Service oriented Middleware for IoT

SOM, based on ROA or SOA Approaches

Trends *Web of Things*

- Two kind of Approches

- Service oriented Architectures :
  - ROA (DAO) : Ressource or data oriented
  - SOA : Sevice oriented

Ressource Oriented Architecture

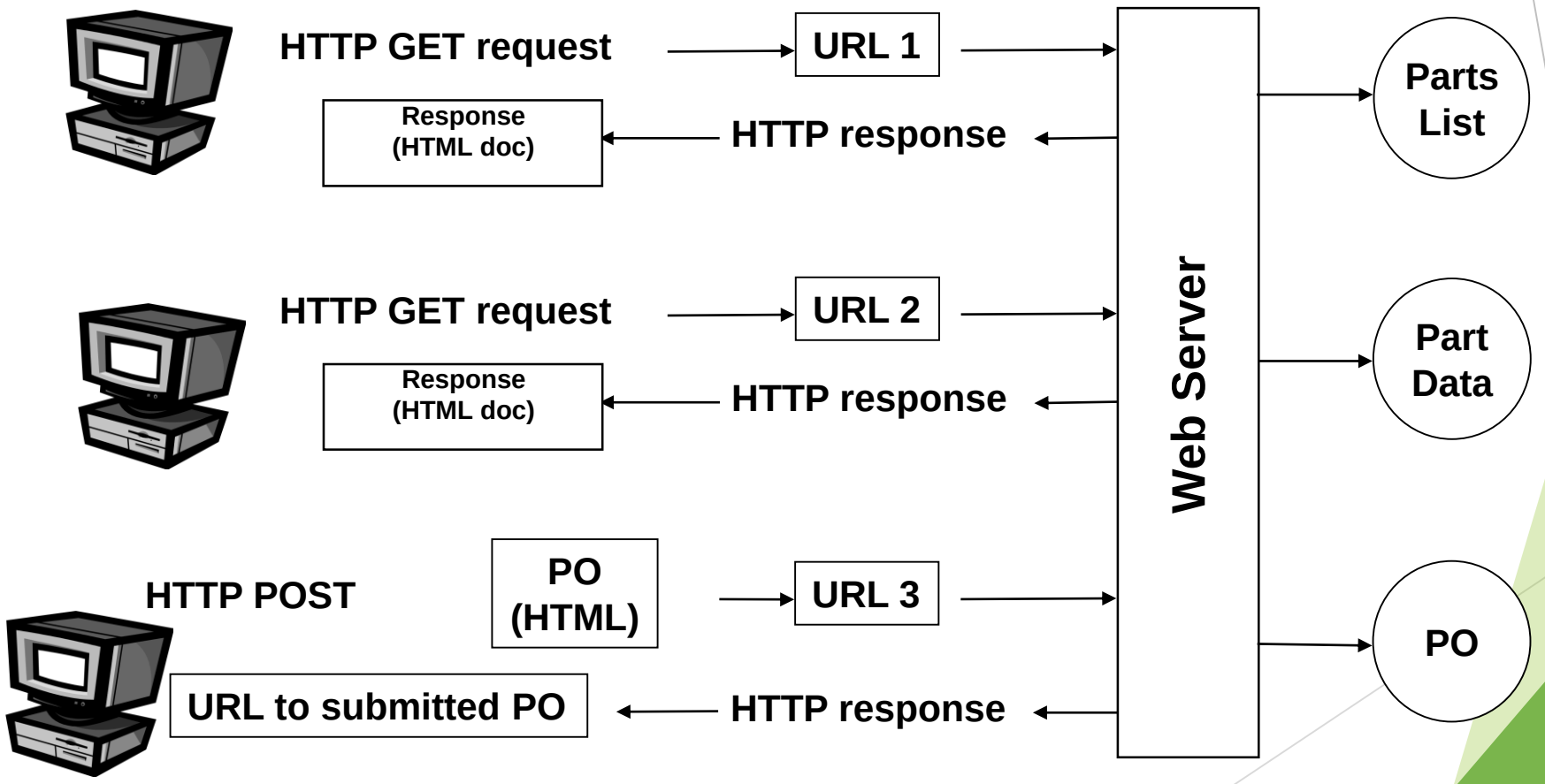
RESTful Web Services

- 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

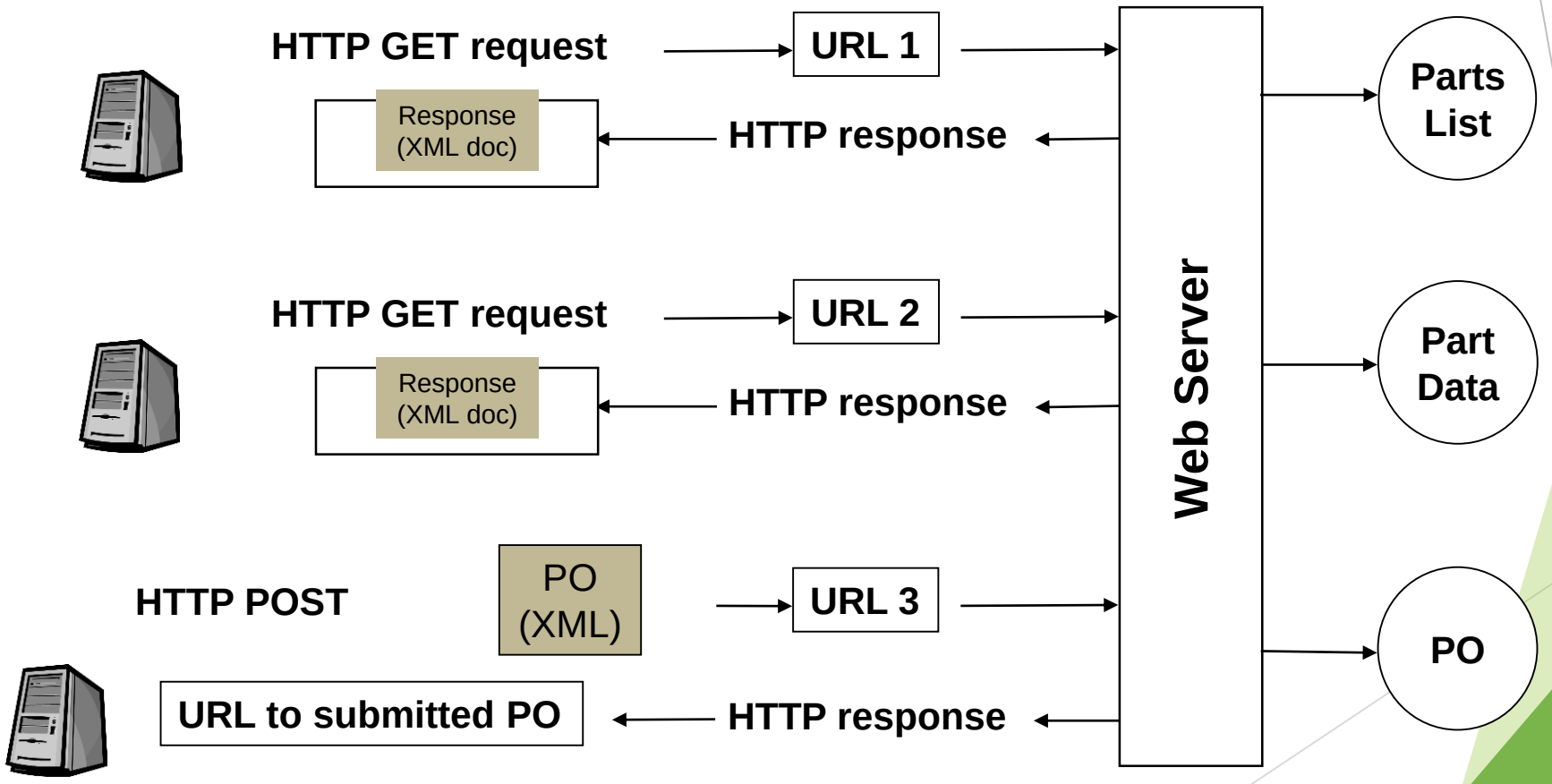
REST architecture

- 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



The RESTful architecture



REST Architecture

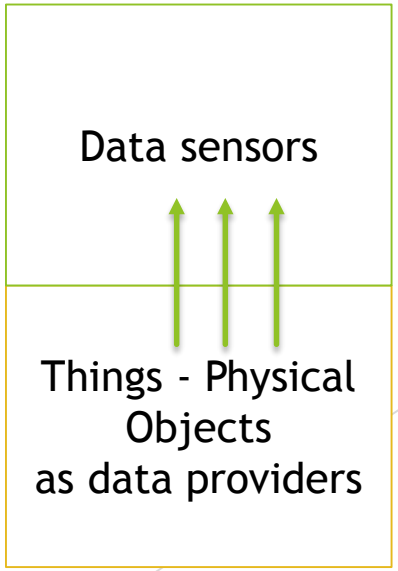
- Servers are stateless and messages can be interpreted without examining history
  - Messages are self-contained
- There is no such thing as a “service”.
  - There are just resources which are accessed through URI
    - URI = generalisation of URL
- Clients navigate through a series of steps towards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

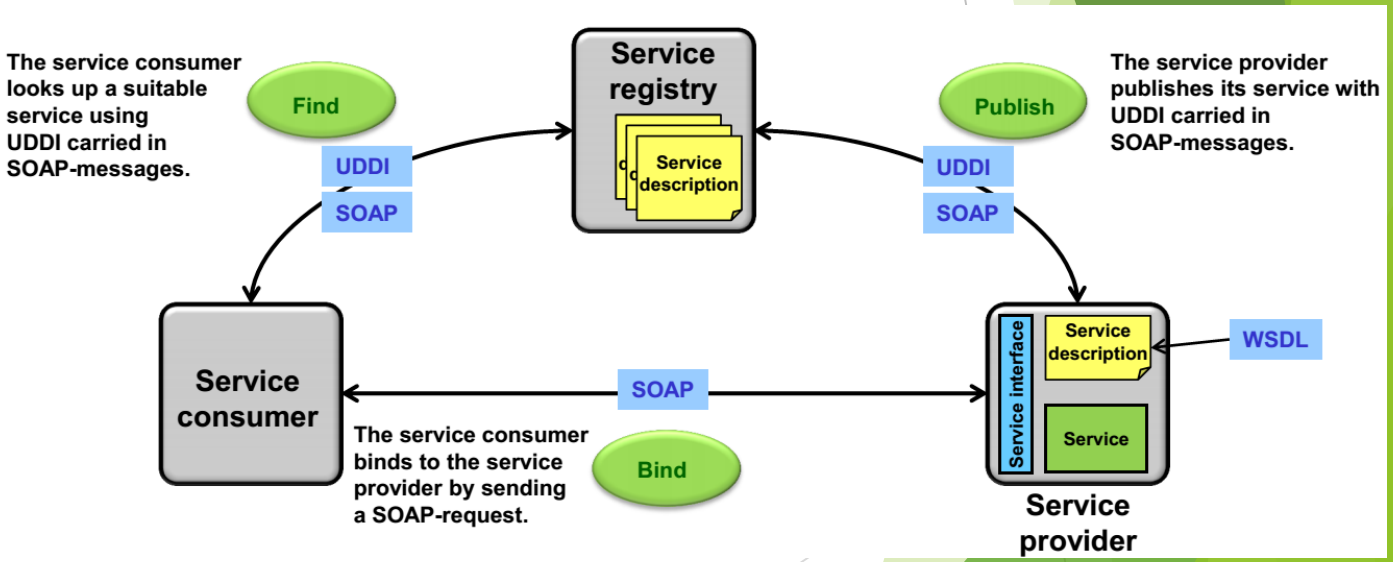
SOA : Service oriented Architecture

- 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 ”.

- SOAP based Web Service, the alternative
- Also called WS-* (for * recommendations, Cf. http://www.w3.org/)

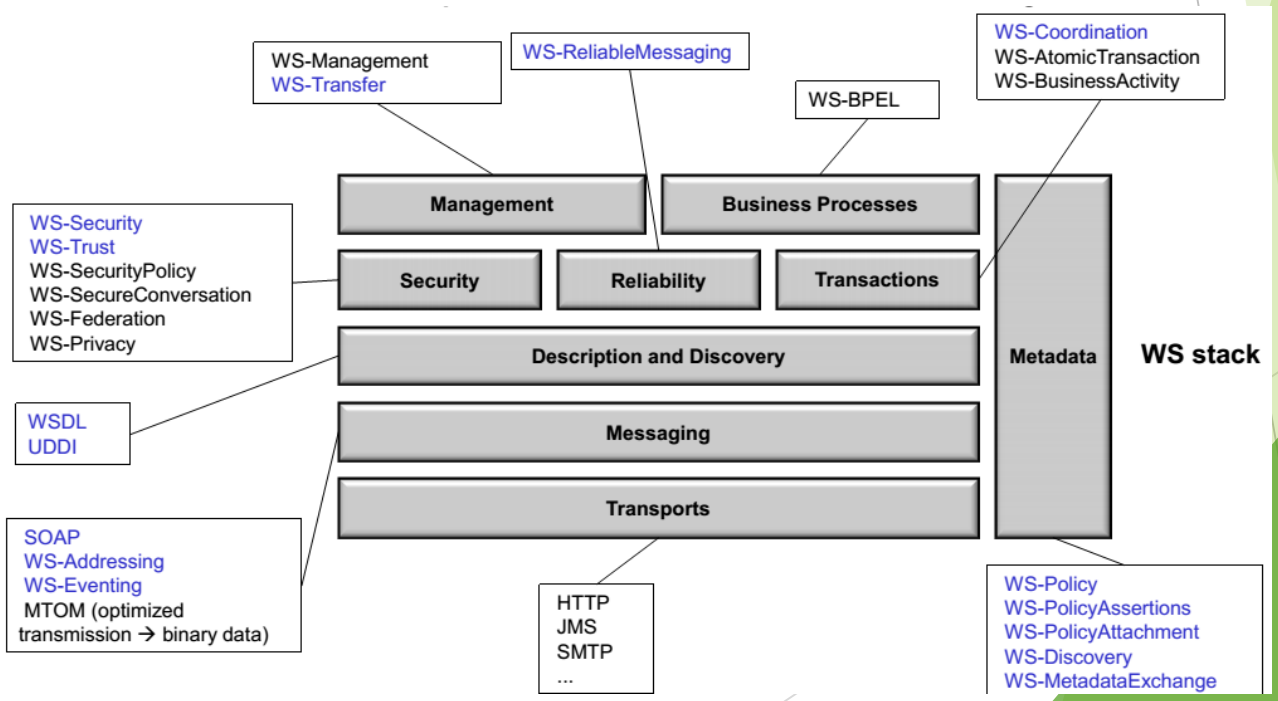
WS-*architecture more than ROA

- 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



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