Информационные процессы, Том 16, № 4, 2016, стр. 362–374

ПЕРЕДАЧА ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СЕТЯХ

Исследование передачи Web-данных в сотовых сетях

А.А. Белогаев, Н.С. Жирнов, А.Н. Красилов, А.И. Ляхов, Е.М. Хоров

Институт проблем передачи информации им. А.А. Харкевича РАН, Москва, Россия

Поступила в редколлегию 30.11.2016

Аннотация—Большинство современных интернет-приложений, генерирующих Web-трафик, используют для его доставки протокол TCP. Ключевым компонентом этого протокола является алгоритм управления окном перегрузки, благодаря которому протокол пытается, с одной стороны, полностью использовать доступную пропускную способность сети, а с другой — предотвратить на промежуточных сетевых узлах перегрузки, которые ведут к потере пакетов. Изначально такие алгоритмы разрабатывались для проводных сетей, однако повсеместное использование сетей LTE и Wi-Fi в качестве сетей доступа к Интернету обусловило высокий интерес исследователей к методам доставки данных в беспроводных сетях. В данной работе мы исследуем, как различные алгоритмы управления окном перегрузки протокола TCP и число параллельных соединений TCP, установленных между одной и той же парой клиент-сервер, влияют на передачу Web-данных в сотовых сетях.

КЛЮЧЕВЫЕ СЛОВА: Беспроводная сеть, веб-трафик, TCP, LTE, алгоритм управления окном перегрузки.

1. ВВЕДЕНИЕ

Повсеместное использование беспроводных сетей в качестве сетей доступа к Интернету обусловило высокий интерес исследователей к методам доставки данных в таких сетях. По сравнению с проводными сетями, для беспроводных сетей характерны меньшая надежность доставки, низкая пропускная способность, большая и сильно меняющаяся со временем задержка, обусловленная как ожиданием в очереди, так и механизмом повторных попыток передачи, используемым практически в любой современной технологии беспроводных сетей. Указанные недостатки приводят к тому, что загрузка Web-страниц происходит слишком долго, и в результате пользователи Интернета оказываются недовольны качеством оказания услуги доступа в Интернет, что особенно критично для операторов сотовых сетей.

Улучшить качество передачи данных можно, как усовершенствуя правила обслуживания трафика на базовой станции сотовых сетей (см., например, [1,2]), так и модифицируя алгоритмы, управляющие генерацией потока данных на сервере. Учитывая, что значительная часть трафика (в том числе и Web-трафик) представляет собой потоки TCP (англ.: Transmission Control Protocol), в последние десятилетия было разработано множество модификаций TCP, повышающих его эффективность в беспроводных сетях. Эти модификации отличаются прежде всего видом алгоритма управления окном перегрузки, направленным, с одной стороны, на полное использование доступной пропускной способности сети, а с другой — на предотвращение перегрузок на промежуточных сетевых узлах, ведущих к потере пакетов. На сегодняшний день существует несколько десятков алгоритмов управления окном перегрузки, наиболее популярные из которых исследуются в данной статье.

Современные Web-страницы состоят из множества компонент, таких как текстовые файлы, изображения, видео, аудио, апплеты, скрипты и др. Для ускорения загрузки Web-страниц браузеры могут загружать различные компоненты одной и той же Web-страницы с помощью

нескольких TCP соединений, установленных параллельно между одними и теми же клиентом и сервером.

В данной работе мы исследуем, как различные алгоритмы управления окном перегрузки, а также число используемых соединений TCP влияют на время загрузки Web-страниц в современных сотовых сетях LTE. Начиная с версии 3.26, выпущенной в 2016 году, в системе имитационного моделирования NS-3 появилась поддержка различных алгоритмов управления окном перегрузки. Это позволяет использовать для моделирования передачи Web-трафика по сети LTE реализованные в среде NS-3 и провалидированные модели стека протоколов LTE и TCP/IP, что, в свою очередь, обеспечивает повторяемость полученных в данной работе экспериментальных результатов.

Дальнейшее изложение статьи построено следующим образом. В разделе 2 приведено базовое описание протокола TCP и исследуемых в работе алгоритмов управления окном перегрузки. Принципы работы Web-приложений и структура Web-трафика приведены в разделе 3. В разделе 4 приводится подробное описание экспериментов и анализ полученных численных результатов. В Заключении содержатся основные выводы работы.

2. ОБЗОР СУЩЕСТВУЮЩИХ АЛГОРИТМОВ УПРАВЛЕНИЯ ОКНОМ ПЕРЕГРУЗКИ ПРОТОКОЛА TCP

TCP [3] является одним из основных протоколов сети Интернет. Он обеспечивает надежную, упорядоченную доставку данных между двумя приложениями, работающими на узлах сети Интернет. Почти все основные интернет-приложения, такие как браузеры, почтовые клиенты, программы удаленного администрирования и многие другие, используют TCP.

Передача данных по протоколу TCP происходит только по установленным логическим соединениям. Соединения идентифицируются парой адресов сетевых узлов, участвующих в обмене информацией, и парой 16-битных чисел — портов, которые идентифицируют взаимодействующие приложения на обоих узлах. Для передачи данных и служебной информации узлы обмениваются пакетами, которые в протоколе TCP называются сегментами. Сегмент содержит заголовок со служебной информацией и, возможно, пользовательские данные. Сегмент без пользовательских данных будем называть пустым.

Чтобы установить соединение, клиент отправляет пустой сегмент, в заголовке которого взведен флаг SYN (от англ. synchronize), в ответ на который сервер отправляет пустой сегмент с флагами SYN и ACK (от англ. acknowledgement). Затем клиент завершает процедуру открытия соединения, отправляя сегмент с флагом ACK.

В протоколе TCP получение всех данных квитируется. Для этого каждый байт с данными в буфере отправителя нумеруется 32-битным числом. Перед отправкой байты данных делятся на сегменты размером не больше MSS (от англ. Maximum Segment Size). В заголовке каждого отправляемого сегмента указывается порядковый номер первого байта пользовательских данных, передаваемых в этом сегменте. Получив сегмент с данными, узел отправляет сегмент с флагом ACK, в котором указывает минимальный порядковый номер еще не полученного байта данных. Таким образом, отправляя сегмент ACK, узел подтверждает, что получил все байты с порядковыми номерами меньше указанного в заголовке этого сегмента. Если узел получает два сегмента ACK с одинаковым значением подтверждаемого порядкового номера, то говорят, что узел получил дубликат подтверждения. Получение дубликатов подтверждений означает, что часть сегментов с данными была потеряна или произошло их переупорядочивание при передаче по сети.

Производительность TCP в значительной степени зависит от объема данных, которые узел-отправитель может отправлять в сеть, не дожидаясь подтверждения — окна перегрузки W (W обычно измеряется в байтах). Таким образом, за время кругового обращения пакета по сети

(англ.: Round-trip time) RTT, за которое пакеты доставляются от одного узла до другого и обратно, можно доставить не более чем W байт данных. Если выбранное значение окна перегрузки будет слишком маленьким, то скорость передачи данных по сети будет также мала. Если же W будет превышать пропускную способность сети, то на некоторых промежуточных узлах будут возникать перегрузки, очередь будет расти, и при достижении некоторого максимального размера очереди пакеты будут отбрасываться. Поэтому во всех существующих версиях протокола TCP размер окна перегрузки выбирается динамически согласно некоторым алгоритмам, которые увеличивают W, если данные успешно доставляются, и уменьшают, если возникает потеря данных. В литературе было предложено огромное количество различных модификаций алгоритма управления окном перегрузки. Ниже мы рассмотрим наиболее популярные алгоритмы, которые описаны в спецификациях международного сообщества по стандартизации сетевых протоколов сети Интернет (англ.: Internet Engineering Task Force, IETF), а также используются в современных операционных системах Linux, Windows, Mac OS.

2.1. Reno

Одним из базовых алгоритмов управления окном перегрузки, описанных в спецификациях IETF, является алгоритм Reno [4]. Работа алгоритма Reno делится на 3 фазы: медленного старта (англ. Slow Start), предотвращения перегрузки (англ. Congestion Avoidance) и быстрого восстановления после потери данных (англ. Fast Recovery). Переключение между режимами Slow Start и Congestion Avoidance осуществляется по достижении окном перегрузки определенного значения – порога медленного старта $ssthresh$. Если $W < ssthresh$, алгоритм работает в режиме Slow Start, иначе – в режиме Congestion Avoidance. Фаза Fast Recovery описывает правила работы после потери данных.

Во время фазы Slow Start окно перегрузки увеличивается на MSS после получения каждого подтверждения. Это, в свою очередь, означает, что окно W растет по экспоненциальному закону: при отсутствии потерь данных за время RTT окно будет увеличиваться в два раза. В фазе Congestion Avoidance рост окна становится линейным: окно W увеличивается на MSS^2/W после каждого подтверждения, т.е. за время RTT окно увеличивается на MSS.

Потеря данных фиксируется по двум событиям: (а) истечение таймера повторной передачи (RTO) и (б) получение подряд трех дубликатов подтверждений.

После истечения таймера RTO изменяется порог медленного старта ($ssthresh = \max(FlightSize/2, 2 \cdot MSS)$, где $FlightSize$ – объем отправленных, но не подтвержденных данных), осуществляется повторная передача потерянного сегмента данных, окно перегрузки уменьшается до минимального значения $W = MSS$. Алгоритм переходит в фазу Slow Start.

После получения трех дубликатов подтверждений алгоритм переходит в фазу FastRecovery. Узел-отправитель изменяет порог медленного старта $ssthresh = \max(FlightSize/2, 2MSS)$, осуществляет повторную передачу первого неподтвержденного сегмента и устанавливает новый размер окна перегрузки $W = ssthresh + 3 \cdot MSS$. При получении последующих дубликатов подтверждений окно перегрузки каждый раз увеличивается на MSS, поскольку эти события означают, что по крайней мере один сегмент был доставлен получателю. Когда потерянный сегмент подтверждается, осуществляется выход из фазы FastRecovery, а окно перегрузки устанавливается равным $W = ssthresh$.

К недостаткам алгоритма относится его низкая производительность при множественных потерях внутри одного окна перегрузки, поскольку при каждой потере уменьшается порог медленного старта и окно перегрузки.

2.2. NewReno

Для решения проблемы множественных потерь пакетов внутри одного окна перегрузки был предложен алгоритм NewReno [5], который отличается от Reno фазой Fast Recovery.

В алгоритме NewReno длительность пребывания в фазе Fast Recovery определяется переменной *Recover*, которая инициализируется при переходе в эту фазу значением порядкового номера последнего отправленного байта данных. NewReno остается в фазе Fast Recovery до подтверждения байта данных с номером *Recover*.

При получении частичного подтверждения (с номером менее *Recover*) происходит передача первого неподтвержденного сегмента данных, уменьшение W на величину, равную размеру данных, получение которых подтверждено данным частичным подтверждением, а также увеличение окна на MSS, если размер подтвержденных данных превосходит MSS (поскольку это означает, что, по крайней мере, один сегмент размера MSS был доставлен до получателя). При получении дублирующих подтверждений окно перегрузки увеличивается на MSS (также по причине того, что один сегмент был доставлен до получателя). При получении полного подтверждения (с номером не менее *Recover*) происходит уменьшение окна по правилам, описанным ниже, и выход из фазы Fast Recovery.

Существует два распространенных способа уменьшения окна перегрузки при выходе из фазы Fast Recovery. Согласно первому способу W устанавливается равным значению порога медленного старта. Однако в данном случае возможна передача большого числа пакетов за короткий интервал времени (поскольку текущее значение $FlightSize$ может быть значительно меньше $ssthresh$), что может привести к переполнению очередей на промежуточных узлах и новым потерям. Согласно второму способу $W = \min(ssthresh, \max(FlightSize, MSS) + MSS)$. В зависимости от объема неподтвержденных данных $FlightSize$ алгоритм входит в фазу Slow Start (если неподтвержденных данных достаточно мало) или в фазу Congestion Avoidance.

К недостаткам алгоритма следует отнести тот факт, что на восстановление одного потерянного сегмента необходимо время, равное RTT. При большом числе потерянных сегментов возможно наступление события истечения таймера RTO, что приведет к сбросу окна перегрузки $W = MSS$ и порога $ssthresh = \max(FlightSize/2, 2 \cdot MSS)$ и переходу в фазу Slow Start.

2.3. HighSpeed (RFC 3649)

Нетрудно показать, что для достижения пропускной способности S, окно перегрузки должно иметь значение $W = S \cdot RTT$. Для достижения такого окна в фазе Congestion Avoidance алгоритму Reno/NewReno потребуется время, равное $S \cdot RTT^2/MSS$. Для сетей с высокой пропускной способностью и/или большим значением RTT данное время может достигать нескольких часов.

Для решения этой проблемы был предложен алгоритм HighSpeed [6], который изменяет фазу Congestion Avoidance алгоритма Reno. Для удобства введем обозначение $W = w \cdot MSS$, где w — это величина окна перегрузки, измеряемая в MSS. Получение каждого подтверждения ведет к увеличению окна перегрузки w на $a(w)/w$ (т.е. увеличению окна на $a(w)$ за время RTT). При получении трех дубликатов подтверждений окно уменьшается на величину $b(w) \cdot w$. Для вычисления коэффициентов используются следующие формулы:

$$b(w) = (B - 0.5)\frac{\log(w) - \log(w_0)}{\log(w_1) - \log(w_0)} + 0.5, \;\; a(w) = 2w^2 p(w)\frac{b(w)}{2 - b(w)},$$

где $w_0 = 38$, $w_1 = 83333$, $B = 0.1$. Переменная $p(w)$ обозначает вероятность потери сегмента данных, для оценки которой используются следующие выражения:

$$p(w) = p_0(w/w_0)^S,\ S = \frac{\log p_1 - \log p_0}{\log w_1 - \log w_0},$$

где $p_1 = 10^{-7}$, $p_0 = 10^{-3}$.

Используя приведенные выше формулы, можно показать, что при значении окна $w = 1000$ MSS коэффициенты $a(w) = 7$ и $b(w) = 0.33$, а при $w = 10000$ MSS коэффициенты равны $a(w) = 30$ и $b(w) = 0.21$. Таким образом, чем больше значение окна перегрузки (а, значит, и оцениваемая пропускная способность канала), тем более «агрессивно» алгоритм HighSpeed начинает увеличивать окно при получении подтверждений и соответственно уменьшать окно на меньшее значение при потере пакетов.

2.4. CUBIC

Алгоритм CUBIC [7], используемый в настоящее время по умолчанию в операционных системах Linux и Mac OS, также предназначен для работы в высокоскоростных сетях и основан на следующей идее. Для эффективной работы алгоритма следует медленно изменять окно перегрузки, когда оно близко к оптимальному значению (пропускной способности сети), и быстро изменять после потери сегментов и при поиске нового оптимального значения.

CUBIC изменяет фазу Congestion Avoidance алгоритма Reno для случая, когда $W > 38 \cdot MSS$. В алгоритме используются две переменные: W_{max} — размер окна перегрузки перед получением трех дубликатов, а также W_{min} — значение окна сразу после выхода из фазы Fast Recovery. Для выбора данных параметров используется следующий механизм быстрого схождения (англ.: Fast Convergence). Если на момент получения трех дубликатов $W > W_{max}$, то $W_{max} = W$, $W_{min} = (1-\beta)W$. Если же $W < W_{max}$, алгоритм полагает, что в сети появились новые соединения, которым необходимо выделить пропускную способность, поэтому $W_{max} = (1 - \beta/2)W$, $W_{min} = (1 - \beta)W$. По умолчанию $\beta = 0.2$.

В начале фазы Congestion Avoidance окно перегрузки устанавливается равным W_{min}. После этого окно изменяется во времени по закону $W(t) = C(t - K)^3 + W_{min}$, где $K = \sqrt[3]{\frac{W_{max}\beta}{C}}$, а $C = 0.4$.

Преимуществом алгоритма CUBIC является то, что скорость изменения окна перегрузки не зависит от текущего значения RTT. В отличие от приведенных выше алгоритмов Reno/NewReno и HighSpeed, при работе TCP-соединений с различными значениями RTT пропускная способность сети распределяется равномерно между ними.

2.5. Vegas

Кроме обнаружения факта перегрузки сети по потерям пакетов, наступление перегрузки можно предсказывать, используя текущие измерения времени RTT передачи пакета из конца в конец. Эта идея была положена в основу алгоритма Vegas [8].

Алгоритм Vegas хранит переменную RTT_{min}, которая равна минимальному RTT, измеренному с момента открытия TCP-соединения. Для предсказывания перегрузки алгоритм сравнивает ожидаемую пропускную способность сети W/RTT_{min} с текущей W/RTT. Заметим, что разница $\delta = W\frac{RTT - RTT_{min}}{RTT}$ между объемом данных, которые сеть способна передавать за время RTT_{min}, и объемом данных, которые сеть передает на данный момент, представляет собой данные, хранящиеся в буферах промежуточных узлах сети.

Vegas модифицирует фазу Congestion Avoidance алгоритма Reno следующим образом. Если δ не превосходит величину α, то размер окна перегрузки увеличивается на MSS, поскольку

считается, что пропускная способность сети используется не полностью. Если размер окна превосходит другую величину β, то окно уменьшается на MSS. В иных случаях размер окна не изменяется. По умолчанию $\alpha = MSS$, $\beta = 3 \cdot MSS$.

Недостатком алгоритма Vegas является то, что он существенно проигрывает в скорости передачи данных при одновременной работе в сети с описанными выше алгоритмами, основанными на обнаружении перегрузки (NewReno, CUBIC). Кроме того, Vegas чувствителен к оценке RTT_{min}: при работе в сетях с большой флуктуацией RTT ожидаемая пропускная способность сети может оцениваться неверно.

2.6. Westwood/Westwood+

В сетях, где потеря пакетов является относительно частым явлением (например, беспроводные сети), алгоритм Reno обладает низкой производительностью, поскольку значение окна перегрузки оказывается заниженным. Алгоритм Westwood [9] основан на идее, что окно перегрузки W необходимо поддерживать близким к оптимальному значению, равному $B \cdot RTT_{min}$, где B — пропускная способность сети.

Алгоритм Westwood оценивает пропускную способность сети на основе мгновенных значений пропускной способности, полученных по формуле: $b_k = \frac{d_k}{t_k - t_{k-1}}$, где t_k — время получения подтверждения с номером k, d_k — число подтвержденных байт. Мгновенные значения пропускной способности сглаживаются с помощью фильтра нижних частот по следующей формуле:

$$\hat{b}_k = \frac{\sigma - 1}{\sigma + 1}\hat{b}_{k-1} + \frac{b_k + b_{k-1}}{\sigma + 1};$$

где $\sigma = \frac{2\tau}{t_k - t_{k-1}}$, а τ — параметр фильтра. В фазе Congestion Avoidance алгоритм Westwood устанавливает значение окна равным $\hat{b}_k \cdot RTT_{min}$.

Одним из недостатков алгоритма Westwood является то, что он получает завышенную оценку пропускной способности сети в случае, когда узел-отправитель данных получает одновременно несколько подтверждений (например, после их задержки на промежуточном узле). Для решения этой проблемы была предложена модификация алгоритма, получившая название Westwood+ [10]. В отличие от Westwood алгоритм Westwood+ оценивает мгновенные значения пропускной способности сети не по получению каждого подтверждения, а за время последнего RTT: $b_k = \frac{D_k}{RTT_k}$, где D_k — объем данных, переданных за время RTT_k. Тем самым обеспечивается более стабильная оценка пропускной способности.

3. МОДЕЛЬ WEB-ТРАФИКА

Передача Web-страниц от удаленного сервера к мобильным устройствам является одним из основных типов трафика в современных сетях LTE. Типичная Web-страница состоит из следующих компонент:

- основного объекта (англ.: main object), который представляет собой текстовый файл в формате HTML;
- вложенных объектов (англ.: embedded objects) — файлов, содержащих текст, изображения, видео, аудио, апплеты, скрипты и другую информацию.

Загрузка Web-страниц на мобильное устройство осуществляется с использованием протокола уровня приложений HTTP (HyperText Transfer Protocol), который, в свою очередь, в качестве нижележащего протокола транспортного уровня использует протокол TCP. Согласно протоколу HTTP мобильное устройство и сервер обмениваются информацией в формате «запрос-ответ».

Упрощенно загрузку Web-страницы можно представить в виде следующий действий.

1. Пользователь запросил Web-страницу, перейдя по гиперссылке или введя ее адрес.
2. Мобильное устройство устанавливает с сервером TCP-соединение, с помощью которого будет осуществляться двухсторонняя передача данных.
3. Используя установленное TCP-соединение, мобильное устройство отправляет запрос на загрузку основного объекта.
4. Получив запрос, сервер отправляет основной объект.
5. Мобильное устройство, получив основной объект, определяет количество вложенных объектов и адреса, по которым они расположены (например, часть вложенных объектов могут быть расположены на другом сервере).
6. Мобильное устройство устанавливает TCP-соединения с другими серверами, на которых находятся вложенные объекты, а также может установить несколько дополнительных TCP-соединений, чтобы осуществлять параллельную загрузку нескольких вложенных объектов с одного и того же сервера.
7. Мобильное устройство отправляет запросы на загрузку вложенных объектов. Отметим, что согласно наиболее распространенной на сегодняшний день версии протокола HTTP (версии HTTP/1.1) запросы внутри одного TCP-соединения отправляются последовательно (т.е. следующий запрос может быть отправлен только после получения ответа на предыдущий запрос).
8. По мере поступления вложенных объектов, Web-страница и расположенные на ней объекты отображаются пользователю.

С точки зрения пользователя основным показателем, определяющим эффективность работы сети, является время загрузки Web-страницы, т.е. длительность интервала времени между запросом пользователя и отображением готовой страницы. Заметим, что время загрузки Web-страницы зависит от многих факторов: (а) скоростей передачи данных от/к базовой станции (они, в свою очередь, зависят от числа обслуживаемых пользователей, условий в беспроводном канале и других факторов), (б) используемой версии протокола TCP и, в частности, алгоритма управления окном перегрузки, (в) числа устанавливаемых TCP-соединений. В разделе 4 данной работы мы детально исследуем влияние вышеперечисленных факторов на время загрузки Web-страницы.

4. ЧИСЛЕННЫЕ РЕЗУЛЬТАТЫ

4.1. Описание экспериментов

Для сравнительного анализа различных алгоритмов управления окном перегрузки протокола TCP рассмотрим следующий сценарий. К базовой станции сети LTE подключены N мобильных устройств. Устройства расположены равномерно в круге радиусом $R = 500$ м, в центре которого находится базовая станция. Базовая станция и сервер, на котором расположены Web-страницы, соединены проводным каналом с пропускной способностью 10 Гбит/с и временем передачи пакета из конца в конец $RTT_{internet} = \{10, 100\}$ мс.

Для управления радиоресурсами базовая станция использует планировщик Proportional Fair (PF) [12]. Согласно PF мобильные устройства, у которых есть данные на передачу, в среднем получают одинаковую долю частотно-временных ресурсов канала. Для каждого мобильного устройства базовая станция поддерживает отдельную очередь пакетов. Пакеты отбрасываются, если время их нахождения в очереди превышает порог $d = 300$ мс. Другие параметры стека протоколов LTE и TCP приведены в таблице 1.

Каждое мобильное устройство генерирует Web-трафик, описанный в разделе 3, со следующими параметрами. Время между началами загрузки двух последовательных страниц имеет усеченное экспоненциальное распределение со средним значением 30 с, минимальным и

Таблица 1. Основные параметры стека протоколов LTE и TCP

Параметр	Значение
Параметры LTE	
Несущая частота восходящего канала	1.93 ГГц
Несущая частота нисходящего канала	2.12 ГГц
Ширина восходящего канала	20 МГц
Ширина нисходящего канала.	20 МГц
Модель распространения радиосигнала	Okumura-Hata [13]
Параметры TCP	
Размер сегмента данных MSS	1450 байт
Размер TCP-буфера	6 Мбайт
Начальное значения окна перегрузки	10 MSS
Минимально значение таймера RTO	1 с
Опция TCP timestamp	включена
Опция TCP SACK	включена

максимальным значением соответственно 10 и 60 с. Размер запроса на уровне приложений фиксирован и равен 800 байт. Размер основного объекта задается усеченным логнормальным распределением со средним 20 Кбайт и дисперсией $22 \cdot 10^6$ байт^2, минимальным и максимальным значением соответственно 100 байт и 200 Кбайт. Размеры вложенных объектов задаются усеченным логнормальным распределением со средним 160 Кбайт и дисперсией $5 \cdot 10^9$ байт^2, минимальным и максимальным значением соответственно 50 байт и 2 Мбайт. Число вложенных объектов задается усеченным распределением Парето с коэффициентом масштаба $x_m = 2$ и показателем степенной зависимости $k = 1{,}1$, минимальным и максимальным значением соответственно 2 и 50.

В экспериментах измеряются следующие величины: (а) T — среднее время загрузки веб-страницы, (б) ρ — доля частотно-временных ресурсов, используемых базовой станцией в нисходящем канале. Длительность каждого эксперимента 600 с. Для каждой точки выполняется 10 независимых прогонов имитационной модели.

Исследование качества обслуживания Web-трафика в сетях LTE в данном сценарии проводилось в среде имитационного моделирования NS-3 [11].

4.2. Анализ численных результатов

В первой серии экспериментов мы исследуем, как время загрузки Web-страниц зависит от используемого алгоритма управления окном перегрузки. На рис. 1 представлены графики зависимости среднего времени T загрузки Web-страницы от числа пользователей N, подключенных к базовой станции, при различных значениях $RTT_{internet}$. В данной серии экспериментов для загрузки каждой Web-страницы устанавливается ровно одно TCP-соединение.

Результаты показывают, что загрузка базовой станции ρ линейно увеличивается с числом пользователей. При этом точка $N = 5$ соответствует нагрузке $\rho = 15\%$, а $N = 30$ соответствует $\rho = 95\%$.

Из графиков, приведенных на рис. 1, можно видеть, что использование алгоритма Vegas приводит к значительному увеличению времени загрузки веб-страниц по сравнению с другими алгоритмами. Связано это с тем, что в беспроводной сети RTT существенно меняется с течением времени, что, в свою очередь, приводит к ошибке в оценке RTT_{min} и соответственно заниженной оценке ожидаемой пропускной способности сети. Алгоритм Westwood+ при определении размера окна перегрузки также использует оценку RTT_{min}, однако при этом использует другой метод оценки пропускной способности сети, который позволяет более точно

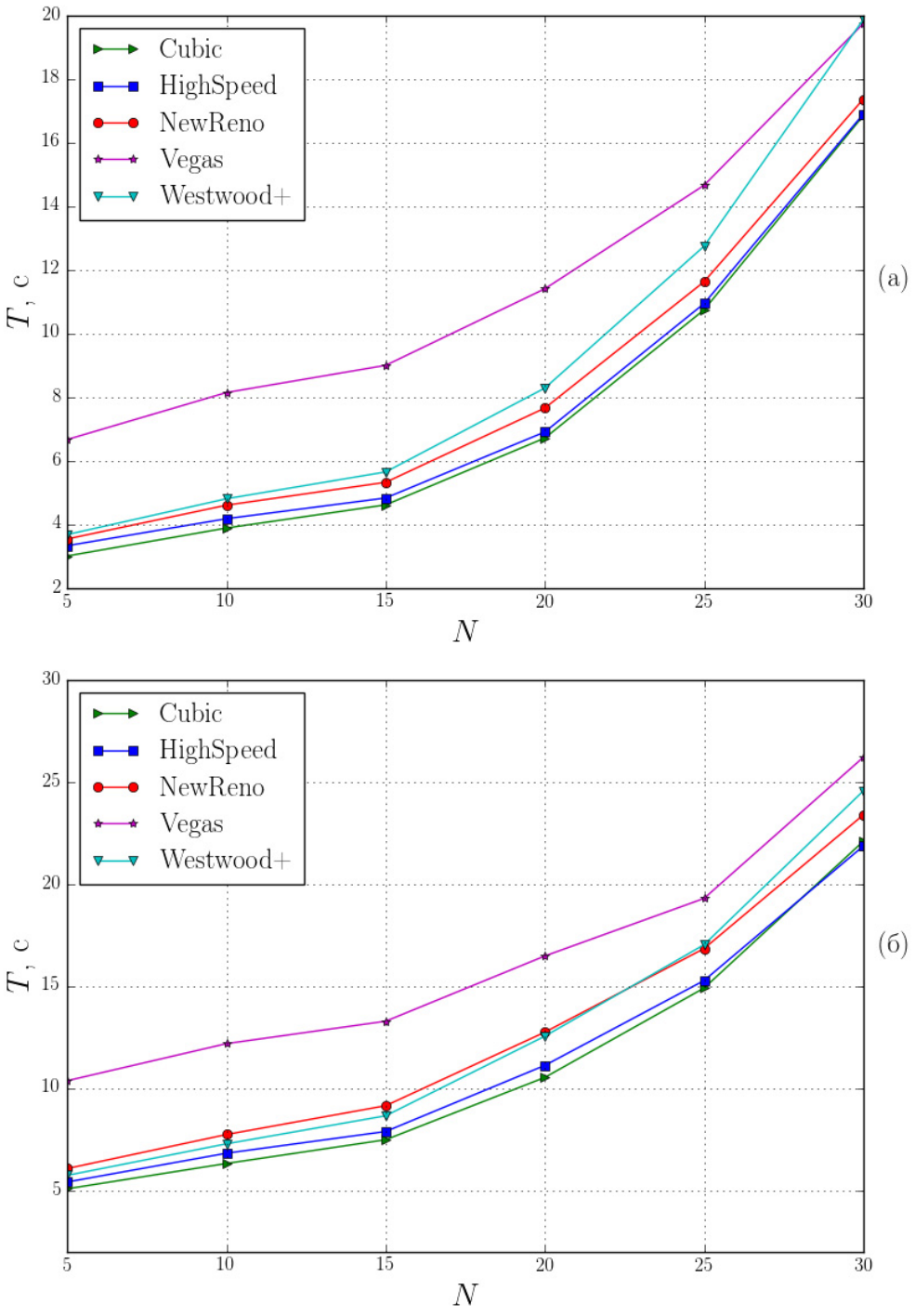

Рис. 1. Сравнение различных алгоритмов управлениея окном перегрузки: (a) $RTT_{internet} = 10$ мс, (b) $RTT_{internet} = 100$ мс

определить доступную пропускную способность сети и за счет этого уменьшить время загрузки Web-страниц.

Среди алгоритмов, которые основаны на определении перегрузки по потерям пакетов, алгоритмы HighSpeed и CUBIC показывают наименьшее время загрузки веб-страниц. Различие между алгоритмами NewReno и HighSpeed/CUBIC становится заметным только при большом времени $RTT_{internet} = 100$ мс. Связано это с тем, что в фазе Congestion Avoidance алгоритм

NewReno увеличивает окно перегрузки только на один MSS за время RTT, в то время как HighSpeed и CUBIC увеличивают окно более «агрессивно», что позволяет быстрее увеличивать скорость передачи данных после потери пакетов.

Если сравнивать между собой результаты при различных значениях $RTT_{internet}$, то можно видеть, что большие значения $RTT_{internet}$ приводят к увеличению времени загрузки Web-страницы для всех алгоритмов управления окном перегрузки. Это вызвано тем, что все алгоритмы в фазе Slow Start увеличивают окно перегрузки только после получения подтверждений. Время между отправкой сегмента данных и получения на него подтверждения, в свою очередь, зависит от $RTT_{internet}$. Таким образом, при увеличении $RTT_{internet}$ время, необходимое для того, чтобы увеличить окно перегрузки до значения, соответствующего доступной пропускной способности сети, также увеличивается, что приводит к увеличению времени загрузки Web-страницы.

Во второй серии экспериментов мы исследуем, как среднее время загрузки Web-страниц зависит от числа устанавливаемых TCP-соединений для каждой загружаемой Web-страницы. В качестве алгоритма управления окном перегрузки используется алгоритм CUBIC.

На рис. 2 показано относительное уменьшение времени загрузки Web-страницы $\sigma = (T_1 - T_x)/T_1$, если для каждой Web-страницы устанавливать x TCP-соединений вместо одного TCP-соединения. Из приведенных графиков можно видеть, что при низкой загрузке сети $\rho < 50\%$ ($N < 15$) использование нескольких TCP-соединений позволяет значительно уменьшить время загрузки Web-страниц. В частности, при $RTT_{internet} = 100$ мс использование четырех соединений позволяет уменьшить время загрузки на 30%.

Данный эффект связан с тем, что в начале передачи Web-страницы все TCP-соединения работают в фазе Slow Start. При открытии TCP-соединения окно перегрузки на отправителе устанавливается в начальное значение $W_0 = 10 \cdot MSS$. При низкой нагрузке сети окна размером W_0 оказывается недостаточно, чтобы использовать всю доступную пропускную способность беспроводного канала. За счет установки x TCP-соединений можно в x раз увеличить число пакетов, передаваемых в сеть в начале загрузки веб-страницы. Кроме того, каждое из TCP-соединений увеличивает окно перегрузки по экспоненциальному закону. Таким образом, увеличение числа TCP-соединений позволяет серверу быстрее достигать доступной пропускной способности сети.

Заметим, что в рассматриваемом сценарии использование $x > 4$ TCP-соединений оказывается нецелесообразным, так как дальнейшее увеличение TCP-соединений не приводит к увеличению скорости передачи данных, которая ограничена пропускной способностью сети. Также можно видеть, что выигрыш от использования нескольких TCP-соединений уменьшается при увеличении нагрузки на сеть (увеличении числа пользователей), так как на каждого пользователя приходится меньшая доля пропускной способности канала. Поэтому одного TCP-соединения оказывается достаточно, чтобы полностью использовать доступную пользователю пропускную способность беспроводного канала. Более того, из приведенных графиков можно видеть, что при высокой нагрузке на сеть ($N > 25$) и малом значении $RTT_{internet} = 10$ мс использование большого числа TCP-соединений ($x > 8$) может приводить к потере большого числа пакетов в фазе Slow Start и соответственно увеличению времени загрузки веб-страниц.

Таким образом, проведенные исследования показывают, что число TCP-соединений, которые необходимо устанавливать для загрузки Web-страниц, в значительной степени зависит от доступной пропускной способности беспроводного канала и нагрузки на базовую станцию, а также от начального значения окна перегрузки на сервере W_0 и времени RTT кругового обращения пакета по сети. Разработка алгоритма адаптивного выбора числа устанавливаемых TCP-соединений является одним из направлений дальнейших исследований.

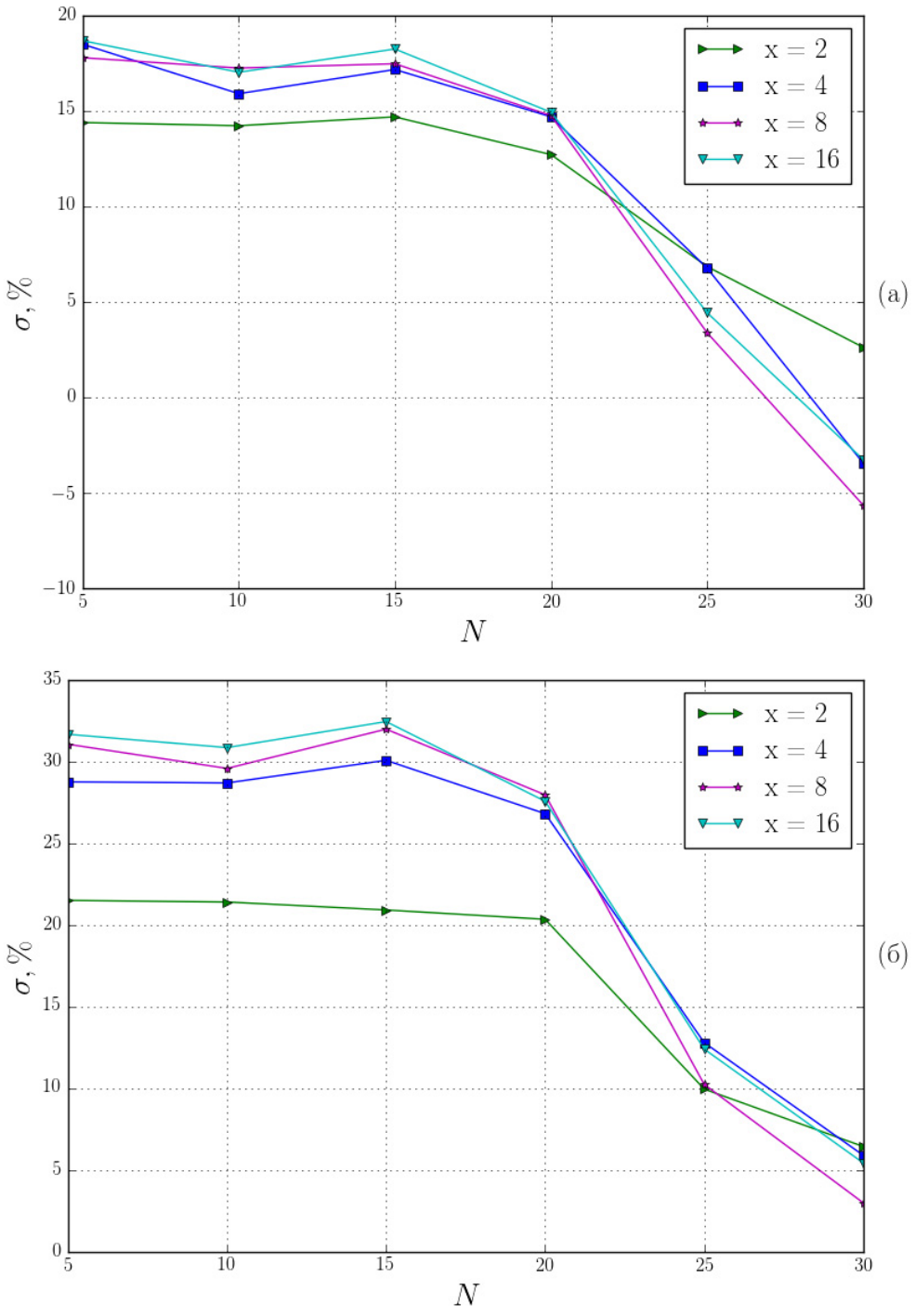

Рис. 2. Исследование влияния числа устанавливаемых TCP-соединений: (a) $RTT_{internet} = 10$ мс, (b) $RTT_{internet} = 100$ мс

5. ЗАКЛЮЧЕНИЕ

В данной работе исследовано, как различные алгоритмы управления окном перегрузки протокола TCP и число параллельных соединений TCP, установленных между одной и той же парой клиент-сервер, влияют на среднее время загрузки Web-страниц в сетях LTE. Выявлено, что большинство современных алгоритмов обладают примерно одинаковой производитель-

ностью. Однако, алгоритмы, базирующиеся на оценке RTT, могут быть неэффективными в беспроводных сетях из-за большой флуктуации размера очереди на базовой станции. Кроме того, установлено, что использование 2-4 параллельных TCP-соединений позволяет на треть сократить время загрузки Web-страниц. В то же время, дальнейшее увеличение числа TCP-соединений нецелесообразно.

СПИСОК ЛИТЕРАТУРЫ

1. Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H Katz. Improving TCP/IP performance over wireless networks // Proceedings of the 1st annual international conference on Mobile computing and networking. Pp. 2–11, ACM, 1995
2. Rehana Kausar. QoS Aware Packet Scheduling in the Downlink of LTE-Advanced Networks. PhD thesis. Queen Mary University of London, 2013. URL: http://www.eecs.qmul.ac.uk/ yue/Thesis/Rehana Kausar.pdf
3. *RFC 793 (Internet Standard): Transmission Control Protocol*, September 1981 `https://tools.ietf.org/html/rfc793`
4. M. Allman, V. Paxson, E. Blanton, *RFC 5681 (Draft Standard): TCP Congestion Control*, September 2009 `https://tools.ietf.org/html/rfc5681`
5. T. Henderson, S. Floyd, A. Gurtov, Y. Nishida *RFC 6582 (Proposed Standard): The NewReno Modification to TCP's Fast Recovery Algorithm*, April 2012 `https://tools.ietf.org/html/rfc6582`
6. S. Floyd *RFC 3649 (Experimental): HighSpeed TCP for Large Congestion Windows*, December 2003 `https://tools.ietf.org/html/rfc3649`
7. I. Rhee, L. Xu, S. Ha, *CUBIC for Fast Long-Distance Networks*, February 2007 `https://tools.ietf.org/html/draft-rhee-tcp-cubic-02`
8. L.S. Brakmo, L.L. Peterson, *TCP Vegas: End to end congestion avoidance on a global internet* // IEEE Journal on Selected Areas in Communications, 13 (8), pp 1465–1480.
9. S. Mascolo, C. Casetti, M. Gerla, M.Y. Sanadidi, R. Wang, *TCP Westwood: Bandwidth estimation for enhanced transport over wireless links* // Proc. 7th annual international conference on Mobile computing and networking, ACM, 2001, pp. 287–297.
10. R. Ferorelli, L.A. Grieco, S. Mascolo, G. Piscitelli, P. Camarda, *Live internet measurements using westwood+ tcp congestion control* // Proc. IEEE Global Telecommunications Conference 2002, Vol. 3, pp. 2583–2587.
11. Network Simulator 3, `http://www.nsnam.org/`
12. P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, A.Viterbi, *CDMA/HDR: A bandwidth efficient high speed data service for nomadic users* //IEEE Commun. Mag., pp. 70-77, July 2000
13. M.Hata, *Empirical formula for propagation loss in land mobile radio services* // IEEE Trans. on Vehicular Technology, vol. 29, pp. 317–325, 1980

On the Web-data Transmission in Cellular Networks

A.A. Belogaev, E.M. Khorov, A.N. Krasilov, A.I. Lyakhov, and N.S. Zhirnov

Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow, Russia

Most of Web applications use TCP protocol for data transmission. The key component of this protocol is a congestion control algorithm. This algorithm tries to fully utilize the available network capacity while avoiding congestion on the middle-boxes, when they have to discard packets. Initially, congestion control algorithms were developed for wired networks. However, the widespread use of Wi-Fi and LTE technologies

for Internet access encouraged research in the area of congestion control algorithms for wireless. In this paper we study how congestion control algorithm and the number of parallel TCP connections established between the same client and server affect Web data transmission in cellular networks.

KEYWORDS: Wireless networks, Web, TCP, LTE, congestion control