Service oriented Middleware for Iam « FIND » Layer

SOM, based on ROA or SOA Approaches

Reference : Service-oriented middleware: A survey Jameela Al-Jaroodi, Nader Mohamed, Journal of Network and Computer Applications, Volume 35, Issue 1, January 2012, Pages 211–220, Collaborative Computing and Applications

Middleware and Service oriented Concepts

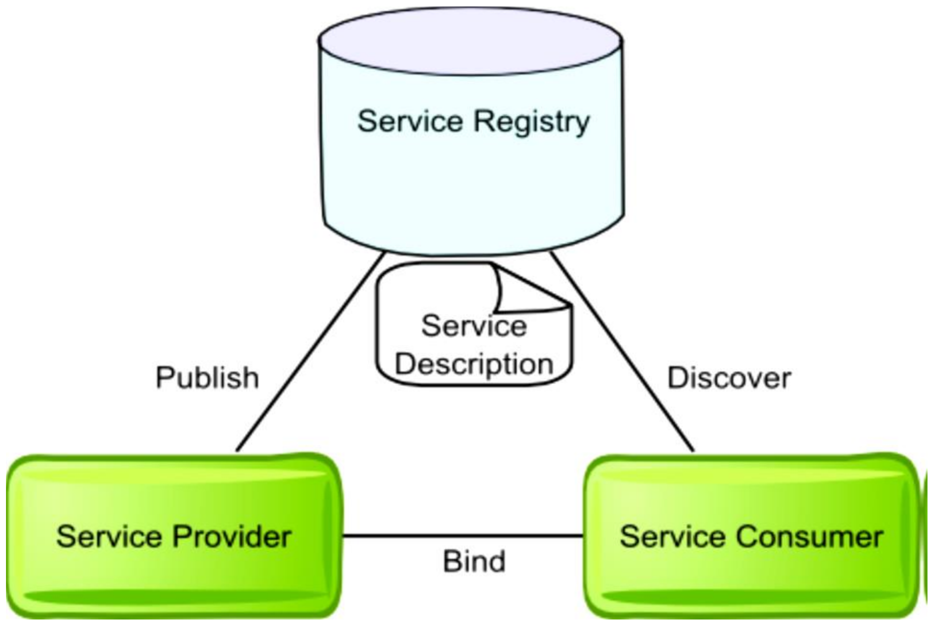
- **Service-oriented Middleware*** is a kind of middleware based on the Service Oriented Architecture (SOA) paradigm that supports the development of distributed software systems in terms of loosely coupled networked services.
- In SOA, networked resources are made available as autonomous software services that can be accessed without knowledge of their underlying technologies.
- Key feature of SOA is that services are independent entities, with well-defined interfaces, which can be invoked in a standard way, without requiring the client to have knowledge about how the service actually performs its tasks.

(*) A Perspective on the Future of Middleware-based Software Engineering, Valérie Issarny, Mauro Caporuscio, Nikolaos Georgantas, Workshop on the Future of Software Engineering : FOSE 2007, 2007, Minneapolis, United States. pp.244-258, 2007, https://hal.inria.fr/inria-00415919

Challenges for the « FIND » Layer

Middleware and Service oriented Concepts

- The SOA style is structured around three key architectural components: (i) service provider, (ii) service consumer, and (iii) service registry
- In SOA-based environments, the Service-Oriented Middleware (SOM) is in charge of enabling the deployment of services and coordination among the three key conceptual elements that characterize the SOA style.
- Popularity of service oriented computing is mainly due to its **Web Service** instantiation.



Trends Web of Things or Web Service for Device

- Two kind of Approches

- Service oriented Architectures :
  - ROA (DAO) : Ressource or data oriented
    - Commnication pattern between service consumer and provider is based on shared URL
    - Principle : Ressources as URL like hyperlinks in a classical Web approach
  - SOA : Service oriented
    - Communication pattern between service consumer and provider is RPC
    - Principle : RPC using SOAP protocol over HTTP

Ressource Oriented Architecture

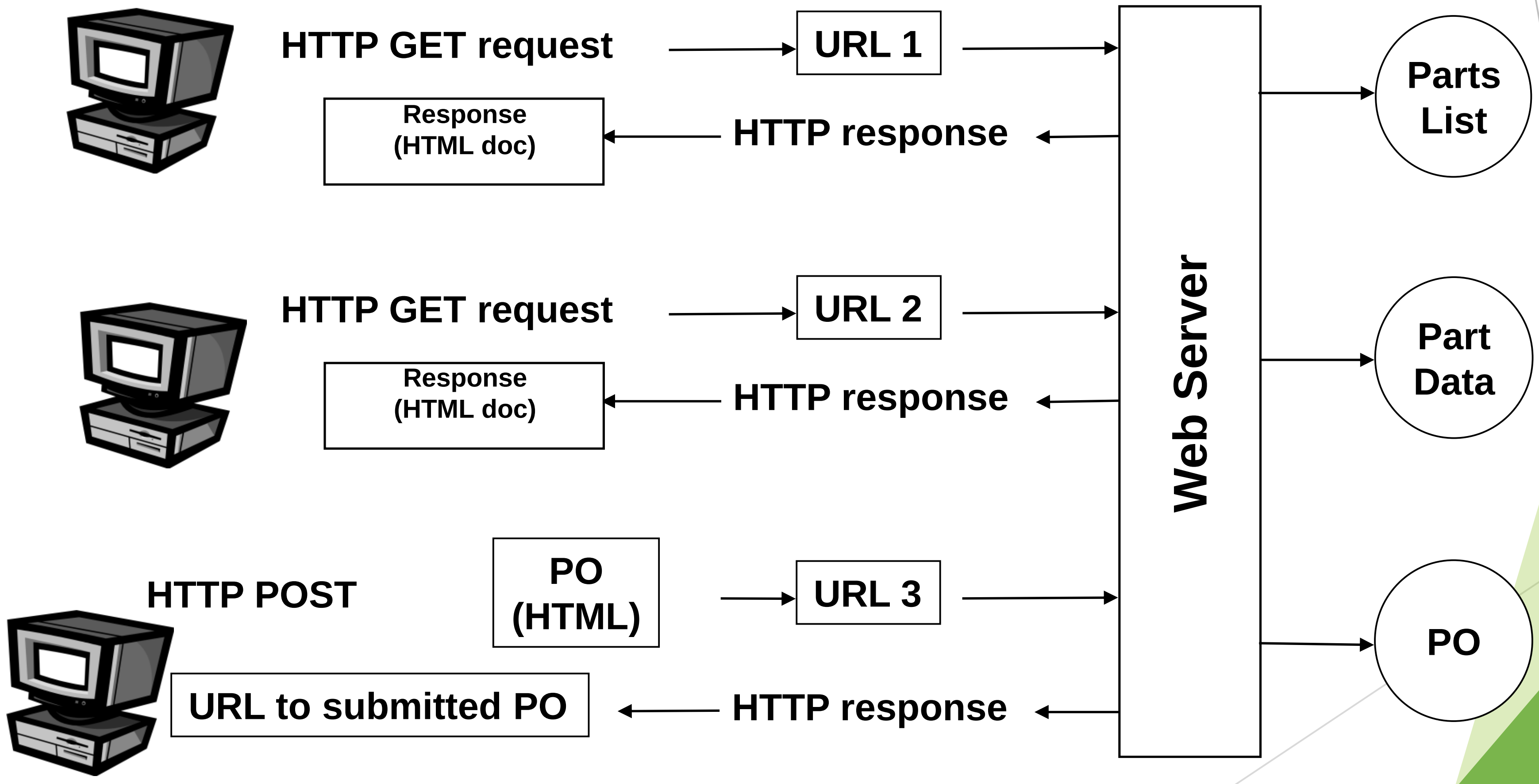
RESTful Web Services

- REpresentational State Transfer
    - Architecture inherent in all web based system since 1994, not explicitly described as an architecture until later
    - An architecture - not a set of standard
        - Web Services is both an architecture and a set of standards
- Goal: To leverage web based standards to allow inter-application communication as simply as possible
    - Matches the ‘standard’ web interaction model
    - Ressources as URL like hyperlinks in a classical Web approach

REST architecture

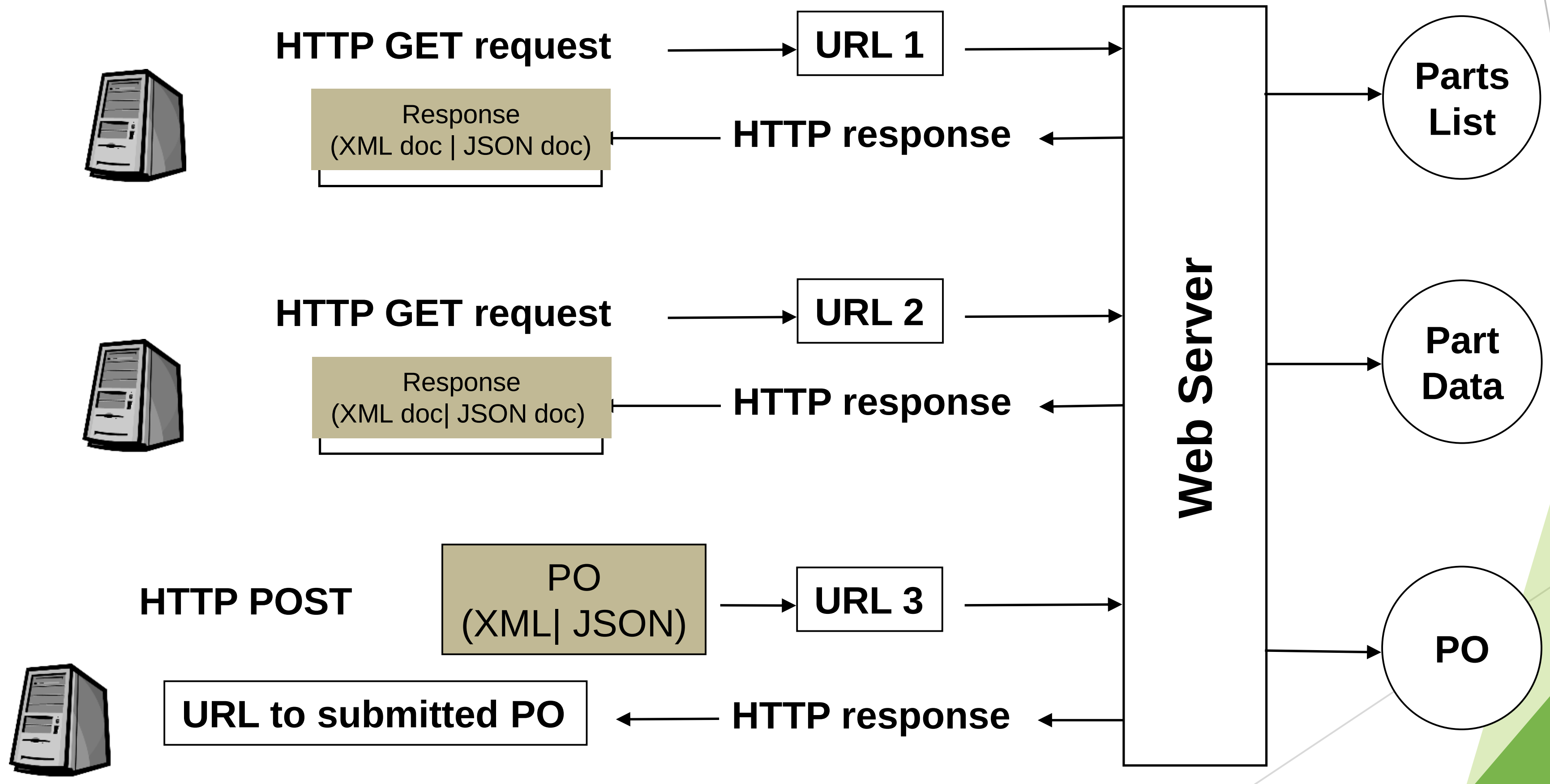
- Uses HTTP operations:
  - GET = "give me some info" (Retrieve)
  - POST = "here's some update info" (Update)
  - PUT = "here's some new info" (Create)
  - DELETE = "delete some info" (Delete)
- Typically exchanges XML documents
  - But supports a wide range of other internet media types
- Example of client side REST request: GET /shoppingcart/5873
  - Server must be able to correctly interpret the client request as there is no explicitly defined equivalent to an interface definition

The standard Web architecture



Copyright © [2005]. Roger L. Costello, Timothy D. Kehoe.

The RESTful architecture



Copyright © [2005]. Roger L. Costello, Timothy D. Kehoe.

REST Architecture

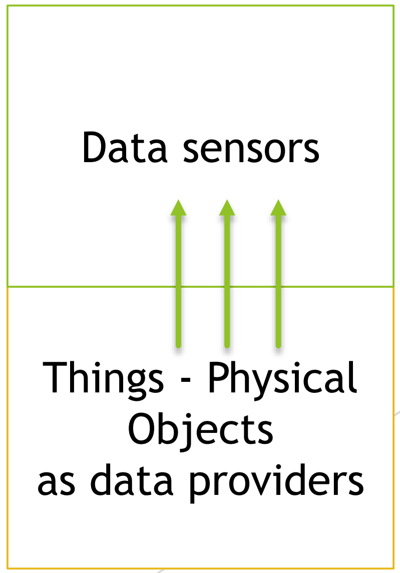
- Servers are stateless and messages can be interpreted without examining history
    - Messages are self-contained
- There is no such thing as a “service”.
    - There are just resources which are accessed through URI
        - URI = generalisation of URL
- Clients navigate through a series of steps towards a goal by following hypertext links (GET) and submitting representations (POST).

ROA and Mashup

- Mashups is “A way to create new Web applications by combining existing Web resources utilizing data and Web APIs” [Benslimane et al., 2008]

- ROA is Well-adapted for Mashups (Composite Web Applications)
- Well-adapted for Web Sensors Network (WSN)

- But lacks for non sensor device … like actuators …



REST – strong versus weak

- Pure REST should use ‘pure’ URI only
  - E.g. GET /shoppingcart/5873
- Many REST implementations also allow parameter passing
  - E.g. GET /shoppingcart/5873?sessionID=123
- Allowing parameter passing makes REST a lot more usable but blurs the architectural principle of statelessness
- Indeed Data can be specific command like instruction code …
  - But is it the purpose ?
  - Is this not another way to rebuild a SOA stack ?

Service oriented architecture (SOAP-WS)

SOA : Service oriented Architecture

- A service provides business functions to its consumer and in ISO 19119 [ISO/TC-211] it is defined as
- “ Distinct part of the functionality that is provided by an entity through interfaces ”.
- Also called WS-* (for * recommendations, Cf. http://www.w3.org/)

- SOAP based Web Service, the alternative
- RPC using SOAP protocol over HTTP

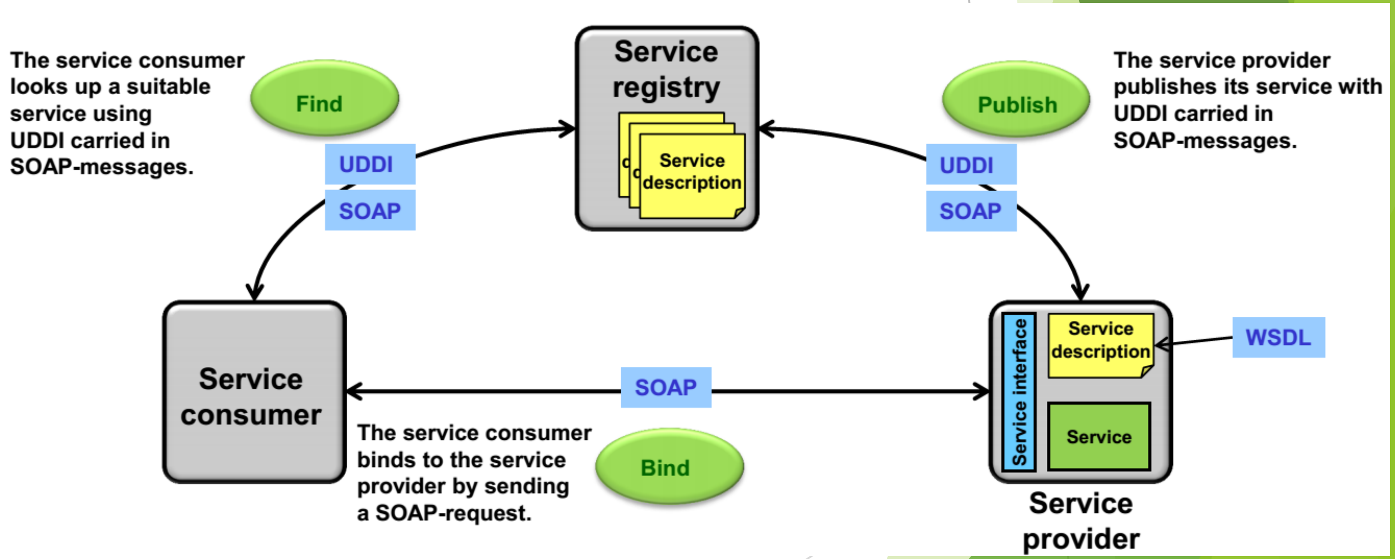
Sample SOAP RPC Message

- <Envelope> est la racine
- <Header>, <Body> et <Fault> sont les enfants :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <soap:Header>
    ...  Header information...
    </soap:Header>
    <soap:Body>
    ... Body information...
    <soap:Fault>   ...Fault information...
    </soap:Fault>
</soap:Body>
</soap:Envelope>
```

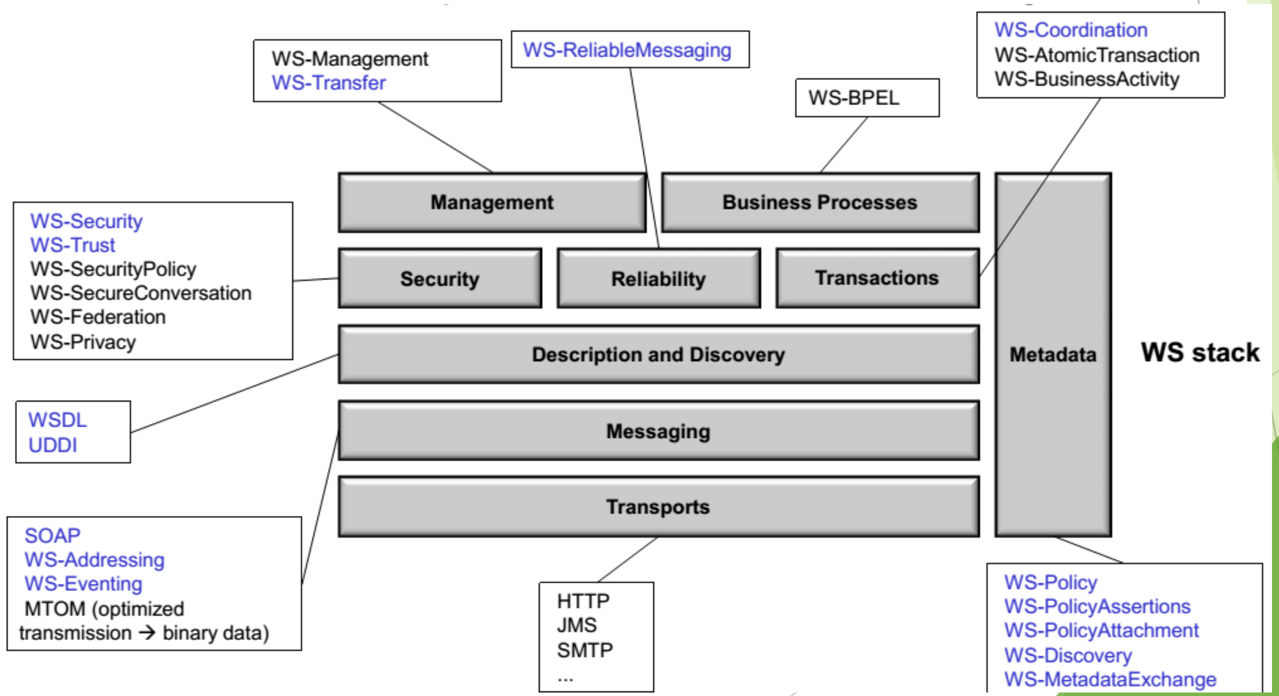
WS-*architecture more than ROA

- SOAP+WSDL+UDDI defines a general model for a web service architecture.
  - SOAP: Simple Object Access Protocol
  - WSDL: Web Service Description Language
  - UDDI: Universal Description and Discovery Protocol
  - Service consumer: User of a service
  - Service provider: Entity that implements a service (=server)
  - Service registry : Central place where available services are listed and advertised for lookup


The service consumer looks up a suitable service using UDDI carried in SOAP-messages.
Find
UDDI
SOAP
Service registry
Service description
Publish
UDDI
SOAP
The service provider publishes its service with UDDI carried in SOAP-messages.
Service consumer
SOAP
The service consumer binds to the service provider by sending a SOAP-request.
Bind
Service interface
Service description
Service
WSDL
Service provider

WS-* Models

- Stack of WS-standards
- The W3C and OASIS WS-stack provide a framework / toolbox for constructing web service architectures



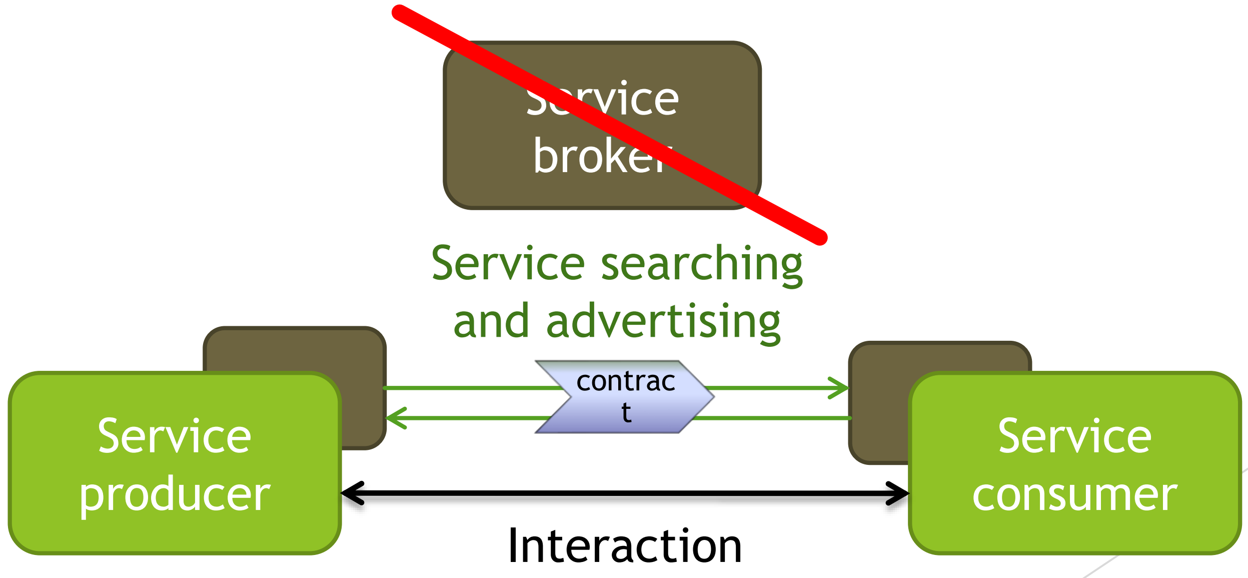
Disadvantages of Web Services

- Low-level abstraction
  - leaves a lot to be implemented
- Interaction patterns have to be built
  - one-to-one and request-reply provided
  - one-to-many?
- No location transparency

Challenges for Dynamicity in the « FIND » Layer

Dynamicity

- Distributed dynamic Research and Discovery
  - Appearance and Disappearance management
  - Allow contextual research and discovery

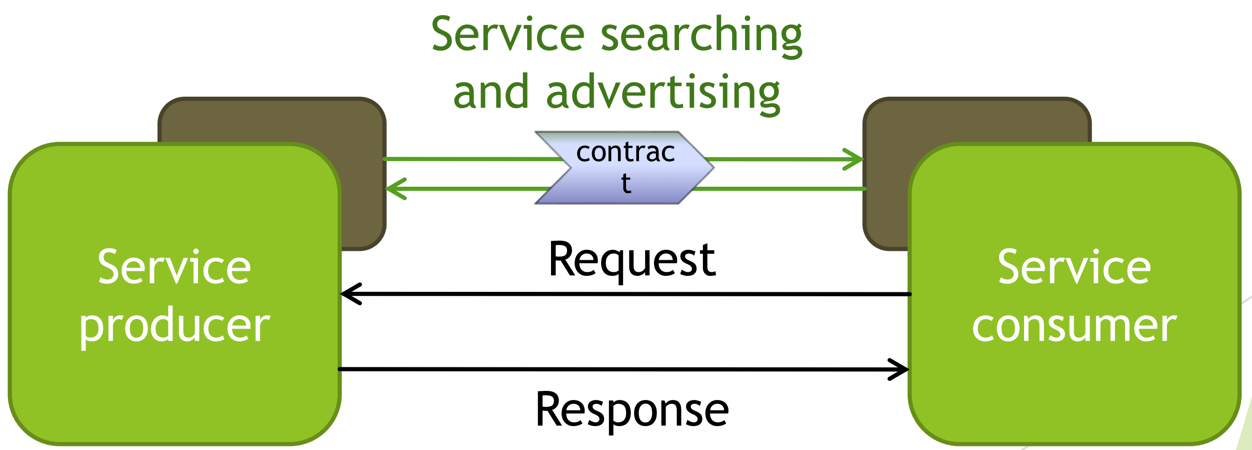


Service Discovery Protocols

- Multicast DNS/DNS-SD: Apple’s protocol
  - Multicast DNS: uses API similar to unicast DNS
- SLP: IETF proposed standard
  - Supported by HP, Novell, Sun Microsystems, Oracle
- SSDP: Microsoft’s protocol
  - Uses HTTP notifications (see bellow), used since XP
- WS-Discovery: Defined by OASIS
  - Latest defined protocol, used in DPWS (see bellow)

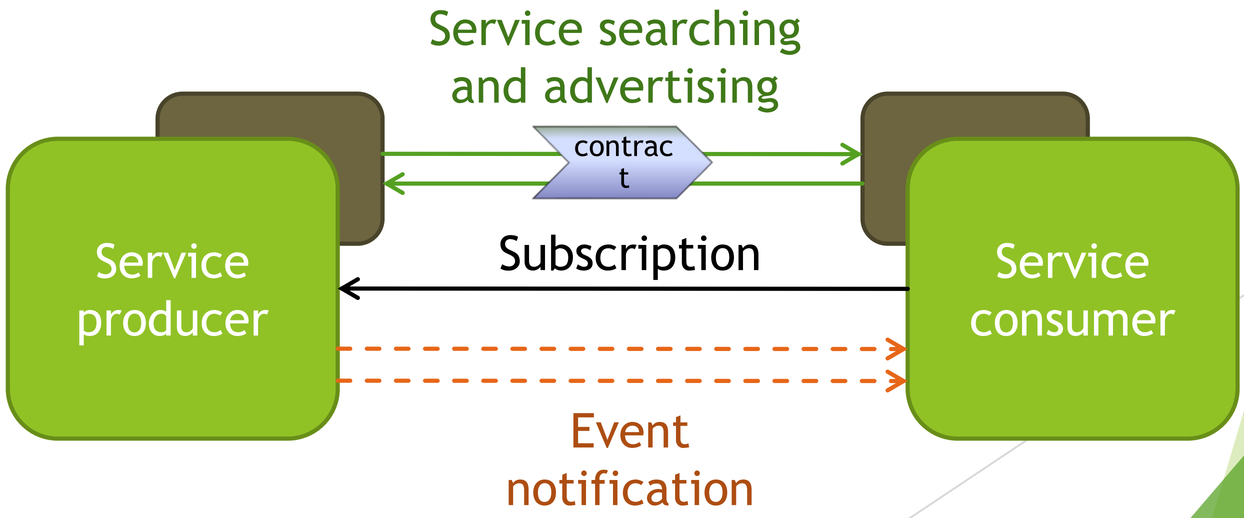
Traditional Interactions: Invocations

- "Classical" way to interact between services
  - Request-Response mechanism



Reactivity

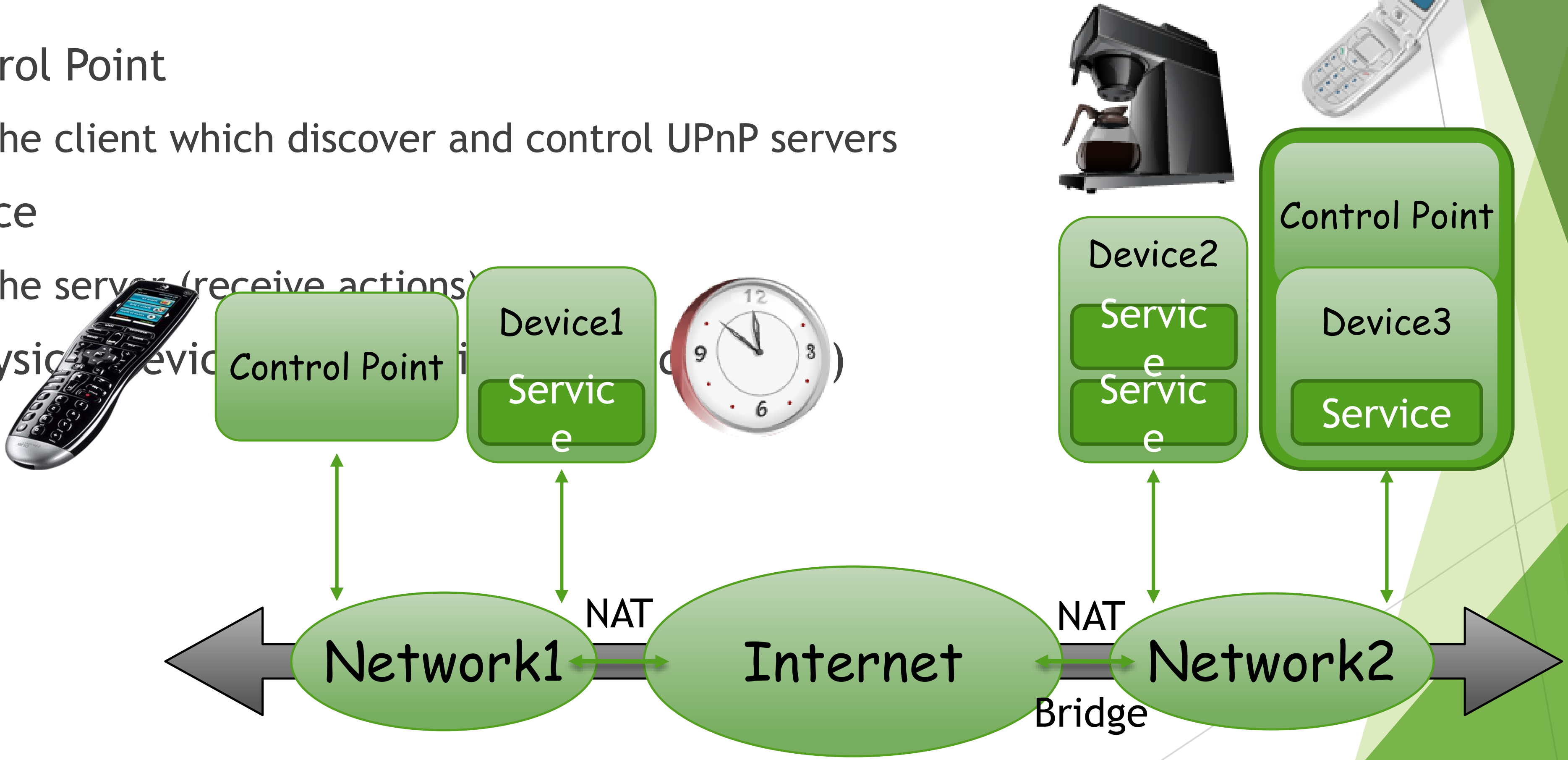
- "New way" of interacting: Eventing interaction model
  - Based on publish/subscribe design pattern
  - Asynchronous messaging (based on push mode)



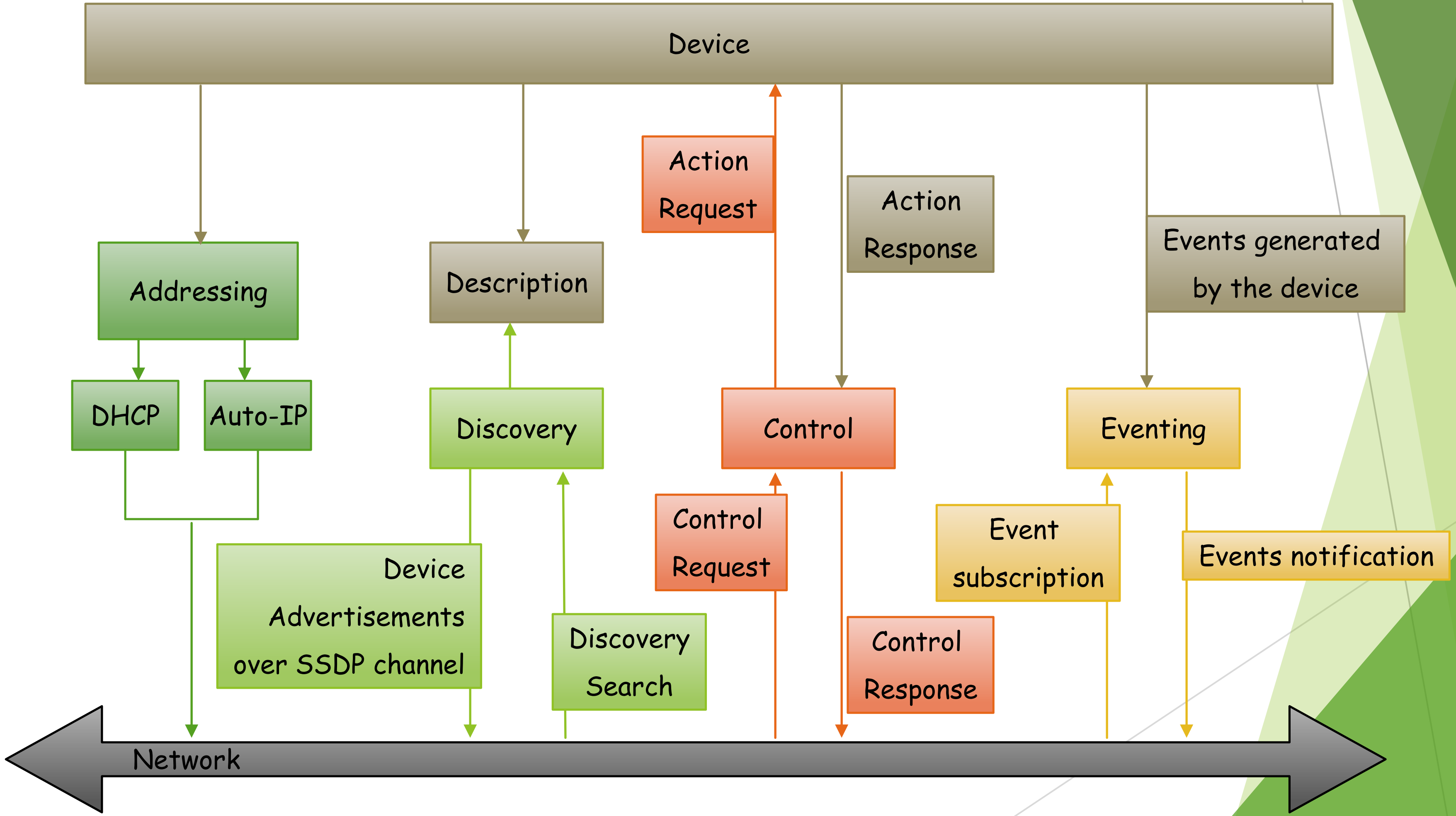
Example of Technologies on Device : UPnP & dPWS

UPnP : Universal Plug and Play

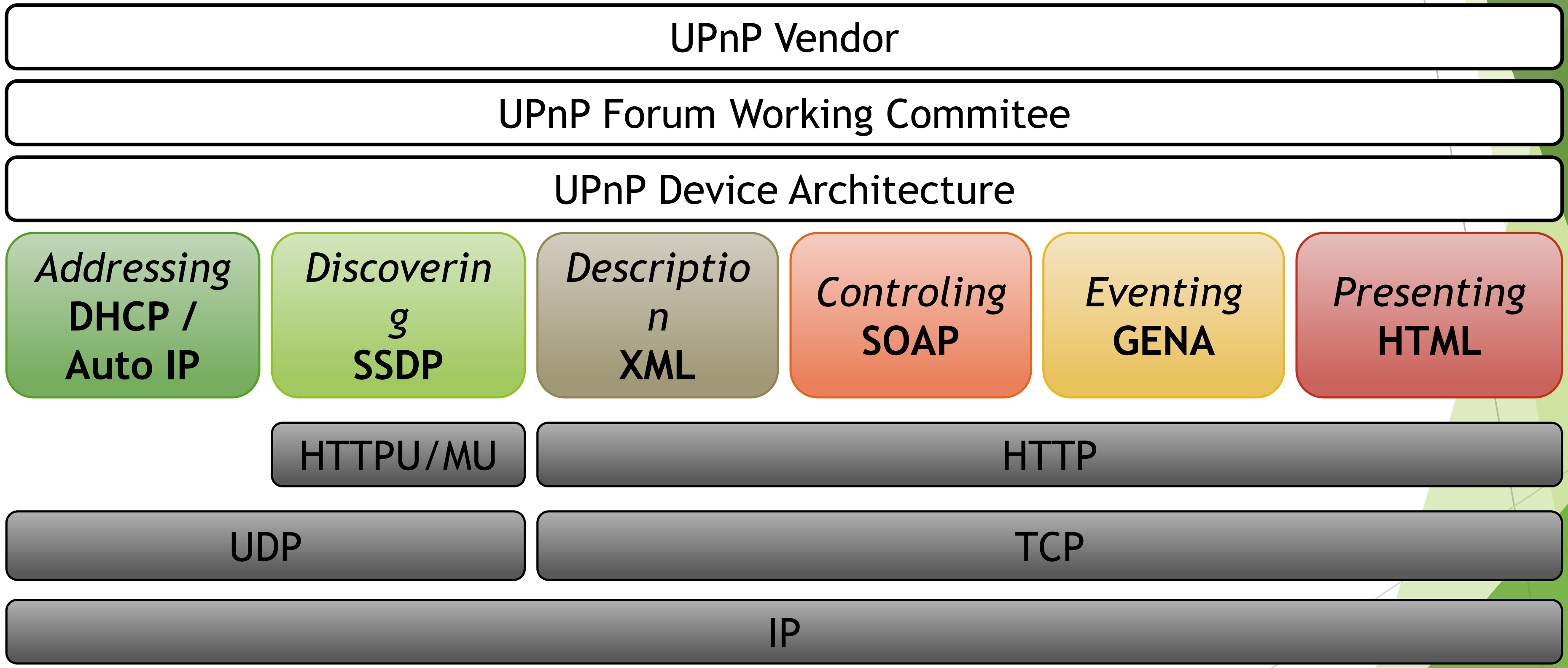
- Control Point
  - The client which discover and control UPnP servers
- Device
  - The serv (receive actions)
- A physic evic i c)



Example of UPnP Device Communications



UPnP Stack and Protocols

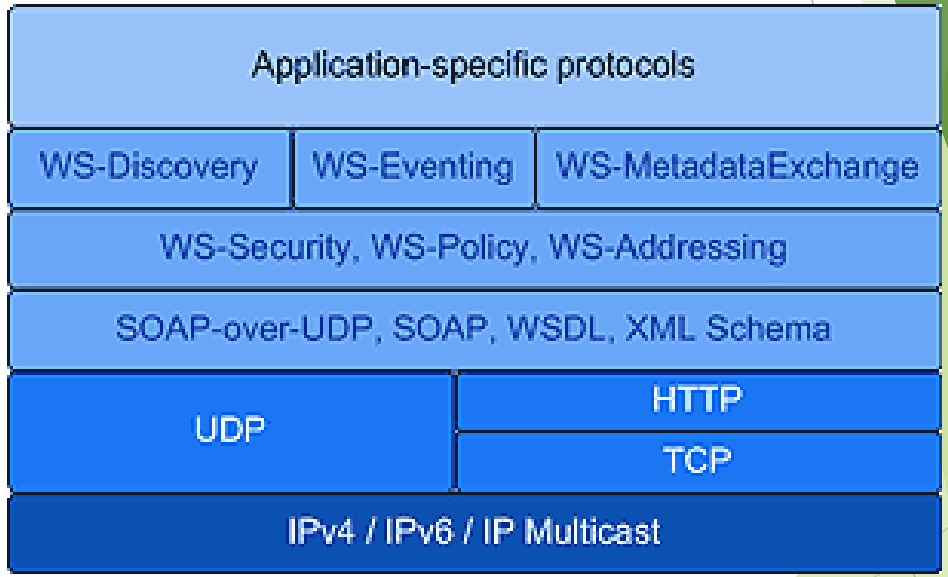


DPWS : Device Profile for Web Services

- DPWS[1]: Same goal as UPnP (UPnP v2)
  - But without backward compatibility
- Using or defining standards
  - WSDL, WS-Discovery, WS-Eventing, …
- Approved as OASIS standard on 30, june 2009
- All or some parts of DPWS already included in Vista, Micro .NET, Windows CE, …

DPWS Stack and Protocols

- Only based on standards
  - SOAP 1.2,
  - XML,
  - XML Schema,
  - WSDL 1.1,
  - WS-Addressing,
  - WS-Transfer,
  - WS-Policy,
  - WS-Security,
  - WS-MetadataExchange,
  - WS-Discovery
  - WS-Eventing


Application-specific protocols
WS-Discovery
WS-Eventing
WS-MetadataExchange
WS-Security, WS-Policy, WS-Addressing
SOAP-over-UDP, SOAP, WSDL, XML Schema
UDP
HTTP
TCP
IPv4 / IPv6 / IP Multicast

DPWS implementations emerged with the help of Research Projects

- European Research Initiative ITEA
  - SIRENA project (2003-2005)
    - SOA4D: SOA for Devices (Java and C Stack)
    - WS4D: Web Services for Devices (Java, Java ME and C Stack)
  - SODA project (Service Oriented Device and Delivery Architecture) (2006-2008)
- EU Research Project
  - SOCRADES (2006-2009) composed by heavyweights like ABB, SAP, Schneider Electric, and Siemens

Using DPWS

- Also Microsoft implementations
  - Micro .NET framework
  - .NET framework (.NET 4.0)
  - Included since Vista (WSDAPI)
- But…
  - For the moment, the 3 main implementations (SOA4D, WS4D, Microsoft) of DPWS do not communicate with other ones…
  - So everybody is a standard !

UPnP vs DPWS

	UPnP	DPWS
Addressing	DHCP, AutoIP	DHCP, AutoIP
Discovery	SSDP	WS-Discovery
Description	UDA Schema	WSDL 1.1
Control	SOAP 0.9, 1.1	SOAP 1.2
Eventing	GENA	WS-Eventing
Presentation	HTTP, HTML	HTTP, HTML