Network Working Group
Request for Comments: 3448
Category: Standards Track

M. Handley
S. Floyd
ICIR
J. Padhye
Microsoft
J. Widmer
University of Mannheim

January 2003

Дружественный к TCP контроль скорости (TFRC) - спецификация протокола

TCP Friendly Rate Control (TFRC): Protocol Specification

Статус документа

В этом документе содержится проект стандартного протокола Internet для сообщества Internet и приглашение к дискуссии в целях развития и совершенствования протокола. Информацию о текущем состоянии стандартизации протокола можно найти в текущей версии документа Internet Official Protocol Standards (STD 1)[1]. Документ может распространяться без ограничений.

Авторские права

Аннотация

В этом документе приведена спецификация протокола TFRC[2], который представляет собой механизм контроля насыщения для потоков с индивидуальной адресацией в среде Internet, обеспечивающей доставку по-возможности[3]. Этот механизм обеспечивает достаточно беспристрастное деление полосы с конкурирующими потоками TCP, но отличается значительно меньшими временными вариациями пропускной способности по сравнению с TCP, что делает этот механизм более подходящим для таких приложений, как телефония или потоковая передача, где важна постоянная скорость потока данных.

Оглавление

[1] В настоящее время действие этого документа отменено RFC 5348. *Прим. перев.*
[2] TCP-Friendly Rate Control.
[3] В оригинале используется термин best efforts. *Прим. перев.*

1. Введение

В этот документе содержится спецификация TFRC, представляющего собой механизм контроля насыщения, разработанный для потоков данных с индивидуальной адресацией в среде Internet, передаваемых одновременно с трафиком TCP [2]. Вместо задания протокола целиком в этом документе дается спецификация механизма контроля насыщения, который может использоваться транспортными протоколами типа RTP [7] в приложениях, включающих сквозной контроль насыщения на уровне приложений, или в контексте контроля насыщения на конечных точках [1]. В документе не рассматриваются форматы пакетов и вопросы надежности. Связанные с реализациями вопросы достаточно кратко рассмотрены в главе 8.

Механизм TFRC разработан для обеспечения разумной беспристрастности распределения полосы при конкуренции с потоками TCP. Разумная беспристрастность означает, что скорость передачи отличается от скорости потока TCP при таких же условиях не более, чем вдвое. Однако TFRC обеспечивает существенно меньшие временные вариации пропускной способности по сравнению с TCP, что делает этот механизм более подходящим для телефонии и потоковых приложений, где относительное постоянство скорости передачи играет важную роль.

Платой за более стабильную пропускную способность по сравнению с TCP в условиях конкуренции за полосу является более медленная реакция TFRC на изменение доступной полосы пропускания. Таким образом, TFRC следует использовать лишь в тех случаях, когда приложениям требуется стабильная пропускная способность и, в частности, предотвращение двукратного снижения скорости передачи, принятого в TCP в ответ на отбрасывание одного пакета. Для приложений, которым просто нужно передать данные за возможно кратчайшее время, рекомендуется использовать TCP или, в тех случаях, когда надежность не требуется, механизм контроля насыщения AIMD[1] с параметрами, близкими к тем, которые применяются в TCP.

Механизм TFRC разработан для приложений, которые используют пакеты фиксированного размера и меняют скорость передачи таких пакетов в ответ на возникновение перегрузок (насыщения). Для некоторых звуковых приложений требуется обеспечение фиксированного интервала времени между передачей последовательных пакетов и варьирование размера пакетов в ответ на возникновение перегрузки. Механизм контроля насыщения, предложенный в этом документе, для таких приложений не подходит, однако эту задачу решает механизм TFRC-PS[2], являющийся вариантом TFRC для приложений с фиксированной частотой передачи и изменением размера пакетов при возникновении насыщения. Спецификация TFRC-PS будет приведена в отдельном документе[3].

Механизм TFRC основан на работе принимающей стороны с расчетом параметров контроля насыщения (частоты потери пакетов) на стороне получателя, а не отправителя. Такой способ хорош для приложений, в которых отправителем является большой сервер, обслуживающий множество одновременных соединений, а получатели имеют достаточно памяти и процессорных ресурсов для выполнения требуемых расчетов. Кроме того, реализованный на приемной стороне механизм лучше подходит для контроля насыщения в системах с групповой адресацией.

2. Терминология

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с BCP 14, RFC 2119 и показывают уровень требований к реализациям в части соответствия TFRC.

3. Механизм протокола

Для контроля насыщения TFRC напрямую использует уравнение пропускной способности для разрешенной скорости передачи, как функции от частоты фактов потери пакетов и времени кругового обхода. Для беспристрастной конкуренции с TCP, механизм TFRC использует принятое в TCP уравнение для пропускной способности, выражающее скорость передачи TCP, как функцию от частоты фактов потери пакетов, времени кругового обхода и размера сегментов. Определим факт потери, как случай утраты одного или более маркированного пакета из окна данных; маркированными считаются пакеты, помеченные явным индикатором насыщения ECN[4] [6].

В общем виде механизм контроля насыщения TFRC можно описать следующим образом:

- Получатель определяет частоту фактов потери пакетов и передает эту информацию отправителю.
- Отправитель использует эти сообщения от получателя для определения времени кругового обхода (RTT[5]).
- Значения частоты фактов потери и RTT передаются в уравнение пропускной способности TFRC и результирующая скорость передачи ограничена значением не более удвоенной скорости приема.
- Отправитель подстраивает скорость передачи в соответствии с допустимой скоростью передачи X.

Динамика TFRC чувствительна к способам проведения измерений и применения результатов. В этом документе приводятся рекомендации по конкретному механизму. Возможно использование иных механизмов, но при этом следует понимать, как этот механизм будет воздействовать на динамику TFRC.

3.1. Уравнение для пропускной способности TCP

Любое реалистичное выражение пропускной способности TCP, как функции RTT и вероятности потери пакетов, следует рассматривать, как подходящее для использования с TFRC. Однако следует отметить, что используемое для пропускной способности TCP уравнение должно отражать поведение повторов передачи по тайм-ауту, поскольку это поведение доминирует при определении пропускной способности TCP в условиях высокой вероятности потерь. Отметим также, что допущения, принятые относительно вероятности потерь в уравнении для пропускной способности, зависят от реального механизма определения вероятности потерь. Хотя это допущение не вполне соответствует

[1] Additive-Increase, Multiplicative-Decrease - аддитивный рост, мультипликативное снижение.
[2] TFRC-PacketSize.
[3] Спецификация TFRC-PS опубликована в RFC 4828. *Прим. перев.*
[4] Explicit Congestion Notification.
[5] Round-trip time.

приведенному ниже уравнению для пропускной способности и описанным механизмам измерения, оно достаточно хорошо подходит на практике.

Уравнение пропускной способности, которое мы рекомендуем для использования в TFRC, является слегка упрощенным вариантом уравнения для Reno TCP из работы [4]. В идеальном случае уравнение для пропускной способности следовало бы создавать на основе SACK[1] TCP, однако тесты и эксперименты показывают, что различия между двумя уравнениями достаточно малы.

Пропускная способность выражается уравнением

$$X = \frac{s}{R\sqrt{2bp/3} + t_{RTO}\left(3\sqrt{3bp/8}\right) p \left(1+32p^2\right)}$$

где:

X - скорость передачи (байт/сек);

s - размер пакетов (байт);

R - время кругового обхода в секундах;

p - вероятность потерь (0 - 1.0) - доля потерянных пакетов в от общего числа переданных пакетов;

t_RTO - тайм-аут повторной передачи TCP в секундах;

b - число пакетов, подтверждаемых одним пакетом TCP ACK.

Упростим это выражение, приняв t_RTO = 4*R. Возможен более точный расчет t_RTO, однако эксперименты с выбранным значением показали достаточно беспристрастное деление полосы с существующими реализациями TCP [9]. Другим возможным вариантом является выбор для t_RTO большего из 2 значений (4R, 1 секунда) в соответствии с рекомендациями по установке для RTO значения не меньше 1 секунды [5].

Многие соединения TCP используют режим отложенных подтверждений, когда пакет подтверждения передается для каждого второго принятого пакета (b = 2). Однако TCP позволяет передавать подтверждения для каждого пакета (b = 1). Поскольку множество реализаций TCP не использует режима отложенных подтверждений, рекомендуется использовать b = 1.

В будущем для приведенного выше уравнения могут использоваться иные параметры TCP. Однако уравнение для пропускной способности в любом случае должно достоверно описывать скорость передачи TCP для обеспечения соответствия механизмам контроля насыщения TCP.

Параметры s (размер пакета), p (вероятность потери) и R (RTT) должны измеряться или рассчитываться реализацией TFRC. Измерение s рассматривается в параграфе 4.1, измерение R - в параграфе 4.3, а измерение p - в разделе 5. Далее в этом документе все значения скоростей приводятся в байтах/сек.

3.2. Содержимое пакетов

Прежде, чем описывать функциональность отправителя и получателя, рассмотрим содержимое пакетов данных, передаваемых отправителем, и пакетов обратной связи, передаваемых получателем. Мы не будем задавать формат пакетов, поскольку TFRC будет использоваться с протоколом транспортного уровня, который и определяет этот формат.

3.2.1. Пакеты данных

Каждый пакет данных, передаваемый отправителем, содержит следующую информацию:

- Порядковый номер. Этот номер увеличивается на 1 с каждым переданным пакетом. Поле должно быть достаточно большим, чтобы в списке недавних пакетов получателя не появлялись разные пакеты с одинаковым порядковым номером.

- Временная метка момента передачи. Будем обозначать ts_i временную метку пакета с порядковым номером i. Разрешение для временных меток обычно измеряется миллисекундами. Эти временные метки используются получателем для определения потерь пакетов, которые следует отнести к одному событию (факту потери). Эти метки получатель возвращает отправителю в качестве «эхо» для того, чтобы тот мог оценить время кругового обхода (это нужно для отправителей, не сохраняющих временных меток переданных пакетов). Отметим, что существует альтернативный вариант использования временных меток, когда значение метки инкрементируется каждую четверть времени кругового обхода; такой точности достаточно для отнесения потерь пакетов к одному событию в контексте протокола, где это понятно как отправителю, так и получателю, а отправитель сохраняет временные метки переданных пакетов.

- Оценка времени кругового обхода отправителем. Оценка, передаваемая в пакете i обозначается R_i. Оценка времени кругового обхода используется получателем вместе с временными метками для определения множества потерь пакетов, относящихся к одному событию. Если отправитель передает передает грубые «временные метки», которые увеличиваются каждую четверть периода кругового обхода, как описано выше, такому отправителю не нужно передавать свою оценку времени кругового обхода.

3.2.2. Пакеты обратной связи

Каждый пакет обратной связи, передаваемый получателем данных, содержит следующую информацию:

- Временная метка последнего принятого пакета (t_recvdata). Если последний принятый пакет имеет номер i, то t_recvdata = ts_i. Эта временная метка используется отправителем для оценки времени кругового обхода и требуется только в тех случаях, когда отправитель не сохраняет временные метки переданных пакетов данных.

[1]Selective acknowledgment - селективное подтверждение.

- Интервал времени между приемом последнего пакета и генерацией данного пакета обратной связи. Будем обозначать этот интервал t_delay.
- Оценка получателем скорости приема данных с момента отправки последнего пакета обратной связи. Будем обозначать этот интервал X_recv.
- Текущее значение вероятности потери по оценке получателя (p).

4. Протокол отправителя данных

Отправитель шлет получателю поток пакетов данных с определенной скоростью. При получении пакета обратной связи от получателя, отправитель данных меняет скорость передачи на основе информации, содержащейся в таком пакете. Если отправитель не получает пакетов обратной связи в течение двух интервалов кругового обхода, он снижает скорость передачи вдвое. Для контроля времени используется таймер обратной связи[1].

Для протокола на передающей стороне зададим следующие этапы:

- измерение среднего размера передаваемого пакета;
- реакция отправителя на получение пакета обратной связи;
- поведение отправителя по завершении отсчета таймера обратной связи;
- предотвращение осцилляций (опционально);
- планирование передачи в операционных системах, не работающих в режиме реального времени.

4.1. Измерение размера пакетов

Параметр s (размер пакета) обычно известен приложению, но могут быть два исключительных случая:

- Размер пакетов данных меняется естественным образом в зависимости от данных. В этом случае, хотя размер пакетов меняется, его вариации не отражаются на скорости передачи. Обычно можно без опаски использовать средний размер пакета в качестве s.
- Для контроля насыщения приложение может менять размер пакетов, а не скорость их передачи. Это обычная практика для аудио-приложений, в которых пакеты данных передаются с фиксированным интервалом, требуемым для представления каждого пакета. Для таких приложений требуется совершенно иной способ измерения параметров.

Второй класс приложений рассматривается отдельно в документе, посвященном TFRC-PS. Оставшаяся часть этого раздела посвящена способам оценки размера пакетов отправителем и организации контроля насыщения путем управления числом пакетов, передаваемых за секунду.

4.2. Инициализация отправителя

Для инициализации отправителя устанавливается скорость передачи X = 1 пакет/сек и таймер обратной связи - 2 секунды. Начальные значения R (RTT) и t_RTO остаются неопределенными, пока не будут установлены в соответствии с приведенными ниже рекомендациями. Начальное значение tld[2] при замедленном старте устанавливается равным -1.

4.3. Поведение отправителя при получении пакета обратной связи

Отправитель знает свою текущую скорость передачи X и поддерживает оценку текущего значения времени кругового обхода R, а также оценивает значение тайм-аута повторной передачи t_RTO.

При получении отправителем пакета обратной связи в момент t_now выполняются следующие действия:

1) Рассчитывается новое значение времени кругового обхода:

```
R_sample = (t_now - t_recvdata) - t_delay
```

2) Обновляется оценка времени кругового обхода:

```
Если ранее не было получено пакета обратной связи
    R = R_sample;
иначе
    R = q*R + (1-q)*R_sample;
```

Алгоритм TFRC не чувствителен к точному значению константы q, но рекомендуется задавать значение 0.9.

3) Обновляется значение тайм-аута:

```
t_RTO = 4*R
```

4) Обновляется значение скорости передачи:

```
If (p > 0)
    ;X_calc рассчитывается с использованием уравнения для пропускной способности TCP.
    X = max(min(X_calc, 2*X_recv), s/t_mbi);
Else
    If (t_now - tld >= R)
        X = max(min(2*X, 2*X_recv), s/R);
        tld = t_now;
```

Отметим, что при p == 0 отправитель находится в фазе замедленного старта, в которой он приблизительно удваивает скорость передачи в течение каждого периода кругового обхода, если не наблюдается фактов потери. Значение s/R дает минимальную скорость передачи в процессе замедленного старта - 1 пакет за время RTT. Параметр t_mbi имеет значение 64 секунды и показывает максимальный интервал между

[1] В оригинале используется термин nofeedback timer. *Прим. перев.*
[2] Time Last Doubled - время последнего удвоения.

передачей пакетов[3] при сохраняющемся отсутствии пакетов обратной связи. Таким образом, при p > 0 отправитель передает по крайней мере каждые 64 секунды.

5) Сбрасывается таймер обратной связи по истечении max(4*R, 2*s/X) секунд.

4.4. Завершение отсчета таймера обратной связи

Если отсчет таймера обратной связи завершился, отправителю следует выполнить следующие операции:

1) Снизить скорость передачи вдвое. Если отправитель принимал от получателя пакеты обратной связи снижение осуществляется путем изменения кэшированной отправителем копии X_recv (скорость приема). Поскольку скорость передачи ограничена значением, не превышающим 2 * X_recv, изменение X_recv будет ограничивать текущую скорость передачи, но позволит отправителю использовать замедленный старт, удваивая скорость передачи через каждый интервал RTT, если пакеты обратной связи будут говорить об отсутствии потерь.

```
If (X_calc > 2*X_recv)
    X_recv = max(X_recv/2, s/(2*t_mbi));
Else
    X_recv = X_calc/4;
```

Значение s/(2*t_mbi) не позволяет снижение скорости передачи до значений менее 1 пакета за 64 секунды в случаях постоянного отсутствия пакетов обратной связи.

2) После этого должно быть пересчитано значение X в соответствии с п 4 в предыдущем параграфе.

Если отсчет таймера обратной связи завершается, когда у отправителя еще нет образца RTT и он не получал еще пакетов обратной связи, этап 1) можно пропустить и напрямую снизить скорость передачи вдвое:

```
X = max(X/2, s/t_mbi)
```

3) Перезапустить таймер обратной связи по истечении max(4*R, 2*s/X) секунд.

Отметим, что прекращение передачи данных отправителем вызывает прекращение отправки получателем пакетов обратной связи. Это будет вызывать запуск таймера обратной связи и снижение X_recv по истечении отсчета таймера. Если отправитель впоследствии снова начнет передачу данных, значение X_recv будет ограничивать скорость передачи и будет выполняться обычная процедура замедленного старта, пока скорость передачи не достигнет значения X_calc.

Если отправитель бездействует с момента запуска таймера обратной связи и значение X_recv меньше 4 пакетов за время кругового обхода, значение X_recv не следует уменьшать вдвое по завершении отсчета таймера. Это позволяет никогда не снижать скорость передачи менее 2 пакетов за время кругового обхода в результате бездействия.

4.5. Предотвращение осцилляций

Для предотвращения осцилляций в средах с низким уровнем статистического мультиплексирования полезно изменить скорость передачи данных отправителем, чтобы предотвратить насыщение за счет снижения скорости по мере роста задержки в очередях (и, следовательно, RTT). Для реализации этого отправитель поддерживает оценку среднего значения RTT за достаточно большой период и меняет свою скорость передачи в зависимости от того, как недавнее значение RTT отличается от среднего. Среднее значение R_sqmean определяется следующим образом:

```
Если ранее не было получено пакетов обратной связи
    R_sqmean = sqrt(R_sample);
иначе
    R_sqmean = q2*R_sqmean + (1-q2)*sqrt(R_sample);
```

Таким образом, R_sqmean изменяется пропорционально квадратному корню измеренного значения RTT. Константу q2 следует устанавливать по аналогии с q; по умолчанию рекомендуется использовать значение 0,9.

Отправитель получает базовое значение скорости передачи X из уравнения пропускной способности. После этого отправитель рассчитывает новое значение скорости передачи X_inst следующим образом:

```
X_inst = X * R_sqmean / sqrt(R_sample);
```

Когда sqrt(R_sample) больше R_sqmean, очередь обычно увеличивается и для стабильной работы требуется снижение скорости передачи.

Примечание. Такое изменение требуется не во всех случаях, особенно при высоком уровне статистического мультиплексирования в сети. Однако рекомендуется вносить это изменение поскольку оно улучшает поведение TFRC в средах с низким уровнем статистического мультиплексирования. Если это изменение не выполняется, рекомендуется использовать очень малые значения q (0 или близкие к нулю значения).

4.6. Планирование передачи пакетов

Поскольку TFRC работает на основе скорости, а операционные системы обычно не способны точно планировать события, необходимо с осторожностью относиться к передаче данных, чтобы поддерживалось корректное среднее значение скорости, несмотря на грубое или нерегулярное планирование в операционной системе. Таким образом, для типичного цикла передачи корректный интервал между передачей пакетов t_ipi будет определяться следующим образом:

```
t_ipi = s/X_inst;
```

Когда отправитель начинает передачу в момент t_0, он рассчитывает значение t_ipi и номинальное время передачи пакета 1 будет t_1 = t_0 + t_ipi. Когда приложение простаивает, оно проверяет текущее значение t_now и запрашивает повторный расчет интервала по истечении t_ipi - (t_now - t_0) секунд. Когда приложение снова планирует передачу, оно опять проверяет текущее значение t_now. Если t_now > (t_1 - delta), пакет 1 передается.

[3] Inter-packet backoff interval.

В этот момент рассчитывается новое значение t_ipi, которое применяется для расчета времени передачи t_2 для пакета 2 - t2 = t_1 + t_ipi. Этот процесс повторяется для каждого пакета с отсчетом времени от номинального момента передачи предыдущего пакета.

В некоторых случаях, когда номинальное время передачи следующего пакета t_i рассчитано, может оказаться, что t_now > t_i - delta. В таких случаях пакет следует передавать без промедления. Таким образом, если операционная система обеспечивает лишь грубую гранулярность таймеров и скорость передачи высока, TFRC может передавать короткие группы пакетов, разделенные интервалами с гранулярностью таймера операционной системы.

Параметр delta служит для обеспечения достаточной гибкости при выборе времени передачи пакетов. Если операционная система имеет гранулярность таймера планирования t_gran секунд, значение delta обычно следует устанавливать равным:

```
delta = min(t_ipi/2, t_gran/2);
```

t_gran равно 10 мсек для многих Unix-систем. Если значение t_неизвестно, обычно можно без опасений принимать его равным 10 мсек.

5. Расчет частоты потерь (p)

Точное и стабильное измерение вероятности потерь имеет первоочередное значение для TFRC. Измерение частоты потери пакетов осуществляется на приемной стороне путем детектирования потери пакетов по порядковым номерам прибывающих пакетов или при получении маркированного пакета. Опишем этот процесс прежде, чем разбираться с остальной частью приемного протокола.

5.1. Детектирование потерь и маркированные пакеты

TFRC предполагает, что в каждом пакете содержится порядковый номер и эти номера увеличиваются на 1 в каждом переданном пакете. Данная спецификация требует, чтобы при повторе передачи потерянного пакета использовался новый порядковый номер в соответствии с последовательностью нумерации. Если транспортный протокол требует при повторе передачи использовать исходный порядковый номер, разработчики транспортного протокола должны обеспечить механизм, позволяющий отличить задержанные пакеты от переданных повторно, и механизм детектирования потери пакетов при повторе передачи.

Получатель поддерживает структуру данных, в которой хранится информация о полученных и пропущенных пакетах. В данной спецификации предполагается, что эта структура представляет собой список полученных пакетов и временных меток момента приема каждого такого пакета. На практике такая структура может использовать более компактное представление, выбранное разработчиками.

Потеря пакета детектируется по факту прибытия по крайней мере трех пакетов с большими порядковыми номерами. Значение три выбрано по аналогии с TCP и обеспечивает TFRC устойчивость к нарушению порядка доставки пакетов. В отличие от TCP, если пакет приходит позднее (после доставки 3 следовавших за ним пакетов), информация об этом пакете может заполнить пробел в записях TFRC и получатель сможет пересчитать вероятность потерь. В будущих версиях TFRC значение 3 для детектирования потери может быть заменено адаптивным значением, учитывающим реальное нарушение порядка доставки, но в данной спецификации механизм такого учета не рассматривается.

Для соединений, поддерживающих ECN прибытие маркированных пакетов трактуется как факт насыщения без ожидания доставки последующих пакетов.

5.2. Трансляция истории потерь в факт потери

TFRC требует устойчивости к нескольким последовательным потерям пакетов, когда эти потери относятся к одному событию (факту потери). Это похоже на поведение протокола TCP, который (обычно) лишь однократно за период RTT уменьшает наполовину окно насыщения. Таким образом, получатель должен отобразить историю потери пакетов в запись о факте потери, трактуя как факт потери утрату одного или более пакетов в течение периода RTT. Для выполнения такого отображения получателю нужно знать значение RTT, которое обычно периодически сообщается отправителем в форме управляющей информации, присоединяемой в конце пакета данных. Для TFRC способ передачи результатов измерения RTT получателю не имеет значения, однако рекомендуется использовать для этого рассчитанное отправителем значение RTT (R в параграфе 4.3).

Для того, чтобы определить, относится потеря или маркированный пакет к новому факту потери или является продолжением существующего, нужно сравнить порядковые номера и временные метки пакетов, принятых получателем. Для маркированного пакета S_new время приема T_new можно определить напрямую. Для потерянного пакета нужно использовать интерполяцию, чтобы определить номинальное «время прибытия». Предположим, что:

S_loss - порядковый номер потерянного пакета;

S_before - порядковый номер последнего пакета, прибывшего с номером меньше S_loss;

S_after - порядковый номер первого пакета, прибывшего с номером больше S_loss;

T_before - время приема S_before;

T_after - время приема S_after.

Отметим, что значение T_before может превышать значение T_after в результате нарушения порядка доставки.

Для потерянного пакета S_loss можно интерполировать его номинальное «время доставки» на основе значений S_before и S_after:

```
T_loss = T_before + ((T_after - T_before) * (S_loss - S_before)/(S_after - S_before));
```

Отметим, что в тех случаях, когда между порядковыми номерами S_before и S_after наблюдался переход через 0, порядковые номера следует изменить с учетом этого факта до выполнения расчетов. Если больший порядковый номер имеет значение S_max b S_before > S_after, замены каждого номера S на S' = (S + (S_max + 1)/2) mod (S_max + 1) будет достаточно.

Если было определено, что потерянный пакет S_old служит началом нового факта потери и мы определили, что пакет S_new был потерян, мы интерполируем номинальное время прибытия пакетов S_old и S_new, как T_old и T_new, соответственно.

Если T_old + R >= T_new, потеря пакета S_new относится к текущему факту потери, в противном случае S_new является первым пакетом, относящимся к новому факту.

5.3. Интервал между потерями

Если интервал между потерями A определен, как начинающийся с пакета S_A, а интервал B - с пакета S_B, число пакетов в интервале между потерями A составляет (S_B - S_A).

5.4. Средний интервал между потерями

Для расчета частоты потерь p сначала вычисли продолжительность среднего интервала между потерями. Это осуществляется с помощью взвешенного фильтра, определяющего средний интервал на основе значений n последних интервалов.

Веса w_0 - w_(n-1) определяются следующим образом:

```
If (i < n/2)
   w_i = 1;
Else
   w_i = 1 - (i - (n/2 - 1))/(n/2 + 1);
```

Таким образом, для n=8, значения весов w_0 - w_7 составят:

```
1.0, 1.0, 1.0, 1.0, 0.8, 0.6, 0.4, 0.2
```

Значение числа интервалов n используется при расчете частоты фактов потерь, определяющей скорость реакции TFRC на изменение уровня насыщения. В соответствии с настоящей спецификацией TFRC не следует использовать со значениями n, существенно превышающими 8, для трафика, который может перемешиваться в сети Internet с трафиком TCP. В крайнем случае при использовании значений n > 8 потребуется некоторое изменение механизмов TFRC для обеспечения более резкого отклика на значительную потерю пакетов в течение 2 и более периодов кругового обхода.

При расчете среднего интервала между потерями требуется решить вопрос о включении последнего интервала. Предлагается включать его только в тех случаях, когда он существенно увеличивает значение среднего интервала между потерями.

Таким образом, если последние интервалы между потерями обозначить от I_0 до I_n и I_0 будет относиться к последнему факту потерь, средний интервал рассчитывается следующим образом:

```
I_tot0 = 0;
I_tot1 = 0;
W_tot = 0;
for (i = 0 to n-1) {
  I_tot0 = I_tot0 + (I_i * w_i);
  W_tot = W_tot + w_i;
}
for (i = 1 to n) {
  I_tot1 = I_tot1 + (I_i * w_(i-1));
}
I_tot = max(I_tot0, I_tot1);
I_mean = I_tot/W_tot;
```

Вероятность потерь p будет равна:

```
p = 1/I_mean;
```

5.5. Дисконтирование истории

Как было показано в параграфе 5.4, последний интервал между потерями дает 1/(0.75*n) часть общего веса при расчете среднего интервала, независимо от продолжительности этого интервала. В этом параграфе описан дополнительный механизм «дисконтирования истории»[1], рассмотренный в работах [3] и [9], который позволяет приемному узлу TFRC подбирать весовые параметры, придавая больший вес последнему интервалу между потерями, когда этот интервал более, чем вдвое превышает рассчитанное значение среднего интервала.

Для дисконтирования истории свяжем коэффициент DF_i (число с плавающей запятой) с каждым интервалом L_i (для i > 0). Общая история дисконтирования для каждого интервала между потерями будет храниться в массиве коэффициентов. В начальный момент значения элементов массива DF_i устанавливаются в 1:

```
for (i = 1 to n) {
  DF_i = 1;
}
```

Дисконтирование истории также использует общий коэффициент DF (число с плавающей запятой), который также имеет начальное значение 1. Сначала посмотрим, как коэффициенты используются при расчете среднего интервала между потерями, а затем опишем изменение коэффициентов с течением времени.

Как описано в параграфе 5.4, средний интервал между потерями вычисляется с использованием n значений предыдущих интервалов I_1, ..., I_n и значения I_0, которое представляет собой число пакетов, полученных с момента последнего факта потерь. Расчет среднего интервала с использованием коэффициентов дисконтирования незначительно отличается от процедуры, описанной в параграфе 5.4:

```
I_tot0 = I_0 * w_0
I_tot1 = 0;
W_tot0 = w_0
W_tot1 = 0;
```

[1]History discounting mechanism.

```
for (i = 1 to n-1) {
  I_tot0 = I_tot0 + (I_i * w_i * DF_i * DF);
  W_tot0 = W_tot0 + w_i * DF_i * DF;
}
for (i = 1 to n) {
  I_tot1 = I_tot1 + (I_i * w_(i-1) * DF_i);
  W_tot1 = W_tot1 + w_(i-1) * DF_i;
}
p = min(W_tot0/I_tot0, W_tot1/I_tot1);
```

Общий коэффициент дисконтирования DF обновляется при получении каждого пакета. Сначала получатель рассчитывает средневзвешенное значение I_mean для интервалов потерь I_1, ..., I_n:

```
I_tot = 0;
W_tot = 0;
for (i = 1 to n) {
  W_tot = W_tot + w_(i-1) * DF_i;
  I_tot = I_tot + (I_i * w_(i-1) * DF_i);
}
I_mean = I_tot / W_tot;
```

Значение I_mean сравнивается с числом пакетов, полученных с момента последнего факта потери - I_0. Если I_0 превышает I_mean более, чем вдвое, это говорит о том, что новый интервал потерь существенно превышает старые значения и значение общего коэффициента DF изменяется для снижения относительного веся более старых интервалов:

```
if (I_0 > 2 * I_mean) {
  DF = 2 * I_mean/I_0;
  if (DF < THRESHOLD)
    DF = THRESHOLD;
} else
  DF = 1;
```

Отличное от 0 значение порога THRESHOLD обеспечивает гарантию того, что информация о более ранних интервалах в периоды высокого насыщения не будет полностью обесценена. Рекомендуется устанавливать THRESHOLD = 0.5. Отметим, что прибытие каждого нового пакета ведет к дополнительному росту I_0 и коэффициент DF будет обновляться.

При новом факте потерь текущий интервал переходит из I_0 в I_1, интервал I_i - в I_(i+1), а интервал I_n отбрасывается. Предыдущий коэффициент DF включается в массив коэффициентов дисконтирования. Поскольку DF_i, показывает коэффициент, связанный с интервалом I_i, значения DF_i в массиве также смещаются при новом факте потерь. Процедура сдвига имеет вид:

```
for (i = 1 to n) {
  DF_i = DF * DF_i;
}
for (i = n-1 to 0 step -1) {
  DF_(i+1) = DF_i;
}
I_0 = 1;
DF_0 = 1;
DF = 1;
```

На этом описание дополнительного механизма дисконтирования истории заканчивается. Подчеркнем что этот механизм является необязательным и позволяет TFRC более быстро реагировать на стремительное прекращение перегрузок, демонстрируемое ростом интервала между потерями.

6. Протокол получателя данных

Получатель периодически направляет отправителю сообщения обратной связи. Пакеты обратной связи в обычных условиях следует передавать по крайней мере по одному за период RTT, если отправитель не передает менее 1 сообщения за период RTT (в последнем случае пакеты обратной связи следует передавать в ответ на каждый принятый пакет). Пакеты обратной связи следует также передавать при новых фактах потерь без ожидания завершения цикла RTT и при получении пакетов с нарушением порядка доставки, когда это ведет к удалению факта потерь из истории.

Если отправитель передает пакеты с высокой скоростью (много пакетов за период RTT) передача пакетов обратной связи несколько раз в течение RTT может обеспечивать преимущества, поскольку позволит быстрее реагировать на изменение результатов измерения RTT и повысит устойчивость к потере пакетов обратной связи. Однако эти преимущества с ростом числа пакетов обратной связи за период RTT растут достаточно медленно.

6.1. Поведение получателя при приеме пакета данных

При получении пакета данных получатель выполняет следующие действия:

1) добавляет пакет в список принятых (историю);

2) сохраняет прежнее значение p, как p_prev и рассчитывает новое значение, как описано в разделе 5;

3) если p > p_prev, таймер обратной связи считается истекшим и выполняются действия, описанные в параграфе 6.2;

   при p <= p_prev никаких действий не предпринимается.

   Однако для оптимизации может потребоваться проверка заполнения пропуска в номерах принятых пакетов и при обнаружении такового объединение двух интервалов между потерями в один. В последнем случае получатель может также незамедлительно отправить сообщение обратной связи. В обычных условиях влияние такой оптимизации незначительно.

6.2. Завершение отсчета таймера обратной связи

При завершении отсчета таймера обратной связи на принимающей стороне должны выполняться определенные действия в зависимости от наличия или отсутствия пакетов, принятых с момента отправки последнего сообщения обратной связи.

Предположим, что на приемной стороне максимальный номер полученного пакета имеет значение S_m, а включенный в этот пакет результат измерения RTT имеет значение R_m. Если с момента отправки предыдущего сообщения обратной связи были получены пакеты данных, получатель выполняет следующие операции:

1) расчет среднего интервала между потерями с использованием описанной выше процедуры;
2) расчет измеренного значения скорости приема X_recv на основе пакетов, принятых в течение предшествующих R_m секунд;
3) подготовка и передача пакета обратной связи, содержащего информацию, описанную в параграфе 3.2.2;
4) сброс и повторный запуск таймера обратной связи на R_m секунд.

Если с момента отправки последнего сообщения обратной связи не было принято пакетов данных, сообщение обратной связи не передается, таймер обратной связи сбрасывается и повторно запускается на R_m секунд.

6.3. Инициализация приемника

Приемник инициализируется первым доставленным ему пакетом. Предположим, что этот пакет имеет порядковый номер i.

При получении первого пакета:

- устанавливается p=0;
- устанавливается X_recv = 0;
- подготавливается и передается пакет обратной связи;
- устанавливается таймер обратной связи на R_i секунд.

6.3.1. Инициализация истории потерь после первого факта потери

Пока не произойдет первая потеря, число пакетов не может напрямую использоваться для расчета скорости передачи и скорость передачи передачи в течение этого периода может быстро изменяться. TFRC предполагает, что корректная скорость после первой потери составляет половину скорости передачи на момент возникновения этой потери. TFRC аппроксимирует эту скорость значением X_recv (скорость приема в течение последнего периода кругового обхода). После первой потери взамен инициализации первого интервала между потерями числом принятых до потери пакетов, приемный узел TFRC рассчитывает интервал между потерями, который будет требоваться для передачи данных со скоростью X_recv и использует это значение для передачи механизму истории потерь.

TFRC делает это путем нахождения некого значения p для которого уравнение пропускной способности из параграфа 3.1 дает скорость передачи, отклоняющуюся от X_recv в пределах 5%, для текущего размера пакетов s и периода кругового обхода R. Для первого интервала между потерями устанавливается значение 1/p (5% погрешность допускается потому, что уравнение пропускной способности сложно обратить и без погрешности пришлось бы рассчитывать p численными методами).

7. Серверные варианты

Возможна реализация механизма TFRC на серверной стороне, когда получатель использует средства гарантированной доставки отправителю информации о потере пакетов, а отправитель рассчитывает вероятность потери и подходящую скорость передачи. Однако в этом документе не задается спецификация деталей серверного варианта.

Основным преимуществом серверной реализации TFRC является то, что отправитель не должен доверять расчетам частоты потерь на стороне получателя. Однако требование гарантированной доставки информации о потерях от получателя к отправителю вносит существенные ограничения в процесс выбора транспортного протокола для поддержки серверных вариантов TFRC.

Вариант TFRC, реализованный на приемной стороне в соответствии с данной спецификацией, напротив, не требует гарантированной доставки пакетов обратной связи. Этот вариант также лучше подходит для таких приложений, как потоковые службы на web-серверах, для которых желателен максимальный перенос нагрузки с серверной стороны на клиентскую.

Варианты реализации механизма на приемной и передающей стороне отличаются также с точки зрения обновлений. Например, для изменения процедуры расчета частоты потерь в серверном варианте потребуется обновление сервера, а при реализации на приемной стороне - обновление клиентов.

8. Вопросы реализации

В этом документе приведена спецификация механизма контроля насыщения TFRC для прикладных и транспортных протоколов. В данном разделе кратко рассматриваются некоторые вопросы реализации механизма.

Для t_RTO = 4*R и b = 1 уравнение пропускной способности из параграфа 3.1 можно записать в виде:

$$X = \frac{s}{R * f(p)}$$

где

$$f(p) = \sqrt{2p/3} + (12\sqrt{3p/8} * p * (1+32p^2)).$$

Вместо вычисления значений функции f(p) можно воспользоваться таблицей заранее подсчитанных значений.

Многие операции умножения (например, q и 1-q для расчета среднего времени кругового обхода, умножение на 4 для тайм-аута) могут быть реализованы с помощью операций сдвига регистра.

Отметим, что дополнительный механизм предотвращения осцилляций, описанный в параграфе 4.5, использует расчет квадратного корня.

Расчет среднего интервала между потерями в параграфе 5.4 включает умножение на весовые коэффициенты w_0 - w_(n-1), которые для n=8 имеют значения:

```
1.0, 1.0, 1.0, 1.0, 0.8, 0.6, 0.4, 0.2.
```

С незначительной потерей точности можно использовать в качестве весовых коэффициентов значения степеней числа два или суммы таких значений, например:

```
1.0, 1.0, 1.0, 1.0, 0.75, 0.5, 0.25, 0.25.
```

Дополнительный механизм дисконтирования истории, описанный в параграфе 5.5, используется при расчете среднего интервала между потерями. Этот механизм предназначен для использования лишь в тех случаях, когда между фактами потери пакетов возникает необычно большой интервал. Для более эффективной работы механизма можно ограничить выбор значений коэффициентов DF_i степенями числа 2.

9. Вопросы безопасности

TFRC является не транспортным протоколом, а механизмом контроля насыщения, предназначенным для использования с такими протоколами. Следовательно, обсуждение вопросов безопасности должно происходить в контексте соответствующего транспортного протокола и его механизмов аутентификации.

Механизмы контроля насыщения потенциально могут использоваться для организации атак на отказ служб. Такая атака может быть реализована путем передачи ложных сообщений обратной связи. Поэтому транспортным протоколам, использующим TFRC, следует принимать меры по защите от приема фальсифицированных пакетов обратной связи. Точные механизмы такой защиты зависят от выбранного транспортного протокола.

Кроме того, механизм контроля насыщения может использоваться «жадными» получателями, которые хотят получать данных больше, нежели позволяет беспристрастное деление полосы. Получатель может предпринять такую попытку за счет передачи серверу обманной информации о приёме пакетов, которые реально были потеряны в результате насыщения. Возможной защитой от такого поведения является включение той или иной формы специальных сигналов (nonce), которые получатель должен возвращать отправителю для подтверждения приема. Однако детали такой защиты зависят от наличия гарантий доставки пакетов на уровне транспортного протокола.

Предполагается, что протоколы, использующие ECN[1] с TFRC будут также поддерживать обратную связь от получателя с использованием ECN nonce [WES02]. ECN nonce представляет собой модификацию ECN с защитой отправителя от нечаянного или злонамеренного сокрытия маркированных пакетов. Однако детали использования таких механизмов зависят от транспортного протокола и не рассматриваются в этом документе.

10. Взаимодействие с IANA

Этот документ не требует согласования с IANA.

11. Благодарности

Мы благодарим за отклики и конструктивные предложения по развитию механизма контроля насыщения на основе уравнения пропускной способности широкий круг людей, включая членов исследовательской группы Reliable Multicast, рабочей группы Reliable Multicast Transport и исследовательской группы End-to-End. Выражаем благодарность также Ken Lofgren, Mike Luby, Eduardo Urzaiz, Vladica Stanisic, Randall Stewart, Shushan Wen и Wendy Lee (lhh@zsu.edu.cn) за отклики к ранней версии этого документа и благодарим Mark Allman за множество откликов по поводу использования документа для создания работоспособных реализаций механизма.

12. Нормативные документы

[1] Balakrishnan, H., Rahul, H., and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts,"[2] Proc. ACM SIGCOMM, Cambridge, MA, September 1999.

[2] Floyd, S., Handley, M., Padhye, J. and J. Widmer, "Equation-Based Congestion Control for Unicast Applications"[3], August 2000, Proc. ACM SIGCOMM 2000.

[3] Floyd, S., Handley, M., Padhye, J. and J. Widmer, "Equation-Based Congestion Control for Unicast Applications: the Extended Version"[4], ICSI tech report TR-00-03, March 2000.

[4] Padhye, J., Firoiu, V., Towsley, D. and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation"[5], Proc. ACM SIGCOMM 1998.

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

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

[7] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

---

[1] Explicit Congestion Notification - явное уведомление о насыщении. *Прим. перев.*

[2] Документ доступен по ссылке http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-771.pdf. *Прим. перев.*

[3] Документ доступен по ссылке http://www.icir.org/tfrc/tcp-friendly.pdf. *Прим. перев.*

[4] См. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.8816&rep=rep1&type=pdf. Прим. перев.

[5] Документ доступен по ссылке http://www.sigcomm.org/sigcomm98/tp/paper25.pdf. *Прим. перев.*

[8] Wetherall, D., Ely, D., N. Spring, S. Savage, and T. Anderson, "Robust Congestion Signaling", IEEE International Conference on Network Protocols, November 2001.

[9] Widmer, J., "Equation-Based Congestion Control", Diploma Thesis, University of Mannheim, February 2000. URL "http://www.icir.org/tfrc/".

13. Адреса авторов

Mark Handley

ICIR/ICSI

1947 Center St, Suite 600

Berkeley, CA 94708

EMail: mjh@icir.org

Sally Floyd

ICIR/ICSI

1947 Center St, Suite 600

Berkeley, CA 94708

EMail: floyd@icir.org

Jitendra Padhye

Microsoft Research

EMail: padhye@microsoft.com

Joerg Widmer

Lehrstuhl Praktische Informatik IV

Universitat Mannheim

L 15, 16 - Room 415

D-68131 Mannheim

Germany

EMail: widmer@informatik.uni-mannheim.de

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

14. Полное заявление авторских прав

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.