物联网通信技术

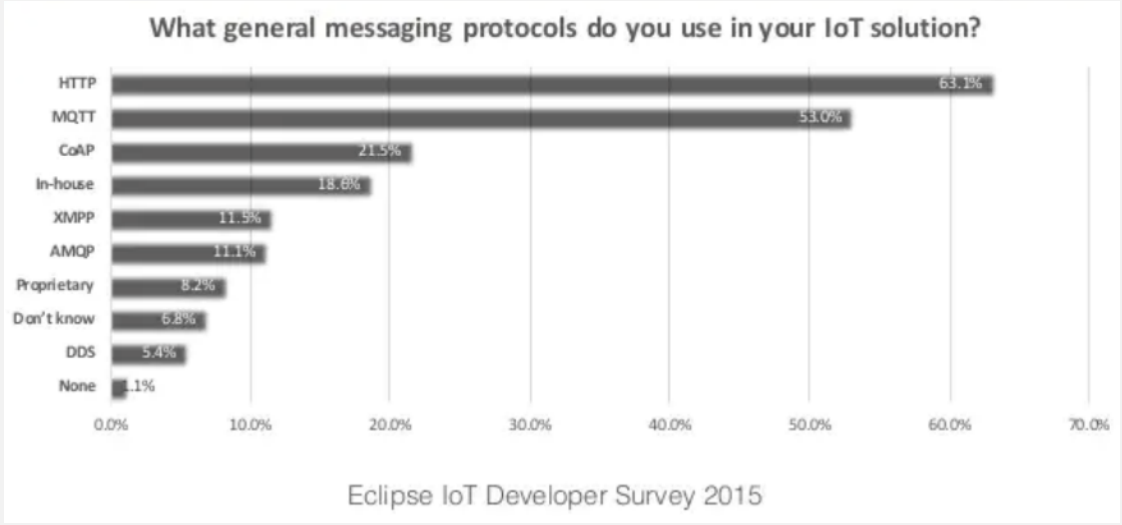
主讲人：宁 磊
Email：ninglei@sztu.edu.cn

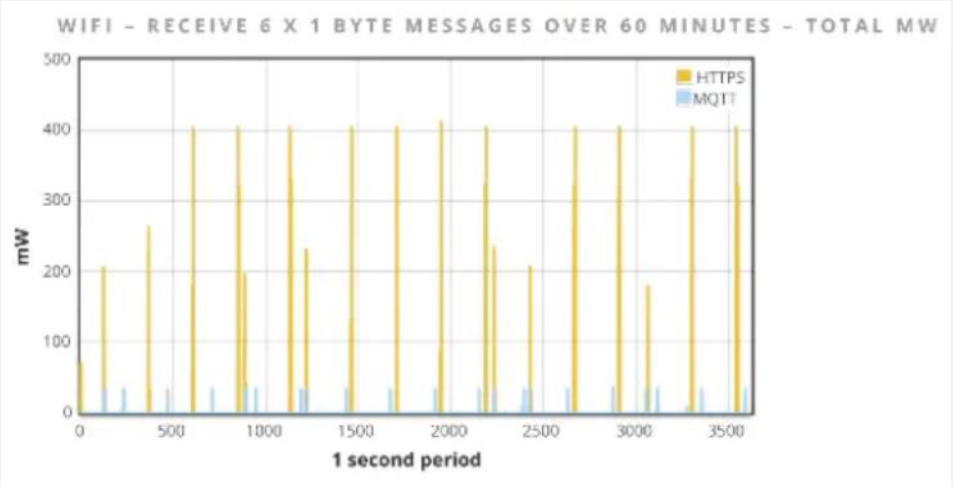
目 录
CONTENTS

- 本章主要内容：MQTT、CoAP、LwM2M、Matter、UART和Modbus传输协议的主要特点，协议概述和应用方法。
- 本章学习目标
  - 了解物联网应用层传输协议的主要特点；
  - 熟悉物联网应用层传输协议的基本原理和使用方法；
  - 掌握物联网应用层传输协议的适用场景。

为什么Http协议还不够?

- HTTP特点
  - 发送Request，接收Response
  - 无连接
  - 媒体独立
  - 无状态
- 物联网设备数据传输需求
  - 一对多分发消息
  - 状态变化需要上报、通知，部分场景实时性要求高
  - 海量节点、小包、部分场景频发
  - 接入与传输网络可靠性一般
  - 供电受限





MQTT 诞生

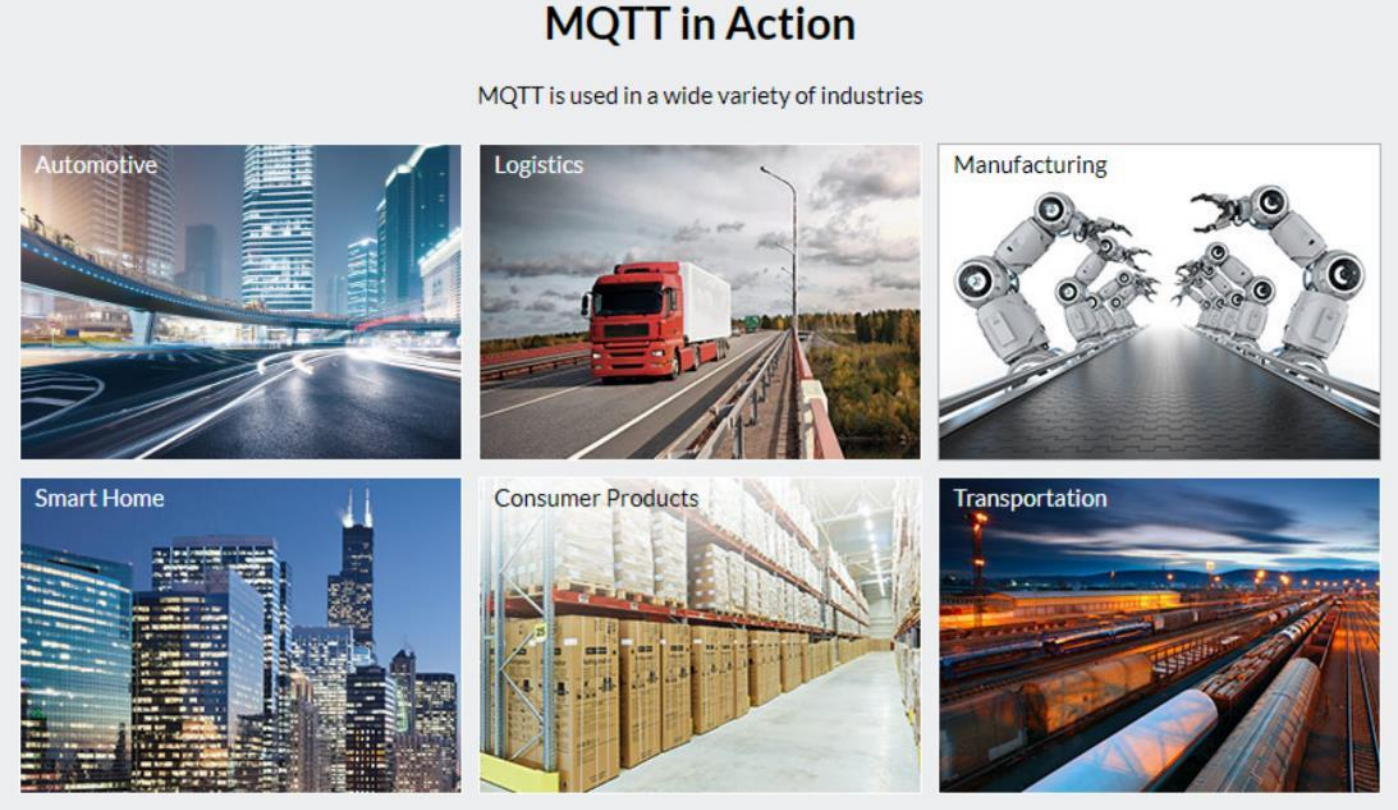
据 Arlen Nipper 在 IBM Podcast 上的自述，MQTT 原名是 MQ TT， 注意 MQ 与 TT之间的空格，其全称为: MQ Telemetry Transport，是九十年代早期，他在参与 Conoco Phillips 公司的一个原油管道数据采集监控系统(pipeline SCADA system)时，开发的一个实时数据传输协议。它的目的在于让传感器通过带宽有限的VSAT(Very Small Aperture Terminal)，与 IBM 的 MQ Integrator 通信。由于 Nipper 是遥感和数据采集监控专业出身，所以按业内惯例给了个 MQ TT 的名字。

MQTT（Message Queuing Telemetry Transport，消息队列遥测传输协议）

MQTT 应用

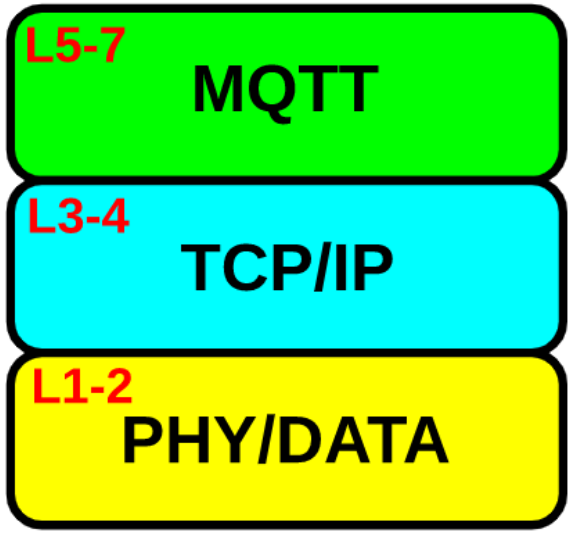
MQTT协议广泛应用于物联网、移动互联网、智能硬件、车联网、电力能源等领域：

- 物联网M2M通信，物联网大数据采集
- Android消息推送，WEB消息推送
- 移动即时消息
- 智能硬件、智能家具、智能电器
- 车联网通信，电动车站桩采集
- 智慧城市、远程医疗、远程教育
- 电力、石油与能源等行业市场



MQTT 特点

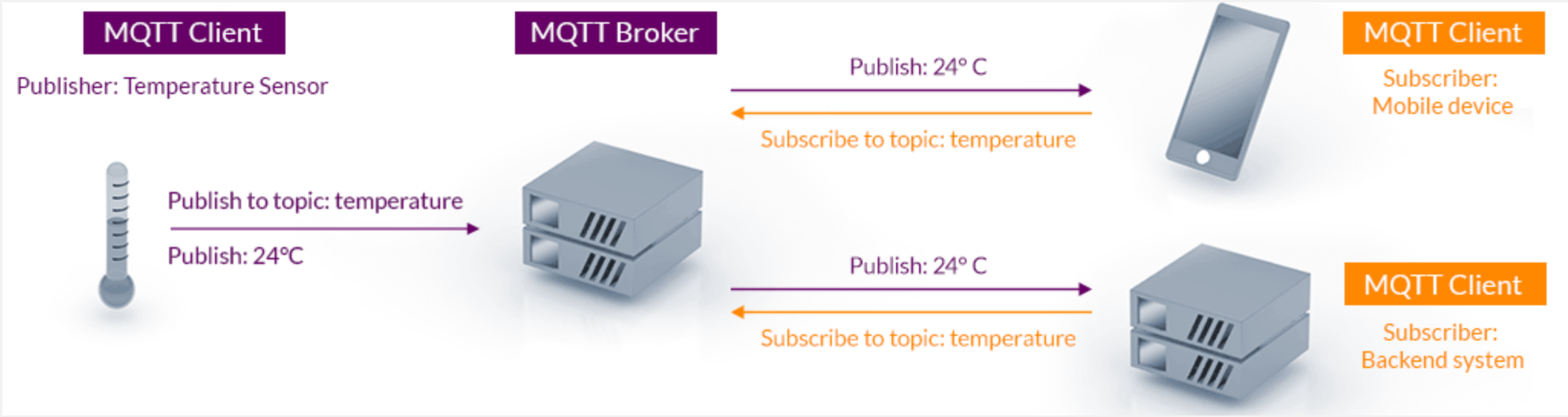
- 开放消息协议，简单易实现
- 发布订阅模式，一对多消息发布
- 基于TCP/IP网络连接,提供有序，无损，双向连接。
- 最小化传输开销和协议交换，有效减少网络流量。
- 消息QoS支持，可靠传输保证



MQTT发布订阅模式

- 发布订阅模式是传统 Client/Server 模式的一种解耦方案，发布者通过 Broker 与订阅者之间通信
- Broker 的作用是将收到的消息通过某种过滤规则，正确地发送给订阅者
- 发布者和订阅者之间不必预先知道对方的存在，比如不需要预先沟通对方的 IP Address 和 Port
- 发布者和订阅者之间不必同时运行，因为 Broker 是一直运行的

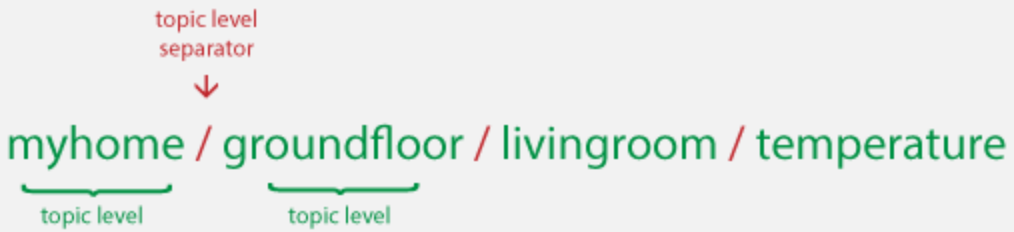
在 MQTT 协议里，上面提到的过滤规则是主题Topic。比如：所有发布到Temperature这个 Topic的消息，都会被Broker转发给已经订阅了Temperature的订阅者。



MQTT主题Topic

主题本质上是一个字符串，MQTT 协议规定主题是 UTF-8 编码的字符串，这意味着，主题过滤器和主题名的比较可以通过比较编码后的 UTF-8 字节或解码后的 Unicode 字符。

- **主题名:**附加在应用消息上的一个标签，服务端已知且与订阅匹配。服务端发送应用消息的一个副本给每一个匹配的客户端订阅；至少包括1个字符，区分大小写



MQTT主题Topic

主题过滤器:订阅中包含的一个表达式，用于表示相关的一个或多个主题。主题过滤器可以使用通配符

- **单层通配符**

加号（“+”）是只能用于单个主题层级匹配的通配符。例如，sport/tennis/+ 匹配 sport/tennis/player1 和 sport/tennis/player2，但是不匹配 sport/tennis/player1/ranking。同时，由于单层通配符只能匹配一个层级，sport/+ 不匹配 sport 但是却匹配 sport/。

MQTT主题Topic

主题过滤器:订阅中包含的一个表达式，用于表示相关的一个或多个主题。主题过滤器可以使用通配符

- **多层通配符**

井字符号（“#”）是用于匹配主题中任意层级的通配符。多层通配符表示它的父级和任意数量的子层级。

例如，如果客户端订阅主题 sport/tennis/player1/#，它会收到使用下列主题名发布的消息：

sport/tennis/player1

sport/tennis/player1/ranking

sport/tennis/player1/score/Wimbledon

消息格式

MQTT 的消息格式分三部分：

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文的类型				标志 Flags			
byte 2	剩余长度							

- **固定长度头部**，2 个字节，所有消息类型里都有
  - MQTT控制报文的类型，第1个字节，二进制位7-4。
  - 标志 Flags，第1个字节，二进制位3-0。
    - DUP：发布消息的副本。用来在保证消息的可靠传输，如果设置为1，则在下面的变长中增加MessageId，并且需要回复确认，以保证消息传输完成，但不能用于检测消息重复发送。
    - QoS：发布消息的服务质量，共有 3 个 QoS 等级
    - RETAIN： 发布保留标识，表示服务器要保留这次推送的信息，如果有新的订阅者出现，就把这消息推送给它，如果为False那么推送至当前订阅者后释放。
  - 剩余长度（Remaining Length），第2个字节，表示当前报文剩余部分的字节数，包括可变报头和负载的数据。剩余长度不包括用于编码剩余长度字段本身的字节数。
- **可变长度头部**，它驻位于固定的头和负载之间。可变头的内容因数据包类型而不同，较常的应用是作为包的标识：很多类型数据包中都包括一个2字节的数据包标识字段。
- **有效载荷 Payload**，只有某些消息类型里有，对于发布者来说有效载荷就是应用消息。

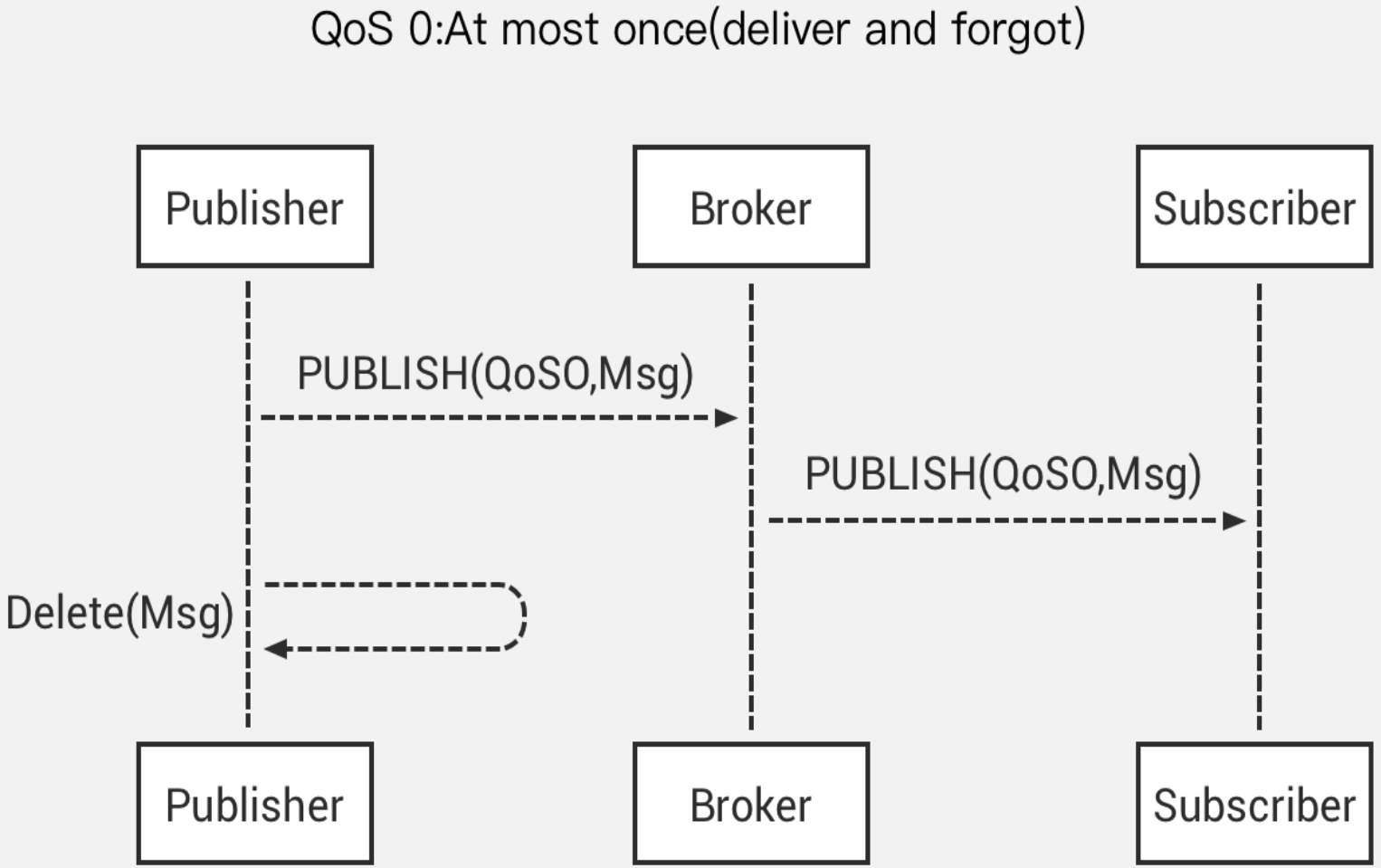
控制报文类型

名称	值	流方向	描述
Reserved	0	不可用	保留位
CONNECT	1	客户端到服务器	客户端请求连接到服务器
CONNACK	2	服务器到客户端	连接确认
PUBLISH	3	双向	发布消息
PUBACK	4	双向	发布确认
PUBREC	5	双向	发布收到（保证第1部分到达）
PUBREL	6	双向	发布释放（保证第2部分到达）
PUBCOMP	7	双向	发布完成（保证第3部分到达）
SUBSCRIBE	8	客户端到服务器	客户端请求订阅
SUBACK	9	服务器到客户端	订阅确认
UNSUBSCRIBE	10	客户端到服务器	请求取消订阅
UNSUBACK	11	服务器到客户端	取消订阅确认
PINGREQ	12	客户端到服务器	PING请求
PINGRESP	13	服务器到客户端	PING应答
DISCONNECT	14	客户端到服务器	中断连接
Reserved	15	不可用	保留位

QoS（服务质量）介绍

QoS 0 - 最多分发一次

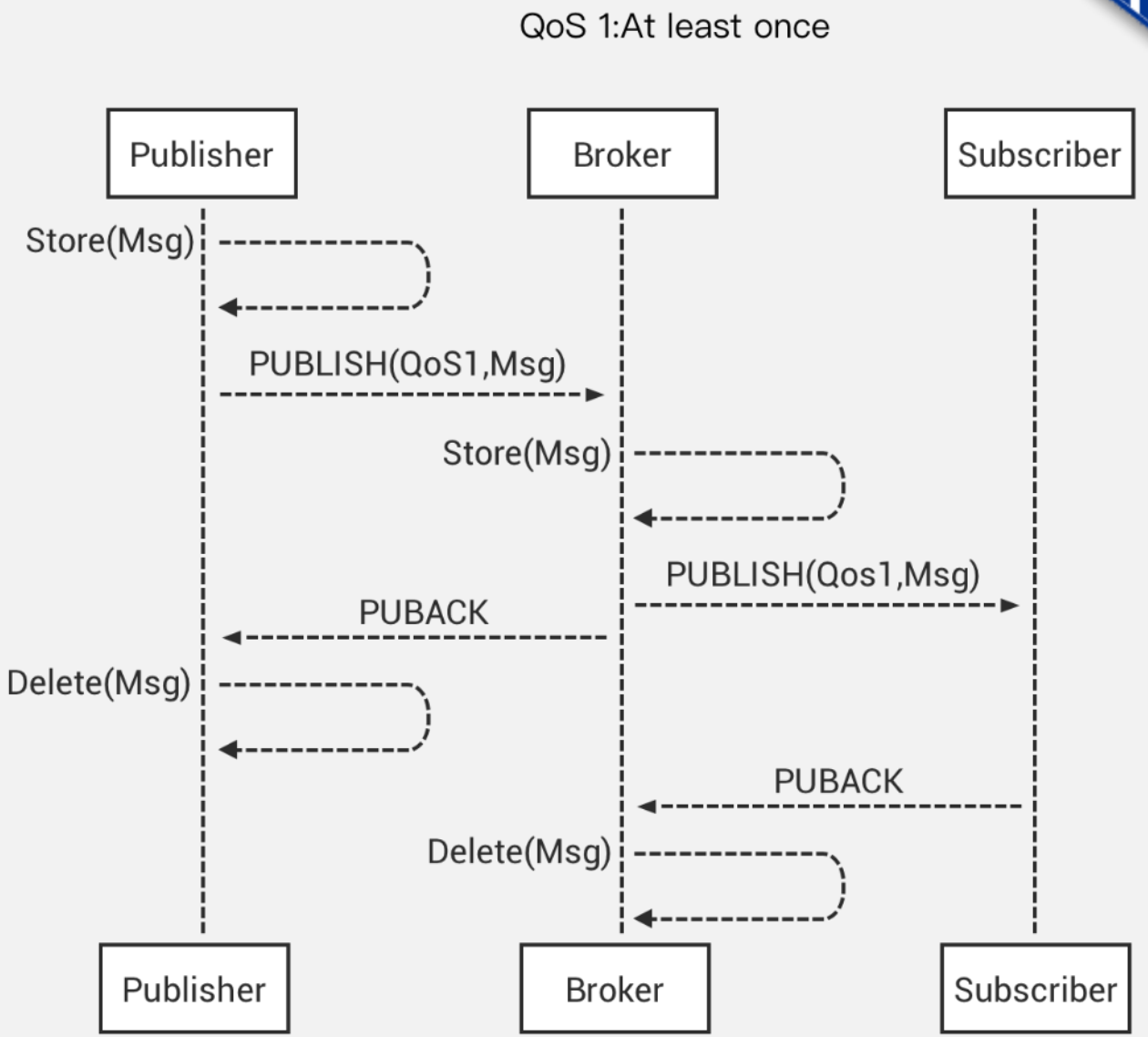
当 QoS 为 0 时，消息的分发依赖于底层网络的能力。发布者只会发布一次消息，接收者不会应答消息，发布者也不会储存和重发消息。消息在这个等级下具有最高的传输效率，但可能送达一次也可能根本没送达。



QoS（服务质量）介绍

Qos 1 - 至少分发一次

当 QoS 为 1 时，可以保证消息至少送达一次。MQTT 通过简单的 ACK 机制来保证 QoS 1。发布者会发布消息，并等待接收者的 PUBACK 报文的应答，如果在规定的时间内没有收到 PUBACK 的应答，发布者会将消息的 DUP 置为 1 并重发消息。接收者接收到 QoS 为 1 的消息时应该回应 PUBACK 报文，接收者可能会多次接受同一个消息，无论 DUP 标志如何，接收者都会将收到的消息当作一个新的消息并发送 PUBACK 报文应答。

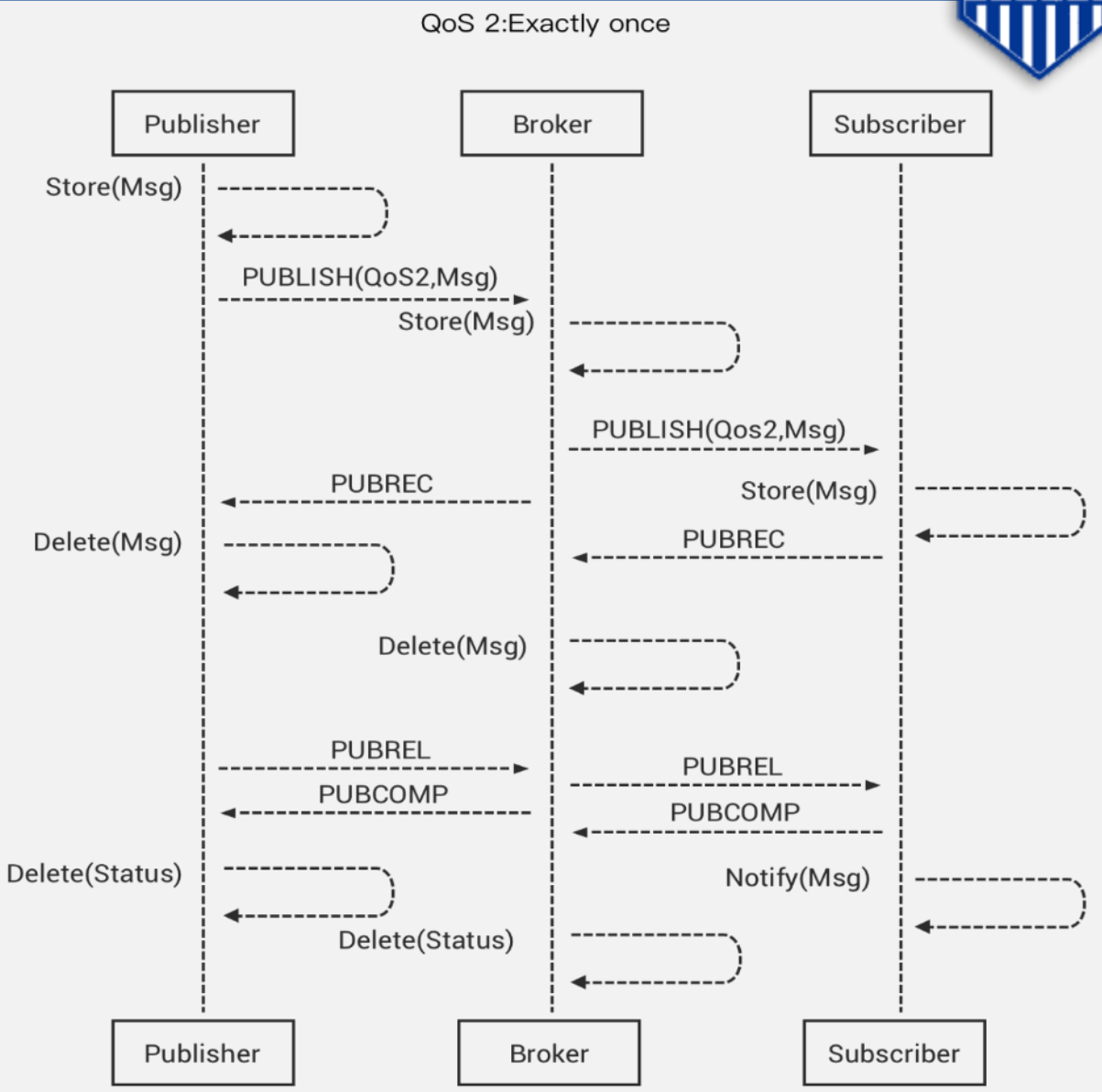


QoS（服务质量）介绍

QoS 2 - 只分发一次

当 QoS 为 2 时，发布者和订阅者通过两次会话来保证消息只被传递一次，这是最高等级的服务质量，消息丢失和重复都是不可接受的。使用这个服务质量等级会有额外的开销。

发布者发布 QoS 为 2 的消息之后，会将发布的消息储存起来并等待接收者回复 PUBREC 的消息，发送者收到 PUBREC 消息后，它就可以安全丢弃掉之前的发布消息，因为它已经知道接收者成功收到了消息。发布者会保存 PUBREC 消息并应答一个 PUBREL，等待接收者回复 PUBCOMP 消息，当发送者收到 PUBCOMP 消息之后会清空之前所保存的状态。

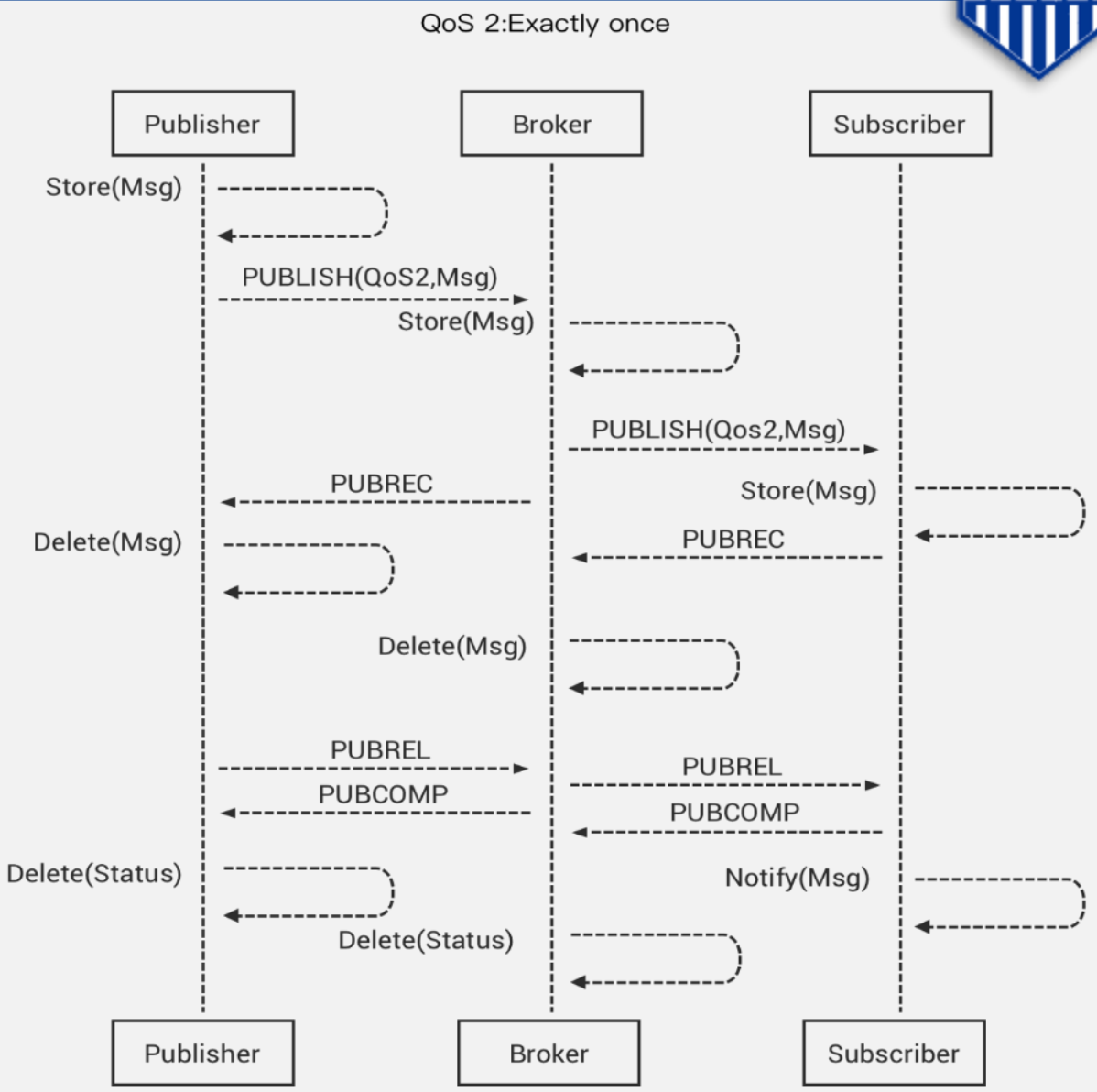


QoS（服务质量）介绍

QoS 2 - 只分发一次

当接收者接收到一条 QoS 为 2 的 PUBLISH 消息时，他会处理此消息并返回一条 PUBREC 进行应答。当接收者收到 PUBREL 消息之后，它会丢弃掉所有已保存的状态，并回复 PUBCOMP。

无论在传输过程中何时出现丢包，发送端都负责重发上一条消息。不管发送端是 Publisher 还是 Broker，都是如此。因此，接收端也需要对每一条命令消息都进行应答。



QoS（服务质量）介绍

MQTT 发布与订阅操作中的 QoS 代表了不同的含义，发布时的 QoS 表示消息发送到服务端时使用的 QoS，订阅时的 QoS 表示服务端向自己转发消息时可以使用的最大 QoS。

不同情况下客户端收到的消息 QoS 可参考下表：

发布消息的 QoS	主题订阅的 QoS	接收消息的 QoS
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	1
2	0	0
2	1	1
2	2	2

总而言之，发布消息的 QoS与主题订阅的 QoS中选择较小的QoS作为订阅者接收消息的QoS

QoS（服务质量）介绍

如何选择 MQTT QoS 等级

QoS 级别越高，流程越复杂，系统资源消耗越大。应用程序可以根据自己的网络场景和业务需求，选择合适的 QoS 级别。

以下情况下可以选择 QoS 0

•可以接受消息偶尔丢失。

•在同一个子网内部的服务间的消息交互，或其他客户端与服务端网络非常稳定的场景。

以下情况下可以选择 QoS 1

•对系统资源消耗较为关注，希望性能最优化。

•消息不能丢失，但能接受并处理重复的消息。

以下情况下可以选择 QoS 2

•不能忍受消息丢失（消息的丢失会造成生命或财产的损失），且不希望收到重复的消息。

•数据完整性与及时性要求较高的银行、消防、航空等行业。

会话保持

MQTT 没有假设设备或 Broker 使用了 TCP 的保活机制，而是设计了协议层的保活机制：在 CONNECT 报文里可设置 Keepalive 字段，来设置保活心跳包 PINGREQ/PINGRESP 的发送时间间隔。当长时间无法收到设备的 PINGREQ 的时候，Broker 就会认为设备已经下线。

总的来说，Keepalive 有两个作用：

- 发现对端下线或者网络中断
- 在长时间无消息交互的情况下，保持连接不被网络设备断开

对于那些想要在重新上线后，重新收到离线期间错过的消息的设备，MQTT 设计了持久化连接：在 CONNECT 报文里可设置 CleanSession 字段为 False，则 Broker 会为终端存储：

- 设备所有的订阅
- 还未被设备确认的 QoS1 和 QoS 消息
- 设备离线时错过的消息

开源 MQTT 服务器如何选择

MQTT - The Standard for IoT Messaging：MQTT英文官方网站

到目前为止，比较流行的开源 MQTT 服务器有几个：

Eclipse Mosquitto：使用 C 语言实现的 MQTT 服务器。Eclipse 组织还还包含了大量的 MQTT 客户端项目：https://www.eclipse.org/paho/#

EMQ X：使用 Erlang 语言开发的 MQTT 服务器，内置强大的规则引擎，支持许多其他 IoT 协议比如 MQTT-SN、 CoAP、LwM2M 等。

Mosca：使用 Node.JS 开发的 MQTT 服务器，简单易用。

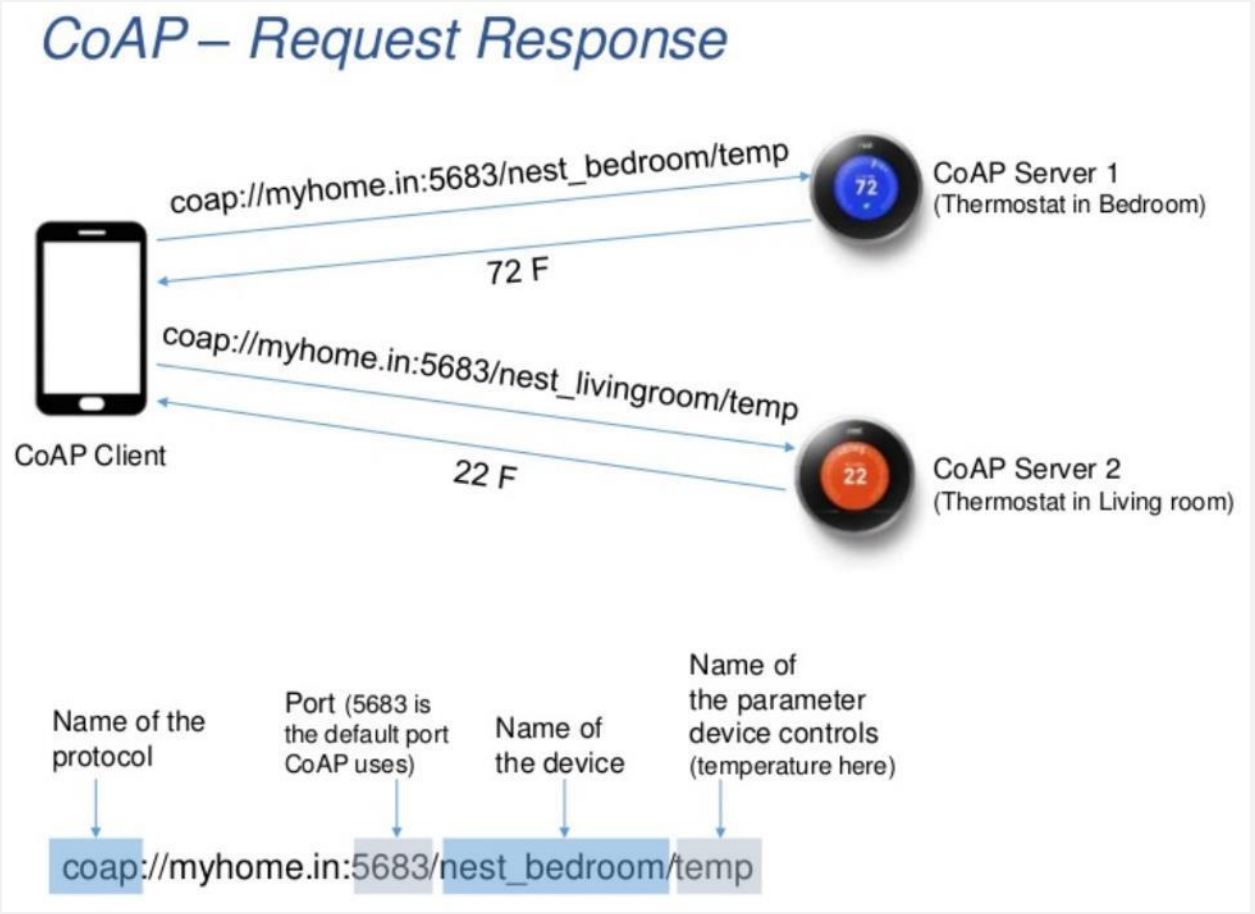
VerneMQ：同样使用 Erlang 开发的 MQTT 服务器。

CoAP 诞生

- 2010年3月，隶属于IETF应用领域的CoRE(Constrained RESTful Environment)工作组宣告成立，为受限节点制定相关的符合REST(Representational State Transfer)风格的应用层协议。
- **The Constrained Application Protocol**（CoAP）是一种专用的Web传输协议，用于受约束的节点和网络。
- CoAP提供了应用程序端点之间的**请求/响应**交互模型，支持服务的资源发现，并包括Web的关键概念，例如URI和Internet媒体类型。

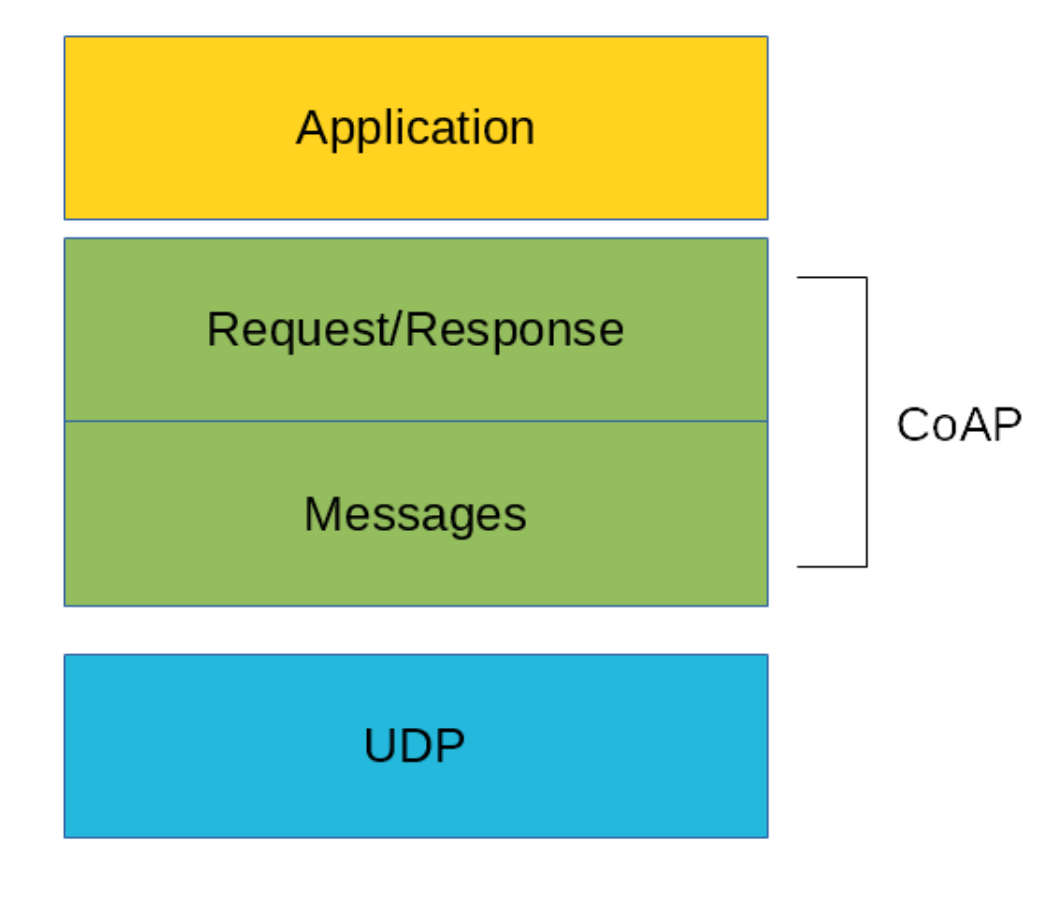
CoAP 应用

- CoAP更适合数据采集的场合，特别是电池供电的传感器设备
- 目前各大IoT平台都支持CoAP协议接入，例如华为物联网平台（OceanConnect），移远物联网平台(Onenet)、阿里云物联网平台都有CoAP协议通信接口



CoAP 特点

- CoAP协议网络运输层为UDP*
- 它基于REST，server的资源地址和互联网一样也有类似url的格式，客户端同样有POST,GET,PUT,DELETE方法来访问server,对HTTP做了简化
- CoAP是二进制格式的，HTTP是文本格式的，COAP比HTTP更加紧凑
- 轻量化，CoAP最小长度仅仅4B
- 支持可靠传输，数据重传，块传输
- 支持IP多播, 即可以同时向多个设备发送请求
- 非长连接通信，适用于低功耗物联网场景。



注：新版本支持TCP，仅限于运输层不可选UDP的场景

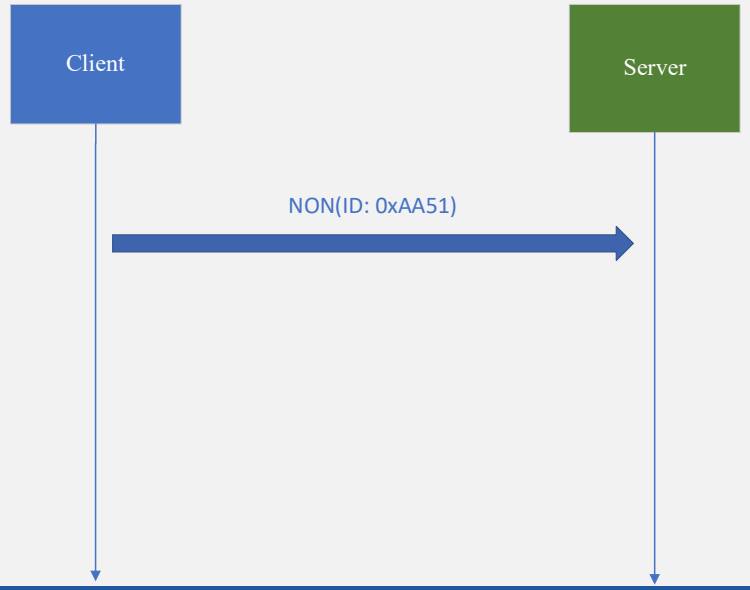
CoAP 消息类型

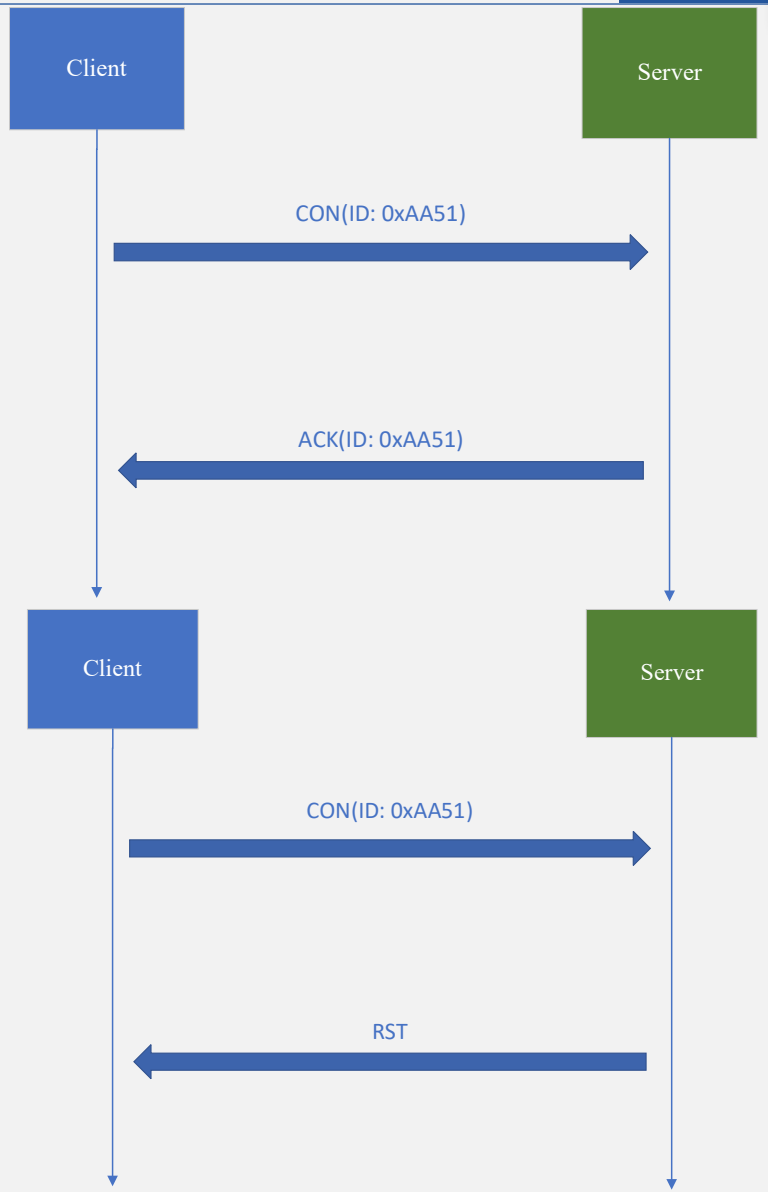
CON—— 需要被确认的请求，如果CON请求被发送，那么对方必须做出响应。

ACK ——应答消息，对应的是CON消息的应答。

RST —— 复位消息，可靠传输时候接收的消息不认识或错误时，不能回ACK消息，必须回RST消息。

NON—— 不需要被确认的请求，如果NON请求被发送，那么对方不必做出回应。



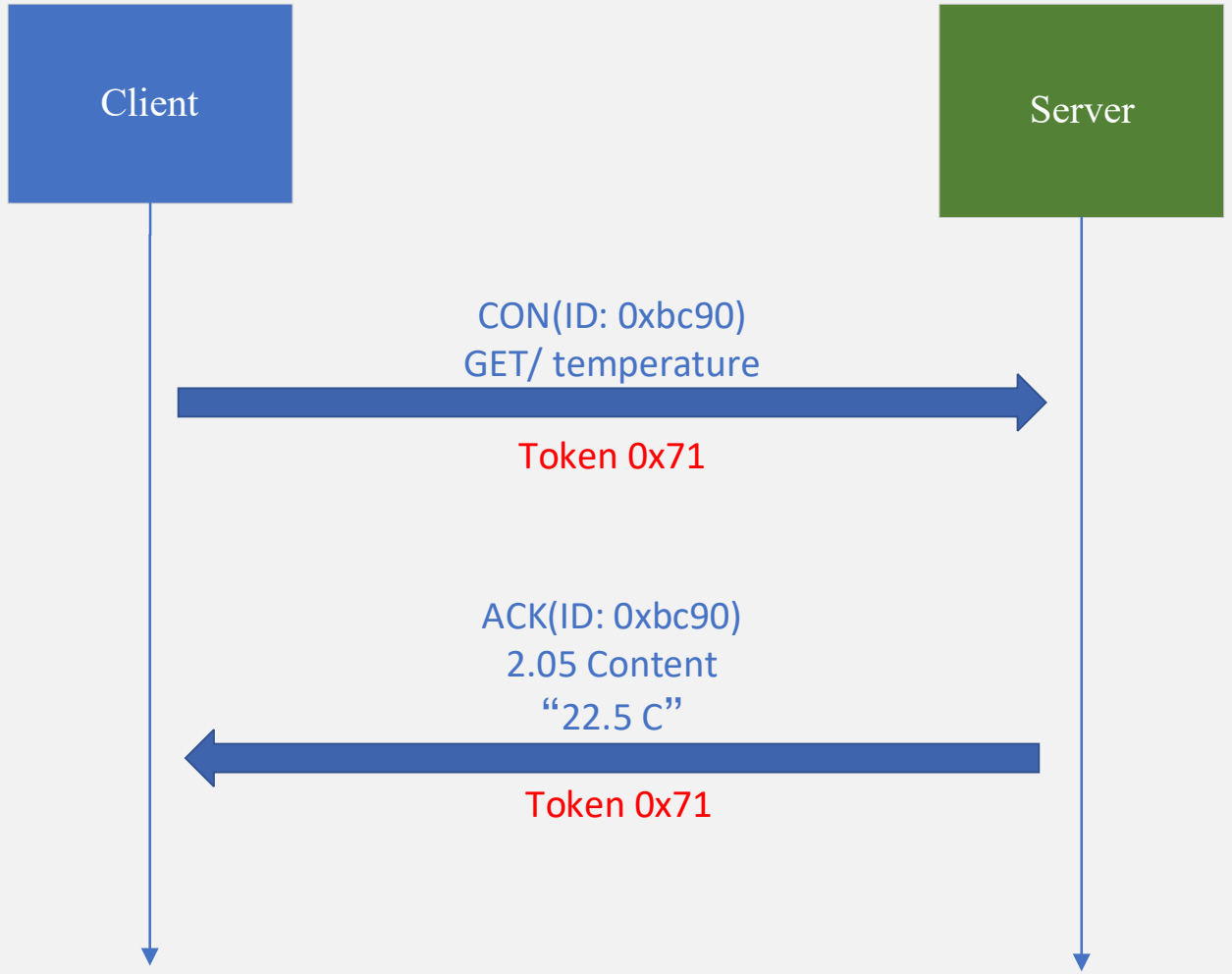


CoAP 请求响应模型

CoAP请求/响应是CoAP抽象层中的第二层。使用“确认”（CON）或“非确认”（NON）消息发送请求。根据服务器是否可以立即响应客户端请求或答案（如果不可用），有两种模式。

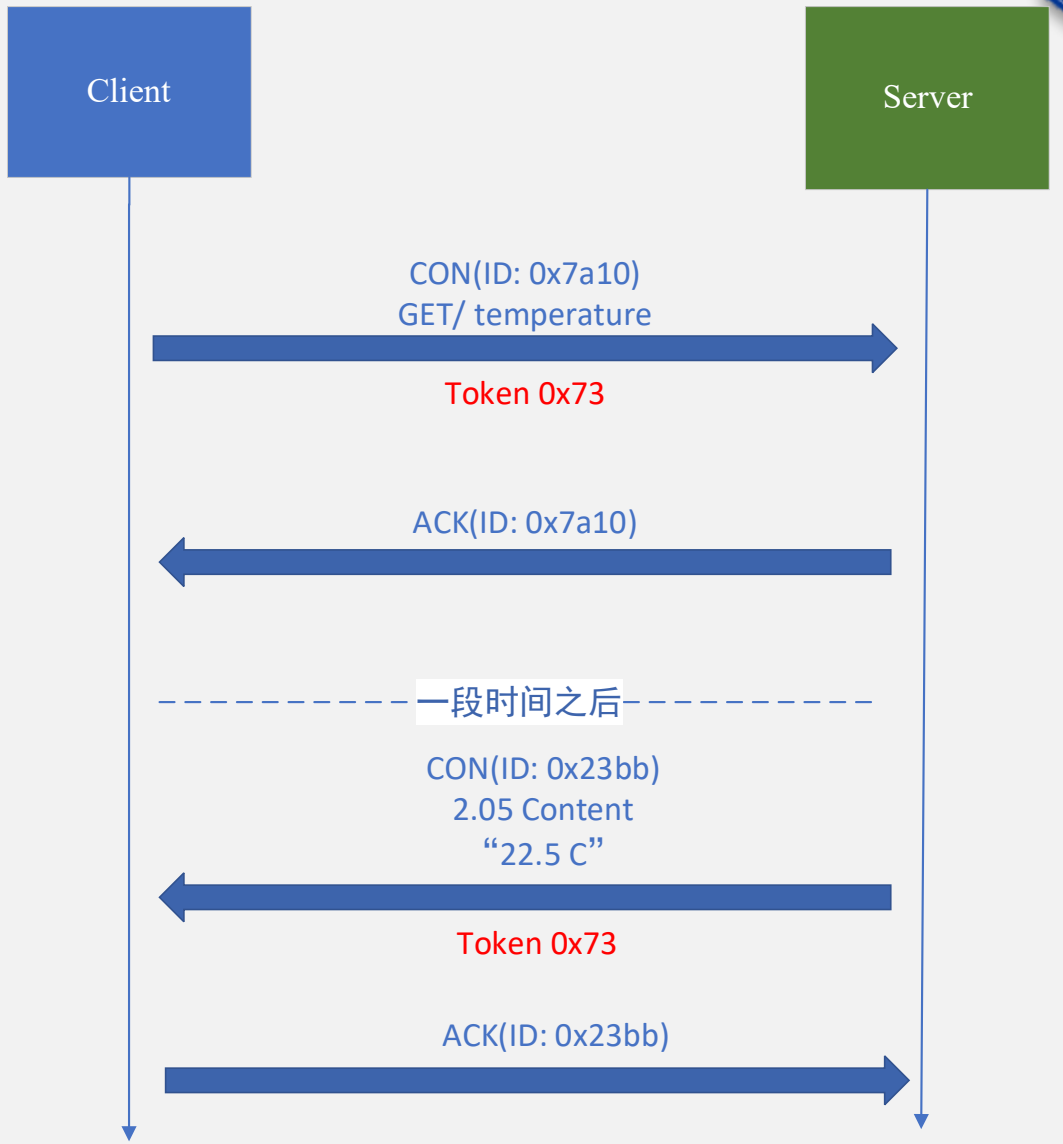
携带模式:客户端发送CON报文向服务器请求，服务器及时响应，且回复的ACK报文中包含响应负载。

携带模式是最常用的请求/响应工作模式，在这种工作模式下，Message ID和Token的作用几乎相同。为了减少CoAP报文长度，更多的情况下只使用Message ID便可。



CoAP 请求响应模型

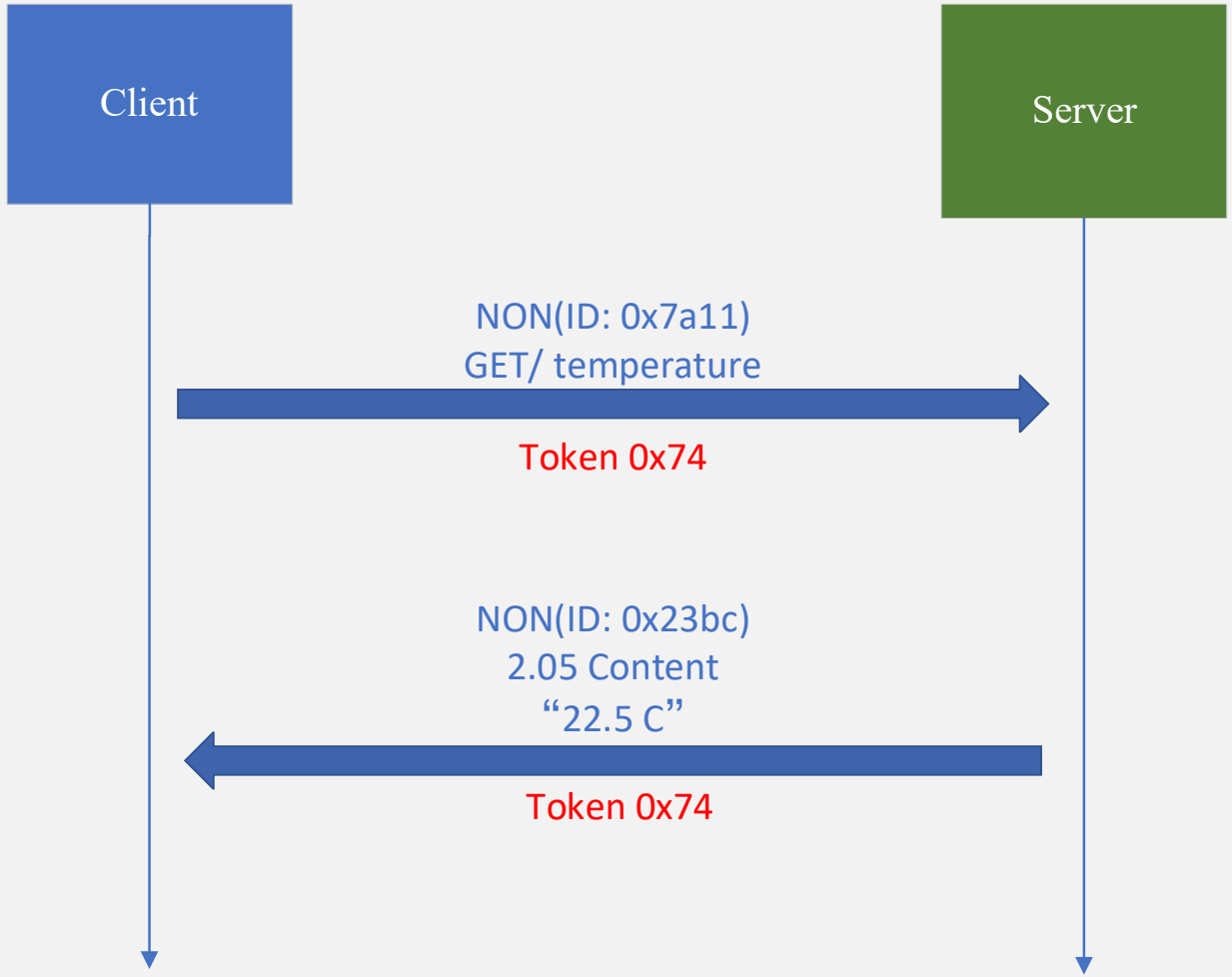
分离模式:客户端发送CON报文向服务器请求，如果服务器无法立即响应来自客户端的请求，则它将发送带有空响应的ACK。一旦响应可用，服务器就会向客户端发送一条新的CON报文，其中包含响应。



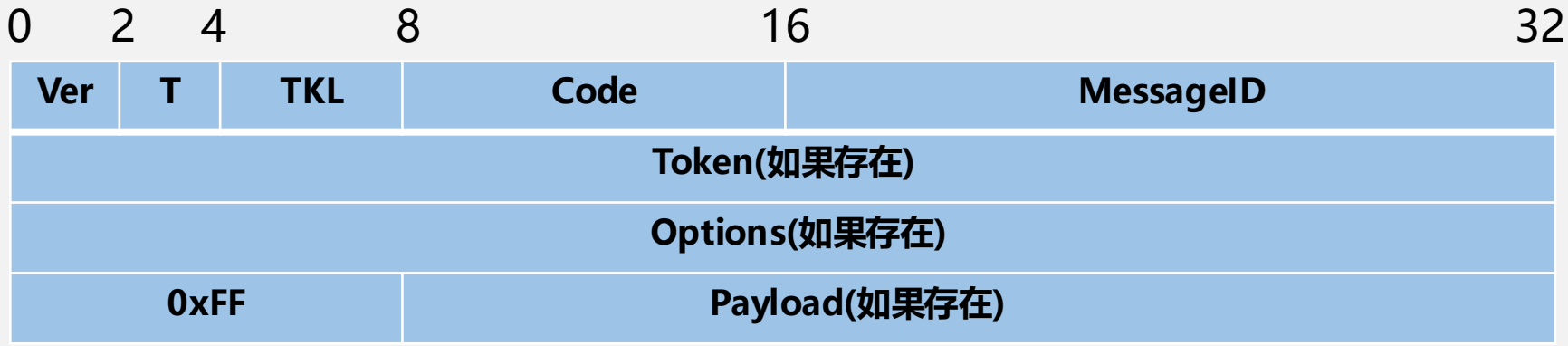
CoAP请求响应模型

非确认模式:客户端发送NON报文向服务器请求，此时服务器无需返回ACK确认，而是返回NON报文对客户端的请求做出响应。

非确认模式是CoAP中最为松散的请求/响应工作模式。允许设备”犯错”。



CoAP 首部分析



CoAP是一个完整的二进制应用层协议，CoAP首部包括版本编号Ver、报文类型T、标签长度指示TKL、准则Code、报文序号Message ID、标签Token、选项Options、分隔符0xFF和负载Payload等几个部分。其中版本编号Ver、报文类型T、标签长度指示TKL、准则Code和报文序号Message ID为必要部分，也就是说这几部分一定会出现在CoAP请求或响应中。而标签Token、选项Options、分隔符0xFF和负载Payload为非必要部分，这些部分均为可选部分。

CoAP 方法

GET方法: 用于查询资源，该资源通过请求中的URI进行识别。由于单个资源服务器可能使用不同的媒体类型展示某资源，所以GET请求中可包括一个或多个Accept选项用于指示客户端期望获得的媒体类型。若GET请求被正确执行，2.05（Content）或2.03（Valid）响应码将会出现在CoAP响应报文中。无论是在HTTP应用中还是在CoAP应用中，GET总是最常用的方法。

POST方法: 要求CoAP请求中的资源描述内容被CoAP服务器处理。如果POST请求被正确执行，那么在服务器上将创建一个新资源。一旦在服务器上创建一个新资源，服务器返回的响应中将包括一个2.01（Create）响应码并且响应负载中包括一个新建资源的URI，该URI可使用一个或多个Location-Path和Location-Query定义。如果POST请求被成功执行但未在服务器上创建新资源，那么响应消息中应包含一个2.04（Changed）响应码；如果POST请求被成功执行但导致服务器上的目标资源被删除，那么响应消息中应包含一个2.02 （Deleted）响应码。

CoAP 方法

PUT方法: 要求服务器根据CoAP请求中的URI和CoAP请求负载中的资源描述信息更新服务器内的指定资源。如果资源已经存在，那么服务器将会更新指定资源，同时返回一个包含2.04（Changed）响应码的CoAP响应；如果资源不存在，那么服务器将会根据请求中的URI创建一个新的资源，同时返回一个包含2.01（Created）响应码的CoAP响应。如果该资源既不能被创建也不能被更新，那么服务器将返回一个合适的错误响应码。

DELETE方法:要求服务器根据CoAP请求中的URI删除服务器中的指定资源。如果资源删除成功，那么服务器应返回一个包含2.02（Deleted）响应码的CoAP响应。

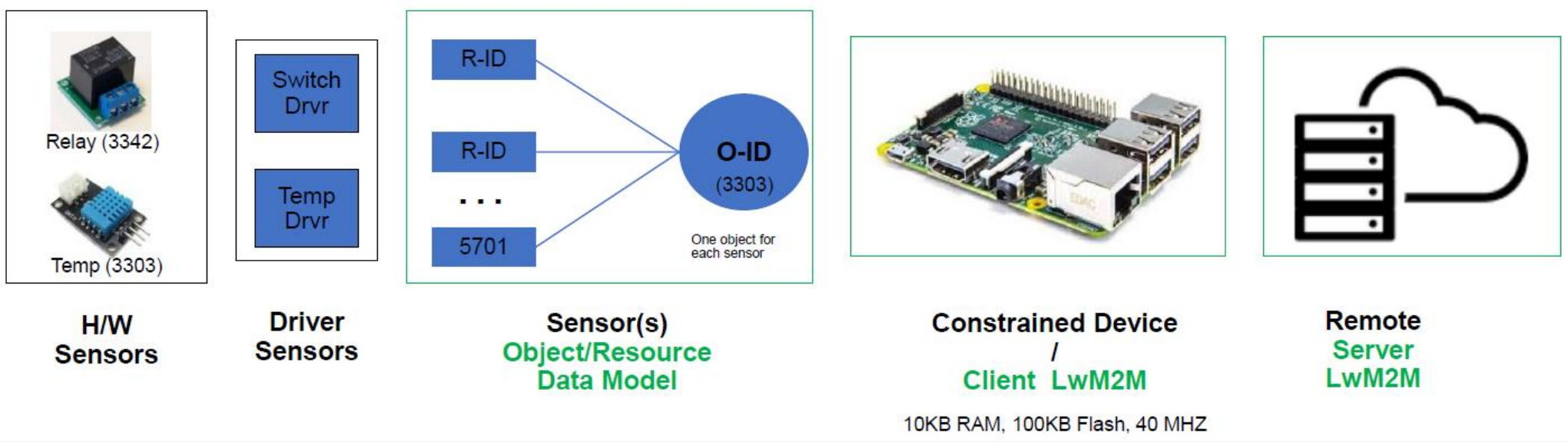
CoAP 开源

名称	开发语言	CoAP版本	客户端/服务端	实现的CoAP特征	开源协议	项目链接地址
libcoap	C	RFC 7252	Client + Server	Observe, Blockwise Transfers	BSD/GPL	http://sourceforge.net/projects/libcoap/develop
Californium	Java	RFC 7252	Client + Server	Observe, Blockwise Transfers, DTLS	EPL+EDL	https://www.eclipse.org/californium
cantcoap	C++/C	RFC 7252	Client + Server		BSD	https://github.com/staropram/cantcoap
CoAP implementation for Go	Go	RFC 7252	Client + Server	Core + Draft Subscribe	MIT	https://github.com/dustin/go-coap
CoAPthon	Python	RFC 7252	Client + Server + Forward Proxy + Reverse Proxy	Observe, Multicast server discovery, CoRE Link Format parsing, Block-wise	MIT	https://github.com/Tanganelli/CoAPthon

LwM2M 简介/主要特点

LwM2M（Lightweight M2M，轻量级M2M），由开发移动联盟（OMA）提出，是一种轻量级的、标准通用的物联网设备管理协议，可用于快速部署客户端/服务器模式的物联网业务。

- 2017年02月，OMA LwM2M 1.0发布；支持UDP，DTLS加密
- 2018年06月，OMA LwM2M 1.1发布，支持TCP，针对LTE-M，NB-IoT和LoRa设备优化
- 2020年11月，OMA LwM2M 1.2发布，支持MQTT/HTTP，针对5G NR设备优化

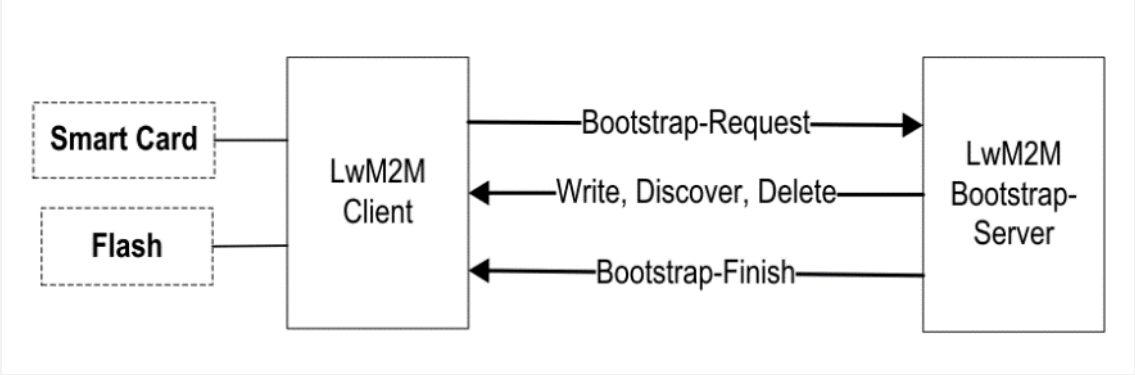


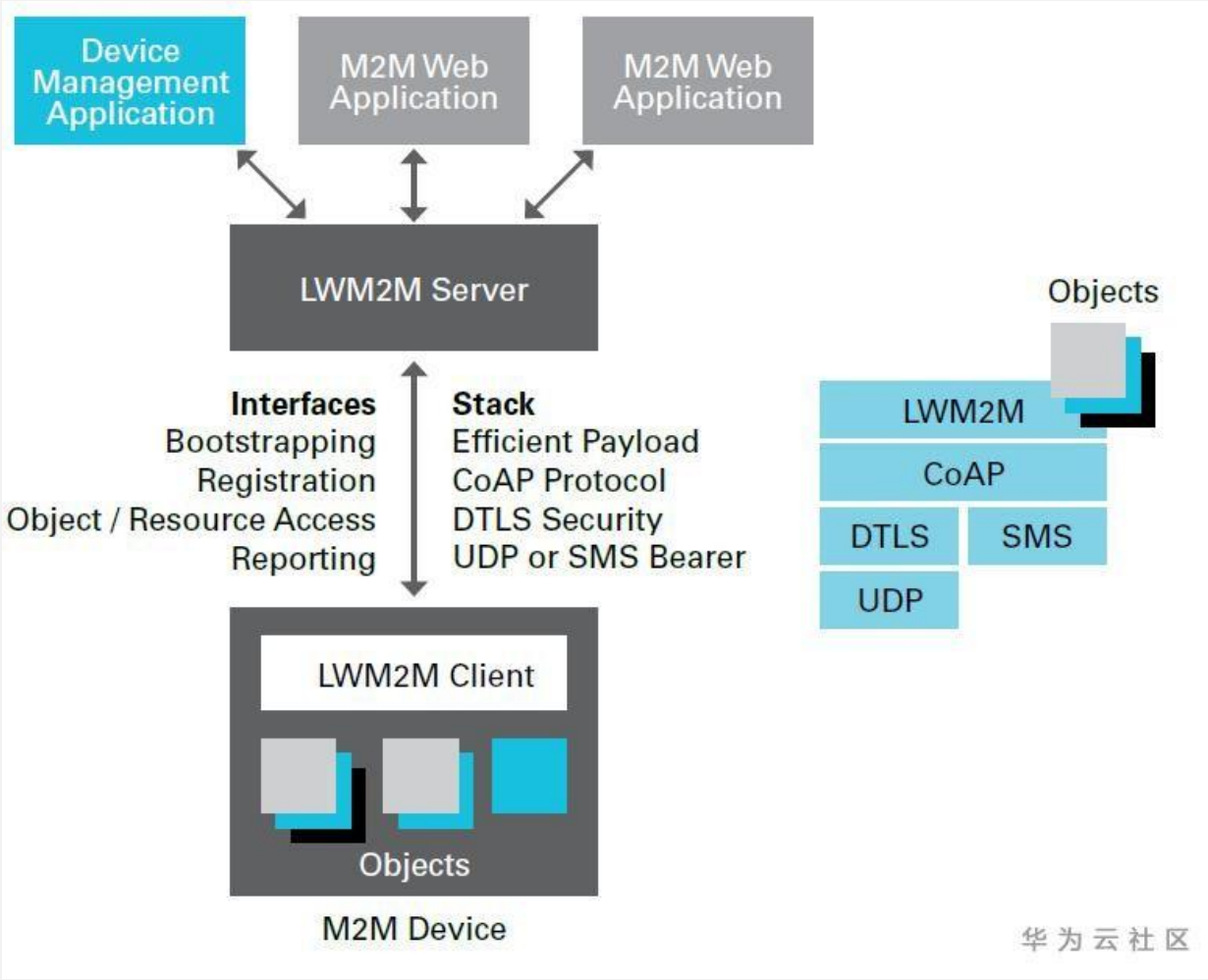
LwM2M 协议架构

LwM2M协议适用于轻量级的各种物联网设备，

LwM2M定义了三个逻辑实体:

- LwM2M Server 服务器；
- LwM2M Client 客户端，负责执行服务器的命令和上报执行结果；
- LwM2M 引导服务器 Bootstrap Server，负责配置LwM2M客户端。





LwM2M 对象定义

- 对象是逻辑上用于特定目的的一组资源的集合。例如固件更新对象，它就包含了用于固件更新目的的所有资源，例如固件包、固件URL、执行更新、更新结果等。
- 使用对象的功能之前，必须对该对象进行实例化，对象可以有多个对象实例，对象实例的编号从0开始递增。
- OMA定义了一些标准对象，LwM2M协议为这些对象及其资源已经定义了固定的ID。例如：固件更新对象的对象ID为5，该对象内部有8个资源，资源ID分别为0~7，其中“固件包名字”这个资源的ID为6。因此，URI 5/0/6表示：固件更新对象第0个实例的固件包名字这个资源。

Name	Object ID	Instances	Mandatory	Object URN
Object Name	16-bit Unsigned Integer	Multiple/Single	Mandatory/Optional	urn:oma:LwM2M:{oma,ext,x}:{Object ID}

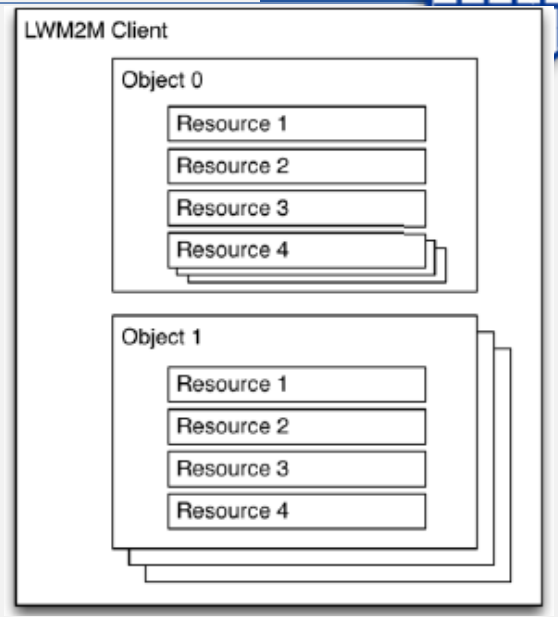
LwM2M 对象定义

OMA为LwM2M协议内置了8个对象。

Object	Object ID	Description
LwM2M Security	0	LwM2M(bootstrap)server的URI,payload的安全模式,一些算法/密钥,server的短ID等消息。
LwM2M Server	1	Server的短ID,注册的生命周期,observe的最小/最大周期,绑定模型等。
Access Control	2	每个Object的访问控制权限。
Device	3	设备的制造商，型号，序列号，电量，内存等消息。
Connectivity Monitoring	4	网络制式，链路质量，IP地址等消息。
Firmware	5	固件包，包的URI，状态，更新结果等。
Location	6	经纬度，海拔，时间戳等。
Connectivity Statistics	7	收集期间的收发数据量，包大小等信息。

LwM2M 资源定义

LwM2M定义了一个资源模型，所有信息都可以抽象为资源以提供访问。资源是对象的内在组成，隶属于对象，LwM2M客户端可以拥有任意数量的资源。和对象一样，资源也可以有多个实例。



资源定义的格式

ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeratio	Units	Description
Resource ID	Resource Name	R(Read), W(Write), E(Execute)	Multiple/Single	Mandatory /Optional	String, Integer, Float, Boolean, Opaque, Time, Objlnk none	If any	If any	Description

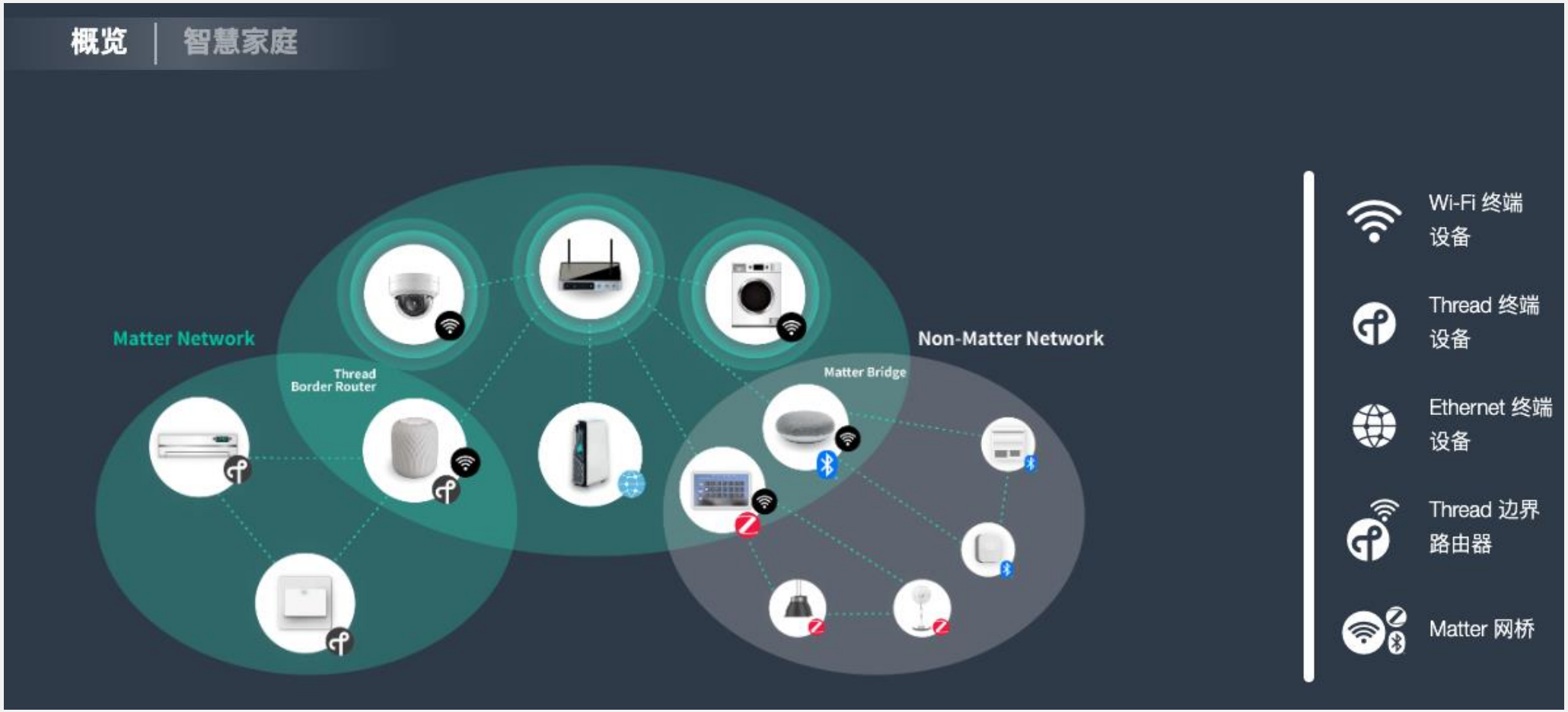
LwM2M 开源实现

- OMA LwM2M DevKit：提供可视化界面与LwM2M服务器交互。
- Eclipse Leshan：基于Java，提供了LwM2M服务器与LwM2M客户端的实现。
- Eclipse Wakaama：基于C，提供了LwM2M服务器与LwM2M客户端的实现。
- AVSystem Anjay：基于C，提供了LwM2M客户端的实现。

MQTT VS CoAP VS LwM2M

类别	MQTT	CoAP	LwM2M
主要通信机制	异步（按字节）	同步（按比特）	同步（按比特）
主要连接方式	TCP	UDP	CoAP+UDP
通信模式	多对多(服务器对服务器，设备对服务器，设备对APP)	多(设备)对一(服务器),系统架构类似于传统Web	多(设备)对一(服务器),系统架构类似于传统Web，设备管理
使用场景	更适用于推送和IM（Instant Messaging）	低功耗物联网	低功耗物联网
功耗	功耗高	功耗低	功耗低
支持平台	华为（OceanConnect），阿里云，腾讯云，中移动(Onenet)等主流云平台	华为（OceanConnect），阿里云，腾讯云，中移动(Onenet)等主流云平台	华为（OceanConnect），阿里云，腾讯云，中移动(Onenet)等主流云平台 定制化
反向控制	可用反向控制	反向控制复杂	反向控制复杂

MATTER



Matter 是由 CSA（Connectivity Standards Alliance，连接标准联盟）发起，在行业领导者的参与和承诺下定义的智能家居行业统一标准，旨在为智能家居设备提供安全可靠的无缝连接。Matter 是基于 IP 的连接协议，支持通过 Wi-Fi、以太网和 Thread (IEEE 802.15.4) 进行数据传输，并使用 Bluetooth LE 进行配网。

串行接口通信

- 串行通信是指使用一条数据线，将数据一位一位地依次传输，每一位数据占据一个固定的时间长度。接收方需要一位一位地从单条数据线上接收数据，并且将它们重新组装成一个数据。串行通信只需少数几条线就可以在节点间交换信息，特别适用于计算机与计算机、计算机与外设之间的远距离通信。
- 完成串行通信的接口电路称为串行接口
- 优点：所需数据线少，通信成本低，传输距离可以从几米到几千米
- 缺点：传输速率慢，效率不高

串行接口通信

- 串行通信常见接口标准
  - EIA-232(**RS-232**)，1970年
  - EIA-422(RS-422)
  - EIA-449(RS-449)
  - EIA-485(**RS-485**)
  - EIA-530(RS-530)

- EIA：美国电子工业协会(Electronic Industries Alliance)
- RS： 推荐标准（Recommended Standard）

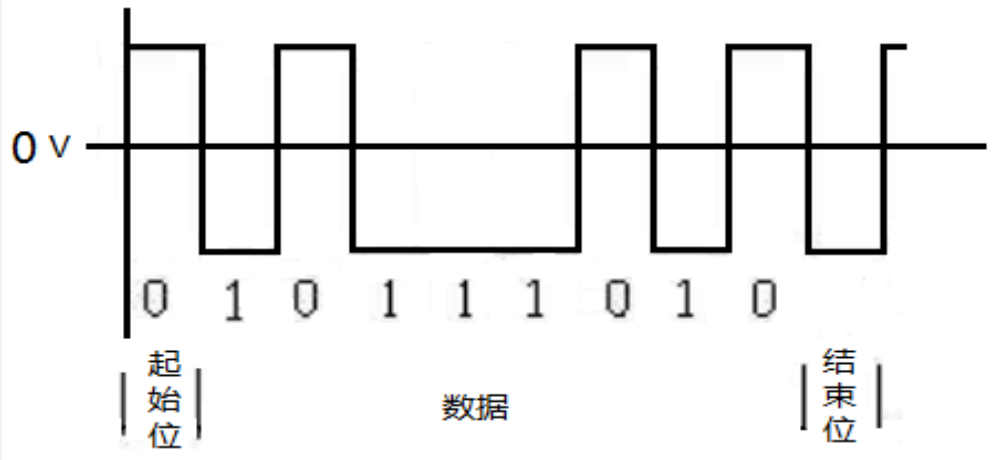
串行接口通信

- 3 种串行接口的部分性能参数

性 能 参 数	RS-232	RS-422	RS-485
工作方式	单端	差分	差分
节点数（通常）	1收1发	1发10收	1发32收
最大传输电缆长度	15米	1219米	1219米
双工方式	全双工(2线)	全双工(4线)	半双工(2线)
优点	简洁、实施成本低	抗共模干扰能力增强	抗共模干扰能力增强
缺点	对噪声较为敏感	成本高	成本较高

串行接口通信-RS232

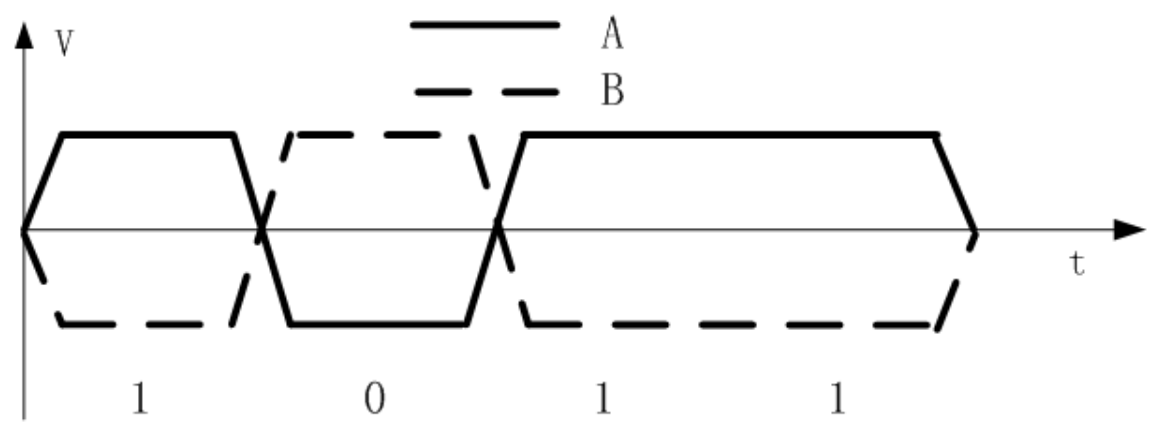
- 电气标准
  - 逻辑1和逻辑0
    - 负电平被规定为逻辑1，负电平振幅：-3V~-15V
    - 正电平被规定为逻辑0，正电平振幅：+3V~+15V
  - 单端不归零编码（NRZ）



  - 与TTL电路连接需要做电平转换

串行接口通信-RS422/485

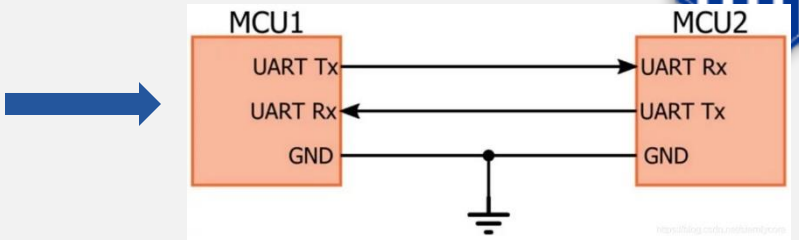
- 电气特性
  - 全/半双工，差分传输
    - 逻辑1，两线间电压差为正电平+2V～+6V
    - 逻辑0，两线间电压差为负电平-2V～-6V



  - 接口信号电平低于RS232，不易损坏接口电路芯片
  - 抗共模干扰能力增强

UART

通用异步收发传输器（Universal Asynchronous Receiver/Transmitter，通常称作UART） 是一种串行异步收发协议



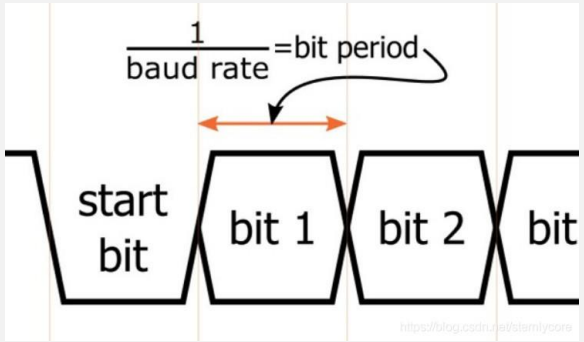
传输方式：传输过程默认是发送端按小端优先序逐字节、逐位传输

空闲位：当总线处于空闲状态时信号线的状态为‘1’，进行数据传输时发送方要先发出一个低电平’0’来表示传输字符的开始。

数据位：数据可以是5，6，7，8，9位，构成一个字符，一般都是8位。先发送最低位最后发送最高位。

奇偶校验位：1位

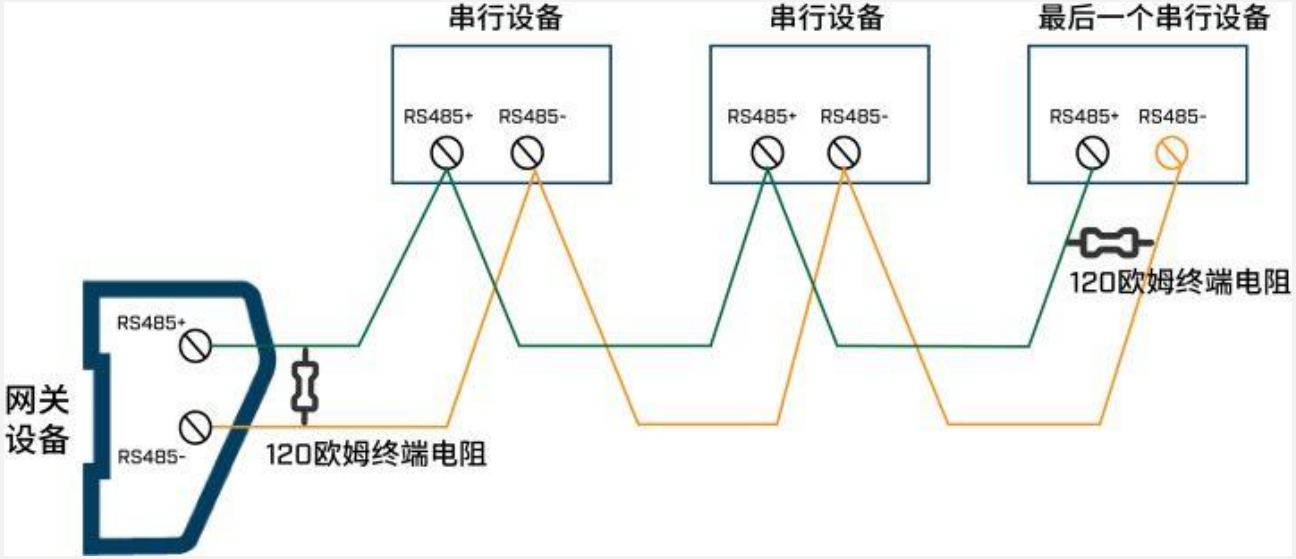
波特率：数据传输速率使用波特率来表示，常见的波特率9600bps，传输一个比特需要的时间是1/9600≈104.2us。



例：'O’的ASCII为79，对应的二进制数据为01001111 ，'K’对应的二进制数据为01001011

串口多用户共享

- 传输模式
  - RS482/422允许在一条平衡总线上连接多个接收器
  - 一个主设备，其余为从设备，完成点对多点双向通信
  - 通常采用主设备呼叫，从设备应答的方式

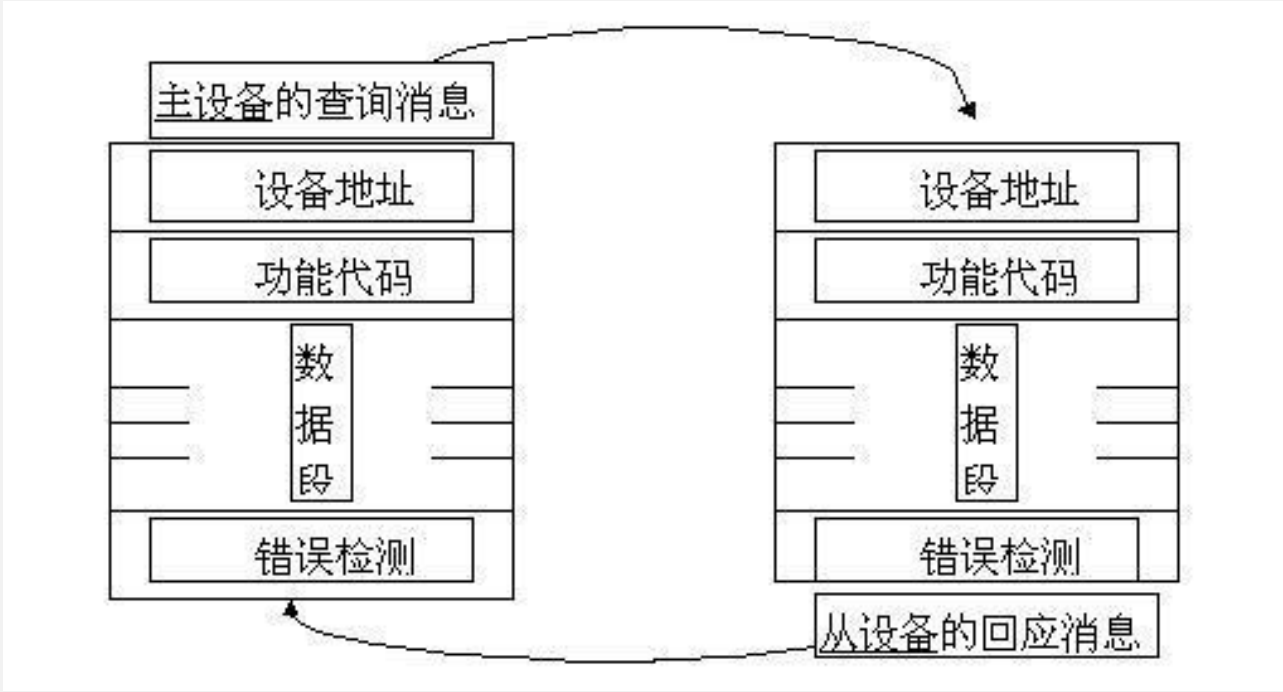


Modbus

- 1979年Modicon（现施耐德电气）公司开发，2002年Modbus组织成立，支持多种传输介质，包括串行通信（如RS-232、RS-485）和TCP/IP网络

典型的通信流程：

- 建立连接：主站与从站建立通信连接。
- 发送请求：主站发送命令（如读取寄存器的请求）到从站。
- 处理请求：从站接收并解析命令，执行请求的操作（读取或写入寄存器）。
- 生成响应：从站将操作的结果或响应数据打包成响应帧发送回主站。
- 断开连接：通信完成后，主站可以选择关闭连接或继续与从站通信。



总结

- 本章主要内容：MQTT、CoAP、LwM2M、Matter、UART和Modbus传输协议的主要特点，协议概述和应用方法。
- 本章学习目标
  - 了解物联网应用层传输协议的主要特点；
  - 熟悉物联网应用层传输协议的基本原理和使用方法；
  - 掌握物联网应用层传输协议的适用场景。