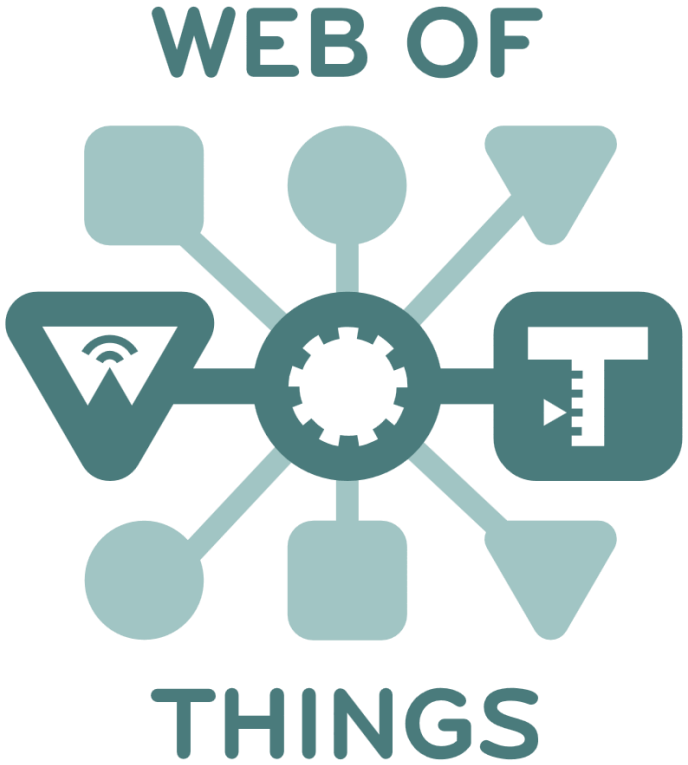
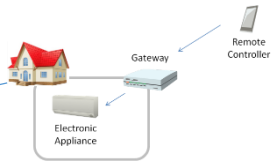


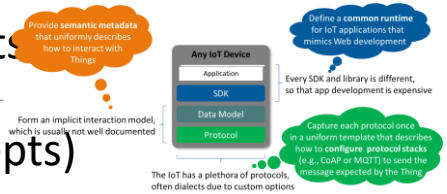
W3C WoT architecture

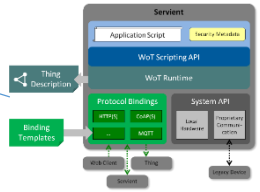
contents

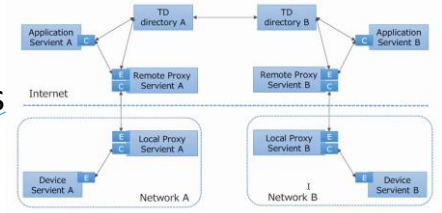
リヨン会議での目次案

1. Introduction
2. Terminology
3. Use Cases
4. Functional Requirements
5. Abstract Architecture (Terminology and concepts) Message flows
6. WoT Building Blocks (Specifications)
7. WoT Servients (Implementation) Lifecycle
8. WoT Deployments
    1. Device Servients
    2. Proxy Servients
    3. Application Servients
    4. Directories
    5. Integrations
9. Security and Privacy Considerations









Introduction

Note: 文書の概要を示すものであり、ドラフトのアブストラクトからテキストをコピー。

- The W3C Web of Things (WoT) is intended to enable interoperability across IoT Platforms and application domains. Primarily, it provides mechanisms to formally describe IoT interfaces to allow IoT devices and services to communicate with each other, independent of their underlying implementation, and across multiple networking protocols. Secondarily, it provides a standardized way to define and program IoT behavior.
- This document describes the abstract architecture for the W3C Web of Things. It is derived from a set of use cases and can be mapped onto a variety of concrete deployment scenarios, several examples of which are given. This document is focused on the standardization scope of W3C WoT, which consists of three initial building blocks that are briefly introduced and their interplay explained.
- The WoT Thing Description (TD) provides a formal mechanism to describe the network interface provided by IoT devices and services, independent of their implementation. Provision of a TD is the primary requirement for a device to participate in the Web of Things. In fact, defining a Thing Description for an existing device allows that device to participate in the Web of Things without having to make any modifications to the device itself. WoT Binding Templates define how a WoT device communicates using a concrete protocol. The WoT Scripting API—whose use is not mandatory—provides a convenient mechanism to discover, consume, and expose Things based on the WoT Thing Description.
- Other non-normative architectural blocks and conditions underlying the Web of Things are also described in the context of deployment scenarios. In particular, recommendations for security and privacy are included, while the goal is to preserve and support existing device mechanisms and properties. In general, W3C WoT is designed to describe what exists rather than to prescribe what to implement.

Terminology

- Action
- Application
- Binding templates
- Client API
- To consume a thing
- Discovery API
- Domain-specific vocabulary
- Event
- Execution environment
- To expose a thing
- Interaction
- Interaction model
- Interaction pattern
- IoT platform
- Local discovery
- Manual discovery
- Nearby discovery
- Network discovery
- Property
- Protocol binding
- Remote discovery
- Scripting API
- Server API
- Servient
- TD
- TD vocabulary
- Thing
- Thing Description (TD)
- Thing directory
- WoT Client
- WoT Interface
- WoT Object
- WoT Runtime
- WoT Server
- CoAP
- CWT
- JSON-LD
- JWT
- RDF

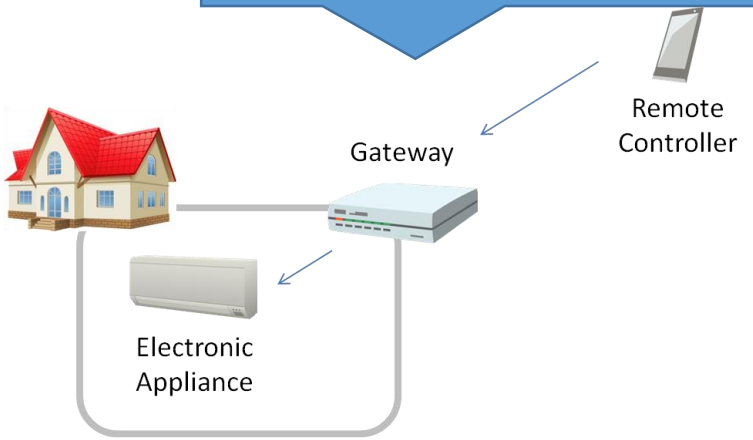
Use case

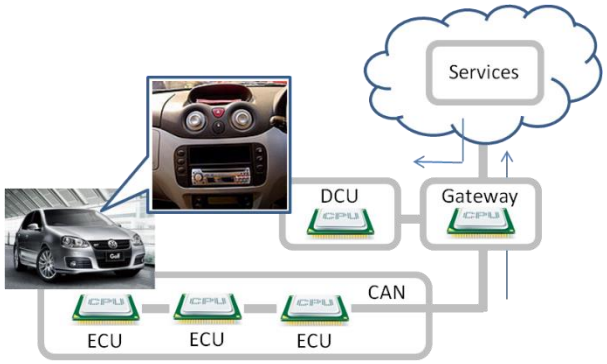
- This section presents the use cases targeted by the W3C WoT and which are used to derive the abstract architecture discussed in 5. WoT Building Blocks. While Smart Home use cases might appear predominant in this section, they should be seen as simply a vehicle to identify fundamental requirements inherent to most application domains. The Smart Home domain is suitable for identifying such general requirements, as most stakeholders can relate to it.

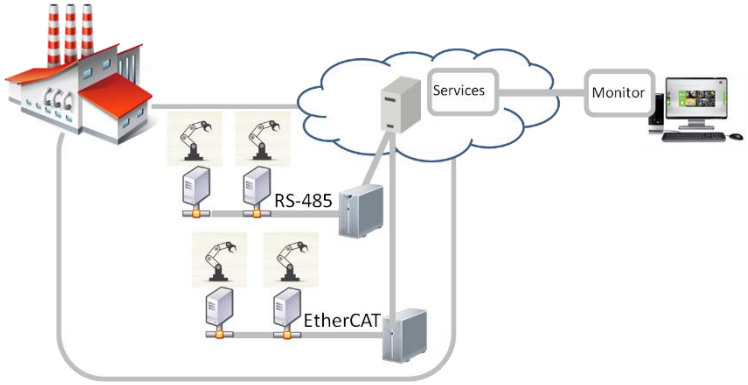
Use cases for WoT

Note: 様々な適用が考えられることで、バリエーションを説明する。実際のプロトコル等も記載

- Smart home
- Smart factory
- Connected car
- …



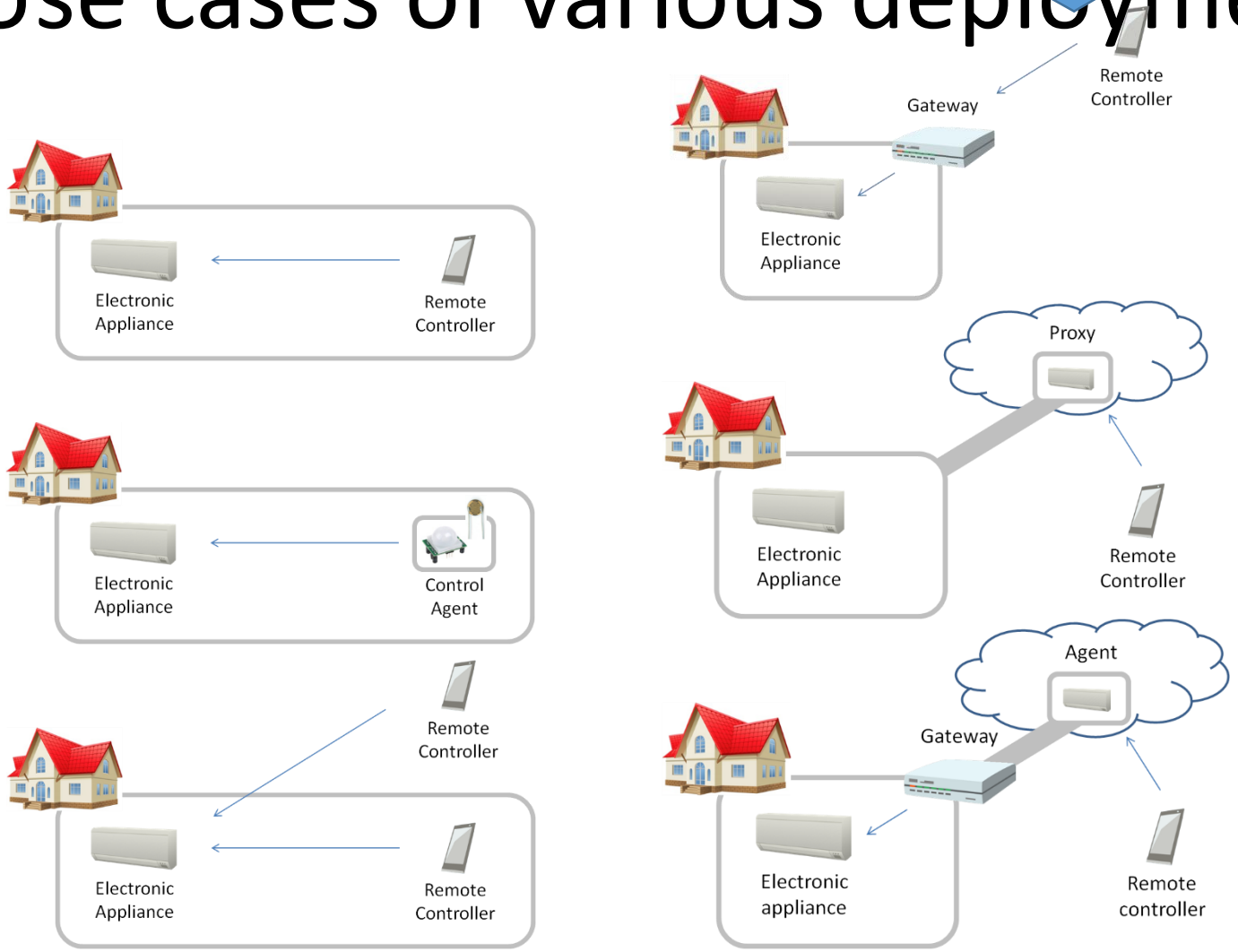




A variety of devices that support diverse communication protocols have been connected in many fields. Common way to manage and handle devices are required for the application developers.

Note: アーキテクチャ構成のバリエーションを記載する
ID管理、デバイスの発見等？

Use cases of various deployment



Various way to connect applications and devices are required for the real situation. Direct connections, gateways supporting their connection, connecting via cloud, …

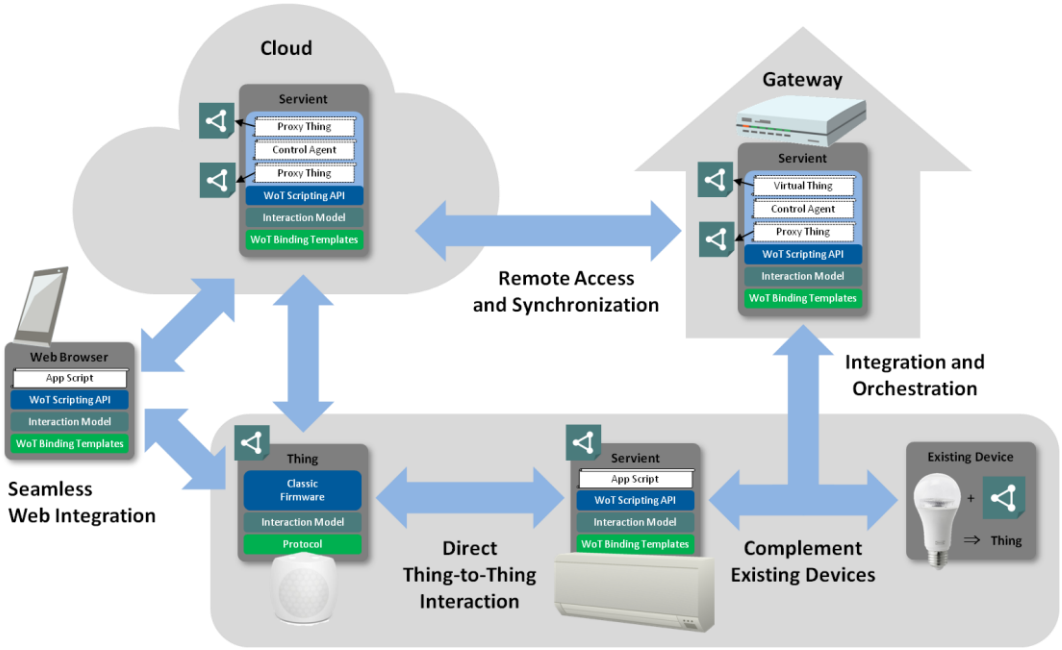
Use cases

- ユースケース
  どのようなプロトコルでつながっているか
  →プロトコルバインディングの必要性
  →ID管理、デバイス管理

- デプロイメント
  デバイス、アプリケーション、プロキシ
  抽象アーキテクチャ→構成要素を明確にする

Summary of use cases

- As described above, various ways to connect with applications, devices, cloud and gateway are aggregated into the same architecture regardless of the fields.

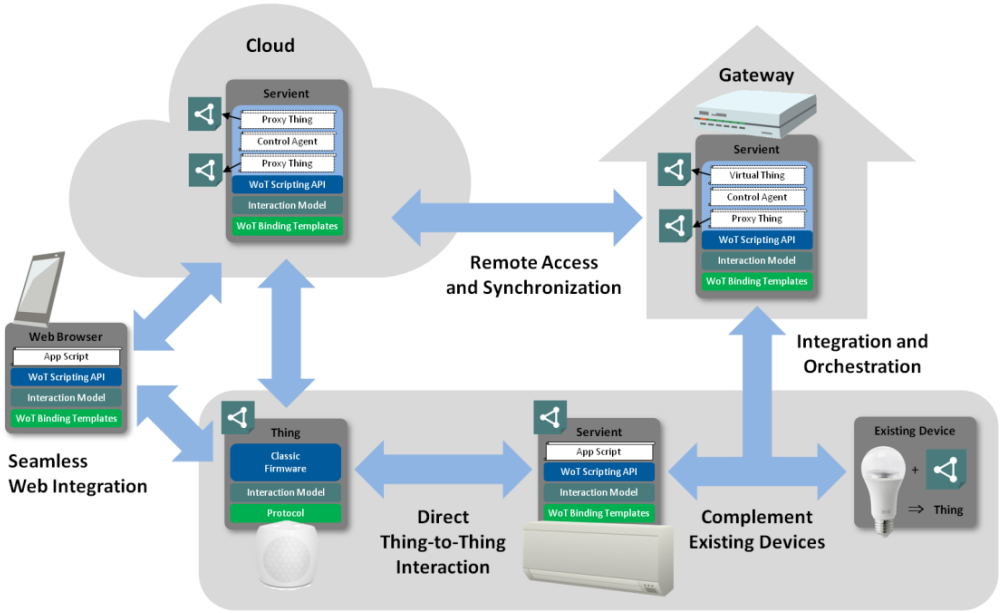


Functional requirements

- 分野を問わずにアプリケーションから同じインタフェースでデバイスにアクセスできる。デバイスが同じ情報モデルを共有する必要がある（Thing Description）
- 分野固有の規格を共通のインタフェースに変換する必要がある（Protocol binding）
- アプリケーションとデバイスとの様々な接続形態に対応するために、アプリ、デバイス、ゲートウェイ（プロキシ）の基本要素を定義し、共通のアーキテクチャを定義する必要がある（Servient）
- 複数のネットワークを超えて、アプリとデバイスが接続される（Proxies architecture）
- アプリケーションからデバイスを発見する方法が必要。そのために、デバイスの情報モデルを登録し、それをネットワークを経由して検索することができる。（Discovery）

Abstract architecture

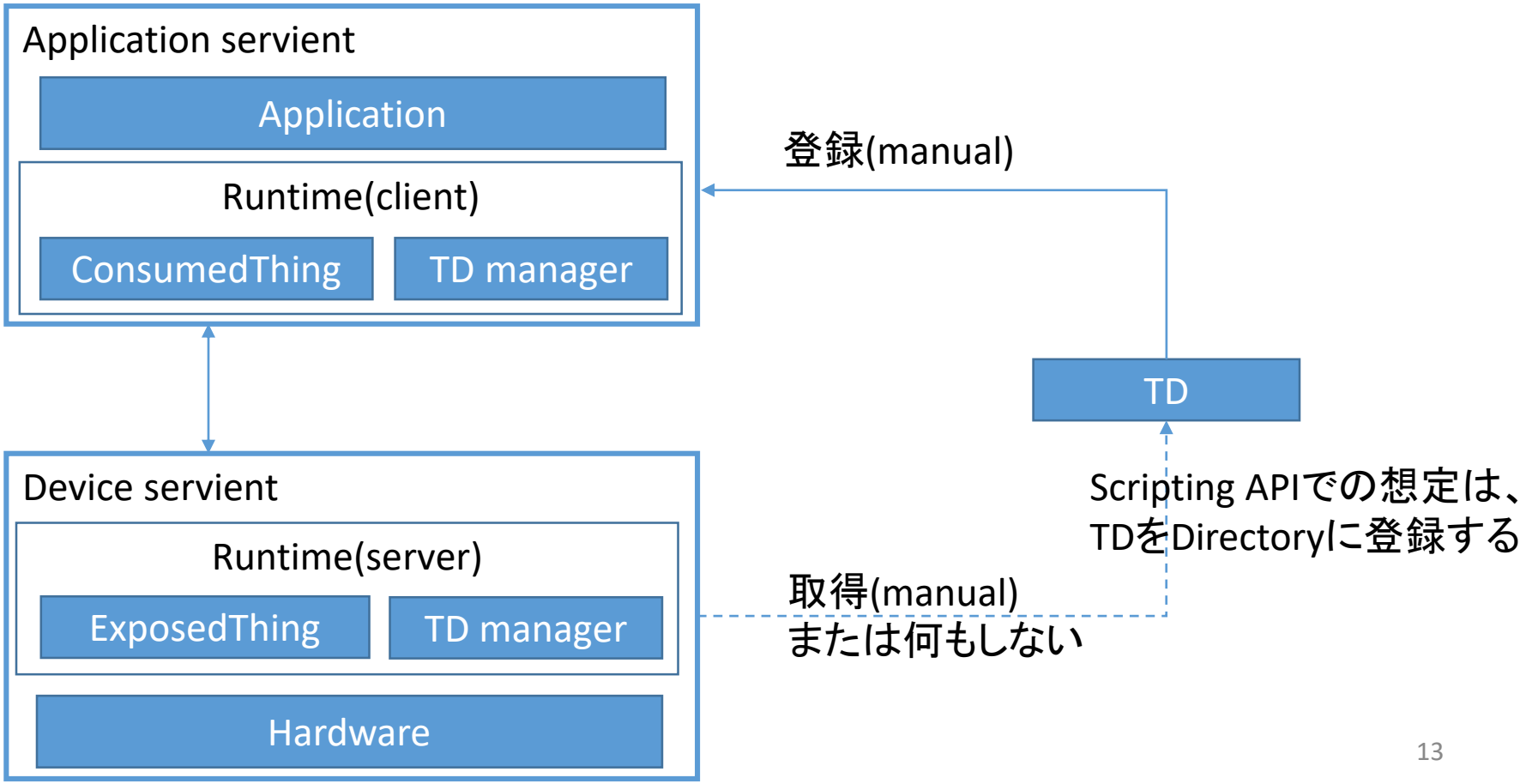
- This section presents the initial WoT building blocks that build up the abstract architecture for the Web of Things. This architecture is derived from the use cases in Section 3. Use Cases and the requirements in Section 4. Functional Requirements. Figure 9 summarizes the high-level goals and requirements and shows the three levels where the WoT building blocks can be applied:
  - the device level,
  - the gateway level (or "edge"), and
  - the cloud level.



Note: 基本的な考え方として、
機能ブロックのみを記載します

Simple configurations of WoT

- Simple configuration 1: Application and Device



Servients

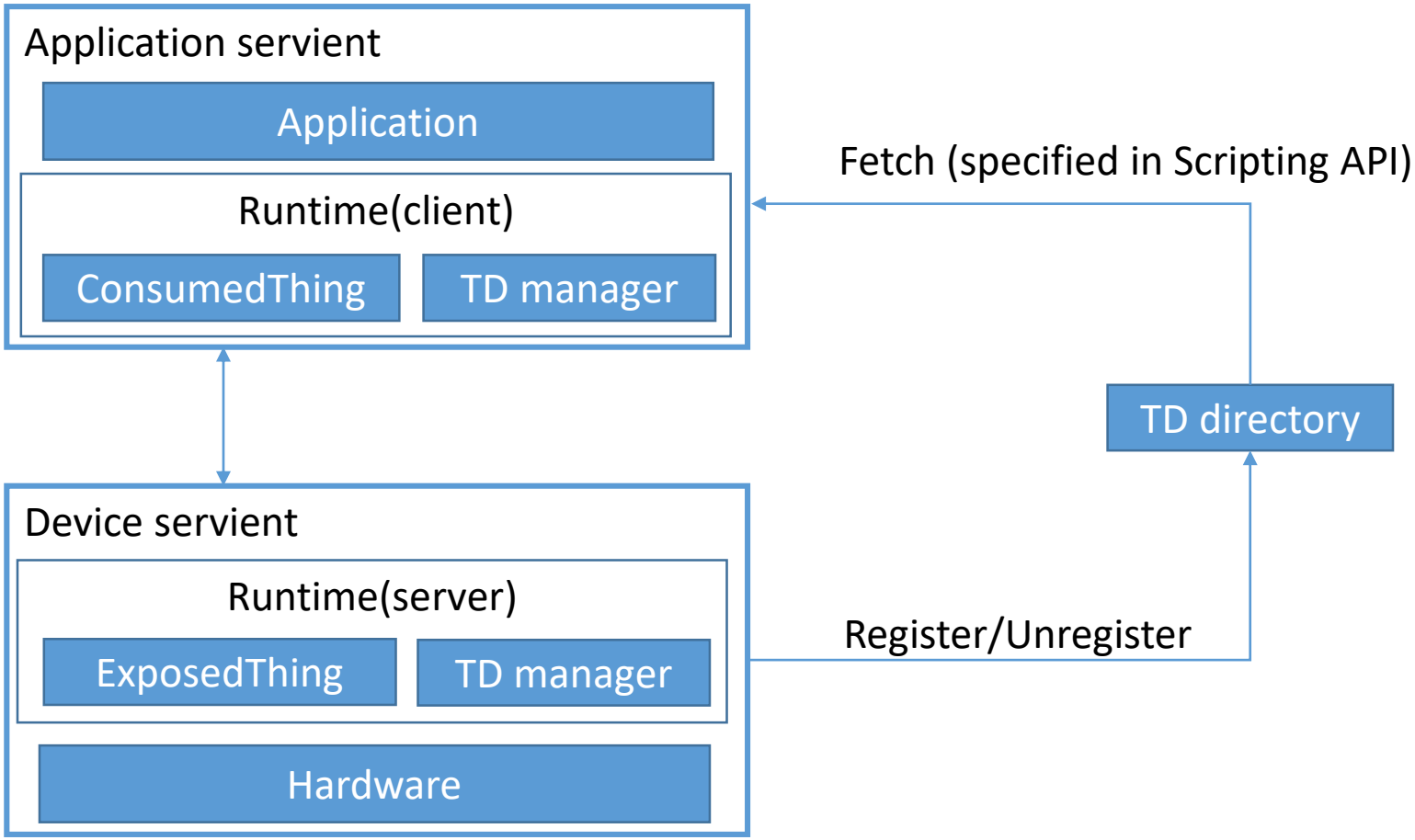
- Device servient (former 5.1 Thing)
  - A Thing is the abstraction of a physical or virtual entity that needs to be represented in IoT applications. This entity can be a device, a logical component of a device, a local hardware component, or even a logical entity such as a location (e.g., room or building).
  - Things provide a network-facing API for interaction (WoT Interface) based on a formal model. These WoT Interfaces are a superset of Web APIs, as Things can also be available over non-Web protocols such as MQTT or ZigBee. The outward-facing WoT Interface is not to be confused with the Scripting API, which is optional and interfaces with application scripts inside the software stack of a Thing.
  - There can be Things, however, that do not provide a WoT Interface and only consist of metadata that is relevant to the application (e.g., the room in which devices are located). In W3C WoT however, a Thing must have a Thing Description; therefore, everything that has a Thing Description is a Thing.
- Application servient

Thing description (former 5.2)

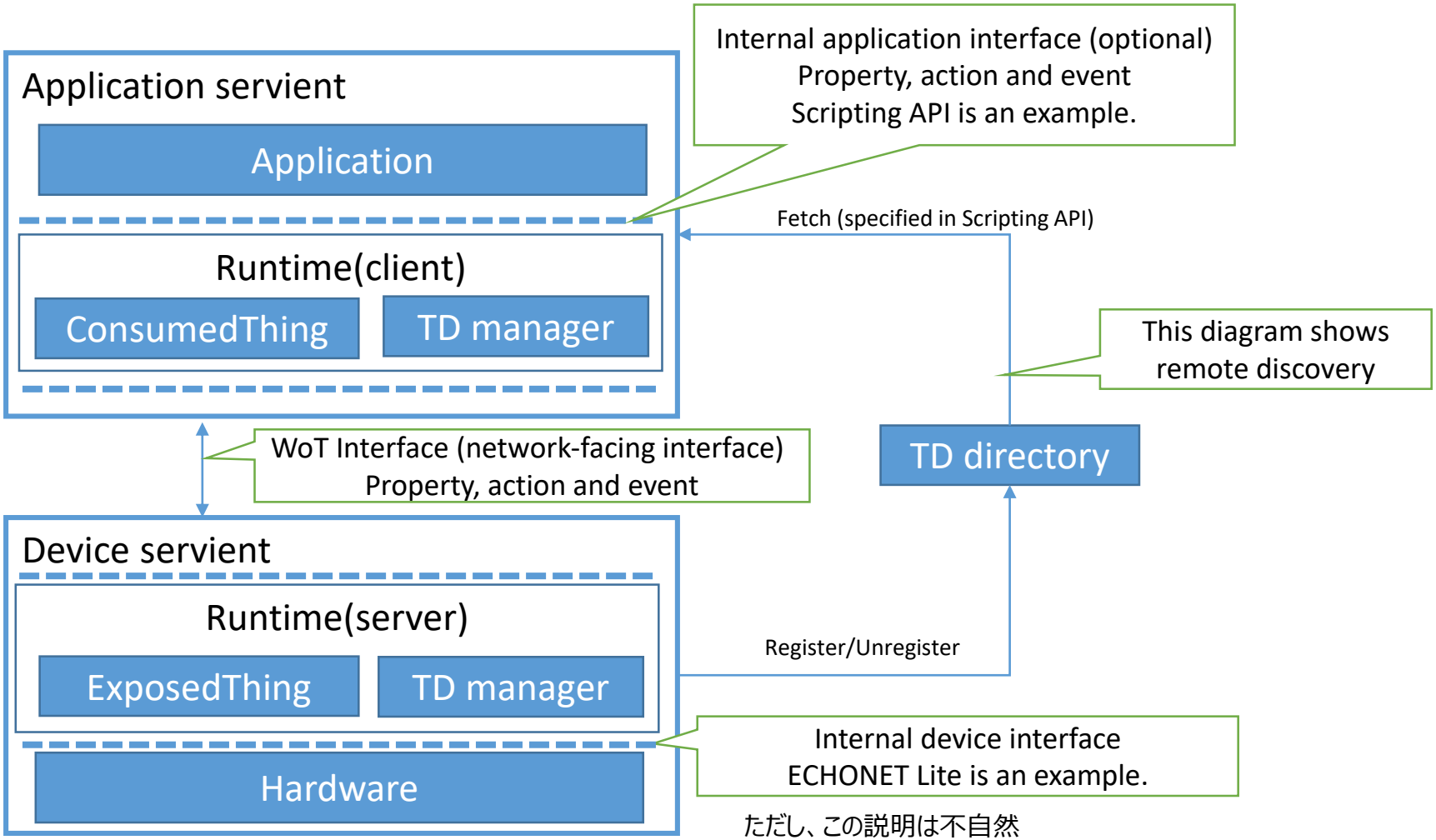
- The WoT Thing Description (TD) is structured data that adheres to a formal model and closes the gap between Linked Data vocabularies and functional APIs of IoT systems. It can be seen as the "HTML for Things". A TD provides general metadata of a Thing as well as metadata about the Interactions, data model, communication, and security mechanisms of a Thing. Usually, TDs make use of domain-specific metadata for which WoT provides explicit extension points. However, any domain-specific vocabulary is out-of-scope of the W3C standardization activity.
- The WoT Thing Description is built around a formal Interaction Model that can support multiple messaging paradigms (i.e, request-response, publish-subscribe, and message passing). The default Interaction Patterns are Property, Action, and Event. These were found to be able to cover the network-facing APIs provided by most IoT Platforms. Properties abstract data points that can be read and often written. Actions abstract invokable processes that may run for a certain time; yet they can also abstract RPC-like interactions in general. Events abstract interactions where the remote endpoint pushes data asynchronously.
- Thing Descriptions are serialized to JSON-LD [JSON-LD] by default. More serialization formats are planned in the future, in particular more concise formats for resource-constrained Things. For now, JSON-LD offers a good trade-off between machine-understandable semantics and usability for developers.
- Thing Descriptions can be managed in Thing Directories, which are aligned with the CoRE Resource Directory [CoRE-RD]. They provide a Web interface for registration, registration updates, and removal, and automatic removal after a given lifetime expired without registration update. Thing Directories also provide a Web interface for lookups, usually including a SPARQL endpoint for semantic queries in addition to simple CoRE Resource Directory [CoRE-RD] lookups.
- The WoT Thing Description fosters interoperability in two ways: First, and foremost, TDs enable machine-to-machine communication in the Web of Things. Second, TDs can serve as a common, uniform format for developers to document and retrieve all details necessary to access IoT devices and make use of their data.

Simple configurations of WoT

- Simple configuration 1: Application and Device



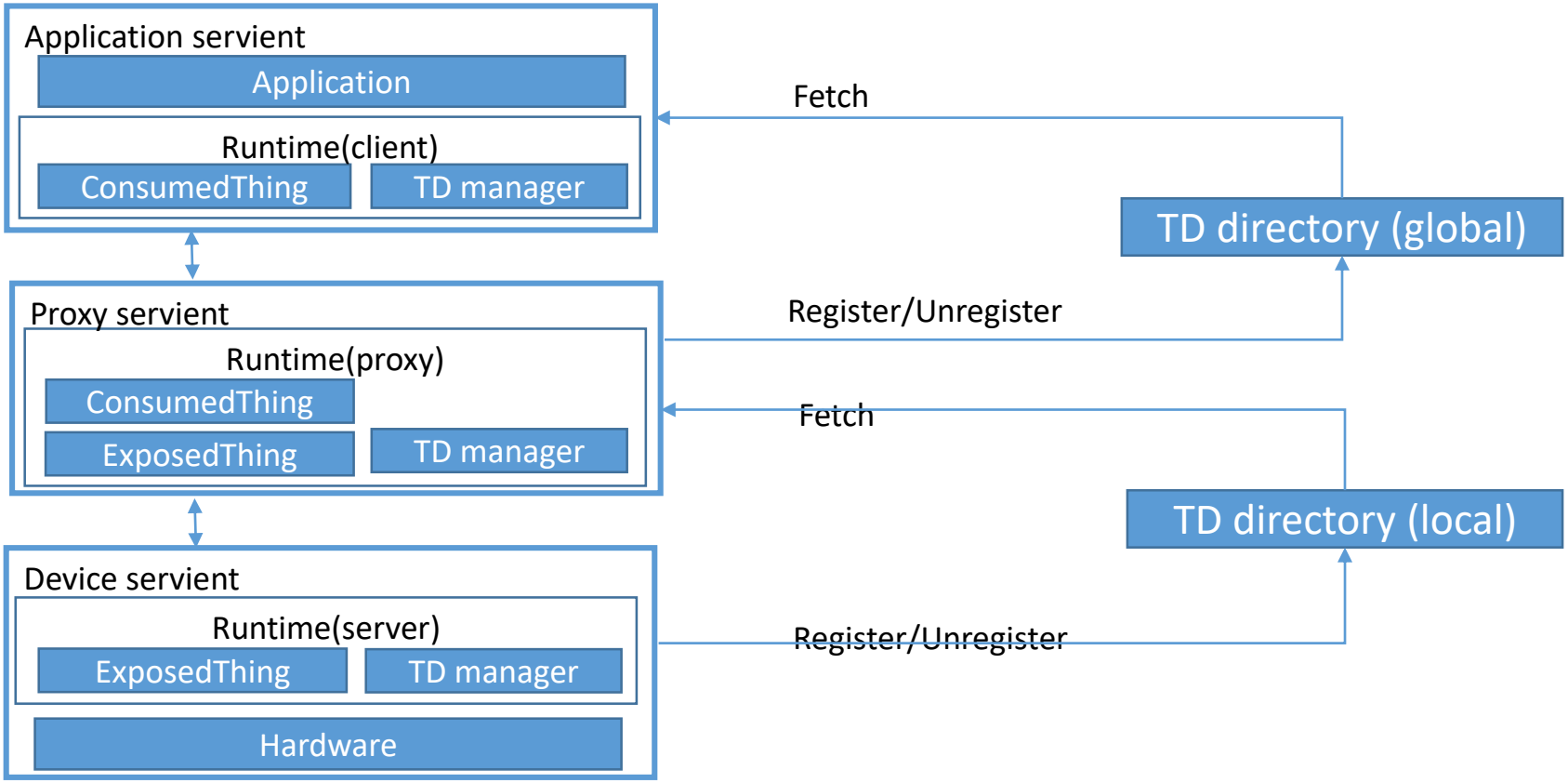
Simple configurations of WoT



ただし、この説明は不自然
Runtimeとハードウェアの間には、何らかのアダプタが必要

Simple configurations of WoT

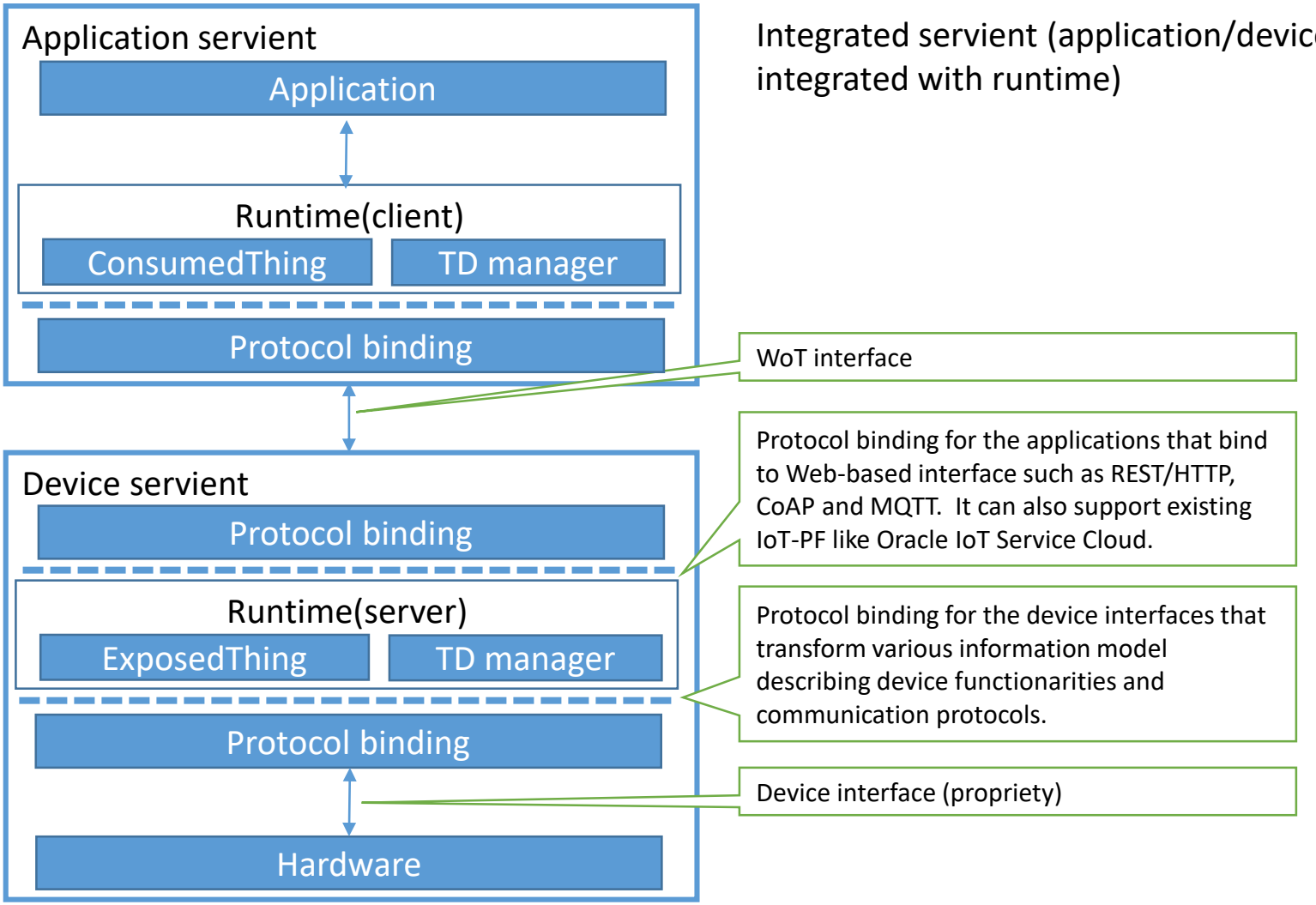
- Simple configuration 2: Proxy



Servients

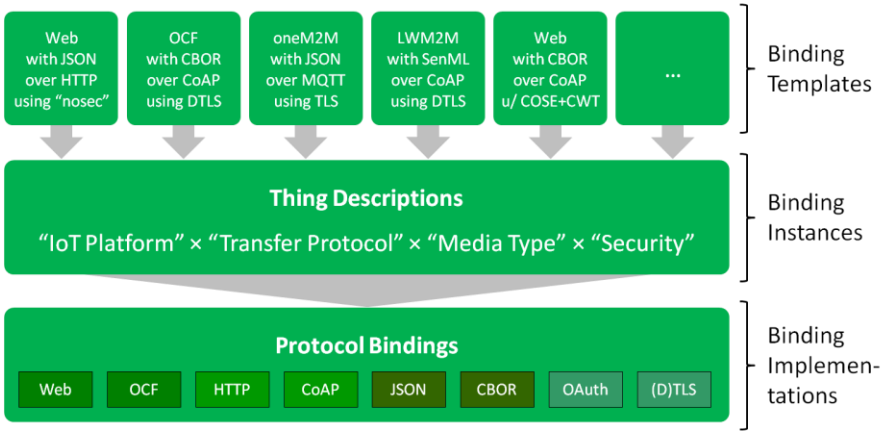
- Proxy servient
- 説明を追加予定

WoT servient architecture



Protocol binding

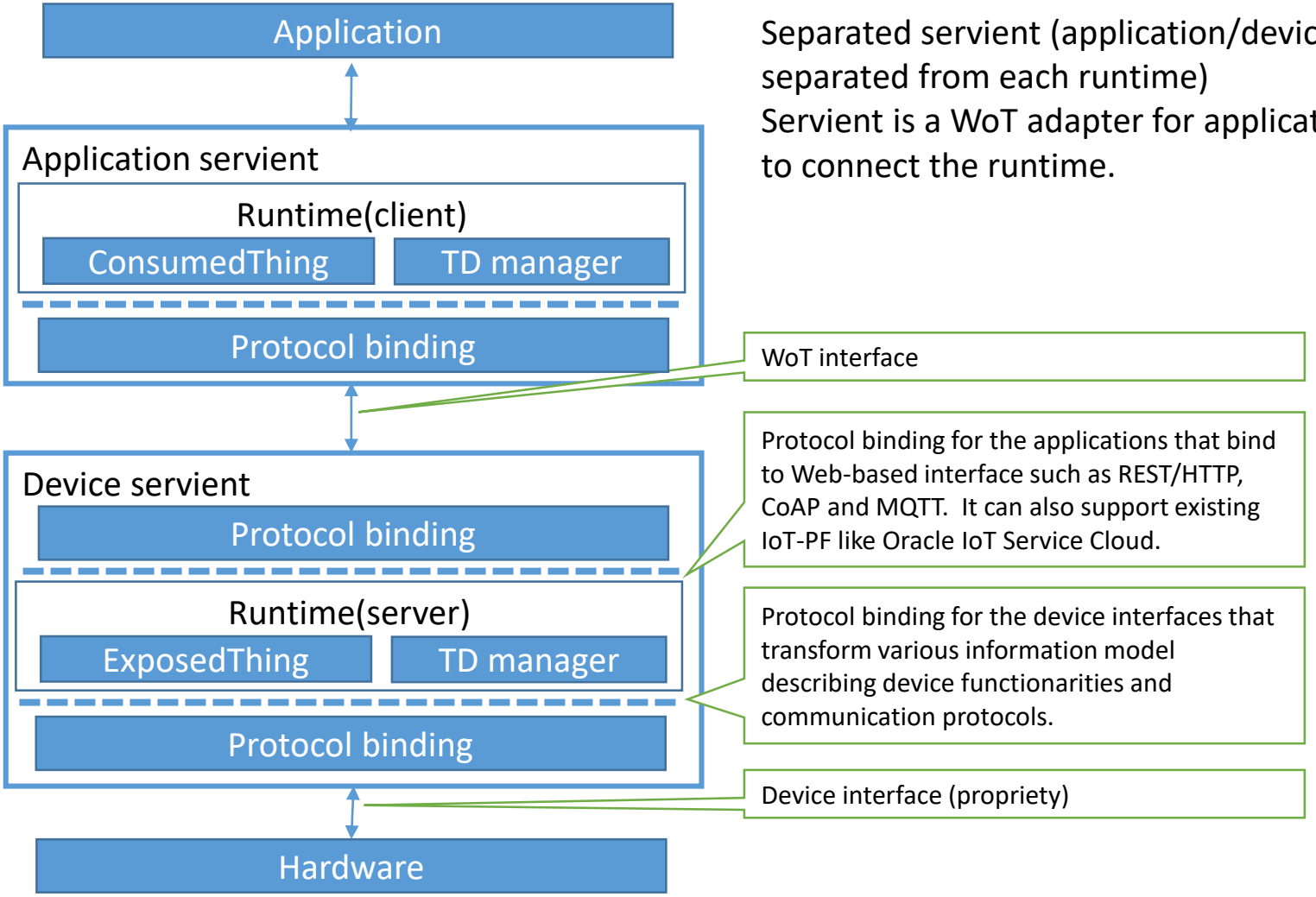
- A great challenge for the WoT is to enable interactions with a myriad of different IoT Platforms (e.g., OCF, oneM2M, RESTful devices not following any particular standard but providing an HTTP or CoAP interface, etc.). The IoT uses a variety of protocols for accessing devices, since no one protocol is appropriate in all contexts. W3C WoT is tackling this variety by including communication metadata in the Thing Description. This metadata can be used to configure the communication stack to produce compliant messages for a wide variety of target IoT Platforms and protocols.
- The WoT Binding Templates are an informal collection of communication metadata blueprints that explain how to interact with different IoT Platforms. When creating a Thing Description for a particular device, the Binding Template for the corresponding IoT Platform can be used and instantiated in the Thing Description for that device.



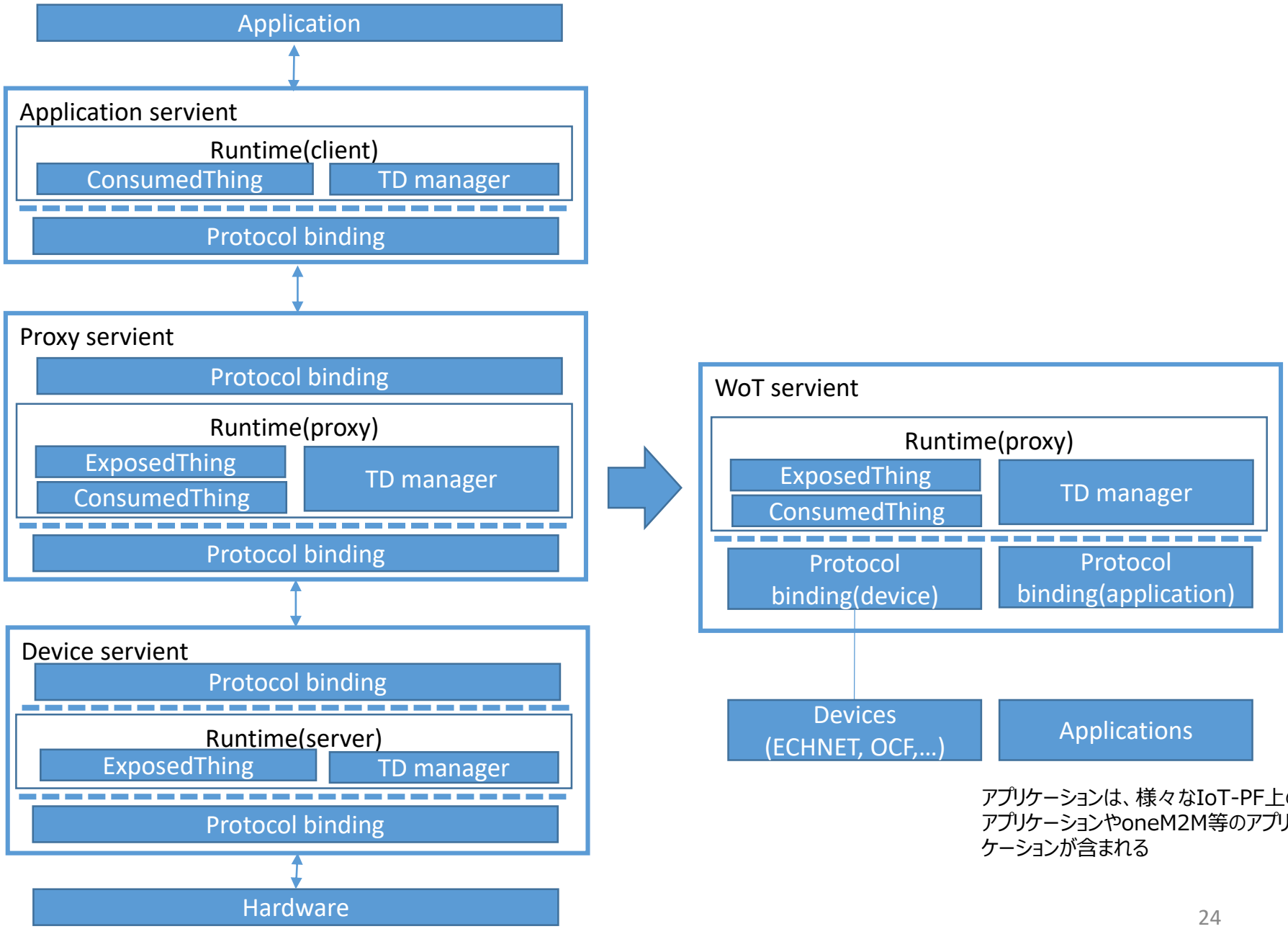
Binding template

- 1.1 Protocol Adaptation
- Protocol Binding Templates enable clients to adapt to the underlying protocol and network-facing API constructions. Once the base protocol (e.g., HTTP, CoAP, MQTT, etc.) is identified, the following adaptions specifiy the particular use within the given Platform.
- 1.1.1 Protocol Methods and Options
- Most protocols have a relatively small set of methods that define the message type, the semantic intention of the message. As a starting point, a superset of REST and PubSub can cover most standard communication patterns. Common methods are GET, PUT, POST, DELETE, PUBLISH, and SUBSCRIBE.
- The protocol methods are mapped to the abstract WoT Interaction verbs readproperty, writeproperty, observeproperty, invokeaction, subscribeevent, unsubscribeevent.
- Possible protocol options are also specified in the Protocol Binding. They are used to select transfer modes, to request notifications from observable resources, or otherwise extend the semantics of the protocol methods.
- 1.1.2 Media Types
- Maximum use should be made of IANA-registered Media Types (e.g., application/json) in order to decouple applications from connected Things. Standard bridges and translations from proprietary formats to Web-friendly languages such as JSON and XML are part of the adaptation needed.
- WoT Protocol Bindings depend on consistent use of Media Types for customization of the upper layers.
- 1.1.3 Payload Structure
- Data serialized to a standard Media Type still remains in a structure specific to the Platform data model and needs to be understood by clients (cf. various types of JSON documents).
- The data definition language of DataSchema elements, described in [TD], allows for describing arbitrary structures by nesting of arrays and objects. Constants and variable specifications may be intermixed.

WoT servient architecture

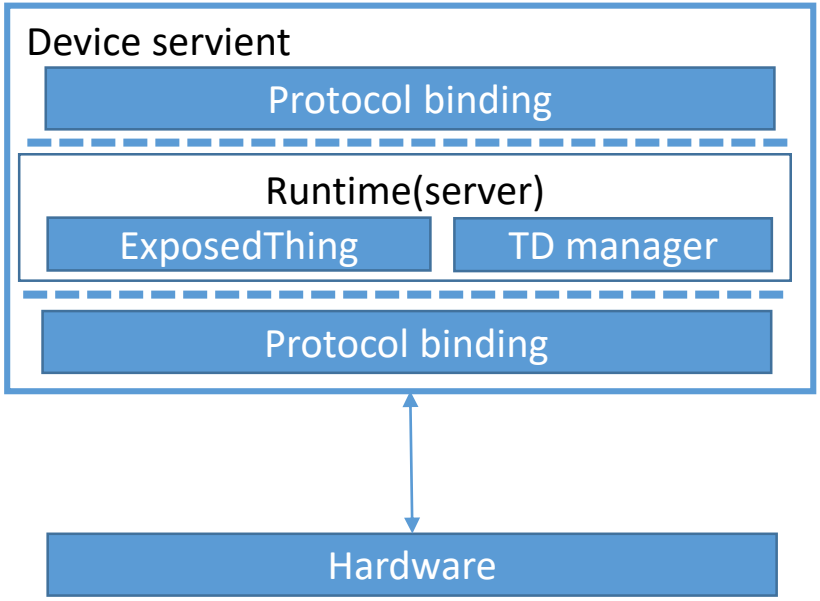


Separated servient (application/device separated from each runtime)
Servient is a WoT adapter for application/device to connect the runtime.

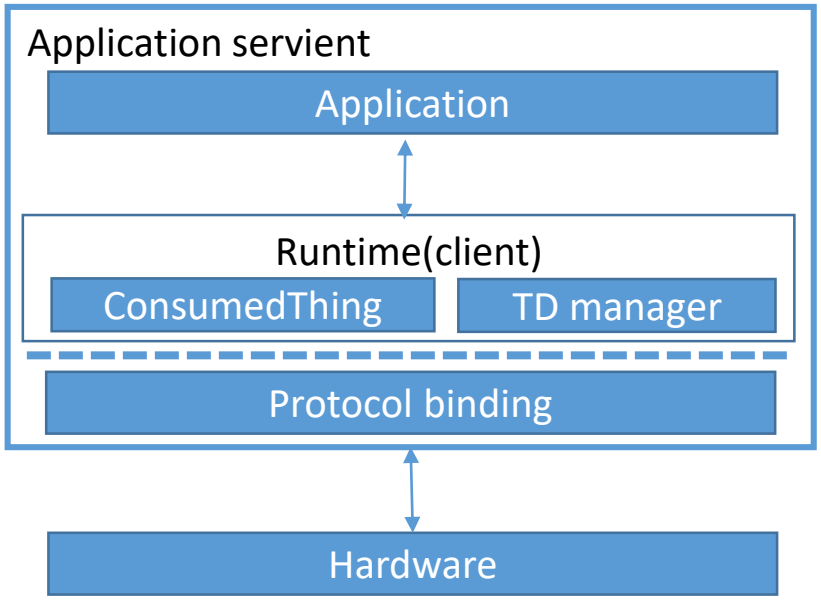


アプリケーションは、様々なIoT-PF上のアプリケーションやoneM2M等のアプリケーションが含まれる

Legacy device binding

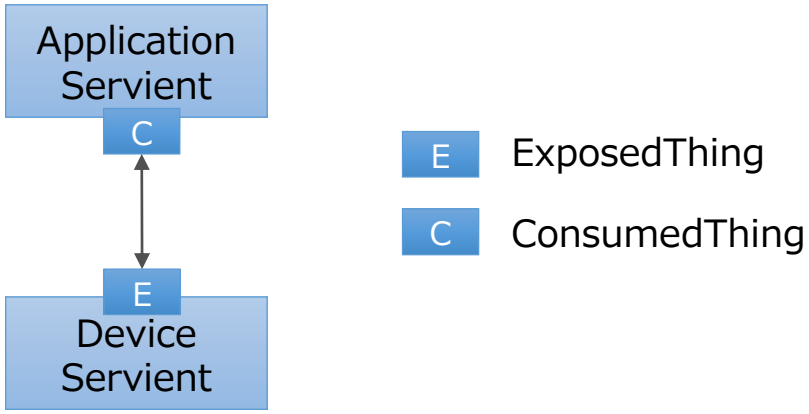


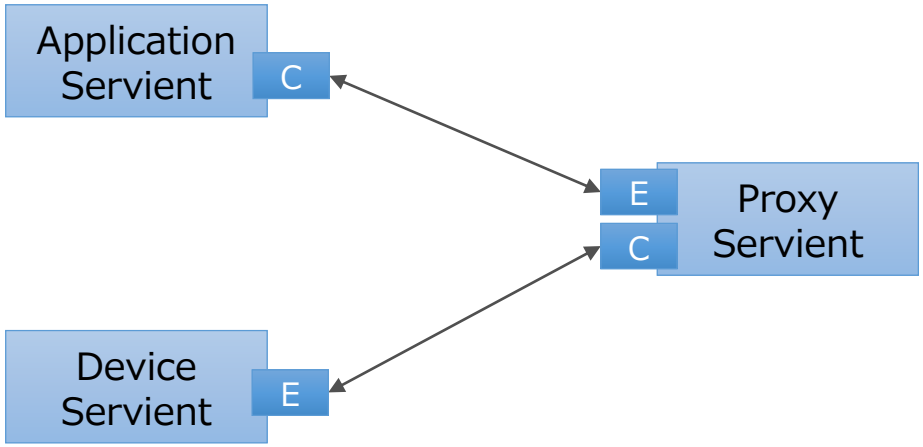
デバイスのbindingについては、多くの場合にはテンプレートをもとに変換することはできず、トランスポート、データ形式をそれぞれ変換する必要がある。

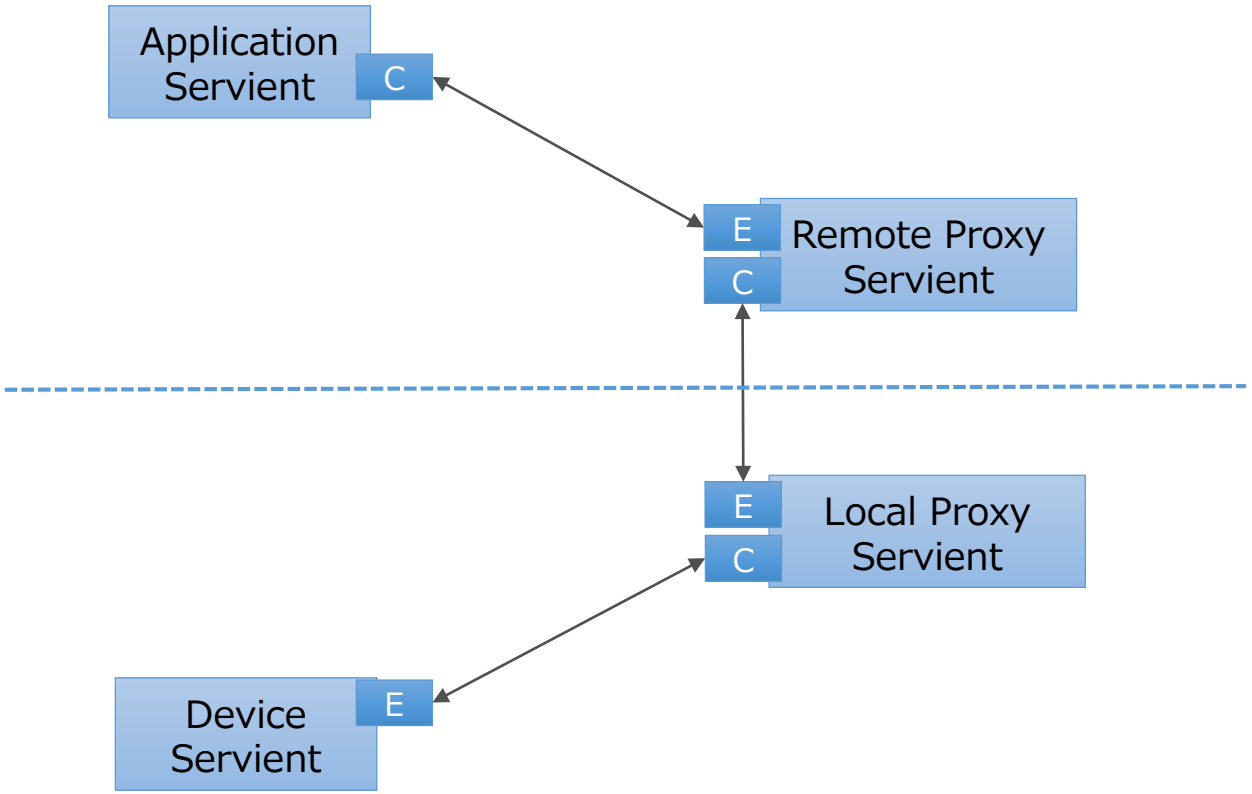


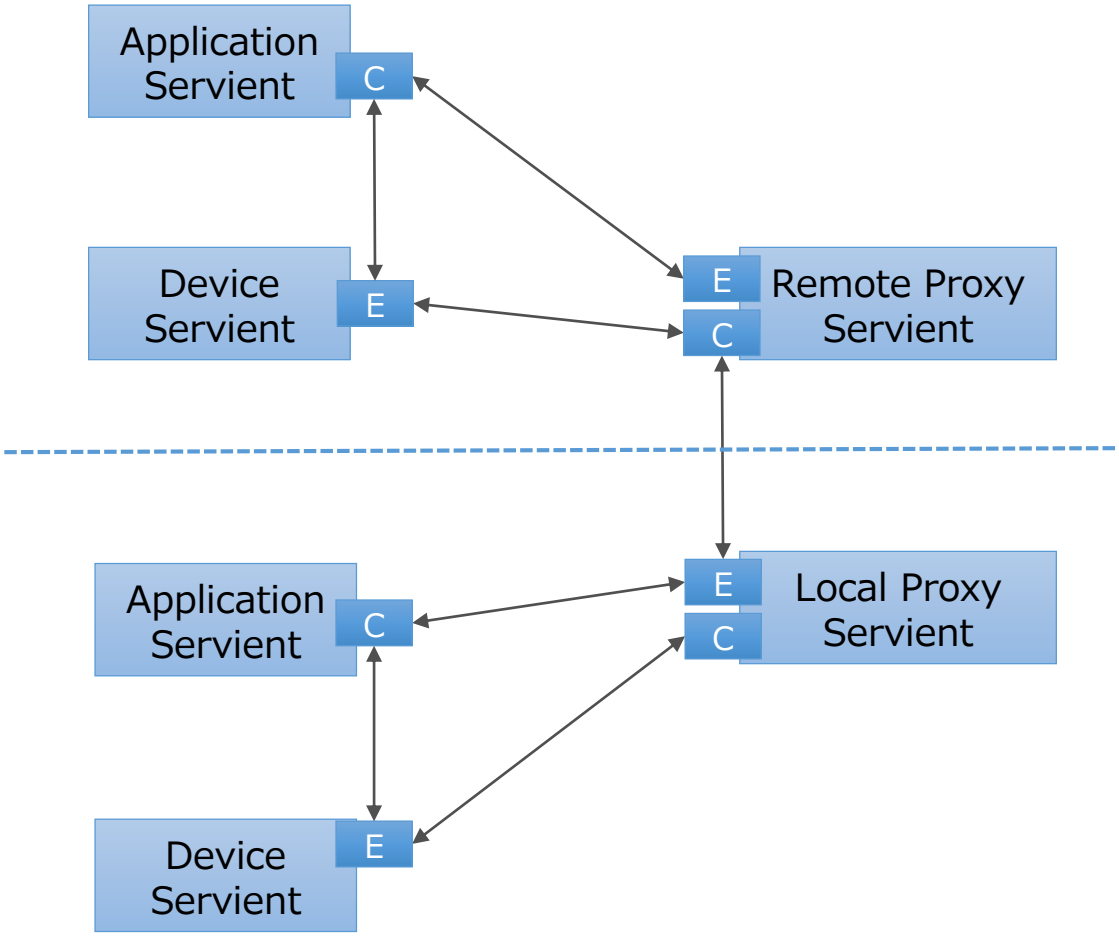
デバイスのインタフェースが、HTTP/CoAP/MQTTの場合には、変換なしにデバイスを接続することができる。

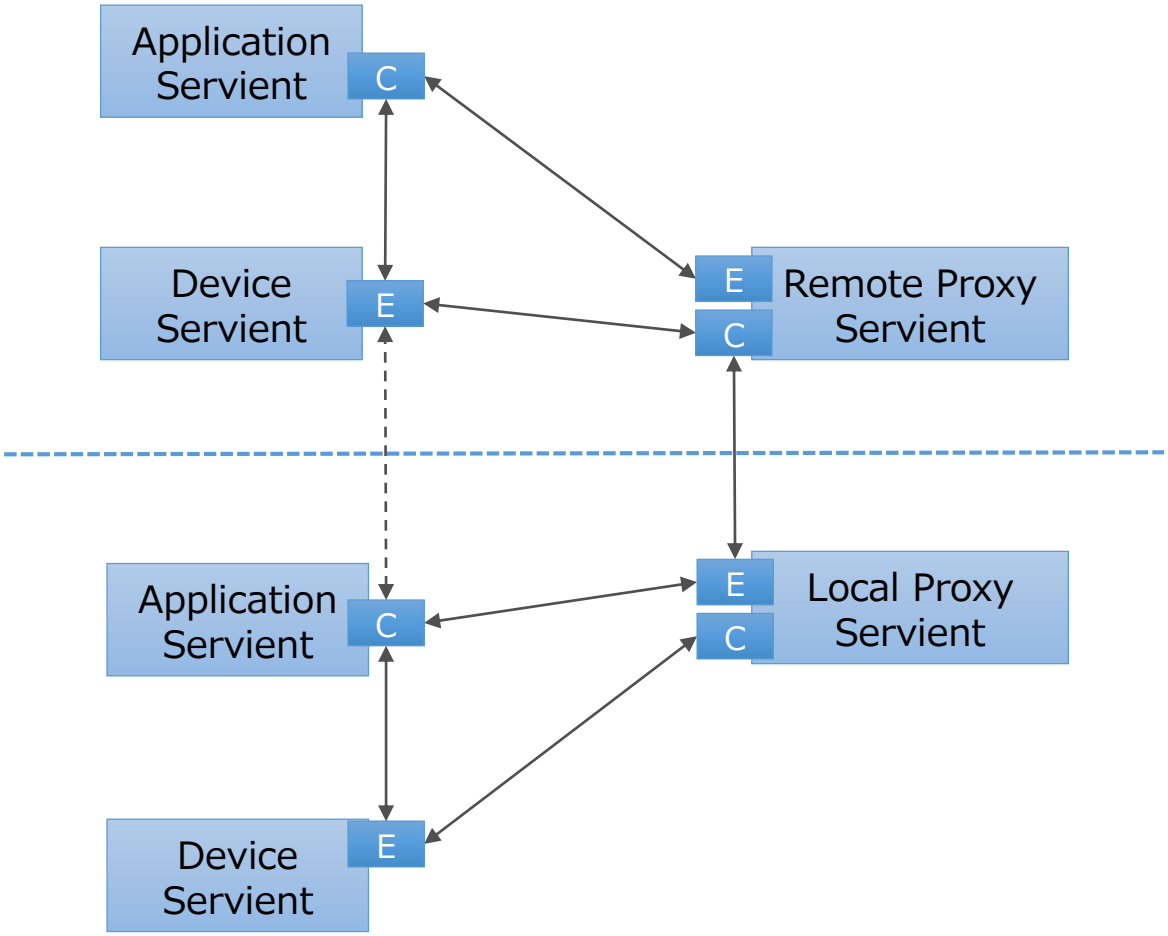
Deployment scenario

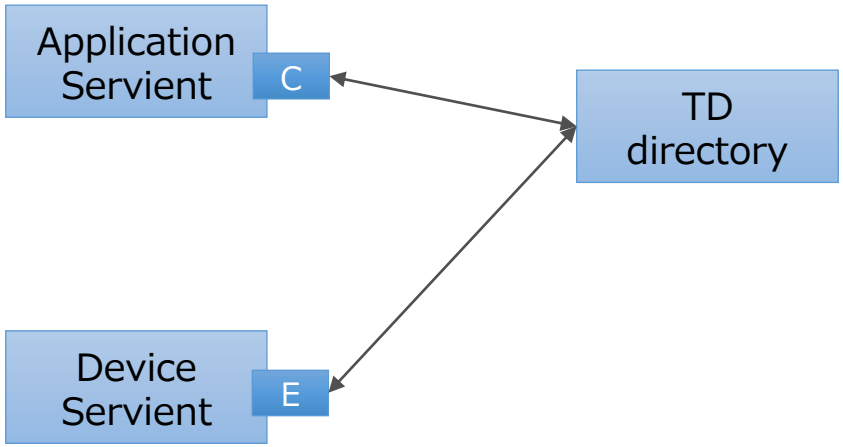


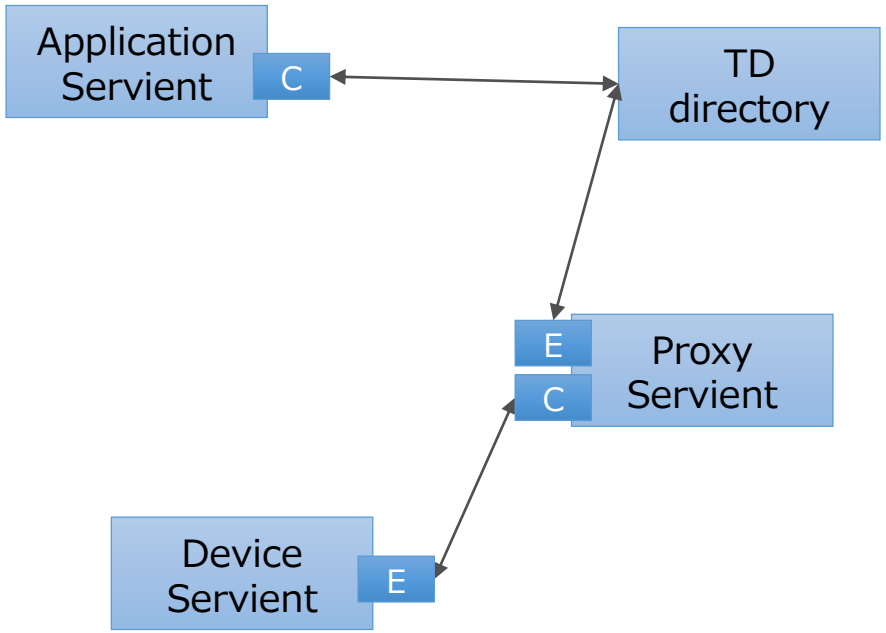

Application
Servient
C
E
C
Proxy
Servient
Device
Servient
E

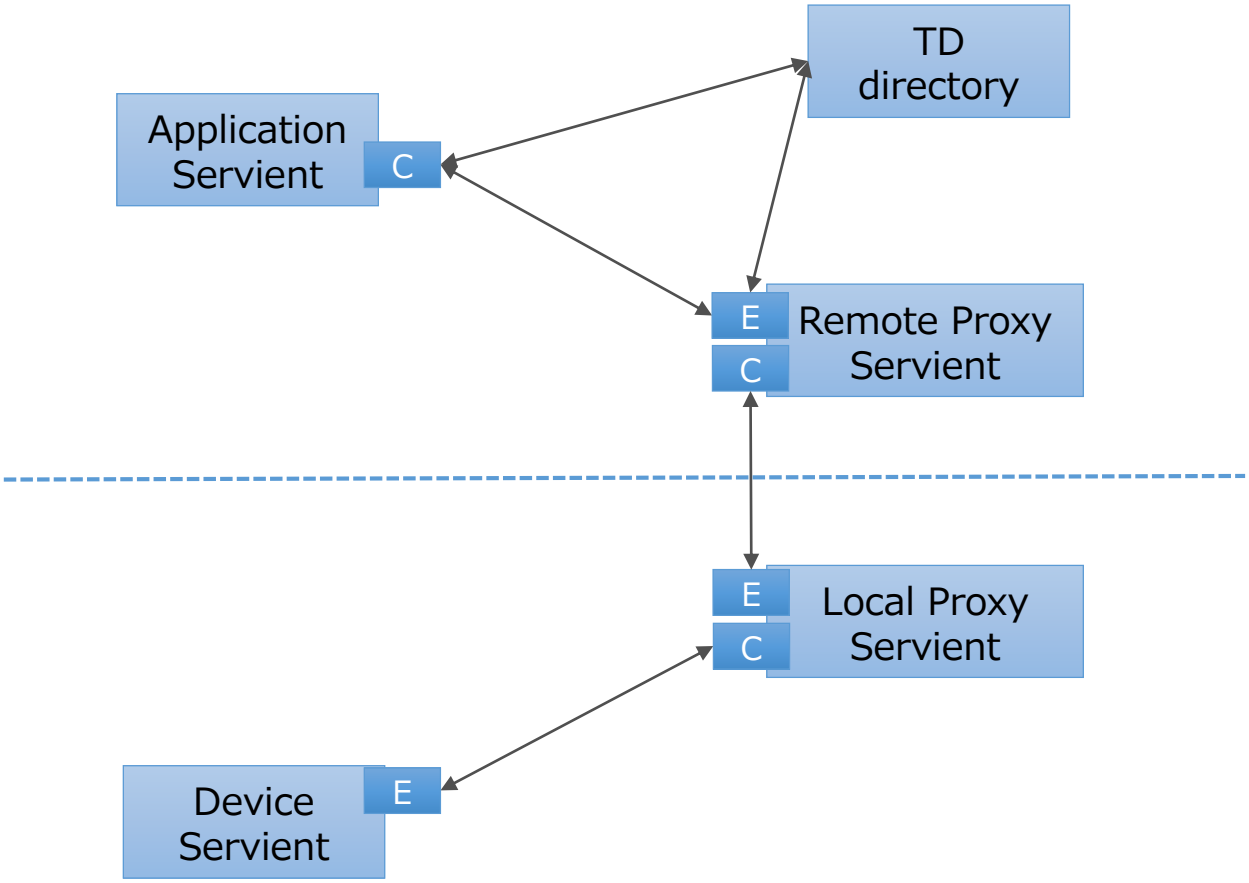

Application
Servient
C
E
C
Remote Proxy
Servient
E
C
Local Proxy
Servient
Device
Servient
E

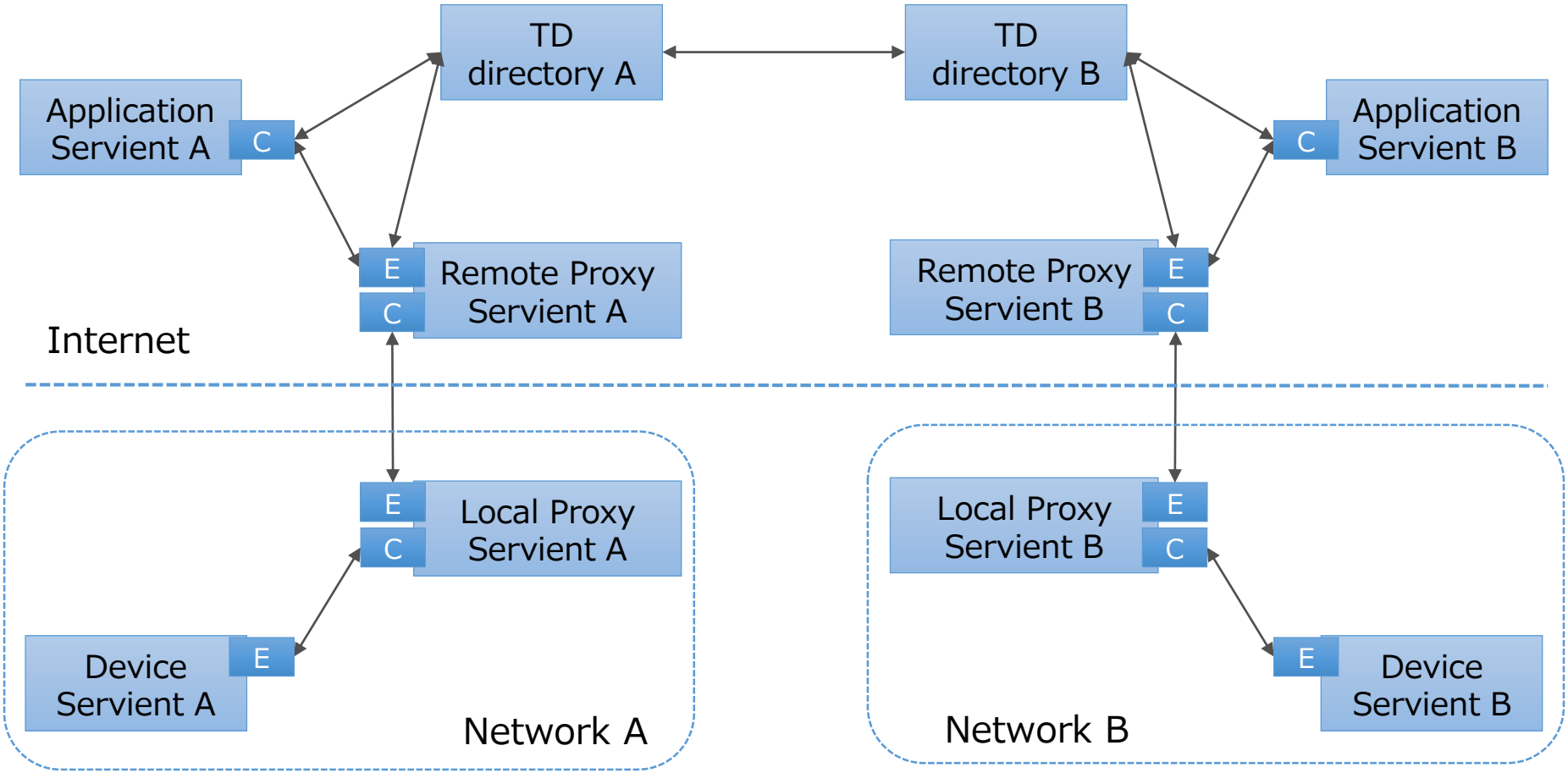

Application Servient
C
Device Servient
E
E
C
Remote Proxy Servient
Application Servient
C
Device Servient
E
E
C
Local Proxy Servient

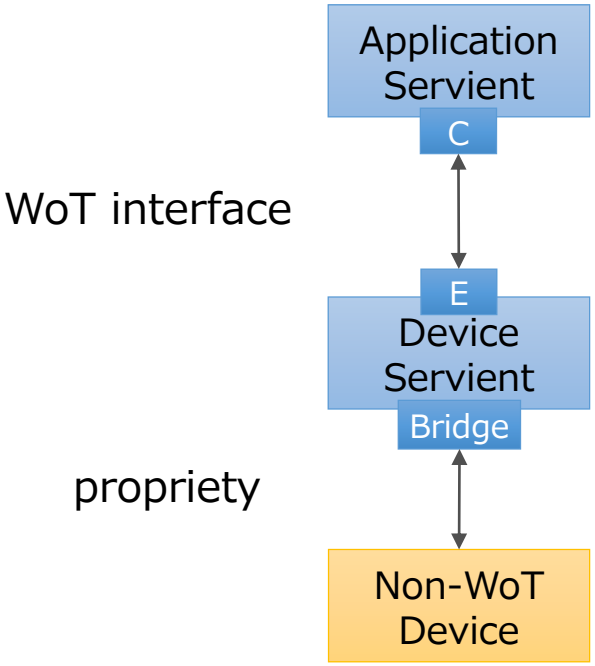

Application Servient
C
Device Servient
E
E
C
Remote Proxy Servient
Application Servient
C
E
C
Local Proxy Servient
Device Servient
E

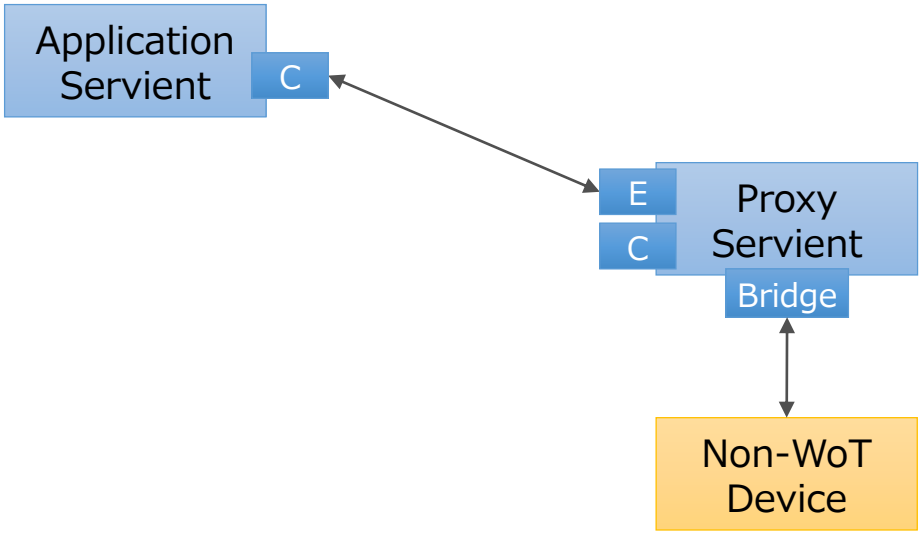

Application
Servient
C
TD
directory
Device
Servient
E

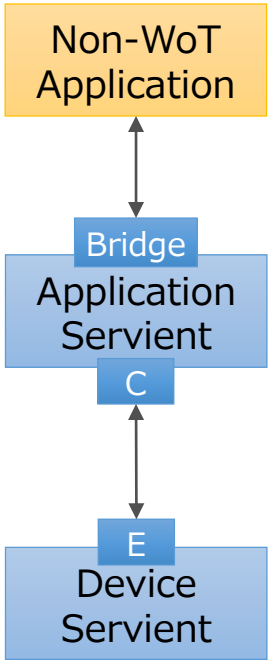

Application Servient
C
TD directory
E
C
Proxy Servient
Device Servient
E

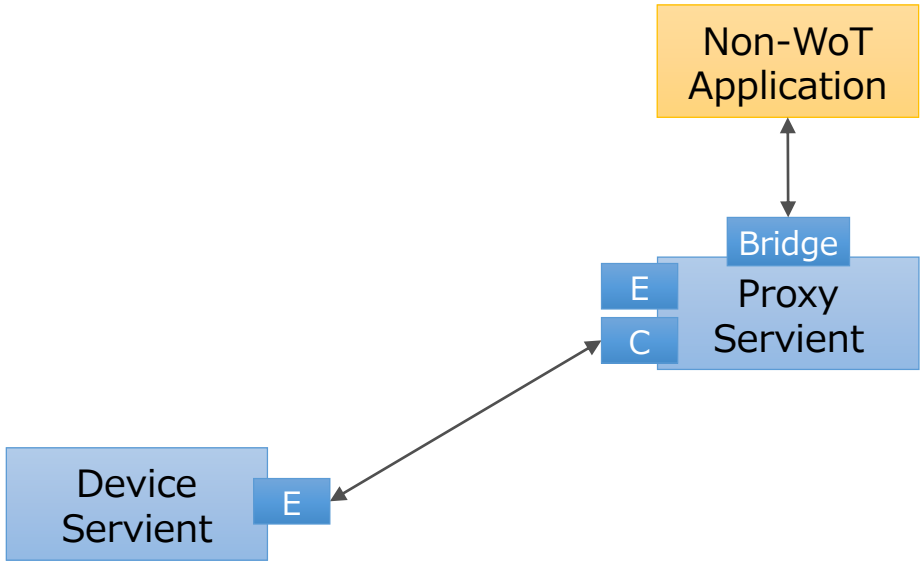

TD
directory
Application
Servient
C
E
C
Remote Proxy
Servient
E
C
Local Proxy
Servient
Device
Servient
E


TD
directory A
TD
directory B
Application
Servient A
C
C
Application
Servient B
E
C
Remote Proxy
Servient A
Remote Proxy
Servient B
E
C
Internet
E
C
Local Proxy
Servient A
Local Proxy
Servient B
E
C
Device
Servient A
E
E
Device
Servient B
Network A
Network B


Application
Servient
C
WoT interface
E
Device
Servient
Bridge
propriety
Non-WoT
Device


Application
Servient
C
E
C
Proxy
Servient
Bridge
Non-WoT
Device


Non-WoT
Application
Bridge
Application
Servient
C
E
Device
Servient


Non-WoT
Application
Bridge
E
C
Proxy
Servient
Device
Servient
E