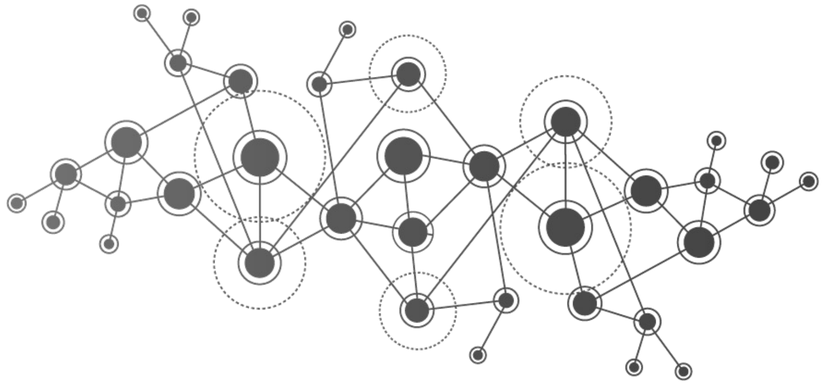
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA



Intelligent Internet of Things

Interoperable IoT Architectures Web of Things

Prof. Marco Picone

A.A 2023/2024

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Interoperable IoT Architectures - Web of Things

- From IoT to WoT
- WoT Definition
- W3C & WoT
- Terminology
- Common Patterns
- Design Principles
- Architectural Components
- Thing Description
- Web Thing
- Use Cases

Main Sources & References:

- Web of Things at W3C - **https://www.w3.org/WoT/**

- Building the Web of Things With examples in Node.js and Raspberry Pi. Dominique D. Guinard and Vlad M. Trifa June 2016, ISBN 9781617292682

2023 - Marco Picone, Ph.D. (marco.picone@unimore.it)

From Internet of Things to Web of Things

- The Internet of Things revolution brings incredible innovations, enormous new opportunities, but also at the same time new risks
- The existing challenging mission is to ensure that the Internet and consequently the IoT is a global public resource, open secure and accessible to all
- This mission has never been more important than today with the IoT where when As new types of devices come online, they bring with them significant new challenges around **security**, **privacy** and **interoperability**.
- Many of the new devices connecting to the Internet are insecure, do not receive software updates to fix vulnerabilities, and raise new privacy questions around the collection, storage, and use of large quantities of extremely personal data.

https://hacks.mozilla.org/2017/06/building-the-web-of-things/

From Internet of Things to Web of Things

	Google nest	Microsoft	amazon		SAMSUNG SmartThings
Cloud Services	Nest Cloud/ Google Cloud	Azure IoT	AWS IoT	iCloud	ARTIK Cloud/ SmartThings
Application Protocols	Weave	AMQP	MQTT	HomeKit	MQTT
Network Protocols	WiFi/Thread	WiFi	WiFi	WiFi/BLE	WiFi/ZigBee/ BLE/Thread
Operating Systems	Linux/Android Things	Windows IoT	Linux/AWS Greengrass	iOS	Linux/ARTIK

- Main IoT devices today use **proprietary vertical technology stacks** which are **built around a central point of control** and which **don’t always talk to each other**
- When they do talk to each other it requires **per-vendor integrations to connect those systems together**
- There are efforts to create standards, but the landscape is extremely complex and there’s still not yet a single dominant model or market leader.

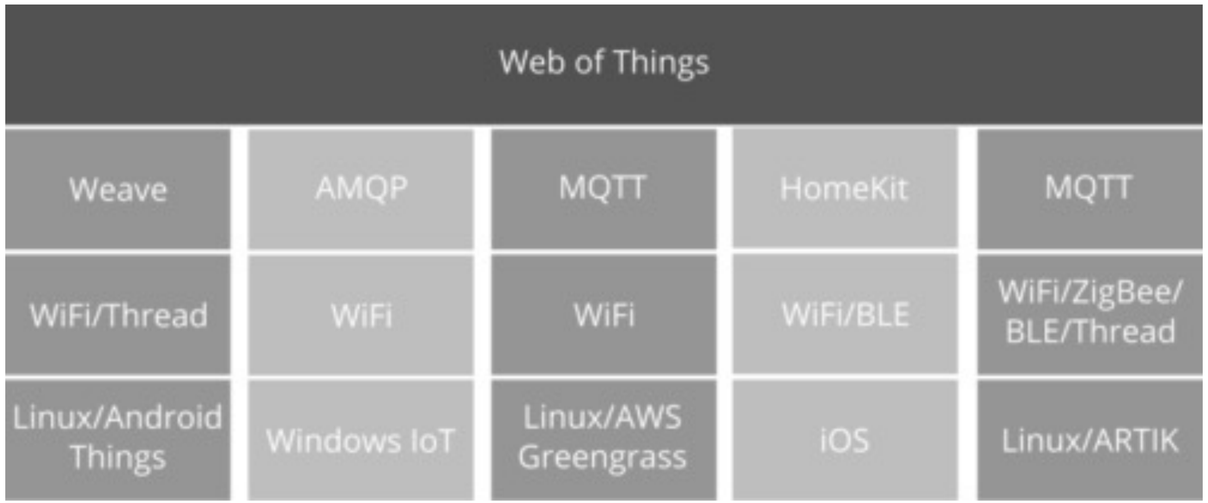
From Internet of Things to Web of Things

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

"Using the Internet of Things today is a lot like sharing information on the Internet before the World Wide Web existed. There were competing hypertext systems and proprietary GUIs, but the Internet lacked a unifying application layer protocol for sharing and linking information."

https://hacks.mozilla.org/2017/06/building-the-web-of-things/

Web of Things (WoT) - Definition



- The “Web of Things” (WoT) is an effort to take the lessons learned from the World Wide Web and apply them to IoT
- WoT aims to create a **decentralized Internet of Things** by using standard interfaces to make things, devices and services **linkable** and **discoverable**, and defining a standard data model and APIs to make them **interoperable**
- The Web of Things **is not just another vertical IoT technology stack** to compete with existing platforms. It is intended as a **unifying horizontal application layer to bridge together multiple underlying IoT protocols**

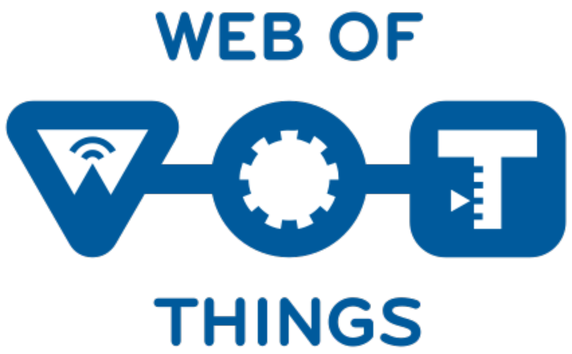
W3C & Web of Things

“The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web. Founded in 1994 and currently led by Tim Berners-Lee, the consortium is made up of member organizations that maintain full-time staff working together in the development of standards for the World Wide Web. As of 21 October 2019, W3C had 443 members. W3C also engages in education and outreach, develops software and serves as an open forum for discussion about the Web.”

- Through the “Web of Things Interest Group” W3C provides a forum for technical discussions to identify use cases and requirements for open markets of applications and services based upon the role of Web technologies for a combination of the Internet of Things (IoT) with the Web of data

https://www.w3.org/WoT/

Web of Things Interest Group



- The Web of Things Interest Group brings together stakeholders interested in the Web of Things to explore ideas prior to **standardization** together with **collaboration** with external groups, e.g. **standards development** organisations and **industry alliances**
- The **mission** of the Web of Things Working Group is to **counter the fragmentation of the IoT** through the **specification of building blocks** that enable **easy integration** of IoT devices and services **across IoT platforms and application domains**

https://www.w3.org/WoT/

W3C - Web of Things - Architecture

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- This WoT Architecture specification describes the abstract architecture for the W3C Web of Things
- This abstract architecture is based on a set of requirements that were derived from use cases for multiple application domains
- A set of modular building blocks are also identified together with the description of how these blocks should work and cooperate together
- The **WoT abstract architecture defines a basic conceptual framework that can be mapped onto a variety of concrete deployment scenarios and several examples**
- The **abstract architecture described in this specification** does not itself define concrete mechanisms or prescribe any concrete implementation.

https://www.w3.org/TR/wot-architecture/

W3C WoT - Introduction

- The goals of the Web of Things (WoT) are to **improve** the **interoperability** and **usability** of the Internet of Things (IoT). Through a collaboration involving many stakeholders **over many years**, several building blocks have been identified that help address these challenges
- This specification is focused on the scope of W3C WoT standardization, which can be broken down into these building blocks as well as the abstract architecture that defines how they are related
- The W3C documents define and describe:
  - Terminology
  - Common Patterns
  - Architecture Building Blocks
  - Things Descriptions
  - Binding Templates
  - Scripting API
  - Security & Privacy Guidelines
  - Use Cases

W3C WoT - Terminology (Main Concepts)

- **WoT Thing Description or Thing Description (TD):** Structured data describing a Thing. A WoT Thing Description comprises general metadata, domain-specific metadata, Interaction Affordances (which include the supported Protocol Bindings), and links to related Things. The WoT Thing Description format is the central building block of W3C WoT.
- **Binding Templates:** A re-usable collection of blueprints for the communication with different IoT platforms. The blueprints provide information to map Interaction Affordances to platform-specific messages through WoT Thing Description as well as implementation notes for the required protocol stacks or dedicated communication drivers.
- **Consumed Thing:** A software abstraction that represents a remote Thing used by the local application. The abstraction might be created by a native WoT Runtime, or instantiated as an object through the WoT Scripting API.
- **Consuming a Thing:** To parse and process a TD document and from it create a Consumed Thing software abstraction as interface for the application in the local runtime environment.
- **Exposed Thing:** A software abstraction that represents a locally hosted Thing that can be accessed over the network by remote Consumers. The abstraction might be created by a native WoT Runtime, or instantiated as an object through the WoT Scripting API.
- **Thing Directory:** A directory service for TDs that provides a Web interface to register TDs (similar to [CoRE-RD]) and look them up (e.g., using SPARQL queries or the CoRE RD lookup interface [CoRE-RD]).
- And counting ….

2023 - Marco Picone, Ph.D. (marco.picone@unimore.it)

W3C WoT - Common Patterns

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Common patterns have the role to **illustrate how devices/things interact with controllers, other devices, agents and servers**
- We use the term client role as an initiator of a transport protocol, and the term server role as a passive component of a transport protocol. **A device can be in a client and server role simultaneously**.
- One example of this dual role is a sensor, that registers itself with a cloud service and regularly sends sensor readings to the cloud. In the response messages the cloud can adjust the transmission rate of the sensor's messages or select specific sensor attributes, that are to be transmitted in future messages. Since the sensor registers itself with the cloud and initiates connections, it is in the 'client' role. However, since it also reacts to requests, that are transmitted in response messages, it also play a 'server' role.
- The following patterns illustrate the **roles**, **tasks**, and **use case** with increasing complexity
- The aim is not to be exhaustive but instead to motivate and support the WoT architecture and building blocks

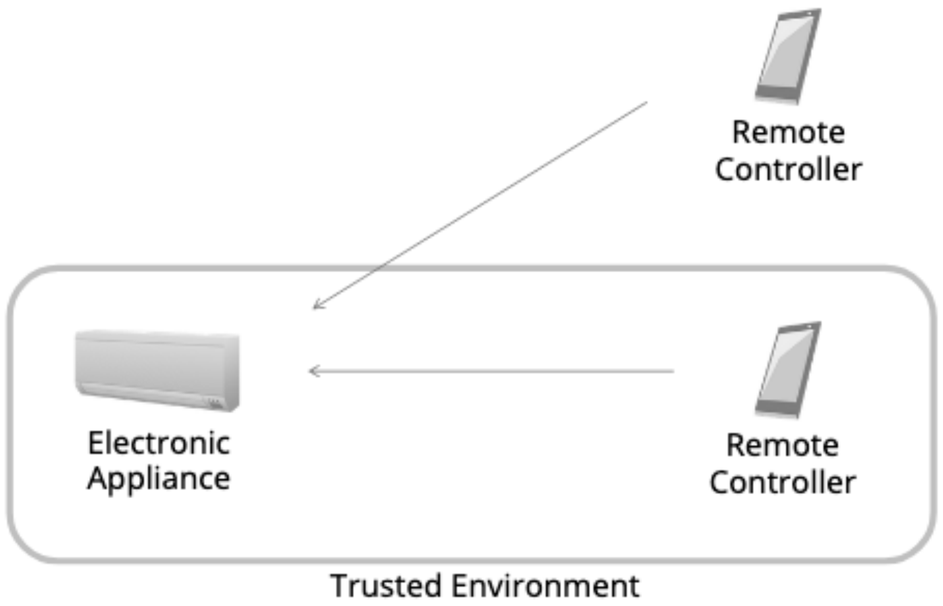
W3C WoT - Pattern - Device Controllers

- The first use case is a local device controlled by a user-operated remote controller as illustrated
- A remote controller can **access** an electronic appliance **through the local (e.g., home) network directly**. In this case, the **remote controller can be implemented by a browser or native application**
- In this pattern, **at least one device** like the electronic appliance has a **server role** that can accept a request from the other devices and responds to them, and sometimes initiates a mechanical action
- The other device like the **remote controller has a client role** that can send a message with a request, like to read a sensor value or to turn on the device
- Moreover, to emit a current state or event **notification** of a device, t**he device may have a client role** that can send a message to another device, which has server roles

W3C WoT - Pattern - Thing to Thing

- An example scenario associato to the Thing to Thing scenario is the following:
  - a sensor detects a change of the room condition (e.g., the temperature exceeding a threshold)
  - it issues a control message like "turn on" to the electronic appliance
- The sensor unit can issue some trigger messages to other devices (multiple actuators for example)
- **In this case, when two devices that have server roles are connected, at least one device must have also a client role that issues a message to the other to actuate or notify.**

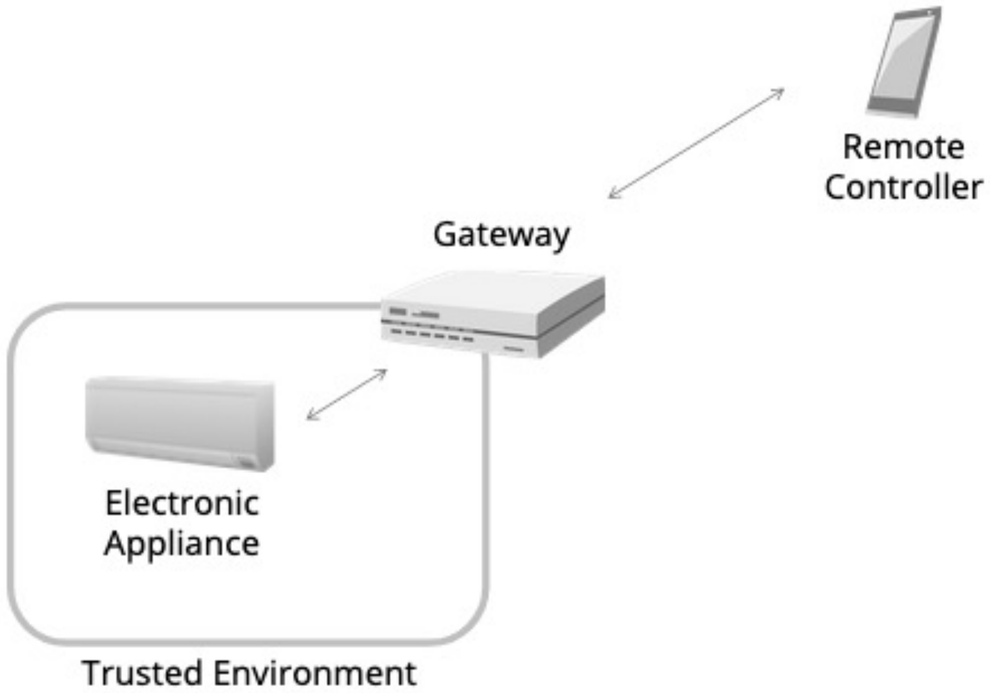
W3C WoT - Pattern - Remote Access



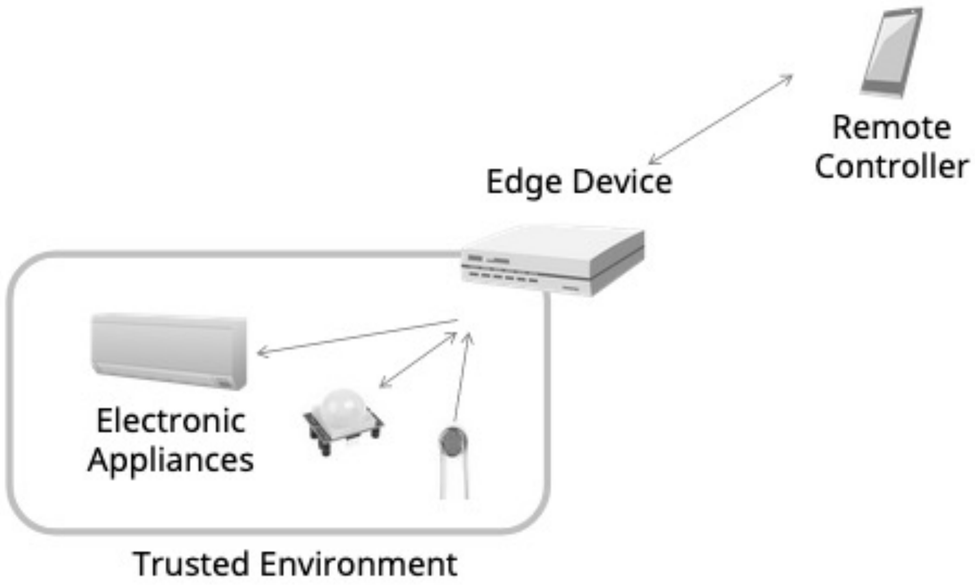
- The **remote controller can switch between different network connections and protocols** (e.g., between a cellular network and a home network, which is using protocols such as Wi-Fi and Bluetooth)
- When the controller is in the **local network** it is a trusted device and **no additional** networking **security** or **access control** is required (local security mechanisms are maintained)
- When it is **outside of the trusted network**, **additional access control and security mechanisms must be applied** to ensure a trusted relationship. Note that in this scenario the network connectivity may change due to switching between different network access points or cellular base stations.
- In this pattern, **the remote controller and the electronic appliance have a client and a server role**

W3C WoT - Pattern - Smart Home Gateway

- The **gateway** is placed **between** a local **network** and the **Internet**
- It manages electronic appliances inside the house and **can receive commands** from a remote controller over the Internet, (e.g., from a smartphone as in the previous use case)
- **It is also is a virtual representation of a device**. The Smart Home Gateway typically **offers proxy and firewall functionality**.
- In this pattern, the home **gateway** has both a **client** and a **server role**
- When the remote controller actuates the electronic appliance, it can connect to the electronic appliance in the client role and to the remote controller with the server role
- When the **electronic appliance emits a message** to the remote controller, **the gateway act as server roles for the electric appliance, and it act as client roles for the remote controller**



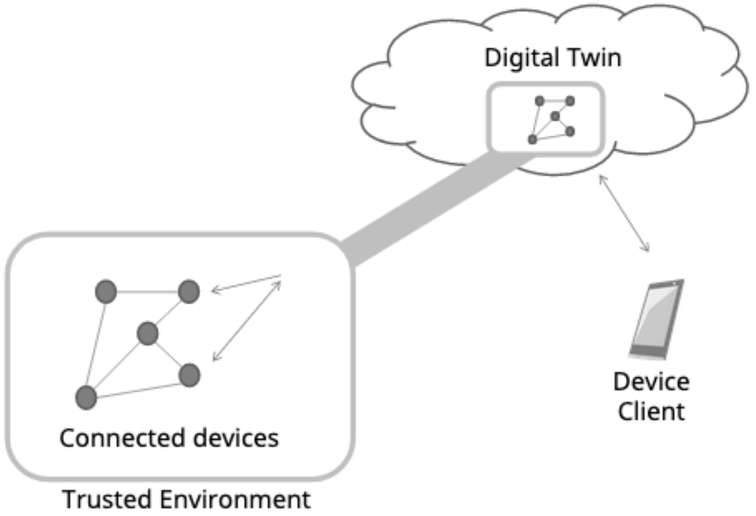
W3C WoT - Pattern - Edge Devices

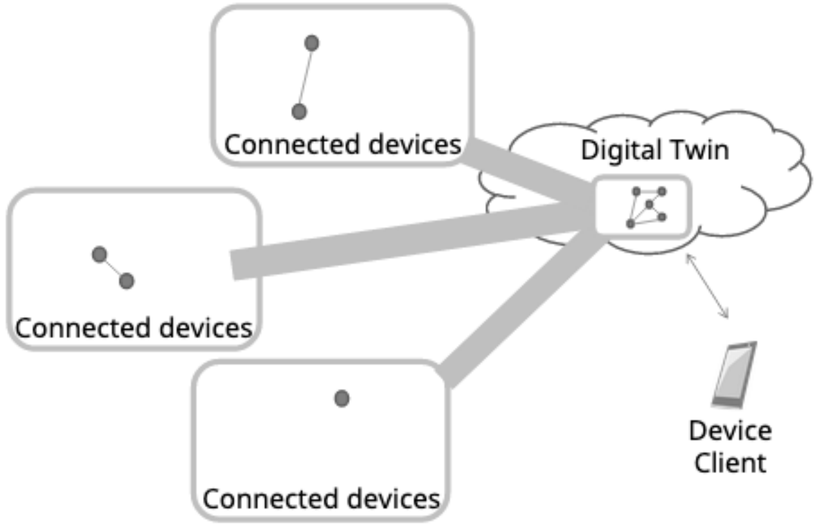


- An Edge Device or **Edge Gateway** is similar to a Smart Home Gateway
- We use the term to indicate **additional tasks that are carried out by the edge gateway**
- Whereas the local gateway primarily just bridges between the public and the trusted network, the **edge device has local compute capabilities and typically bridges between different protocols**
- Edge devices are typically used in industrial solutions, where they can provide **preprocessing**, **filtering** and **aggregation** of data provided by connected devices and sensors.

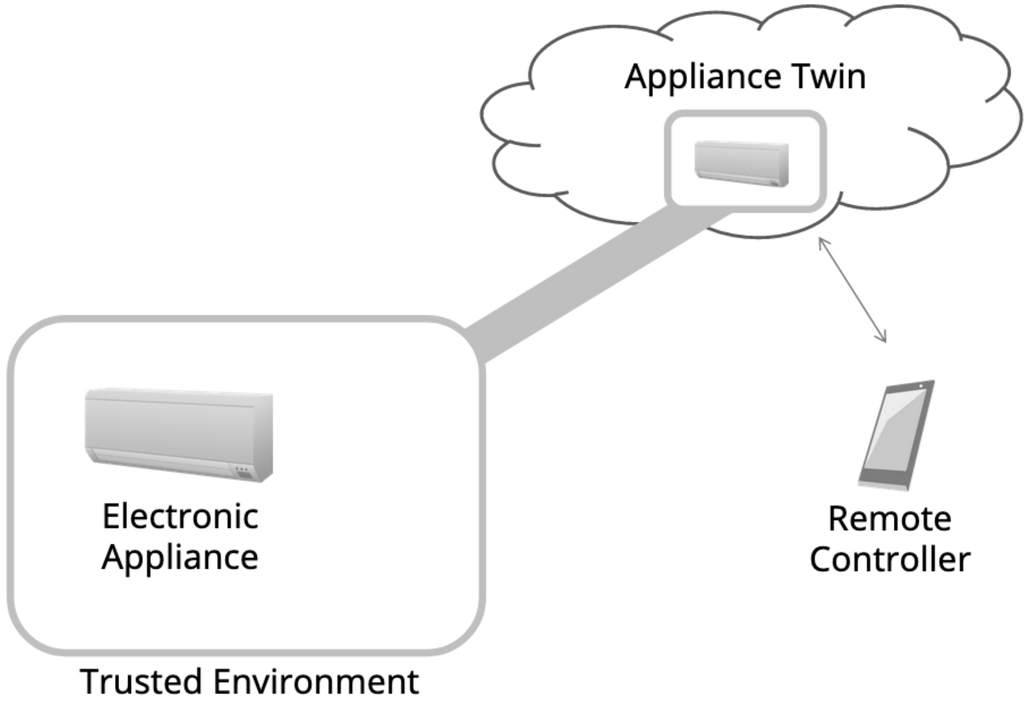
W3C WoT - Pattern - Digital Twins

- A digital twin is a virtual representation, i.e. a model of a device or a group of devices that resides on a cloud server or edge device
- It can be used to represent real-world devices which may not be continuously online, or to run simulations of new applications and services, before they get deployed to the real devices
- Digital twins can model a single device, or they can aggregate multiple devices in a virtual representation of the combined devices
- Digital twins can be realized in different ways, depending on whether a device is already connected to the cloud, or whether it is connected to a gateway, which itself is connected to the cloud.



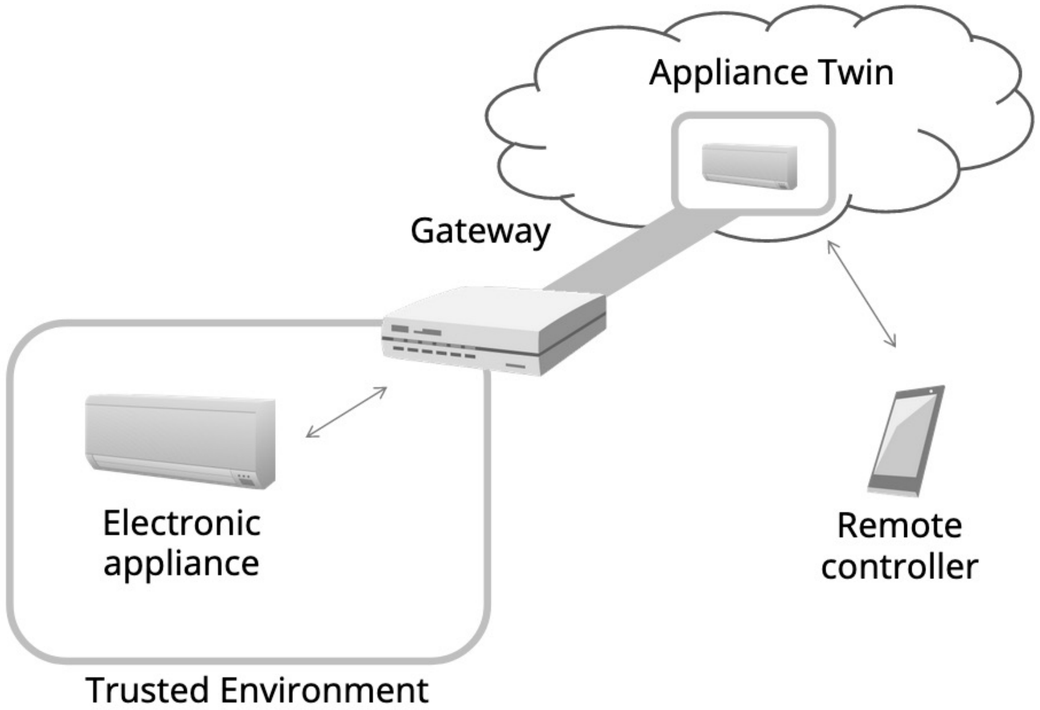


W3C WoT - Pattern - Digital Twins - Cloud-ready Devices



- In that Patters electronic appliances are “Cloud-ready” and directly connected to the cloud
- The **cloud mirrors** the **appliances** and, **acting as a digital twin**, can receive commands from remote controllers (e.g., a smartphone)
- Authorized controllers can be located anywhere, as the digital twin is globally reachable.

W3C WoT - Pattern - Digital Twins - Legacy Devices



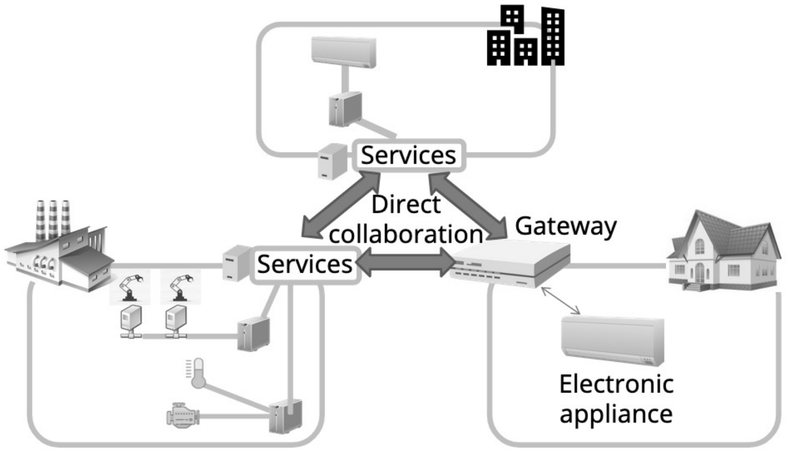
- In this case the legacy **electronic appliances cannot directly connect to the cloud** (for different reasons related for example to security or processing and networking capabilities)
- The **gateway** is needed to **relay the connection**. The gateway works as:
  - **integrator** of a variety of legacy communication protocols both in the physical and logical view
  - **firewall** toward the Internet
  - **privacy** filter which substitutes real image and/or speech, and logs data locally
  - **local agent** in case the network connection is interrupted
  - **emergency services** running locally when fire alarms and similar events occur
- The **cloud mirrors** the **gateway** with all connected appliances and acts as a **digital twin** that manages them in the cloud in conjunction with the gateway
- Furthermore, the cloud can receive commands from **remote controllers** (e.g., a smartphone), which can be **located anywhere**

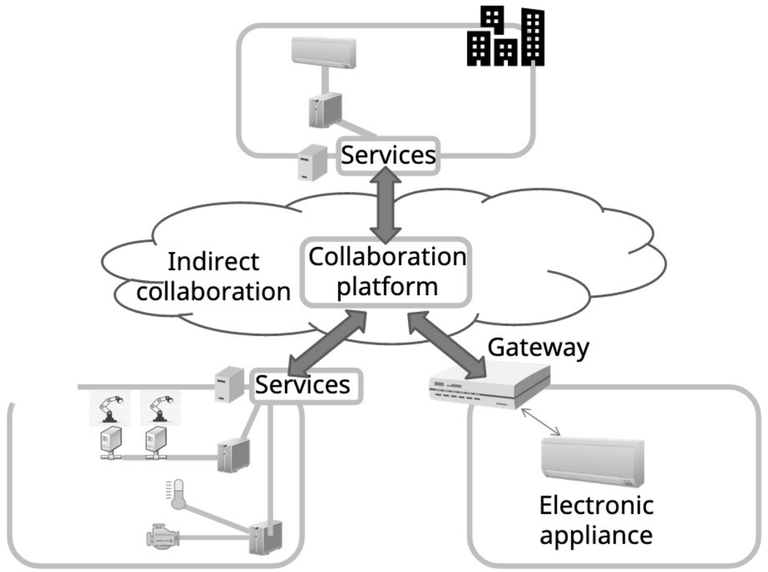
W3C WoT - Pattern - Multi Cloud

- Typical IoT deployments consist of multiple (thousands) of devices and without a standardized mechanism, the management of firmware updates for specific clouds require a lot of effort and hinders wider scale IoT adoption
- The primary benefit of a standardized mechanism for describing devices and device types is the capability of deploying devices to different cloud environments without the need of doing customization at device software / firmware level, i.e., installing cloud specific code to a device
- This implies that the solution is flexible enough to describe devices in a way that allows on-boarding and using devices in **multiple IoT cloud environments**
- This drives adoption of Web of Things devices, since it enables easy usage of new devices in an existing deployment, as well as migration of existing devices from one cloud to the other

W3C WoT - Pattern - Cross-domain Collaboration

- In Cross-Domain Collaboration Pattern each system involves and interact with other systems in other domains
- For example -> Smart Factory with Smart City, Smart City with Smart Home
- This type of system is called **"Symbiotic" ecosystem** [IEC-FOTF]
- There are two collaboration models:
  - **Direct Collaboration**: model, systems exchange information directly with each other in a **peer-to-peer** manner
  - **Indirect Collaboration**: systems exchange information via some **collaboration platform**
- In order to maintain and continue this collaboration, **each system provides the metadata** of their **capabilities** and **interfaces** and adapts itself to others.

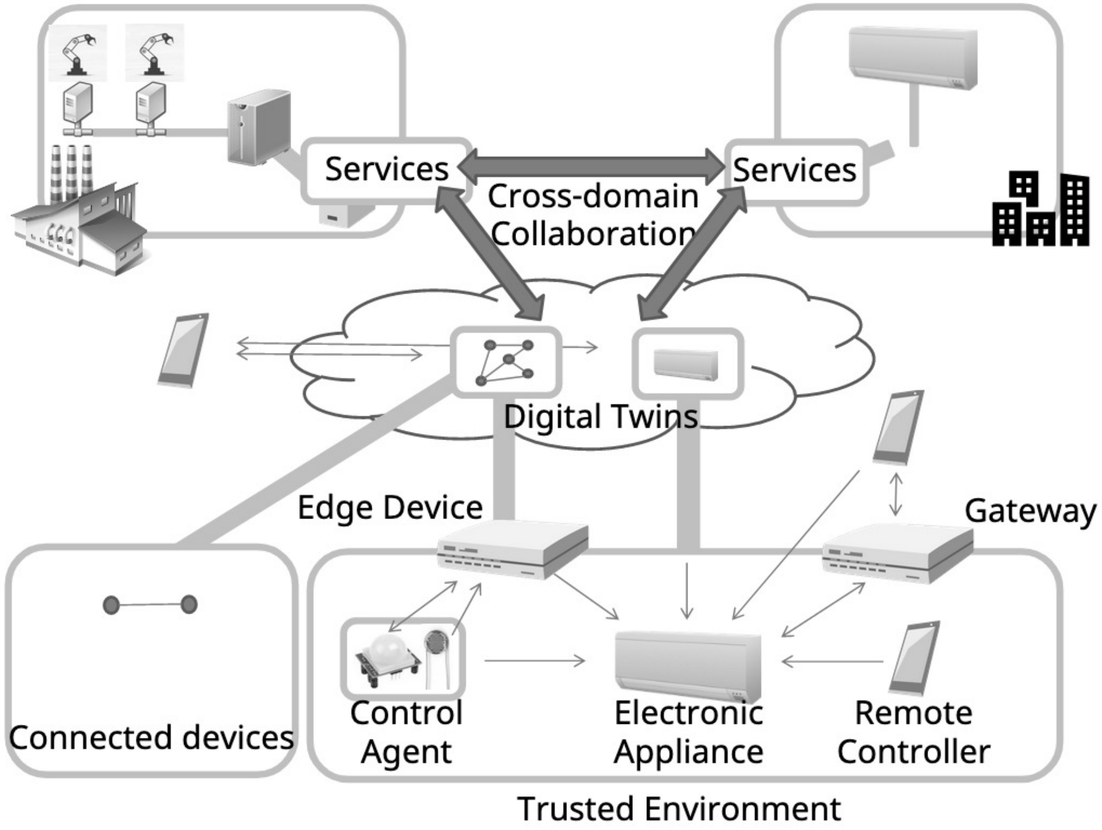




[IEC-FOTF] Factory of the future. IEC. October 2015. URL: https://www.iec.ch/whitepaper/pdf/iecWP-futurefactory-LR-en.pdf

W3C WoT - Pattern - Overview

- The presented WoT patterns describes various architecture application scenarios and interaction forms
- In these patterns, some functional entities such as the devices including the legacy devices, controllers, gateways and cloud servers are located at physical locations such as inside building, outside buildings, and data centers
- In a transport protocol layer, each entity arbitrarily selects a suitable role for communications.
- Regardless of this, in application layer, an application sees that a device provides abstract interfaces to interact and the application can interact with the device using their abstract interfaces.



W3C WoT - Architectural Common Principles

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- WoT architecture:
  - should enable **mutual interworking** of **different eco-systems** using web technology
  - should be based on the **web architecture using RESTful APIs**
  - should allow to use **multiple payload formats** which are commonly used in the web.
  - must **enable different** device **architectures** and must not force a client or server implementation of system components

W3C WoT - Architectural Common Principles

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Additional fundamental design principles are related to:
  - **Flexibility** -> There are a wide variety of physical device configurations for WoT implementations. The **WoT** abstract **architecture** should be able to be **mapped to and cover all of the variations**.
  - **Compatibility** -> There are already many existing IoT solutions and ongoing IoT standardization activities in many business fields. The **WoT should provide a bridge between these existing and developing IoT solutions and Web technology based on WoT concepts**. The WoT should be upwards compatible with existing IoT solutions and current standards.
  - **Scalability** -> WoT must be able to scale for IoT solutions that **incorporate thousands to millions of devices**. These devices may offer the same capabilities even though they are created by different manufacturers.
  - **Interoperability** -> WoT must provide **interoperability across device and cloud manufacturers**. It must be possible to take a WoT enabled device and connect it with a cloud service from different manufacturers out of the box.

W3C WoT - Thing Functionalities

- WoT architecture should allow **things** to have functionalities such as
  - **reading** thing's status information
  - **updating** thing's status information which might cause actuation
  - **subscribing** to, receiving and unsubscribing to notifications of changes of the **thing's status** information.
  - **invoking functions** with input and output parameters which would cause certain actuation or calculation.
  - **subscribing** to, receiving and **unsubscribing** to **event notifications** that are more general than just reports of state transitions.

W3C WoT - Description Mechanisms

- WoT architecture should support a **common description mechanism which enables describing things and their functions**
- Such descriptions should be **not only human-readable, but also machine-readable**.
- Such descriptions should allow **semantic annotation** of its structure and described contents.
- Such description should be able to be **exchanged using multiple formats** which are commonly used in the web.
- WoT architecture should allow describing thing's attributes such as
  - name
  - explanation
  - version of spec, format and description itself
  - links to other related things and metadata information
- Such descriptions should support also **internationalization**

W3C WoT - Network

- WoT architecture should support multiple web protocols which are commonly used.
- Such protocols include
  - **protocols** commonly used in the **internet**
  - **protocols** commonly used in the **local area network**
- WoT architecture should allow using **multiple web protocols to access to the same functionality**
- WoT architecture should allow using a **combination of multiple protocols to the functionalities of the same thing (e.g., CoAP, MQTT, HTTP and WebSocket)**

W3C WoT - Deployment & Application

- WoT architecture should **support a wide variety of thing capabilities such as edge devices** with resource restrictions and virtual things on the cloud, based on the same model.
- WoT architecture should support **multiple levels of thing hierarchy with intermediate entities such as gateways and proxies**
- WoT architecture should s**upport accessing things in the local network** from the **outside** of the local network (the internet or another local network), considering **network address translation**
- WoT architecture should allow **describing applications** for a wide variety of things such as edge device, gateway, cloud and UI/UX device, using web standard technology based on the same model.

W3C WoT - Legacy Adoption

- WoT architecture should allow **mapping of legacy IP and non-IP protocols to web protocols**, supporting various topologies, where such **legacy protocols are terminated and translated**
- WoT architecture should allow **transparent use of existing IP protocols without translation**, which follow RESTful architecture.
- WoT architecture must not enforce client or server roles on devices and services. An **IoT device can be either a client or a server, or both, depending on the system architecture; the same is true of edge and cloud services**

W3C WoT - Main Architectural Components

- The use cases help to identify basic components such as devices and applications, that access and control those devices, proxies (i.e., gateways and edge devices) that are located between devices. An additional component useful in some use cases is the directory, which assists with discovery.
- Those components are connected to the internet or field networks in offices, factories or other facilities. Note that all components involved may be connected to a single network in some cases, however, in general components can be deployed across multiple networks.
- Main components and aspects are related to:
  - Devices
  - Applications
  - Digital Twins
  - Discovery
  - Security
  - Accessibility

W3C WoT - Devices

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

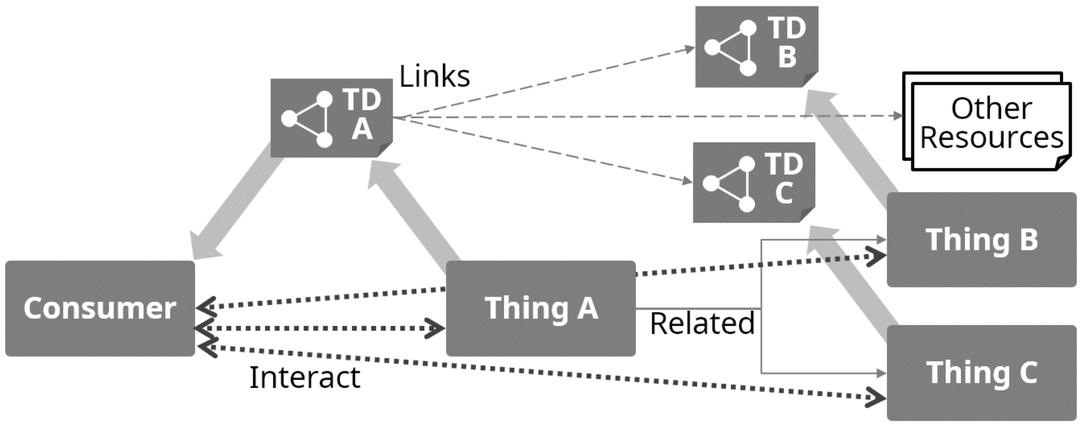
- **Access to devices is made using a description of their functions and interfaces**
- This description is called **Thing Description (TD)**
- A Thing Description includes a **general metadata about the device**, information models representing **functions**, **transport protocol** description for operating on information models, and **security** information.
- General metadata contains device identifiers (**URI**), device information such as **serial number**, **production date**, **location** and other human readable information.
- Information models defines device attributes, and represent device's internal settings, **control** functionality and **notification functionality**
- **Devices that have the same functionality have the same information model regardless of the transport protocols used.**

W3C WoT - Devices

- Because many systems based on Web of Things architecture are crossing system Domains, **vocabularies and meta data (e.g., ontologies) used in information models should be commonly understood by involved parties**
- In addition to **REST** transports, **Pub/Sub** transports are also **supported**.
- **Security** information includes descriptions about **authentication**, **authorization** and secure **communications**
- Devices are required to put TDs either inside them or at locations external to the devices, and to make TDs accessible so that other components can find and access them.

W3C WoT - Applications

- **Applications** need to be able to generate and use **network** and **program** interfaces **based on metadata** and **descriptions**
- Applications have to be able to **obtain these descriptions** through the network, therefore, need to be able to conduct **search** operations and acquire the necessary descriptions over the network



W3C WoT - Digital Twins

- **Digital Twins need to generate program interfaces internally based on metadata** (descriptions), and to represent virtual devices by using those program interfaces
- A **twin** has to **produce** a **description** for the virtual device and make it externally available
- Identifiers of virtual devices need to be newly assigned, therefore, are different from the original devices. This makes sure that virtual devices and the original devices are clearly recognized as separate entities
- **Transport and security mechanisms and settings of the virtual devices can be different from original devices if necessary**. Virtual devices are required to have descriptions provided either directly by the twin or to have them available at external locations. In either case it is required to make the descriptions available so that other components can find and use the devices associated with them.

W3C WoT - Discovery

- For TDs of devices and virtual devices to be accessible from devices, applications and twins, there needs to be a **common way to share TDs**
- **Directories** can serve this requirement by providing **functionalities** to allow **devices** and **twins** themselves **automatically** or the users to manually **register** the **descriptions**.
- **Descriptions** of the devices and virtual devices need to be **searchable** by external entities.
- Directories have to be able to process search operations with search keys such as keywords from the general description in the device description or information models.

W3C WoT - Security

- **Security information** related to devices and virtual devices **needs to be described in device descriptions**
- This includes information for **authentication/authorization** and **payload encryption**.
- **WoT** architecture should support multiple security mechanism commonly used in the web, such as **Basic, Digest, Bearer and OAuth2.0**

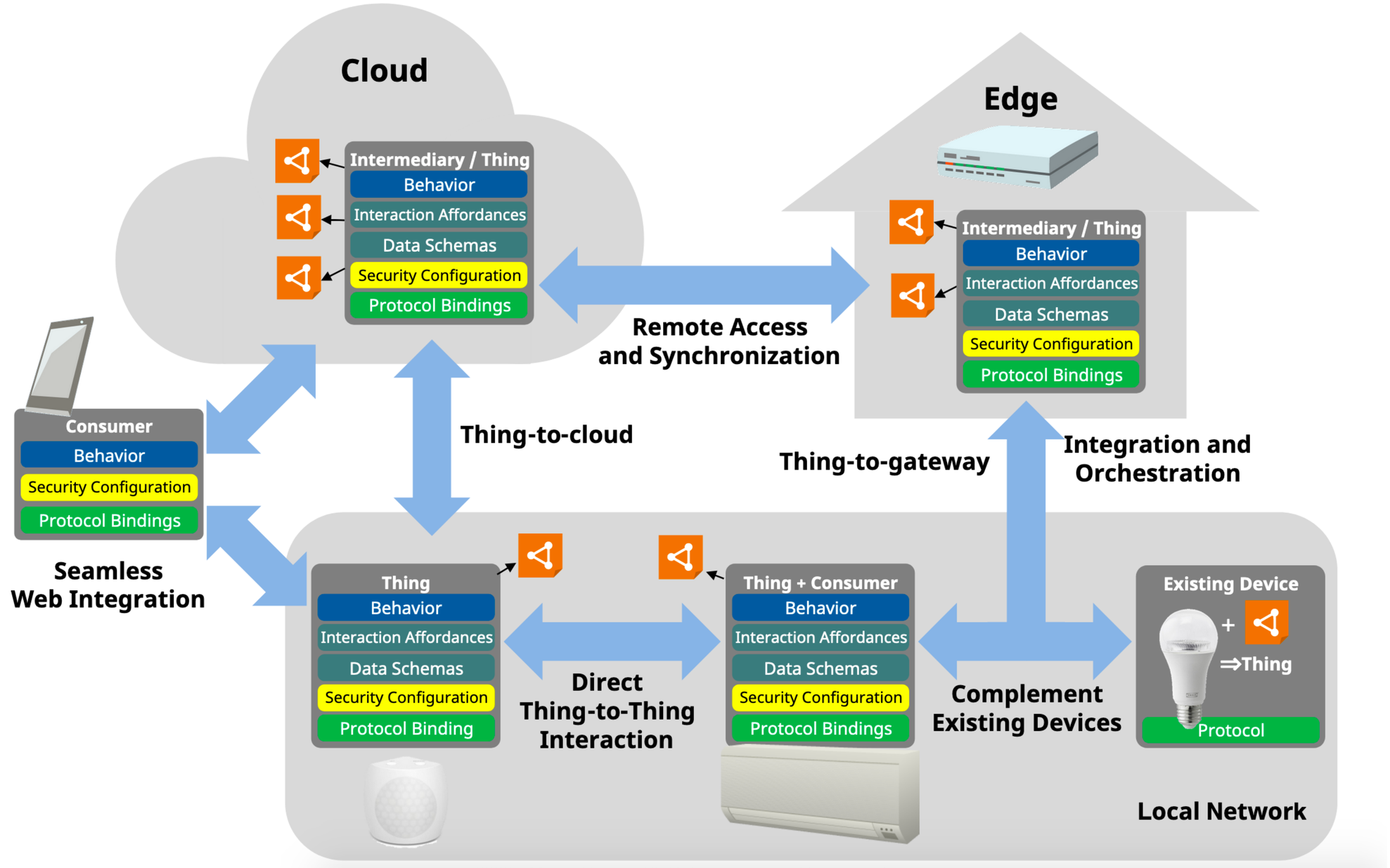
W3C WoT - Accessibility

- The Web of Things **primarily targets machine-to-machine communication**
- The humans involved are usually developers that integrate Things into applications
- **End-users will be faced with the front-ends of the applications or the physical user interfaces provided by devices themselves. Both are out of scope of the W3C WoT specifications.**
- Given the focus on IoT instead of users, accessibility is not a direct requirement, and hence is not addressed within this specification
- There is, however, an interesting aspect on accessibility: Fulfilling the requirements above enables machines to understand the network-facing API of devices. This can be utilized by accessibility tools to provide user interfaces of different modality, thereby **removing barriers to using physical devices and IoT-related applications**

W3C WoT - Architecture

- The concepts of W3C WoT are applicable to all levels relevant for IoT applications:
  - device level
  - edge level
  - cloud level
- This fosters common interfaces and APIs across the different levels and enables various integration patterns such as:
  - Thing-to-Thing
  - Thing-to-Gateway
  - Thing-to-Cloud
  - Gateway-to-Cloud
  - Cloud federation, i.e., interconnecting cloud computing environments of two or more service providers, for IoT applications

W3C WoT - Architecture - Abstract Overview



W3C WoT - Architecture - Affordances

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- A central aspect in W3C WoT is the provision of **machine-understandable metadata** (i.e., WoT Thing Descriptions). Ideally, such metadata is self-descriptive, so that Consumers are able to identify what capabilities a Thing provides and how to use the provided capabilities
- A key to this self-descriptiveness lies in the concept of **affordances**
- The term has been adopted in the field of Human-Computer Interaction (HCI) based on the definition by Donald Norman: **"'Affordance' refers to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used."**
- A simple example for this is a door with a handle. The door handle is an affordance, which suggests that the door can be opened. For humans, a door handle usually also suggests how the door can be opened

W3C WoT - Architecture - Affordances

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- In the context of the Web of Things (WoT), **affordances** refer to the capabilities or actions that an object or device offers to users or other devices
- Affordances describe **what an object can do or how it can be used in a specific context**
- In the perspective of WoT, affordances apply to things connected to the network and represented through Thing Descriptions (TDs)
- In summary, affordances in the Web of Things represent the **functionalities and actions offered by connected devices**, and they are **essential for standardization, interoperability, and interaction among different things and services within the WoT ecosystem**
- This approach facilitates the development of WoT applications that can consistently leverage the capabilities of devices, improving user experience and the flexibility of IoT solutions.

W3C WoT - Architecture - Affordances

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Affordances are relevant in the context of the Web of Things from the following perspective:
  - **Affordance Descriptions:** Within a TD the various affordances of a Thing are defined. For example, a TD might define the actions that a smart light bulb can perform, such as "turn on," "turn off," or "adjust brightness." These actions represent the affordances of the light bulb.
  - **Interoperability:** Specifying affordances in a TD is essential to ensure interoperability between WoT devices and services. WoT application developers can access and use these affordances consistently, regardless of the manufacturer or type of device, thanks to the standardization of definitions within TDs.
  - **Communication & Interaction:** Affordances allow WoT devices to communicate and interact with each other. For example, if a security camera defines a "capture image" action, a WoT application can trigger this affordance to capture an image from the camera.
  - **Semantics & Meaning:** Affordances are associated with specific meanings within WoT. These meanings are defined using ontologies or semantic models, enabling a shared understanding of devices and their capabilities among different actors in the WoT.
  - **Security & Authorization:** Affordances can be associated with security controls and authorizations. For instance, access to certain affordances might require authentication and authorization, ensuring that only authorized users can perform specific actions.

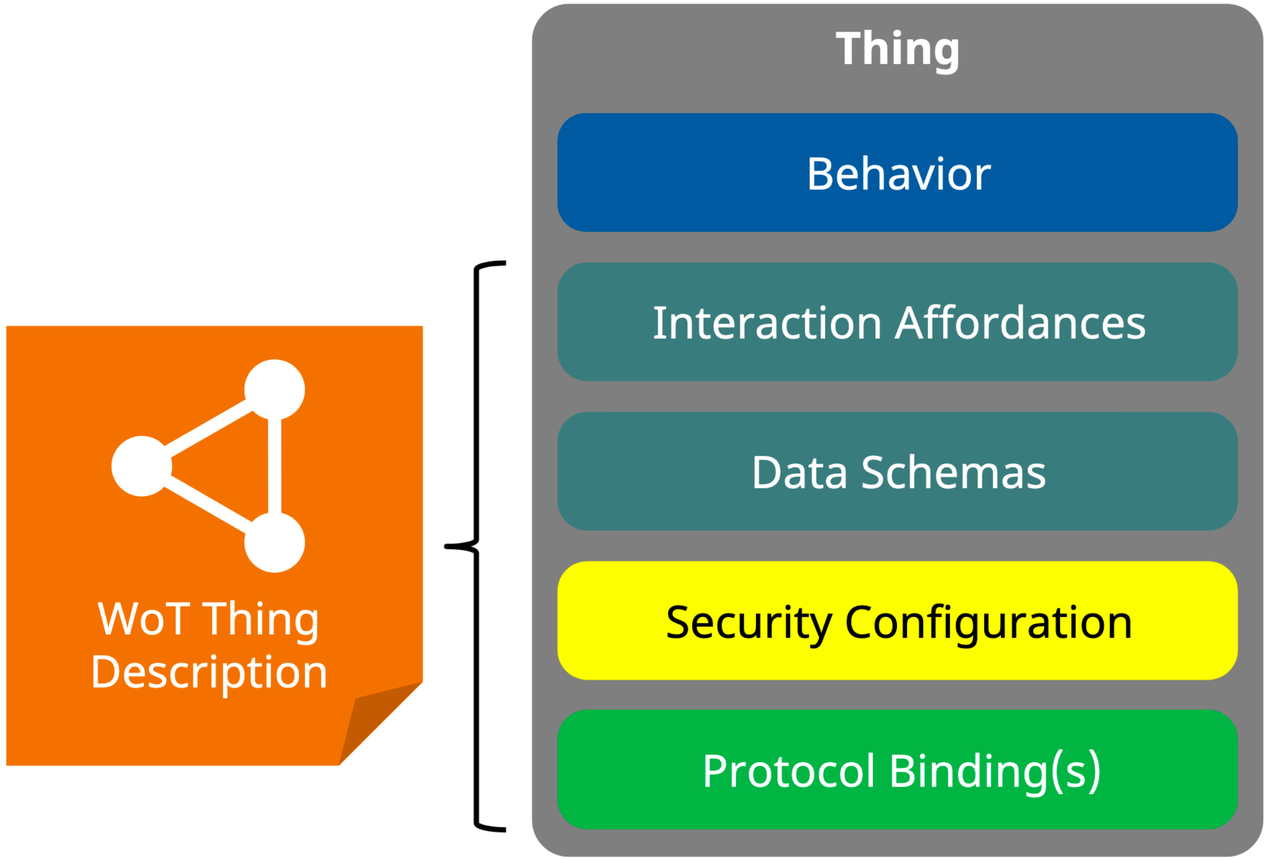
W3C WoT - Architecture - Affordances

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- The hypermedia principle, which is one of the core foundations of the REST architectural style, demands that any piece of information available on the Web be linked to other pieces of information so that the consumer of the information gets explicit knowledge about how to navigate the Web and control Web applications
- In the WoT context, the simultaneous presentation of information and control (provided in the form of hyperlinks) is a mechanism that affords Web clients the means to drive Web applications
- In this context, **an affordance is the description of a hyperlink (e.g., via a link relation type and link target attributes) suggesting Web clients how to navigate and possibly how to act on the linked resource. Hence, links provide navigation affordances**
- The **Web of Things** defines **Interaction Affordances** as **metadata** of a Thing that **shows** and **describes** the possible choices to Consumers, thereby suggesting **how Consumers may interact with the Thing**
- A general Interaction Affordance is navigation, which is activated by following a link, thereby enabling Consumers to browse the Web of Things. The other Interaction Models are: Properties, Actions, and Events.

W3C WoT - Architecture - Web Things

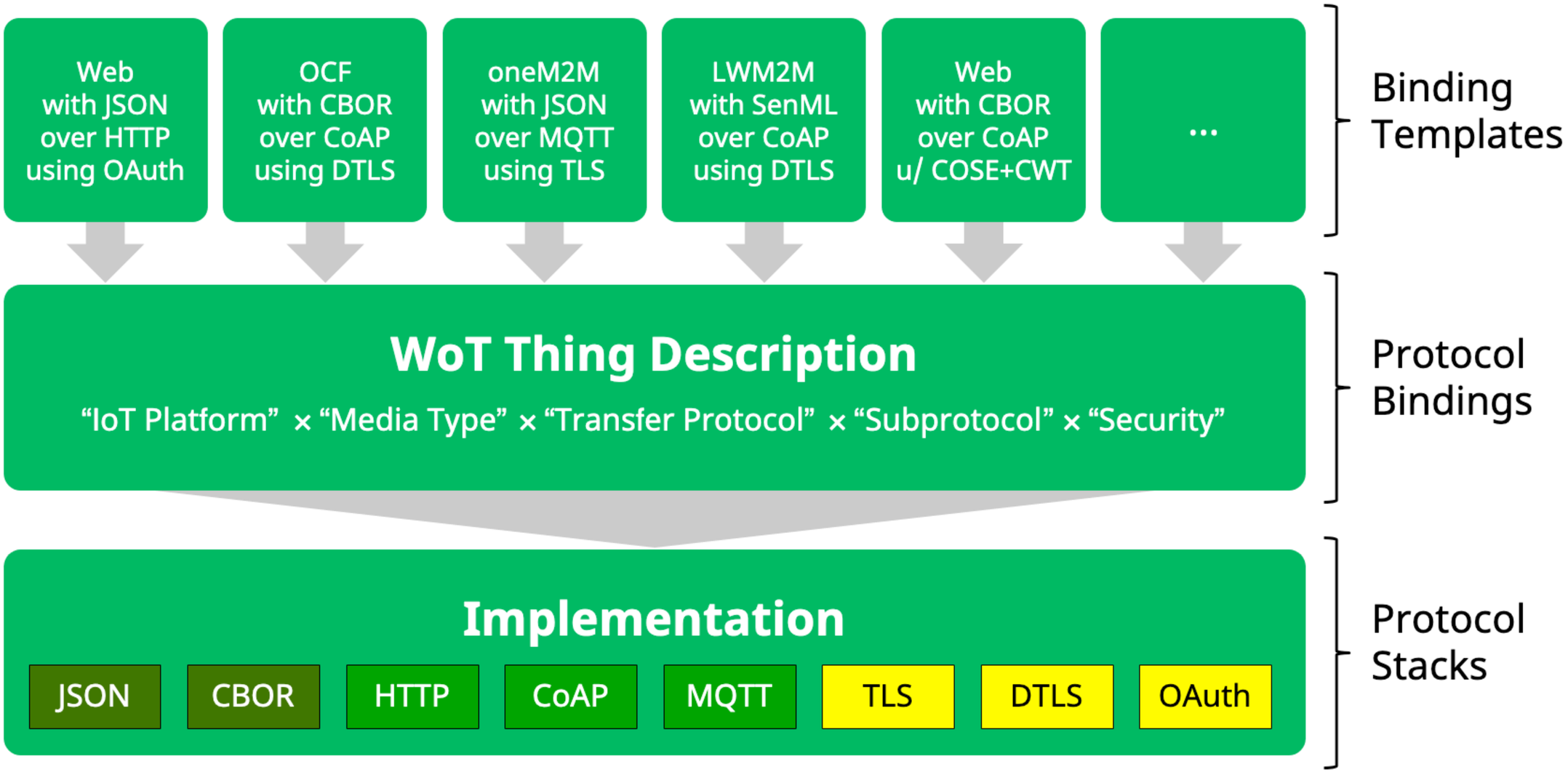
- A Web Thing has four architectural aspects of interest:
  - Behavior
  - Interaction Affordances
  - Data Schemas
  - Security configuration
  - Protocol Bindings



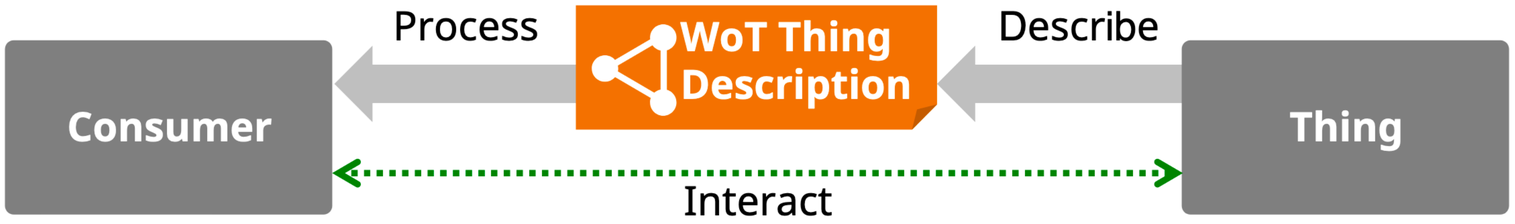
W3C WoT - Architecture - Web Things

- The **Behavior** aspect of a Thing includes both the **autonomous behavior** and the **handlers for the Interaction Affordances**
- The **Interaction Affordances** provide a model of **how** Consumers can **interact** with the Thing through **abstract operations**, but **without reference to a specific network protocol or data encoding**
- A **Data Schema** describes the information model and the related payload structure and corresponding data items that are passed between Things and Consumers during interactions
- The **Protocol Binding** adds the additional detail needed to map each Interaction Affordance to concrete messages of a certain protocol. In general, different concrete protocols may be used to support different subsets of Interaction Affordances, even within a single Thing
- The **Security Configuration** aspect of a Thing represents the mechanisms used to control access to the Interaction Affordances and the management of related Public Security Metadata and Private Security Data.

W3C WoT - Protocol Bindings



W3C WoT - Architecture - Thing Description



- A **Thing** is the abstraction of a physical or virtual entity (e.g., a device or a room) and is described by standardized metadata
- In W3C WoT, the description metadata MUST be a **WoT Thing Description (TD)**
- Consumers MUST be able to parse and process the TD representation format, which is based on JSON
- The format can be processed either through classic **JSON** libraries or a **JSON-LD** (Json for Linked Data) processor, as the underlying information **model** is **graph-based**
- A TD is instance-specific (i.e., describes an individual Thing, not types of Things) and is the default external, textual (Web) representation of a Thing
- There MAY be other representations of a Thing such as an HTML-based user interface, simply an image of the physical entity, or even non-Web representations in closed systems.

W3C WoT - Architecture - Thing Description

- To be a **Thing** (in the W3C WoT Ecosystem), however, at least **one TD representation MUST be available**
- The **WoT Thing Description** is a **standardized**, **machine-understandable representation** format that **allows** Consumers:
  - to **discover** and **interpret** the **capabilities of a Thing** (through semantic annotations)
  - to **adapt** to **different implementations** (e.g., different protocols or data structures) when interacting with a Thing
- The use of **TDs enables the interoperability across different IoT platforms** for example through the us of different ecosystems and standards

W3C WoT - Thing Description- Example

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-WoTLamp-1234",
    "title": "MyLampThing",
    "securityDefinitions": {
        "basic_sc": {"scheme": "basic", "in":"header"}
    },
    "security": ["basic_sc"],
    "properties": {
        "status" : {
            "type": "string",
            "forms": [{"href": "https://mylamp.example.com/status"}]
        }
    },
    "actions": {
        "toggle" : {
            "forms": [{"href": "https://mylamp.example.com/toggle"}]
        }
    },
    "events":{
        "overheating":{
            "data": {"type": "string"},
            "forms": [{
                "href": "https://mylamp.example.com/oh",
                "subprotocol": "longpoll"
            }]
        }
    }
}
```

Property is accessible via (the secure form of) the HTTP protocol with a GET method at the URI https://mylamp.example.com/status (announced within the forms structure by the href member), and will return a string-based status value

To toggle the switch status using the POST method on the https://mylamp.example.com/toggle resource, where POST is again a default assumption for invoking Actions.

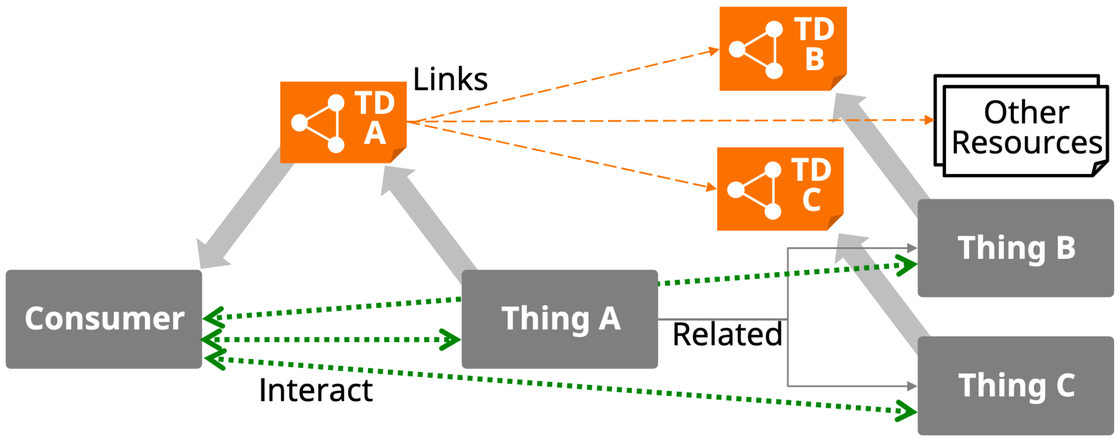
The subscription to be notified upon a possible overheating event of the lamp can be obtained by using HTTP with its long polling subprotocol on https://mylamp.example.com/oh

W3C WoT - Architecture - Things Composition

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- A **Thing** can also be the **abstraction of a virtual entity**
- A virtual entity is the **composition of one or more Things** (e.g., a room consisting of several sensors and actuators)
- One option for the composition is to provide a **single**, **consolidated WoT Thing Description** that contains the superset of capabilities for the virtual entity
- If the **composition** is too **complex**, its TD may link to **hierarchical sub-Things** within the composition
- The main TD acts as entry point and only contain general metadata and potentially overarching capabilities. This allows grouping of certain aspects of more complex Things.

W3C WoT - Architecture - Things Linking

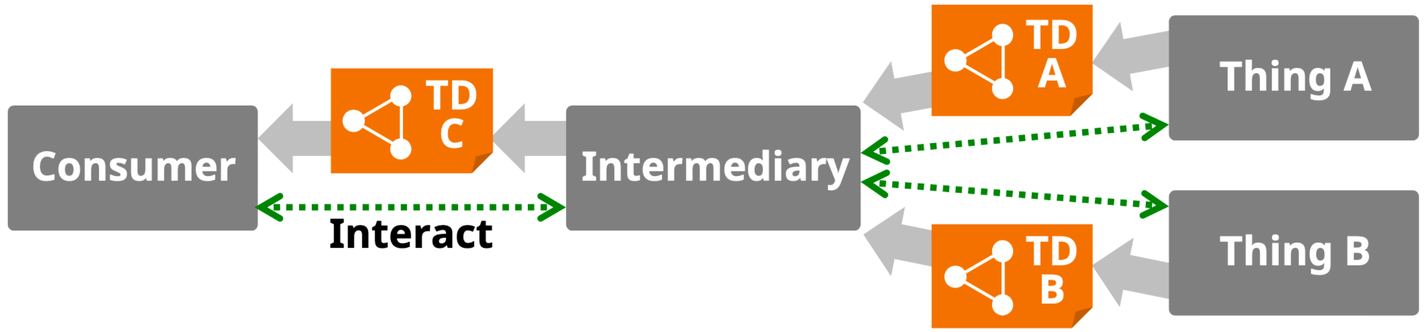


- Linking does not only apply to hierarchical Things, but relations between Things and other resources in general
- Link relation types express how Things relate, for instance, a switch controlling a light or a room monitored by a motion sensor
- Other resources related to a Thing can be manuals, catalogs for spare parts, CAD files, a graphical UI, or any other document on the Web
- Overall, Web linking among Things makes the Web of Things navigable, for both humans and machines
- This can be further facilitated by providing Thing directories that manage a catalog of available Things, usually by caching their TD representation
- WoT Thing Descriptions MAY link to other Things and other resources on the Web to form a Web of Things.

W3C WoT - Architecture - Things Hosting

- Things must be hosted on networked system components with a software stack to realize interaction through a network-facing interface, the **WoT Interface of a Thing**
- One example of this is an HTTP server running on an embedded device with sensors and actuators interfacing the physical entity behind the Thing abstraction
- However, **W3C WoT does not mandate where Things are hosted**; it can be on the IoT device directly, an Edge device such as a gateway, or the cloud
- A typical deployment challenge is a scenario, where local networks are not reachable from the Internet, usually because of IPv4 Network Address Translation (NAT) or firewall devices. To remedy this situation, W3C WoT allows for Intermediaries between Things and Consumers

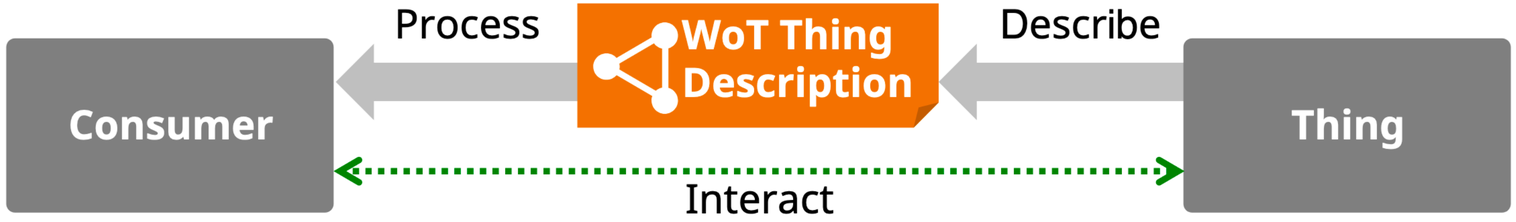
W3C WoT - Architecture - Things Hosting



- Intermediaries can act as proxies for Things, where the Intermediary has a WoT Thing Description similar to the original Thing, but which points to the WoT Interface provided by the Intermediary
- I**ntermediaries may also augment existing Things with additional capabilities or compose a new Thing out of multiple available Things**, thereby forming a virtual entity
- **To Consumers, Intermediaries look like Things**, as they possess WoT Thing Descriptions and provide a WoT Interface, and hence might be indistinguishable from Things in a layered system architecture like the Web [REST]
- An identifier in the WoT Thing Description MUST allow for the correlation of multiple TDs representing the same original Thing or ultimately unique physical entity

W3C WoT - System Components and Their Interconnectivity

Abstract



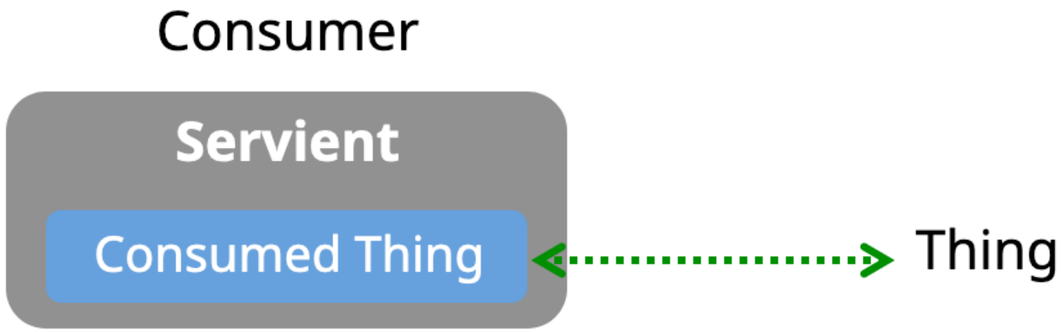
Software Implementation

- We have seen and described the WoT architecture in terms of the **abstract** WoT architecture components such as **Things**, **Consumers** and **Intermediaries**
- When those abstract WoT architecture components are implemented as a software stack to take a specific role in the WoT architecture, such software stacks are called **Servients**
- Systems that are based on the WoT architecture involve **Servients**, which are communicating with each other to achieve the goals of a system.

W3C WoT - Thing, Servient & Exposed Thing

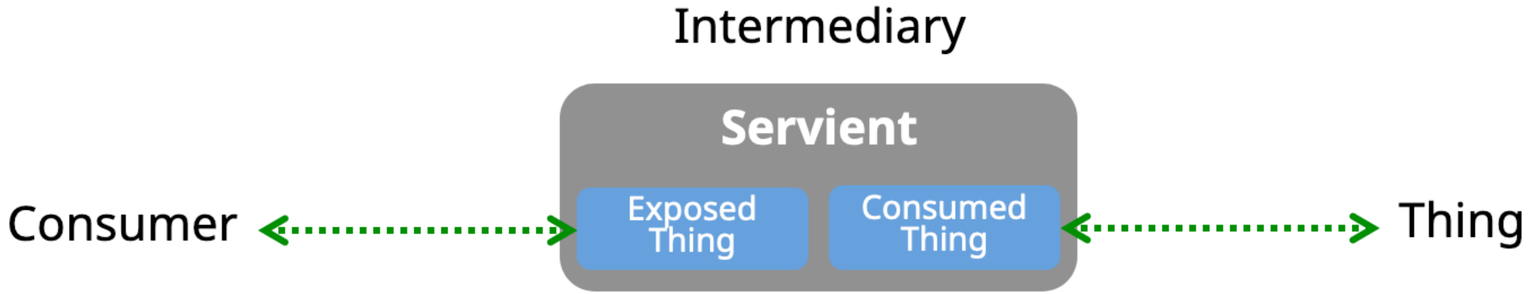
- A **Thing** can be implemented by a **Servient**
- In a **Thing**, a **Servient** software stack contains a representation of a **Thing** called **Exposed Thing**
- It makes its **WoT Interface** available to **Consumers** of the **Thing**
- The **Exposed Thing** may be used by other software components on the **Servient** (e.g., applications) to implement the behavior of the thing.

W3C WoT - Consumer, Servient & Consumed Thing



- Consumers **are always implemented by Servients**, as they must be able to process the Thing Description (TD) format
- They must also have a protocol stack that can be configured through Protocol Binding information contained in the TDs
- In a Consumer, a Servient software stack provides a representation of a Thing called **Consumed Thing**, and makes it available to those applications running on the Servient that need to process TDs to interact with Things
- A Consumed Thing instance in the Servient software stack **serves to separate the protocol level complexity from applications**. It is communicating with Exposed Things on behalf of the application.

W3C WoT - Consumer, Servient & Consumed Thing

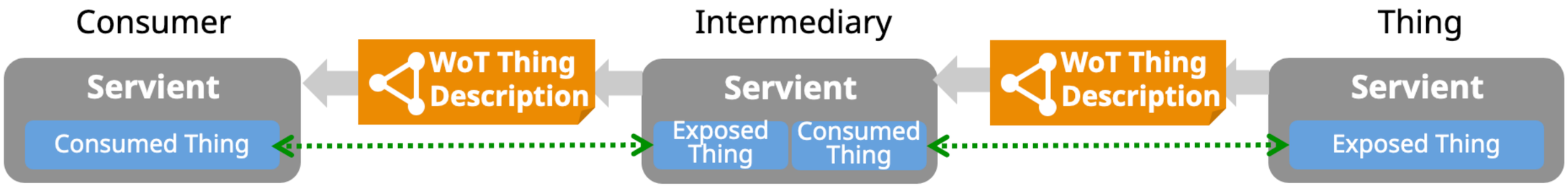


- Similarly, an **Intermediary** is yet another WoT architecture component implemented by a Servient
- An Intermediary is **located between a Thing and its Consumers**
- It performs the roles of both a Consumer (to the Thing) and a Thing (to the Consumers)
- In an Intermediary, a Servient software stack contains the representations of both a Consumer (Consumed Thing) and a Thing (Exposed Thing)

W3C WoT - Direct Communication

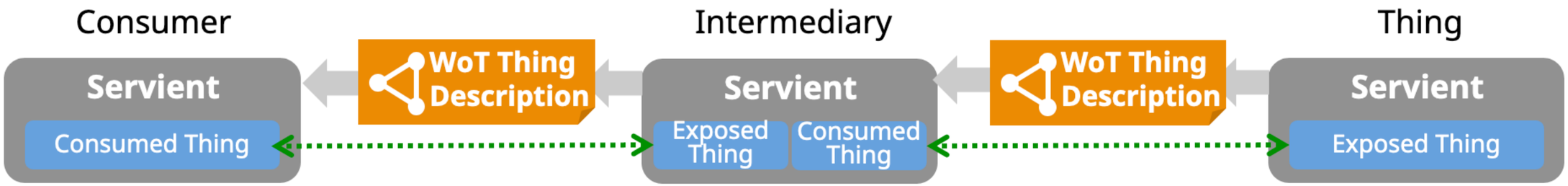
- The Figure shows the **direct communication** between a **Thing**, which is exposing Interaction Affordances through Thing Descriptions, and a **Consumer** that uses the Thing by means of the Interaction Affordances
- **Direct communication applies** when both Servients **use the same network protocol(s) and are accessible to each other**
- An **Exposed Thing is the software representation of a Thing abstraction**, serving a WoT Interface of the Interaction Affordances provided by the Thing.
- **A Consumed Thing is the software representation of a remote Thing being consumed by a Consumer**, serving as the **interface to the remote Thing for the applications**
- A **Consumer** can **generate a Consumed Thing** instance by **parsing and processing a TD document**
- Interactions between a Consumer and a Thing are performed by the Consumed Thing and the Exposed Thing **exchanging messages over a direct network connection between them**

W3C WoT - Indirect Communication 1/2



- **An Intermediary is required if the Servients use different protocols or if they are on different networks that require authentication and provide access control (e.g. firewalls).**
- An **Intermediary combines Exposed Thing and Consumed Thing** functionality.
- The functionality of Intermediaries includes:
  - **relaying** messages for the Interaction Affordances between a Consumer and a Thing
  - optionally **caching** the Thing's data for faster response
  - **transforming** communication when the functionality of the Thing is extended by the Intermediary
- In an Intermediary, a Consumed Thing creates a proxy object of the Exposed Thing of a Thing, and a Consumer can access the proxy object (i.e., the Exposed Thing of the Intermediary) through its own Consumed Thing.
- **Consumer and Intermediary can communicate in a different protocol than Intermediary and Thing.** For example, an Intermediary can provide a bridge between a Thing that uses CoAP and the application of a Consumer that uses HTTP.

W3C WoT - Indirect Communication 2/2



- Even when there are multiple different protocols used between Intermediary and Things, **Consumer can indirectly communicate with those Things using a single protocol through the Intermediary**
- The same is true for the authentication. **The Consumed Thing of a Consumer only needs to authenticate with the Exposed Things of the Intermediary using a single security mechanism, while the Intermediary might need multiple security mechanism to authenticate with different Things.**
- Usually, an **Intermediary generates the Thing Description for its proxy object based on the Thing Description of the originating Thing**. Depending on the requirements of the use cases, the TD for the proxy object may either use the same identifier as the TD of the original Thing, or it gets assigned a new identifier. **If necessary, a TD generated by an Intermediary MAY contain interfaces for other communication protocols.**

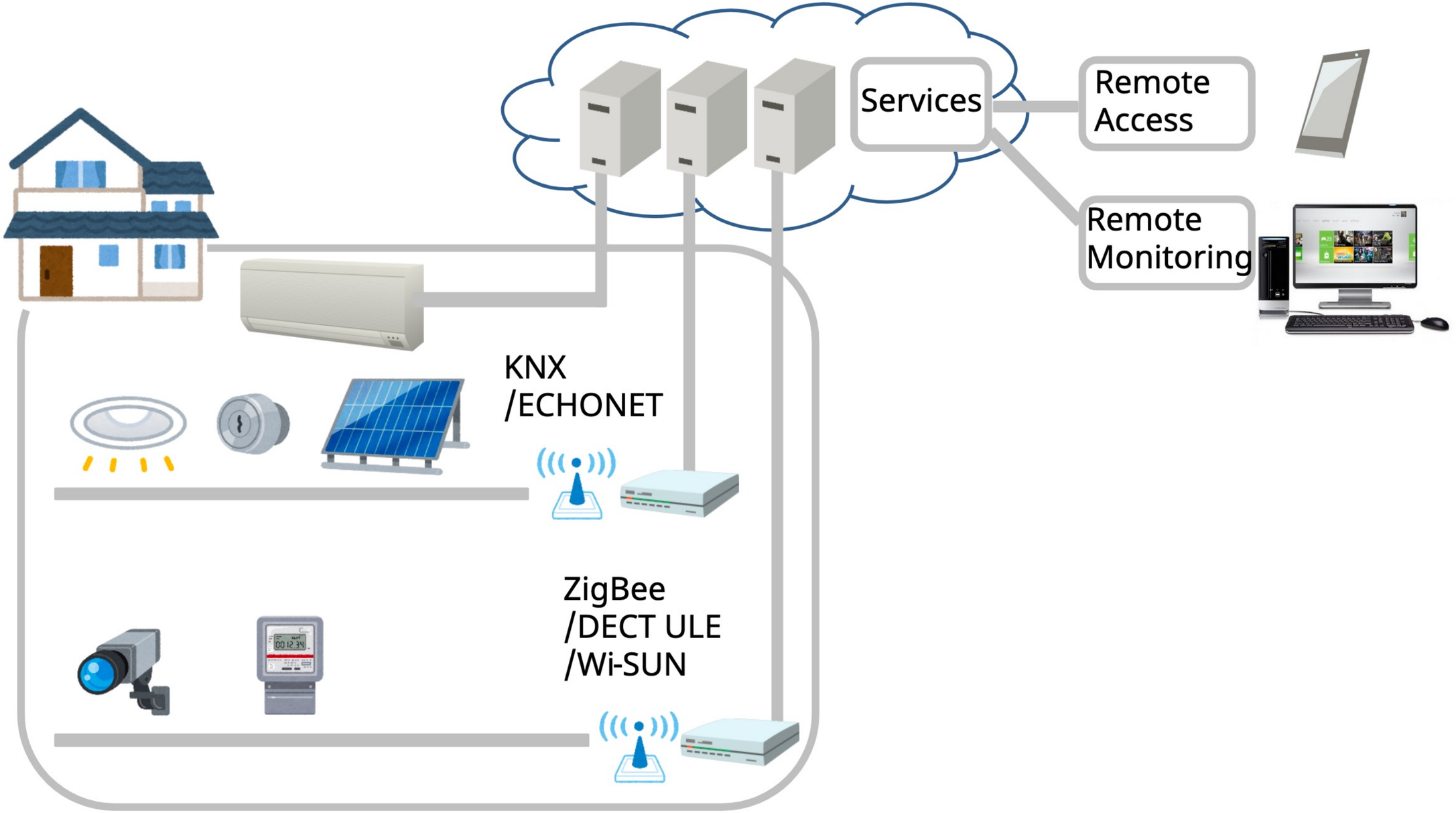
W3C WoT - Use Cases & Application Domains

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Main considered target Use Cases and Application Domains are related to:
  - Consumer
  - Industrial
  - Transportation & Logistics
  - Utilities
  - Oil and Gas
  - Insurance
  - Engineering and Construction
  - Agriculture
  - Healthcare
  - Environment Monitoring
  - Smart Cities
  - Smart Buildings
  - Connected Cars
  - and counting …

https://www.w3.org/TR/wot-architecture/

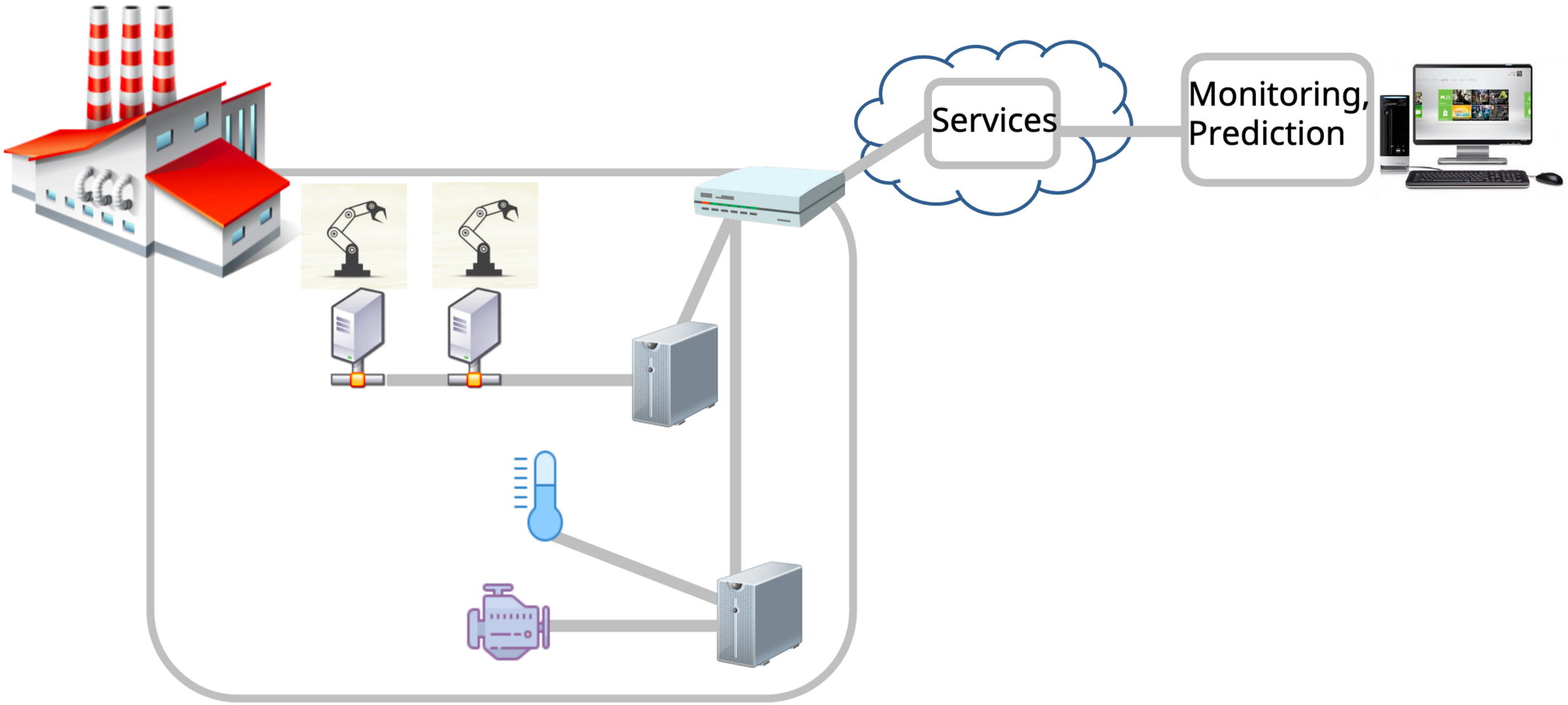
W3C WoT - Use Cases - Consumer



W3C WoT - Use Cases - Consumer (e.g. Smart Home)

- In the consumer space there are multiple **assets** that benefit from being **connected** such as remote access and control, voice control and home automation. Furthermore, they also enables device manufacturers to monitor and **maintain devices remotely** in order to realize value added services such as energy management and security surveillance
- For example in the Smart Home context lights and air conditioners can be turned off based on room occupancy. Window blinds can be closed automatically based on weather conditions and presence. Energy and other resource consumption can be optimized based on usage patterns and predictions.
- The previously illustrated Use Case shows an example of a Smart Home. In this case, gateways are connected to edge devices such as sensors, cameras and home appliances through corresponding local communication protocols such as KNX, ECHONET, ZigBee, DECT ULE and Wi-SUN. **Multiple gateways can exist in one home, while each gateway can support multiple local protocols**
- Gateways can be connected to the cloud through the internet, while some appliances can be connected to the cloud directly. Services running in the cloud collect data from edge devices and analyze the data, then provide value to users through the edge devices and other UX devices

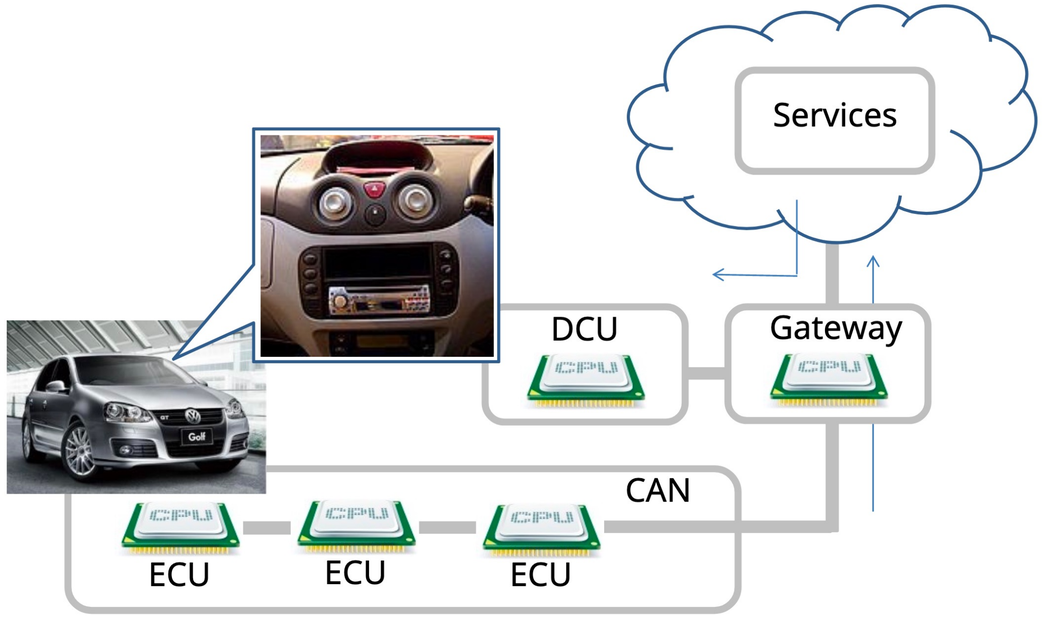
W3C WoT - Use Cases - Industrial



W3C WoT - Use Cases - Industrial

- Smart factories require advanced monitoring of the connected manufacturing equipment as well of the manufactured products. They benefit from predictions of machine failures and early discovery of anomalies to prevent costly downtime and maintenance efforts.
- Additionally, monitoring of connected manufacturing equipment and the environment at the production facility for the presence of poisonous gases, excessive noise or heat increases the safety of the workers and reduces the risks of incidents or accidents.
- Real-time monitoring and KPI calculations of production equipment helps to detect productivity problems and optimize the supply chain.
- In previous example, field-level, cell and line controllers automate different factory equipment based on industrial communication protocols such as PROFINET, Modbus, OPC UA TSN, EtherCAT, or CAN. An industrial edge device collects selected data from various controllers and makes it available to a cloud backend service, e.g., for remote monitoring via a dashboard or analyzes it for preventive maintenance.

W3C WoT - Use Cases - Connected Car



- In this context, a gateway connects to car components through CAN and to the car navigation system through a proprietary interface
- Services running in the cloud or on the edge (e.g., 5G MEC) collect data pushed from car components and analyze the data from multiple cars to determine traffic patterns
- The gateway can also consume external services, in this case, to get traffic data and show it to the driver through the car navigation system.

W3C WoT – Discovery Approach

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- WoT Discovery allows authenticated and authorized entities to find WoT Thing Descriptions satisfying a set of criteria, such as having certain semantics, or containing certain interactions
- On the other hand, in order to support security and privacy objectives, the WoT Discovery process must not leak information to unauthorized entities. This includes leaking information that a given entity is requesting certain information, not just the information distributed in the Thing Descriptions themselves
- In the literature there are several approaches associated to IoT discoverability as reported and described in:
  - Arne Bröring, Soumya Kanti Datta, and Christian Bonnet. 2016. A Categorization of Discovery Technologies for the Internet of Things. In Proceedings of the 6th International Conference on the Internet of Things (IoT'16). Association for Computing Machinery, New York, NY, USA, 131–139. https://doi.org/10.1145/2991561.2991570
  - Simone Cirani; Luca Davoli; Gianluigi Ferrari; Rémy Léone; Paolo Medagliani; Marco Picone; Luca Veltri., "A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things," in IEEE Internet of Things Journal, vol. 1, no. 5, pp. 508-521, Oct. 2014, doi: 10.1109/JIOT.2014.2358296.

Source: https://www.w3.org/TR/wot-discovery/

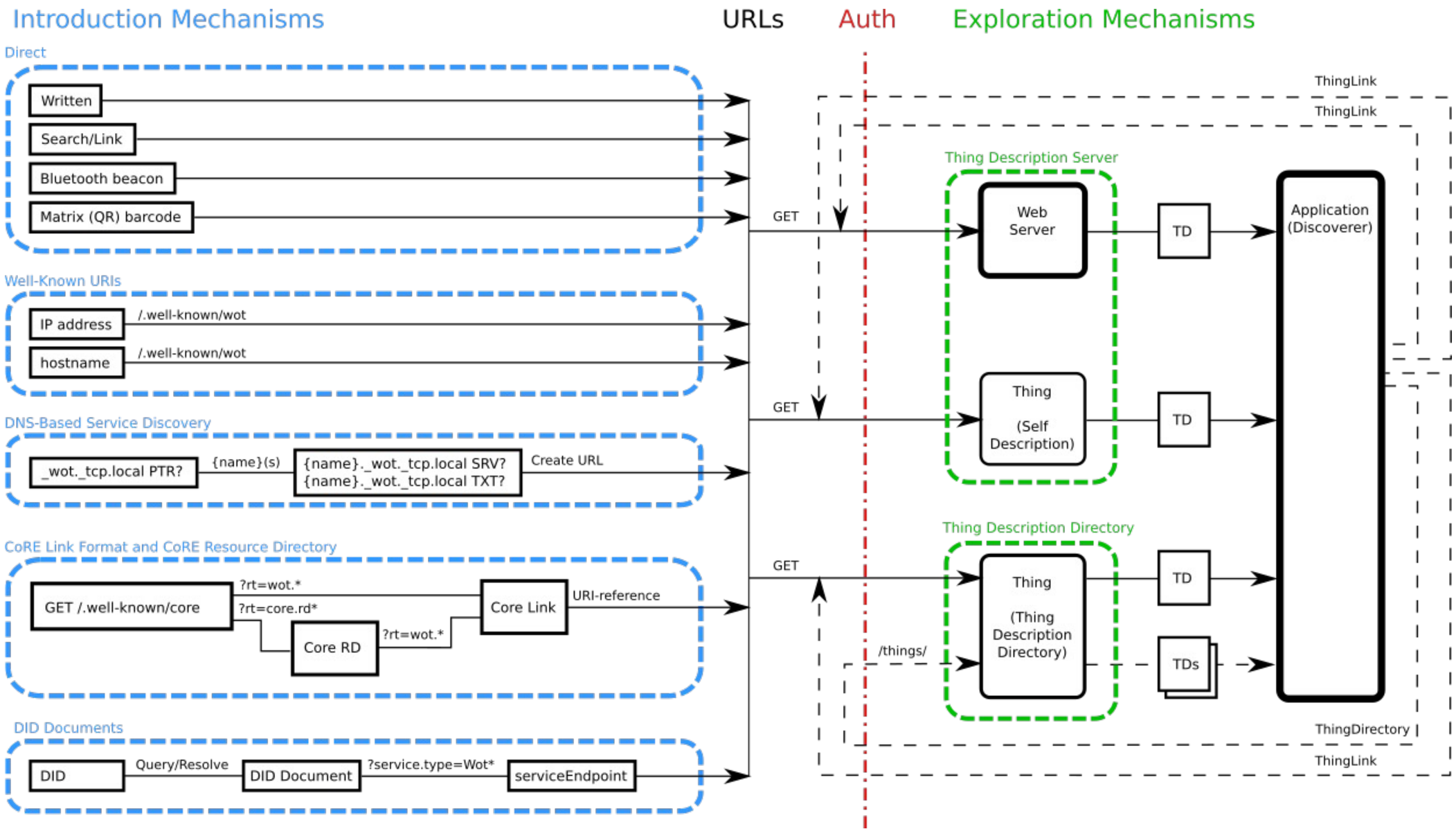
W3C WoT – Discovery Approach

- One of the main WoT objectives is to establish a mechanism that not only uses best practices to protect metadata, but that can be upgraded to support future developments and best practices as needed
- WoT takes into account discovery in a broad sense to include both **local and non-local mechanisms:**
  - local mechanism might use a broadcast protocol
  - non-local mechanisms might go beyond the current network segment where broadcast is not scalable, and so a different approach, such as a search service, is needed
- WoT approach is to **use existing mechanisms as needed to bootstrap into a more general and secure metadata distribution system**

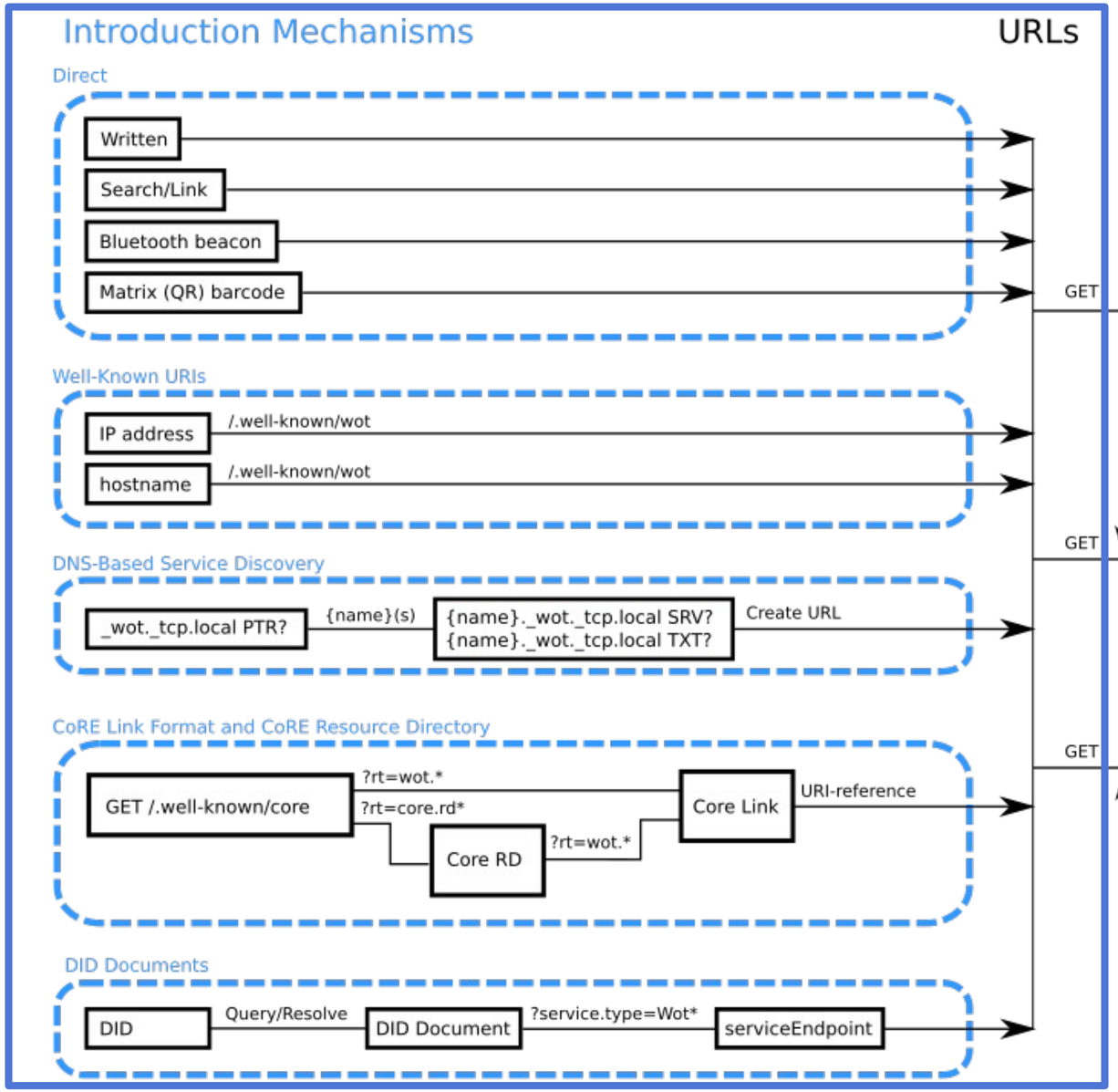
W3C WoT – Discovery Approach

- The envisioned approach is to balance between simple structures such as **key-value pairs** and **advanced and complex query approaches** (e.g., as SPARQL approach) potentially suitable for some advanced use cases, might require too much effort for many anticipated IoT applications
- The idea is to adopt a new and effective query approach **balancing simpler query mechanism and a way to exploit WoT capabilities and functionalities** supported by the WoT Thing Description in terms of highly structured data schemas and semantic annotations

W3C WoT – Discovery Approach



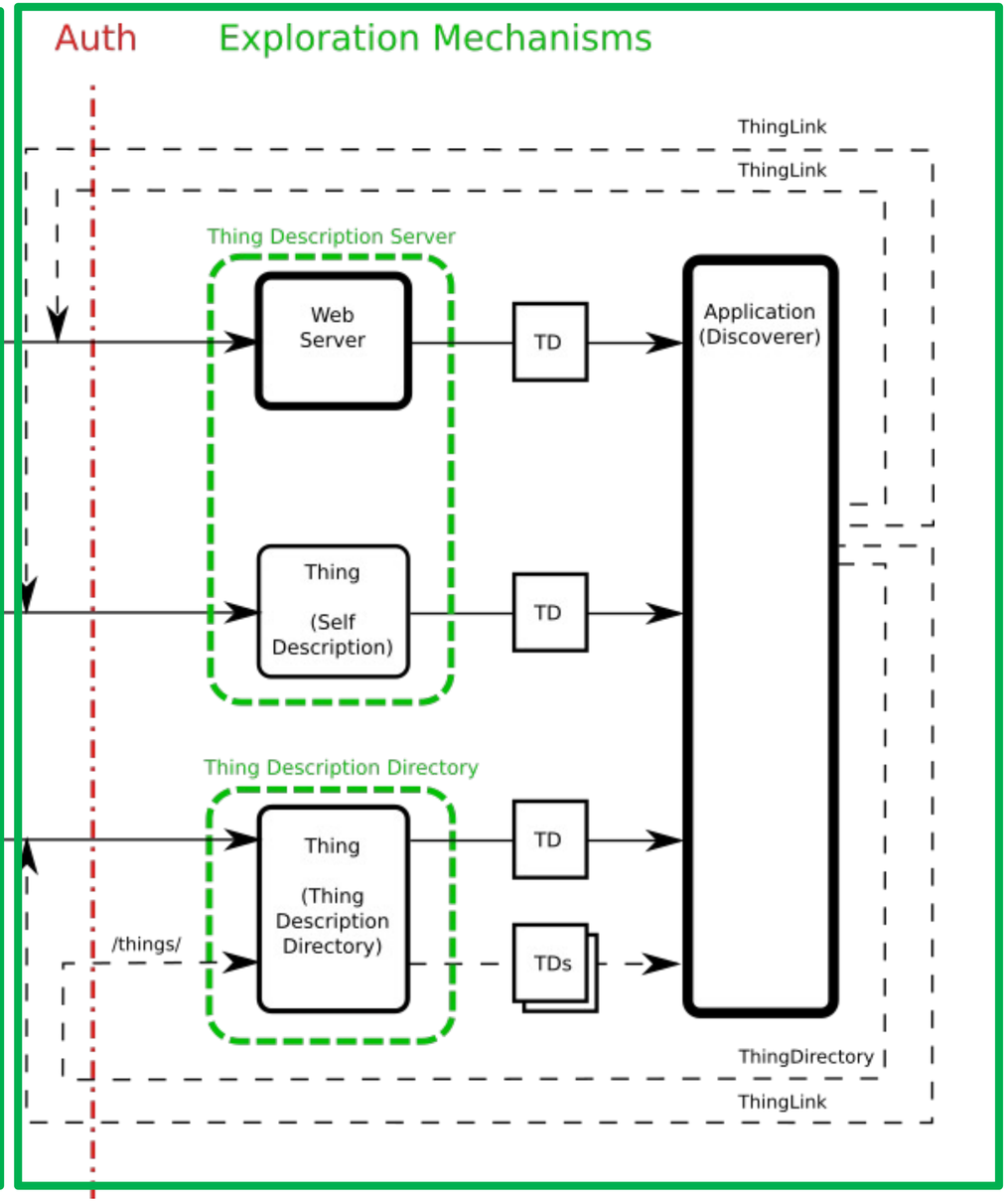
W3C WoT – Discovery Approach



- The intention is that Introduction mechanisms are relatively open **“first contact” mechanisms** to **provide a starting point** for the rest of the Discovery process
- Introductions can be any mechanism that can return a URL
- The URLs reference to **“Exploration”** services pointing to a Thing Description document

W3C WoT – Discovery Approach

- Each URL provided by the Introduction phase always points at an **Exploration service endpoint** returning a single **Thing Description**
- In general Thing Descriptions might be provided in various ways and in particular may not be self-describing. For example:
  - devices not designed with the WoT specification in mind will not be capable of serving their own TD;
  - battery-powered devices may not be online most of the time;
  - some devices may not be powerful enough to manage and serve TDs directly.
- The Thing Description for such Things should be provided by separate services.
- A TD referenced by an Introduction URL may describe a **Thing Description Directory (TDD)** service responsible for maintaining a (possibly dynamic) database of TDs supporting also queries that can be used to selectively retrieve TDs.
- The second special case is a **Thing Link.** It is also a TD that holds a link to a TD hosted elsewhere. A TDD can also store Thing Links which can redirect to other TDDs, allowing for a **linked directory structure**.



W3C WoT – Introduction mechanisms

- **Introduction mechanisms** are meant to be ***"first contact"*** solutions to provide a starting point for the rest of the Discovery process
- These methods should be designed to be suitable for different use cases, including both local and non-local scenarios but at the end they can in fact be provided by **any mechanism that can return a URL**
- Introductions, **do not include any security or privacy controls** and so **should not provide metadata directly**
- Provided URLs provided should point to **"Exploration Services"**
- **Exploration services** actually do **provide metadata**, but only **after suitable authentication and access controls have been applied**

W3C WoT – Discovery - Introduction Mechanisms

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- **Direct:** To obtain an URL of an exploration service, any mechanism that results in a single URL MAY be used. This includes:
  - Bluetooth beacons, QR codes, and written URLs to be typed by a user
  - A **request** on all such URLs **MUST result in a TD** for self-describing Things, this can be the TD of the Thing itself
  - If the URL references a Thing Description Directory, this MUST be the Thing Description of the Thing Description Directory

W3C WoT – Discovery - Introduction Mechanisms

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- **Well-Known URIs**: A Thing or Thing Description Directory MAY use the Well-Known Uniform Resource Identifier **[RFC8615]** to advertise its presence.
- If a Thing or Thing Description Directory use the Well-Known Uniform Resource Identifier [RFC8615] to advertise its presence, it MUST register its own Thing Description into the following path: /.well-known/wot. When a request is made at the above Well-Known URI, the server MUST return a Thing Description as prescribed in 7. Exploration Mechanisms.

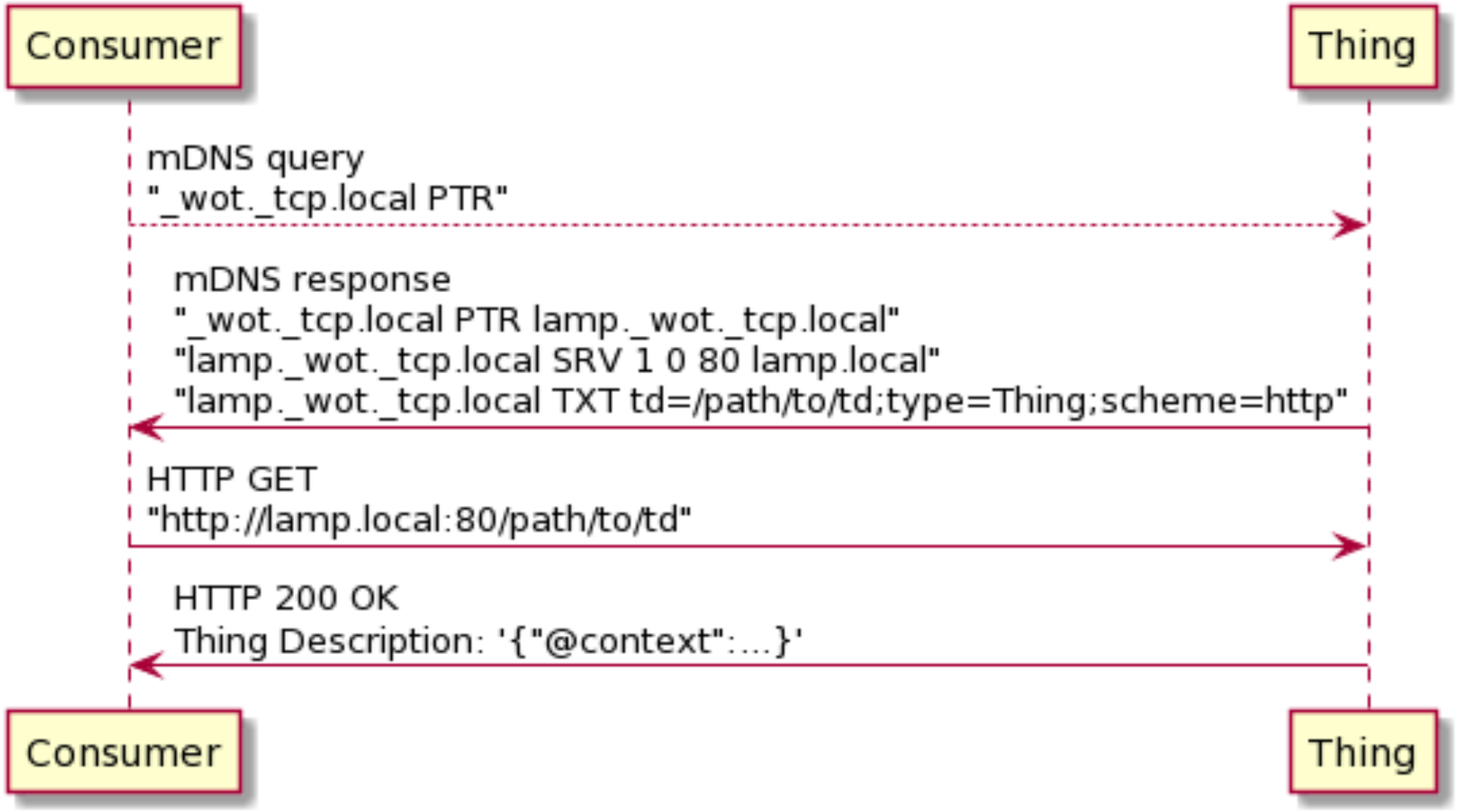
W3C WoT – Discovery - Introduction Mechanisms

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- **DNS-Based Service Discovery:** A Thing or Thing Description Directory MAY use DNS-Based Service Discovery **(DNS-SD) [RFC6763]**.
- This can be also be used on the same local network in combination with Multicast DNS **(mDNS) [RFC6762]**.
- The following table lists the service names and protocols for advertising their presence

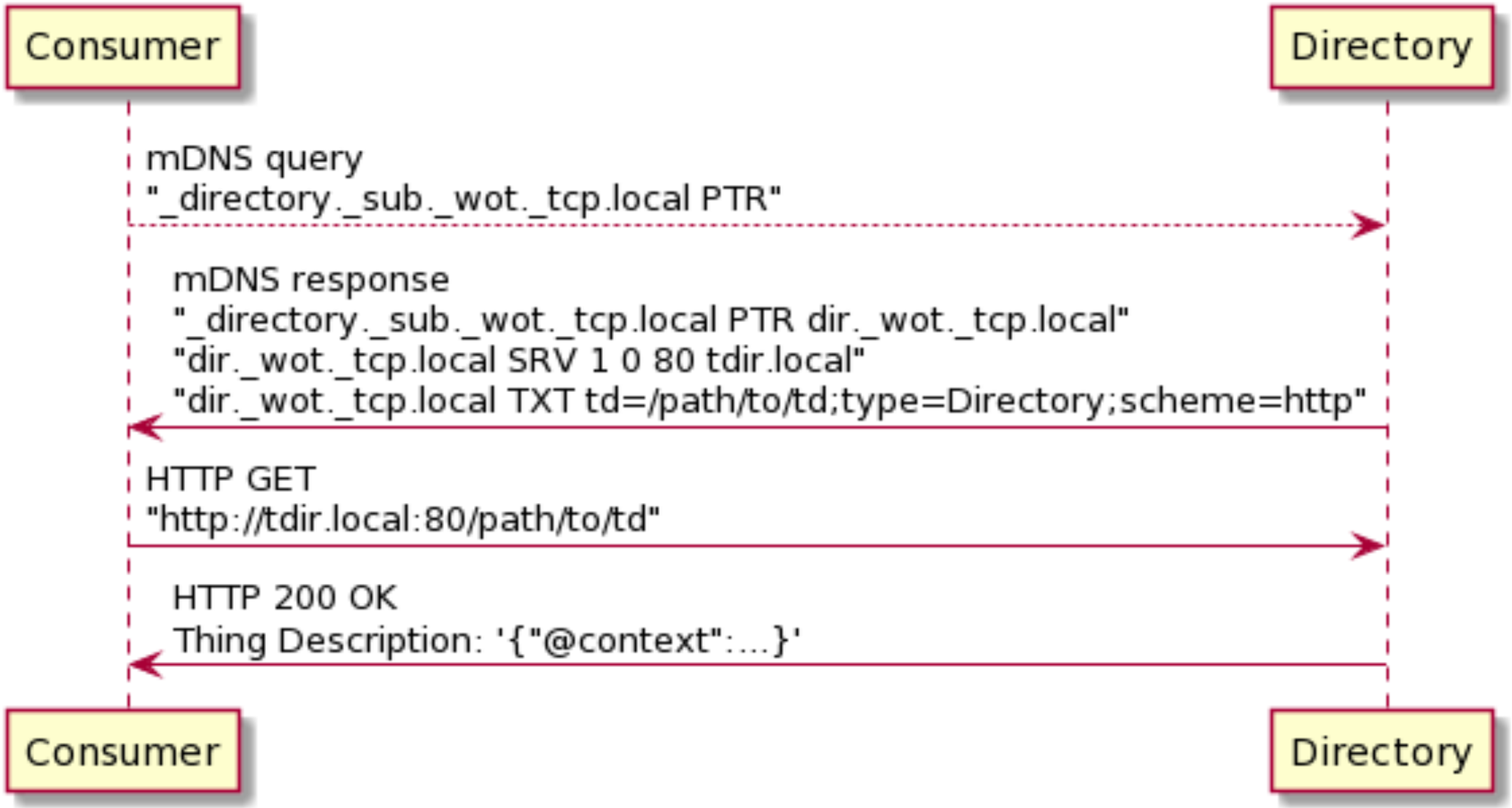
Service name	*Thing or TDD*	*Protocol*
`_wot._tcp`	Thing	HTTP over TCP, HTTP over TLS/TCP, CoAP over TCP, or CoAP over TLS/TCP
`_directory._sub._wot._tcp`	TDD	HTTP over TCP, HTTP over TLS/TCP, CoAP over TCP, or CoAP over TLS/TCP
`_wot._udp`	Thing	CoAP over UDP or CoAP over DTLS/UDP

W3C WoT – Discovery - DNS-Based



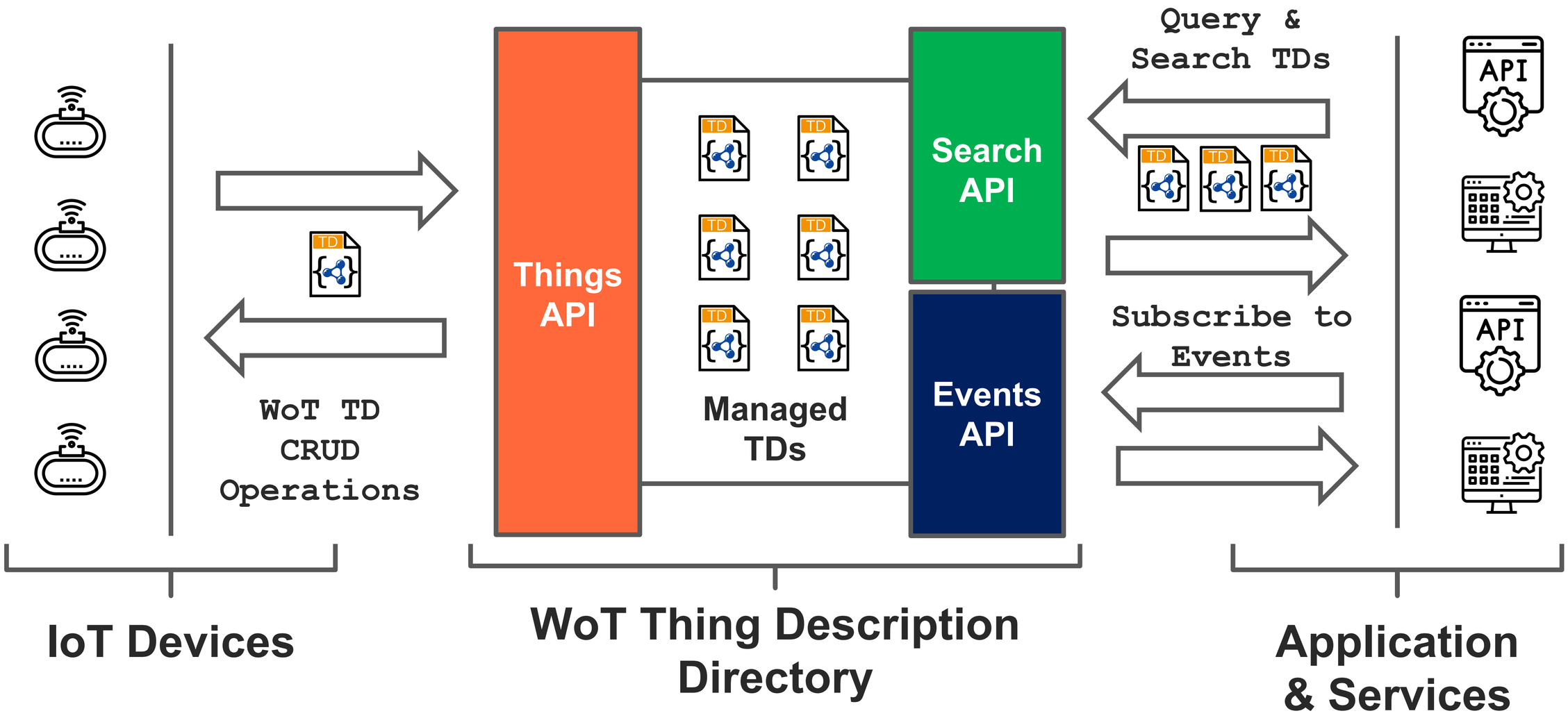
Using mDNS for discovery of the Thing Description of a Thing

W3C WoT – Discovery - DNS-Based

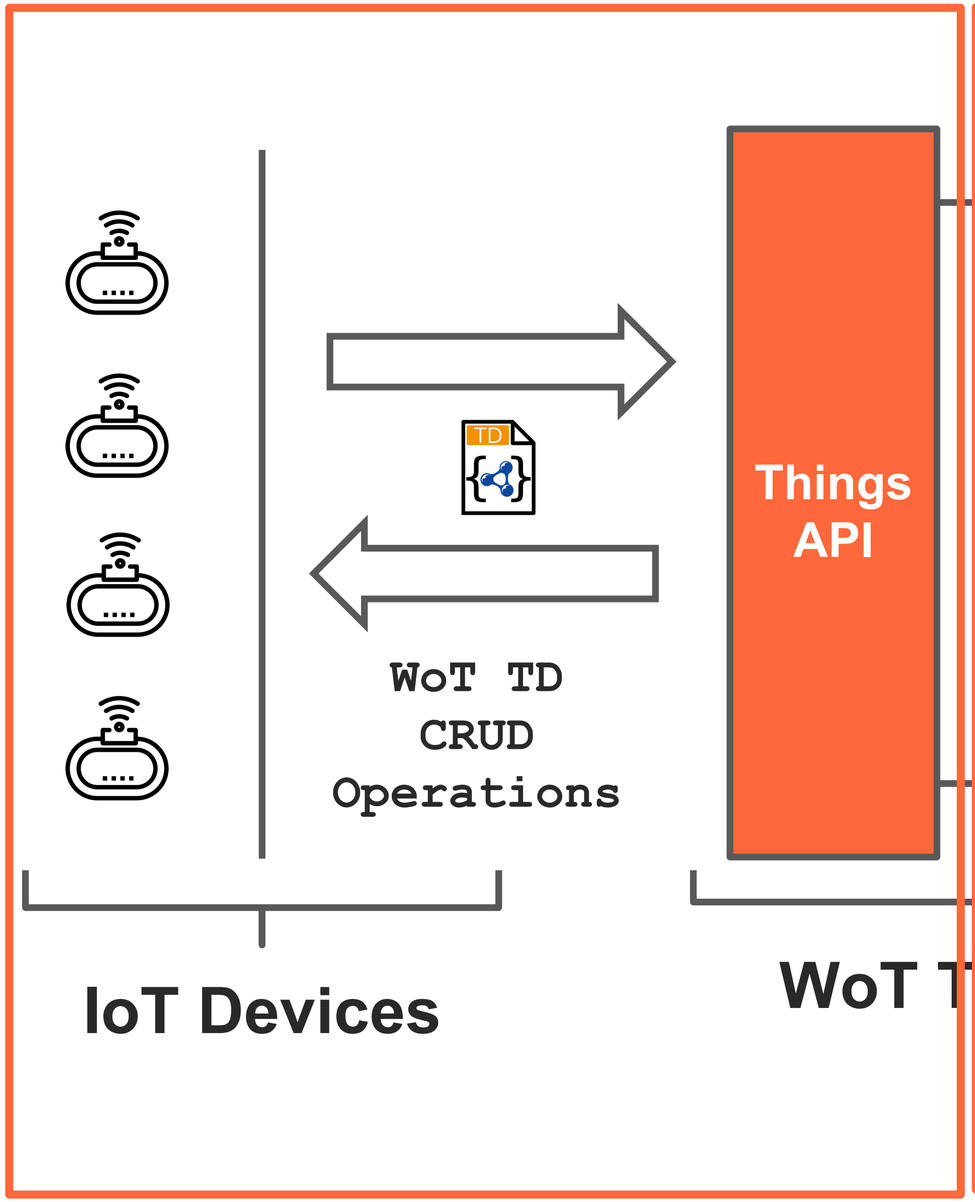


Using mDNS for discovery of the Thing Description of a Thing Description Directory (TDD)

W3C WoT – Thing Description Directory



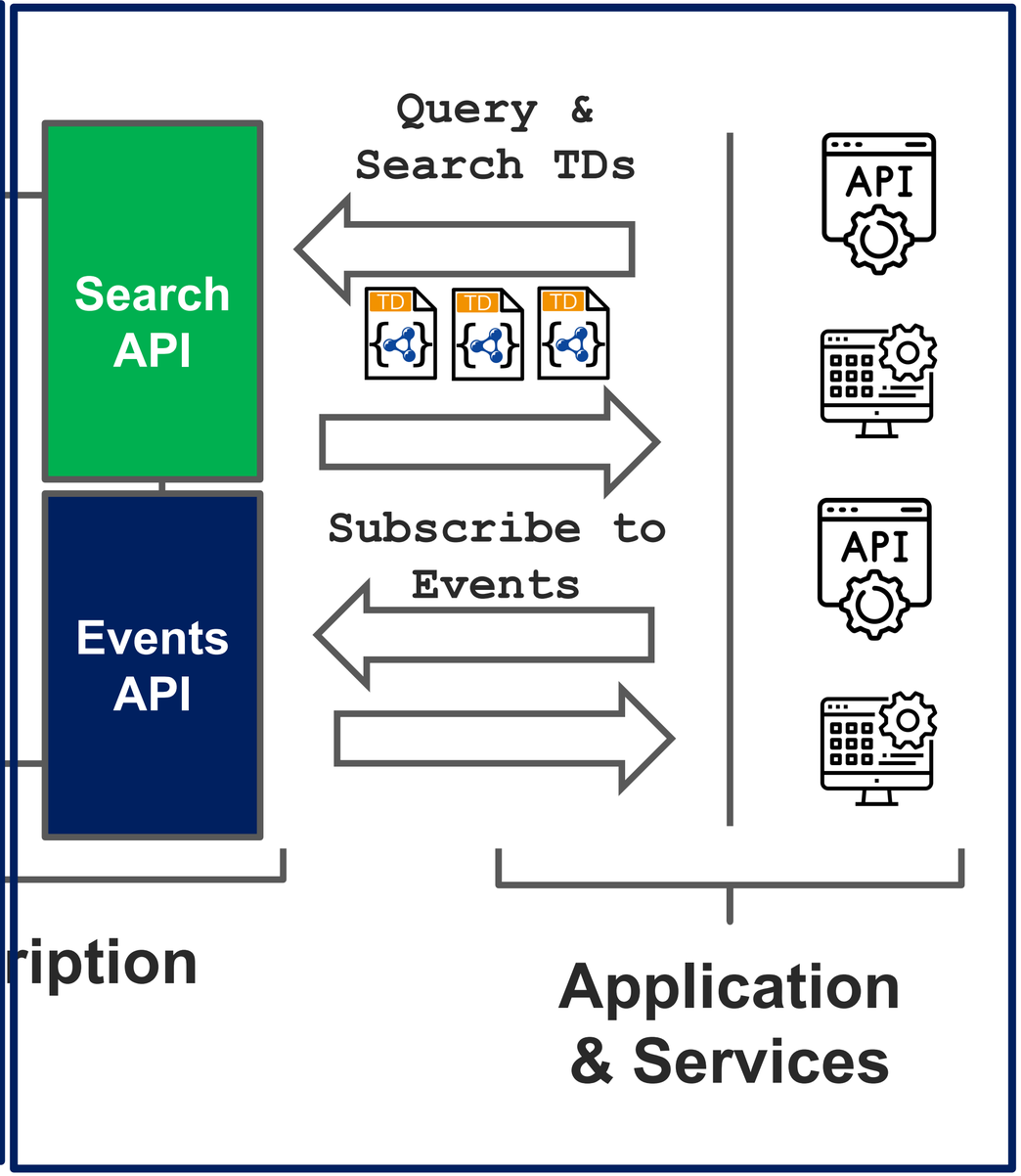
W3C WoT – Thing Description Directory



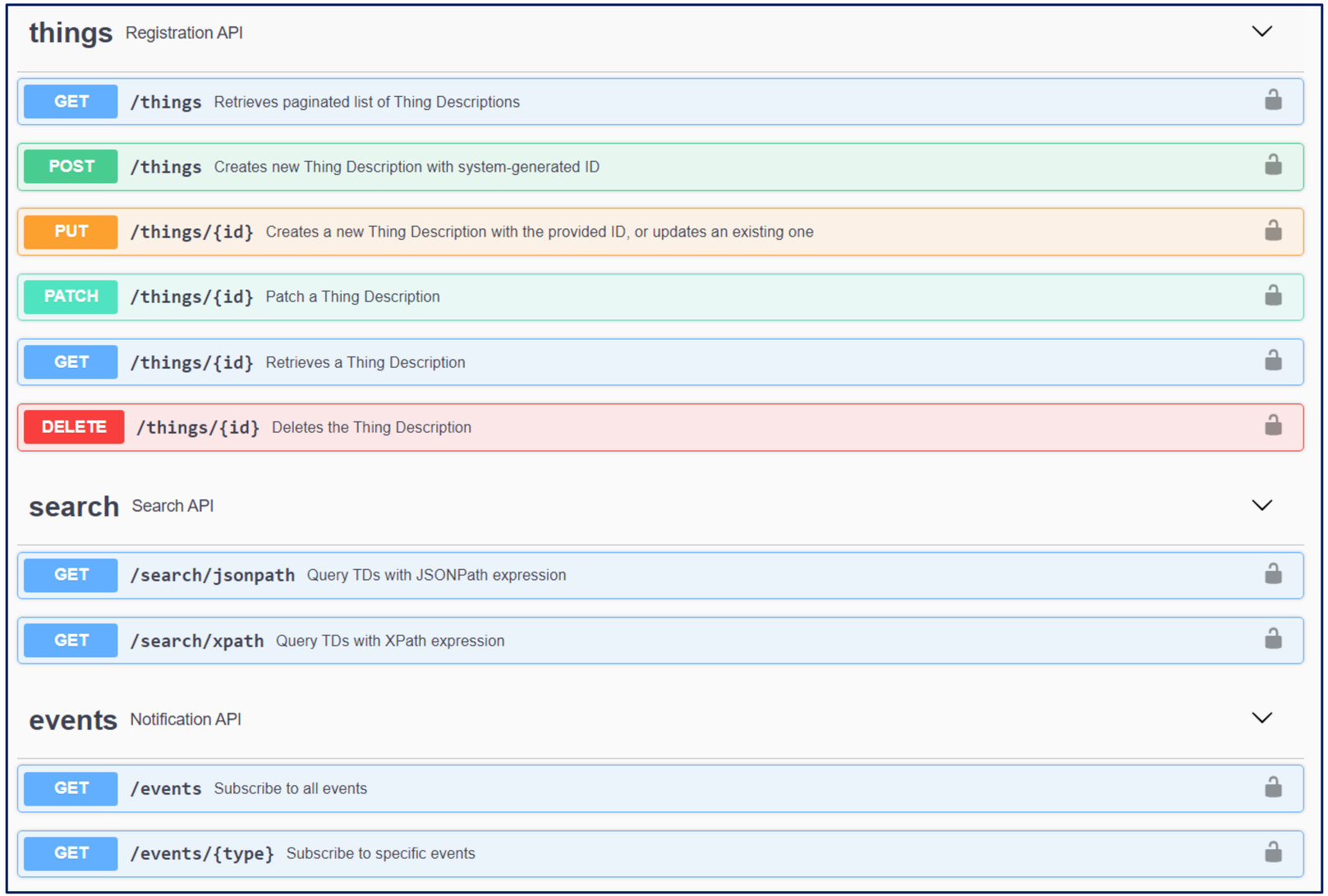
- The **Things API** is a **RESTful HTTP API** served at the **/things** endpoint providing interfaces to **create, retrieve, update, delete, and list (CRUDL) TDs.** The design of this API is in accordance with [RFC7231] and [REST-IOT].
- The API MUST provide the **interface to list TDs.** The Search API allows filtering and selection from this list; see 7.3.2.3 Search API.
- The API MAY provide the **interfaces to create, read, update, and delete (CRUD) individual TDs.**
- The default serialization format for all request and success response bodies MUST be JSON, with JSON-LD (Json Linked Data) 1.1 [JSON-LD11] syntax to support extensions and semantic processing
- Directories MAY accept alternative representations based on request's indicated Content-Type or Content-Encoding headers. This is useful for applications that need to provide representations other than raw JSON as input to a directory.
- Directories MAY provide alternative representations through server-driven content negotiation, that is by honouring the request's Accept and Accept-Encoding headers and responding with the supported TD representation and equivalent Content-Type and Content-Encoding headers. This is useful for applications that need to retrieve representations other than raw JSON from a directory, such as Gzip-compressed JSON

W3C WoT – Thing Description Directory

- The **Event API** is to notify clients about the **changes to TDs maintained within the directory**
- The Notification API MUST follow the **Server-Sent Events (SSE)** specifications to serve events to clients at **/events** endpoint
- In particular, the server responds to successful requests with **200 (OK) status and text/event-stream Content Type**
- Re-connecting clients may continue from the last event by providing the last event ID as Last-Event-ID header value. The server SHOULD provide an event ID as the id field in each event and respond to re-connecting clients by delivering all missed events.
- **Event Types:** The server MUST produce events attributed to the lifecycle of the Thing Descriptions within the directory using **thing_created**, **thing_updated**, and **thing_deleted** event types.
- **Event Filtering:** The API enables server-side filtering of events to reduce resource consumption by delivering only the events required by clients. Client libraries may offer additional filtering capabilities on the client-side. The server MUST support event filtering based on the event type given by the client upon subscription. For example, given the URI Template **/events{/type}:**
  - **/events/thing_created** instructs the server to only deliver events of type thing_created
  - **/events instructs** the server to deliver all events
- The clients need to subscribe separately to receive a subset of the events (e.g. only thing_created and thing_deleted) from the server. When using HTTP/2, multiple subscriptions on the same domain (HTTP streams) get multiplexed on a single connection.
- **Event Data:** The event data MUST **contain the JSON serialization of the event object**. The **event data object is a Partial TD or the whole TD object depending on the request**. The event data object MUST at least include the identifier of the TD created, updated, or deleted at that event in Partial TD form.

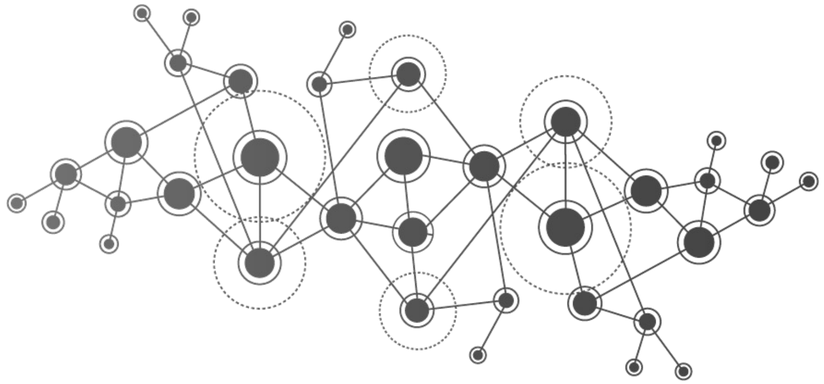


W3C WoT – Thing Description Directory (Example)



LinkSmart Project Official Repo: https://github.com/linksmart/thing-directory
Swagger API Explorer: http://linksmart.eu/swagger-ui/dist/?url=https://raw.githubusercontent.com/linksmart/thing-directory/master/apidoc/openapi-spec.yml#/

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA



Intelligent Internet of Things

Interoperable IoT Architectures Web of Things

Prof. Marco Picone

A.A 2023/2024