RTO Network Working Group
Request for Comments: 1323
Obsoletes: RFC 1072, RFC 1185

V. Jacobson
LBL
R. Braden
ISI
D. Borman
Cray Research

May 1992

Расширения TCP для высокой производительности

TCP Extensions for High Performance

Статус документа

Этот документ содержит проект стандарта IAB для сообщества Internet и служит приглашением к дискуссии в целях развития и совершенствования протокола. Информацию о текущем состоянии стандартизации для этого протокола вы можете найти в документе «IAB Official Protocol Standards»[1]. Допускается свободное распространение документа.

Аннотация

Этот документ представляет набор расширений TCP, обеспечивающих повышение производительности для путей с большими значениями произведения полосы пропускания на время кругового обхода (bandwidth*delay) и надежную работу через каналы с очень высокой скоростью. Документ определяет новые поции TCP для масштабирования окон и временных меток, которые должны обеспечить совместимость и интероперабельность с реализациями TCP, не поддерживающими это расширение. Временные метки используются для двух разных механизмов - RTTM[2] и PAWS[3]. Селективные подтверждения не включены в этот документ.

Этот документ объединяет в себе информацию, содержащуюся в RFC 1072 и RFC 1185, отменяя действие обоих документов. Приведённые в этом документе спецификации являются более детальными. В Приложении C рассматриваются отличия от упомянутых RFC.

Оглавление

[1] В настоящее время документ называется “Internet Official Protocol Standards” (STD1). Текущая версия STD1 опубликована в RFC 3700. *Прим. перев.*
[2] Round Trip Time Measurement - измерение времени кругового обхода.
[3] Protect Against Wrapped Sequences - защита от перехода порядкового номера через максимально возможное значение.

1. Введение

Протокол TCP [Postel81] был разработан для обеспечения гарантированной доставки практически во всех средах передачи, независимо от скорости, величины задержки, потери и дублирования, а также изменения порядка доставки сегментов. Работа реализаций TCP в настоящее время[1] оптимизирована для передачи данных со скоростями от 100 бит/с до 10^7 бит/с и временами кругового обхода от 1 мсек до 100 секунд. Недавние исследования показали, что протокол TCP может работать с самыми разными путями Internet, начиная от каналов ввода-вывода со скоростью 800 Мбит/с и заканчивая модемными соединениями со скоростью 300 бит/с [Jacobson88a].

Использование оптических каналов привело к значительному росту скорости и наиболее скоростные пути вышли за те пределы, для которых изначально создавался протокол TCP. В этом документе определен набор незначительных расширений протокола TCP, которые учитывают рост производительности сетей. Документ основан на утративших силу RFC 1072 [Jacobson88b] и RFC 1185 [Jacobson90b].

Документ не дает однозначного ответа на вопрос: «Сколь быстро может работать TCP?» В документе отдельно рассматриваются вопросы производительности и надежности, в зависимости от различных параметров.

1.1 Производительность TCP

Производительность TCP зависит не только от скорости передачи, как таковой, но и от произведения скорости передачи на время кругового обхода. Это произведение bandwidth*delay определяет количество данных, которые «заполнят трубу». Эта величина определяет размер буферов, требуемых на стороне отправителя и получателя для обеспечения максимальной производительности соединения TCP через данный путь, т. е., объем данных с еще не подтвержденной доставкой, которые протокол TCP должен обеспечивать для «заполнения трубы». При больших значениях произведения bandwidth*delay могут возникать проблемы с производительностью TCP. Будем обозначать пути через Internet с большим значением произведения, как «long, fat pipe[2]», а сети, содержащие такие пути, как «LFN» (произносится «elephan(t)»[3]).

Широкополосные спутниковые каналы (например, DARPA Wideband Net) относятся к числу LFN. Например, спутниковый канал со скоростью DS1 имеет произведение скорости на время кругового обхода порядка 10^6 или более битов; это соответствует 100 находящимся в сети сегментам TCP размером по 1200 байтов каждый. Наземные оптические каналы также относятся к числу LFN. Например, при задержке международного канала 30 мсек и полосе DS3 (45 Мбит/с) произведение составит 10^6 битов.

При использовании современных реализаций TCP через пути LFN возникают 3 фундаментальных проблемы.

(1) Ограничение размера окна

Заголовок TCP использует 16-битовое поле для передачи отправителю информации о размере окна приема. Следовательно, максимальный размер окна приема составляет 2^{16} байт = 64 Кбайт[4].

Для решения этой проблемы в главе 2 данного документа определена новая опция TCP - Window Scale, позволяющая задавать размер окна более 2^{16}. Эта опция определяет неявный множитель, который используется в качестве коэффициента при определении реального размера окна из значения, полученного в заголовке TCP.

(2) Восстановление при потерях

Потеря пакетов в LFN может приводить к катастрофическому снижению производительности. Вплоть до недавнего времени корректно работающие реализации TCP могли уменьшать «размер трубы» в ответ на потерю каждого пакета и требовать использования процедуры замедленного старта для восстановления. Недавно были предложены алгоритмы Fast Retransmit[5] и Fast Recovery[6] [Jacobson90c]. Суммарный эффект использования этих алгоритмов заключается в том, что при потере одного пакета не происходит «сужения трубы». Однако при потере более одного пакета в окне обычно будет возникать таймаут повтора передачи с соответствующим «сужением трубы» и использованием алгоритма slow start.

Увеличение размера окна с учетом полосы LFN ведет к повышению вероятности отбрасывания более одного пакета в окне. Это может существенно снижать производительность работы TCP через LFN. Кроме того, если механизм контроля насыщения основан на использовании той или иной формы случайного отбрасывания пакетов на шлюзе, отбрасывание пакетов со случайным интервалом между ними станет более распространенным и приведет к росту вероятности отбрасывания более одного пакета в окне.

Для обобщения механизмов Fast Retransmit/Fast Recovery на случай потери множества пакетов в окне требуется механизм селективных подтверждений. В отличие от кумулятивных подтверждений TCP, селективные подтверждения дают отправителю полную картину сегментов, которые уже находятся в очереди на приемной стороне, и пакетов, которые еще не доставлены. Некоторые преимущества использования селективных подтверждений были приведены в работе [NBS85] и механизм селективных подтверждений был включен во множество экспериментальных протоколов Internet – VMTP [Cheriton88], NETBLT [Clark87] и RDP [Velten84], а также предложен для OSI TP4 [NBS85]. Однако при работе без LFN селективные подтверждения снижают число повторов передачи пакетов, но не приводят к росту производительности, что делает применимость этого достаточно сложного механизма весьма спорной. Вместе с тем, для случаев работы через LFN важность селективных подтверждений может значительно вырасти.

RFC 1072 определяет новую опцию TCP для передачи селективных подтверждений (SACK). Однако имеется ряд важных технических проблем с форматом и семантикой опции SACK. Поэтому данная опция была исключена из предлагаемого в этом документе списка расширений. Эта опция однако может быть возвращена в процессе стандартизации[7].

---

[1] Май 1992 г. *Прим. перев.*
[2] Длинная, толстая труба.
[3] Слон. *Прим. перев.*
[4] В оригинале указано «65Kbytes». *Прим. перев.*
[5] Ускоренный повтор передачи.
[6] Ускоренное восстановление.
[7] В несколько измененной форме эта опция была «возвращена» в RFC 2018. *Прим. перев.*

(3) Измерение времени кругового обхода

TCP обеспечивает гарантированную доставку данных за счет повтора передачи сегментов, которые не были подтверждены в течение интервала RTO[1]. Аккуратное динамическое определение значения RTO имеет важное значение для производительности TCP. Значение RTO определяется путем оценки среднего значения и вариаций времени кругового обхода RTT (интервал между передачей сегмента и получением подтверждения его доставки [Jacobson88a]).

В параграфе 3.2[2] определяется новая опция TCP Timestamps и механизм ее использования, позволяющий определять время почти для каждого сегмента (включая повторы), практически без дополнительных издержек на вычисления. Для этого механизма используется аббревиатура RTTM[3], чтобы отличить его от других вариантов использования опции.

1.2 Надежность TCP

Далее мы рассмотрим вопросы надежности доставки. Повышение скорости передачи ведет к росту производительности TCP, как bandwidth*delay. Однако повышение скорости передачи может угрожать надежности доставки TCP по той причине, что не будут выполняться допущения, используемые протоколом TCP для детектирования дубликатов и нарушения порядка доставки.

Особенно серьезные неприятности могут возникать в результате нечаянного повторного использования порядковых номеров TCP в сегментах данных. Предположим, что «старый сегмент-дубликат» (например, дубликат сегмента данных, задержанного в очередях Internet) доставлен получателю в неподходящий момент и его порядковый номер попадает в текущее окно. При проверке контрольной суммы ошибки обнаружено не будет и в результате может возникнуть трудно детектируемое повреждение данных. Получение старого дубликата сегмента ACK отправителем также может приводить к серьезным неприятностям, вплоть до блокирования соединения и последующей передачи RST.

Надежность TCP зависит от наличия границы для времени жизни сегмента - MSL[4]. Значение MSL в общем случае требуется для любого транспортного протокола с гарантированной доставкой, поскольку поле порядкового номера имеет конечный размер и, следовательно, возможно повторное использование порядковых номеров. В стеке протоколов Internet значение MSL ограничивается временем жизни (TTL) пакетов на уровне IP.

Дублирование порядковых номеров может возникать в двух случаях:

(1) Порядковые номера достигают максимального значения в течение срока существования данного соединения.

Порядковые номера TCP имеют размер 32 бита. При достаточно высокой скорости верхняя граница 32-битового пространства порядковых номеров может быть достигнута за то время, что сегмент будет находиться в очередях.

(2) Прежняя инкарнация соединения.

Предположим, что соединение разрывается путем корректного или аварийного завершения и незамедлительно создается такое же (с той же парой сокетов) новое соединение. Задержанный сегмент из разорванного соединения может попасть в текущее окно и новая инкарнация воспримет его, как корректный.

От дубликатов из более ранних инкарнаций (случай 2) можно избавиться с использованием текущего фиксированного значения MSL, определенного спецификацией TCP (см. параграф 5.3 и Приложение B к этой спецификации). Однако для случая (1) предотвращение повторного использования порядковых номеров в одном соединении требует ограничивать MSL в соответствии со скоростью передачи и для высоких скоростей нужен новый механизм.

Если максимальная эффективная полоса, которая доступна TCP через конкретный путь, составляет B байтов в секунду, для безошибочной работы требуется выполнение следующего условия:

$2^{31}/B > MSL$ (сек) [1]

Сеть	*B*8*	*B байт/с*	*Twrap (сек)*
ARPANET	56 кбит/с	7 кбайт/с	$3*10^5$ (~3,6 дней)
DS1	1,5 Вбит/с	190 кбайт/с	10^4 (~3 часа)
Ethernet	10 Мбит/с	1,25 Мбайт/с	1700 (~ 30 мин.)
DS3	45 Мбит/с	5,6 Мбайт/с	380
FDDI	100 Мбит/с	12,5 Мбайт/с	170
Gigabit	1 Гбит/с	125 Мбайт/с	17

Приведенная справа таблица содержит значения Twrap = 2^{31}/B для некоторых распространенных значений полосы B.

Очевидно, что достижение границы значений порядковых номеров не составляет проблемы для сетей с коммутацией пакетов со скоростью 56 кбит/с и даже для сетей Ethernet 10 Мбит/с. С другой стороны, при скоростях DS3 и FDDI значение Twrap становится сравнимым со значением MSL=2 мин., заданным спецификацией TCP [Postel81]. При переходе к гигабитной скорости значение Twrap становится слишком малым для того, чтобы надежная доставка могла обеспечиваться принятым в Internet механизмом TTL.

16-битовое поле окна TCP ограничивает эффективную полосу B значением 2^{16}/RTT, где RTT – время кругового обхода в секундах [McKenzie89]. Если значение RTT достаточно велико, это будет ограничивать полосу B значением, которое удовлетворяет требованиям выражения [1] для большого значения MSL. Например, рассмотрим трансконтинентальную

[1]Retransmission timeout - таймаут повтора передачи.
[2]В оригинале ошибочно указано, что эта опция определена в главе 4. *Прим. перев.*
[3]Round Trip Time Measurement – измерение времени кругового обхода
[4]Maximum Segment Lifetime - максимальный срок жизни сегмента.

магистраль с RTT = 60 мсек. (определяется физическими параметрами). Если произведение bandwidth*delay ограничено размером максимального окна TCP (64 кбайт), полоса B будет ограничена значением 1,1 Мбит/с независимо от того, какова физическая скорость канала передачи. Это соответствует времени использования всего пространства порядковых номеров Twrap= 2000 сек., что достаточно безопасно для современной сети Internet.

Важно понимать, что причиной ограничения является не маленькое окно, а широкая полоса. В качестве примера рассмотрим (очень скоростную) ЛВС FDDI с диаметром 10 км. Используя значение скорости света, можно рассчитать значение RTT для кольца, как $(2*10^4)/(3*10^8)$ = 67 мксек, и значение delay*bandwidth = 833 байта. Соединение TCP через такую ЛВС при размере окна 833 байта будет работать с полной скоростью 100 Мбит/с и пространство порядковых номеров будет полностью использовано за время порядка 3 минут, что очень близко к значению MSL для TCP. Таким образом, очень высокая скорость может создавать проблемы с надежностью в результате достижения максимального порядкового номера даже без применения расширенных окон.

Протокол Ватсона Delta-T [Watson81] включает механизмы сетевого уровня для точного исполнения MSL. В отличие от этого в протоколе IP механизм исполнения MSL определен слабо и еще более слабо реализован в Internet. Следовательно, неразумно полагаться на использование MSL для соединений TCP и нереально задавать значения MSL меньше принятых в настоящее время (например, меньше заданного спецификацией TCP значения 120 сек.).

Возможным решением проблемы достижения границы порядковых номеров является увеличение размера поля порядкового номера TCP. Например, поле порядкового номера (и номера подтверждения) можно расширить до 64 битов. Это можно сделать путем изменения заголовка TCP или введения новой опции.

В главе 5 предлагается иной механизм (PAWS[1]) повышения надежности TCP для скоростей передачи, превосходящих предсказуемые пределы. PAWS использует опцию TCP Timestamps, определенную в главе 4, для защиты от появления старых дубликатов в том же соединении.

1.3 Использование опций TCP

Все определенные в этом документе расширения используют новые опции TCP. Мы должны решить два вопроса, связанные с использованием опций TCP, - (1) совместимость и (2) дополнительные издержки.

Требуется обратить внимание на совместимость, т. е., интероперабельность с существующими реализациями. Единственная определенная ранее опция TCP MSS[2] может появляться только в сегментах SYN. Каждой реализации следует игнорировать неизвестные опции в сегментах SYN (предполагается, что большинство реализаций делает это). Однако некоторые ошибочные реализации TCP могут аварийно завершать работу при первом появлении опции в сегменте, отличном от SYN. Следовательно, для каждого расширения, определенного ниже, опции TCP будут передаваться в сегментах, отличных от SYN только в тех случаях, когда обмен опциями в сегментах SYN показал, что обе стороны понимают расширение. Более того, опции расширения будут передаваться в сегментах SYN,ACK только в ответ на получение соответствующей опции в начальном сегменте SYN.

Может возникнуть вопрос о расходе полосы и дополнительных вычислениях, связанных с опциями TCP. Те опции, что появляются в сегментах SYN, не будут, очевидно, создавать проблем с производительностью. Создание соединения TCP требует выполнения значительного объема специального кода и обработка опций не приведет к существенному росту издержек в этом случае.

С другой стороны, опция Timestamps может появляться в любых сегментах данных и подтверждений (ACK), добавляя 12 байтов к 20-байтовому заголовку TCP. Мы надеемся, что полоса, сэкономленная за счет предотвращения ненужных повторов, будет больше, нежели расход полосы в связи с увеличением заголовка.

Существует также проблема дополнительного расхода вычислительных ресурсов на обработку опций переменного размера, выровненных по границе байта, в частности для процессоров с RISC-архитектурой. В Приложении A приводится рекомендуемая схема опций в заголовке TCP, обеспечивающая разумное выравнивание полей. Применяя идею метода Header Prediction[3], TCP может быстро проверить эту схему и при положительном результате использовать быстрый путь. Хосты, которые используют эту каноническую схему, будут эффективно применять опции, как набор полей с фиксированным форматом, появляющихся в заголовке TCP. Однако для сохранения философии и схемы опций протокол TCP должен быть готов к разбору произвольных полей опций, пусть и менее эффективно.

В заключение отметим, что большинство механизмов, определенных в этом документе, важны для LFN и высокоскоростных сетей. Для сетей с невысокими скоростями при оптимизации производительности эти механизмы могут **не** использоваться. Разработчики TCP, озабоченные производительностью на низкоскоростных путях, могут отключить эти механизмы для медленных путей или разрешить пользователю или администратору отключать их самостоятельно.

2. Опция масштабирования окна TCP

2.1 Введение

Расширение для масштабирования окна увеличивает поле окна TCP до 32 битов и использует коэффициент масштабирования для передачи этого 32-битового размера в 16-битовом поле Window заголовка TCP (SEG.WND в RFC 793). Коэффициент масштабирования передается в новой опции TCP Window Scale. Эта опция передается только в сегменте SYN, следовательно коэффициент масштабирования фиксирован для каждого направления после того, как соединение организовано. Другим вариантом решения была установка коэффициента масштабирования для каждого сегмента TCP. Будет некорректно передавать опцию Window scale только при изменении коэффициента масштабирования, так как опция TCP в сегменте подтверждения не будет доставляться с гарантией (если ACK не объединяентся с передаваемым в обратном направлении пакетом данных). Фиксация коэффициента масштабирования при организации соединения снижает издержки, но недостатком этого варианта является невозможность изменения коэффициента масштабирования окна в процессе работы соединения.

[1] Protect Against Wrapped Sequence - защита от перехода порядкового номера через максимально возможное значение.
[2] Максимальный размер сегмента. *Прим. перев.*
[3] Предсказание заголовка.

Максимальное окно приема и, следовательно, коэффициент масштабирования определяются максимальным размером приемного буфера. В типовой современной реализации максимальное буферное пространство задается по умолчанию, но может быть изменено пользовательской программой до того, как будет организовано соединение TCP. Это определяет коэффициент масштабирования и, следовательно, не требуется дополнительный пользовательский интерфейс для масштабирования окна.

2.2 Опция Window Scale

Трехбайтовая опция Window Scale может передаваться в сегментах SYN. Опция служит двух целям: (1) показать, что протокол TCP готов масштабировать окна приема и передачи, а также (2) передать значение коэффициента масштабирования, который будет использоваться для окна приема. Таким образом, реализации TCP, готовой к масштабированию окон, следует передавать эту опцию даже при использовании коэффициента 1. Коэффициент масштабирования задается степенью числа 2 и представляется в логарифмическом масштабе, чтобы можно было реализовать масштабирование путем операции двоичного сдвига.

Опция TCP Window Scale (WSopt)

```
+---------+---------+---------+
| Kind=3  |Length=3 |shift.cnt|
+---------+---------+---------+
```

Тип: 3, размер: 3 байта

Эта опция служит предложением, а не обещанием - обе стороны должны передать опции Window Scale в своих сегментах SYN для того, чтобы разрешить масштабирование окна в каждом направлении. Если масштабирование окна разрешено, реализация TCP, которая передает эту опцию, будет смещать на shift.cnt битов вправо свой истинный размер окна (SEG.WND) перед включением размера окна в заголовок. Значение shift.cnt может быть нулевым (коэффициент масштабирования окна приема равен 1).

Эта опция может быть передана в начальном сегменте SYN (т. е., сегменте с флагом SYN, но без флага ACK) или в сегменте SYN,ACK (но только в том случае, когда опция Window Scale была получена в начальном сегменте SYN). Опции Window Scale в сегментах без флага SYN следует игнорировать.

Поле Window в сегменте с флагом SYN (т. е., SYN или SYN,ACK) никогда не масштабируется.

2.3 Использование опции Window Scale

Ниже описана модель реализации масштабирования окна с использованием нотации RFC 793 [Postel81]:

- Все размеры окон трактуются как 32-битовые значения размера буфера для хранения блока управления соединением и локальных расчетов. Размеры задаются для окна передачи (SND.WND), окна приема (RCV.WND) и окна насыщения.
- Состояние соединения зависит также от двух коэффициентов масштабирования Snd.Wind.Scale и Rcv.Wind.Scale, применяемых к входящим и исходящим полям размера окна, соответственно.
- Если модуль TCP получает сегмент SYN, содержащий опцию Window Scale, он передает в ответ свою опцию Window Scale в сегменте SYN,ACK.
- Опция Window Scale передается с shift.cnt = R, где R задает значение коэффициента, которое TCP будет использовать для своего окна приема.
- При получении сегмента SYN с опцией Window Scale, содержащей shift.cnt = S, модуль TCP устанавливает для Snd.Wind.Scale значение S, а для Rcv.Wind.Scale - R; в противном случае для обоих параметров Snd.Wind.Scale и Rcv.Wind.Scale устанавливается нулевое значение.
- Поле размера окна SEG.WND в заголовке каждого входящего сегмента (за исключением сегментов SYN) смещается влево на Snd.Wind.Scale перед обновлением значения SND.WND:

```
SND.WND = SEG.WND << Snd.Wind.Scale
```

(предполагается, что другие условия RFC793 выполнены и используется нотация языка C << для сдвига влево).

- Поле размера окна для исходящих сегментов (за исключением SYN) получается из истинного размера путем сдвига SEG.WND вправо на Rcv.Wind.Scale:

```
SEG.WND = RCV.WND >> Rcv.Wind.Scale
```

TCP определяет «старые» и «новые» данные в сегментах, проверяя, попадает ли их порядковый номер в 2^{31} байтов от левого края окна (данные, которые не попадают, считаются «старыми»). Чтобы новые и старые данные никогда не путались, левый край окна отправителя находится не более, чем на 2^{31} байта от правого края окна получателя. Аналогичное условие выполняется и относительно левого края окна получателя и правого края окна отправителя. Поскольку правый и левый край окна отправителя или получателя отличаются на размер окна, а окна отправителя и получателя могут быть расфазированы не более, чем на величину размера окна с учетом приведенных выше ограничений, удвоенный максимальный размер окна должен быть меньше 2^{31} или

```
max window < 2**30
```

Поскольку максимальный размер окна в 2^S (где S задает число битов сдвига при масштабировании окна) превышает 2^{16}-1 (максимальный размер окна без масштабирования), для максимального размера окна гарантируется значение меньше 2^{30}, если S <= 14. Таким образом, значение счетчика сдвига не должно быть более 14 (это значение разрешает размер окна 2^{30} байт = 1 Гбайт). Если получена опция Window Scale со значением shift.cnt, превышающим 14, модулю TCP следует записать сообщение об ошибке в системный журнал и использовать 14 взамен указанного в полученной опции значения.

Коэффициент масштабирования применяется только к значению поля Window, переданному в заголовке TCP; каждый модуль TCP, использующий расширенные окна, будет локально поддерживать для размера окна 32-битовые значения.

Например, окно насыщения, рассчитываемое алгоритмами Slow Start и Congestion Avoidance, не будет увеличиваться на коэффициент масштабирования, поэтому масштабирование окна не будет оказывать влияния на расчет окна насыщения.

3. RTTM - измерение времени кругового обхода

3.1 Введение

Точная и своевременная оценка RTT необходима для адаптации изменений в условиях передачи трафика и предотвращения нестабильности, называемой «congestion collapse[1]» [Nagle84] в загруженной сети. Однако точное измерение RTT может оказаться затруднительным как в теории, так и для реализации.

Многие реализации TCP измеряют RTT на основе выборки единственного пакета в окне. Хотя этот метод дает адекватную оценку RTT для небольших окон, точность его неприемлема для LFN. Если рассматривать оценку RTT как задачу обработки сигналов (каковой она и является), сигнал данных с некоторой частотой (скорость передачи пакетов) будет выбираться с низкой частотой («скорость» окна). Такая редкая выборка сигналов противоречит критерию Найквиста и может, следовательно, вносить «эффект наложения» в измерение RTT [Hamming77].

Хорошая система оценки RTT с консервативным расчетом таймаута повторной передачи может быть устойчивой к «наложению» при частоте выборки, близкой к частоте данных. Например, для окна в 8 пакетов частота выборки[2] составляет 1/8 от частоты данных, отличаясь меньше, чем на порядок. Однако, когда окно включает десятки и сотни пакетов, оценка RTT может давать серьезные ошибки, приводящие к ненужным повторам передачи.

Отбрасывание пакетов осложняет проблему. Zhang [Zhang86], Jain [Jain86] и Karn [Karn87] показали невозможность аккумулировать надежные оценки RTT, если при такой оценке используются передаваемые повторно сегменты. Поскольку до повтора передачи было передано полное окно данных, все сегменты этого окна будут подтверждены до того, как будет сделана следующая выборка RTT. Это означает, что между измерениями RTT пройдет по крайней мере время передачи дополнительного окна и, поскольку частота ошибок близка к одной ошибке на размер окна (например, 10^{-6} ошибок на бит для спутниковой сети Wideband), корректное измерение RTT становится невозможным.

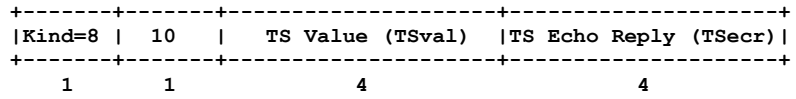
Решение этой проблемы, значительно упрощающее работу отправителя, состоит в следующем - используя опции TCP, отправитель включает временную метку в каждый сегмент данных, а получатель возвращает эти временные метки в сегментах ACK. После этого одна операция вычитания дает отправителю точное значение RTT для каждого сегмента ACK (который будет соответствовать каждому сегменту данных при разумном поведении получателя). Мы назвали этот механизм RTTM.

Применение механизма RTTM является жизненно важным при использовании больших окон. В противном случае открывается дверь для некоторых опасных нестабильностей в результате наложений. Более того, эта опция может оказаться полезной для всех модулей TCP, поскольку она упрощает работу отправителя.

3.2 Опция TCP Timestamps

Протокол TCP является симметричным и позволяет в любой момент передавать данные в обоих направлениях, следовательно возврат временных меток также может осуществляться в обоих направлениях. Для простоты и симметрии будем предполагать, что временные метки всегда передаются и возвращаются в обоих направлениях. Для повышения эффективности поля timestamp и timestamp reply объединены в одну опцию TCP Timestamps.

Опция TCP Timestamps (TSopt):



тип: 8

размер: 10 байтов

Опция Timestamps передает два 4-байтовых поля временных меток. Поле Timestamp Value (TSval) содержит текущее значение временной метки передающей опцию стороны TCP.

Поле Timestamp Echo Reply (TSecr) корректно только при установленном флаге ACK в заголовке TCP; в этом случае поле содержит значение временной метки, переданное удаленной стороной TCP в поле TSval опции Timestamps. Если поле Tsecr некорректно, его значение должно быть нулевым. Значение TSecr в общем случае берется из последней принятой опции Timestamp, однако имеются некоторые исключения, описанные ниже.

TCP может передавать опцию Timestamps (TSopt) в начальном сегменте SYN (т. е., в сегменте с флагом SYN, но без флага ACK), а также в других сегментах, если опция TSopt была получена в начальном сегменте SYN для этого соединения.

3.3 Механизм RTTM

Значение временной метки, которое будет передаваться в Tsval, берется из показаний (виртуальных) часов, которые называют «timestamp clock[3]». Показания этого хронометра для измерения RTT должны быть по крайней мере приблизительно пропорциональны реальному времени.

[1] Коллапс насыщения.
[2] При выборке одного пакета из окна. *Прим. перев.*
[3] Хронометр временных меток.

```
TCP  A                                          TCP B
          <A,TSval=1,TSecr=120> ------>
<---- <ACK(A),TSval=127,TSecr=1>
          <B,TSval=5,TSecr=127> ------>
<---- <ACK(B),TSval=131,TSecr=5>
. . . . . . . . . . . . . . . . . . . . . . . .
          <C,TSval=65,TSecr=131> ------>
<---- <ACK(C),TSval=191,TSecr=65>
(и т. д.)
```

Приведенный на рисунке пример иллюстрирует односторонний поток данных без потери сегментов. A, B, C... представляют блоки данных с последовательными порядковыми номерами, а ACK(A),... представляют соответствующие кумулятивные подтверждения. Два поля временных меток опции Timestamps показаны, как <TSval=x, TSecr=y>. Каждое поле TSecr содержит значение полученного последним поля TSval.

Строка точек показывает паузу (продолжительностью в 60 временных интервалов), в течение которой узел A ничего не передает. Отметим, что эта пауза завышает значение RTT, которое узел B может взять из полученного в сегменте данных C значения TSecr=131. Таким образом, при одностороннем потоке данных механизм RTTM для обратного направления дает значение, увеличенное на продолжительность паузы в передаче данных. Однако приведенное ниже правило позволяет предотвратить ненужное увеличение RTT:

> Значение Tsecr, полученное в сегменте, используется для обновления среднего значения RTT лишь в тех случаях, когда сегмент подтверждает некие новые данные (т. е., номер подтверждения смещен вперед относительно левого края окна передачи).

Поскольку TCP B не передает данных, сегмент данных C не подтверждает никаких новых данных, поэтому увеличенный на продолжительность паузы в передаче результат RTTM не используется узлом B для обновления времени кругового обхода.

3.4 Какую метку следует возвращать

Если получено более одной опции Timestamps до передачи сегмента отклика, модуль TCP должен выбрать для возврата одно из значений Tsvals, игнорируя остальные. Для минимизации числа хранимых получателем состояний (например, числа необработанных значений Tsval) получателю следует сохранять не более одной временной метки в блоке управления соединением.

Следует рассмотреть три ситуации:

A) *Задержанные подтверждения*.

> Многие реализации TCP подтверждают лишь каждый K-ый сегмент из группы сегментов, прибывающих в течение короткого интервала времени. Такой подход обычно называют «отложенным подтверждением». Отправитель данных TCP должен измерить эффективное значение RTT, включая дополнительное время, связанное с отложенными подтверждениями, поскольку в противном случае будут переданы ненужные повторы. Таким образом, при использовании отложенных подтверждений получателю следует возвращать то значение поля Tsval, которое было включено в первый из неподтвержденных сегментов.

B) *Пропуск в порядковых номерах* (потеря сегментов).

> Отправитель будет продолжать передачу данных до заполнения окна, а получатель может генерировать подтверждения по прибытии сегментов с нарушением порядка (например, для активизации ускоренного повтора).
>
> Потеря сегмента может говорить о перегрузке в сети и в такой ситуации отправителю следует быть консервативным при решении вопроса о повторе передачи. Более того, лучше переоценить значение RTT, нежели недооценить его. В подтверждениях сегментов, доставленных с нарушением порядка, следует по этой причине указывать временную метку из последнего принятого сегмента, который расположен впереди окна.
>
> Такая же ситуация происходит и при нарушении порядка следования пакетов в сети.

C) *Заполнение пропуска в порядковых номерах*.

> Сегмент, который заполняет пропуск, представляет наиболее свежую информацию для измерения характеристик сети. С другой стороны, значение RTT, рассчитанное для более раннего сегмента, может включать таймаут повторной передачи для отправителя, что окажет отрицательное воздействие на полученное отправителем среднее значение RTT. Таким образом, следует передавать значение временной метки из последнего сегмента, который заполняет пропуск в порядковых номерах.

Описанный ниже алгоритм обработки опций Timestamps для синхронизированного соединения учитывает все три рассмотренных выше случая.

(1) Состояние соединения сохраняется в двух 32-битовых переменных - TS.Recent содержит временную метку для возврата в поле TSecr при передаче сегмента, а Last.ACK.sent - поле ACK из последнего переданного сегмента. Значение Last.ACK.sent будет равно RCV.NXT за исключением тех случаев, когда передача подтверждения была отложена.

(2) Если значение Last.ACK.sent попадает в диапазон порядковых номеров входящего сегмента

```
SEG.SEQ <= Last.ACK.sent < SEG.SEQ + SEG.LEN
```

поле TSval из этого сегмента копируется в TS.Recent; в противном случае TSval игнорируется.

(3) Когда TSopt передается, в поле TSecr помещается текущее значение TS.Recent.

Приведенные ниже примеры иллюстрируют эти правила. A, B, C... представляют сегменты, занимающие последовательные блоки порядковых номеров, а ACK(A),... представляют соответствующие сегменты подтверждения. Отметим, что ACK(A) имеет такой же порядковый номер, как B. Для простоты представлено только одно направление возврата временных меток.

- Пакеты доставляются с сохранением порядка, некоторые подтверждения откладываются (см. рисунок).

```
                                        TS.Recent
<A, TSval=1> ------------------->
                                            1
<B, TSval=2> ------------------->
                                            1
<C, TSval=3> ------------------->
                                            1
          <---- <ACK(C), TSecr=1>
(и т. д.)
```

- В случае (A) возвращается временная метка из наиболее старого неподтвержденного сегмента.

- Пакеты доставляются с нарушением порядка и каждый пакет подтверждается (см. рисунок).

```
                                        TS.Recent
<A, TSval=1> ------------------->
                                            1
         <---- <ACK(A), TSecr=1>
                                            1
<C, TSval=3> ------------------->
                                            1
         <---- <ACK(A), TSecr=1>
                                            1
<B, TSval=2> ------------------->
                                            2
         <---- <ACK(C), TSecr=2>
                                            2
<E, TSval=5> ------------------->
                                            2
         <---- <ACK(C), TSecr=2>
                                            2
<D, TSval=4> ------------------->
                                            4
         <---- <ACK(E), TSecr=4>
(и т. д.)
```

В случае (B), возвращается временная метка из последнего сегмента, который находится впереди левого края окна, пока не будет доставлен отсутствующий сегмент, для которого временная метка возвращается в соответствии с (C). Такая же последовательность будет наблюдаться при потере и повторной передаче сегментов B и D..

4. PAWS - защита от перехода порядкового номера через максимум

4.1 Введение

В параграфе 4.2 описан простой механизм для отказа от приема старых сегментов-дубликатов, которые могут повредить активное соединение TCP; будем называть этот механизм PAWS (Protect Against Wrapped Sequence numbers – защита от достижения максимального значения порядкового номера). PAWS работает в масштабе одного соединения TCP, используя состояние, хранящееся в блоке управления соединением. В параграфе 4.3 и Приложении C обсуждается влияние механизма PAWS на избавление от старых дубликатов из предыдущих инкарнаций того же соединения.

4.2 Механизм PAWS

PAWS использует ту же опцию TCP Timestamps, что и механизм RTTM, описанный выше, и предполагает, что каждый принятый сегмент TCP (включая сегменты ACK) содержит временную метку SEG.TSval, значения полей которой не снижаются с течением времени. Основная идея заключается в том, что сегмент может быть отброшен как старый дубликат, если он получен со значением временной метки SEG.TSval, которое меньше полученного недавно значения временной метки для этого соединения.

В обоих механизмах PAWS и RTTM временные метки представляют собой 32-битовые целые числа без знака (модуль 2^{32}). Таким образом, отношение «меньше чем» для временных меток трактуется так же, как для порядковых номеров TCP и могут использоваться такие же методы реализации Если s и t – значения временных меток, s < t, если 0 < (t - s) < 2^{31} при использовании арифметики 32-битовых целых чисел без знака.

Выбор принимаемых временных меток для сохранения с целью последующего сравнения должен гарантировать монотонный рост значений. Например, можно сохранить временную метку из сегмента, который был получен последним и располагается на левом краю окна приема (т. е., самый свежий из неподтвержденных сегментов). Однако вместо этого мы выберем значение TS.Recent, описанное в параграфе 3.4 для механизма RTTM, поскольку использование общего значения для механизмов PAWS и RTTM упрощает реализацию обоих. Как сказано в параграфе 3.4, TS.Recent отличается от временной метки из полученного последним сегмента с корректным порядком доставки только в случае использования отложенных подтверждений и, следовательно, меньше, чем на одно окно. Любой из вариантов выбора будет обеспечивать защиту для случаев перехода порядковых номеров через верхнюю границу пространства нумерации.

RTTM задается синхронно, поэтому временные метки Tsval, передаваемые в сегментах данных и подтверждений, будут возвращаться в полях Tsecr, передаваемых в ответ подтверждений или сегментов данных. PAWS подвергает все входящие сегменты однотипной проверке и, следовательно, обеспечивает защиту от дубликатов сегментов данных и подтверждений. Дополнительный асимметричный алгоритм будет защищать от старых дубликатов ACK - отправитель

данных будет отвергать входящие сегменты ACK, в которых значение TSecr меньше, чем аналогичное значение, сохраненное из последнего сегмента ACK, в котором номер подтверждения располагается впереди левого края окна передачи.

Временные метки Tsval, передаваемые в сегментах SYN и SYN,ACK, используются для инициализации PAWS. Механизм PAWS защищает от старых дубликатов сегментов без флага SYN и дубликатов сегментов SYN, полученных в засинхронизированном состоянии соединения. Дубликаты сегментов SYN и SYN,ACK, полученные при отсутствии соединения, будут отбрасываться обычной процедурой 3-этапного согласования и проверки порядковых номеров TCP.

Рекомендуется **не** передавать временных меток в сегментах RST и принимать сегменты RST независимо от наличия в них временной метки. Появление старых дубликатов сегментов RST не представляется вероятным и функции их удаления следует отдавать предпочтение перед временными метками.

4.2.1 Базовый алгоритм PAWS

Алгоритм PAWS требует выполнения перечисленных ниже операций для всех входящих сегментов синхронизированного соединения:

R1) Если в полученном сегменте имеется опция Timestamps, SEG.TSval < TS.Recent и значение TS.Recent корректно (см. ниже), принятый сегмент трактуется, как неприемлемый:

в ответ передается подтверждение, как указано в RFC 793 (стр. 69[1]), а сегмент отбрасывается[2].

R2) Если сегмент не попадает в окно, его следует отвергнуть (обычная обработка TCP).

R3) Если прибывающий сегмент удовлетворяет условию SEG.SEQ ≤ Last.ACK.sent (см. параграф 3.4), временная метка записывается в TS.Recent.

R4) Если прибывающий сегмент не нарушает порядок доставки (т. е., располагается на левом краю окна), он воспринимается нормально.

R5) В остальных случаях сегмент трактуется как попадающий в окно и доставленный с нарушением порядка (т. е., сегмент помещается в очередь для доставки пользователю).

Этапы R2, R4 и R5 являются стандартными этапами обработки TCP, определенными в RFC 793.

Важно отметить, что временная метка проверяется сразу же при доставке сегмента получателю, независимо от того, прибыл сегмент в соответствии с нормальным порядком или должен быть помещен в очередь для более поздней доставки. Рассмотрим это на примере.

> Предположим, что была передана последовательность сегментов A.1, B.1, C.1, ..., Z.1, в которой буква представляет порядковый номер, а цифра – временную метку. Предположим также, что сегмент B.1 был потерян. Временная метка в TS.TStamp будет иметь значение 1 (из сегмента A.1), поэтому сегменты C.1, ..., Z.1 будут рассматриваться как приемлемые и помещены в очередь. Когда B будет передан повторно, как B.2 (с использованием новой временной метки), он заполнит пропуск и приведет к подтверждению всех сегментов вплоть до Z и доставке их пользователю. При этом временные метки из пакетов, находящихся в очереди, **не** будут проверяться заново, поскольку эти сегменты уже были приняты. При получении сегмента B.2 для переменной TS.Stamp будет установлено значение 2.

Это правило обеспечивает разумную производительность при возникновении потерь. В любой момент в процессе доставки находится полное окно данных и при потере одного пакета на целое окно произойдет нарушение порядка доставки с буферизацией полученных пакетов на приемной стороне (до ~2^{30} байтов данных); временная метка не должна приводить к отказу от доставки этих данных.

В некоторых маловероятных ситуациях правила R1-R4 могут приводить к неоправданному отбрасыванию некоторых пакетов, как показано в следующем примере.

> Предположим, что сегменты A.1, B.1, C.1, ..., Z.1 переданы с соблюдением порядка и сегмент B.1 потерян. Кроме того, предположим, что некоторые из сегментов C.1, ... Z.1 были задержаны и получены после доставки получателю повторного сегмента B.2. Эти задержанные сегменты будут неоправданно отброшены, поскольку в них будут содержаться устаревшие временные метки.

Описанная ситуация маловероятна. Если повтор передачи был вызван таймаутом, некоторые из сегментов C.1, ... Z.1 должны быть задержаны, на время, превышающее RTO. Это представляется маловероятным, если не предположить возникновение множества ложных таймаутов и повторов передачи. Если повтор передачи сегмента B был вызван алгоритмом Fast Retransmit (т. е, дубликатами ACK), находящиеся в очереди сегменты, которые вызвали передачу этих дубликатов, уже должны быть получены.

Даже при задержке сегмента на время больше RTO механизм Fast Retransmit [Jacobson90c] будет приводить к повторной передаче задержанных пакетов одновременно с B.2, избавляя от дополнительного периода RTT и, следовательно, минимально снижая производительность.

Нам не известно ситуаций, в которых бы использование временных меток приводило бы к существенному снижению производительности в результате неоправданного отбрасывания сегментов.

4.2.2 Хронометр временных меток

Важно понимать, что алгоритм PAWS не требует синхронизации часов отправителя и получателя. Временные метки включаются в сегменты отправителем и используются им же для измерения времени RTT. Получатель же трактует временные метки просто как монотонно возрастающие порядковые номера без привязки к реальному времени. С точки зрения получателя временная метка работает как логическое расширение старших битов порядкового номера.

---

[1] Страница 23 перевода RFC 793. *Прим. перев.*

[2] Передача сегмента ACK требуется для того, что сохранить возможность работы механизма детектирования и восстановления полуоткрытых соединений TCP. Примером может служить рисунок 10 в RFC 793.

Используемый получателем алгоритм предъявляет некоторые требования к хронометру временных меток.

(a) Хронометр временных должен быть не «слишком медленным».

Показания хронометра должны изменяться не менее одного раза за время передачи 2^{31} байтов. Фактически, для того, чтобы хронометр приносил отправителю пользу при оценке времени кругового обхода, его показания должны изменяться не реже одного раза за время передачи окна данных и даже при использовании расширения окна в соответствии с RFC 1072 2^{31} байтов данных должны размещаться по крайней мере в двух окнах.

Переходя к цифрам, можно сказать, что часы с изменением показаний 1 раз в секунду будут обеспечивать предотвращение старых дубликатов при скорости канала ~8 Гбит/с. Временные метки с разрешением 1 мсек будут работать при скорости канала до 8 Тбит/с ($8*10^{12}$ бит/с)!

(b) Хронометр временных должен быть не «слишком быстрым».

Время полного цикла отсчета времени хронометром должно быть больше MSL. Поскольку временная метка имеет размер 32 бита, а MSL не более 255 секунд, максимально допустимая скорость хода хронометра составляет 1 «тик» за каждые 59 нсек.

Однако желательно иметь значительно более продолжительный цикл отсчета для того, чтобы обрабатывать устаревшие временные метки в бездействующих соединениях (см. параграф 4.2.3) и снизить требования к MSL для защиты от перехода порядковых номеров через верхнюю границу. При дискретности хронометра в 1 мсек 32-битовые временные метки начнут повторяться через 24,8 дней. Таким образом, это обеспечит возможность предотвращения старых дубликатов из того же соединения, если значение MSL не превышает 24,8 дней. Такой вариант представляется очень безопасным - значения MSL в 24,8 дней или больше могут вероятно предполагаться шлюзовыми системами без требования точного исполнения MSL на основе значения TTL на уровне IP.

С учетом сказанного выше мы выбрали период изменения значений хронометра временных меток в диапазоне от 1 мсек до 1 сек. Этот диапазон также соответствует требованиям механизма RTTM, которому не требуется разрешение, превышающее гранулярность таймера повторной передачи (т. е., десятки или сотни миллисекунд).

Механизм PAWS предъявляет жесткое требование монотонности изменения показаний хронометра временных меток на стороне отправителя. Способы выполнения этого требования зависят от оборудования и программ, используемых в системе.

- Некоторые хосты имеют встроенные аппаратные часы, которые гарантируют монотонность изменения временных меток в период между сбросами хоста.
- Можно использовать прерывание системных часов для периодического инкрементирования двоичного целого числа на 1.
- Хронометр временных меток может быть связан с системными часами, к показаниям которых может быть добавлено то или иное смещение. Начальное смещение равно нулю. При возникновении необходимости генерации нового значения временной метки, смещение может быть скорректировано, чтобы значение новой временной метки было не меньше предыдущего значения метки (которое следует сохранять для сравнения).

4.2.3 Устаревшие временные метки

Если соединение в течение достаточно долгого времени бездействует, бит знака хронометра временных меток другой стороны TCP поменяет значение, хранящееся в переменной TS.Recent значение становится слишком старым и механизм PAWS будет приводить к отбрасыванию всех последующих сегментов, «замораживая» соединение (пока бит знака хронометра временных меток не восстановит свое значение).

При использовании выбранного периода изменения показаний хронометра временных меток (от 1 мсек. до 1 сек.) период смены знака будет составлять от 24,8 до 24800 дней. Маловероятно, что кто-то будет сохранять бездействующие соединения TCP дольше 24 дней. Тем не менее, внесение каких-либо ограничений на срок жизни соединений TCP нежелательно.

Поэтому мы требуем, чтобы реализация PAWS включала механизм признания значения TS.Recent «некорректным», если соединение бездействует более 24 дней. Другим вариантов решения проблемы устаревших временных меток является передача сегментов keepalive с очень малой частотой, которая, тем не менее, превышает частоту перехода значения таймера временных меток через максимум (например, можно передавать 1 такое сообщение в день). Эти сообщения будут вносить пренебрежимо малые издержки. Однако спецификация протокола не включает сегментов keepalive, поэтому было выбрано решение констатировать некорректность TS.Recent, как сказано выше.

Отметим, что модуль TCP не знает частоты и, следовательно, времени полного цикла отсчета хронометра другого узла TCP, поэтому ему следует исходить из худших предположений. Корректность TS.Recent требуется проверять только в тех случаях, когда не проходит базовая проверка PAWS для временной метки (например, если SEG.TSval < TS.Recent). Если обнаружена некорректность TS.Recent, сегмент принимается независимо от результата проверки временной метки и правило R3 обновляет TS.Recent значением TSval из нового сегмента.

Для детектирования продолжительности бездействия соединения TCP может обновлять показания часов или значение временной метки, связанные с соединением, например, при обновлении значения TS.Recent. Детали этого процесса определяются реализацией.

4.2.4 Предсказание заголовка

Header prediction [Jacobson90a] представляет собой метод реализации высокопроизводительного транспортного протокола, который важнее всего для высокоскоростных каналов. Этот метод оптимизирует код для более общего случая корректного приема сегментов с сохранением порядка. При использовании предсказания заголовков получатель задает вопрос: «Является этот сегмент следующим по порядку?» Ответ на этот вопрос требует меньшего числа процессорных команд, нежели ответ на вопрос: «Попадает ли этот сегмент в окно?»

Добавление предсказания заголовков к процедурам работы с временными метками ведет к приведенной ниже рекомендуемой процедуре обработки прибывающего сегмента TCP:

H1) Проверка временной метки (см. R1 выше).

H2) Процедура предсказания заголовка - если сегмент является следующим по порядку и не требуется специальной дополнительной обработки, сегмент принимается, значение временной метки сохраняется и этап H3 пропускается.

H3) Выполняется обычная обработка сегмента в соответствии с RFC 793. Эта обработка включает отбрасывание сегментов, выходящих за пределы окна, может включать передачу подтверждения и включает размещение в очереди относящихся к окну сегментов, которые доставлены с нарушением порядка.

Возможен вариант замены местами этапов H1 и H2, т. е. выполнение **сначала** процедуры предсказания заголовка H2, а затем выполнения H1 и перехода к H3 только при негативном результате предсказания заголовка. Это может обеспечить повышение производительности, поскольку проверка временной метки на этапе H1 с большой вероятностью дает позитивный результат, а для ее выполнения требуется использование интервальной арифметики на конечном поле, что занимает достаточно много ресурсов. Выполнение этой проверки для каждого отдельного сегмента противоречит философии предсказания заголовка. Мы надеемся, что это изменение может снизить расход процессорного времени на обработку заголовка TCP в высокоскоростных сетях на 5-10%.

Однако перенос этапа H2 в начало связан с опасностью - сегмент из 2^{32} прошлых байтов может быть принят точно в неподходящее время и ошибочно принят на этапе предсказания заголовка. В работе [Jacobson90b] был отмечен ряд причин, по которым вероятность такого события можно считать пренебрежимо малой.

> Если все сегменты показывают себя, как старые дубликаты, вероятность того, что старый дубликат точно соответствует левому краю окна, будет равна частному от деления размера максимального сегмента (MSS) на размер пространства порядковых номеров. Это значение должно быть меньше 2^{-16}, поскольку значение MSS должно быть меньше 2^{16} (например, для каналов FDDI это отношение будет $2^{12}/2^{32} = 2^{-20}$). Однако для наиболее старого сегмента сохранение его в сети Internet менее очевидно и вероятность того, что старый дубликат точно попадает в левый край окна, должна быть много меньше 2^{-16}.
>
> 16-битовая контрольная сумма TCP также вносит вклад в «базовую ненадежность». Протокольный механизм, надежность которого превосходит надежность контрольной суммы TCP, следует рассматривать, как «достаточно хороший», т. е. не вносящий существенного вклада в общее число ошибок. Поэтому мы считаем, что можно игнорировать возможность восприятия старых дубликатов, если выполнять процедуру предсказания заголовков до проверки временной метки.

Однако вероятностные аргумента принимаются не всеми и в настоящий момент принято решение, согласно которому повышение производительности в общем случае не оправдывает возникающего риска. Следовательно, рекомендуется выполнять сначала этап H1, а затем H2.

4.3. Дубликаты из прежних инкарнаций соединения

Механизм PAWS обеспечивает защиту от ошибок, связанных с достижением порядковым номером верхней границы диапазона на высокоскоростных соединениях. Сегменты из предыдущих инкарнаций того же соединения также могут быть причиной появления старых дубликатов. В обоих случаях механизм TCP для предотвращения таких ошибок зависит от контроля максимального срока жизни сегмента (MSL) на уровне IP (см. Приложение к RFC 1185, где этот вопрос рассмотрен детально). В отличие от случая достижения максимальных значений порядковых номеров требование соблюдения MSL для предотвращения дубликатов из прежних инкарнаций не зависит от скорости передачи. Если уровень IP ограничивает рекомендуемым значением (2 минуты) MSL для протокола TCP и протокол TCP следует правилам, соединения TCP не будут подвергаться опасному воздействию дубликатов из предыдущих инкарнаций, независимо от скорости работы сети. Следовательно, механизм PAWS в этом случае не требуется.

Остается вопрос - может ли механизм PAWS обеспечить дополнительную защиту от дубликатов из предшествующих соединений, которая позволила бы ослабить требование ограничения MSL на уровне IP. В Приложении B, где рассматривается этот вопрос, показано, что нужны дополнительные допущения и/или механизмы, помимо PAWS. Это выходит за пределы данного расширения.

5. Заключение и благодарности

Этот документ содержит набор расширений TCP для обеспечения эффективной работы протокола на путях с большим значением произведения полосы пропускания на время кругового обхода и повышения надежности при работе по высокоскоростным каналам. Эти расширения разработаны с учетом обеспечения совместимости с реализациями TCP, которые не поддерживают таких расширения.

Механизмы реализуются с использованием новых опций TCP для масштабирования окна и временных меток. Метки используются двумя разными механизмами - RTTM (измерение времени кругового обхода) и PAWS (защита от ошибок при достижении верхней границы порядковых номеров).

Опция Window Scale была изначально предложена Mike St. Johns из USAF/DCA. Текущая форма этой опции была разработана Mike Karels из UC Berkeley в ответ на более громоздкую схему, которую предложил Van Jacobson. Lixia Zhang помог с описанием механизма PAWS, предложенного в RFC 1185.

В заключение отметим, что значительная часть этой работы основана на результатах обсуждений в рабочей группе End-to-End Task Force теоретических ограничений транспортных протоколов вообще и TCP, в частности. Позднее члены рабочей группы вместе с другими участниками почтовой конференции end2end внесли существенный вклад в поиск недостатков в алгоритмах и описании. Авторы выражают свою признательность за эту работу.

6. Литература

[Clark87] Clark, D., Lambert, M., and L. Zhang, "NETBLT: A Bulk Data Transfer Protocol", RFC 998, MIT, March 1987.

[Garlick77] Garlick, L., R. Rom, and J. Postel, "Issues in Reliable Host-to-Host Protocols", Proc. Second Berkeley Workshop on Distributed Data Management and Computer Networks, May 1977.

[Hamming77] Hamming, R., "Digital Filters", ISBN 0-13-212571-4, Prentice Hall, Englewood Cliffs, N.J., 1977.

[Cheriton88] Cheriton, D., "VMTP: Versatile Message Transaction Protocol", RFC 1045, Stanford University, February 1988.

[Jacobson88a] Jacobson, V., "Congestion Avoidance and Control", SIGCOMM '88, Stanford, CA., August 1988.

[Jacobson88b] Jacobson, V., and R. Braden, "TCP Extensions for Long-Delay Paths", RFC 1072, LBL and USC/Information Sciences Institute, October 1988.

[Jacobson90a] Jacobson, V., "4BSD Header Prediction", ACM Computer Communication Review, April 1990.

[Jacobson90b] Jacobson, V., Braden, R., and Zhang, L., "TCP Extension for High-Speed Paths", RFC 1185, LBL and USC/Information Sciences Institute, October 1990.

[Jacobson90c] Jacobson, V., "Modified TCP congestion avoidance algorithm", Message to end2end-interest mailing list, April 1990.

[Jain86] Jain, R., "Divergence of Timeout Algorithms for Packet Retransmissions", Proc. Fifth Phoenix Conf. on Comp. and Comm., Scottsdale, Arizona, March 1986.

[Karn87] Karn, P. and C. Partridge, "Estimating Round-Trip Times in Reliable Transport Protocols", Proc. SIGCOMM '87, Stowe, VT, August 1987.

[McKenzie89] McKenzie, A., "A Problem with the TCP Big Window Option", RFC 1110, BBN STC, August 1989.

[Nagle84] Nagle, J., "Congestion Control in IP/TCP Internetworks", RFC 896, FACC, January 1984.

[NBS85] Colella, R., Aronoff, R., and K. Mills, "Performance Improvements for ISO Transport", Ninth Data Comm Symposium, published in ACM SIGCOMM Comp Comm Review, vol. 15, no. 5, September 1985.

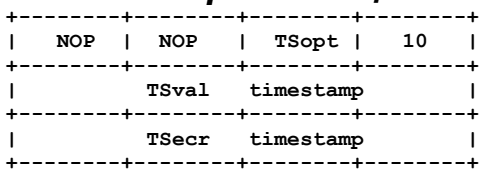
[Postel81] Postel, J., "Transmission Control Protocol – DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.

[Velten84] Velten, D., Hinden, R., and J. Sax, "Reliable Data Protocol", RFC 908, BBN, July 1984.

[Watson81] Watson, R., "Timer-based Mechanisms in Reliable Transport Protocol Connection Management", Computer Networks, Vol. 5, 1981.

[Zhang86] Zhang, L., "Why TCP Timers Don't Work Well", Proc. SIGCOMM '86, Stowe, Vt., August 1986.

Приложение A: Предложения по реализации



Показанная на рисунке схема рекомендуется для передачи опций в сегментах без флага SYN для достижения максимально возможного выравнивания на 32- и 64-битовых машинах.

Приложение B: Дубликаты из прежних инкарнаций соединения

Рассматриваются два случая: (1) системный сбой (с потерей состояния соединений) с перезапуском и (2) повторная организация соединения без потери данных о состоянии. Эти ситуации рассматриваются в отдельных параграфах.

B.1 Системный отказ с потерей состояния

Бездействие TCP в течение периода MSL используется после системного сбоя/перезапуска с потерей информации о состоянии соединений. Разъяснения приведены в параграфе «Когда нужно сохранять паузу[1]» спецификации протокола TCP [Postel81]. Продолжительность требуемого интервала MSL не зависит от скорости передачи в сети. Текущее значение TCP MSL (2 минуты) представляется разумным компромиссом, поскольку многие системы загружаются после отказа в течение примерно такого времени.

Однако для упрощения требований MSL (или обеспечения дополнительной защиты от повреждения данных) может использоваться опция временных меток. Если временные метки используются и для их значений гарантируется монотонный рост даже в случаях отказа/перезапуска системы (т. е., гарантируется, что значение первой метки отправителя после перезапуска системы будет больше значения последней метки до перезапуска), период бездействия становится необязательным.

Для полного отказа от периода бездействия нужно, чтобы часы хоста были синхронизированы с часами, которые сохраняют отсчет времени в период отказа/перезапуска системы и обеспечивают точность меток не хуже «одного тика». От этого жесткого требования можно отступить, чтобы воспользоваться преимуществами аппроксимации при синхронизации часов. Предположим, что часы всегда синхронизируются за N «тиков» и время загрузки (при необходимости дополненное периодом бездействия) составляет более N «тиков». Это будет гарантировать монотонный рост значений временных меток, что позволяет использовать метки для отбрасывания старых дубликатов даже без выполнения требований MSL.

[1] Название параграфа спецификации дано в соответствии с переводом. *Прим. перев.*

B.2 Повторная организация соединения

При завершении соединения TCP задержка 2*MSL в состоянии TIME-WAIT связывает пару сокетов на 4 минуты (см. параграф 3.5 в [Postel81]). Работающее по протоколу TCP приложение, которое закрыло соединение и открывает новое (например, при передаче данных по протоколу FTP в потоковом режиме), должно каждый раз выбирать новую пару сокетов. Задержка TIME-WAIT используется в двумя разными целями:

(a) *Реализация полнодуплексного согласования для гарантированного закрытия TCP*.

Подходящее время для задержки на этапе окончательного закрытия соединения реально не связано с MSL - оно зависит от RTO для сегментов FIN и, следовательно, от значения RTT для пути. (Можно сказать, что передавшая сегмент FIN сторона знает требуемый уровень надежности и, следовательно, она должна быть способна определить размер задержки TIME-WAIT для получателя FIN. Это можно обеспечить с помощью подходящей опции TCP в сегментах FIN.)

Хотя формально верхняя граница для RTT не задается, опыт эксплуатации сетей говорит, что значения RTT более 1 крайне маловероятны. Таким образом, 4-минутная задержка в состоянии TIME-WAIT удовлетворительно обеспечивает надежное полнодуплексное завершение TCP. Отметим еще раз, что эта задержка не зависит от требований MSL и скорости сети.

Состояние TIME-WAIT может вызывать опосредованные проблемы, если приложению требуется достаточно часто закрывать одно соединение и открывать другое, поскольку число доступных на хосте портов TCP меньше 2^{16}. Однако высокая скорость сети не является основным источником этой проблемы - RTT ограничивает скорость открытия и закрытия соединений. Следовательно, с ростом скорости сети эта проблема не усугубляется.

(b) *Обеспечить исчезновение старых дубликатов сегментов*.

Для замены этой функции состояния TIME-WAIT нужен механизм, работающий с разными соединениями. Механизм PAWS определен строго для работы в рамках одного соединения - последняя временная метка TS.Recent хранится в блоке управления соединением и отбрасывается при закрытии соединения.

В TCP может быть добавлен дополнительный механизм кэширования на уровне хоста последних временных меток, полученных из всех соединений. Кэшированные метки могут использоваться механизмом PAWS для отбрасывания старых дубликатов сегментов из предшествующих инкарнаций соединения, если гарантируется, что значение таймера временных меток увеличилось хотя бы на один «тик» с момента открытия старого соединения. Это будет требовать, чтобы за время задержки TIME-WAIT + RTT значения временных меток на стороне отправителя различались по крайней мере на «один тик». Подобное расширение не входит в число предложений данного RFC.

Отметим наличие варианта механизма, который предложили Garlick, Rom, и Postel [Garlick77], требующий от каждого хоста поддерживать для соединений записи, включающие максимальные значения порядковых номеров для этого соединения. При использовании вместо номеров временных меток достаточно сохранять одно значение на каждый удаленный хост[1], независимо от числа одновременных соединений с этим хостом.

Приложение C: Отличия от RFC 1072 и RFC 1185

Расширения протокола, определенные в этом документе, имеют ряд существенных отличий от расширений RFC 1072 и RFC 1185.

(a) Опция SACK была «отложена» во втором документе.

(b) Точные правила передачи временных меток (см. параграф 3.4) имеют ряд существенных отличий. Прежние правила могли приводить к занижению значения RTT в некоторых ситуациях (отбрасывание и нарушение порядка достаки).

(c) Одно значение TS.Recent сейчас используется двумя разными механизмами RTTM и PAWS. Это упрощение стало возможным в результате изменения (b).

(d) Устранена неоднозначность RFC 1185 в пользу включения временных меток в сегменты ACK, наряду с сегментами данных. Это поддерживает симметрию протокола TCP.

(e) Опции echo и echo reply из RFC 1072 объединены в одну опцию Timestamps в целях сохранения симметрии и упрощения обработки.

(f) Решена проблема устаревших временных меток для соединений с продолжительным бездействием (параграф 4.2.2).

(g) В RFC 1185 предлагается сначала использовать предсказание заголовка, а потом - проверку временной метки. На основе некоторого скептицизма в части вероятностных аргументов, приведенных в параграфе 4.2.4, принято решение об изменении порядка операций.

(h) Спецификация изменена так, что опции расширения будут передаваться в сегментах SYN,ACK только в тех случаях, когда эти опции были получены в соответствующих сегментах SYN. Это обеспечивает наиболее консервативный подход к вопросам взаимодействия с реализациями, которые не поддерживают расширений.

В дополнение к существенным изменениям данный RFC пытается однозначно задать спецификации алгоритмов путем представления изменений в правила обработки событий из RFC 793 (см. Приложение E).

Приложение D: Использованные обозначения

Далее описаны обозначения, использованные в данном документе.

Опции

WSopt - опция масштабирования окна TCP.
TSopt - опция TCP Timestamps (временные метки)

[1] С которым организовано соединение TCP. *Прим. перев.*

Поля опций

shift.cnt - байт масштабирования размера окна в WSopt.
TSval - 32-битовое значение поля Timestamp Value в TSopt.
TSecr - 32-битовое значение поля Timestamp Reply в TSopt.

Поля опций в текущем сегменте

SEG.TSval - поле TSval из TSopt в текущем сегменте.
SEG.TSecr - поле TSecr из TSopt в текущем сегменте.
SEG.WSopt - 8-битовое значение в WSopt

Значения таймеров

my.TSclock - 32-битовые значения временных меток из локального источника.
my.TSclock.rate - период my.TSclock (от 1 мсек до 1 сек).

Переменные состояния соединений

TS.Recent - полученная последней временная метка.
Last.ACK.sent - переданное последним поле ACK.

Snd.TS.OK - 1-битовый флаг.
Snd.WS.OK - 1-битовый флаг.

Rcv.Wind.Scale - коэффициент масштабирования (показатель степени) окна приема.
Snd.Wind.Scale - коэффициент масштабирования (показатель степени) окна передачи.

Приложение E: Обработка событий

Вызов OPEN

. . .

Выбирается начальное значение порядкового номера для передачи (ISS). Передается сегмент SYN вида:

```
<SEQ=ISS><CTL=SYN><TSval=my.TSclock><WSopt=Rcv.Wind.Scale>
```

. . .

Вызов SEND

Состояние CLOSED (TCB не существует)

...

Состояние LISTEN

Если задан удаленный сокет, пассивное соединение заменяется активным и выбирается ISS. Передается сегмент SYN, содержащий опции Tsval=my.TSclock и WSopt=Rcv.Wind.Scale. Устанавливается значение SND.UNA для ISS, SND.NXT для ISS+1.

Переход в состояние SYN-SENT. ...

Состояние SYN-SENT

Состояние SYN-RECEIVED

...

Состояние ESTABLISHED

Состояние CLOSE-WAIT

Буфер сегментируется и передается с прицепленным подтверждением (значение подтверждения = RCV.NXT). ...

Если установлен флаг срочности (urgent) ...

Если установлен флаг Snd.TS.OK, опция TCP Timestamps Tsval=my.TSclock,TSecr=TS.Recent включается включается в каждый сегмент.

Масштабируется окно приема в заголовке сегмента для передачи:

```
SEG.WND = (SND.WND >> Rcv.Wind.Scale).
```

Прибытие сегмента

...

Если состояние LISTEN, то

сначала проверить флаг RST

...

затем проверить флаг ACK

...

в-третьих, проверить флаг SYN

Если флаг SYN установлен, проверить security. Если ...

...

Если SEG.PRC < TCB.PRC, продолжить.

Проверить опцию масштабирования окна (Wsopt). Если она присутствует, сохранить SEG.WSopt в Snd.Wind.Scale и установить флаг Snd.WS.OK. В противном случае установить нулевые значения Snd.Wind.Scale и Rcv.Wind.Scale, а также сбросить флаг Snd.WS.OK.

Проверить опцию Tsopt. Если она присутствует, сохранить SEG.TSval в переменной TS.Recent и установить флаг Snd.TS.OK.

Установить RCV.NXT = SEG.SEQ+1, IRS = SEG.SEQ, прочие элементы управления и текст помещаются в очередь для последующей обработки. Следует выбрать значение ISS и передать сегмент SYN вида:

```
<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>
```

Если установлен флаг Snd.WS.OK, включить в сегмент опцию WSopt=Rcv.Wind.Scale. Если установлен флаг Snd.TS.OK, включить в сегмент опцию TSopt с TSval=my.TSclock,TSecr=TS.Recent. Установить Last.ACK.sent = RCV.NXT.

Установить SND.NXT = ISS+1 и SND.UNA = ISS. Состояние соединения следует сменить на SYN-RECEIVED. Отметим, что все прочие элементы управления и данные (комбинируемые с SYN) обрабатываются в состоянии SYN-RECEIVED, но повторять обработку SYN и ACK не следует. Если прослушивание не было задано полностью (т. е., не полностью задан удаленный сокет), следует заполнить незаданные поля.

В-четвертых, прочие элементы управления и текст

...

Если состояние SYN-SENT, то

сначала проверить флаг ACK

...

в-четвертых, проверить флаг SYN

...

Если флаг SYN установлен, а поля security/compartment[1] и precedence[2] имеют приемлемые значения, устанавливается RCV.NXT = SEG.SEQ+1, IRS = SEG.SEQ, а любые сегменты из очереди на повтор, которые в связи с этим становятся подтвержденными, следует удалить.

Проверить опцию масштабирования окна (Wsopt). Если опция присутствует, сохранить SEG.WSopt в Snd.Wind.Scale, в противном случае установить нулевые значения для Snd.Wind.Scale и Rcv.Wind.Scale.

Проверить опцию Tsopt. Если она присутствует, сохранить SEG.TSval в переменной TS.Recent и установить флаг Snd.TS.OK в блоке управления соединением. Если установлен флаг ACK, используется значение my.TSclock - SEG.TSecr в качестве начальной оценки RTT.

Если SND.UNA > ISS (сегмент SYN подтвержден), сменить состояние соединения на ESTABLISHED, сформировать и передать сегмент ACK:

```
<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>
```

Если установлен флаг Snd.Echo.OK, включить в сегмент ACK опцию Tsopt с Tsval=my.TSclock,TSecr=TS.Recent. Установить Last.ACK.sent = RCV.NXT.

Можно включить данные и элементы управления, размещенные в очереди на передачу. Если в сегментеимеются другие элементы управления или данные, их обработка будет продолжена на шестом этапе, описанном ниже, где проверяется флаг URG. В остальных случаях происходит возврат управления.

Иначе осуществляется переход в состояние SYN-RECEIVED, формируется и передается сегмент SYN,ACK:

```
<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>
```

Если установлен флаг Snd.Echo.OK в сегмент включается опция Tsopt с Tsval=my.TSclock,TSecr=TS.Recent. Если установлен флаг Snd.WS.OK, в сегмент включается опция WSopt=Rcv.Wind.Scale. Устанавливается Last.ACK.sent = RCV.NXT.

Если в сегменте имеются другие элементы управления или данные, их обработка продолжается после перехода в состояние ESTABLISHED. Осуществляется возврат управления.

В-пятых, если ни один из флагов SYN и RST не установлен, сегмент отбрасывается с возвратом управления.

В противном случае

Сначала проверяется порядковый номер

Состояние SYN-RECEIVED

Состояние ESTABLISHED

Состояние FIN-WAIT-1

Состояние FIN-WAIT-2

Состояние CLOSE-WAIT

Состояние CLOSING

Состояние LAST-ACK

Состояние TIME-WAIT

Сегменты обрабатываются по порядку. Начальные проверки служат для отбрасывания старых дубликатов и дальнейшая обработка осуществляется в соответствии со значениями SEG.SEQ. Если содержимое сегмента относится к старой и новой части, обрабатывать следует только новую часть.

Масштабируется поле окна приема:

```
TrueWindow = SEG.WND << Snd.Wind.Scale,
```

[1]Безопасность/разделение.
[2]Предпочтения.

и значение TrueWindow используется на последующих этапах вместо SEG.WND.

Проверяется наличие в сегменте опции Timestamps и установка флага Snd.TS.OK. Если опция есть и флаг установлен:

Если SEG.TSval < TS.Recent, проверяется, не бездействовало ли соединение более 24 дней. Если то и другое справедливо, сегмент является неприемлемым и для него выполняются описанные ниже действия.

Если SEG.SEQ = Last.ACK.sent, значение SEG.ECopt сохраняется в переменной TS.Recent.

Существует четыре случая при проверке приемлемости входящего сегмента:

...

Если входящий сегмент неприемлем, в ответ следует передать подтверждение (если установлен флаг RST, сегмент отбрасывается с возвратом управления):

```
<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>
```

Для Last.ACK.sent устанавливается значение SEG.ACK из подтверждения. Если установлен флаг Snd.Echo.OK, в сегмент ACK включается опция Timestamps с TSval=my.TSclock,TSecr=TS.Recent. Устанавливается Last.ACK.sent = SEG.ACK и передается сегмент ACK. После передачи подтверждения неприемлемый сегмент отбрасывается с возвратом управления.

...

В-пятых, проверяется поле ACK.

Если бит ACK не установлен, сегмент отбрасывается с возвратом управления.

Если флаг ACK установлен

...

Состояние ESTABLISHED

Если SND.UNA < SEG.ACK =< SND.NXT, устанавливается SND.UNA <- SEG.ACK. Рассчитывается новое значение периода кругового обхода. Если установлен флаг Snd.TS.OK, используется my.TSclock - SEG.TSecr; в противном случае используется оценка времени, прошедшего с момента передачи первого сегмента из очереди на повтор. Все сегменты из очереди повтора в результате считаются полностью подтвержденными ...

...

В-седьмых, обрабатываются данные сегмента.

Состояние ESTABLISHED

Состояние FIN-WAIT-1

Состояние FIN-WAIT-2

...

Передается подтверждение вида:

```
<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>
```

Если установлен флаг Snd.TS.OK в сегмент ACK включается опция Timestamps с TSval=my.TSclock,TSecr=TS.Recent. Для Last.ACK.sent устанавливается значение SEG.ACK из подтверждения и подтверждение передается. Это подтверждение следует прицеплять к передаваемому сегменту, если это не приведет к недопустимой задержке подтверждения.

...

Вопросы безопасности

В данном документе вопросы безопасности не рассматриваются.

Адреса авторов

Van Jacobson
University of California
Lawrence Berkeley Laboratory
Mail Stop 46A
Berkeley, CA 94720
Phone: (415) 486-6411
EMail: van@CSAM.LBL.GOV

Bob Braden
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
Phone: (310) 822-1511
EMail: Braden@ISI.EDU

Dave Borman
Cray Research
655-E Lone Oak Drive
Eagan, MN 55121
Phone: (612) 683-5571
Email: dab@cray.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru