RFC 3205

Использование протокола HTTP, в качестве базового уровня для приложений.

Причины для использования

- знакомый и хорошо известный протокол

- совместим с существующими браузерами

- возможность реиспользования существующих клиентских и серверных библиотек

- удобно, если для проекта в любом случае необходимо использовать http

Причины для использования

- скорость прототипирования сервера (например cgi)

- возможность использования существующих механизмов безопасности (HTTP digest auth, SSL, TLS)

- пропускается стандартными файерволами и/или прокси-серверами

При проектировании своего собственного протокола поверх уже существующего, возникают вопрос о том, нужно ли вообще использовать существующий протокол? А если нужно, то каким образом?

1. Должно ли приложение использовать порт, отличный от 80?

2. Должно ли приложение использовать стандартные методы (GET/POST etc.) или необходимо объявить свои собственные методы?

3. Нужно ли приложению использовать стандартную схему (идентификатор протокола) “http:” или необходимо использовать свою?

4. Есть ли необходимость в объявлении своих собственных типов MIME?

Проблемы выбора при проектировании протокола с использованием HTTP

Необходимо определить, может ли вообще HTTP протокол использоваться в качестве основы, а если может, то в каком объеме?

1. Сложность (Complexity)

HTTP – простой протокол. Однако существуют расширения увеличивающие его сложность: поддержка постоянных соединений, байтовые диапазоны, согласовывание контента, поддержка кэширования и другие.

Эти расширения удобны для WEB-приложений, но далеко не всегда требуются приложениям работающим поверх HTTP.

Даже при использовании существующих библиотек для работы с HTTP, сложность взаимодействия может получится больше, чем при использовании своего протокола.

2. Избыточность (Overhead)

Использование TCP сессий и полей HTTP приводит к задержкам и избыточности при частой передаче данных небольшого размера.

С другой стороны, можно использовать постоянные соединения для передачи нескольких запросов/ответов HTTP.

3. Безопасность (Security)

Не для всех приложений может подойти схема с использованием digest-like аутентификации или TLS, поскольку для них существует множество ограничений.

Проблема basic/digest аутентификации — общие секретные данные. Подходит для небольших групп.

3. Безопасность (Security)

Нет смысла использовать протокол HTTP только ради использования встроенных в фреймворк систем безопасности.

3. Безопасность (Security)

SSL/TLS позволяют аутентифицировать обе стороны, однако есть проблема получения корневых и самоподписанных сертификатов. Кроме того клиент не всегда может иметь возможность аутентификации.

Используемые по-умолчанию механизмы шифрования SSL/TLS, как правило слабы и, при использовании современного оборудования, расшифровываются за несколько дней или часов.

4. Совместимость с прокси, сетевыми экранами и NAT

Это одна из основных причин для использования HTTP.

5 вопросов на которые надо ответить, при решении использовать HTTP

1. Подходит ли HTTP в качестве подходящего транспорта для приложения?

5 вопросов на которые надо ответить, при решении использовать HTTP

2. Можно ли будет использовать новый протокол в браузерах без изменений?

5 вопросов на которые надо ответить, при решении использовать HTTP

3. Подходят ли существующие механизмы безопасности HTTP для приложения?

5 вопросов на которые надо ответить, при решении использовать HTTP

4. Подходит ли парадигма “теории ответных кодов” для вашего приложения?

5 вопросов на которые надо ответить, при решении использовать HTTP

5. Есть ли необходимость в использовании HTTP для сервера, вне зависимости от вашего приложения?

Проблемы при использовании 80 порта

Необходимо использовать новый порт, если любое из этих условий подходит для вашего приложения:

- “новый сервис” и классический http-сервер используют различные наборы данных;
- существует причина для создания нового сервиса в виде отдельного процесса или отдельного от “классического” кода;
- существует причина для необходимости легкого различения трафика относящегося к вашему приложению;

В остальных случаях предпочтительно создавать новые расширения, за счет добавления новых методов.

Проблемы с использованием “http:”

Схема указывает на тип используемого ресурса или сервиса. Для решения можно использовать критерии:

- планируется широкое распространение схемы;
- в любом случае будет использоваться новый порт;
- требуется отличная от http предварительная настройка либо взаимодействие;
- пользовательские приложения могут значительно облегчить жизнь пользователя (например: http: и mailto: в браузере)

Проблемы с использованием типов MIME

Типы MIME подсказывают приложению, как обработать данные ответа. При этом надо уделить внимание:

- может ли использоваться существующие фреймфорки (например text/directory или XML)?
- можно ли использовть типы сообщений или multipart MIME?
- планируется ли пересылать эти же данные по электронной почте?
- MIME описывает тип данных, и не должен использоваться для описания семантики.

Проблема выбора между существующими и новыми методами HTTP

Хотя предпочтительно использовать новые методы вместо использования нового порта, это само по себе не означает, что можно отказаться от нового порта или новой схемы URL.

Использование существующего кода

Хотя использование существующего кода для клиента, сервера или прокси является одной из важнейших причин, необходимо помнить, что протокол HTTP не проектировался для использования в качестве траспортного, поэтому в существующих реализациях многие вещи считаются неизменными или существуют проблемы с расширением этих реализаций.

Проблемы при использовании “теории ответных кодов”

Нежелательно расширять статусные коды – возможно, что они будут закреплены за другими ситуациями.

Основная проблема возникает при использовании прокси- или кошируюшего сервера, т.к. стандартные реализации не знают о ваших расширениях.

Желательно придерживаться следующего руководства:

- использовать стандартные статусы для уведомления об ошибках в запросе или заголовке запроса;

- если ошибки обнаружены в теле сообщения, то можно использовать коды 200 или 500 для уведомления, причем необходимо предусмотреть механизм “антикэширования” в случае использования кода 200;

- приложение может возвращать код 200 в случае успеха и неуспеха, в таком случае приложение должно определить конкретную ситуацию в теле сообщения;

- приложение, которое не может корректно работать с прокси- и кэширующими серверами не должно использовать протокол HTTP в качестве своей основы.

Выводы и рекомендации по использованию HTTP, как основы для своего приложения

Все протоколы должны предоставлять адекватный уровень защиты.

Требования к защите зависят от приложения, однако требуется помнить, что использование HTTP/TLS не предоставляет адекватной защиты во всех случаях и окружениях.

Новые протоколы – вне зависимости, основаны они на http или нет – не должны маскироваться под уже существующие протоколы, с целью обойти пользовательские сетевые фильтры.

В общем случае новые протоколы или сервисы не должны использовать существующие схемы такие, как “http:”

Каждый новый протокол, базирующийся на HTTP обязан учитывать в своей спецификации, каким образом будет использоваться HTTP, включая то, как клиент и сервер будут взаимодействовать посредством прокси-сервера.

Необходимо следовать руководству, описанному выше, по использованию статусных кодов HTTP.