Network Working Group
Request for Comments: 2581
Obsoletes: 2001
Category: Standards Track

M. Allman
NASA Glenn/Sterling Software
V. Paxson
ACIRI / ICSI
W. Stevens
Consultant

April 1999

Контроль насыщения в TCP

TCP Congestion Control

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[1] можно найти в документе «Internet Official Protocol Standards» (STD 1). Допускается свободное распространение данного документа.

Авторские права

Тезисы

В этом документе определены 4 связанных между собой алгоритма контроля насыщения[2] для протокола TCP - slow start[3], congestion avoidance[4], fast retransmit[5] и fast recovery[6]. Кроме того, в документе указано, как протоколу TCP следует начинать передачу после сравнительно долгого периода бездействия, а также рассмотрены различные методы генерации подтверждений.

1. Введение

В этом документе даны спецификации четырех алгоритмов контроля насыщения для протокола TCP [Pos81]: 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 [Bra89].

Данный документ является обновлением [Ste97]. В дополнение к спецификациям алгоритмов контроля насыщения документ указывает, как должны себя вести соединения TCP после сравнительно долгого периода бездействия, а также разъясняет некоторые вопросы, касающиеся генерации TCP ACK.

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

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 [MD90], RMSS (см. следующее определение) и других факторов. Размер не включает заголовков и опций TCP/IP.

---

[1] Этот документ частично обновлен RFC 3390, а впоследствии полностью отменен RFC 5681. *Прим. перев.*
[2] При переводе термина “congestion” термин “перегрузка” использовался наряду с термином “насыщение”. *Прим. перев.*
[3] Замедленный старт.
[4] Предотвращение насыщения.
[5] Ускорение повторной передачи.
[6] Ускоренное восстановление.

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 [Bra89]. 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, разрешенных протоколу для передачи. В любой момент для TCP **недопустима** передача данных с порядковыми номерами, превышающими значение суммы наибольшего из подтвержденных 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 еще не подтверждены.

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).

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

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

Начало передачи в сеть с неизвестными условиями требует от TCP достаточно медленной проверки сети с целью определения доступной «емкости» 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.

IW - начальное значение cwnd - **должно** быть не более 2*SMSS байтов и **недопустимо** делать это значение превышающим размер 2 сегментов.

Отметим, что нестандартные экспериментальные реализации TCP **могут** разрешать использование начального окна IW большего размера, как определено в уравнении (1) [AFP98]:

```
IW = min (4*SMSS, max (2*SMSS, 4380 байтов))          (1)
```

В таких случаях отправитель TCP **может** использовать начальное окно размером в 3 или 4 сегмента, если суммарный размер этих сегментов не превышает 4380 байтов. Данный документ **не разрешает** использование таких расширений. Однако далее в этом документе обсуждается использование выражения (1), как направления для экспериментов в части изменений параметров (но не как часть данной спецификации алгоритмов контроля насыщения).

Начальное значение ssthresh **может** быть сколь угодно высоким (например, некоторые реализации используют в качестве порога размер анонсируемого окна), но значение порога может быть уменьшено при возникновении насыщения. 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, подтверждающего доставку новой порции данных. Замедленный старт завершается, когда размер окна насыщения cwnd превышает порог ssthresh (или становится равным этому порогу).

В процессе предотвращения перегрузки размер окна cwnd увеличивается на 1 полноразмерный сегмент за каждый период кругового обхода RTT[2]. Предотвращение насыщения продолжается до тех пор, пока насыщение наблюдается. Для обновления значений cwnd в процессе предотвращения перегрузки обычно используется выражение:

```
cwnd += SMSS*SMSS/cwnd                     (2)
```

Такое увеличение окна выполняется при каждом входящем пакете ACK, не являющемся дубликатом. Выражение (2) обеспечивает допустимое приближение для описанного выше увеличения окна cwnd на 1 полноразмерный сегмент для каждого периода RTT. Отметим, что для соединения, в котором получатель подтверждает прием каждого сегмента

[1]Порог замедленного старта.
[2]Round-trip time.

данных, выражение (2) задает несколько более агрессивный подход, нежели добавление 1 сегмента на каждый период RTT, а для получателей, подтверждающих каждый второй пакет, выражение (2) задает менее агрессивный подход по сравнению с добавлением 1 сегмента.

Примечание для разработчиков: Поскольку в реализациях TCP обычно используется целочисленная арифметика, выражение (2) может не приводить к увеличению размера окна cwnd, когда окно насыщения очень велико (больше, чем SMSS*SMSS). Если выражение (2) дает нулевой результат, его **следует «**округлять» до 1 байта.

Примечание для разработчиков: В старых реализациях используется дополнительная положительная константа в правой части выражения (2). Такой подход некорректен и может вести к снижению производительности [PAD+98].

Другим подходящим способом увеличения окна cwnd в процессе предотвращения перегрузки является подсчет числа байтов новых данных, которые были подтверждены пакетами ACK (недостатком этого метода является необходимость поддержки дополнительной переменной состояния). Когда число подтвержденных байтов достигнет значения cwnd, размер окна cwnd может быть увеличен на величину до SMSS байтов. Отметим, что в процессе предотвращения перегрузки размер окна cwnd **недопустимо** увеличивать более, чем на размер одного полноразмерного сегмента в течение периода RTT или на величину, вычисляемую с помощью выражения 2.

Примечание для разработчиков: Некоторые реализации поддерживают размер окна cwnd в байтах, а другие - в полноразмерных сегментах. В последнем случае использование выражения (2) 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 3:

```
ssthresh = max (FlightSize/2, 2*SMSS)            (3)
```

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

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 **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.

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 экспериментального алгоритма восстановления (loss recovery) типа NewReno [FH98].

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 (4 идентичных подтверждения ACK без доставки между ними каких-либо других пакетов), как индикацию потери сегмента. После получения 3 дубликатов ACK протокол TCP выполняет повторную передачу сегмента, который представляется потерянным, без ожидания завершения отсчета таймера повтора передачи.

После того, как алгоритм быстрого повтора передаст те данные, которые представляются включенными 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 приемном 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. При получении третьего дубликата ACK устанавливается значение ssthresh, которое не превышает значения выражения 3.
2. Повторяется передача потерянного сегмента и устанавливается cwnd = ssthresh + 3*SMSS. Это искусственно «увеличивает» размер окна насыщения на число сегментов (три), которые покинули сеть и буферизованы получателем.
3. Для каждого принятого дополнительного дубликата ACK значение cwnd увеличивается на SMSS. Это искусственно увеличивает окно насыщения для того, чтобы отразить выход из сети дополнительных сегментов.
4. Передается новый сегмент, если это разрешено значениями cwnd и окна, анонсируемого получателем.

[1]Loss window.

5. При получении следующего пакета ACK, подтверждающего новые данные, устанавливается cwnd = ssthresh (значение порога, заданное в п. 1). Это называется «уменьшением» размера окна насыщения.

   Этому пакету ACK следует быть подтверждением, вызванным повтором в п. 1 в течение одного периода RTT после повтора (хотя подтверждение может прийти быстрей при наличии существенного нарушения порядка доставки сегментов данных на приемной 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]. Один из вариантов решения этой проблемы описан в документе [FH98].

4. Дополнительные вопросы

4.1 Перезапуск соединения после бездействия

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.

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 распространен 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 = IW.

Отметим, что нестандартные экспериментальные расширения TCP, определенные в [AFP98], используют RW = min(IW, cwnd), определяя значение IW в соответствии с приведенным выше уравнением (1).

Использование принятого последним сегмента для решения вопроса об уменьшении cwnd не приводит к снижению размера окна насыщения cwnd в распространенном случае для продолжительных соединений HTTP [HTH98]. В таких случаях сервер WWW получает запрос до передачи данных клиенту WWW. Получение запроса ведет 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 приемной стороне **следует** использовать алгоритм задержки подтверждений, описанный в [Bra89]. 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.

Требование, в соответствии с которым пакеты ACK **следует** генерировать по крайней мере для каждого второго полноразмерного пакета, в документе [Bra89] указано в одном месте как **следует**, в в другом – как **должно**. Ввиду неоднозначности мы будет использовать трактовку **следует**. Подчеркнем 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 [PAD+98].

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 передает 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 [Bra89], 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 определен специфический алгоритм, для получателя желательно попытаться предотвратить подобные ситуации (например, путем подтверждения каждого второго сегмента независимо от их размера). Повторим еще 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

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

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 [стр. 42, Pos81][Cla82].

4.3 Механизмы восстановления при потерях

Исследователями TCP было предложено множество алгоритмов восстановления при потере сегментов, повышающих эффективность работы алгоритмов fast retransmit и fast recovery. Некоторые из таких алгоритмов (например, [FF96, MM96a, MM96b]) основаны на использовании опции селективных подтверждений (SACK[1]) [MMFR96], а для других SACK не требуется [Hoe96,FF96,FH98]. 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.

Следовательно, при обнаружении первой потери данных в окне, для ssthresh **должно** быть установлено значение, не превышающее значение выражения (3). 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.

Алгоритмы, описанные в [Hoe96, FF96, MM96a, MM6b], соответствуют принципам четырех базовых алгоритмов контроля насыщения, определенных в данном документе.

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 путем фабрикации избыточных дубликатов подтверждений. Таким способом можно добиться снижения ssthresh до минимального значения 2*SMSS, заставляющего соединение незамедлительно переходить в фазу предотвращения перегрузки с соответствующим снижением скорости передачи данных.

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 путем фабрикации избыточных дубликатов подтверждений или избыточных подтверждений для новых данных. Концептуально такая атака может привести к связанному с насыщением коллапсу для части сети.

6. Изменения по сравнению с RFC 2001

Данный документ существенно отличается по тексту от своего предшественника и задача создания списка различий трудно разрешима. Назначение данного документа заключается не в изменении рекомендаций RFC 2001, а в дальнейшем прояснении ситуаций, которые не были детально рассмотрены в RFC 2001. В частности, данный документ предлагает реализациям TCP способ выхода из сравнительно продолжительного простоя, а также оговаривает и уточняет некоторые вопросы генерации подтверждений TCP ACK. И, наконец, значение начального размера окна насыщения увеличено в данной спецификации с одного сегмента до двух.

Благодарности

Описанные здесь алгоритмы разработал Ван Якобсон (Van Jacobson).

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 включен с разрешения издательства Addison-Wesley.

Neal Cardwell, Sally Floyd, Craig Partridge и Joe Touch внесли множество полезных предложений.

Литература

[AFP98] Allman, M., Floyd, S. and C. Partridge, "Increasing TCP's Initial Window Size, RFC 2414[3], September 1998.

[Bra89] Braden, R., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1122, October 1989.

[Bra97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[Cla82] Clark, D., "Window and Acknowledgment Strategy in TCP", RFC 813, July 1982.

[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.

[FH98] Floyd, S. and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582, April 1999.

[1]TCP selective acknowledgment.
[2]Partial acknowledgment.
[3]Этот документ устарел и заменен RFC 3390. *Прим. перев.*

[Flo94] Floyd, S., "TCP and Successive Fast Retransmits. Technical report", October 1994. ftp://ftp.ee.lbl.gov/papers/fastretrans.ps.

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

[HTH98] Hughes, A., Touch, J. and J. Heidemann, "Issues in TCP Slow-Start Restart After Idle", Work in Progress[1].

[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.

[MD90] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

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

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

[MMFR96] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgement Options", RFC 2018, October 1996.

[PAD+98] 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.

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

[Pos81] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

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

[Ste97] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, January 1997.

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

Адреса авторов

Mark Allman

NASA Glenn Research Center/Sterling Software

Lewis Field

21000 Brookpark Rd. MS 54-2

Cleveland, OH 44135

216-433-6586

EMail: mallman@grc.nasa.gov

roland.grc.nasa.gov/~mallman

Vern Paxson

ACIRI / ICSI

1947 Center Street

Suite 600

Berkeley, CA 94704-1198

Phone: +1 510/642-4274 x302

EMail: vern@aciri.org

W. Richard Stevens

1202 E. Paseo del Zorro

Tucson, AZ 85718

520-297-9416

EMail: rstevens@kohala.com

www.kohala.com/~rstevens

[1]По результатам этой работы не было опубликовано RFC. Черновой вариант документа доступен по ссылке tools.ietf.org/html/draft-ietf-tcpimpl-restart-00. *Прим. перев.*

Перевод на русский язык

Николай Малых

nmalykh@gmail.com

Полное заявление авторских прав