МИРЭА
Российский
технологический
университет

Технологические основы Интернета Вещей

Лекция 3 - Языки разметки и протоколы межмашинного взаимодействия

Жматов Дмитрий Владимирович
кандидат технических наук, доцент
доцент кафедры Математического обеспечения и стандартизации информационных технологий

Особенности работы с данными:

- Большое количество структурированных данных
- Асинхронная передача данных
- Низкая пропускная способность «последней мили»
- В одной сети могут работать устройства с разной версией прошивки/производителя

Языки разметки данных:

Язык разметки (текста) в компьютерной терминологии — набор символов или последовательностей, вставляемых в текст для передачи информации о его выводе или строении.

Текстовый документ, написанный с использованием языка разметки, содержит не только сам текст (как последовательность слов и знаков препинания), но и дополнительную информацию о различных его участках — например, указание на заголовки, выделения, списки и т. д.

Основные элементы языка разметки данных:

Теги (tags) – специальные символы, позволяющие отличать в документе описание разметки от описания данных.
Элемент – это тэги в совокупности с их содержанием(данными).
Атрибут используется при определении элемента, чтобы задать какие-либо параметры, уточняющие характеристики данного элемента.

Виды разметки

- Структурная разметка задаёт структуру документа.
- Семантическая (контентная) разметка информирует о содержании данных

Язык SGML

SGML (англ. Standard Generalized Markup Language — стандартный обобщённый язык разметки) — метаязык, на котором можно определять язык разметки для документов.

Язык SGML

Основные части SGML-документа:

- **SGML-декларация** — определяет, какие символы и ограничители могут появляться в приложении;
- **Document Type Definition** — определяет синтаксис конструкций разметки, может включать дополнительные определения, такие, как символьные ссылки-мнемоники;
- **спецификация семантики, относится к разметке** — также даёт ограничения синтаксиса, которые не могут быть выражены внутри DTD;
- **содержимое SGML-документа** — по крайней мере, должен быть корневой элемент.

Язык HTML

HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине.

Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык HTML

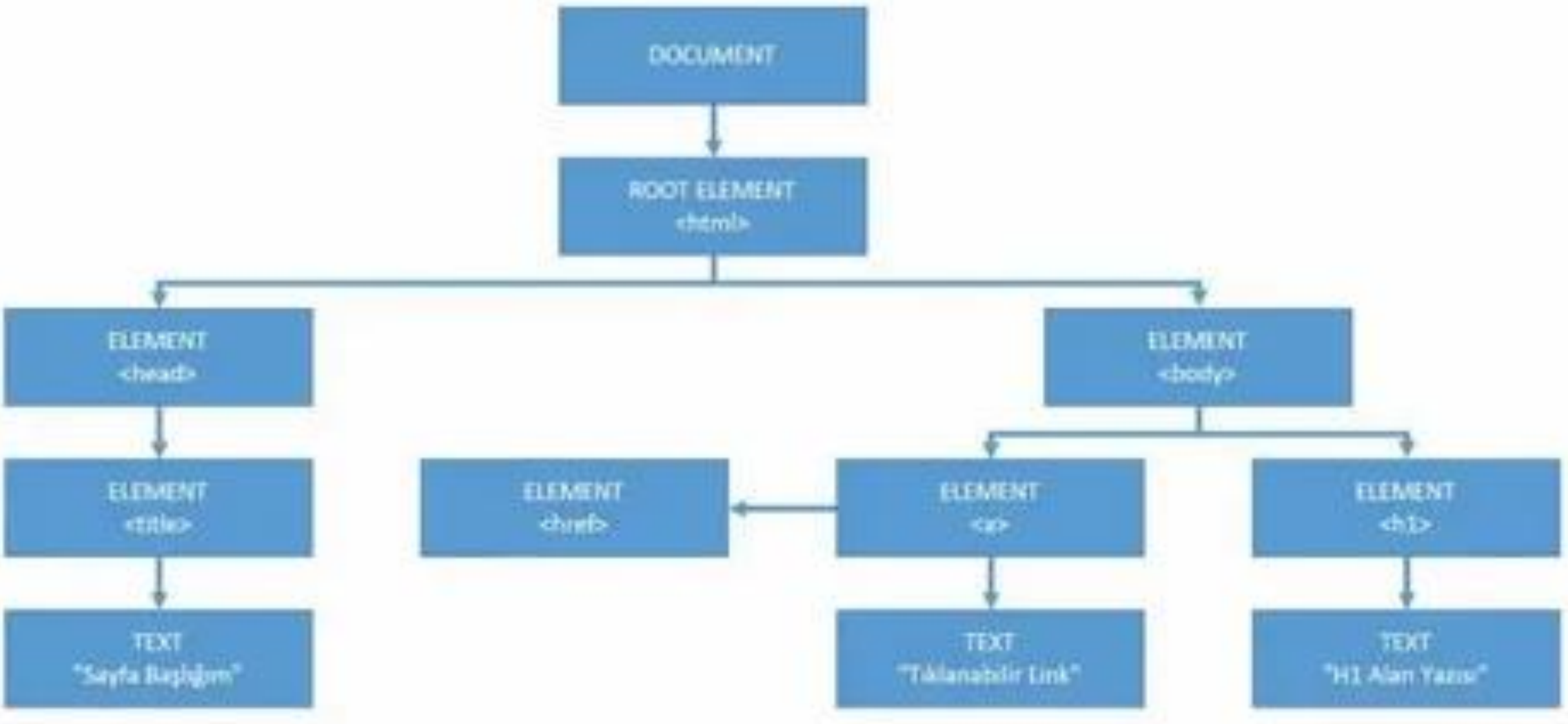
Язык HTML до 5-й версии определялся как приложение SGML (стандартного обобщённого языка разметки по стандарту ISO 8879).

Спецификации HTML5 формулируются в терминах DOM (объектной модели документа).

Язык XHTML является более строгим вариантом HTML, он следует синтаксису XML и является приложением языка XML в области разметки гипертекста.

Язык HTML

Спецификации HTML5 формулируются в терминах DOM (объектной модели документа).



Применение HTML в Интернете Вещей

HTML хорошо подходит в web технологиях, так как позволяет передавать большие объемы информации

Основной недостаток – высокая избыточность данного языка

Язык XML

XML (англ. eXtensible Markup Language) — расширяемый язык разметки. Рекомендован Консорциумом Всемирной паутины (W3C).

Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров

XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете.

Язык XML

Язык называется **расширяемым**, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка.

Расширение XML — это конкретная грамматика, созданная на базе XML и представленная словарём тегов и их атрибутов, а также набором правил, определяющих какие атрибуты и элементы могут входить в состав других элементов

Язык XML

XML документ должен содержать корневой элемент. Этот элемент является «родительским» для всех других элементов.

Все элементы в XML документе формируют иерархическое дерево. Это дерево начинается с корневого элемента и разветвляется на более низкие уровни элементов.

Все элементы могут иметь подэлементы (дочерние элементы):

```
<корневой>
    <потомок>
      <подпотомок>.....</подпотомок>
    </потомок>
</корневой>
```

Язык XML

```
<?xml  version="1.0"  encoding="windows-
1251"?>
<book category="WEB">
   <title lang="en">Learning XML</title>
   <author>Erik T. Ray</author>
   <year>2003</year>
   <price></price>
</book>
```

Первая строка — это XML декларация. Здесь определяется версия XML (1.0) и кодировка файла. На следующей строке описывается корневой элемент документа <book> (открывающий тег). Следующие 4 строки описывают дочерние элементы корневого элемента (title, author, year, price). Последняя строка определяет конец корневого элемента </book> (закрывающий тег).

Язык XML

Документ XML состоит из элементов (elements). Элемент начинается открывающим тегом (start-tag) в угловых скобках, затем идет содержимое (content) элемента, после него записывается закрывающий тег (end-teg) в угловых скобках.

Информация, заключенная между тегами, называется содержимым или значением элемента: <author>Erik T. Ray</author>. Т.е. элемент author принимает значение Erik T. Ray. Элементы могут вообще не принимать значения.

Язык XML

Элементы могут содержать атрибуты, так, например, открывающий тег <title lang="en"> имеет атрибут lang, который принимает значение en. Значения атрибутов заключаются в кавычки (двойные или ординарные).

Некоторые элементы, не содержащие значений, допустимо записывать без закрывающего тега. В таком случае символ / ставится в конце открывающего тега:
<name first="Иван" second="Петрович" />

Вложенность в языке XML

XML документ должен содержать корневой элемент. Этот элемент является «родительским» для всех других элементов.

Все элементы в XML документе формируют иерархическое дерево. Это дерево начинается с корневого элемента и разветвляется на более низкие уровни элементов.

Все элементы могут иметь подэлементы (дочерние элементы):

```
<корневой>
    <потомок>
      <подпотомок>.....</подпотомок>
    </потомок>
</корневой>
```

Структура языка XML

Структура XML документа должна соответствовать определенным правилам. XML документ отвечающий этим правилам называется валидным (англ. Valid — правильный) или синтаксически верным. Соответственно, если документ не отвечает правилам, он является невалидным .

Основные правила синтаксиса XML:

1. Теги XML регистрозависимы — теги XML являются регистрозависимыми. Так, тег <Letter> не то же самое, что тег <letter>.

Открывающий и закрывающий теги должны определяться в одном регистре:

<Message>Это неправильно</message>

<message>Это правильно</message>

2. XML элементы должны соблюдать корректную вложенность:

Некорректная вложенность

Корректная вложенность

Структура языка XML

Структура XML документа должна соответствовать определенным правилам. XML документ отвечающий этим правилам называется валидным (англ. Valid — правильный) или синтаксически верным. Соответственно, если документ не отвечает правилам, он является невалидным .

Основные правила синтаксиса XML:

3. У XML документа должен быть корневой элемент — XML документ должен содержать один элемент, который будет родительским для всех других элементов. Он называется корневым элементом.

В большинстве XML файлов корневым элементом является <Файл></Файл>. После закрывающего тега </Файл> больше ничего быть не должно.

4. Значения XML атрибутов должны заключаться в кавычки:

<note date="12/11/2007">Корректная запись</note>

<note date=12/11/2007>Некорреткная запись</note>

Сущности языка XML

Некоторые символы в XML имеют особые значения и являются служебными. Если вы поместите, например, символ < внутри XML элемента, то будет сгенерирована ошибка, так как парсер интерпретирует его, как начало нового элемента.

Сущность	Символ	Значение
&lt;	<	меньше, чем
&gt;	>	больше, чем
&amp;	&	амперсанд
&apos;	'	апостроф
&quot;	"	кавычки

Парсеры XML

Рассмотрим основные методы работы с XML на примере объемного XML файла (более 100Mb):

1. Simple XML
2. DOM
3. xml_parser (SAX)
4. XMLReader

Парсеры XML

Simple XML

Минусы: работает очень медленно, собирает весь файл в память, дерево составляется в отдельных массив.

Плюсы: простота работы, работа «из коробки» (требует библиотеки libxml которая включена практически на всех серверах)

DOM

Минусы: работает очень медленно, как и все предыдущие примеры собирает весь файл в память.

Плюсы: На выходе привычный DOM с которым очень легко работать.

Парсеры XML

xml_parser и XMLReader.
Данные библиотеки работают чтением файла построчно, что актуально для больших документов

Метод	Время выполнения (19 Mb)	Время выполнения (190 Mb)
Simple XML	0.46 сек	4.56 сек
DOM	0.52 сек	4.09 сек
xml_parse	0.22 сек	2.25 сек
XML Reader	0.26 сек	2.18 сек

Язык JSON

JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

За счёт своей лаконичности по сравнению с XML формат JSON может быть более подходящим для сериализации сложных структур. Применяется в веб-приложениях как для обмена данными между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения).

Язык JSON

Поскольку формат JSON является подмножеством синтаксиса языка JavaScript, то он может быть быстро десериализован встроенной функцией eval().
JSON-текст представляет собой (в закодированном виде) одну из двух структур:

Набор пар ключ: значение. В различных языках это реализовано как запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка (регистрозависимая: имена с буквами в разных регистрах считаются разными), значением — любая форма.
Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

Язык JSON

В качестве значений в JSON могут быть использованы:

запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.

массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, т.е. не содержать ни одного значения.

Язык JSON

В качестве значений в JSON могут быть использованы:
число (целое или вещественное).
литералы true, false и null.
строка — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки.
Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты \', \", \\, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.

Сравнение языков разметки

```
{
   "firstName": "Иван",
   "lastName": "Иванов",
   "address": {
      "streetAddress": "Московское ш., 101, кв.101",
      "city": "Ленинград",
      "postalCode": 101101
   },
   "phoneNumbers": [
      "812 123-1234",
      "916 123-4567"
   ]
}
```

```
<person>
  <firstName>Иван</firstName>
  <lastName>Иванов</lastName>
  <address>
    <streetAddress>Московское ш., 101, кв.101</streetAddress>
    <city>Ленинград</city>
    <postalCode>101101</postalCode>
  </address>
  <phoneNumbers>
    <phoneNumber>812 123-1234</phoneNumber>
    <phoneNumber>916 123-4567</phoneNumber>
  </phoneNumbers>
</person>
```

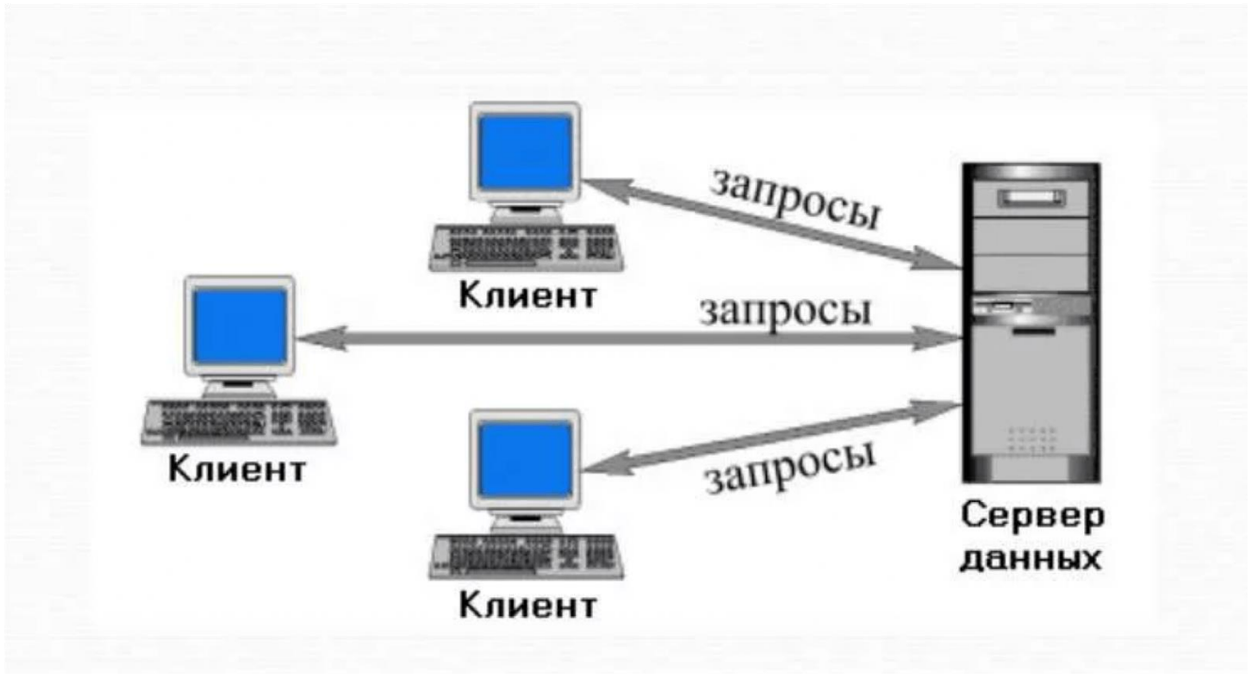
Вывод: JSON лучше подходит для разметки данных при их передаче в системах Интернета Вещей

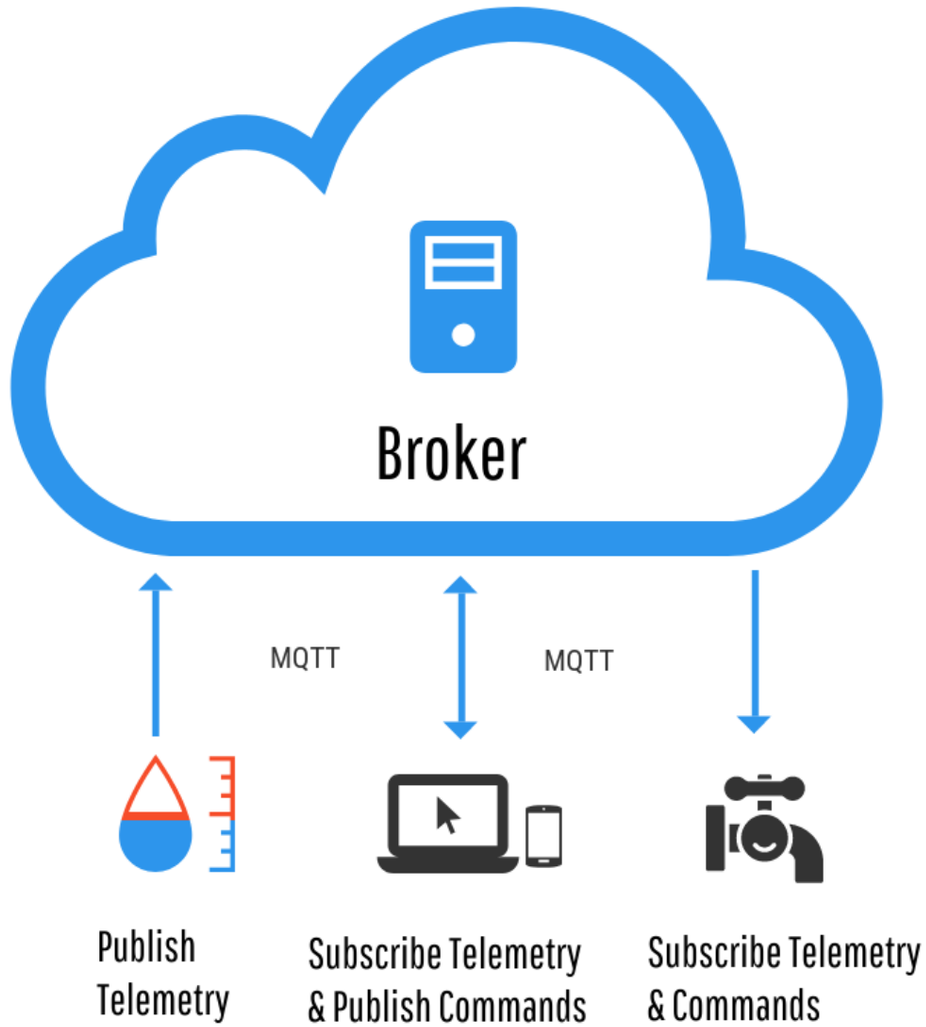
Протокол HTTP

HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных изначально — в виде гипертекстовых документов в формате «HTML», в настоящий момент используется для передачи произвольных данных.

Основой HTTP является технология «клиент-сервер», то есть предполагается существование:
Потребителей (клиентов), которые инициируют соединение и посылают запрос;
Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

Протокол HTTP





Устройств – миллионы, и не все из них онлайн
100% времени!

Основное требование к протоколам передачи данных

Горизонтальное масштабирование — разбиение системы на более мелкие структурные компоненты и разнесение их по отдельным физическим машинам (или их группам), и (или) увеличение количества серверов, параллельно выполняющих одну и ту же функцию. Масштабируемость в этом контексте означает возможность добавлять к системе новые узлы, серверы, процессоры для увеличения общей производительности. Этот способ масштабирования может требовать внесения изменений в программы, чтобы программы могли в полной мере пользоваться возросшим количеством ресурсов

Протоколы межмашинного взаимодействия

DDS (Data Distribution Service) – реализует шаблон публикации-подписки для отправки и приема данных, событий и команд среди конечных узлов. Узлы-издатели создают информацию, «topic» (темы, разделы: температура, местоположение, давление) и публикуют шаблоны.

Узлам, заинтересовавшимся в данных разделах, DDS прозрачно доставляет созданные шаблоны. В качестве транспорта – UDP. Также DDS позволяет управлять параметрами QoS (качество облуживания).

Протоколы межмашинного взаимодействия

CoAP (Constrained Application Protocol) – с точки зрения пользователя похож на протокол HTTP, но отличается малым размером заголовков, что подходит для сетей с ограниченными возможностями.

Использует архитектуру клиент-сервер и подходит для передачи информации о состоянии узла на сервер (сообщения GET, PUT, HEAD, POST, DELETE, CONNECT). В качестве транспорта – UDP.

Протоколы межмашинного взаимодействия

XMPP (Extensible Messaging and Presence Protocol) – давно используется в сети Интернет для передачи сообщений в режиме реального времени, благодаря формату XML подходит для использования в сетях IoT.

Работает поверх архитектур издатель-подписчик и клиент-сервер. Также используется для адресации устройств в небольших сетях (адресация вида «name@domain.com»).

Протокол MQTT

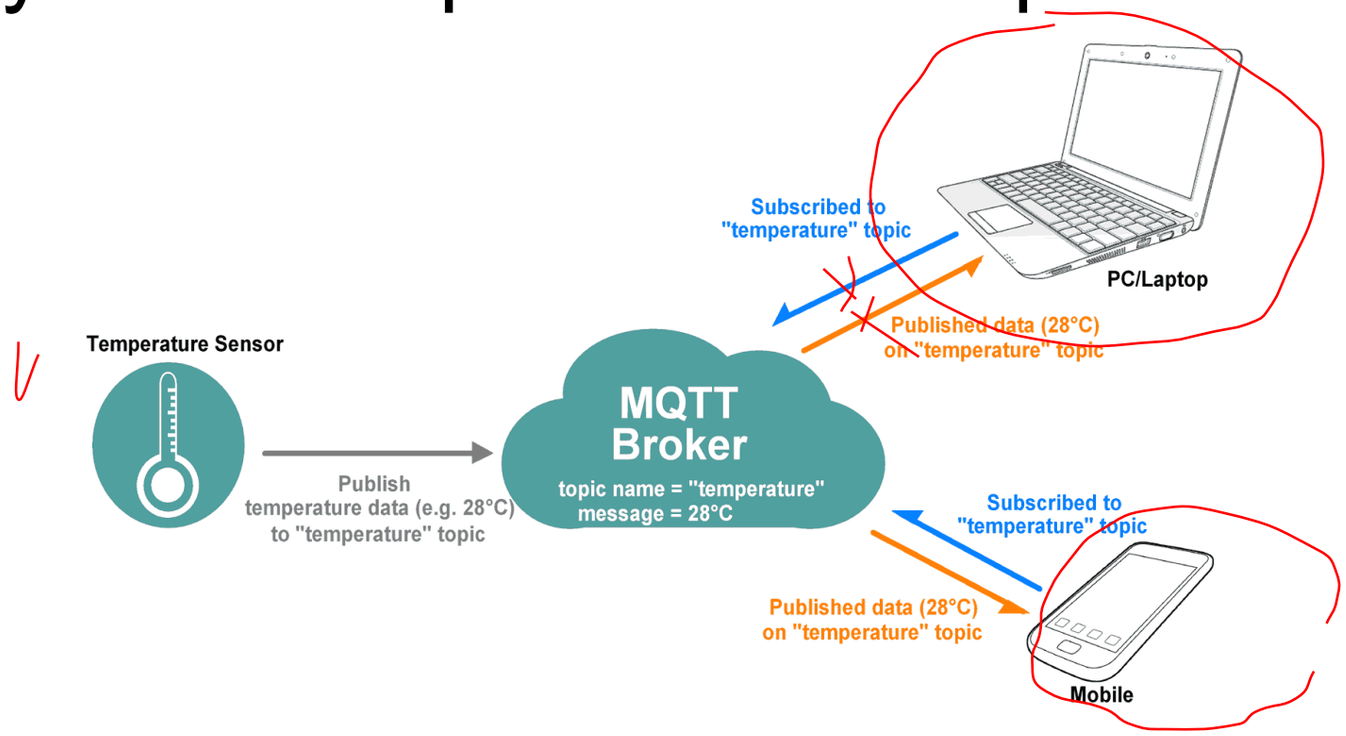
MQTT (Message Queue Telemetry Transport) – упрощённый сетевой протокол, работающий поверх TCP/IP, осуществляет сбор данных от множества узлов и передачу на сервер.

Основывается на модели издатель-подписчик с использованием промежуточного сервера – брокера (приоритезация сообщений, формирование очередей и др.). В качестве транспорта – TCP. На основе MQTT был сформирован специализированный протокол MQTT-SN для сенсорных сетей.

Протокол MQTT

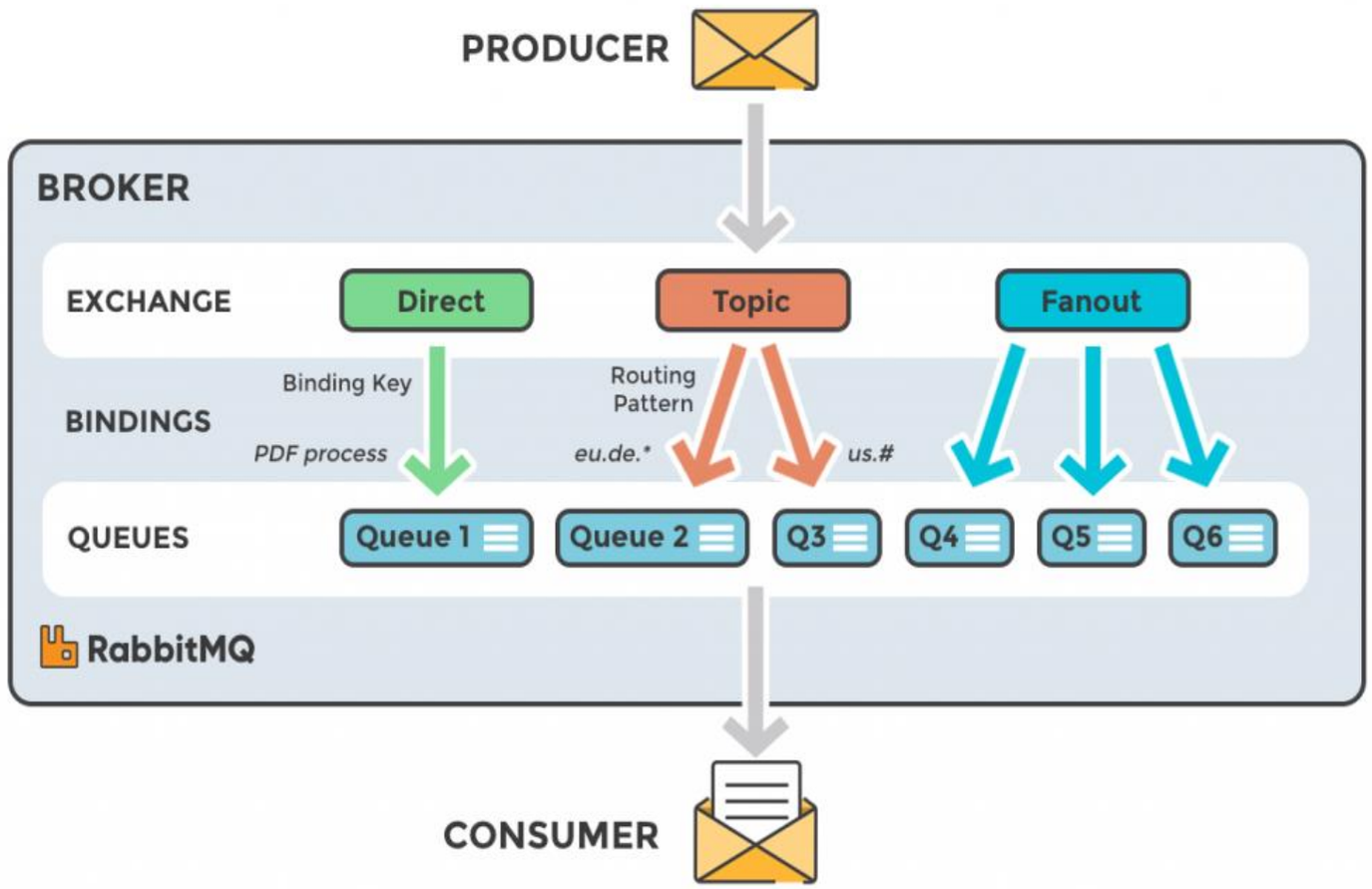
Система связи, построенная на MQTT, состоит из сервера-издателя, сервера-брокера и одного или нескольких клиентов.

Издатель не требует каких-либо настроек по количеству или расположению подписчиков, получающих сообщения. Кроме того, подписчикам не требуется настройка на конкретного издателя



Протоколы межмашинного взаимодействия

Любые данные, опубликованные или полученные брокером MQTT, будут закодированы в двоичном формате, поскольку MQTT является бинарным протоколом. Это означает, что для получения исходного содержимого нужно интерпретировать сообщение



Протокол MQTT

Преимущества MQTT:

- Нейтрален к содержимому сообщения
- Идеально подходит для распределённых коммуникаций «один ко многим» и разъединённых приложений
- Оснащён функцией LWT (Last Will and Testament, «последняя воля и завещание») для уведомления сторон об аномальном отключении клиента
- Полагается на TCP/IP для базовых задач связи
- Разработан для доставки сообщений по шаблонам «максимум один раз», «минимум один раз» и «ровно один раз»

Протокол MQTT

RabbitMQ — программный брокер сообщений на основе стандарта AMQP — тиражируемое связующее программное обеспечение, ориентированное на обработку сообщений. Создан на основе системы Open Telecom Platform, написан на языке Erlang, в качестве движка базы данных для хранения сообщений использует Mnesia.

Apache Kafka — распределённый программный брокер сообщений, проект с открытым исходным кодом, разрабатываемый в рамках фонда Apache. Написан на языке программирования Scala и Java.

Протокол AMQP

AMQP (Advanced Message Queuing Protocol) — открытый протокол для передачи сообщений между компонентами системы. Основная идея состоит в том, что отдельные подсистемы (или независимые приложения) могут обмениваться произвольным образом сообщениями через AMQP-брокер, который осуществляет маршрутизацию, возможно гарантирует доставку, распределение потоков данных, подписку на нужные типы сообщений.

Протокол AMQP

AMQP основан на трёх понятиях:
Сообщение (message) — единица передаваемых данных, основная его часть (содержание) никак не интерпретируется сервером, к сообщению могут быть присоединены структурированные заголовки.

Очередь (queue) — здесь хранятся сообщения до тех пор, пока не будут забраны клиентом. Клиент всегда забирает сообщения из одной или нескольких очередей.

Протокол AMQP

Точка обмена (exchange) — в неё отправляются сообщения. Точка обмена распределяет сообщения в одну или несколько очередей. При этом в точке обмена сообщения не хранятся. Точки обмена бывают трёх типов:

fanout — сообщение передаётся во все прицепленные к ней очереди;

direct — сообщение передаётся в очередь с именем, совпадающим с ключом маршрутизации (routing key) (ключ маршрутизации указывается при отправке сообщения);

topic — нечто среднее между fanout и direct, сообщение передаётся в очереди, для которых совпадает маска на ключ маршрутизации, например, app.notification.sms.# — в очередь будут доставлены все сообщения, отправленные с ключами, начинающимися с app.notification.sms.

Протокол AMQP