ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

DOI: 10.25586/RNU.V9187.21.02.P.101 УДК 004.051

К.Д. Новиков, М.В. Раскатова

ЭНЕРГОЭФФЕКТИВНОСТЬ МОБИЛЬНЫХ ВЕБ-ПРИЛОЖЕНИЙ

Рассмотрен и предложен способ сравнительной характеристики потребления энергии тремя асинхронными методами на основе HTTP в мобильных устройствах. Все эксперименты сосредоточены на моделях на базе HTTP, которые позволяют веб-серверу передавать данные в клиентский браузер через протокол HTTP: сокеты, длинный опрос и обычный опрос. Проведен анализ полученных результатов для получения более точного представления о влиянии методов и различных браузеров на энергопотребление асинхронной связи через протокол HTTP. Предложенный способ позволит мобильным веб-разработчикам снизить энергопотребление веб-приложений для мобильных устройств с помощью выбора лучшего асинхронного метода и/или мобильного браузера, что улучшит производительность при использовании протокола HTTP с точки зрения энергопотребления.
Ключевые слова: мобильные приложения, HTTP, Polling, WebSocket, Интернет.

K.D. Novikov, M.V. Raskatova

AN ENERGY EFFICIENCY OF MOBILE WEB APPLICATIONS

The purpose of this article is to study and create a comparative characterization of the power consumption of three asynchronous HTTP-based methods in mobile devices. All experiments focus on HTTP-based models, which allow the web server to send data to the client browser via the HTTP protocol: sockets, long polling and normal polling. The paper analyzes the results to gain a better understanding of the effect of methods and different browsers on the power consumption of asynchronous communication, over the HTTP protocol. With the results of the survey, mobile web developers should be able to reduce the power consumption of mobile web applications by simply choosing the best asynchronous method and/or mobile browser, which will improve the performance when using the HTTP protocol, in terms of power consumption.
Keywords: mobile applications, HTTP, Polling, WebSocket, Internet.

Вводные замечания

В настоящее время мобильные устройства являются самыми распространенными вычислительными устройствами и постепенно становятся основным способом доступа в сеть Интернет. Аккумулятор – это один из самых важных ресурсов в таких персональных вычислительных гаджетах, причем сетевые коммуникации потребляют довольно много энергии в любом приложении. Трафик на основе HTTP является наиболее энергоемким, а это означает, что разработчики мобильных приложений должны знать, сколько энергии требуется для различных альтернатив веб-коммуникаций.

Использование смартфонов является необходимостью для некоторых видов деятельности, например, получения доступа к социальным сетям, обмену сообщениями или просмотру интернет-страниц. Именно требование предоставления постоянного доступа в Интернет наряду с использованием подсветки смартфона больше всего расходует ре-

Новиков Константин Дмитриевич
магистрант Национального исследовательского университета «Московский энергетический институт». Сфера научных интересов: .NET Core, ASP.NET, Xamarin, сети передачи данных, C#, C++. Автор 1 опубликованной научной работы.
E-mail: ultimatedoker@gmail.com

Раскатова Марина Викторовна
кандидат технических наук, доцент кафедры вычислительных машин, систем и сетей Национального исследовательского университета «Московский энергетический институт», доцент кафедры информационных технологий и естественнонаучных дисциплин Российского нового университета. Сфера научных интересов: разработка программного обеспечения, информационные системы. Автор около 40 опубликованных научных работ.
E-mail: marina@raskatova.ru

сурс аккумуляторной батареи. Скрытое от пользователя, постоянное использование связи часто является виновником разряда батареи.

Вторая проблема заключается в том, что экраны и процессоры мобильных устройств постоянно совершенствуются, тогда как аккумуляторы мобильных устройств практически не развиваются. Таким образом, даже последние модели телефонов с большей емкостью батареи могут иметь меньшее время работы от одного заряда, чем их старые модели. С точки зрения ОС Android беспроводная связь (Wi-Fi, 3G, 4G, 5G, Bluetooth) является одним из основных потребителей энергии в смартфонах (более 40% общего объема батареи).

Ожидается, что в 2021 г. объем мобильного трафика данных в мире достигнет 49 эксабайт в месяц при среднегодовом темпе роста в 47%. Бо́льшая часть интернет-трафика мобильных устройств основана на протоколе HTTP. Таким образом, снижение энергопотребления за счет передачи данных может оказать значительное влияние на общее энергопотребление устройства и улучшить срок службы аккумулятора устройства.

Протокол HTTP

Изначально взаимодействие по протоколу HTTP происходило каждый раз, когда пользователь нажимал на ссылку: новый контент извлекался с сервера и отображался в браузере. Однако в настоящее время бо́льшая часть веб-содержимого извлекается в автоматическом режиме без вмешательства пользователя (мгновенные сообщения, уведомления). Это значит, что всякий раз, когда появляется новое содержимое, сервер может протолкнуть эту информацию в браузер, что позволяет пользователям своевременно получать обновления. Данная технология получила название server push.

Протокол HTTP/2 поддерживает технологию server push в рамках своей спецификации, в то время как HTTP/1.1 является синхронным протоколом запроса/отклика прикладного веб-приложения. Таким образом, чтобы добиться проталкивания контента с сервера с помощью HTTP/1.1, разработчикам предлагается использовать решения на уровне приложений, если им требуется технология асинхронной связи [3, 6]. Однако по

состоянию на 2020 г. HTTP/2 используется всего лишь на 35,4% всех сайтов. Большинство веб-приложений требуют, чтобы сервер посылал данные клиенту асинхронно по мере изменения состояния системы и без необходимости вмешательства пользователя.

Несмотря на то, что HTTP/1.1 является синхронным протоколом, у разработчиков веб-приложений существуют различные подходы для достижения асинхронного взаимодействия. Первый вариант, который до сих пор широко используется, заключается в эмуляции асинхронного взаимодействия по синхронному каналу связи с использованием метода Polling. Популярной разновидностью метода является так называемый длинный опрос (Long Polling), при котором HTTP-ответ задерживается на определенное время или до тех пор, пока данные не станут доступны.

Помимо этих методов недавно язык разметки HTML ввел в своей последней версии протокол WebSocket для реализации асинхронного взаимодействия, который не поддерживается HTTP/1.1. WebSocket позволяет осуществлять двунаправленное полнодуплексное постоянное соединение между клиентом и сервером с помощью сокетов. Основываясь на двунаправленном соединении, сервер может активно посылать (проталкивать) данные в браузер. Соединение WebSocket построено поверх TCP и имеет лишь небольшие накладные расходы по сравнению с HTTP.

Второй вещью, которая может повлиять на производительность, а также на энергопотребление, является браузер. Несмотря на то, что все браузеры предоставляют примерно одинаковые функциональные возможности, они могут иметь существенные отличия на уровне движка. При доступе к сайтам с мобильного устройства особенно важно обратить внимание на потребление и производительность используемого мобильного браузера. Таким образом, мобильные веб-разработчики также должны иметь в виду ограниченную мощность, доступную на мобильных устройствах, и возможную необходимость разработки соответствующих приложений.

Целью работы является исследование влияния трех различных асинхронных методов протокола HTTP (Short Polling, Long Polling и WebSocket) и двух различных мобильных браузеров (Chrome и Firefox) на энергоэффективность. Исследование подразумевает использование HTTP/1.1. Хотя HTTP/2 повышает производительность, эта версия используется на относительно малом количестве веб-сайтов [3].

Зная энергопотребление выбранного метода асинхронной связи, мобильные веб-разработчики смогут снизить энергопотребление front-end части веб-приложений на мобильных устройствах, выбрав наиболее подходящий веб-браузер и асинхронный механизм передачи данных.

Short Polling

В стандартной модели HTTP сервер не может инициировать соединение с клиентом или посылать клиенту незапрашиваемый HTTP-ответ. В связи с этим для более быстрого получения новых данных браузер периодически опрашивает сервер на предмет нового контента, посылая HTTP-запрос. При традиционном, или «коротком», способе опроса клиент регулярно посылает на сервер запросы, и каждый запрос пытается получить все доступные события или данные. Если же нет доступных данных, то сервер возвращает пустой ответ, и клиент ждет некоторое время, прежде чем отправить очередной запрос на

опрос (интервал опроса). Реализация опроса на стороне клиента полагается на возможности, включенные по умолчанию в браузеры, такие как JavaScript.

При наличии совместимого сервера и совместимого браузера клиенту просто необходимо инстанцировать объект XMLHttpRequest. XMLHttpRequest – это объект JavaScript, позволяющий получить доступ к DOM. Объект XMLHttpRequest может использоваться для обмена данными с веб-сервером без вмешательства пользователя (также известен как AJAX) [7]. Это означает, что можно обновлять части веб-страницы без перезагрузки всей страницы. Клиент инстанцирует XMLHttpRequest и отправляет его на сервер. После этого клиент считывает HTTP-ответ, который содержит обновленное содержимое, если в данных сервера появились новые данные из последнего HTTP-запроса. В противном случае ответ не содержит никакой полезной нагрузки в теле. Основной цикл взаимодействия приложения, использующего HTTP-polling, представлен на рисунке 1, *а*.

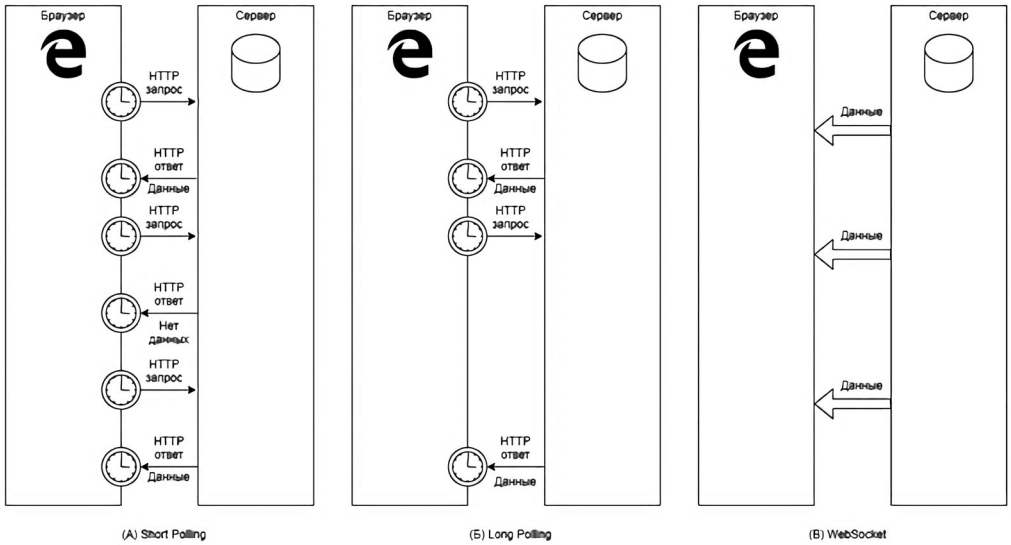

Рис. 1. Асинхронные методы сетевого взаимодействия: а – основной цикл взаимодействия приложения, использующего HTTP-polling; б – основной цикл взаимодействия приложения, использующего «длинный» опрос; в – основной цикл взаимодействия клиентского приложения с помощью WebSockets

Однако опрос, несмотря на свою простоту, имеет априорный недостаток: ресурсы, потребляемые клиентом (использование браузера и сети), сильно зависят от частоты обновления данных и интервала опроса. Если доступность данных низкая (например, порядка секунд), то высокая частота опроса может привести к недопустимой нагрузке на браузер, сеть или и на то, и на другое.

Long Polling

Для устранения недостатков метода Short Polling существует несколько серверных механизмов программирования. Одним из наиболее распространенных механизмов является «длинный» опрос, при котором сервер пытается не сразу отвечать на каждый запрос

HTTP, посылая ответ HTTP только тогда, когда есть данные. Это решение позволяет веб-серверу посылать данные клиентам, когда появляются новые данные, и браузеру не нужно знать о периодическом опросе, а также регулировать интервал опроса. «Длинный» опрос может доставлять данные для браузера, избегая при этом задержек, испытываемых клиентскими приложениями из-за интервала классического опроса или пустых ответов, тем самым сводя к минимуму задержки при доставке сообщений и использовании ресурсов обработки/сетевых ресурсов.

Учитывая наличие совместимого сервера и совместимого браузера, для отправки запроса клиенту нужно просто инстанцировать объект XMLHttpRequest и отправить его на сервер (методы открываются и отправляются). После этого клиент ждет http-ответа. Как только сервер получает новые данные, он завершает HTTP-взаимодействие, посылая соответствующий Http-ответ. Основной цикл взаимодействия приложения, использующего «длинный» опрос, представлен на рисунке 1, *б*.

Протокол WebSocket

WebSocket – это протокол, который позволяет использовать TCP-соединение, установленное между браузером и веб-сервером, для данных с небольшими накладными расходами по сравнению с HTTP. Используя это соединение, back-end сторона Web-сервера может асинхронно передавать данные клиенту, когда они доступны [2]. Перед обменом данными/сообщениями через это соединение протокол WebSocket требует начальной установки соединения. Обмен сообщениями осуществляется в виде фреймов, которые содержат либо текстовые, либо двоичные данные.

Получив WebSocket-совместимый сервер и совместимый браузер, клиент, используя JavaScript API, должен просто инстанцировать объект WebSocket и начать прослушивать события данных, передаваемые сервером (метод onMessage). Основной цикл взаимодействия клиентского приложения с помощью WebSockets можно увидеть на рисунке 1, *в*.

Исследование

Для измерения энергопотребления программных приложений, работающих на смартфонах, существует множество инструментов, основанных как на аппаратном, так и на программном обеспечении. Несмотря на то, что аппаратные измерения обеспечивают более высокую точность, мы не можем использовать их, так как они позволяют оценить энергопотребление всего устройства, а целью данного исследования является изучение энергопотребления на программном уровне. В нашем случае с учетом имеющихся устройств для тестирования идеально подойдет программное решение от компании Qualcomm: Trepn Profiler.

Испытания были проведены на Samsung Galaxy S10. Сайты развернуты на веб-сервере Glassfish в операционной системе Ubuntu. Аппаратная сторона представляет собой ПК, построенный на Intel Core i7 3,40 ГГц с 32 гигабайтами оперативной памяти. Веб-сервер имеет гигабитное Ethernet-соединение с сетью.

В качестве единицы измерения используется джоуль (J). Один джоуль эквивалентен одному ватту мощности, излучаемой или рассеиваемой в секунду. Периоды обновления данных, выбранные для тестов, составляют (в миллисекундах) 1000, 5000 и 30 000.

Для измерения и сравнения потребления трех асинхронных механизмов коммуникации в двух разных мобильных браузерах было развернуто три простых сайта, совместимых с мобильными браузерами. Каждый сайт позволяет протестировать один из трех рассмотренных механизмов коммуникации. Сервер генерирует события случайным образом с гауссовыми распределениями.

В работе было измерено энергопотребление мобильного браузера, когда он получает данные с помощью трех механизмов связи. Были проведены эксперименты длительностью 5 минут, так как согласно различным работам взаимодействие с планшетами или определенными веб-приложениями, например играми, новостными приложениями или доступом в социальные сети, длится более 1 минуты и, как правило, имеет такую длительность. Большинство уведомлений и обновлений происходит с интервалом более 30 секунд. Что касается размера данных, то в работе рассматриваются три различных размера сообщений – 140, 560 и 2240 байт. Выбор обусловлен размерами сообщений самых популярных приложений. Например, сообщение от Twitter имеет размер 140 байт, а сообщения, требующие фрагментации на IP-уровне, зачастую имеют размер 2240 байт. Каждый тест был повторен 30 раз.

На рисунках 2–7 показаны результаты этих экспериментов.

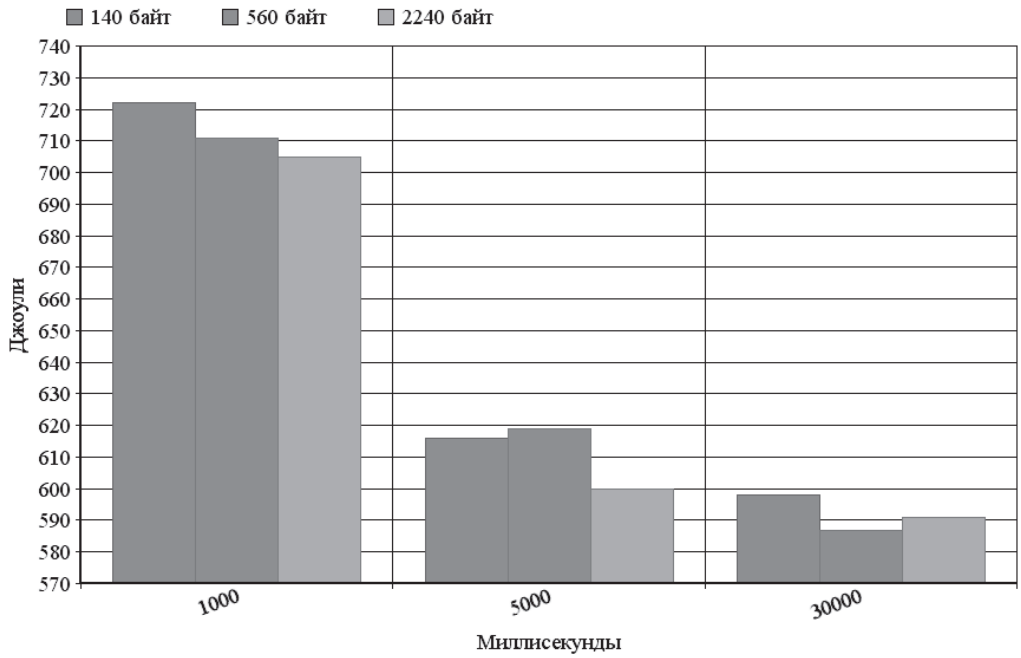

Рис. 2. Метод Short Polling через Chrome

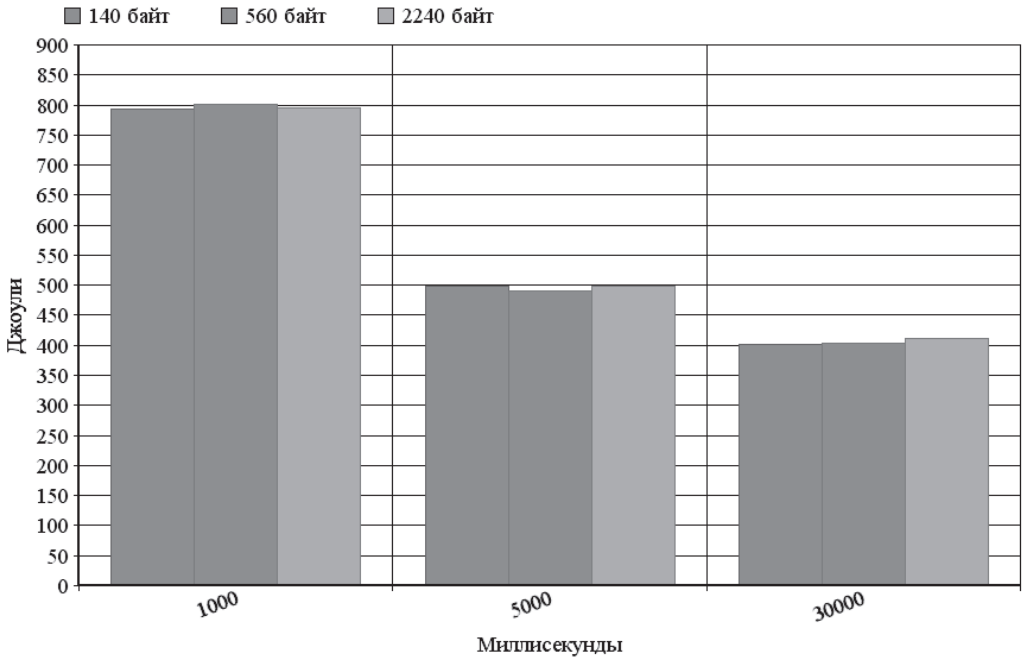

Рис. 3. Метод Short Polling через Firefox

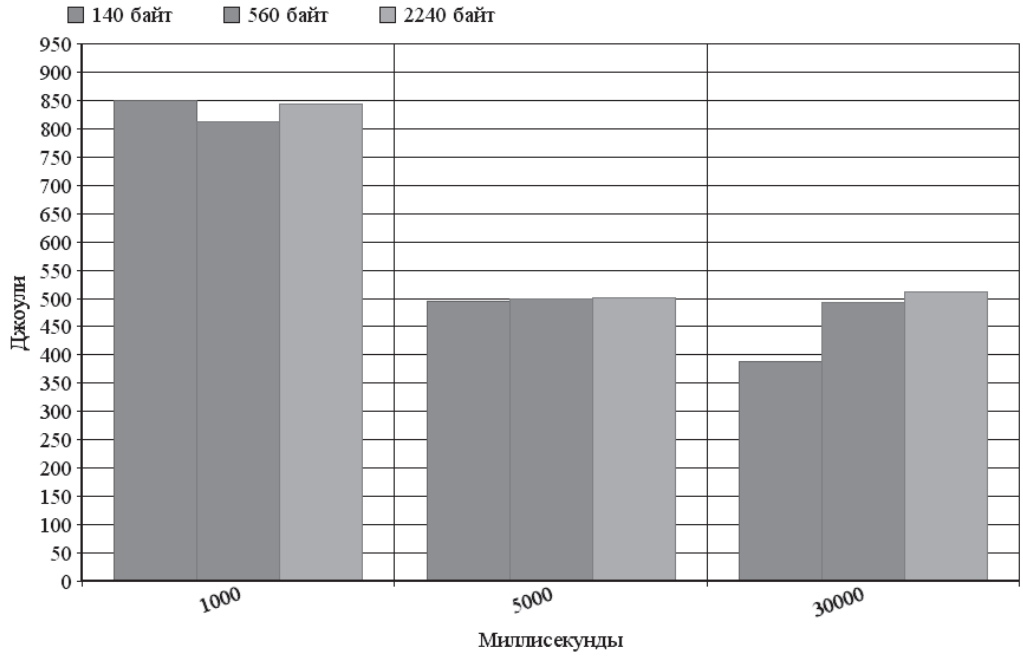

Рис. 4. Метод Long Polling через Chrome

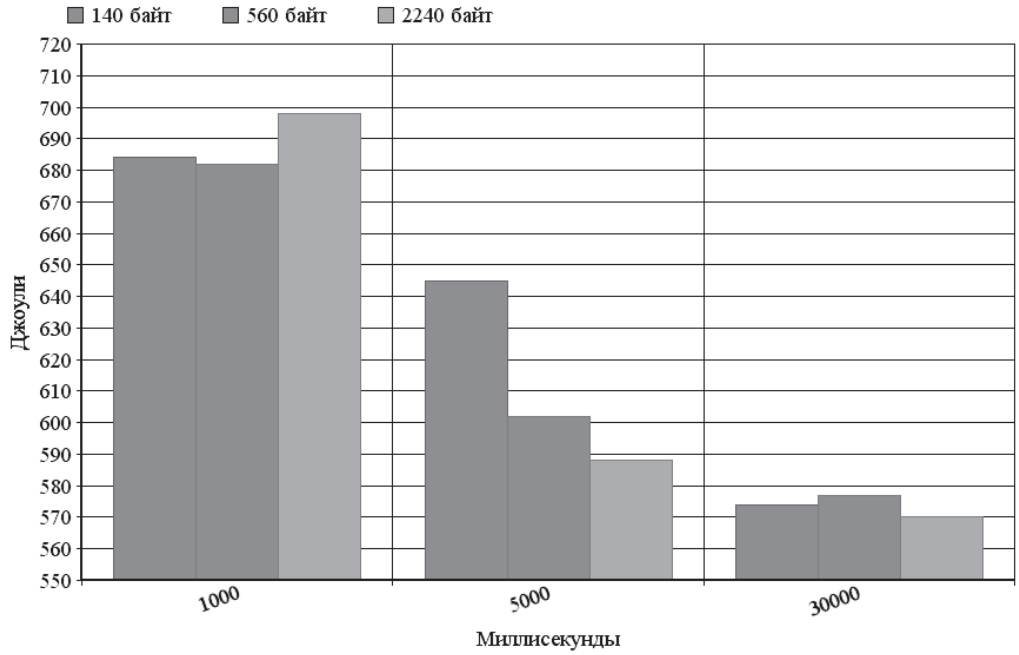

Рис. 5. Метод Long Polling через Firefox

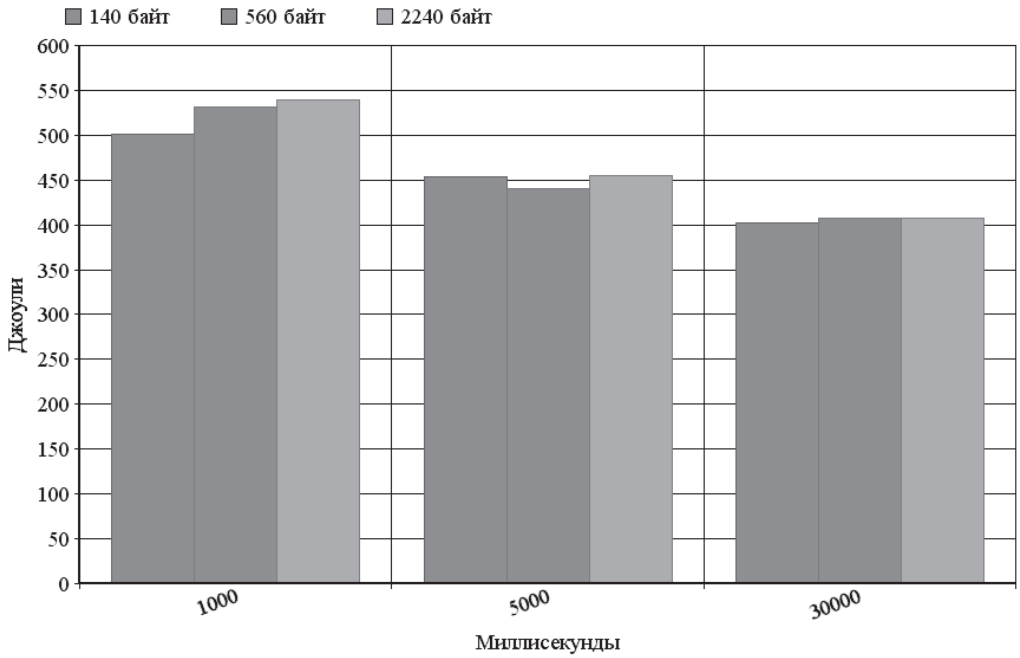

Рис. 6. Метод WebSocket через Chrome

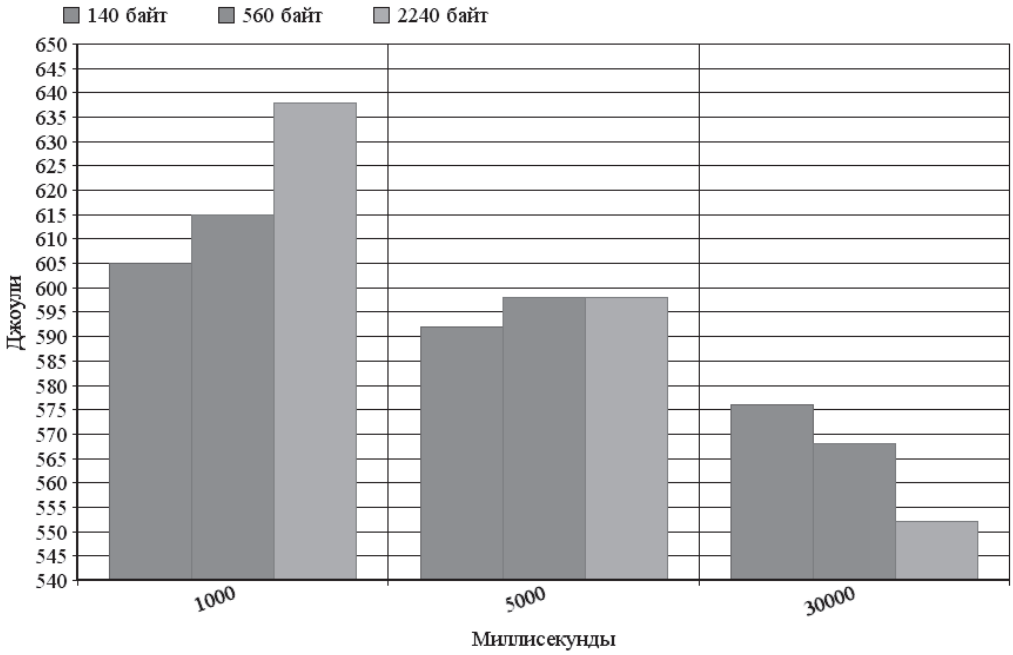

Рис. 7. Метод WebSocket через Firefox

Заключение

Если рассматривать энергопотребление разных видов приложений с точки зрения рядового пользователя, то Chrome показывает лучшую среднюю энергоэффективность.

В ходе исследования было установлено, что самый простой метод асинхронного обмена данными (Short Polling) является одновременно и самым энергозатратным. Также можно заметить некоторую схожесть в энергопотреблении при использовании методов Long Polling и WebSocket, однако в целом WebSocket проявил себя более энергосберегающим методом работы, несмотря на свои прикладные расходы.

Можно сделать вывод, что при использовании браузера от Google, а также оптимизации сетевого кода в сторону использования WebSocket можно добиться приблизительно двукратного улучшения энергоэффективности в сравнении с использованием метода Short Polling.

Литература

1. *Дакетт Дж.* Javascript и jQuery. Интерактивная веб-разработка / пер. с англ. М.А. Райтмана; под ред. Е.А. Истоминой. М.: Эксмо, 2017. 640 с.
2. *Дейтел П., Дейтел Х., Уолд А.* Android для разработчиков / пер. с англ. Е.А. Матвеева. 3-е изд. СПб.: Питер, 2016. 512 с.
3. *Куроуз Дж., Росс К.* Компьютерные сети. Нисходящий подход / пер. с англ. М.А. Райтмана. М.: Эксмо, 2016. 912 с.
4. *Медникс З., Дорнин Л., Мик Б., Накамура М.* Программирование под Android. СПб.: Питер, 2012. 496 с.

5. *Никсон Р.* Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / пер. с англ. Н. Вильчинского; ред. Н. Гринчик. 4-е изд. СПб.: Питер, 2017. 768 с.
6. *Олифер В.Г., Олифер Н.А.* Компьютерные сети. Принципы, технологии, протоколы: учебник. 5-е изд. СПб.: Питер, 2017. 992 с.
7. *Пауэрс Ш.* Добавляем Ajax / пер. с англ. В. Красовского; под ред. Е. Кондуковой. СПб.: Питер, 2012. 448 с.
8. *Moroney L., Pars R., Grieb J.* Foundations of ASP.NET AJAX. N.-Y.: Apress, 2007. 268 p.
9. Python и Twisted – Заметки о параллельной обработке данных (мультипроцессности) / Хабр [Электронный ресурс]. – URL: https://habr.com/ru/post/97201/ (дата обращения: 04.12.2020).
10. *Stark J., Jepson B., MacDonald B.* Building Android Apps with HTML, CSS, and JavaScript: Making Native Apps with Standards-Based Web Tools. 2nd ed. Sebastopol, CA: O'Reilly Media, 2012. 178 p.

References

1. Duckett J. (2014) *JavaScript and JQuery: Interactive Front-End Web Development*. 1st ed. Wiley. 640 p.
2. Datel P., Datel H., Wald A. (2015) *Android 6 for Programmers: An App-Driven Approach*. 3rd ed., Pearson. 422 p.
3. Curose J.F, Ross K.W. (2012) *Computer Networking: A Top-Down Approach*. 6th ed. Pearson. 864 p.
4. Mednieks Z., Dornin L., Meike B., Nakamura M. (2011) *Programming Android*. 1st ed. Sebastopol, CA, O'Reilly Media. 504 p.
5. Nixon R. (2014) *Learning PHP, MySQL & JavaScript with jQuery, CSS & HTML5*. 4thed. Sebastopol, CA, O'Reilly Media. 812 p.
6. Olifer V.G., Olifer N.A. (2017) *Komp'yuternye seti. Printsipy, tekhnologii, protokoly: uchebnik* [Computer Networks. Principles, Technology, Protocols]. 5th ed. Saint Petersburg, Piter Publishing House. 992 p. (in Russian).
7. Powers Sh. (2007) *Adding Ajax*. 1sted. Sebastopol, CA, O'Reilly Media. 400 p.
8. Moroney L., Pars R., Grieb J. (2007) *Foundations of ASP.NET AJAX*. N.-Y., Apress. 268 p.
9. Python and Twisted – Notes on Parallel Data Processing (Multiprocessing). *Habr*. Available at: https://habr.com/ru/post/97201/ (date of the application: 04.12.2020) (in Russian).
10. Stark J., Jepson B., MacDonald B.(2012) *Building Android Apps with HTML, CSS, and JavaScript: Making Native Apps with Standards-Based Web Tools*. 2nd ed. Sebastopol, CA, O'Reilly Media. 178 p.