TCP

Облачный мониторинг и управление

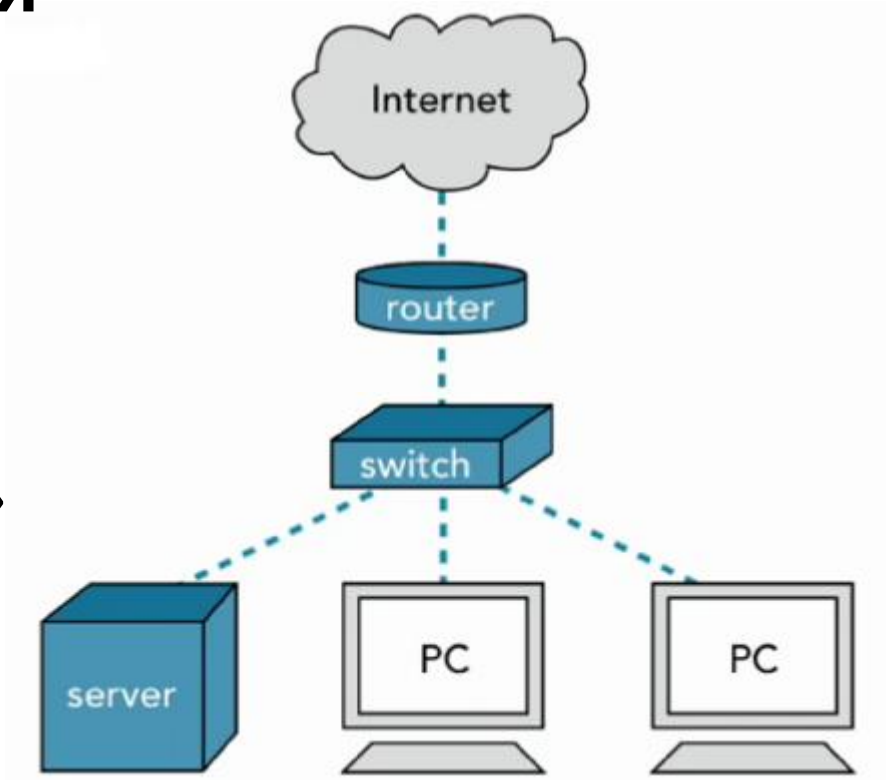
План работы

- «Облака»
- World Wide Web
- HTTP
- Web ресурсы
- Web сервисы
- Поддержка от Microchip
- Демонстрация облачного мониторинга и управления
- Демонстрационный сервер
- Реализация встраиваемого HTTP клиента
- Лабораторная работа

«Облака»

Что такое «облака»?

- Широко используется как метафора, обозначающая интернет
- Подразумевается взаимодействие «поставщика услуги» и «потребителя услуги»



Что такое «облака»? (NIST)

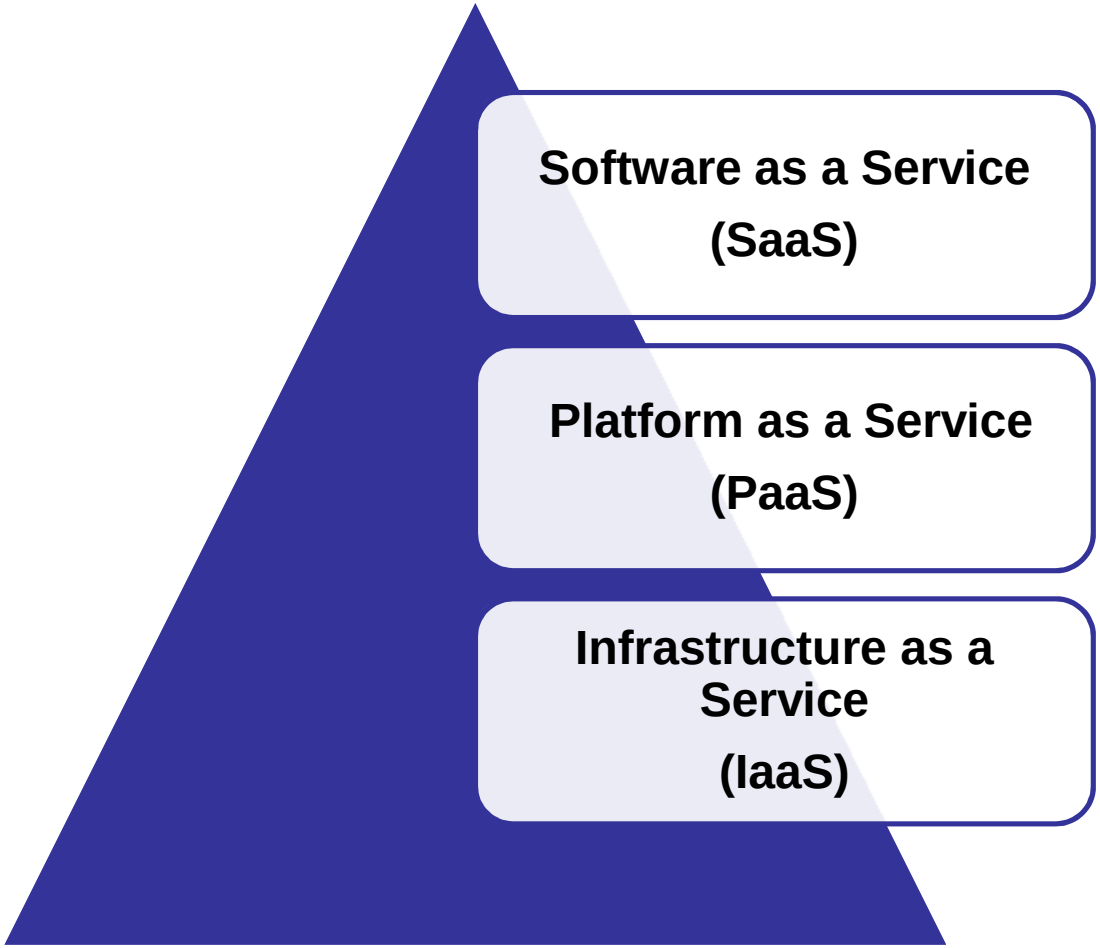
- **Сокращение для “Облачные вычисления”**

- **Определение NIST [1]:**

  Облачные вычисления – это модель, позволяющая быстро и с минимальными для поставщика услуг усилиями предоставить удобный сетевой доступ к выделяемому пулу конфигурируемых компьютерных ресурсов (т.е. сетей, серверов, памяти, приложений и сервисов).

1) http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Модели сервисов NIST



http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Инфраструктура как сервис (IaaS)

- **Определение NIST:**

  Поставляемая клиенту возможность использовать процессор, память, сети и другие компьютерные ресурсы таким образом, что клиент может развертывать и запускать на этих ресурсах произвольные приложения …

  Клиент не управляет низлежащей облачной инфраструктурой, но имеет контроль над операционной системой, памятью и развернутыми приложениями…

http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Платформа как сервис (PaaS)

- **Определение NIST[1]:**

  Поставляемая клиенту возможность развертывать на облачной инфраструктуре собственные или заимствованные приложения …

  Клиент не управляет низлежащей облачной инфраструктурой …, но имеет контроль над развернутыми приложениями…

http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Приложение как сервис (SaaS)

- **Определение NIST[1]:**

  Поставляемая клиенту возможность использовать приложения провайдера на облачной инфраструктуре …

  Клиент не управляет низлежащей облачной инфраструктурой…

http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Применение облачных вычислений

- **Облачные вычисления**
  - Предоставление компьютерных ресурсов по запросу
- **Облачное хранение**
  - Работа с файлами
- **Облачные приложения**
  - Приложения, предоставляемые в интернете. Работа через браузер.
  - За хостинг и IT обслуживание отвечают «облака»

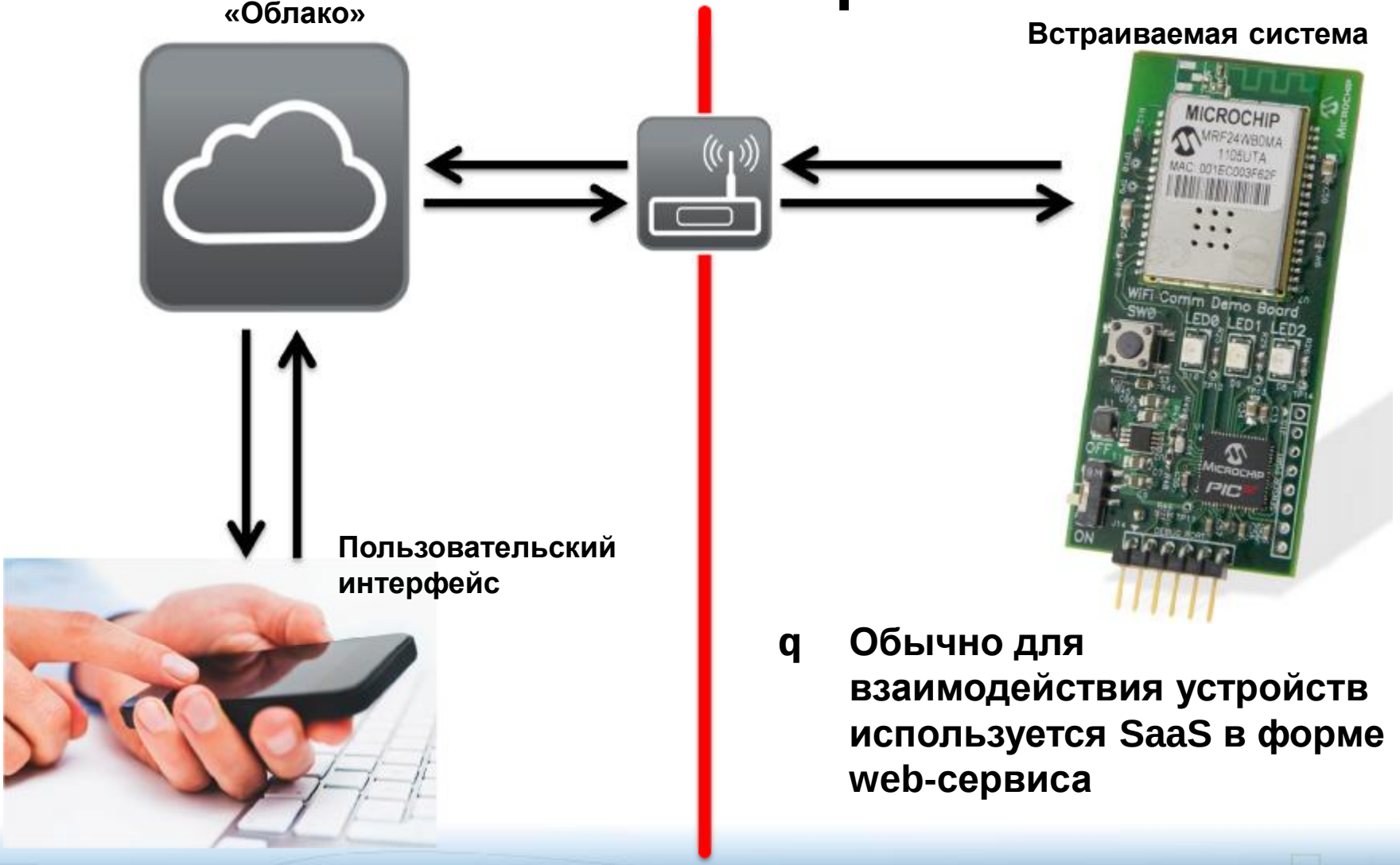
«Облака» для встраиваемых систем

- **Все применения, указанные на предыдущем слайде!**
  - Дополнительная вычислительная мощность, память и т.д.
- **Удаленное обновление ПО**
- **Удаленный мониторинг и управление**
- **Использование сторонних web-сервисов для добавления новых возможностей продукции**
- **Коммуникация «облако» - «облако»**

Облачные вычисления
Как выглядит встраиваемое «облачное» решение?

«Облако»

Встраиваемая система



Пользовательский интерфейс

- Обычно для взаимодействия устройств используется SaaS в форме web-сервиса

World Wide Web

Web сегодня

- Изначально web (www) был создан для удобного удаленного доступа к мультимедийным (преимущественно текстовым) документам.
- Сегодня web – это глобальная распределенная система для создания и обмена информацией
- Масштабируемая, безопасная и надежная для приложений, которые базируются на ее основополагающих принципах
- Преимущественно “human-driven”
- В последнее время быстрый рост использования как платформы для распределенного программирования и автоматизации
  - “Web Services”
  - www.programmableweb.com

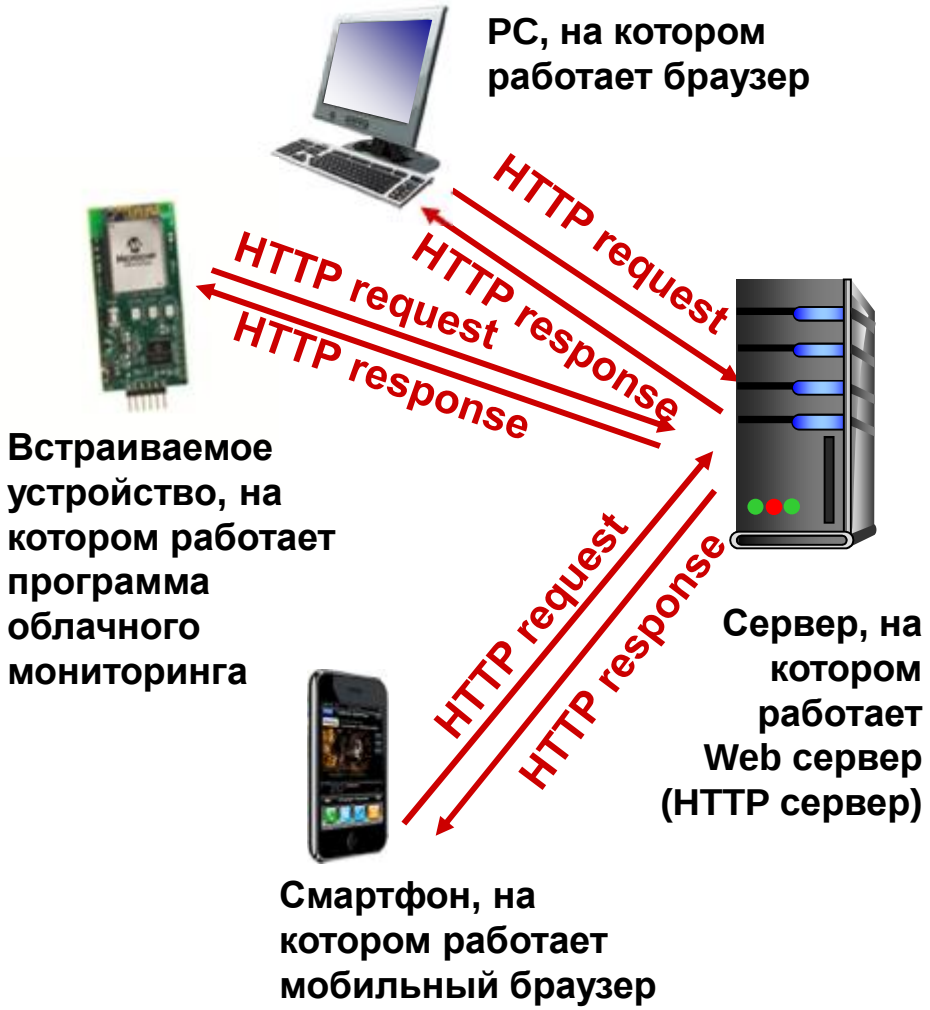
Основные термины Web

- **URI (Uniform Resource Identifier)**
  - Синтаксис для определения адресов доступа к “ресурсам”
- **“Гипертекстовый” документ**
  - Документ, который может содержать ссылки на другой документ
- **Hypertext Transport Protocol (HTTP)**
  - Протокол сообщений для передачи гипертекстовых документов по сети
- **Hypertext Markup Language (HTML)**
  - Язык разметки для создания гипертекстовых документов
- **Representational State Transfer (REST)**
  - «Архитектурный стиль» для Web – набор формализованных рекомендаций по организации взаимодействия в рамках Web
- **Web – сервис**
  - Метод для взаимодействия электронных устройств через Web

HTTP

HTTP

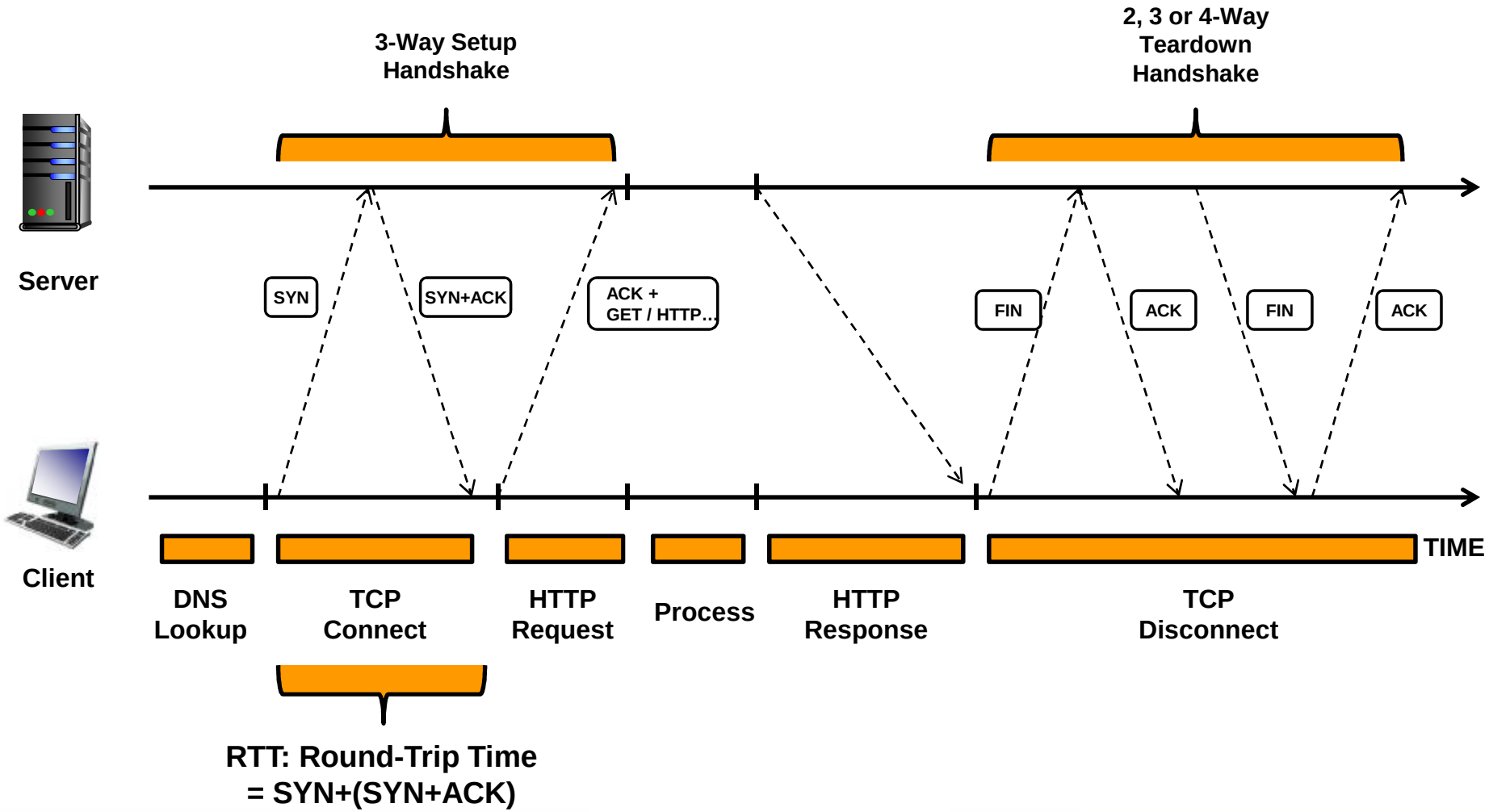
- **HTTP: hypertext transfer protocol (“RFC2616”)**
  - Протокол уровня приложения для Web
- **Модель клиент/сервер**
  - **клиент:** устройство, которое запрашивает, принимает (через HTTP протокол) и потребляет web ресурсы
  - **сервер:** устройство, которое посылает (через HTTP протокол) представление web ресурсов в ответ на запрос



HTTP

- **Использует TCP соединения:**
  - Клиент инициирует TCP соединение к серверу с заданным IP адресом на порт 80
  - Web сервер принимает TCP соединение от клиента
  - Клиент и Web сервер обмениваются HTTP сообщениями
  - TCP соединение закрывается (с любой из сторон)
- **Базовые транзакции не используют состояний (“stateless”)**
  - Работа Web сервера не зависит от информации о предыдущих запросов клиентов
    - **Принимает запрос**
    - **Получает ресурс**
    - **Выдает ответ**
  - Это не означает, что ответ всегда одинаковый т.к для получения ресурса web сервер может обращаться к серверу приложений, который может поддерживать состояния

HTTP Транзакция

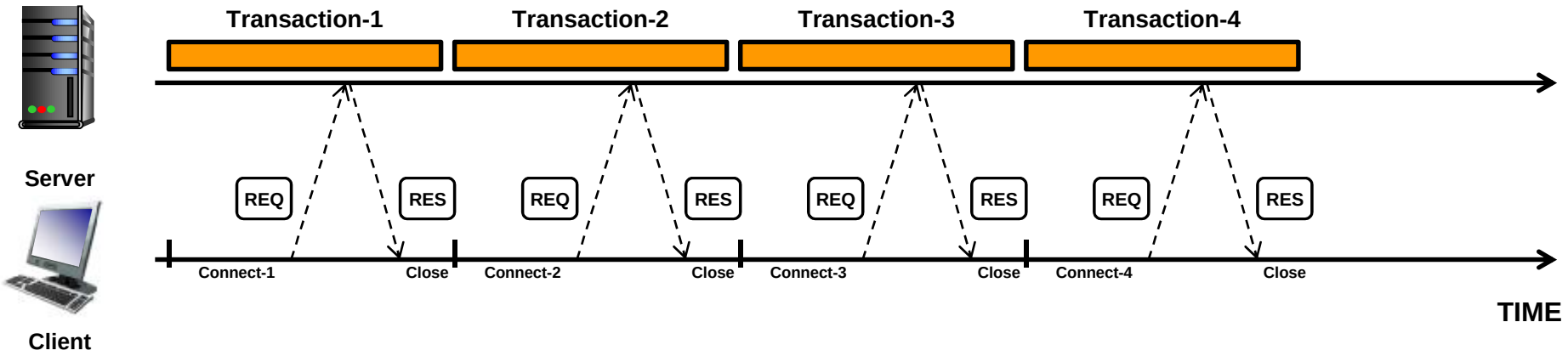


Управление TCP соединением

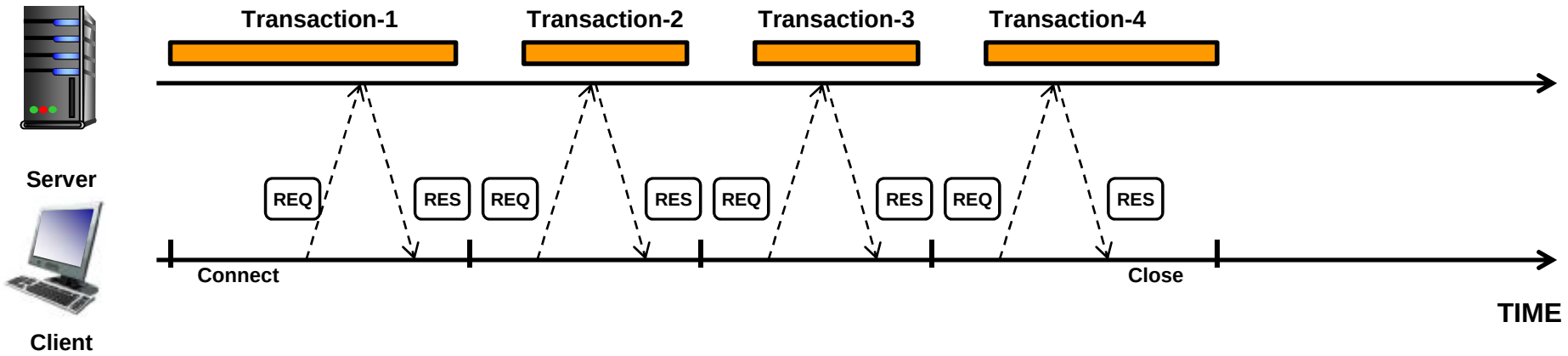
Для нескольких HTTP транзакций

- **Множественные последовательные соединения**
  - 1 транзакция на соединение
- **Постоянные соединения**
  - Несколько транзакций на соединение
- **Конвейерные постоянные соединения**
  - Несколько накладываемых транзакций на соединение
  - Машина состояний на стороне клиента
- **Параллельные соединения**
  - Несколько транзакций через несколько соединений

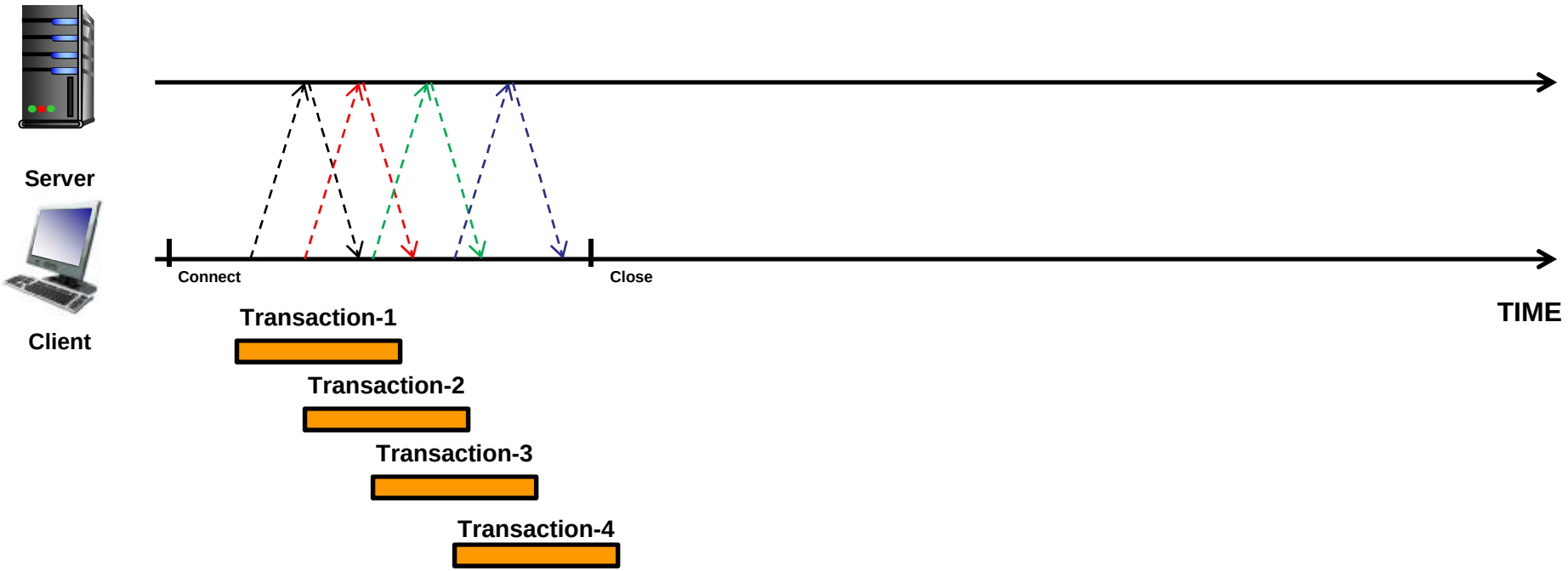
Последовательные соединения



Постоянные соединения



Конвейерные постоянные соединения



Установка постоянных соединений

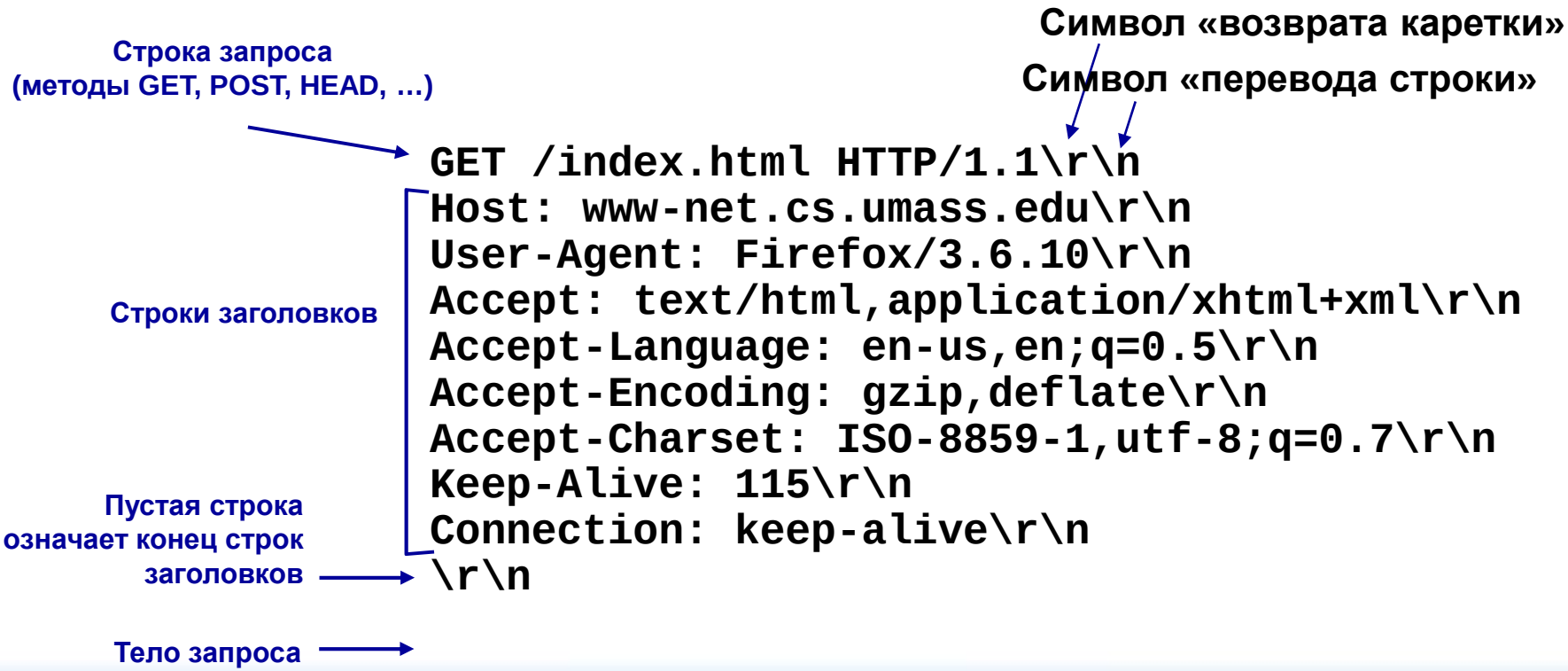
- **HTTP/1.0**
  - Через заголовок запроса:
    - **`Connection: Keep-Alive`**
  - Если сервер желает сохранить соединение, он отвечает с заголовками :
    - **`Connection: Keep-Alive`**
    - **`Keep-Alive: max=5, timeout=120`**
- **HTTP/1.1**
  - Постоянные соединения включены по умолчанию
  - Клиент считает соединение открытым до тех пор, пока не примет от сервера ответ с заголовком
    - **`Connection: close`**
  - Когда клиент хочет завершить передачу запросов, он посылает в последнем запросе заголовок
    - **`Connection: close`**

Установка конвейерных постоянных соединений

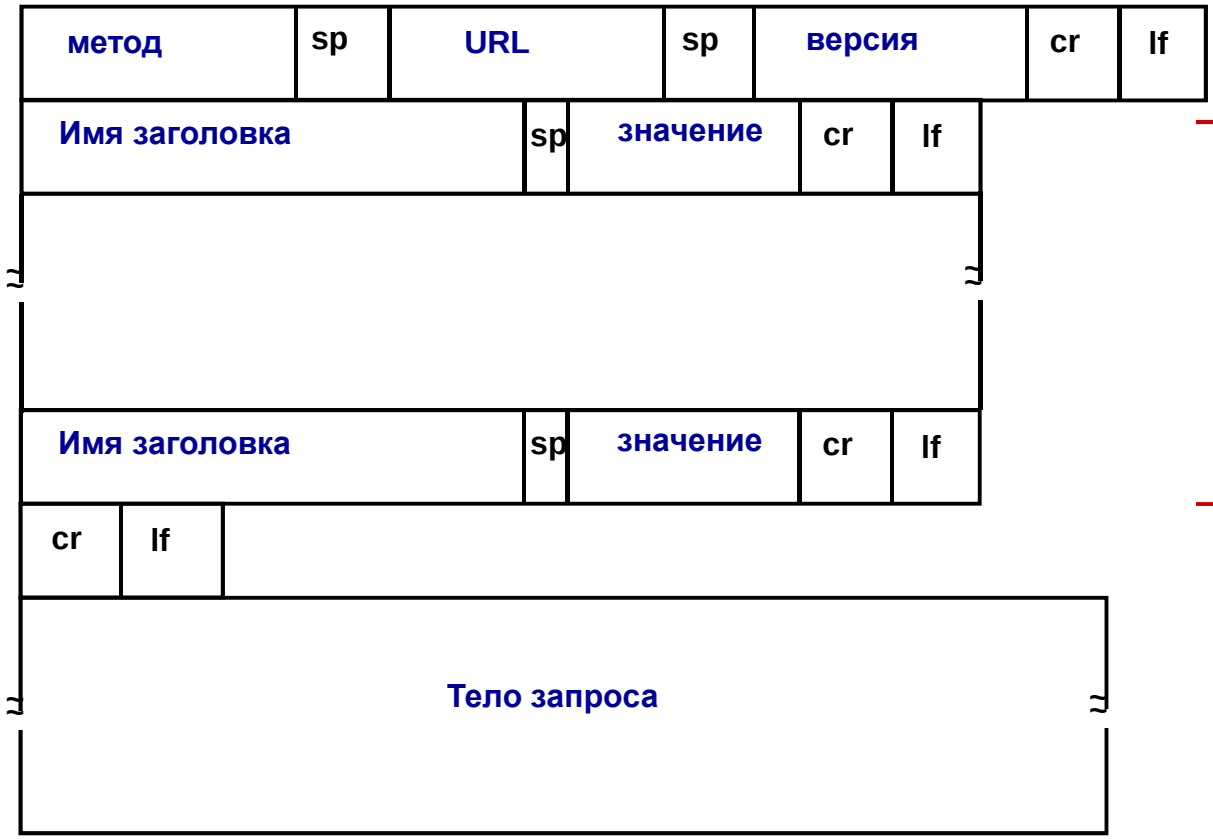
- **HTTP/1.0**
  - Не поддерживается
- **HTTP/1.1**
  - Поддерживается
  - Большинство современных Web серверов работают с такими соединениями без проблем

HTTP Request

- **Два типа сообщений HTTP: *request, response***
- **HTTP request message:**
  - ASCII (читаемый формат)



Сообщение HTTP Request
общий формат



HTTP методы

- **Действия, которые выполняются над ресурсами**
- **Основные методы HTTP/1.1:**
  - GET – запрос представления ресурса
  - POST – отсылка данных на сервер
  - HEAD – запрос только заголовков
  - PUT – замена (полная/частичная) ресурса
  - DELETE – удаление ресурса

HTTP Response

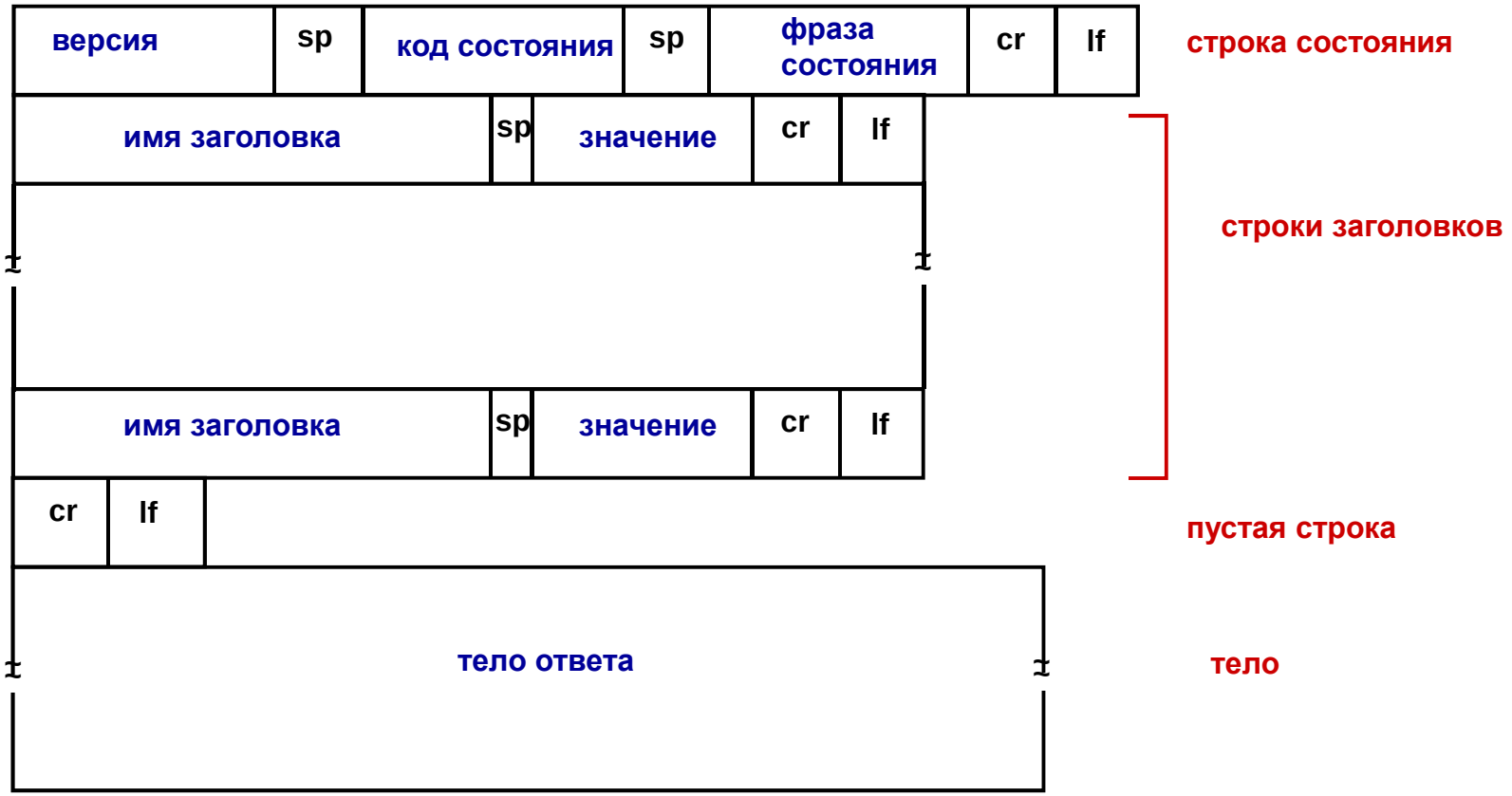
Строка состояния

Строки заголовков

Данные

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

Сообщение HTTP Response
общий формат



Коды состояния HTTP ответов

- **Код и фраза состояния передаются в первой строке ответа сервера**
- **Категории кодов:**
  - 200-299: Коды успешного выполнения
    - **Запрос обработан успешно; конкретный код зависит от типа запроса**
  - 300-399: Коды перенаправления
    - **Запрашиваемый объект перемещен, новое положение указано в ответе**
  - 400-499: Коды ошибки клиента
    - **Запрос не распознан сервером**
  - 500-599: Коды ошибки сервера
    - **Ошибка сервера во время обработки запроса**
- **http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.1**

Авторизация доступа в HTTP

- **HTTP доступ к ресурсам может контролироваться следующими методами авторизации**
  - Базовая авторизация
    - **Передача закодированного значения имя:пароль для данного ресурса**
  - Авторизация по дайжесту
    - **Передача хеш-функции от имени пользователя и других параметров**
- **На практике этих методов не достаточно. Для надежной авторизации необходимо применять в сочетании а авторизацией на транспортном уровне (SSL или TLS)**

Авторизация транспортного уровня (SSL или TLS)

- **Реализуется над TCP уровнем**
- **SSL: Secure Sockets Layer**
- **TLS: Transport Layer Security**
  - Более новая альтернатива SSL
- **Обеспечивает**
  - Авторизацию сервер/клиент
  - Целостность данных
  - Шифрование
- **Идентифицируется схемой “`https`”**
  - Заменяет вызовы API работы с TCP сокетами на специальные вызовы API с поддержкой SSL/TLS
- **Обязательно необходимо использовать для ресурсов с повышенной безопасностью**

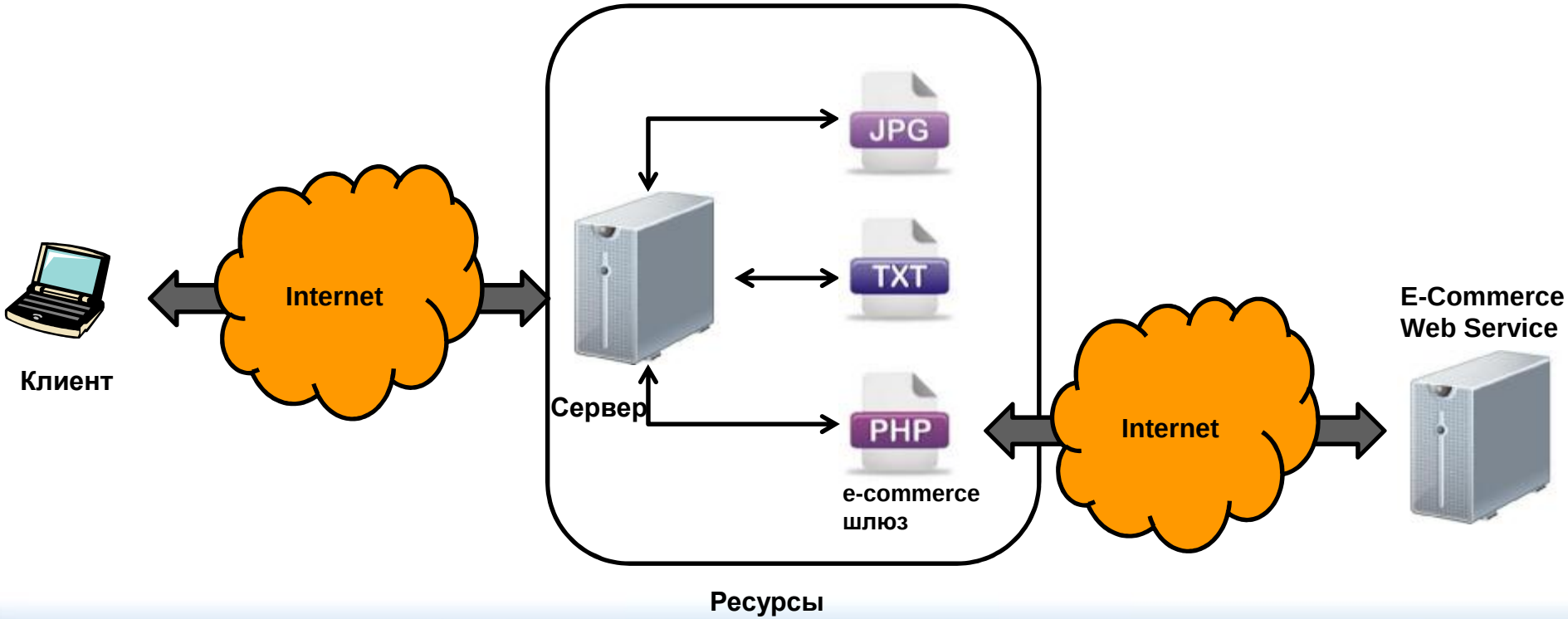
Web ресурсы

Ресурсы

- **Каждая индивидуально адресуемая единица информации в web называется «ресурсом»**
- **Примеры ресурсов :**
  - Документы
  - Изображения
  - “Сегодняшняя погода в СПб”
  - “FFT последних 1024 отсчетов”

Ресурсы

- ..другими словами, ресурсы – это содержимое, которое может предоставить web
- ресурс={идентификатор, представление}



Идентификатор ресурса

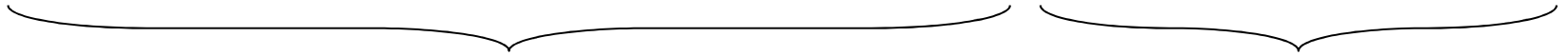
- **Каждый ресурс адресуется уникальным идентификатором**
  - URI – Uniform Resource Identifier
    - **URL – Uniform Resource Locator**
      - Описывает положение ресурса на сервере
    - **URN – Uniform Resource Name**
      - Служит как уникальное имя
- **За выбор идентификатора отвечает автор (владелец) ресурса**
  - Имеет смысл выбирать идентификатор так, чтобы он отображал суть ресурса
    - **`http://myhome.com/basement/windowsensor`**

Анатомия URL

`<scheme>://<host>:<port>/<path>?<query>`

Пример:
URL для ресурса “страница поиска google” с запросом “Microchip”:

`http://www.google.com/search?as_q=Microchip`



Идентификатор ресурса

Запрос (опционально)

Представления

- **Клиент получает представление состояния ресурса**
  - Ресурс (например, «Температура в Санкт-Петербурге») может быть представлен в формате HTML для браузера и в формате XML для встраиваемого устройства
- **Представление является отображением ресурса, а не самим ресурсом**
  - Можно выбирать наилучшие представления для разных клиентов!

Представления в HTTP сообщениях

- **Желаемое представление ресурса может содержаться в запросе клиента**
- **Текущее представление ресурса содержится в ответе сервера**
- **Сервер решает принять или отклонить запрос клиента**
- **Сообщения могут содержать дополнительные метаданные, описывающие представление (размер, формат и т.д.)**

Представление ресурсов

- **В теле HTTP ответа содержится представление запрашиваемого ресурса**
  - Содержит данные и метаданные
- **Если сообщения HTTP рассматривать как «вагоны», то представление ресурса - это «груз»**

```
HTTP/1.0 200 OK
Server: Apache/2.2.22
Date: Thu, 4 Apr 2013 00:01:05 GMT
Content-type: text/plain
Content-length: 18

Hi! I'm a message!
```

Заголовки представления

Тело представления

Представление ресурса

Заголовки представления

- **Метаданные, описывающие представление в HTTP сообщении**
- **Основные HTTP/1.1 заголовки представления:**
  - **`Content-type:`** тип представления
  - **`Content-length:`** размер сообщения
  - **`Content-encoding:`** преобразования, требующиеся для данного представления
  - **`ETag:`** хэш, отображающий «свежесть» ресурса

Представления при обмене клиент-сервер

- Клиент может запросить представления, которые он понимает
  - `“Accept: text/html, text/xml”`
- Ответ сервера указывает реальное представление, выдаваемое в теле ответа
  - При успешном ответе(запрашиваемое представление может быть предоставлено)
    - `“Content-type: text/xml”`
  - При ошибке (запрашиваемое представление невозможно)
    - `“406 Not Acceptable”`

Примеры представлений разного типа

```
Content-type: text/plain

Hi! I’m a message!
```

```
Content-type: text/html

<!DOCTYPE html>
<html>
<body>
<p>Hi! I’m a message!</p>
</body>
</html>
```

```
Content-type: text/xml

<?xml version=“1.0”?>
<note>
<body>Hi! I’m a message!</body>
</note>
```

Идентификация клиента

- **Иногда серверу надо знать кто к нему обращается**
  - Управление доступом к ресурсам
  - Персонализация в E-commerce
- **Механизмы**
  - HTTP заголовки
  - По IP адресу клиента
  - Пользовательский ввод (логин)
  - Cookies

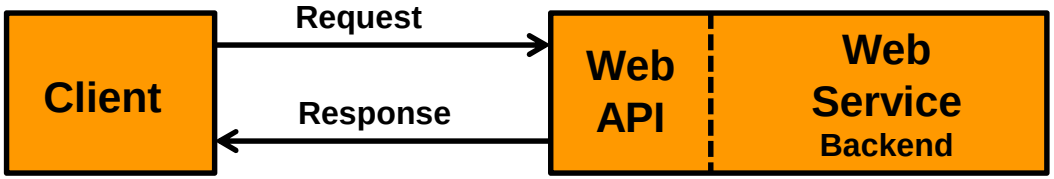
Web сервисы

Web - сервисы

- **Предоставляют доступ к «облачным» ресурсам серверов (SaaS)**
- **Предназначены преимущественно для взаимодействия «машина-машина»**
- **Используют HTTP протокол**
- **Доступ через публикуемый Web API**
  - Машинно–ориентированные представления

Web API

- **Клиенты используют API для работы с web сервисами**
  - “Web API” – это «лицо» сервиса



- **Web API для сервиса определяет:**
  - HTTP методы
  - адреса URL
  - метаданные (заголовки)
  - Формат представлений

RESTful API

- Web API, который соответствует архитектурному стилю REST называется “RESTful API”
- URL, используемые для web API указывают на ресурс, или коллекцию ресурсов
  - `http://example.com/resources`
- Используются только стандартизованные представления (JSON, XML, HTML etc)
- Используются только встроенные HTTP методы
  - POST, GET, PUT, DELETE

HTTP методы “RESTful” API

URI ресурса	GET	PUT	POST	DELETE
URI **коллекции**, Например: http://example.com /resources/	Выдает список элементов коллекции	Заменяет одну коллекцию другой целиком.	Создает новый элемент коллекции. Назначает новому элементу URI, который обычно возвращается в ответе сервера	**Delete** удаляет коллекцию целиком.
URI **элемента**, Например: http://example.com /resources/item17	Выдает детальное представление указанного элемента коллекции	Заменяет указанный элемент на новый	Обычно не используется. В некоторых случаях может создавать новый элемент внутри указанного элемента.	Удаляет указанный элемент.

Примеры стилей API

- **Non-RESTful**
  - Использует GET и для чтения, и для обновления
  - GET /Update.php?pot=1023&leds=00
- **RESTful**
  - Обновление: PUT /basement/entrydoor
  - Чтение: GET /basement/entrydoor

Машинно-ориентированные представления

- **XML :** e**X**tensible **M**arkup **L**anguage

```
<?xml version=“1.0”?>
<weatherdata>
  <location>
     <name>St. Petersburg</name>
   </location>
  <forecast>
     <symbol number="500" name="light rain" var="10d"/>
     <temperature day="19.04" min="15" max="19.04“ />
  </forecast>
</weatherdata>
```

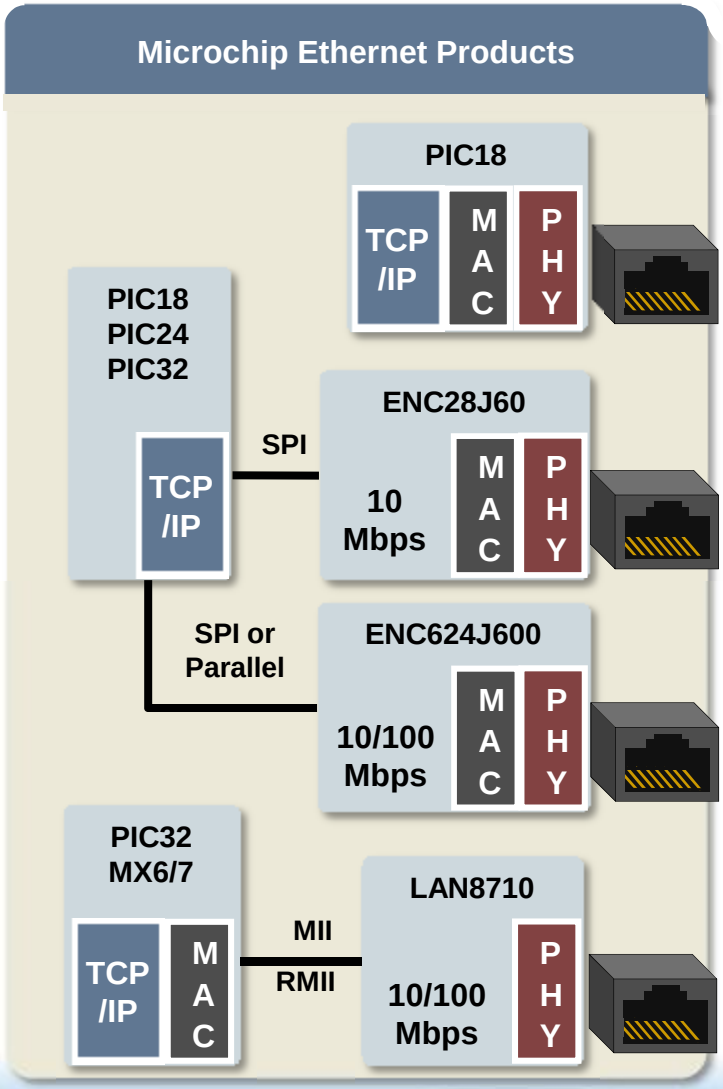
- **JSON**

```
{"city":{"id":1851632,"name":“St. Petersburg”},"weather":
[{"id":501,"main":"Rain","description":"moderate rain“
,"icon":"10d"}],"temp":{"day“:19.84,"min“:15,"max“:20}}
```

Поддержка от Microchip

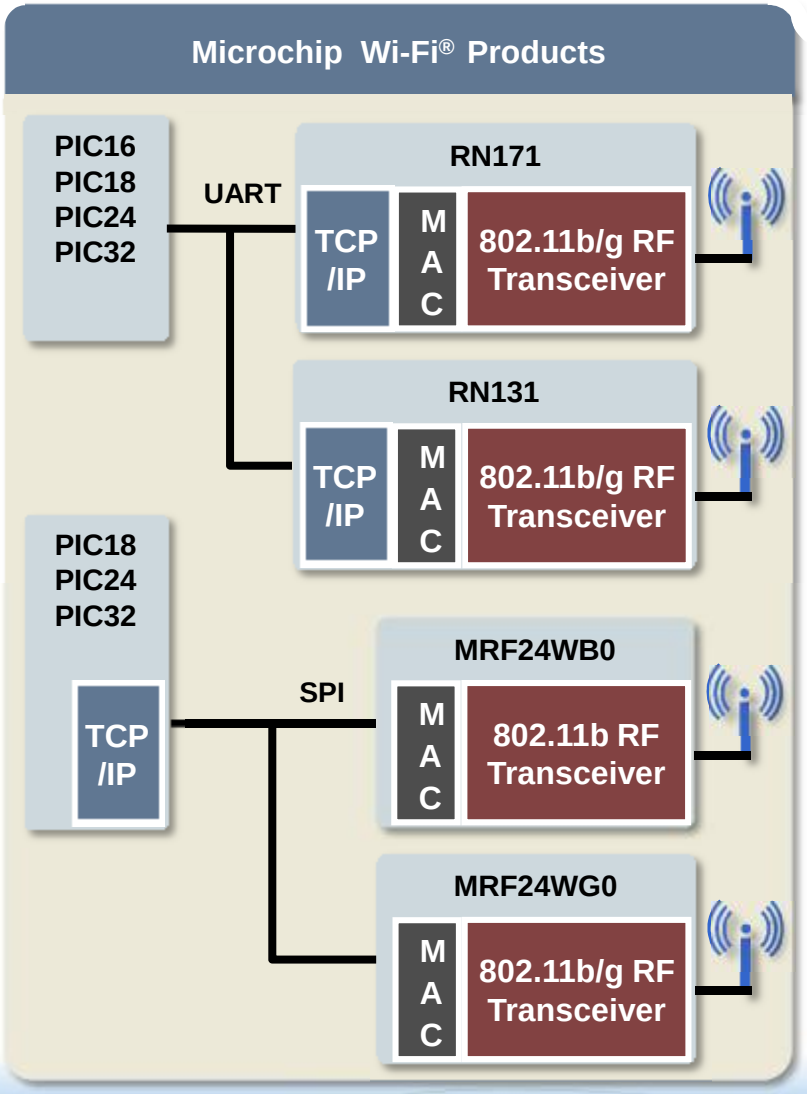
Что требуется для работы встраиваемых систем с web сервисами ?

Ethernet продукция Microchip



- **Микроконтроллеры**
  - **PIC18F97J60 (MAC & PHY)**
  - **PIC32 MX6/MX7 (MAC)**
- **Ethernet контроллеры**
  - **TCP/IP связь обеспечивается внешним PIC**
  - **ENC28J60 (10 Mbps)**
  - **ENC624J600 (10/100 Mbps)**
- **Ethernet трансиверы (SMSC)**
  - **LAN8710, LAN8720, LAN874x**

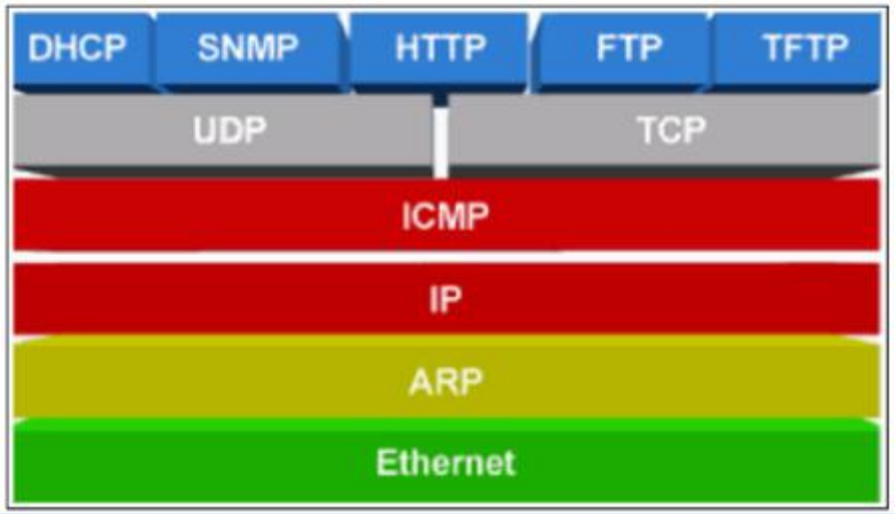
Wi-Fi® продукция Microchip



- **Автономные модули (Roving Networks)**
  - **Сами обеспечивают TCP/IP связь**
  - **RN171, RN131**
- **RF трансиверы**
  - **TCP/IP связь обеспечивается внешним PIC**
  - **802.11b:**
    - **MRF24WB0**
  - **802.11b/g**
    - **MRF24WG0**

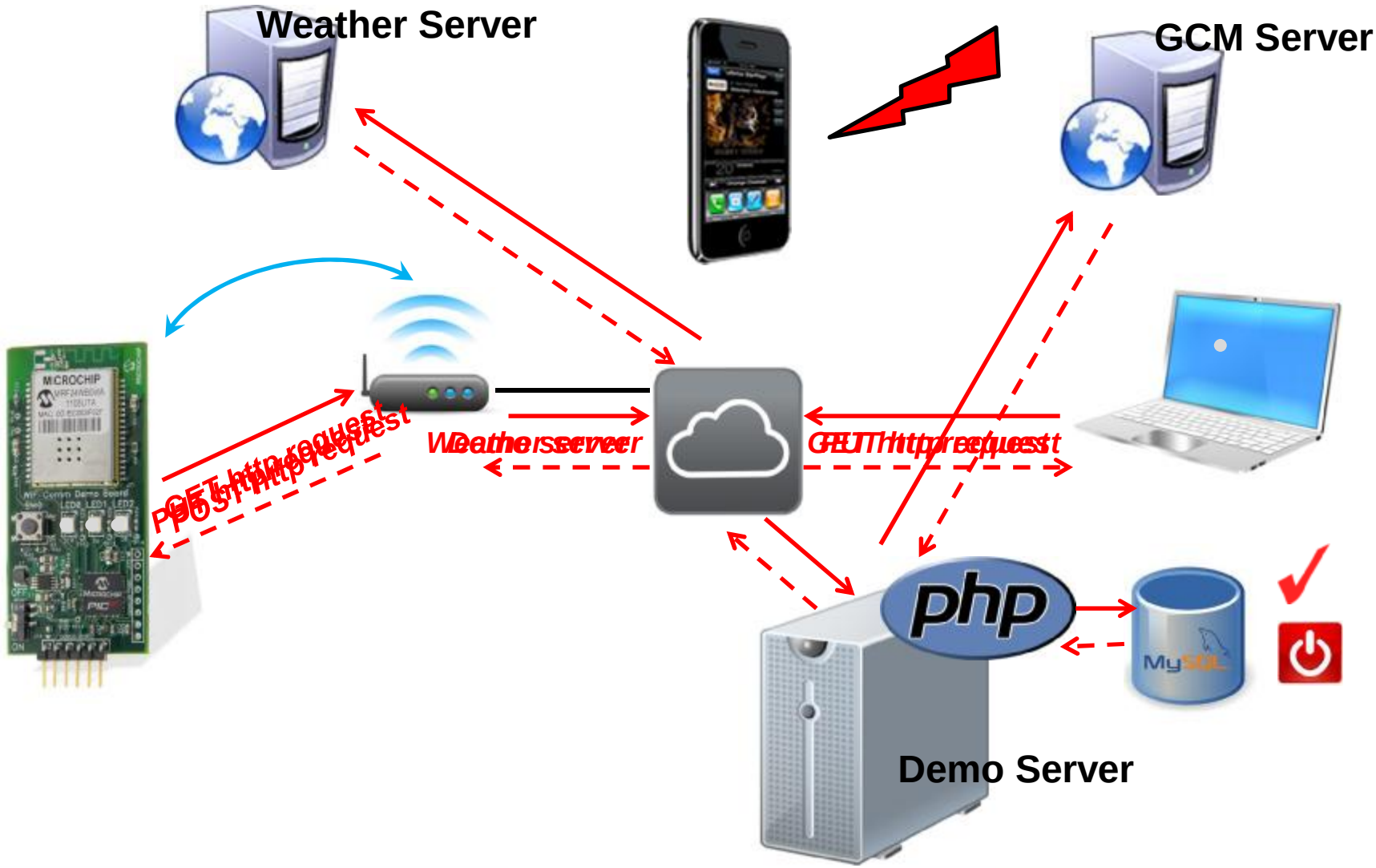
TCP/IP стек Microchip

- Бесплатная библиотека позволяющая вашему приложению работать через TCP/IP
- API для Berkeley (BSD) сокетов упрощает интеграцию вашего приложения и TCP/IP стека
- Включает готовые стандартные модули приложений
  - HTTP, SNMP, DHCP etc…
- Включает готовые файлы аппаратной конфигурации для отладочных плат Microchip
- Включает готовые TCP/IP демо-приложения



Демонстрация облачного мониторинга и управления

Взаимодействующие элементы



Демонстрационный сервер

Демонстрационный сервер

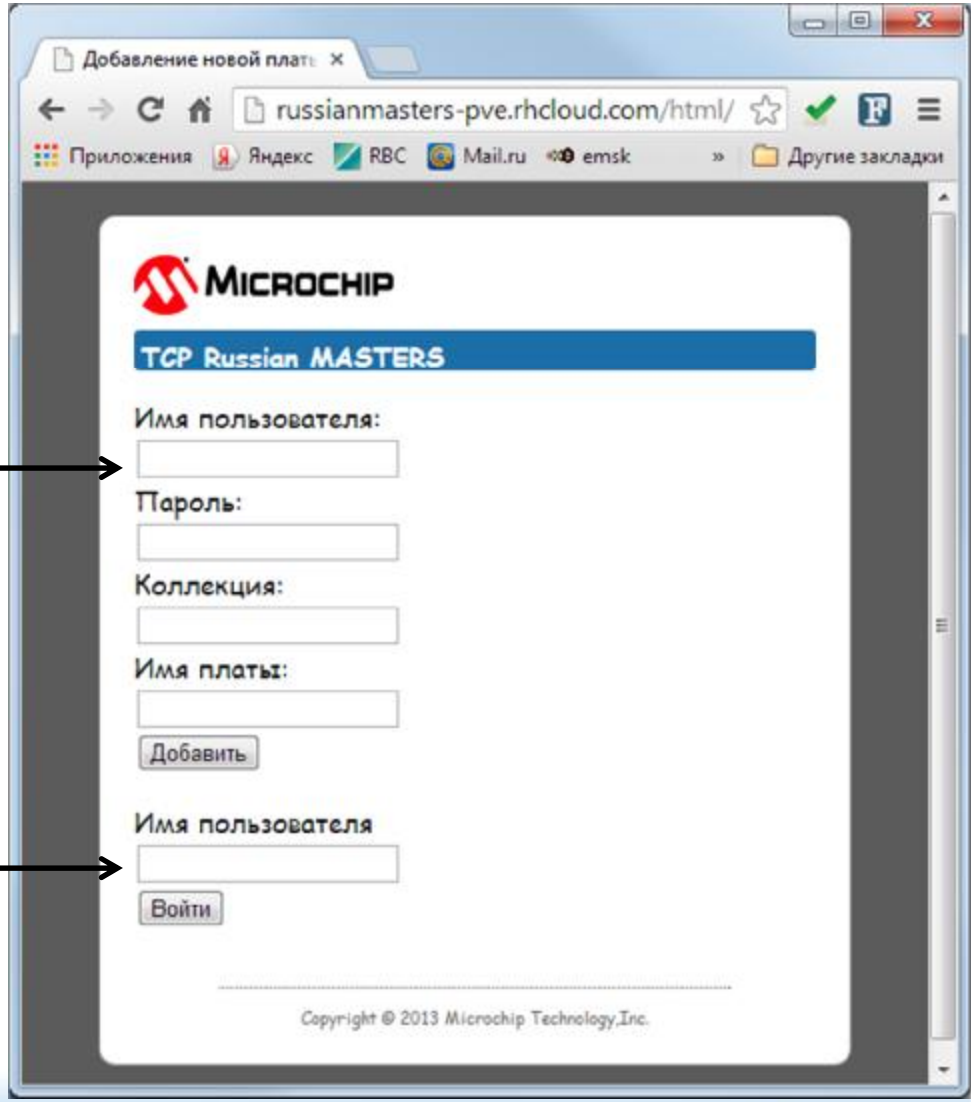
- Демонстрация облачного мониторинга и управления
- Хостинг: RedHat OpenShift
- URL: http://russianmasters-pve.rhcloud.com/
- Язык серверного приложения: PHP
- База данных: MySQL
- Документация по API: http://russianmasters-pve.rhcloud.com/html/api/index.htm

Что делает Web сервер

- **Сервер выполняет следующие задачи:**
  - Принимает установку соединения от клиента
  - Принимает HTTP запрос(ы)
  - Обрабатывает запрос – интерпретирует сообщение
  - Выполняет действие над указанным ресурсом
  - Формирует HTTP ответ
  - Посылает ответ клиенту

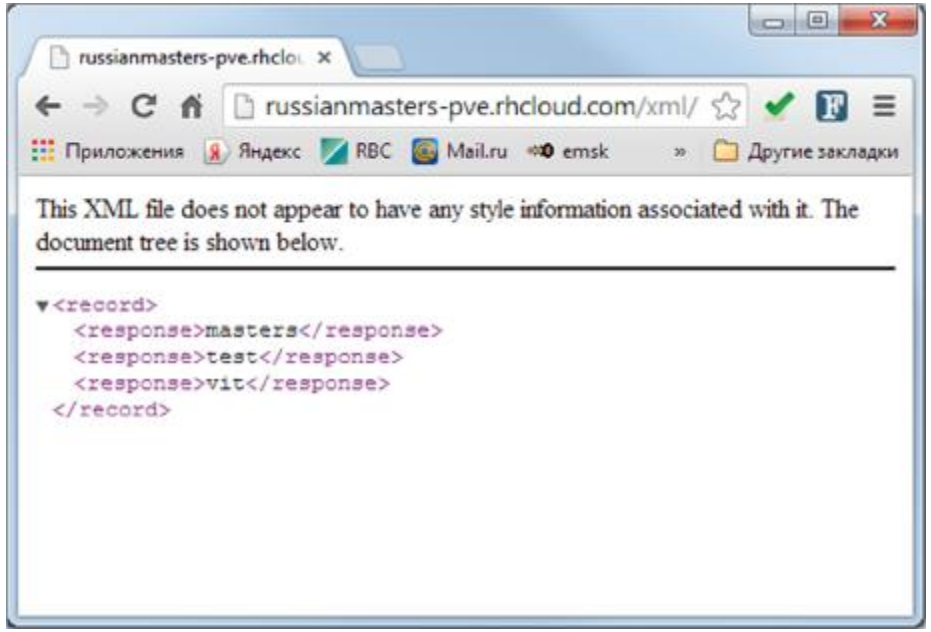
russianmasters-pve.rhcloud.com/html

- **Основной вход для браузера**
- **Используется для регистрации**
  - Имя пользователя
  - Пароль
  - Имя коллекции
  - Имя платы
- **Используется для входа в коллекцию**



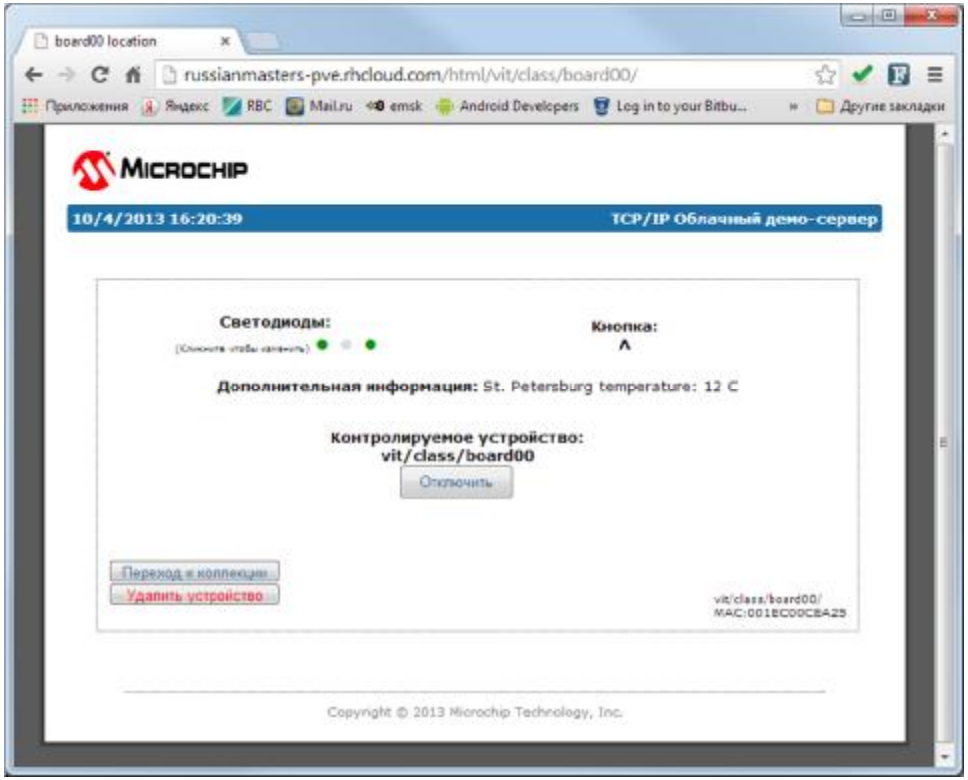
russianmasters-pve.rhcloud.com/xml

- Основной вход для встраиваемого клиента
- Защищен паролем пользователя
- GET возвращает XML список зарегистрированных пользователей



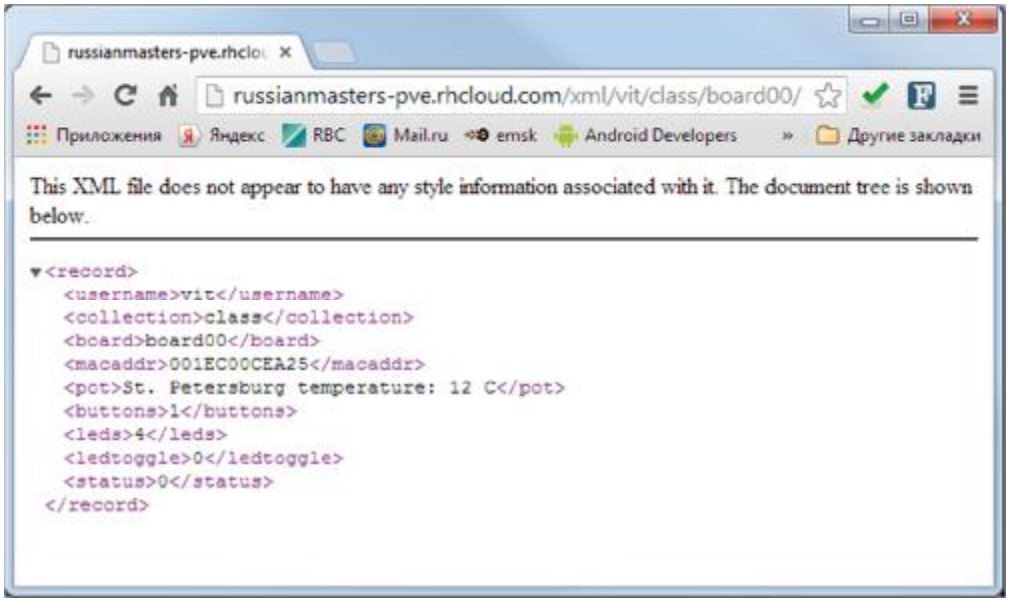
/html/username/collection/board

- URL «человеко-ориентированного» входа для доступа к плате
- Защищено паролем пользователя



/xml/username/collection/board

- URL «машинно-ориентированного» входа для доступа к плате
- Защищено паролем пользователя
- Воспринимает PUT запросы для обновления данных от платы
- Обрабатывает GET запросы для выдачи информации о командах управления



Демонстрационный сервер

http://russianmasters-pve.rhcloud.com

- **Ресурсы**
  - “Домашняя страница сайта”
  - “Список пользователей“, зарегистрированных на сервере
  - “Список коллекций“, заведенных на сервере (по каждому пользователю)
  - “Список плат в коллекции“
  - “Состояние платы“ : {username, collection, board, macAddr, pot, leds, buttons, ledtoggle, status}

Демонстрационный сервер

http://russianmasters-pve.rhcloud.com

URI ресурсов

“Домашняя страница“:	/
“Список пользователей”:	/xml
“Коллекции пользователя":	/<xml/html>/{username}
«Список плат в коллекции":	/<xml/html>/{username}/{collection}
«Состояние платы":	/<xml/html>/{username}/{collection}/{board}
«Документация API":	/html/api

Web API

- **Тип представления ресурсов:**
  - XML
- **Пример представления ресурса “Состояние платы”**

```
<?xml version="1.0"?>
<record>
<username>vit</username>
<collection>class</collection>
<board>board00</board>
<macaddr>00066681F503</macaddr>
<pot>Hello</pot>
<leds>05</leds>
<ledtoggle>0</ledtoggle>
<status>0</status>
</record>
```

Web API

- **HTTP методы API**

http://russianmasters-pve.rhcloud.com/html/api/index.htm

Resource Path	POST (create resource)	GET (read resource)	PUT (update resource)	DELETE (delete resource)
/xml/	1. Not implemented. Return error code 405	2. Return a list of all {username}	3. Not implemented. Return error code 405	4. Not implemented. Return error code 405
/xml/{username}/	5. Not implemented. Return error code 405	6. Return a list of all {collection} for a specific {username}	7. Not implemented. Return error code 405	8. PIC: not implemented. Return error code 405. Browser: delete all {collection} records in {username}
/xml/{username}/{collection}/	9. Not implemented. Return error code 405	10. Return a list of all {board} for a specific {collection} and {username}	11. Not implemented. Return error code 405	12. PIC: not implemented. Return error code 405. Browser: delete all {board} records in {collection}
/xml/{username}/{collection}/{board}/	13. Not implemented. Return error code 405	14. Return the complete record for a specific {board}	15. If {board} exists, update its record with supplied data	16. PIC: not implemented. Return error code 405. Browser: delete the specific {board} from the {collection}

Определение PIC® клиента

- **Заготовок HTTP запроса**
  - `User-Agent: MicrochipHTTPClient/0.1`
- **Позволяет серверу переправлять запросы от платы на ту версию API, которая была на момент производства платы.**

Реализация встраиваемого HTTP клиента

Требования к клиенту

- Простая машина состояний
- Авторизация
- Работа с различными методами web API различных web сервисов

Аппаратная часть

- **Wi-Fi G Demo Board**
  DV102412
- **Программатор PICkit 3**
  PG164130

Варианты TCP/IP стека Microchip

www.microchip.com/mla

Stack Version	8-bit (PIC18)	16-bit (PIC24/dsPIC® DSC)	32-bit (PIC32)
MLA TCPIP Stack v5	√	√	√
Beta-v6	-	-	√ **Используется в этом классе**
MPLAB® “Harmony” TCPIP Stack v6	-	-	√ (CQ4-13’)

Реализация

- **RESTful**
  - Полное управление HTTP сообщениями: методы, URL, заголовки
- **Управление TCP соединениями через сокеты**
  - 1-Non-persistent TCP сокет на транзакцию
  - Каждая транзакция должна содержать заголовок: “Connection: close”
- **Выполнение HTTP транзакций «по кругу»**
  - Можно добавлять задержки
- **Поддержка базовой авторизации HTTP для тех транзакций, которые этого требуют**

Контроль доступа

- Через HTTP заголовки

```
GET /xml/ HTTP/1.1
Host: russianmasters-pve.rhcloud
User-Agent: MicrochipHTTPClient/0.1
Authorization: Basic cm9vdDpyb290
Connection: close
```

Ограничения

- **Машина состояний не поддерживает перенаправления (304)**
- **HTTPs не поддерживается**
- **Нет обработки кодов ошибок HTTP**
  - Ошибки игнорируются
  - Транзакция повторяется
- **Max. 8 HTTP заголовков а HTTP сообщении**
  - Плюс автоматически добавляются заголовки Authorization: Basic xxx и Content-Length
- **Все сообщения должны укладываться в одну HTTP транзакцию**
  - Не поддерживается передача по частям (chunk)
  - Транзакция автоматически прерывается, если принято более 4096 байт

tcp6_demo_app_tasks()

- **Машина состояний клиента**
  - Инициализация
  - Выбрать очередную транзакцию из списка транзакций
  - Сформировать параметры HTTP запроса для транзакции
  - Подтвердить, что плата получила IP адрес
  - Определить IP адрес сервера (DNS)
  - Открыть TCP сокет
  - Выполнить HTTP транзакцию (клиент посылает "Connect: close" чтобы сервер закрыл соединение)
  - Закрыть TCP сокет
  - Обработать полученный HTTP ответ
  - Перейти к следующей транзакции…

Добавление HTTP транзакций

```
// Default HTTP transaction list executed by the client
// ref: TCP6 Web API v016
const HTTP_TRANSACTION_STATIC HTTP_TRANSACTION_LIST[] =
{
    {
        // TCP6 Web API
        // http://linux.mchpcloud.com/html/api
        "18",                                       // id
        "GET",                                      // method
        "http",                                     // scheme
        "linux.mchpcloud.com",                      // host
        "80",                                       // port
        "/xml/dennis/basement/windoweast/",         // path
        "HTTP/1.1",                                 // version
        {
           "Host: linux.mchpcloud.com",             // headers
           "User-Agent: MicrochipHTTPClient/0.1",
           "Authorization: Basic ZGVubmlzOmRlbm5pcw==",
           "Connection: close",
           "\0",
           "\0",
           "\0",
           "\0",
        },
        0,                                          // requires a body?
        0,                                          // dispatch delay (seconds)
    },
```

Каждая HTTP транзакция идентифицируется ее индексом в таблице

Позволяет задавать любые транзакции для работы с разными Web API

Добавление HTTP транзакций

- **Для каждой транзакции необходимо добавить еще один case в оператор `switch()`**

```
case SM_RESOURCE_UPDATE:
        // update resource data structure for the current http transaction
        switch(httpTransactionIndex)
        {
            case 0 ... 1:
                strcpy(myResourceState.collection, "basement");
                strcpy(myResourceState.board, "windoweast");
                // transition to next state (SM_HTTPC_TRANSACTION_CONFIG)
                smMyDemoAppTasks = SM_HTTPC_TRANSACTION_CONFIG;
                break;
            case 2:
               // transition to next state (SM_HTTPC_TRANSACTION_CONFIG)
                smMyDemoAppTasks = SM_HTTPC_TRANSACTION_CONFIG;
                break;
            default:
                break;
        }
```

Список демонстрационных транзакций

- **TCP6 Demo выполняет следующие транзакции:**
  - К демонстрационному серверу
    - **PUT на /XML/username/collection/board**
      - Обновляет на сервере состояния светодиодов, кнопки и дополнительной информации
    - **GET на /XML/username/collection/board**
      - Считывает команду управления светодиодами
    - **POST на /send.php (опционально)**
      - Запрашивает передачу уведомления на Android
  - К сервису погоды Yahoo!
    - **GET на weather.yahooapis.com**
      - Считывает текущую температуру в СПб каждые 30 секунд
      - Сохраняет значение температуры в качестве дополнительной информации

Лабораторная работа: Взаимодействие с сервером

MICROCHIP
MASTERs 2013

Спасибо !

Вопросы ?