АНАЛИЗ ТЕХНОЛОГИИ WEBSOCKETS

Губарев И.А., Бурак Д.И.

Институт информационных технологий БГУИР,
г. Минск, Республика Беларусь

Савенко А.Г. – м.т.н., старший преподаватель

В статье проводится анализ технологии WebSocket, рассматривается принцип работы, структура пакета данных, анализируются преимущества и недостатки данной технологии.

WebSockets – это расширение стандартного HTTP-протокола, которое позволяет устанавливать постоянную двухстороннюю асинхронную связь между клиентом (браузером) и сервером без ограничения на тип передаваемых данных в режиме реального времени, пересылать любые данные, на любой домен, безопасно и почти без лишнего сетевого трафика. WebSocket работает над TCP по защищённому соединению (протокол WSS) и обычному (протокол WS).

Предшественником данной технологии был AJAX – подход к построению интерактивных пользовательских интерфейсов веб-приложений, разработанный в 2005 годуизаключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. Осуществлялось это путем отправки javascript асинхронных HTTP запросов. Это позволяло создавать интерактивные сайты с динамически изменяемы информацией.

Технология AJAX имеет свои преимущества и недостатки. К преимуществам можно отнести:

1) Распространённость. Является самой популярной технологией для обмена информацией между сервером и клиентом;

2) Простота реализации. Библиотека для работы ajax встроена в JS. На стороне севра не требуется сторонние для работы с технологией. Т.к. запрос идет к серверу по средствам HTTP запроса.

К недостаткам технологии можно отнести:

1) Отсутствие постоянного подключение к серверу. Для получение обновленных данных клиент должен постоянно посылать данные на сервер для получение новых данных. Так как не всегда есть новые данные на сервер клиент будет впустую нагружать сервер;

2) Для персонализированных данных при каждом запросе нужна индикация;

3) Используется ресурсный протокол HTTP. Пакеты HTTP имеет большой размер. При частом использование запросы будет потреблять много трафика вследствие этого система будет тормозить при низкой скорости интернета;

4) сервер не может обмениваться информацией с клиентом без его инициативы. Сервер не может предать новые данные без нового запроса от пользователя.

Дальнейшим развитием технологии явилась технология WebSocket.

Начальное соединение устанавливается с помощью протокола HTTP, а вот дальнейшее общение клиента с сервером происходит по протоколу WebSocketProtocol имеющим структуру пакета представленную на рисунке 1.

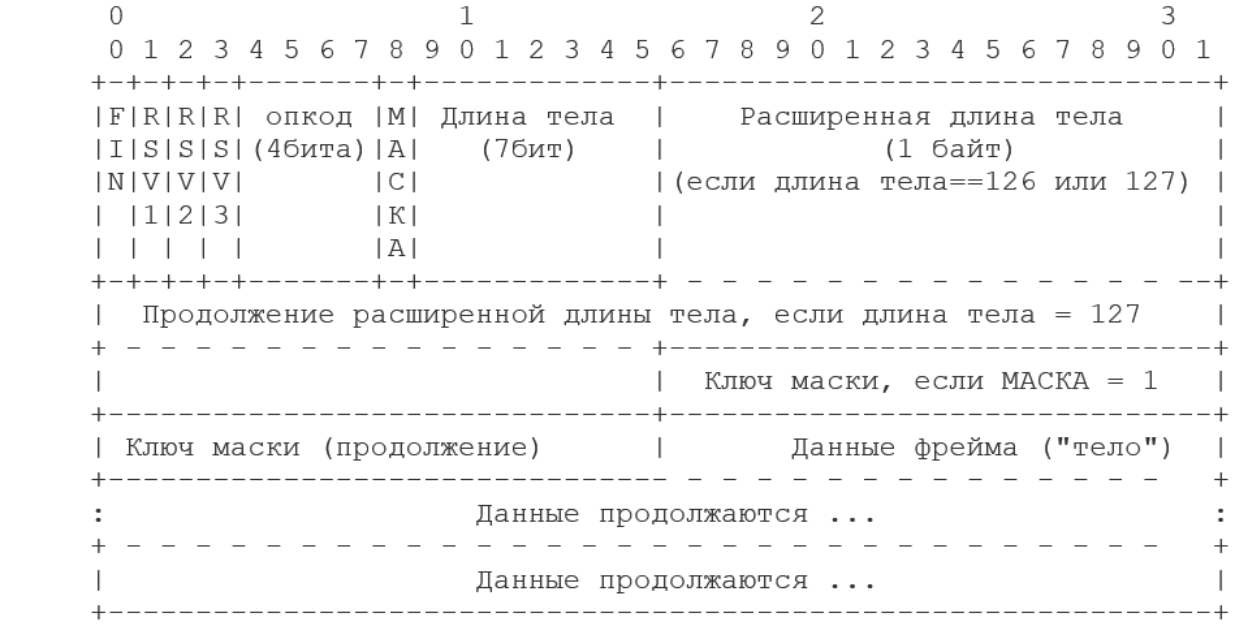

Рисунок 1 – Структура пакета WebSocketProtocol

FIN: равен 1 бит и обозначает заключительный пакет уже отправленных данных (будь то файл или другие большие фрагментированные данные). У всех кадров, кроме последнего, этот фрагмент установлен в 0, у последнего – в 1. Если сообщение состоит из одного единственного фрейма (кадра), то FIN в нём равен 1.

RSV1, RSV2, RSV3: по 1 бит каждый. В обычном WebSocketони равны нулю и предназначены по большей части для расширения протокола. Расширение может записать в эти биты свои значения.

Опкод: равен 4 бита и задаёт тип кадра, который позволяет интерпретировать находящиеся в нём данные. Возможные значения:

– 0x1 обозначает текстовый фрейм;

– 0x2 обозначает двоичный фрейм;

– 0x3-7 зарезервированы для будущих кадров с данными;

– 0x8 обозначает закрытие соединения этим кадром;

– 0x9 обозначает PING;

– 0xA обозначает PONG;

– 0xB-F зарезервированы для будущих управляющих фреймов;

– 0x0 обозначает фрейм-продолжение для фрагментированного сообщения. Он интерпретируется, исходя из ближайшего предыдущего ненулевого типа.

Маска: равна 1 бит. Если этот бит установлен, то данные фрейма маскированы. Более подробно маску и маскирование мы рассмотрим далее.

Длина тела: 7 битов, 7+16 битов, или 7+64 битов.

Если значение поле «Длина тела» лежит в интервале от 0 до 125, то оно обозначает длину тела (используется далее), а если равно 126, то следующие 2 байта интерпретируются как 16-битное беззнаковое целое число, содержащее длину тела. Если 127, то следующие 8 байт интерпретируются как 64-битное беззнаковое целое число, содержащее длину.

Такая хитрая схема нужна, чтобы минимизировать накладные расходы. Для сообщений длиной 125 байт и меньше хранение длины потребует всего 7 битов, для бóльших (до 65536) – 7 битов + 2 байта, ну а для ещё бóльших – 7 битов и 8 байт. Этого хватит для хранения длины сообщения размером в гигабайт и более.

Ключ маски: равен 4 байта. Если бит Маска установлен в «0», то этого поля нет. Если в «1», то эти байты содержат маску, которая налагается на тело.

Данные фрейма (тело).

Состоит из «данных расширений» и «данных приложения», которые идут за ними. Данные расширений определяются конкретными расширениями протокола и по умолчанию отсутствуют. Длина тела должна быть равна указанной в заголовке.

Фрагментация – позволяет отправлять сообщения в тех случаях, когда на момент начала отправки пакетов полный размер ещё неизвестен.

В протокол встроена проверка связи при помощи управляющих фреймов типа PING и PONG.

Этот функционал встроен в браузерную реализацию, так что браузер ответит на PING сервера, но управлять им из JavaScript нельзя.

Чистое закрытие – сторона, желающая закрыть соединение (при том что обе стороны в WebSocket равноправны) отправляет закрывающий кадр (опкод 0x8), в теле которого указывает причину закрытия. В браузерной реализации эта причина будет содержаться в свойстве «reason» события «onclose».

Наличие такого фрейма позволяет отличить «чистое закрытие» от обрыва связи.

Можно выделить следующие преимущества технологии WebSocket:

1) Комплексные веб-приложения. Нет ограничений на количество сессий на сервере.

2) Кросс-доменные приложения. Возможно использовать кросс-доменное запросы (можно посылать запросы с любого домена).

3) Время жизни канала. Соединение с сервером может быть использовано не ограниченно количество времени.

4) Высокая скорость работы. Высокая скорость работы обеспечивается путем использования TCP пакетов для передачи данных, а также использованием собственного протокола передачи данных.

К недостаткам технологии WebSocket можно отнести следующие:

1) Сложность реализации. Для реализации технологии существует мало инструментов. Поэтому разработчикам нужно реализовывать свою архитектуру для реализации технологии;

2) Малое распространение на данный момент;

3) Высокая сложность внедрение в систему где уже используется другие схожие технологи (ajax). Для внедрения в систему нужно изменять бизнес логику или перевести сервера на другую технологии (Node.js).