо** увеличивать размер окна насыщения более, чем на SMSS байтов за период RTT. Этот метод позволяет

---

[1] Порог замедленного старта.

[2] В RFC Errata предложено дополнение этого абзаца, содержащее следующий текст: «Значение IW для случаев, когда 1095 < SMSS ≤ 2190 байтов, будет отличаться от рекомендаций [RFC3390]. Новое определение будет сохранять предшествующее значение cwnd, кратное SMSS, и предотвращать передачу неполных сегментов.». *Прим. перев.*

[3] Appropriate Byte Counting - подходящий учёт байтов.

[4] Round-trip time.

реализациям TCP увеличивать cwnd на 1 сегмент за период RTT в случаях задержки ACK и обеспечивает устойчивость к атакам Division.

Другим общепринятым способом, с помощью которого TCP **может** менять значение cwnd в процессе предотвращения перегрузки, является уравнение (3):

```
cwnd += SMSS*SMSS/cwnd                    (3)
```

Это изменение выполняется на каждый входящий сегмент ACK, который подтверждает новые данные. Уравнение (3) обеспечивает подходящее приближение для лежащего в основе принципа увеличения cwnd на 1 полноразмерный сегмент за период RTT (отметим, что для соединений, в которых получатель подтверждает каждый второй пакет, уравнение (3) обеспечивает менее агрессивное поведение - увеличение cwnd примерно на каждые два периода RTT).

Примечание для разработчиков. Поскольку в реализациях TCP обычно используется целочисленная арифметика, выражение (3) может не приводить к увеличению размера окна cwnd, когда окно насыщения очень велико (больше, чем SMSS*SMSS). Если выражение (3) даёт нулевой результат, его **следует «**округлять» до 1 байта.

Примечание для разработчиков. В старых реализациях используется дополнительная положительная константа в правой части выражения (3). Такой подход некорректен и может вести к снижению производительности [RFC2525].

Примечание для разработчиков. Некоторые реализации поддерживают размер окна cwnd в байтах, а другие - в полноразмерных сегментах. В последнем случае использование выражения (3) становится затруднительным и может оказаться предпочтительным механизм, рассмотренный в предыдущем абзаце.

Когда отправитель TCP обнаруживает потерю сегмента с помощью таймера повтора передачи, для переменной ssthresh **должно** устанавливаться значение, не превышающее значение выражения 4:

```
ssthresh = max (FlightSize / 2, 2*SMSS)          (4)
```

Как было отмечено выше, FlightSize показывает количество данных, которые ещё находятся в сети (переданы, но не подтверждены).

С другой стороны, когда отправитель TCP обнаруживает потерю сегмента с помощью таймера повторной передачи и данный сегмент уже был повторно передан с помощью этого таймера по крайней мере один раз, значение ssthresh не меняется.

Примечание для разработчиков. Легко ошибиться и использовать cwnd вместо FlightSize, что может в некоторых реализациях приводить к увеличению порога до значений, превышающих rwnd.

Более того, при возникновении тайм-аута (как указано в [RFC2988]) необходимо устанавливать для размера окна насыщения cwnd значение, не превышающее размер окна потерь LW[1], которое равно 1 полноразмерному сегменту (независимо от значения IW). Следовательно, после повтора передачи отброшенного сегмента отправитель TCP использует замедленный старт для увеличения окна от 1 полноразмерного сегмента до нового значения ssthresh, после чего снова включается механизм предотвращения перегрузки.

Как показано в [FF96] и [RFC3782], основанное на замедленном старте восстановление после тайм-аута может приводить необоснованным повторам передачи, ведущей к дублированию подтверждений. Реакция на приём таких дубликатов ACK в реализациях TCP существенно различается. Этот документ не задаёт способа трактовки таких подтверждений, но здесь отмечается, что могут быть получены некоторые преимущества в результате проведения дополнительных исследований и спецификации такого поведения.

3.2 Алгоритмы Fast Retransmit и Fast Recovery

Получателю TCP **следует** незамедлительно передавать дубликат ACK при получении сегмента с нарушением порядка доставки. Это делается для того, чтобы с помощью пакета ACK информировать отправителя о том, что сегмент был получен с нарушением порядка и указать порядковый номер ожидаемого сегмента. С точки зрения отправителя дубликат ACK может быть вызван различными проблемами в сети. Во-первых, причиной может служить отбрасывание сегментов. В этом случае все сегменты после отброшенного будут порождать дубликаты ACK. Во-вторых, дубликаты ACK могут быть обусловлены нарушением порядка доставки сегментов (например, при доставке по разным путям [Pax97]). Наконец, причиной появления дубликатов ACK может быть репликация пакетов ACK или сегментов данных в сети. В дополнение к сказанному получателю TCP **следует** незамедлительно передавать подтверждение ACK при получении сегмента, который полностью или частично заполняет пропуски в порядковых номерах. Это позволит предоставить своевременную информацию отправителю, выполняющему восстановление после потери с использованием тайм-аута повторной передачи (retransmission timeout), быстрого повтора (fast retransmit) или улучшенного алгоритма восстановления (loss recovery), описанного в параграфе 4.3.

Отправителю TCP **следует** использовать алгоритм быстрого повтора для детектирования потери и исправления ошибки с использованием входящих дубликатов ACK. Алгоритм быстрого повтора использует прибытие 3 дубликатов ACK (как определено в параграфе 2, без каких-либо промежуточных сегментов ACK, вызывающих смещение SND.UNA), как индикацию потери сегмента. После получения 3 дубликатов ACK протокол TCP выполняет повторную передачу сегмента, который представляется потерянным, без ожидания завершения отсчёта таймера повтора передачи.

После того, как алгоритм быстрого повтора передаст те данные, которые представляются включёнными в отсутствующий сегмент, алгоритм «быстрого восстановления» (fast recovery) регулирует передачу новых данных, пока не будет получено подтверждение ACK, не являющееся дубликатом. Алгоритм замедленного старта не используется по той причине, что получение дубликатов ACK не только указывает на потерю сегмента, но и говорит о высокой вероятности того, что сегменты покинули сеть (хотя массированное дублирование сегментов в сети может сделать такое допущение некорректным). Иными словами, поскольку получатель может генерировать дубликат ACK только при получении сегмента, считается, что этот сегмент покинул сеть и находится в приёмном буфере, более не потребляя ресурсов сети. Более того, поскольку «синхронизация» ACK сохраняется [Jac88], отправитель TCP может продолжать передачу новых сегментов (хотя и со сниженным значением cwnd).

Алгоритмы быстрого повтора и быстрого восстановления обычно реализуются вместе, как описано ниже.

[1]Loss window.

1. При получении отправителем первого и второго дубликата ACK TCP **следует** передать сегмент не переданных ранее данных в соответствии с [RFC3042] - эта возможность обеспечивается тем, что при анонсируемом получателем окне приёма выполняется условие FlightSize ≤ cwnd + 2*SMSS и имеются новые данные для передачи. В дальнейшем отправителю TCP **недопустимо** изменять значение cwnd для отражения этих двух сегментов [RFC3042]. Отметим, что отправителям, использующим SACK [RFC2018], **недопустимо** передавать новые данные, пока входящие дубликаты подтверждений содержат новую информацию SACK.

2. При получении третьего дубликата ACK TCP **должен** установить значение ssthresh, которое не превышает значения выражения (4). При использовании [RFC3042] ограниченную передачу дополнительных данных **недопустимо** учитывать в этом расчёте.

3. Потерянный сегмент, начиная с SND.UNA, **должен** быть передан заново, а для cwnd устанавливается значение ssthresh + 3*SMSS. Это искусственно «увеличивает» размер окна насыщения на число сегментов (три), которые покинули сеть и буферизованы получателем.

4. Для каждого принятого дополнительного дубликата ACK (после третьего) значение cwnd **должно** увеличиваться на SMSS. Это искусственно увеличивает окно насыщения для того, чтобы отразить выход из сети дополнительных сегментов.

   Примечание. В работе [SCWA99] рассмотрена атака со стороны получателя, в которой отправителю передаётся множество фальшивых дубликатов ACK с целью расширения cwnd и соответствующего повышения скорости отправки с передающей стороны. TCP **может**, следовательно, ограничивать кратность расширения cwnd в процессе восстановления после потерь до числа остающихся в сети сегментов (или близкого значения).

   Примечание. Если расширенный механизм восстановления (типа описанного в параграфе 4.3) не используется, такое увеличение FlightSize может (в соответствии с уравнением (4)) вызывать незначительное расширение cwnd и ssthresh, поскольку несколько сегментов между SND.UNA и SND.NXT, предполагающиеся вышедшими из сети, по прежнему будут отражены в значении FlightSize.

5. При наличии новых не переданных данных, если позволяет новое значение cwnd и анонсированное получателем окно, TCP **следует** передать 1*SMSS байтов ранее не передававшихся данных.

6. При получении следующего пакета ACK, подтверждающего ранее не подтверждённые данные TCP **должен** установить cwnd = ssthresh (значение порога, заданное в п. 2). Это называется «уменьшением» размера окна насыщения.

   Этому пакету ACK следует быть подтверждением, вызванным повтором в п. 3 в течение одного периода RTT после повтора (хотя подтверждение может прийти быстрей при наличии существенного нарушения порядка доставки сегментов данных на приёмной стороне). Кроме того, этому пакету ACK следует подтверждать все промежуточные сегменты, переданные между потерянным сегментом и получением третьего дубликата ACK, если ни один из этих сегментов не был потерян.

Примечание: Известно, что этот алгоритм в общем случае не обеспечивает достаточно эффективного восстановления при множественных потерях в одном «звене[1]» пакетов [FF96]. Решение этой проблемы описана в параграфе 4.3.

4. Дополнительные вопросы

4.1 Перезапуск соединения

Хорошо известной проблемой, связанной с описанными выше алгоритмами контроля насыщения TCP, является потенциальная возможность возникновения недопустимо высоких уровней трафика, передаваемого через соединение TCP после сравнительно долгого простоя. После завершения периода бездействия TCP не может использовать ACK-синхронизацию для передачи новых сегментов в сеть, поскольку все подтверждения ACK уже получены из сети. Следовательно, как было сказано ранее, TCP после продолжительного простоя потенциально может передать блок данных размером cwnd со скоростью среды. Кроме того, изменение условий в сети при продолжительном бездействии может сделать неточной оценку пропускной способности сети между двумя точками на основе cwnd.

Документ [Jac88] рекомендует использовать замедленный старт для восстановления передачи после сравнительно долгого простоя. Замедленный старт позволяет восстановить ACK-синхронизацию, как это делается на начальном этапе работы соединения. Этот механизм довольно широко распространён и работает следующим образом. Когда TCP не получает сегмент в течение времени, превышающего тайм-аут повторной передачи, размер окна насыщения cwnd уменьшается до значения RW[2] перед началом передачи.

В данном стандарте мы определяем RW = min(IW,cwnd).

Использование принятого последним сегмента для решения вопроса об уменьшении cwnd не приводит к снижению размера окна насыщения cwnd в распространённом случае для продолжительных соединений HTTP [HTH98]. В таких случаях Web-сервер получает запрос до передачи данных Web-клиенту. Получение запроса ведёт к отрицательному результату при проверке бездействия соединения и позволяет TCP начать передачу со значением cwnd, которое может оказаться недопустимо большим.

Поэтому протоколу TCP **следует** устанавливать перед началом передачи для окна cwnd значение, не превышающее RW, если протокол TCP не передавал данных в течение времени, превышающего тайм-аут повтора передачи.

4.2 Генерация подтверждений

На приёмной стороне **следует** использовать алгоритм задержки подтверждений, описанный в [RFC1122]. При использовании этого алгоритма для получателя **недопустима** избыточная задержка генерации подтверждений. В частности, пакеты ACK **следует** генерировать по крайней мере для каждого второго полноразмерного сегмента и генерация подтверждения **должна** происходить в течение не более 500 мсек с момента доставки первого неподтвержденного пакета.

[1] Группа пакетов, одновременно находящихся в сети между отправителем и получателем. *Прим. перев.*
[2] Restart window – окно рестарта.

Требование, в соответствии с которым пакеты ACK **следует** генерировать по крайней мере для каждого второго полноразмерного пакета, в документе [RFC1122] указано в одном месте как **следует**, в в другом – как **должно**. Ввиду неоднозначности мы будет использовать трактовку **следует**. Подчеркнём также, что уровень **следует** означает, что разработчик может отказаться от выполнения этого требования лишь после тщательной оценки возможных последствий. Обсуждение проблем, связанных с невыполнением требования генерации подтверждений не реже, чем для каждого второго полноразмерного сегмента, приводится в параграфе Stretch ACK violation документа [RFC2525].

В некоторых случаях между отправителем и получателем может отсутствовать согласие по поводу того, что собой представляет полноразмерный сегмент. Реализация протокола считается совместимой с требованиями этого параграфа, если она передаёт по крайней мере одно подтверждение каждый раз по получении 2*RMSS байтов новых данных от отправителя (RMSS – максимальный размер сегмента, указанный получателем отправителю, или принятое по умолчанию значение 536 байтов [RFC1122], если получатель не указал опцию MSS при организации соединения). Отправитель может быть вынужден использовать сегменты меньшего, чем RMSS, размера в результате ограничения MTU, path MTU или воздействия иных факторов. В качестве примера рассмотрим случай, когда получатель анонсирует RMSS = X, но отправитель использует сегменты меньшего размера Y (Y < X) вследствие ограничения path MTU или значения MTU на стороне отправителя. Получатель будет генерировать подтверждения ACK более редко, если он будет дожидаться доставки 2*X перед генерацией ACK. Очевидно, что подтверждения для 2 сегментов по Y байтов передавались бы чаще. Следовательно, несмотря на то, что не определён специфический алгоритм, для получателя желательно попытаться предотвратить подобные ситуации (например, путём подтверждения каждого второго сегмента независимо от их размера). Повторим ещё раз, что **недопустима** задержка передачи ACK более чем на 500 мсек в результате ожидания доставки второго полноразмерного сегмента.

Сегменты, доставленные с нарушением порядка, **следует** подтверждать незамедлительно для того, чтобы ускорить процесс восстановления. Для включения алгоритма ускоренного повтора получателю **следует** незамедлительно передать дубликат ACK при получении сегмента данных после пропуска порядковых номеров. Для обеспечения обратной связи с отправителем, выполняющим процедуру восстановления, получателю **следует** незамедлительно передать подтверждение ACK при получении сегмента, который полностью или частично заполняет пропуски в порядковых номерах.

Для получателя TCP **недопустимо** генерировать более одного подтверждения ACK для каждого входящего сегмента, если это подтверждение не является обновлением предлагаемого размера окна в тех случаях, когда принимающее приложение забирает новые данные [[RFC813] и стр. 42[RFC793].

4.3 Механизмы восстановления при потерях

Исследователями TCP было предложено и опубликовано в RFC множество алгоритмов восстановления при потере сегментов, повышающих эффективность работы алгоритмов fast retransmit и fast recovery. Некоторые из таких алгоритмов (например, [FF96], [MM96a], [MM96b] b [RFC3517]) основаны на использовании опции селективных подтверждений (SACK[1]) [RFC2018], а для других (например, [Hoe96], [FF96] и [RFC3782]) SACK не требуется. Алгоритмы, работающие без SACK, используют «частичные подтверждения[2]» (пакеты ACK, которые подтверждают новые данные, но не подтверждают пропущенные при наличии потерь) для включения механизма повтора передачи. Хотя данный документ не стандартизует ни один из алгоритмов, которые могут повышать эффективность fast retransmit/fast recovery, такие алгоритмы неявно разрешены, если они соответствуют общим принципам базовых алгоритмов, описанных выше.

Т. е., при обнаружении первой потери в окне для ssthresh **должно** быть установлено значение, не превышающее значение уравнения (4). Далее, пока все сегменты в окне данных не будут восстановлены, число сегментов, передаваемых в течение каждого периода RTT, **должно** быть не более половины от числа остававшихся на момент обнаружения потери сегментов. И, наконец, после успешной передачи всех потерянных сегментов в данном окне для параметра cwnd **должно** быть установлено значение, не превышающее ssthresh, и для дальнейшего увеличения cwnd **должен** использоваться механизм предотвращения перегрузки. Потери в двух последовательных окнах данных или потерю при повторе передачи следует трактовать как двухкратную индикацию насыщения и, следовательно, значения cwnd и ssthresh **должны** в таких случаях уменьшаться дважды.

Мы **рекомендуем** разработчикам реализаций TCP применять ту или иную форму улучшенного восстановления после потерь, способную справиться с множественными потерями в одном окне данных. Алгоритмы, описанные подробно в [RFC3782] и [RFC3517] соответствуют приведённым выше общим принципам. Отметим, что набор соответствующих общим принципам алгоритмов не исчерпывается двумя указанными, однако эта пара алгоритмов была проверена на практике и в настоящее время стандартизуется.

5. Вопросы безопасности

Данный документ требует от реализации TCP снижать скорость передачи при повторе передачи по тайм-ауту и доставке дубликатов подтверждений. Атакующий может оказывать негативное влияние на работу соединений TCP, организуя потерю пакетов данных или подтверждений, а также путём фабрикации избыточных дубликатов подтверждений.

С учётом атак ACK division, описанных в [SCWA99], данная спецификация **рекомендует** увеличивать размер окна насыщения на основе числа байтов, недавно подтверждённых в каждом поступающем сегменте ACK вместо увеличения на постоянное значение при каждом поступлении сегмента ACK (как описано в параграфе 3.1).

Работа Internet в значительной степени основана на корректности реализации этих алгоритмов, обеспечивающих стабильность сети и предотвращение коллапса в результате насыщения. Атакующий может заставить конечные точки TCP отвечать более агрессивно на угрозу насыщения путём фабрикации избыточных дубликатов подтверждений или избыточных подтверждений для новых данных. Концептуально такая атака может привести к связанному с насыщением коллапсу для части сети.

[1]TCP selective acknowledgment.
[2]Partial acknowledgment.

6. Различия между RFC 2001 и RFC 2581

[RFC2001] был подвергнут существенной редакторской правке, не позволяющей, по сути, создать список различий между [RFC2001] и [RFC2581]. Целью подготовки [RFC2581] было не изменение рекомендаций, содержащихся в [RFC2001], а разъяснение вопросов, которые не были детально рассмотрены в [RFC2001]. В частности, [RFC2581] содержит рекомендации в части поведения соединений TCP после сравнительно долгого бездействия, а также спецификации и разъяснения по некоторым вопросам генерации TCP ACK. Кроме того, была вдвое (с одного сегмента до 2) увеличена верхняя граница начального размера окна насыщения.

7. Отличия от RFC 2581

Добавлено определение дубликатов подтверждений на основе определения, используемого в BSD TCP.

В документе указано, что поведение по отношению к дубликатам ACK после завершения отсчёта таймера повторной передачи требует дальнейшего изучения и не задается явно в данном документе.

Изменены требования к начальному размеру окна, чтобы позволить больший размер[1], стандартизованный в [RFC3390]. В дополнение к этому было детализировано поведение для случаев, когда значение начального размера окна оказывается слишком большим в результате определения Path MTU [RFC1191].

Изменены рекомендации по выбору начального значения порога ssthresh - указано, что его **следует** делать произвольно большим, взамен указанного в предыдущей версии «**можно** делать произвольно большим». Это сделано для обеспечения дополнительных рекомендаций разработчикам.

В процессе замедленного старта явно рекомендовано использование Appropriate Byte Counting [RFC3465] с L=1*SMSS. Метод увеличения cwnd, заданный в [RFC2581], по-прежнему явно допускается. Рекомендуется использовать подсчёт байтов в процессе предотвращения перегрузки (congestion avoidance), хотя метод из [RFC2581] и другие безопасные методы остаются допустимыми.

Прояснена трактовка ssthresh при тайм-ауте повтора передачи. В частности, указано, что при первом повторе передачи данного сегмента для ssthresh должно устанавливаться значение, равное половине FlightSize, которое остаётся постоянным в случае дополнительных повторов передачи того же сегмента.

Разъяснено описание ускоренного повтора (fast retransmit) и ускоренного восстановления (fast recovery), а также рекомендовано применение ограниченной передачи[2] [RFC3042].

Реализациям TCP **можно** ограничивать число дубликатов ACK, которые могут искусственно завышать значение cwnd при восстановлении в случаях потери, числом сегментов, остающихся для передачи, с целью предотвращения атак на базе обманных подтверждений, описанных в [SCWA99].

Изменено значение окна при перезапуске (restart window) с IW на min(IW,cwnd). Такое поведение было отмечено в [RFC2581], как экспериментальное.

Разработчикам TCP рекомендовано реализовать улучшенный алгоритм восстановления при потере сегментов, соответствующий изложенным в этом документе принципам.

Обновлено рассмотрение вопросов безопасности с учётом атак ACK division и рекомендацией использовать подсчёт байтов для предотвращения таких атак.

8. Благодарности

Основные алгоритмы, описанные здесь, разработал Van Jacobson [Jac88, Jac90]. В дополнение к ним совместно с Hari Balakrishnan и Sally Floyd был разработан алгоритм Limited Transmit[3] [RFC3042]. Начальное значение окна насыщения, указанное в этом документе, получено в результате работы с Sally Floyd и Craig Partridge [RFC2414, RFC3390].

W. Richard (Рич) Stevens написал первую версию этого документа [RFC2001] и был соавтором второй версии [RFC2581]. В настоящем документе многие вопросы были разъяснены, благодаря вкладу Рича в понимание контроля насыщения TCP, а также его помощи по многим вопросам, относящимся к сетям.

Мы хотели бы подчеркнуть, что ответственность за недостатки и ошибки, встречающиеся в этом документа, лежит исключительно на авторах текущей версии.

Часть текста данного документа заимствована из книг «TCP/IP Illustrated, Volume 1: The Protocols» Ричарда Стивенса (W. Richard Stevens) (Addison-Wesley, 1994) и «TCP/IP Illustrated, Volume 2: The Implementation» Гери Райта (Gary R. Wright) и Ричарда Стивенса (Addison-Wesley, 1995). Этот материал включён с разрешения издательства Addison-Wesley.

Anil Agarwal, Steve Arden, Neal Cardwell, Noritoshi Demizu, Gorry Fairhurst, Kevin Fall, John Heffner, Alfred Hoenes, Sally Floyd, Reiner Ludwig, Matt Mathis, Craig Partridge и Joe Touch внесли множество полезных предложений.

9. Литература

9.1. Нормативные документы

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[1] Larger Initial Window.

[2] Строго говоря, явная рекомендация использования «» в документе отсутствует. В параграфе 3.2 лишь сказано: «При использовании [RFC3042] ограниченную передачу дополнительных данных **недопустимо** учитывать в этом расчёте.». *Прим. перев.*

[3] Ограниченная передача.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Дополнительная литература

[CJ89] Chiu, D. and R. Jain, "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks"[1], Journal of Computer Networks and ISDN Systems, vol. 17, no. 1, pp. 1-14, June 1989.

[FF96] Fall, K. and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and SACK TCP", Computer Communication Review, July 1996, ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z.

[Hoe96] Hoe, J., "Improving the Start-up Behavior of a Congestion Control Scheme for TCP"[2], In ACM SIGCOMM, August 1996.

[HTH98] Hughes, A., Touch, J., and J. Heidemann, "Issues in TCP Slow-Start Restart After Idle", Work in Progress, March 1998.

[Jac88] Jacobson, V., "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.

[Jac90] Jacobson, V., "Modified TCP Congestion Avoidance Algorithm", end2end-interest mailing list, April 30, 1990. ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail.

[MM96a] Mathis, M. and J. Mahdavi, "Forward Acknowledgment: Refining TCP Congestion Control", Proceedings of SIGCOMM'96, August, 1996, Stanford, CA. Доступна по ссылке http://www.psc.edu/networking/papers/papers.html

[MM96b] Mathis, M. and J. Mahdavi, "TCP Rate-Halving with Bounding Parameters", Technical report. Доступна по ссылке http://www.psc.edu/networking/papers/FACKnotes/current.

[Pax97] Paxson, V., "End-to-End Internet Packet Dynamics"[3], Proceedings of SIGCOMM '97, Cannes, France, Sep. 1997.

[RFC813] Clark, D., "Window and Acknowledgement Strategy in TCP", RFC 813, July 1982.

[RFC2001] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, January 1997.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[RFC2414] Allman, M., Floyd, S., and C. Partridge, "Increasing TCP's Initial Window", RFC 2414, September 1998.

[RFC2525] Paxson, V., Allman, M., Dawson, S., Fenner, W., Griner, J., Heavens, I., Lahey, K., Semke, J., and B. Volz, "Known TCP Implementation Problems", RFC 2525, March 1999.

[RFC2581] Allman, M., Paxson, V., and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC2883] Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP", RFC 2883, July 2000.

[RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[RFC3042] Allman, M., Balakrishnan, H., and S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit", RFC 3042, January 2001.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3390] Allman, M., Floyd, S., and C. Partridge, "Increasing TCP's Initial Window", RFC 3390, October 2002.

[RFC3465] Allman, M., "TCP Congestion Control with Appropriate Byte Counting (ABC)", RFC 3465, February 2003.

[RFC3517] Blanton, E., Allman, M., Fall, K., and L. Wang, "A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP", RFC 3517, April 2003.

[RFC3782] Floyd, S., Henderson, T., and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 3782, April 2004.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[SCWA99] Savage, S., Cardwell, N., Wetherall, D., and T. Anderson, "TCP Congestion Control With a Misbehaving Receiver"[4], ACM Computer Communication Review, 29(5), October 1999.

[Ste94] Stevens, W., "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley, 1994.

[WS95] Wright, G. and W. Stevens, "TCP/IP Illustrated, Volume 2: The Implementation", Addison-Wesley, 1995.

Адреса авторов

Mark Allman

International Computer Science Institute (ICSI)

1947 Center Street

[1]Эта статья доступна по ссылке http://pages.cs.wisc.edu/~suman/courses/740/papers/chiu89isdn.pdf. *Прим. перев.*
[2]Эта статья доступна по ссылке http://conferences.sigcomm.org/sigcomm/1996/papers/hoe.pdf. *Прим. перев.*
[3]Эта статья доступна по ссылке http://conferences.sigcomm.org/sigcomm/1997/papers/p086.pdf. *Прим. перев.*
[4]Эта статья доступна по ссылке http://www.cs.washington.edu/homes/tom/pubs/CCR99.pdf. *Прим. перев.*

Suite 600
Berkeley, CA 94704-1198
Phone: +1 440 235 1792
EMail: mallman@icir.org
http://www.icir.org/mallman/

Vern Paxson
International Computer Science Institute (ICSI)
1947 Center Street
Suite 600
Berkeley, CA 94704-1198
Phone: +1 510/642-4274 x302
EMail: vern@icir.org
http://www.icir.org/vern/

Ethan Blanton
Purdue University Computer Sciences
305 North University Street
West Lafayette, IN 47907
EMail: eblanton@cs.purdue.edu
http://www.cs.purdue.edu/homes/eblanton/

Перевод на русский язык
Николай Малых
электронная почта: nmalykh@protokols.ru
сайт: http://www.nmalykh.org