Модел. и анализ информ. систем. Т. **19**, № **4** (2012) 37–47

УДК 004.728.5

Сравнительный анализ производительности транспортных протоколов Trickles и TCP в условиях высокой нагрузки на коммуникационную сеть

Никитинский М.А.[1], Чалый Д.Ю.[2]

Ярославский государственный университет им. П.Г. Демидова

e-mail: nikitinskiy@gmail.com, chaly@uniyar.ac.ru

получена 4 сентября 2012

Ключевые слова: транспортный протокол, моделирование, анализ производительности

Исследуется производительность протокола Trickles, который предназначен для управления передачей на транспортном уровне архитектуры TCP/IP. Для этого была построена имитационная модель протокола в системе ns-2, определены характеристики производительности, которые были экспериментально измерены на моделях сетевых инфраструктур.

1. Введение

Современный мир невозможно представить без коммуникационных сетей, чья гибкая инфраструктура позволяет разрабатывать и внедрять новые сетевые приложения, взрывной рост количества которых наблюдается в последние годы. Производительность и безопасность сетевых приложений зависит от согласованной работы коммуникационных протоколов, в совокупности образующих коммуникационную архитектуру. В настоящее время наиболее распространенной является архитектура TCP/IP, которая используется в сети Интернет. С точки зрения этой архитектуры все узлы сети можно разделить на два класса: конечные, генерирующие и потребляющие сетевой трафик, а также промежуточные, занимающиеся передачей этого трафика. Протоколы транспортного уровня, в частности протокол TCP (Transmission Control Protocol), представляют собой важный элемент этой архитектуры и решают задачу эффективной передачи данных между конечными узлами сети. Начальная спецификация протокола TCP была опубликована в 1981 году в [2], и с тех пор этот протокол является центральным объектом многих актуальных исследований

[1]Работа поддержана грантом РФФИ № 12-01-00281-а; Минобрнауки РФ в рамках Госзадания вузу на выполнение НИР №8.5370.2011.

[2]Работа поддержана грантом РФФИ № 11-07-00549-а; Минобрнауки РФ в рамках Госзадания вузу на выполнение НИР №8.5370.2011.

в области сетевых технологий, научные результаты которых привели к созданию новых сетевых стандартов [3, 4, 6, 5, 8, 9, 10, 11, 7, 12].

В целом большинство проведенных за все время исследований можно охарактеризовать как усовершенствование алгоритмов работы транспортного протокола без существенных внесений изменений в логическую структуру функциональных элементов этого протокола. Однако в последние годы в связи с внедрением мобильных устройств, высоконагруженных и облачных систем, а также сенсорных сетей возникает задача эффективной работы протокола непосредственно на конечных узлах сети, так как они могут обладать ограниченными ресурсами (например, по объему оперативной памяти и энергопотреблению), либо иметь нетривиальную внутреннюю организацию (например, в случае облачных систем). Таким образом возникает задача модификации транспортного протокола с целью его более эффективной работы непосредственно на конечных узлах сети, по возможности, без существенного снижения производительности по передаче данных. Одним из подходов к решению этой задачи является протокол Trickles [1], в котором управляющие параметры соединения хранятся только на одном конечном узле сети, тогда как для функционирования стандартного TCP необходимо их хранение на обоих узлах сети. Таким образом, мы получаем протокол, где основная нагрузка по хранению и управлению параметрами транспортного соединения находится на одной стороне соединения, в данном случае — клиенте. Следовательно, облегчается функционирование высоконагруженных серверов, что было показано в [1]. Однако в этой работе не был проведен всесторонний анализ производительности протокола Trickles, что является важной задачей при разработке любого транспортного протокола. Целью настоящей статьи является исследование производительности протокола Trickles и его сравнительный анализ с другими стандартными протоколами. Для решения этой задачи мы используем методы имитационного моделирования и прикладной пакет ns-2 [15].

2. Транспортный протокол Trickles

1. Сравнительный анализ моделей работы протоколов TCP и Trickles

Главной задачей протокола TCP является надежная и эффективная передача данных между конечными системами через ненадежную среду передачи — коммуникационную сеть, которая может терять, переупорядочивать и искажать передаваемые данные. Для простоты будем считать, что передача ведется в одном направлении, и будем называть *сервером* сторону, передающую данные, а *клиентом* — принимающую. При этом каждый передаваемый байт данных уникально пронумерован возрастающей последовательностью чисел. Надежность передачи данных обеспечивается при помощи механизма кумулятивных подтверждений — успешное получение каждой порции данных (которая также называется сегментом), переданных сервером, должно быть подтверждено клиентом. В том случае, если за определенное время подтверждение не пришло, сервер повторно передает данные [2, 7]. Эффек-

тивность передачи заключается в том, чтобы использовать все доступные ресурсы коммуникационной сети, не допуская ее излишних простоев и, по возможности, ее перегрузки. Для этого сервер при помощи алгоритма управления потоком [3, 5] пытается определить объем данных, который может находиться в сети в транзите. При этом клиент и сервер работают согласованно и для реализации протокола TCP на обоих концах соединения хранится структура данных, которая называется TCB (Transmission Control Block), где хранятся необходимые параметры (рис. 1а). Таким образом, можно говорить, что состояние транспортного соединения распределено между сервером и клиентом и состоит из TCB сервера и TCB клиента. При этом можно представить формальную модель каждого из концов соединения в виде системы переходов [17], которая изменяет эти параметры и управляет передачей данных и подтверждений.

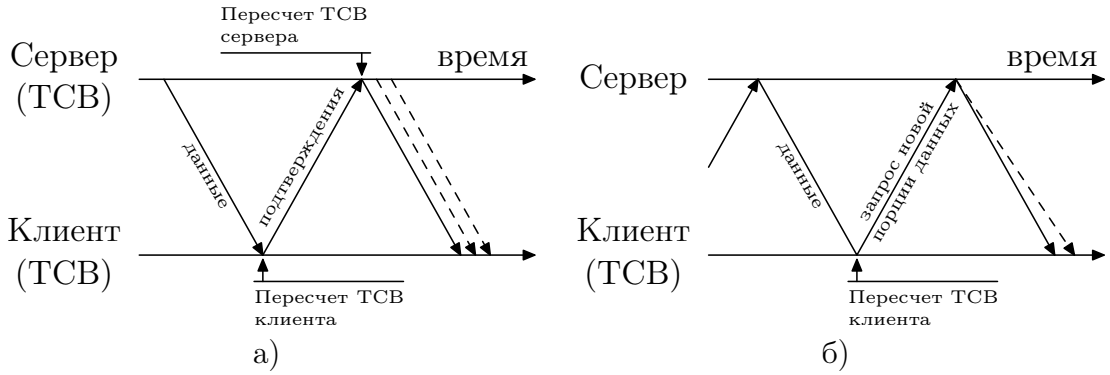


Рис. 1. Модели работы транспортных протоколов: а) TCP, б) Trickles

Основной идеей, лежащей в основе протокола Trickles, является перенос всех управляющих соединением параметров на сторону клиента, освобождая сервер от хранения этих параметров. При этом модель клиента все так же можно представить как систему переходов, а модель сервера становится функцией, при помощи которой рассчитывается очередное действие и ответ сервера. Параметром этой функции является содержимое пришедшего сегмента данных. По сравнению с сегментом TCP, сегмент протокола Trickles несет больше информации, в частности в нем хранятся управляющие параметры алгоритма управления потоком. Это реализуется при помощи опциональных полей сегмента протокола TCP [2]. Что касается процесса передачи данных, то в отличие от протокола TCP, где активной стороной является сервер, в протоколе Trickles инициативу проявляет клиент, отправляя запросы серверу с требованием очередной порции данных, а сервер лишь обслуживает такие запросы (рис. 1б). Так как в модели протокола Trickles состояние транспортного соединения постоянно хранится только на одном конце соединения, то такой класс протоколов мы называем *асимметричными* протоколами или протоколами с *нераспределенным состоянием* соединения.

Такое изменение модели работы транспортного протокола имеет ряд преимуществ [1]. Так как в сегменты включена вся необходимая информация о соединении, то серверная часть может быть реплицирована между несколькими физическими

устройствами. В беспроводных сетях возможен оперативный переброс транспортного соединения с одного устройства на другое, что является непростой задачей при использовании классического протокола TCP. Другим преимуществом является повышенная устойчивость к DoS-атакам (DoS — Denial of Service), целью которых является исчерпание оперативной памяти сервера путем установки большого количества полуоткрытых соединений.

2. Алгоритм работы протокола Trickles

В предыдущем разделе мы описали модель транспортного протокола Trickles, которая представляет собой отправку клиентом запросов серверу и возврат последним требуемых данных. Очевидно, что для того чтобы такой протокол был эффективен, в транзите, в зависимости от состояния коммуникационной сети, должно быть несколько таких запросов. В этом разделе мы опишем алгоритмы, которые используются протоколом Trickles для обеспечения такой работы.

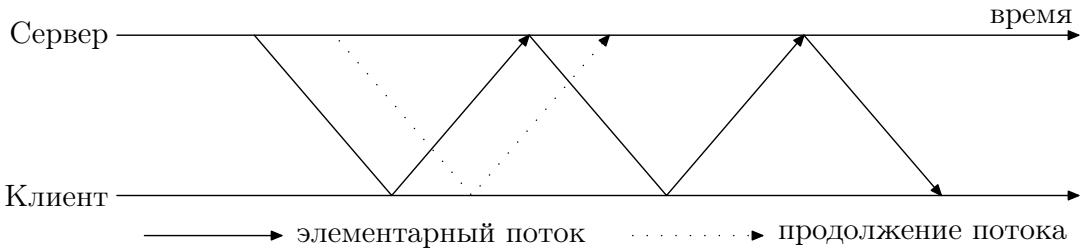


Рис. 2. Концепции продолжения и элементарного потока

Итак, Trickles-соединение состоит из запросов к серверу и ответов клиенту, причем одновременно может выполняться несколько таких запросов. Мы будем называть *продолжением потока* (continuation) сегмент, следующий от сервера к клиенту и обратно. Последовательность продолжений потока образует *элементарный поток* (trickle) (рис. 2). Таким образом, можно говорить, что в процессе передачи данных происходит параллельная работа нескольких элементарных потоков (рис. 3). Во время транзита в сети каждый элементарный поток независим от других элементарных потоков, их синхронизация происходит только на стороне клиента.

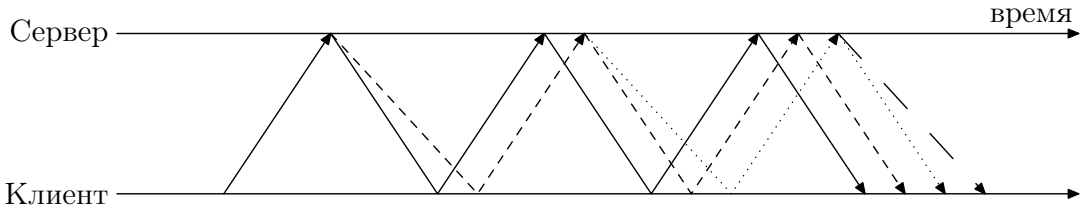


Рис. 3. Параллельный характер соединения протокола Trickles

Опишем алгоритм управления потоком протокола Trickles, так как именно эта часть протокола в большей степени влияет на его производительность. Этот алгоритм работает отдельно для каждого элементарного потока в том смысле, что

когда сегмент, принадлежащий некоторому потоку, находится в транзите, в нем хранятся все необходимые параметры. При этом именно серверная часть протокола преобразует эти параметры и предпринимает дальнейшие действия по управлению элементарными потоками. Когда сегмент приходит на сервер, существует три возможных варианта развития событий: сервер отправляет один сегмент с данными в ответ на запрос, продолжая поток; сервер увеличивает количество элементарных потоков; сервер уничтожает поток, ничего не отправляя в ответ на запрос. Так как один элементарный поток в сети представлен одним сегментом, то количество работающих элементарных потоков представляет собой текущую оценку протоколом пропускной способности сети.

При создании алгоритма управления потоком протокола Trickles [1] авторы попытались сделать его соответствующим алгоритму TCP Reno [5]. Последний алгоритм для управления потоком использует на стороне сервера две переменные: $cwnd$, которая содержит текущую оценку пропускной способности сети, и $ssthresh$, которая используется для определения характера роста первой переменной — экспоненциальный (если $cwnd < ssthresh$) или линейный (в противном случае). Так как на стороне сервера Trickles никаких параметров не хранится, то для расчета значения $cwnd$ для k-го сегмента используется следующая формула [1]:

$$TCPCwnd(k) = \begin{cases} startcwnd + (k - TCPBase) & \text{если } k < A, \\ ssthresh & \text{если } A \leqslant k < A + ssthresh, \\ F(k - A) & \text{если } A + ssthresh \leqslant k \end{cases}$$

где $startcwnd$ — начальное значение $cwnd$; $TCPBase$ — номер сегмента, когда последний раз переопределялось $startcwnd$;

$$A = ssthresh - startcwnd + TCPBase$$

и $F(N)$ — это наибольшее целое, меньшее положительного значения x, который является корнем уравнения

$$\frac{(x-1)x - (ssthresh - 1)sshtresh}{2} - N = 0.$$

При этом параметры $startcwnd$, $TCPBase$ и $ssthresh$ передаются в сегменте TCP как опциональные поля.

Так же как и алгоритм управления потоком протокола TCP Reno, протокол Trickles начинает свою работу с этапов медленного старта (slow start) и предотвращения перегрузки (congestion avoidance). Во время этих этапов по пришествии сегмента с номером k происходит следующее:

1. Рассчитывается значение $CwndDelta = TCPCwnd(k) - TCPCwnd(k-1)$.

2. Отправляется $CwndDelta + 1$ сегментов с данными: продолжается оригинальный элементарный поток и создается $CwndDelta$ новых. Сервер предполагает, что k сегментов были успешно получены и $TCPCwnd(k-1)$ еще находятся в пути, таким образом, сервер отправляет сегменты начиная с номера $k + TCPCwnd(k-1)$.

В случае потерь сегментов протокол TCP Reno использует алгоритмы быстрой повторной передачи и восстановления (fast retransmit, fast recovery). Для этого используются дублированные подтверждения (duplicate acknowledgements), по пришествии определенного количества которых определяется потерянный сегмент, который передается, не дожидаясь срабатывания таймера повторной передачи. Протокол Trickles для определения потерянных сегментов использует SACK-блоки [4, 6]. Для сегмента с номером k, пришедшего на сервер во время режима быстрой повторной передачи и восстановления, протоколом Trickles выполняются следующие действия:

1. Пусть $firstLoss$ — это номер первого потерянного сегмента, тогда вычисляются следущие параметры:

$$cwndAtLoss = TCPCwnd(firstLoss - 1)$$

$$lossOffset = k - firstLoss$$

$$newCwnd = TCPCwnd(k)/2$$

   Здесь $cwndAtLoss$ — значение $cwnd$ при потере, $lossOffset$ — количество сегментов с момента потери, $newCwnd$ — новое значение $cwnd$.

2. Целью режима быстрой повторной передачи и восстановления является уничтожение $n = cwndAtLoss - newCwnd$ элементарных потоков и продолжение $newCwnd$ элементарных потоков. Это делается при помощи следующего алгоритма: (а) Если $cwndAtLoss - lossOffset + 1 \leqslant newCwnd$, то элементарный поток продолжается, иначе — прекращается. (б) Если k-й сегмент следует непосредственно после потерянных, то мы генерируем все потерявшиеся элементарные потоки, удовлетворяющие условию (а).

После выхода из режима быстрой повторной передачи и восстановления происходит обновление параметров: $TCPBase$ указывает на точку восстановления, а $ssthresh = startCwnd = newCwnd$.

В случае множественных потерь, то есть ситуации, когда теряется более одного сегмента, используются повторные передачи, которые инициируются срабатыванием таймера повторной передачи. Протоколом Trickles таймер устанавливается на стороне клиента согласно стандарту [7]. При срабатывании таймера повторной передачи или определении множественной потери сервер совершает следующие действия:

1. Устанавливаем новые значения $ssthresh$ и $cwnd$:

$$ssthresh = TCPCwnd(firstLoss - 1)/2$$

$$cwnd = InitialCwnd,$$

   где $InitialCwnd$ — это начальное значение $cwnd$.

2. Уничтожаем все элементарные потоки, оставляем только $cwnd$ элементарный поток, для того чтобы перейти в режим медленного старта.

3. Моделирование и анализ протокола Trickles

Основными методами исследования производительности транспортных протоколов являются имитационное моделирование, либо натурные эксперименты. Нами был выбран метод имитационного моделирования, так как существует пакет ns-2 [15], в котором есть большое количество моделей различных версий протокола TCP и который является стандартом де-факто для проведения таких исследований.

Система ns-2 [15] является объектно-ориентированным ПО, ядро которого реализовано на языке C++, а исследуемые модели коммуникационных сетей описываются на языке OTcl. Использование двух языков программирования объясняется тем, что, с одной стороны, модели должны быстро выполняться, что достигается при помощи использования языка C++, а с другой стороны, наличие интерпретируемого языка OTcl позволяет быстро разрабатывать эти модели.

Центральной концепцией системы ns-2 является агент, который является сущностью, выполняющейся на узле сети. В наших экспериментах в роли агента выступала разработанная модель протокола Trickles. Типичным сценарием построения модели протокола в системе ns-2 является следующая процедура:

- Добавить модель пакетов, которые используются протоколом для обмена данными.
- Написать класс агента, который моделирует работу протокола. Для простоты значительное количество транспортных протоколов в системе ns-2 моделируется при помощи двух агентов: один исполняет роль сервера, а второй — клиента.
- Предыдущие два пункта реализуются на языке C++. Чтобы разработанный агент стал доступен для построения исследуемых коммуникационных сетей, необходимо произвести его привязку к языку OTcl.

Создавая структуру пакета, мы опирались на вариант, предложенный в [1].

В системе ns-2 стандартный класс агента *Agent* включает в себя виртуальные методы, которые моделируют обработку пришедших пакетов и тайм-ауты. Для серверной части агент реализуется переопределением метода *Agent::recv()*, который обрабатывает пришедший запрос от клиента и сразу же отправляет ответ. Для реализации функционала клиента переопределяется этот же метод, а также метод *Agent::timeout()* для моделирования тайм-аута повторной передачи [7].

1. Результаты экспериментов

На основании анализа работы [1] нами была создана оригинальная модель протокола Trickles. Эта модель использовалась для построения модели коммуникационной инфраструктуры, показанной на рис. 4. Данная инфраструктура содержит два промежуточных устройства $N0$ и $N1$, максимальная длина очереди на которых равна 50 сегментам. Размер сегмента предполагается равным 1540 байтов. Информационный трафик передается от узлов, помеченных как *server*, к узлам, помеченным как *client*. В обратном направлении передаются подтверждения полученных данных.

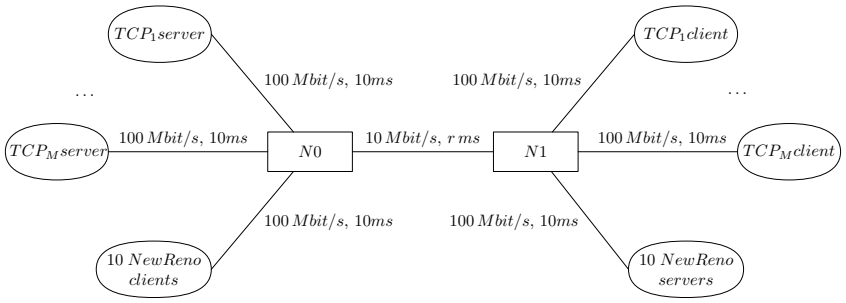


Рис. 4. Исследуемая конфигурация сети

В каждом эксперименте исследуется производительность экземпляров протокола, обозначенных $TCP_1server \ldots TCP_Mserver$. При этом в качестве конкретных экземпляров протокола TCP используются встроенные в пакет ns-2 модели протоколов TCP Reno [13], TCP SACK [4, 6, 13], TCP Vegas [14], TCP NewReno [10], которые сравнивались с созданной моделью протокола Trickles. В рамках каждого эксперимента моделируется передача данных в течение 600 секунд. При этом M экземпляров протокола не только конкурируют друг с другом за сетевые ресурсы, в сети присутствует также сторонний трафик, затрудняющий передачу подтверждений, который генерируется десятью экземплярами протокола NewReno. Они включаются одновременно и работают в периоды со 150-й до 300-й секунды, а также с 450-й по 600-ю секунду. В данной серии экспериментов изменяются два параметра. Первый параметр — количество одновременно работающих экземпляров исследуемого протокола, M, изменяется от 10 до 50 с шагом 1. Вторым параметром является значение задержки передачи r для канала между устройствами $N0$ и $N1$, который изменяется от 10 до 120 мс с шагом 10 мс.

Для каждого эксперимента было рассчитано две характеристики производительности. Характеристика goodput показывает, какой объем данных был передан от сервера к клиенту, и для одного соединения рассчитывается по формуле

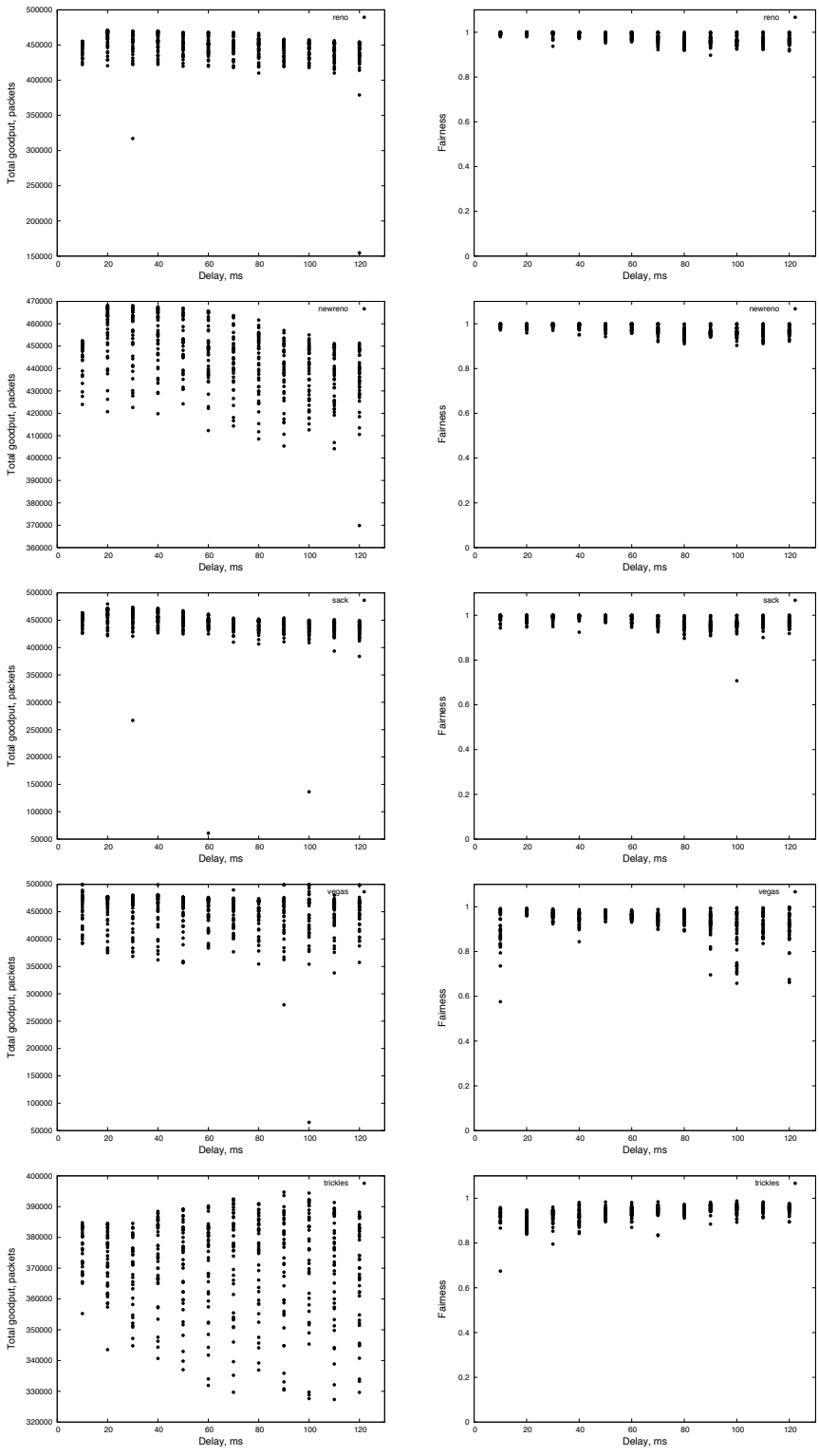
$$goodput = \frac{send\_data - retransmitted\_data}{t},$$

здесь $send\_data$ — это общее количество сегментов, которые были переданы протоколом, $retransmitted\_data$ — это количество повторных передач, а t — общее время эксперимента. Второй характеристикой является справедливость использования общих ресурсов по Джейну [16], которая определяется по формуле

$$J_{FI} = \frac{(\sum_{i=1}^{M} b_i)^2}{M \times \sum_{i=1}^{M} b_i^2},$$

где b_i — это goodput i-го соединения, а M — количество транспортных соединений, которое разделяет общий ресурс.

Результаты экспериментов показаны на рис. 5. В качестве goodput на графиках приведен суммарный goodput всех работавших в процессе эксперимента соединений в зависимости от задержки передачи канала между узлами $N0$ и $N1$. Видно, что



TCP Reno

TCP NewReno

TCP SACK

TCP Vegas

Trickles

Рис. 5. Результаты экспериментов. Слева — совокупный goodput соединений, справа — индекс справедливости по Джейну

протокол Trickles передавал сравнимое с остальными протоколами количество данных. Также был посчитан индекс справедливости по Джейну, который показывает, что при работе нескольких протоколов происходит достаточно честное разделение ресурсов между всеми работающими соединениями. Таким образом, результаты экспериментов показывают, что протокол Trickles является достаточно эффективным транспортным протоколом.

Отсюда можно сделать вывод, что, несмотря на существенную разницу в логической организации транспортного соединения, протокол Trickles является достаточно эффективным по сравнению со стандартными версиями протокола TCP.

4. Заключение

В результате работ была создана оригинальная имитационная модель протокола Trickles, а также накоплен и проанализирован значительный объем экспериментальных данных. Несмотря на существенную разницу в логической организации транспортного соединения, транспортный протокол Trickles является достаточно эффективным по сравнению со стандартными версиями протокола TCP. Однако проведенные эксперименты указали на некоторые недостатки протокола Trickles, главным из которых является его нестабильность. Эта нестабильность проявляется в сильной зависимости эффективности его работы от условий коммуникационной сети. Таким образом, дальнейшая работа будет посвящена исправлению этого недостатка.

Список литературы

1. Shieh, A., Myers, A. C., Sirer, E. G. A stateless approach to connection-oriented protocols // ACM Trans. Comput. Syst. 26, 3, Article 8 (September 2008), 50 pages.

2. Transmission Control Protocol. DARPA Internet Program. Protocol Specification // RFC793,September, 1981. Web site: www.rfc-editor.org

3. Jacobson V., Braden R., Borman D. TCP Extensions for High Performance // RFC1323,May, 1992. Web site: www.rfc-editor.org

4. Mathis M., Mahdavi J., Floyd S., Romanow A. TCP Selective Acknowledgement Options // RFC2018, October, 1996. Web site: www.rfc-editor.org

5. Allman M., Paxson V., Stevens W. TCP Congestion Control // RFC2581, April, 1999. Website: www.rfc-editor.org

6. Floyd S., Mahdavi J., Mathis M., Podolsky M. An Extension to the Selective Acknowledgement (SACK) Option for TCP // RFC2883, July, 2000. Web site: www.rfc-editor.org

7. Paxson V., Allman M. Computing TCP's Retransmission Timer // RFC2988, November, 2000. Web site: www.rfc-editor.org

8. Allman M., Balakrishnan H., Floyd S. Enhancing TCP's Loss Recovery Using Limited Transmit // RFC3042, January, 2001. Web site: www.rfc-editor.org

9. Allman M., Floyd S., Partridge C. Increasing TCP's Initial Window // RFC3390, October,2002. Web site: www.rfc-editor.org

10. Floyd S., Henderson T., Gurtov A. The NewReno Modification to TCP's Fast Recovery Algorithm // RFC3782, April, 2004. Web site: www.rfc-editor.org

11. V. Paxson, M. Allman, J. Chu, M. Sargent. Computing TCP's Retransmission Timer // RFC6298, June, 2011. Web site: www.rfc-editor.org

12. J. Touch, A. Mankin, R. Bonica. The TCP Authentication Option // RFC5925, June, 2010. Web site: www.rfc-editor.org

13. K. Fall, S. Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP // Computer Communications Review, Vol. 26, Issue 3, July, 1996, p. 5–21.

14. S. Low, L. Peterson, L. Wang. Understanding TCP Vegas: A Duality Model // Tech. Rep. TR-628-00, Princeton University, November, 2000, 22 pages.

15. S. McCanne and S. Floyd. ns Network Simulator // http://www.isi.edu/nsnam/ns/

16. Jain R. The art of computer systems performance analysis. // Jon Wiley and Sons, 1991.

17. D.Ju. Chaly, V.A. Sokolov. An Extensible Coloured Petri Net Model of a Transport Protocol for Packet Switched Networks // Proceedings of PACT'2003, LNCS Vol. 2763, Springer, 2003. p. 66–75

Performance Analysis of the Transport Protocols Trickles and TCP under High-load Network Conditions

Nikitinskiy M.A., Chalyy D.Ju.

Keywords: transport protocol, modeling, performance analysis

We study the performance of the Trickles protocol which is a transport-layer protocol for the TCP/IP architecture. We have implemented the original model of the Trickles protocol for the ns-2 simulator and defined performance metrics which were measured using ns-2 simulations.

Сведения об авторах: **Никитинский Михаил Александрович**, Ярославский государственный университет им. П.Г. Демидова, аспирант
Чалый Дмитрий Юрьевич, Ярославский государственный университет им. П.Г. Демидова, канд. физ.-мат. наук, доцент