Wifi简介

一、什么wifi

- Wi-Fi的来源： 1999年时各个厂商为了统一兼容802.11标准的设备而结成了一个标准联盟，称为Wi-Fi 联盟。也即，Wi-Fi实际为制定802.11无线网络的组织，并非代表无线网络。但是后来人们逐渐习惯用Wi-Fi来称呼802.11协议。

- 目前无线局域网（WLAN）,主流采用802.11协议。故常直接称为WIFI网络。

wifi常见标准协议

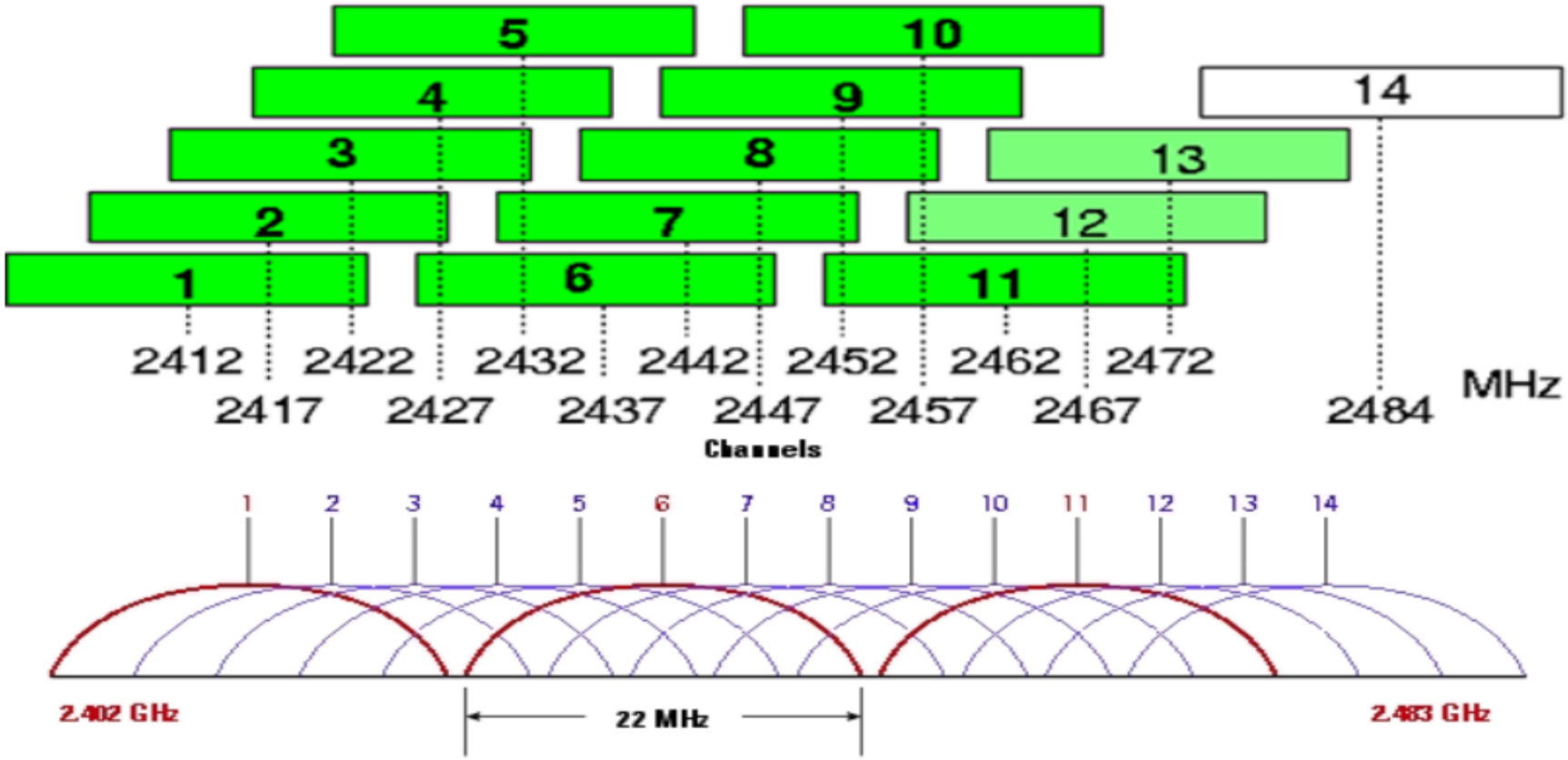
协议	发布时间	频率	带宽	最大速率	兼容性
802.11	1997.06	2.4GHZ	20MHz	2Mbps	802.11
802.11b	1999.09	2.4GHZ	20MHz	11Mbps	802.11b
802.11a	1999.09	5GHZ	20MHz	54Mbps	802.11a
802.11g	2003.06	2.4GHZ	20MHz	54Mbps	802.11b/g
802.11n	2009.10	2.4/5GHZ	20/40MHz	600Mbps	802.11a/b/g
802.11ac	2012.01	5GHZ	20/40/80/160MHz	6.9Gbps	802.11a/b/g/n

目前市面上的设备大多为802.11n和802.11ac

wifi信道（信道也称作频段）

- wifi每个信道带宽都为20Mhz
- 在2.4G频段，共有11个信道，但相邻间有重叠，实际只有3个不重叠信道。
- 在5G频段，早期是5个信道，现在有13个信道。每个信道彼此都不重叠。

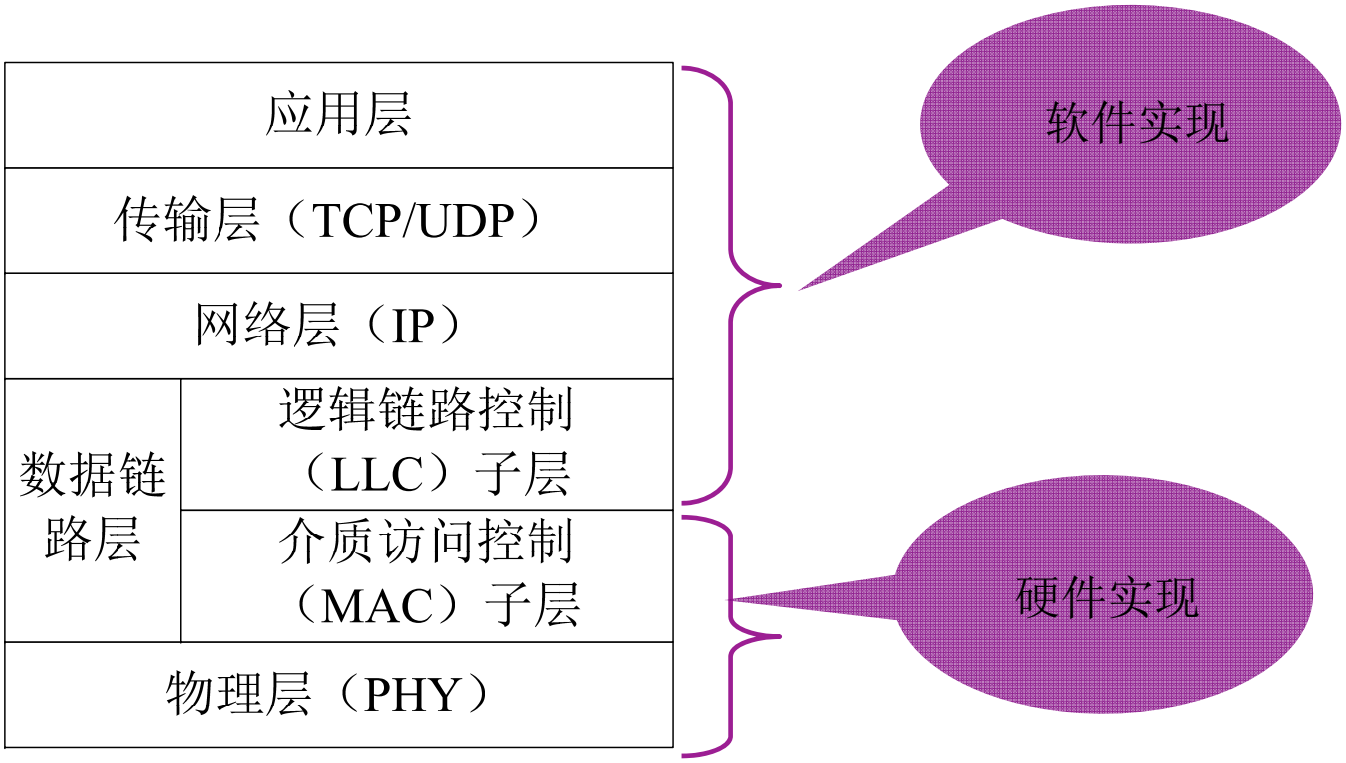
2. 4G信道


5
10
4
9
14
3
8
13
2
7
12
1
6
11
2412
2422
2432
2442
2452
2462
2472
2417
2427
2437
2447
2457
2467
2484
MHz
Channels
1
2
3
4
5
6
7
8
9
10
11
12
13
14
2.402 GHz
22 MHz
2.483 GHz

wifi 2.4G与5G优缺点

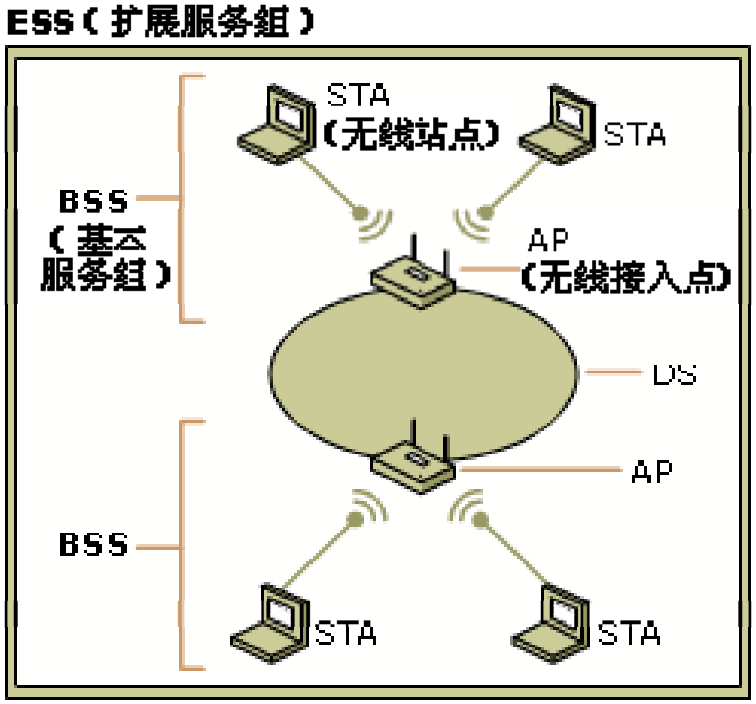
- 2.4GHz频段由于使用ISM频段（使用该频段不用授权，但功率要小于1w），干扰较多。如蓝牙；另外一些家用设备，如微波炉，也对该频段产生干扰。
- 目前2.4G的wifi设备过多，且实际只有3个信道，彼此互相干扰。
- 5G有13个信道，设备也少，干扰很小。
- 5G缺点，频点较高，在空间传输时衰减较为严重。同样功率传输距离大致是2.4G的一半。对于家用而言，就是穿墙差。

wifi协议层



MAC层以上，wifi网络与有线网络是一样的。

二、wifi网络拓扑结构



- **模式一：基础网，是有中心**（即有AP）

1. 站点（STA）：例，手机
2. 无线接入点（AP）：例，无线路由器
3. 基本服务组件（BSS）：一个AP与多个STA
4. 扩展服务组（ESS）：多个BSS，有相同的SSID
5. 分配系统（DS）：构建BSS间互联和漫游分配等，并使ESS对上层表现得就像一个BSS一样。

该模式下，各STA间必须通过AP来通信

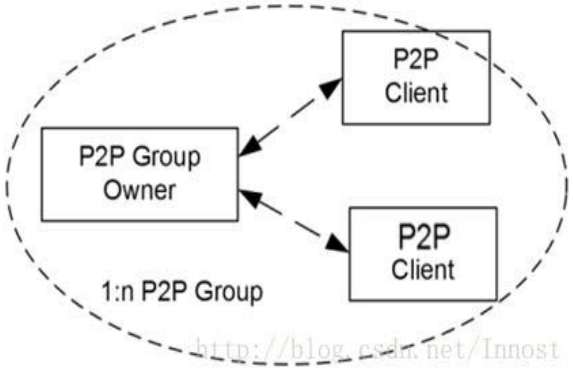
SSID：我们安装AP时会，为该AP分配的一个不超过32字节的服务集标识符。也即，我们手机连接wifi时，看到的wifi名称。

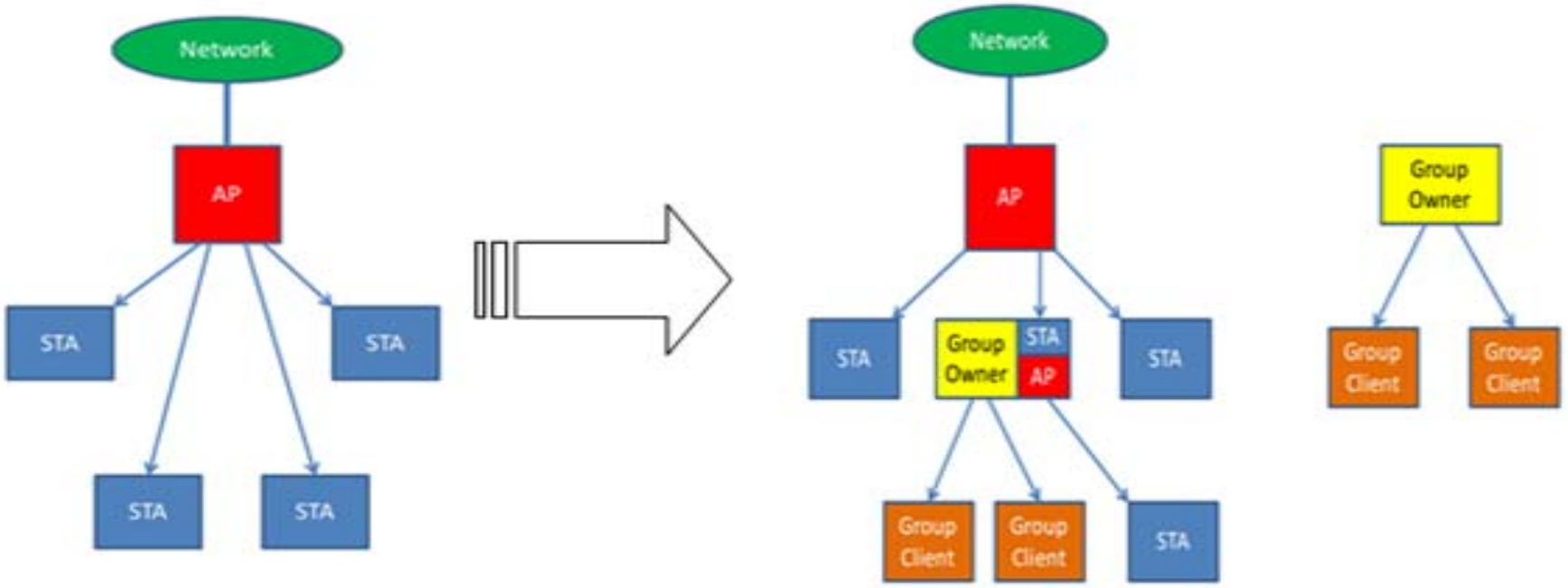
AP一些参数配置

- SSID号：如，XMUNET
- 是否开启SSID广播
- 信道：如，自动
- 模式：如，11n only、11bgn mixed
- 信道带宽：如，自动
- 最大发送速率：如，150Mbps
- 安全设置：如，WPA-PSK/WPA2-PSK
- MAC地址过滤：通过MAC限制可连接的设备

- **模式二：Wifi-direct**（改进型的ad hoc，也称wifi P2P）

1. STA与STA直接通信，并不需要AP的参与；其中一台STA会起到传统意义上的AP的作用，称为Group Owner(GO),另外一台STA则称为Group Client(GC)，像连接AP一样连接到GO。GO和GC不仅可以是一对一，也可以是一对多；比如，一台GO可以同时连接着多台GC。
2. Wi-Fi Direct和传统wifi技术并不是互斥的：GO可以可以像AP一样为几台GC提供服务；它同时可以像传统的STA一样，连接到某个AP；它同时自己也可以是一个AP。




Network
AP
STA
STA
STA
STA
Network
AP
STA
Group Owner
STA
AP
STA
Group Client
Group Client
STA
Group Owner
Group Client
Group Client

Wifi direct特点

- 试图取代蓝牙
- 移动便携
- 即时可用
- 易用性：有个可选服务发现功能，能发现网内可用的服务
- 用途：共享内容，直接打印，有无线投影机，也可直接投影

三、wifi接入过程（有AP）

1. 发现可用网络

2. 选择网络

3. 认证

4. 关联

接入过程——发现可用网络

- 方式一：被动扫描

i） AP周期性地发送信标（Beacon）帧，其包含AP的MAC地址、网络名称（SSID）、支持的速率、认证方式、加密算法等。

ii） 用户主机扫描所有信道，找出可能位于该区域的所有AP发出的信标帧。

- 方式二：主动扫描（因AP的SSID是可以配置为不广播的）

i） 由用户主机在每个信道上发送探测请求（Probe request）帧，寻找与STA所属有相同SSID的AP。（也即，必须先知道AP的SSID）

ii） AP发送探测响应（Probe responce）帧回应，其中包含的信息和信标帧类似。若找不到相同SSID的AP，则一直扫描。

接入过程——选择一个网络进入认证阶段

接入过程——认证

- 认证是STA向AP证明其身份的过程
- 只有通过身份认证的站点才能进行无线接入访问
- 认证可以通过MAC地址进行，也可以通过用户名/口令进行
- 认证有开放系统认证、共享密钥认证、密钥身份验证三种

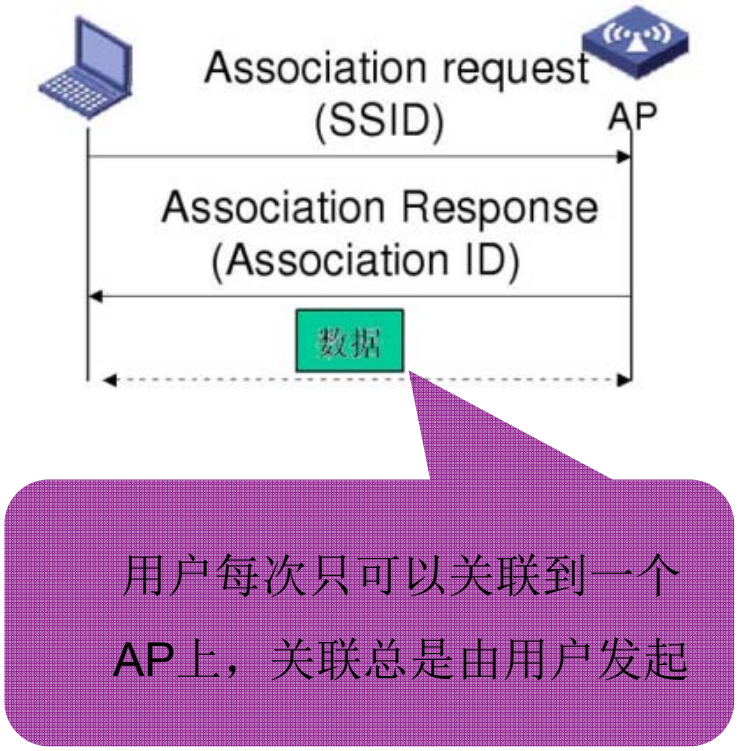
  认证方式：早期为WEP，而后为WPA，WPA2为WPA的加强版
  WPA，WPA2都有个人和企业模式，
  个人模式：名称为WPA-PSK/WPA2-PSK，密钥是共享的
  企业模式：名称为WPA/WPA2，每个人可以有不同的用户名、密码，其需要有一个raidus服务器

- STA和AP均可通过解除认证来终结认证关系

接入过程——关联

如果用户想通过AP接入无线网络，必须同特定的AP关联。

a) 当用户通过网络名称选择指定网络并通过AP认证后，就可以向AP发送关联请求帧。
b) AP将用户信息添加到数据库，向用户回复关联响应，此过程也常被称为注册。
c) 关联建立后，便可以传输数据，且后续的数据传输只能在两者之间进行。
d) 再关联：STA在从一个老的AP移动到新的AP时通过再关联和新的AP建立关联，再关联前必须经历认证过程。
e) 去关联：STA和AP均可以通过去关联和对方解除关联关系。
f) 当STA扫描到信号更强的信的AP时，需先和原来的AP去关联，在才能和新的AP建立关联。



HTTP协议简介

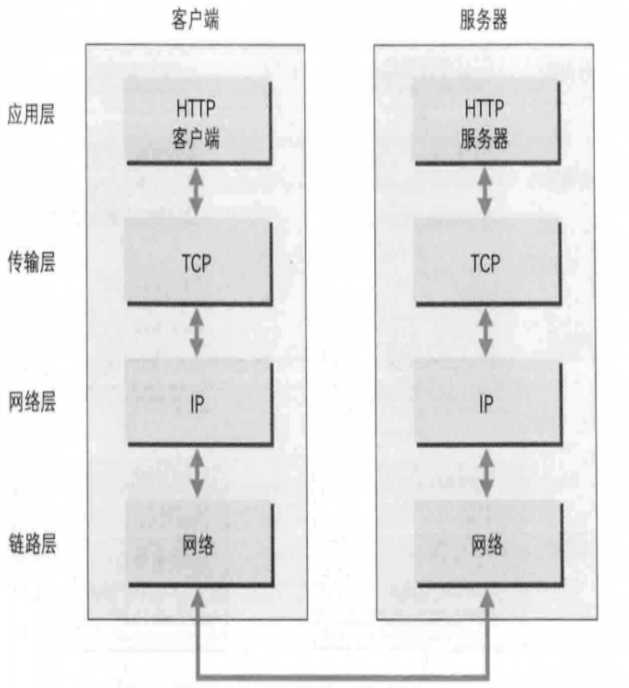
HTTP版本

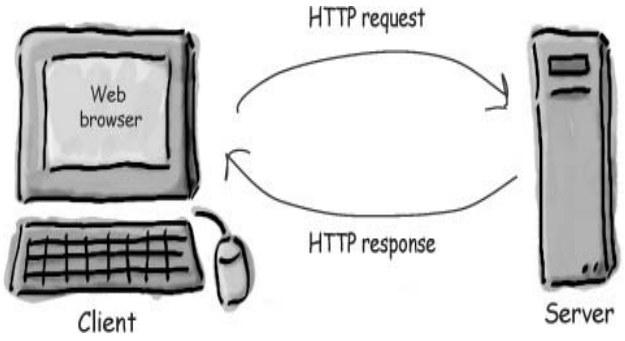
- HTTP/0.9（http 1990年问世，并没有正式标准，其作为http/1.0之前版本统称）
- HTTP/1.0（1996年5月）
- HTTP/1.1（1997年1月，目前主流，本讲以该版本为主）
- HTTP/2（2015年，不叫 HTTP/2.0，因为标准委员会不打算再发布子版本，下一个新版本将是 HTTP/3）

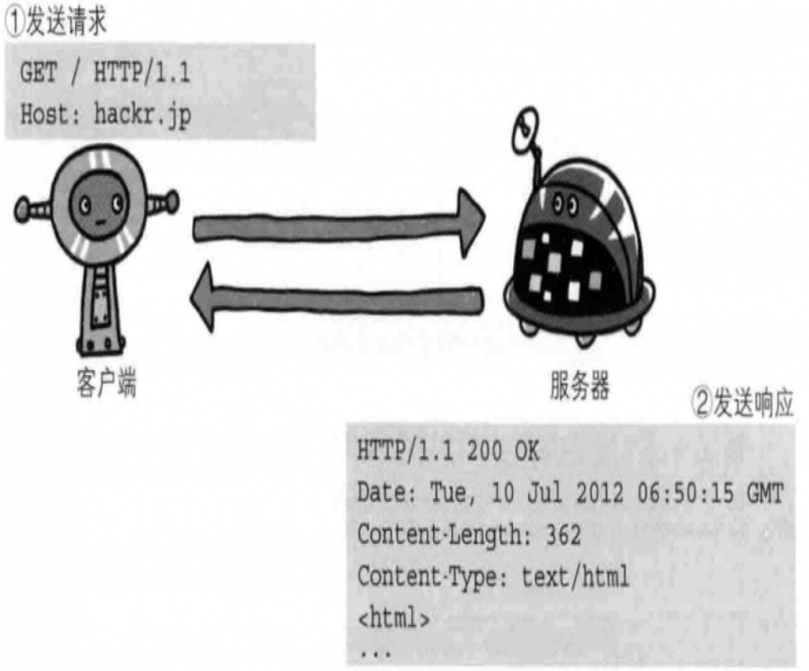
HTTP过程简示

http://user:passwd@www.baidu.com:80/index.html

协议://用户名及密码@　　主机:端口(80)/　　文件名及其路径







http报文结构

http协议交互的信息称为Http报文，分为请求报文和响应报文

起始行	请求行或状态行
头部	请求或响应头部
空行	<回车><换行>
主体	

请求报文例子：

GET /somedir/page.html HTTP/1.1
Host:www.chinaitlab.com
Connection:close
User-agent:Mozilla/4.0
Accept-language:zh-cn
(空行)

响应报文例子：

HTTP/1.1 200 0K
Connectlon:close
Date: Thu, 13 Oct 2005 03:17:33 GMT
Server: Apache/2.0.54 (Unix)
Last-Nodified:Mon,22 Jun 1998 09;23;24 GMT
Content-Length:682
Content-Type:text/html
(空行)
<html>
.....

HTTP请求方法

```
GET /  HTTP/1.1
Host:www.baidu.com
.....
(空行)
```

GET	客户端向服务器请求某个资源。 请求参数和对应的值附加在 URL 后面，利用一个问号(“?”)代表URL 的结尾与请求参数的开始，传递参数长度受限制。 例如，/index.jsp?id=100&op=bind
POST	与GET方法相近。 请求参数封装在HTTP报文的主体中，以名称/值的形式出现，可以传输大量数据。
HEAD	与GET方法类似，但只返回HTTP报文头部。
OPTIONS	用来查询服务器上指定资源的支持的方法
DELETE	删除服务器上的文件
PUT	给服务器上传文件
TRACE	回显服务器收到的请求，主要用于测试或诊断
CONNECT	使用隧道与代理服务器通信

HTTP状态码

响应报文：
HTTP/1.1 200 0K
……

	类别	原因短语
1XX	Informational（信息性状态码）	接收的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕
3XX	Redirection（重定向状态码）	需要进行附加操作以完成请求
4XX	Client Error（客户端错误状态码）	服务器无法处理请求
5XX	Server Error （服务器错误状态码）	服务器处理请求出粗

- 200 0K; 客户端发来请求被正常处理了。
- 400 Bad Request;请求报文中存在语法错误。
- 404 Not Found;服务器上不存在所请求的文档。
- 500 Internal Server Error;服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。这个问题一般是服务器端的源代码出现错误时出现。
- 503 Service Unavailable;服务器临时维护或者过载，当前无法处理请求，将在一段时间后恢复。

cookie

- HTTP协议是无状态协议，不对之前发生过的请求和响应的状态进行管理。
- cookie通过在请求和响应报文中写入cookie来控制客户端状态。
- 服务端发送set-cookie的头部字段信息，让客户端保存cookie。
- 客户端在发送请求报文时，自动加入cookie的值发送出去。

```
GET  /read/  HTTP/1.1
Host: xmu.cn
(没有cookie信息)
```

第一次请求

```
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 07:12:20 GMT
Server: Apache
Set-Cookie: sid=1342077140226724; path=/;
expires=Wed,  10-Oct-2012 07:12:20 GMT
Content-Type: text/plain; charset=UTF-8
.....
```

第一次请求的回应

```
GET  /image/  HTTP/1.1
Host: xmu.cn
Cookie: sid=1342077140226724
```

第二次请求

HTTP安全性和身份认证

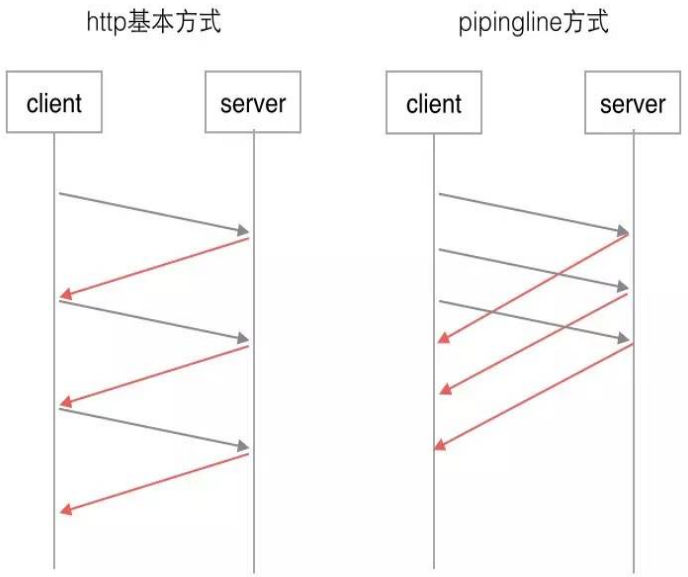
- HTTP不安全处
  - 使用明文（不加密），可能被窃听。
  - 不验证对方是谁，可能遭遇伪装。如，伪网站
  - 无法证明报文完整性，可能遭篡改。如，中间攻击
- HTTPS
  - 与SSL组合使用的HTTP被称为HTTPS 。即在SSL建立安全通信线路后，进行HTTP通信。也就是说，并非是一种新的协议。
- 身份认证（让Web页面只让特定人浏览）
  - BASIC认证（基本认证）
  - DIGEST认证（摘要认证）
  - SSL客户端认证
  - FormBase认证（基于表单认证）

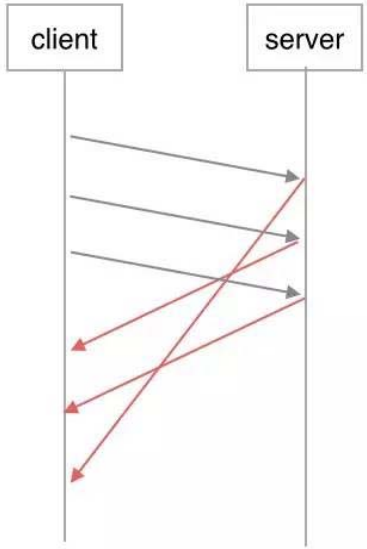
http/2

2015年正式发布

减少了网络延迟，提升了web性能

- 二进制分帧
- 头部压缩
- 多路复用
- 服务端推送





http/2 多路复用

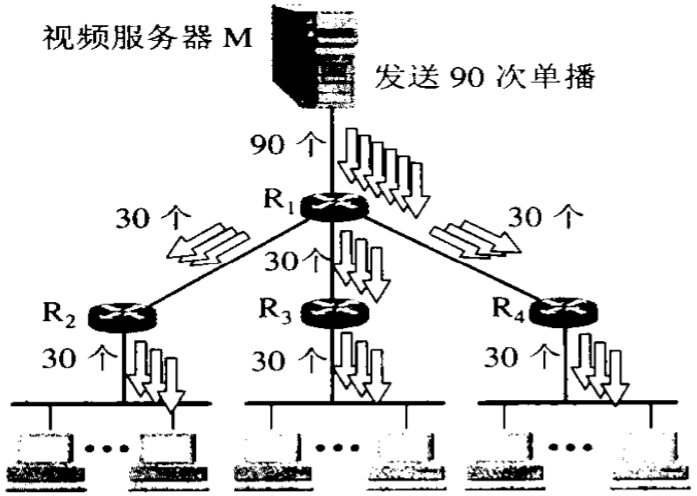
多播与Upnp

多播（或者称组播）

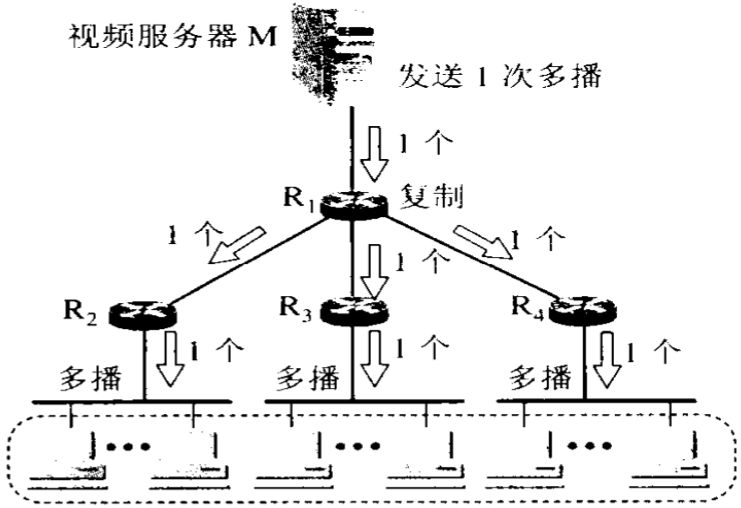
- 多播，一定范围内的IP地址

IP地址划分		
A类	1.0.0.0-127.255.255.255	目前，不再使用A、B、C类划分， 改为，无分类编址CIDR。 如：128.14.35.7/20 表示前20位是网络号，后12位是主机号
B类	128.0.0.0-191.255.255.255	
C类	192.0.0.0-223.255.255.255	
D类	224.0.0.0-239.255.255.255	多播地址
E类	240.0.0.0-254.255.255.255	保留

多播特点



共有 90 个主机接收视频节目



多播组成员共有 90 个

- 多播与单播类似，只是多播包能同时被多台主机接收，其数据包只要发送一次
- 多播地址可共用，不会冲突
- 多播地址只能作目的地址，且只能用于UDP
- 多播因易引起“网络泛洪”，一般路由器都会禁用
- 广播是一种特殊的多播

多播地址分配

多播地址	用途
224.0.0.0	保留
224.0.0.1	本子网上的所有参与多播的主机和路由器
224.0.0.2	本子网上的所有参与多播的路由器
224.0.0.3-224.0.0.255	特殊用途
224.0.1.0-224.0.1.255	协议或应用预留
224.0.2.0-238.255.255.255	公网使用
239.0.0.0-239.255.255.255	内部网使用

特殊IP地址

网络号	主机号	源地址使用	目的地址使用	含义
0	0	可以	不可	在本网络上的本主机
0	host-id	可以	不可	在本网络上的某个主机host-id
全1	全1	不可	可以	只在本网络上广播（路由器不会转发）
net-id	全1	不可	可以	对网络号net-id的所有主机广播
127	非全0 或非全1	可以	可以	本地环路测试用

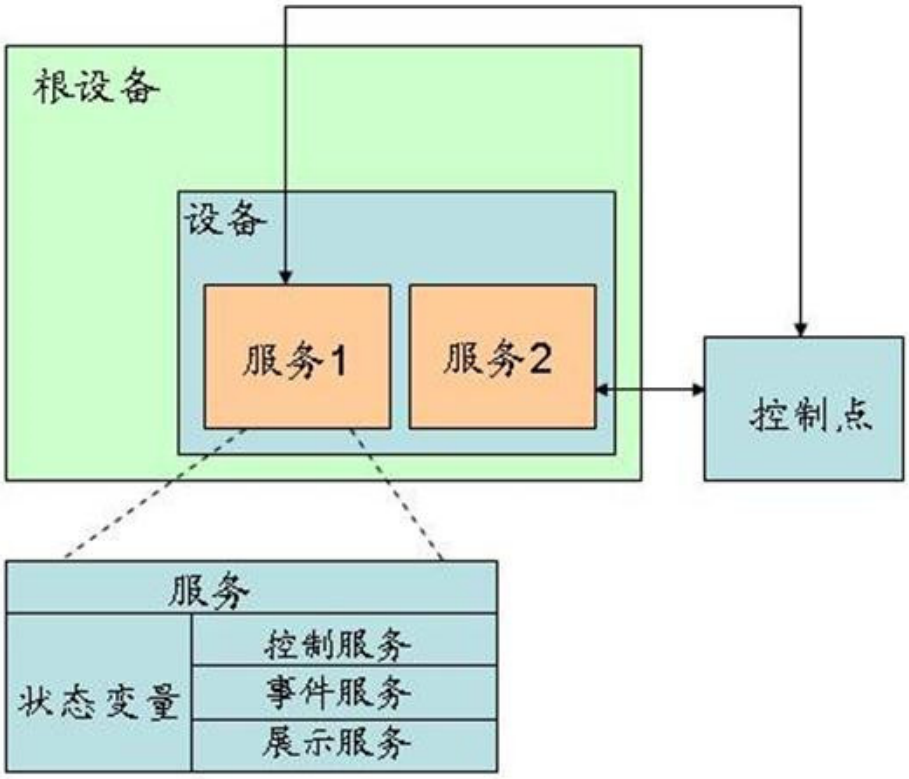
*广播和多播仅应用于UDP

UPNP的简介

- UPnP(Universal Plug and Play,通用即插即用)，微软在1999年提出的一个标准，目的为共享设备和共享资源提供透明访问。
- UPnP ，设备一接入网络就能使用。即任何设备只要一接上网络，所有在网络上的设备就能知道有新设备加入，它们之间就能彼此沟通，能直接使用或控制它。
- Upnp，建立在TCP/IP基础上，利用 XML 进行表述和 HTTP 进行传输。
- 例如，网络打印机

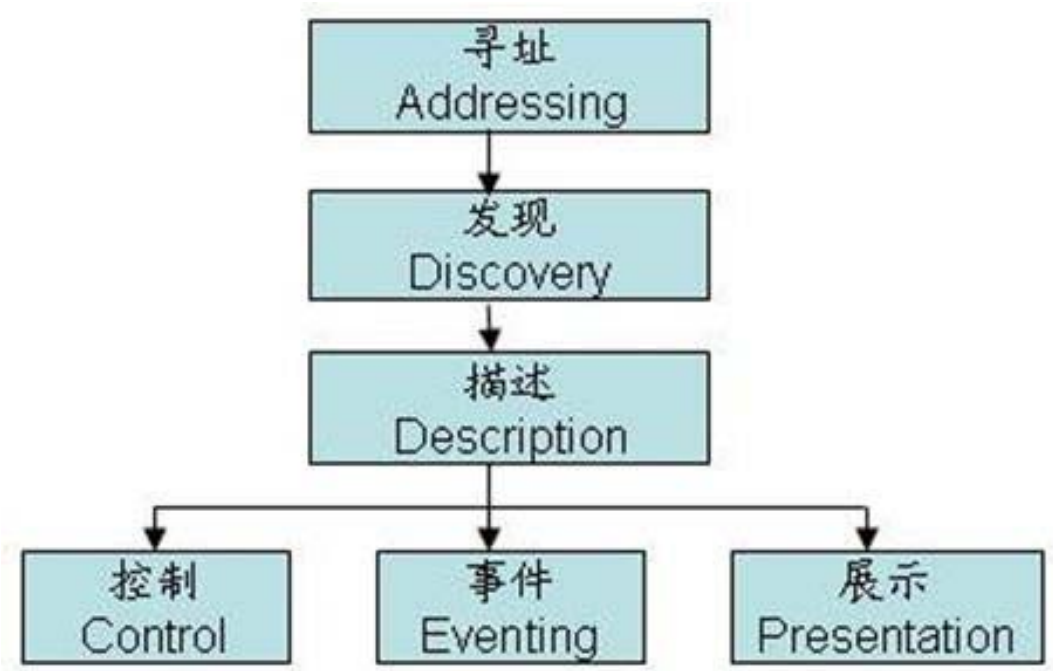
Upnp协议层和组件图

<table>
<tr><td colspan="4">UPnP设备制造商</td></tr>
<tr><td colspan="4">UPnP工作组</td></tr>
<tr><td colspan="4">UPnP设备类</td></tr>
<tr><td>SSDP</td><td>Multicast GENA</td><td>SOAP</td><td>GENA</td></tr>
<tr><td colspan="2">HTTPU/HTTPMU</td><td colspan="2">HTTP</td></tr>
<tr><td colspan="2">UDP</td><td colspan="2">TCP</td></tr>
<tr><td colspan="4">IP</td></tr>
</table>

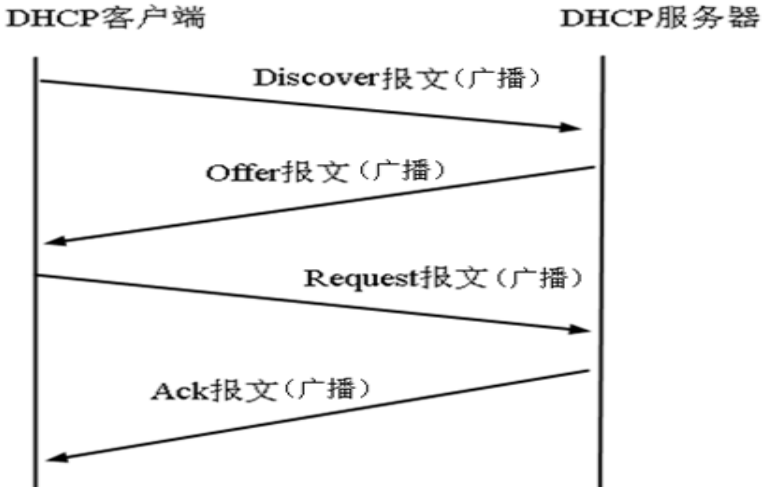


Upnp流程

- 寻址
- 发现
- 描述
- 控制
- 事件
- 展示



设备寻址（即获得IP地址）



- UPnP网络的基础是TCP/IP协议族，所以UPnP设备第一步就是要获得一个IP地址，即设备寻址。
- 通过DHCP服务器获取IP：使用本机地址0.0.0.0，目的地址255.255.255.255，向本地广播，发送DISCOVER消息；而后DHCP以广播方式回应该消息，其目的MAC地址为发送消息主机的MAC地址。
- 若网络上无DHCP服务，使用Auto-IP时：设备在地址范围169.254/16范围中查找空闲的地址。
- 使用Auto-IP的设备必须定时检测DHCP服务器是否存在：若有DHCP回应，设备必须释放Auto-IP分配的地址，而使用新IP。

设备发现(一)

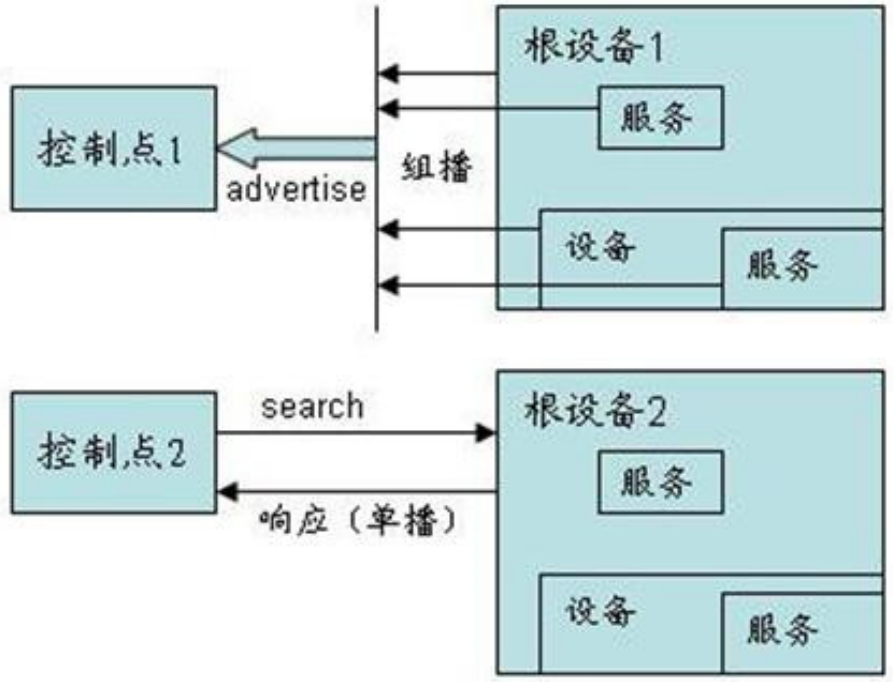
设备发现，使用SSDP（Simple Service Discovery Protocol，简单发现协议），其包格式使用HTTP1.1格式

起始行，为如下三种

NOTIFY * HTTP/1.1

M-SEARCH * HTTP/1.1

HTTP/1.1 200 OK



设备发现方式一：设备加入网络后，定期向组播地址发送ssdp:alive消息，一般周期为30分钟

NOTIFY * HTTP/1.1

Host:239.255.255.250:1900

Cache-control:max-age=1800

Location:http://192.168.0.1:49152/des.xml

Nt:upnp:rootdevice

Nts:ssdp:alive

.....

设备退网时，发送ssdp:bytebyte消息

设备更新时，发送ssdp:update消息

*Location：设备的URL，控制点通过此地址获得设备更详细的信息

设备发现(二)

设备发现方方式二：控制节点加入网络后，寻找自己感兴趣的设备。其也向组播地址发送类似如下消息：

M-SEARCH* HTTP/1.1
Host:239.255.255.250:1900
Man:"ssdp:discover"
Mx:5
ST:ssdp:rootdevice

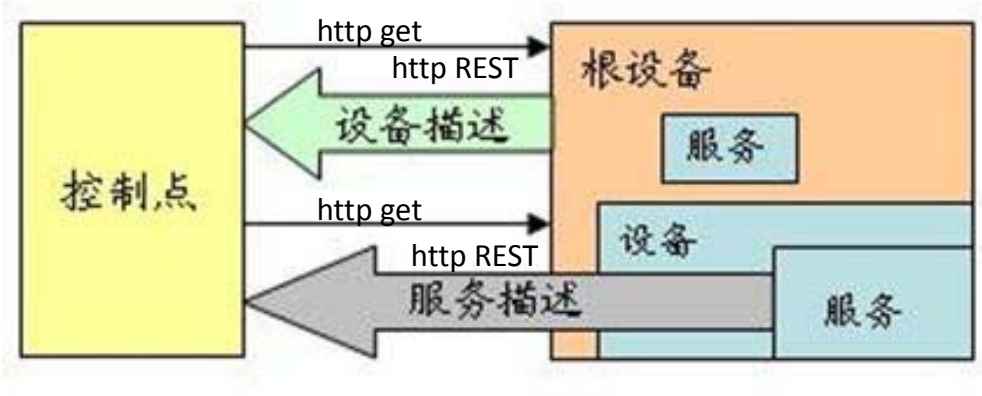
设备回应：

HTTP/1.1200 OK
Cache-control:max-age=1800
Location:http://192.168.0.1:2345/xx.xml
Server:Microsoft-Windows-NT/5.1UPnP/1.0 UPnP-Device-Host/1.0
.....

描述（一）

设备发现后，控制节点对设备知之甚少，通过设备URL来取得设备描述；其通过HTTP协议来完成。获得设备描述后，通过其找到其他URL。

Upnp对设备或服务等的描述，都采用XML方式表示。



设备描述符示例：

```
<?xmlversionxmlversion="1.0" encoding="utf-8"?>
<rootxmlnsrootxmlns="urn:schemas-upnp-org:device-1-0">
 <specVersion>
  <major>1</major>
  <minor>1</minor>
 </specVersion>
 <device>
  <deviceType>urn:schemas-upnp-org:device:BinaryLight:1</deviceType>
   ......
   <serviceList>
    <service>
     <serviceType>urn:schemas-upnp-org:service:SwitchPower:1</serviceType>
     <serviceId>urn:upnp-org:serviceId:SwitchPower:1</serviceId>
     <SCPDURL>/SwitchPower1.xml</SCPDURL>
      <controlURL>/SwitchPower/Control</controlURL>
     <eventSubURL>/SwitchPower/Event</eventSubURL>
    </service>
   </serviceList>
   <presentationURL>/SwitchPower/Present</presentationURL>
 </device>
</root>
```

通过设备描述符可获得其他的
*SCPDURL：服务描述的URL
*controlURL：用于控制的URL
*eventSubURL：用于订阅事件的URL
*presentationURL：用于展示的URL

描述（二）

服务描述符示例：

```
<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
 <specVersion>
  <major>1</major>
  <minor>1</minor>
 </specVersion>
 <actionList>
   <action>
    <name>SetTarget</name>
     <argumentList>
      <argument>
       <name>NewTargetValue</name>

<relatedStateVariable>Target</relatedStateVariable>
       <direction>in</direction>
      </argument>
     </argumentList>
   </action>
   <action>
    <name>GetTarget</name>
     <argumentList>
      <argument>
       <name>RetTargetValue</name>

<relatedStateVariable>Target</relatedStateVariable>
       <direction>out</direction>
      </argument>
     </argumentList>
   </action>
   <action>
     <name>GetStatus</name>
      <argumentList>
       <argument>
        <name>ResultStatus</name>
        <relatedStateVariable>Status</relatedStateVariable>
        <direction>out</direction>
       </argument>
      </argumentList>
    </action>
  </actionList>
  <serviceStateTable>
   <stateVariable sendEvents="no">
    <name>Target</name>
    <dataType>boolean</dataType>
    <defaultValue>0</defaultValue>
   </stateVariable>
   <stateVariable sendEvents="yes">
    <name>Status</name>
    <dataType>boolean</dataType>
     <defaultValue>0</defaultValue>
   </stateVariable>
  </serviceStateTable>
</scpd>
```

actionList定义了“动作”：

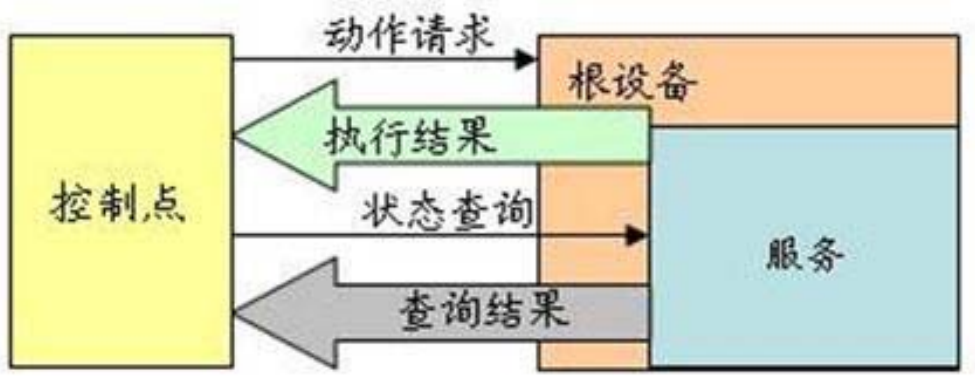
- 名称（name）
- 若干参数（argument）组成。
  - 参数名（name）
  - 传递方向（direction：取值in或者out）
  - 关联的状态变量（relatedStateVariable）

serviceStateTable定义了“状态变量”：一个控制点可以订阅（subscribe）状态的变化，从而得到通知。

- 名称（name）
- 数据类型（dataType）

控制（一）

在接收设备和服务描述之后，控制点可以向这些服务发出动作请求或者状态查询消息。消息使用以XML描述的SOAP格式。



SOAP（Simple Object Access Protocol）简单对象访问协议

SOAP消息格式，使用XML：

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:Header>
.....
</soap:Header>

<soap:Body>
<soap:Fault>
.....
</soap:Fault>
</soap:Body>
</soap:Envelope>
```

SOAP包含以下元素：

- 必需的 Envelope 元素
- 可选的 Header 元素，包含头部信息
- 必需的 Body 元素，包含所有的调用和响应信息
- 可选的 Fault 元素，提供有关在处理此消息所发生错误的信息

一些重要的语法规则：

- SOAP 消息必须用 XML 来编码
- SOAP 消息必须使用 SOAP Envelope 命名空间
- SOAP 消息必须使用 SOAP Encoding 命名空间
- SOAP 消息不能包含 DTD 引用
- SOAP 消息不能包含 XML 处理指令

控制（二）

Upnp SOAP请求格式：

POST /control/url HTTP/1.1
HOST: hostname:portNumber
CONTENT-TYPE: text/xml;charset="utf-8"
CONTENT-LENGTH: length ofbody
USER-AGENT: OS/versionUPnP/1.1 product/version
SOAPACTION:"urn:schemas-upnp-org:service:serviceType:v#actionName"

```
<?xml version="1.0"?>
<s:Envelope  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <s:Body>
      <u:actionName xmlns:u="urn:schemas-upnp-
org:service:serviceType:v">
         <argumentName>in arg value</argumentName>
      </u:actionName>
   </s:Body>
</s:Envelope>
```

*actionName和argumentName就是动作名称和参数名称。

Upnp SOAP响应格式：

HTTP/1.1 200 OK
CONTENT-TYPE: text/xml;charset="utf-8"
DATE: when response wasgenerated
SERVER: OS/version UPnP/1.1product/version
CONTENT-LENGTH: bytes inbody

```
<?xmlversionxmlversion="1.0"?>
<s:Envelope  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <s:Body>
      <u:actionNameResponse xmlns:u="urn:schemas-upnp-
org:service:serviceType:v">
         <argumentName>out arg value</argumentName>
      </u:actionNameResponse>
   </s:Body>
</s:Envelope>
```

事件

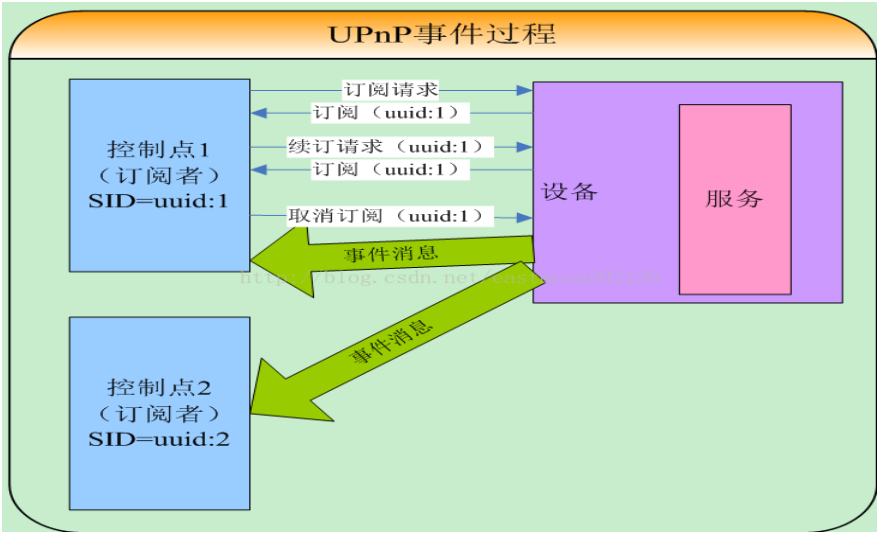
在服务期间，变量值发生了变化，就会产生了一个事件。事件服务器根据配置要求把事件向整个网络进行广播。

另一方面，控制点也可以事先向事件服务器订阅事件信息，使得该控制点感兴趣的事件及时准确地传送过来。

事件消息使用以XML描述的GENA格式。

订阅请求格式示例：

SUBSCRIBE *publisher path* HTTP/1.1
HOST: *publisherhost:publisher port*
USER-AGENT: *OS/version* UPnP/1.1 *product/version*
CALLBACK: *<deliveryURL>*
NT: upnp:event



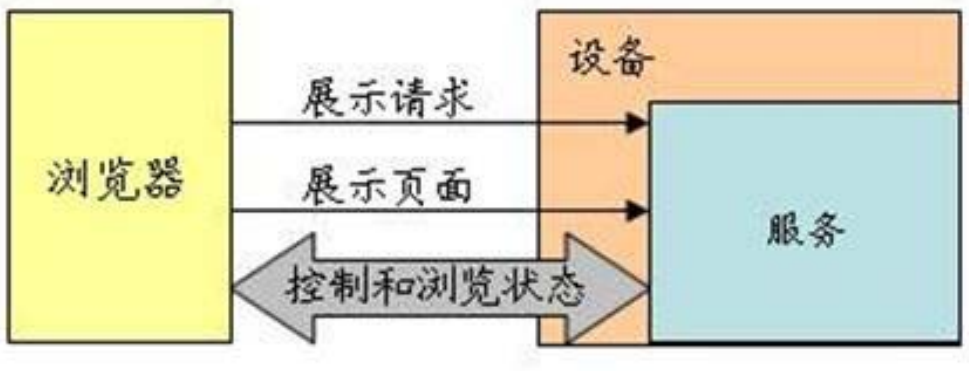
事件消息格式示例：

NOTIFY *delivery path* HTTP/1.1
HOST: *delivery host:delivery port*
CONTENT-TYPE: text/xml; charset="utf-8"
NT: upnp:event
NTS: upnp:propchange
SID: uuid:*subscription-UUID*
SEQ: *event key*
CONTENT-LENGTH: *bytes in body*

```
<?xml version="1.0"?>
<e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-0">
  <e:property>
    <variableName>new value</variableName>
  </e:property>
</e:propertyset>
```

展示

展示就是一个Web服务器，控制点使用浏览器浏览或控制设备。其URL在设备描述中。



XML简介

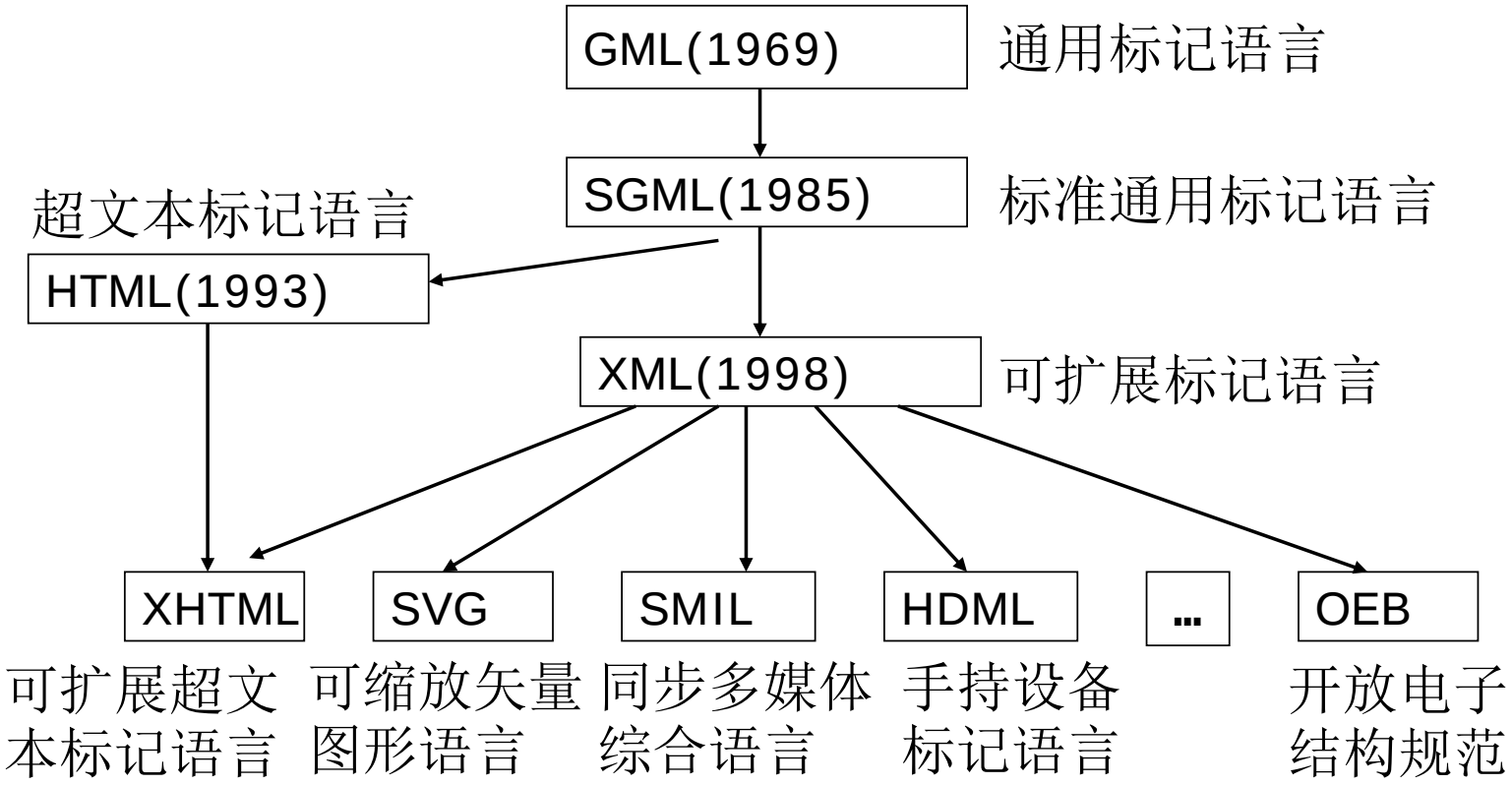
一个XML实例

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<resume>
    <name>朱元璋</name>
    <preName>朱重八</preName>
    <gender>男</gender>
    <occupation>皇帝</occupation>
</resume>
```

XML特点

- XML（eXtensible Markup Language），即可扩展标记语言
- 一种标记语言，用来描述数据，着重在“数据是什么”
- 结构化数据
- 简单、自由、可扩展
- 面向对象的特性（有属性）
- 数据与格式分离，阅读、维护方便
- 易于程序间数据交互

SGML语言发展


GML(1969)
通用标记语言
SGML(1985)
标准通用标记语言
超文本标记语言
HTML(1993)
XML(1998)
可扩展标记语言
XHTML
SVG
SMIL
HDML
...
OEB
可扩展超文本标记语言
可缩放矢量图形语言
同步多媒体综合语言
手持设备标记语言
开放电子结构规范

XML 与 HTML对比

比较内容	HTML	XML
可扩展性	不具有扩展性	可用于定义新的标记语言
侧重点	如何显示数据	数据的结构化组织
语法要求	不要求标记的嵌套、配对等，不要求标记之间具有一定的顺序	严格要求嵌套、配对，须遵循DTD或Schema要求
可读性及可维护性	难于阅读、维护	结构清晰，便于阅读、维护
数据和显示的关系	内容描述与显示方式整合为一体	内容描述与显示方式相分离
大小写敏感性	不区分大小写	区分大小写
标记集合	有固定标记，如 	无固定标记，自定义

XML结构

```
[1]<?xml version="1.0" encoding="utf-8" standalone="yes"?>
[2]<?xml-stylesheet type="text/css" href="books.css"?>
[3]<books>
[4]    <book category="通信">
[5]        <title lang="中文">通信原理</title>
[6]        <author>李四</author>
[7]    </book>
[8]    <book category="计算机">
[9]        <title lang="中文">XML原理</title>
[10]       <author>张三</author>
[11]   </book>
[12]</books>
[13]<!-- 一个XML的例子 -->
```

属性

注释

- XML文档由三大部分组成
  - 序言（可选）：第1-2行

    XML声明、注释、处理指令
  - 主体：第3-12行

    有且仅有一个根元素
  - 尾声（可选）：第13行

    注释、处理指令

XML声明

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

- XML声明一般是XML文档的第一行
- XML声明由以下几个部分组成：
  - version - -文档符合XML1.0规范，目前有1.0、1.1版本，大多还只支持1.0
  - encoding（可选） - -文档字符编码，默认为"UTF-8"
  - standalone（可选） - -文档定义是否在一个文件内（是否引用外部实体）
    - standalone="yes"
    - standalone="no"

XML处理指令

<?xml-stylesheet type="text/css" href="books.css"?>

- 语法：<?*处理指令名称　参数*?>
  - 以“<?”开头
  - 以“?>”结尾
  - <?xml ?>和<?XML ?> 被保留
- XML解析器会把处理指令原封不动传递给XML应用程序
  - 处理指令如何解释由外部应用程序决定（可以忽略或传给其他程序）
  - 处理指令可以出现在文中任何地方
  - 例，自行开发一个XML应用程序，处理<?topdf path=”/pdf-file” ?>
- <?xml-stylesheet ?>
  - 其可被大多数Web浏览器解释
  - 样式表处理指令，必须出现在序言中，即根元素之前

XML注释

<!-- 一个XML的例子 -->

- 语法：<?*处理指令名称　参数*?>
  - 以“<!--”开头
  - 以“-->”结尾
- 注释内容中不要出现--；
- 不要把注释放在标记中间；
  - <Name <!--the name-->>张三</Name>
- 注释不能嵌套；
- 不能放在XML声明之前

XML元素

- 元素的形式：**<标记>字符数据</标记>**
- 元素中还可以再嵌套别的元素
- 空元素：<TITLE/>
- 所有的XML文件都**至少包含一个形式良好的根元素**（即不能说非空标记）。根元素，又称为文件标记。
- XML中开始和结束标记之间的文字称作“字符数据”，而把标记内的标示文字称作“标记”。

XML属性

- 属性的形式：**<标记名 属性名1="属性值1" 属性名2="属性值2" />**
- 一个元素可以有多个属性
- 一个元素标记中，同属性名称只能出现一次。

XML命名空间

- 命名空间可以看作一个标记名前缀。
- 命名空间用来防止标记名冲突。

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
  <td>Apples</td>
  <td>Bananas</td>
  </tr>
</table>
<f:table xmlns:f="http://www.w3school.com.cn/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

命名空间语法：

xmlns:*namespace-prefix*="*namespaceURI*"

默认命名空间：

xmlns="*namespaceURI*"

```
<d:student xmlns:d="http://www.develop.com/student">
```

- student 是标记名
- xmlns关键字用于声明命名空间绑定关系
- http://www.develop.com/student命名空间的标识。
- d命名空间前缀，其实际代表http://www.develop.com/student。

XML特殊字符数据（一）

- 字符部分不能包含特殊用途的字符
  - <、>、&、”、'
  - 使用
- 例子：XML 原理&应用
  - 非法：<title>XML 原理&应用<title/>
  - 正确：<title>XML 原理&应用<title/>

预定义实体	符号
<	<
>	>
&	&
"	"
'	'

XML特殊字符数据（二）CDATA标记

- 用CDATA表示特殊字符
  - 语法形式：<![CDATA[文本内容]]>
- 在CDATA标记下，所有字符当作简单字符处理
- 例子：XML 原理&应用
  - <title><![CDATA[XML 原理&应用]]><title/>
- CDATA标记中不能包含“]]”
- CDATA标记不能嵌套

XML相关基本技术

- XML文档格式约束、校验： DTD和Schme
- 格式化XML： CSS
- 文档转化： XSLT
- 路径寻址： XPath
- 解析： DOM、SAX、PULL（android）

JSON简介

JSON数据结构

- 无序键值对集合
  { “a”:1, ”b”:2, ”c”:3 }
- 数组（有序列表）
  [1, 2, 3, ”hello world”]
- 值类型
  - 字符串：双引号包含内容，“hello world”
  - 数值：
  - 布尔：true、false
  - 空：null
  - 数组、键值对：即可互相嵌套

```
{
   "indexes" : [5, 10, 15],
   "names" : [张三, 李四, 刘某],
   "备注" :  [ {"abc":123 },  true,  null ]
 }
```

JSON优点：
格式简洁、体积小、更利于数据交换