**Министерство науки и высшего образования Российской Федерации** **Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Московский государственный технический университет** **имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Курс «Разработка интернет-приложений»

Домашнее задание №1

«Код Хэмминга»

Выполнил:

студент группы ИУ5-53Б

Лобанов И. И.

Проверил:

Канев А.И.

2023 г.

**Задание:**

Реализовать поверх long polling свой канальный уровень для исправления ошибок (код Хемминга).

Данное задание учит идее современных беспроводных сетей, что случаются искажения битов. Одна из сторон случайно «портит» биты в данных.

Нужно восстановить код или найти ошибку. На экране выводятся оригинальное сообщение, «испорченное», восстановленное. И можно сравнить, смогли найти ошибку или нет.

**Что такое Long Polling ?**

Long Polling - это технология, которая используется при построении модели клиент - сервер. Её принцип заключается в следующем: вместо традиционного ответа сервера на запрос клиента, мы устанавливаем “длинное” (оно же непрерывное) соединение, которое означает, что две стороны находятся в полной готовности получить/отправить данные.

Когда клиент отправляет запрос на сервер (рис. 1), он ожидает получения измененных данных, которые ему необходимы.



Рисунок 1. Условная схема Long Polling.

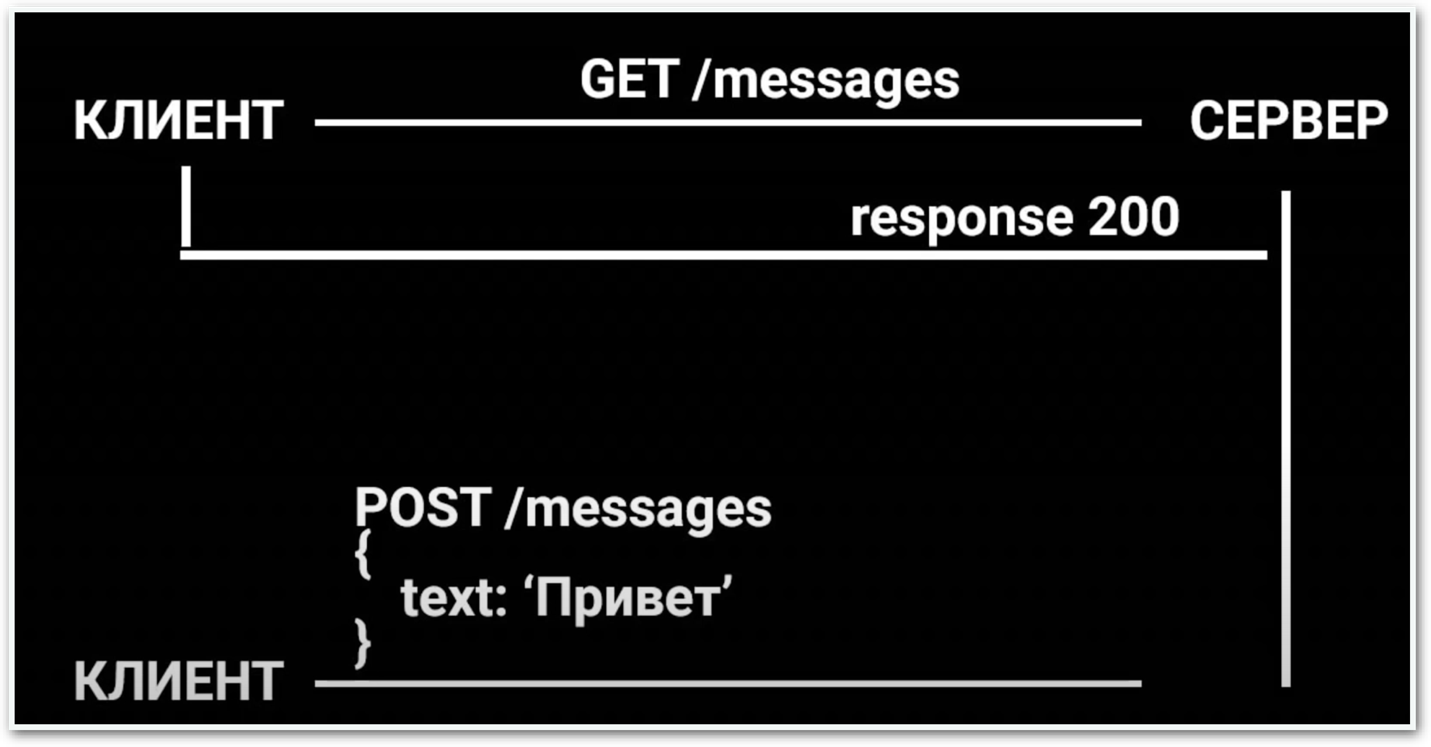
То есть, если при GET запросе данные, которые мы запрашиваем, не поменялись, тогда сервер не отправляет ответ, а ждет изменений, чтобы уже потом сформировать данные и направить источнику запроса (рис. 2).

Рисунок 2. Получение ответа от сервера.

После получение ответа от сервера, клиент сразу отправляет запрос, чтобы снова “прослушивать” изменения на сервере (цикл повторяется). Стоит сказать, что клиент будет непрерывно связан с сервером, пока не появятся изменения или пока не истечет заданное время ожидания.

**Преимущества использования Long Polling:**

* Клиент получает обновления мгновенно по мере их доступности.
* Уменьшается нагрузка на сервер, поскольку клиенты не постоянно опрашивают его (например, Polling).
* Эффективно использовать ресурсы сети, так как данные передаются только при наличии обновлений или событий.
* Позволяет создавать реактивные веб-приложения с встроенной функциональностью обновления в реальном времени.

**Несмотря на эффективность Long Polling, есть и некоторые недостатки:**

* Увеличивается задержка между обновлениями данных, поскольку клиент должен ждать ответа от сервера.
* Требует поддержки со стороны сервера для работы с множеством ожидающих клиентов и управления подключениями.
* Может привести к увеличению нагрузки на сеть из-за постоянного поддержания открытого соединения.

**Область применения:**

Длинные опросы прекрасно работают, когда сообщения приходят редко. Если сервер быстро отдает ответ на запрос, лучше использовать такой протокол как Websocket.

**Код Хэмминга.**

Код Хэмминга - это алгоритм, который позволяет сохранить данные при передаче по сети в целостности посредством использования контрольных битов. Благодаря данному алгоритму, помехи, возникающие в процессе запроса/ответа, будут зафиксированы и восстановлены. Однако есть некоторое ограничение на количество восстанавливаемых битов информации: в стандартной реализации мы можем восстановить только 1 бит информации, и обнаружить 2 ошибки, но не исправить ни одну из них (делается с помощью бита четности).

**Реализация алгоритма**

Данный алгоритм состоит из двух этапов:

* Кодирование информации на сервере, используя контрольные биты с порядковым номером степени двойки.
* Добавление бита четности.
* Пересчет контрольных битов на сервере и их сравнение с контрольными битами на сервере.
* Сравнение бита четности доставленного сообщения и бита четности пересчитанного сообщения.
* В случае совпадения всех битов сообщение считается доставленным без потерь.
* Если биты четностей совпадают, а контрольные нет, значит возникло 2 ошибки.
* Если контрольные биты и биты четностей не совпадают, тогда 1 ошибка.
* Если контрольные биты совпадают, а биты четностей нет, значит возникла 1 ошибка: бит четности попортился.

Для наглядности картины приведем диаграмму развертывания (рис. 3). Пусть веб-браузер с помощью WEB API посылает запрос на сообщение HTTP-серверу.

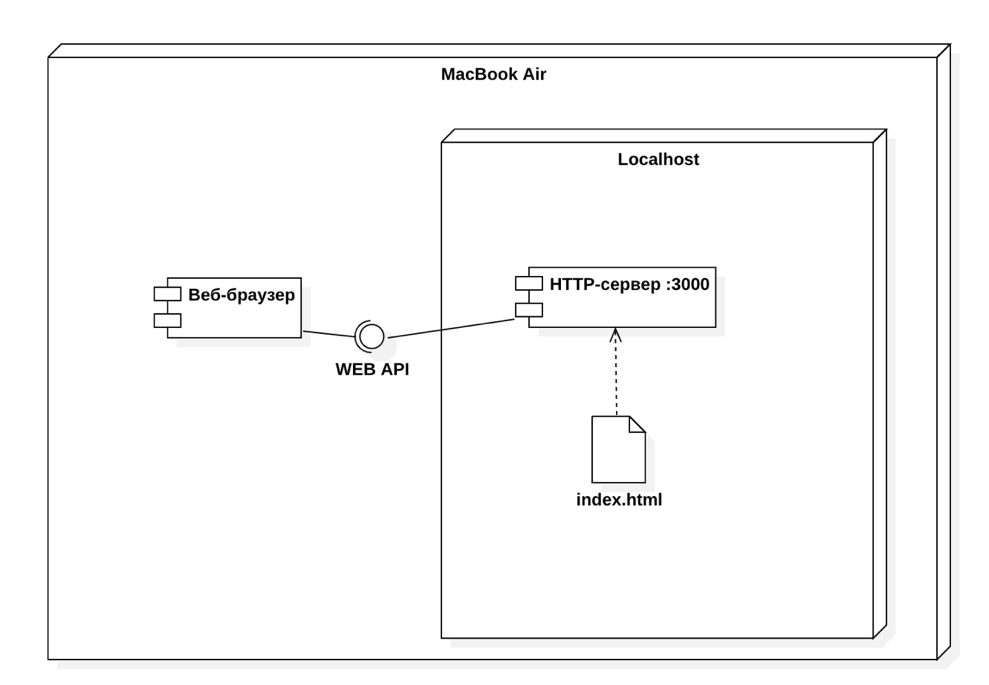


Рисунок 3. Диаграмма развертывания.

Воспользуемся картинкой (рис. 4), чтобы понять принцип работы контрольных битов.

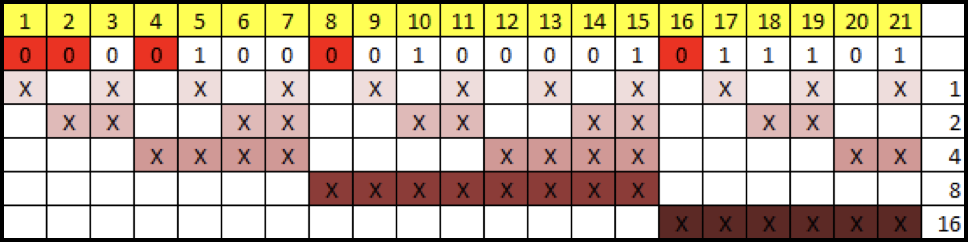


Рисунок 4. Кодирование информации с помощью Хэмминга

Условно, у нас есть 21 байт информации на сервере, который нужно преобразовать в код Хэмминга и передать на клиент. Мы сможем сделать самокорректирующиеся данные с помощью контрольных битов. На рис. они помечены красным цветом. Их местоположение задается уравнением 2**n**, где n - неотрицательное целое число. Расположение понятно, теперь по какому принципу они заполняются и что контролируют?

Воспользуемся следующим правилом: *контрольный бит с номером N контролирует все последующие N бит через каждые N бит, начиная с позиции N.*

Глядя на рис. 3, мы видим, что бит с индексом 1 (контрольный бит) контролирует помимо себя 3, 5, 7, …. 21 биты, бит с индексом 2 - 2и3, 6и7… и т.п. После того как мы поместили контрольные нулевые биты, необходимо посчитать их значения. Для этого мы сложим все значения контролируемых ими бит соответственно, т.е. контрольный бит с индексом 1 будет складывать значения битов 3,5,7…, а контрольный бит с индексом 2 будет складывать значения в битах 2,3,6,7 и т.д. После получения общего числа единиц в контролируемых битах мы кладем в контролируемый бит остаток от деления на 2 полученного числа (можно использовать операцию XOR). Таким образом считаются все контрольные биты. Данные отправляются на клиент (см рис.5).

После получения данных происходит аналогичный счет контрольных битов полученной информации. В случае полного совпадения данные считаются успешно доставленными. Если же контрольные биты не совпадают, значит были помехе при передаче.

Код Хэмминга позволяет решить эту ситуацию следующим образом: при изменение какого-либо бита информации поменяют свое значение и контрольные биты, порядковые номера которых и составляет индекс испорченного бита. Имея номер бита, мы конвертируем его и получаем исходные данные с клиента.

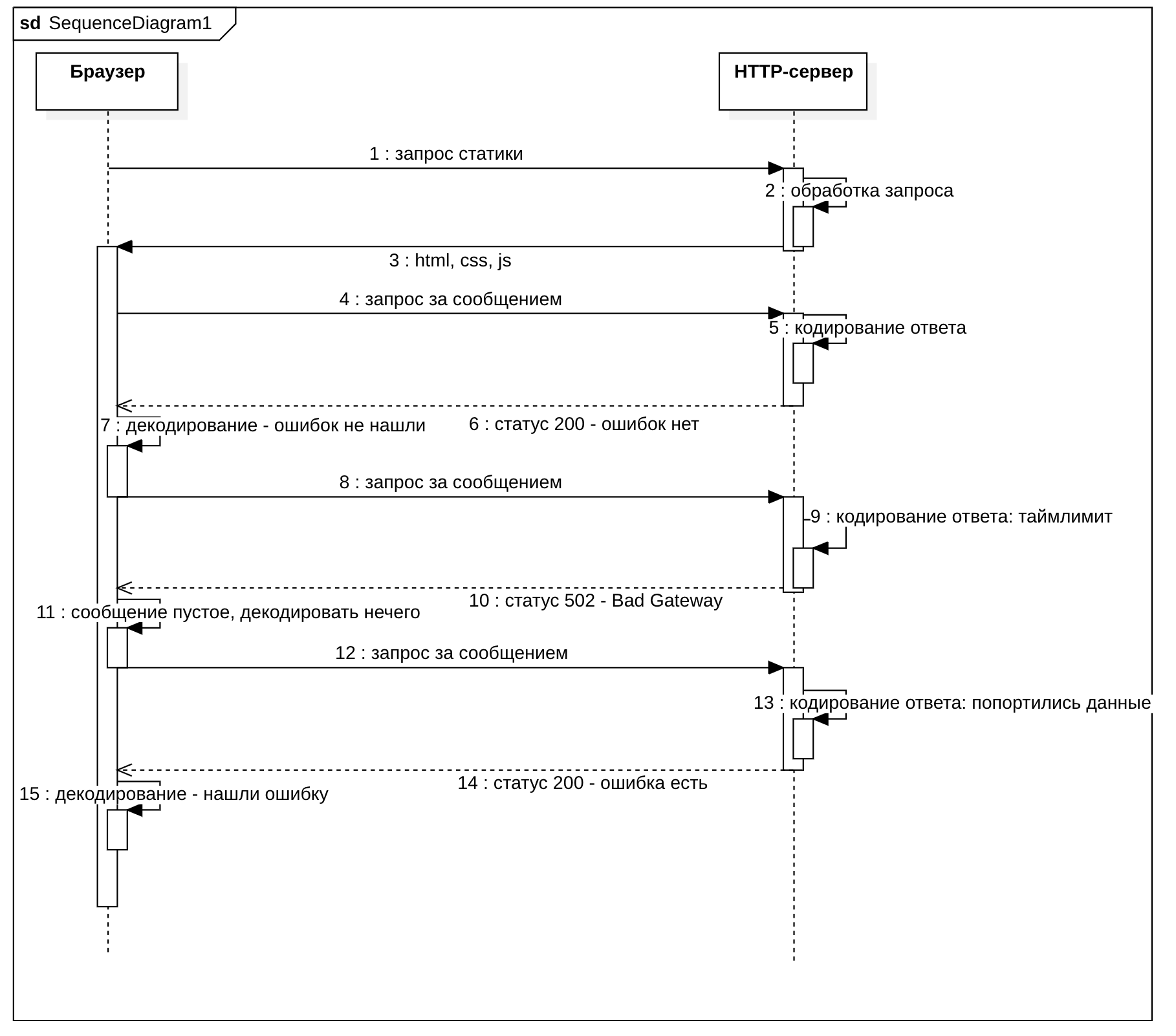


Рисунок 5. Диаграмма последовательности отправки сообщения и его декодирования.