智慧整合感控系統概論
Introduction to Cyber-Physical Systems

物聯網軟體架構
(M2M Software Architecture)

國立臺北科技大學電子工程系
授課教師：李昭賢 副教授
電子郵件：chlee@ntut.edu.tw
校內分機：2288

行動寬頻尖端技術跨校教學聯盟

Taipei Tech Electronic Engineering

文大行動智慧聯網跨校聯盟

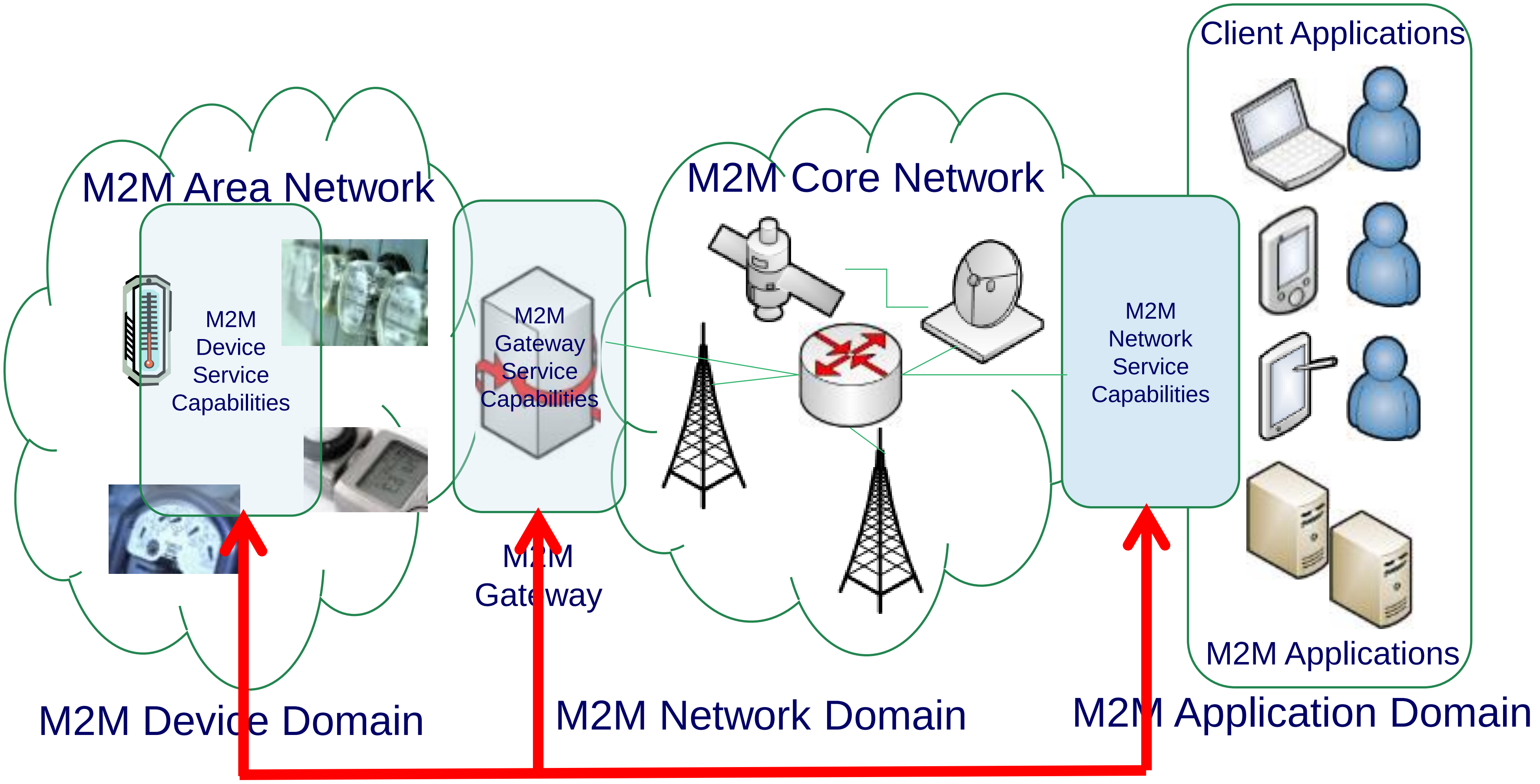
http://www.cc.ntut.edu.tw/~chlee/

學習目標

1 M2M Common Service Layer

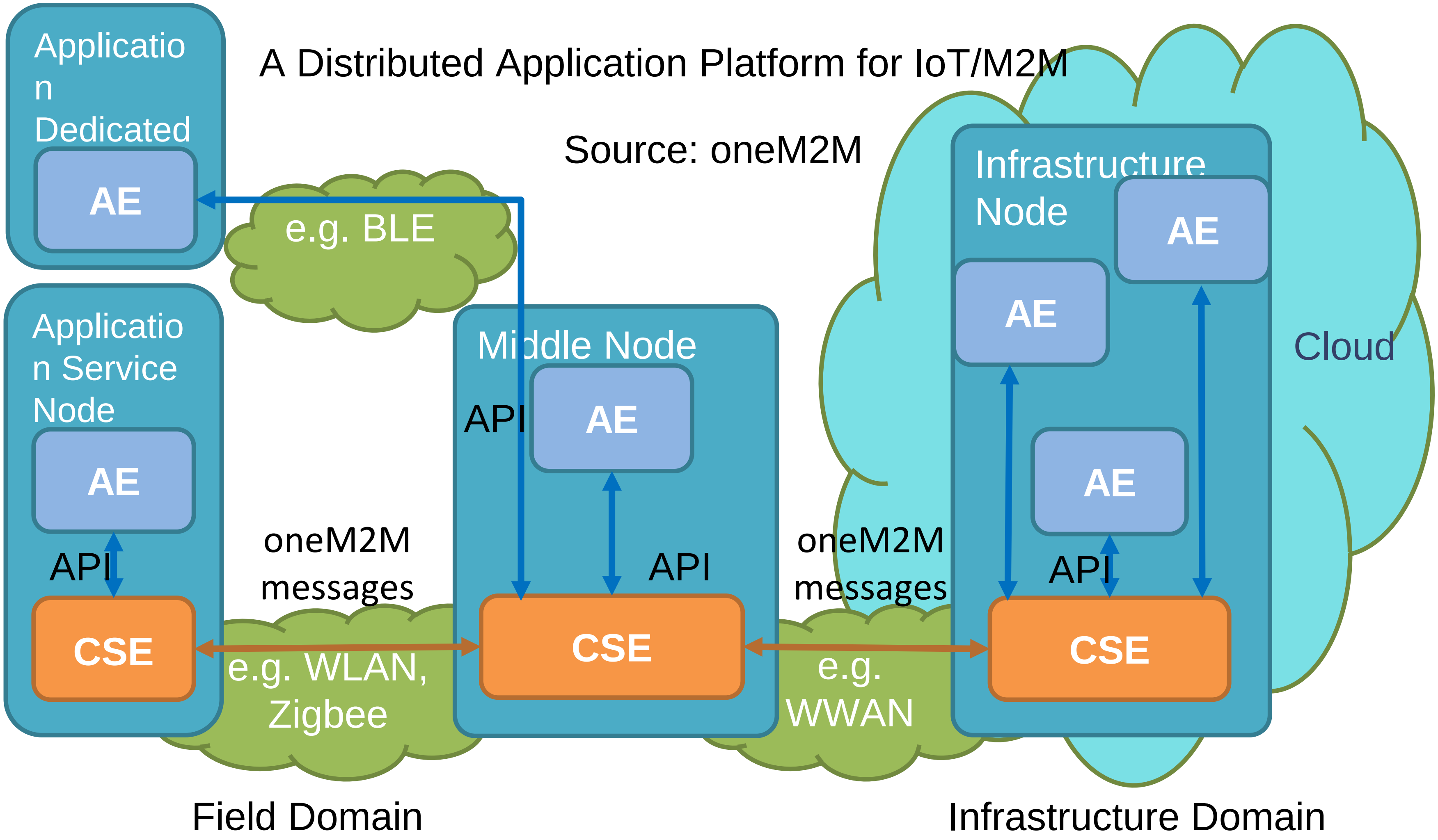
2 REST Architectural Style for M2M

M2M Service Architecture



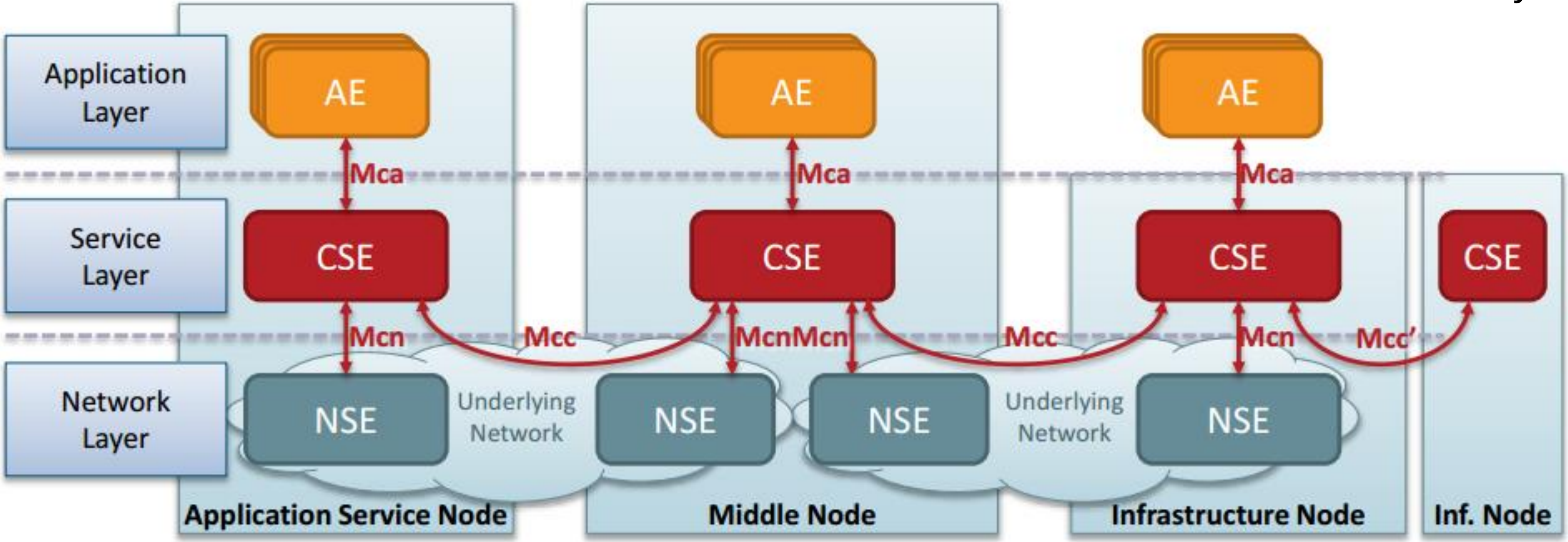
IoT/M2M Service Platform

oneM2M Network



oneM2M Functional Architecture

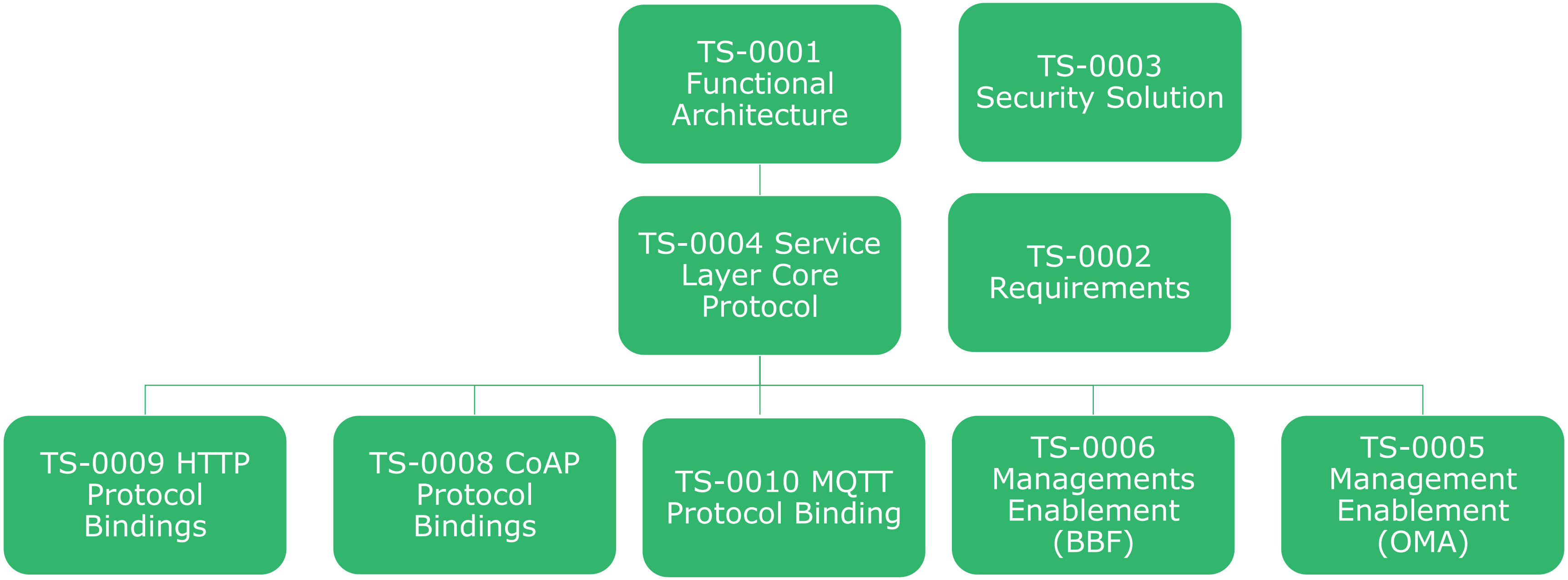
AE: Application Entity
CSE: Common Service Entity
NSE: Network Service Entity



A Distributed Application Platform for IoT/M2M

Source: oneM2M

Release 1 Technical Specifications


TS-0001 Functional Architecture

TS-0003 Security Solution

TS-0004 Service Layer Core Protocol

TS-0002 Requirements

TS-0009 HTTP Protocol Bindings

TS-0008 CoAP Protocol Bindings

TS-0010 MQTT Protocol Binding

TS-0006 Managements Enablement (BBF)

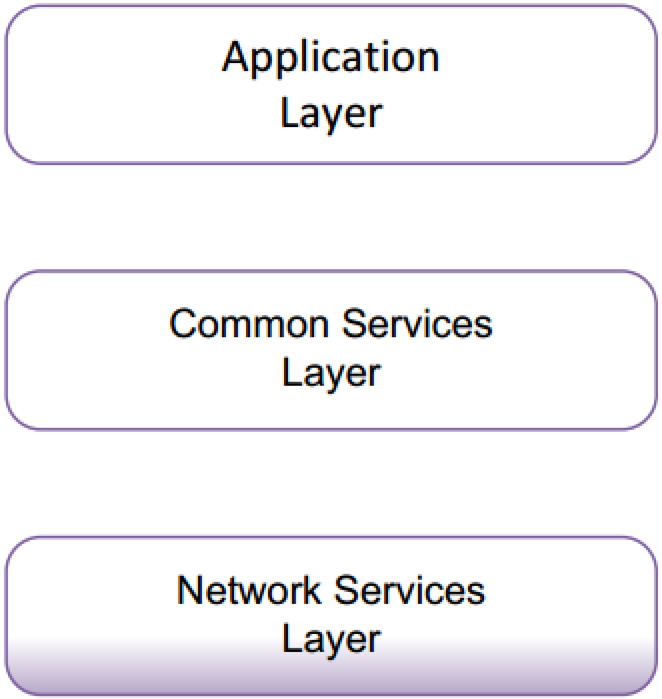
TS-0005 Management Enablement (OMA)

TS-0011 Acronyms and Definitions

Release 1 was released on February 4, 2015.

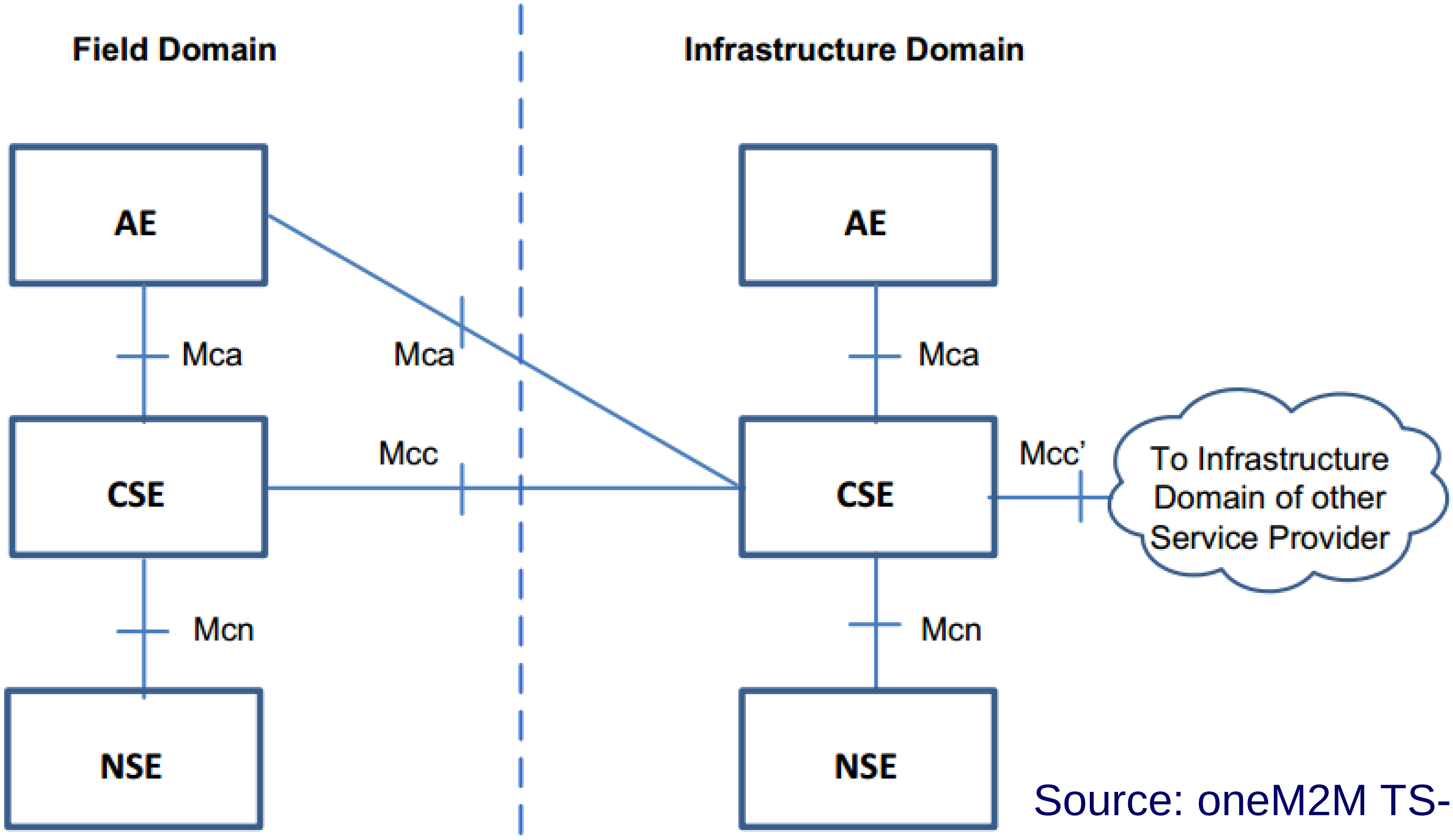
Functional Architecture

- It describes the end-to-end oneM2M functional architecture, including:
  - Functional entities and Reference points.
- It focuses on the Service Layer aspects.
- Underlying Network-independent view of the end-to-end services.
- The Underlying Network is used for the transport of data and potentially for other services.



Source: oneM2M TS-0001

Functional Architecture (2)



Source: oneM2M TS-0001

Note: Mch and mc, mo, ms, Ia reference points not shown

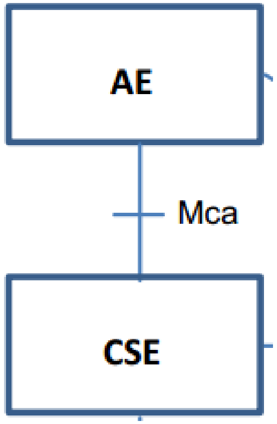
Functional Entities

- Application Entity (AE)
  - An Application Entity represents an instantiation of Application logic for end-to-end M2M solutions.
- Common Services Entity (CSE)
  - A Common Services Entity represents an instantiation of a set of “Common Service Functions" (CSFs) of the M2M environments.

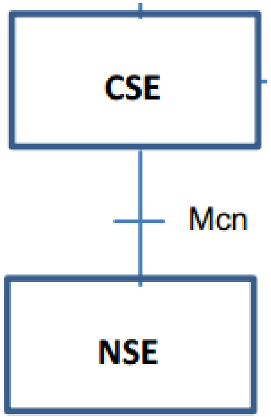
Functional Entities

- Network Services Entity (NSE)
  - A Network Services Entity provides services from the Underlying Network (UN) to the CSEs.
    - Underlying Networks provide data transport services between entities in the oneM2M System.
    - Such data transport services are not included in the NSE.

Reference Points (1)



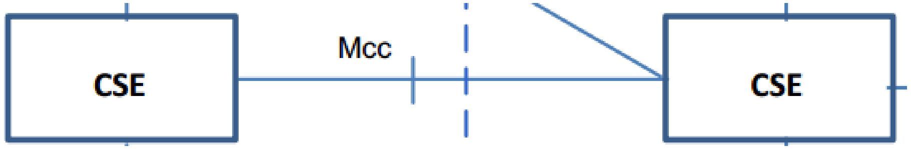
Mca, It enables the AE to use the services supported by the CSE, and for the CSE to communicate with the AE.

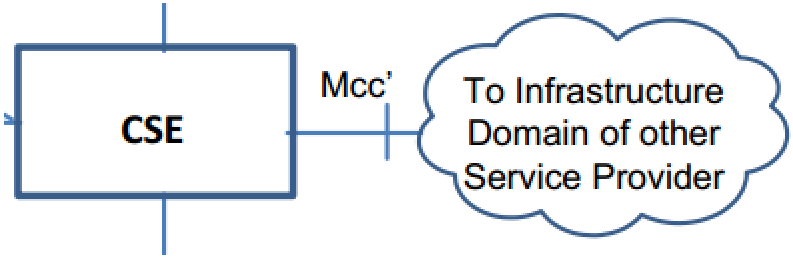


Mcn, It enables a CSE to use the supported services (other than transport and connectivity services) provided by the NSE.

Reference Points (2)

Mcc, it enables a CSE to use the services supported by another CSE. Either on the same or different domains.



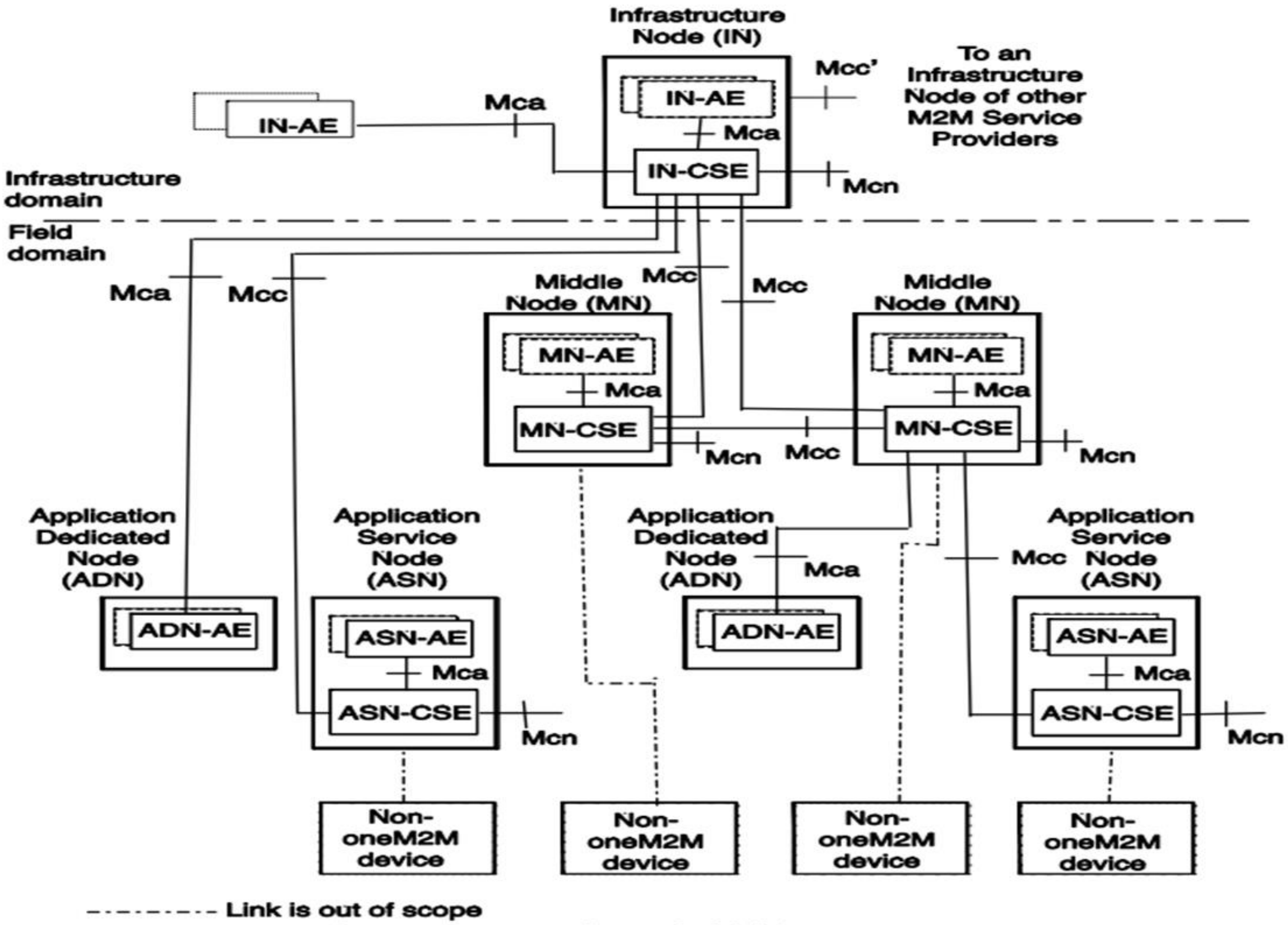


Mcc', It enables a CSE of an infrastructure node residing in the Infrastructure Domain of an M2M Service Provider to communicate with a CSE of another infrastructure node residing in the Infrastructure Domain of another M2M Service Provider to use its supported services, and vice versa.

Reference Points (3)

- Mch, Communication flows which transfer Charging Data Record (CDRs) generated by the Infrastructure Node (IN) to an external charging server cross the Mch reference point.
- The Mch reference point may be mapped to reference points of other specifications, e.g. Rf reference point on a 3GPP Underlying Network.

Overall oneM2M Functional Architecture



Source: oneM2M TS-0001

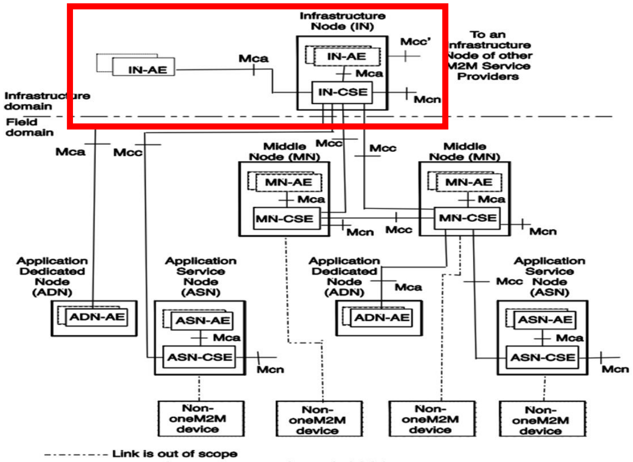
Types of oneM2M nodes

- CSE-Capable Node is a functional entity that contains one oneM2M Common Services Entity and zero or more oneM2M Application Entities.
  - Application Service Node (ASN)
  - Middle Node (MN)
  - Infrastructure Node (IN)

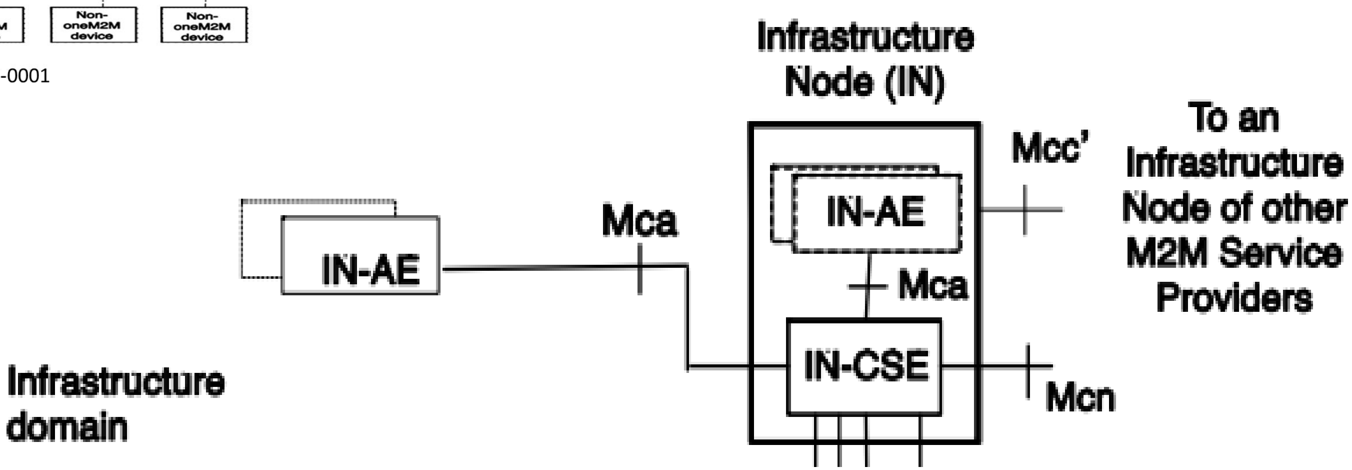
Types of oneM2M nodes

- Non-CSE-Capable Node is a functional entity that contains one or more oneM2M Application Entities and no oneM2M Common Services Entity.
  - Application Dedicated Node (ADN)
  - Application Entity at Infrastructure Domain

Infrastructure Domain

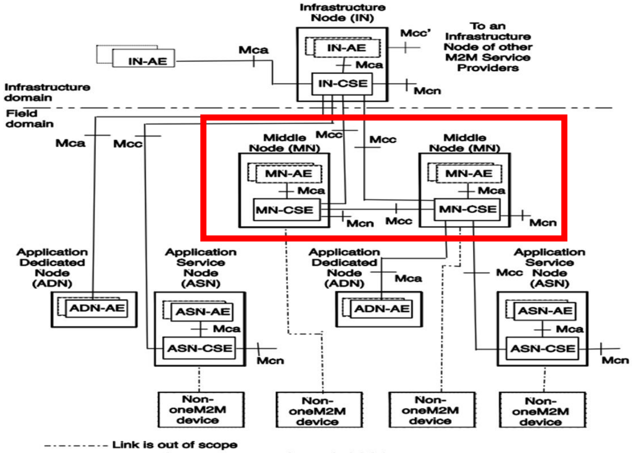


Source: oneM2M TS-0001



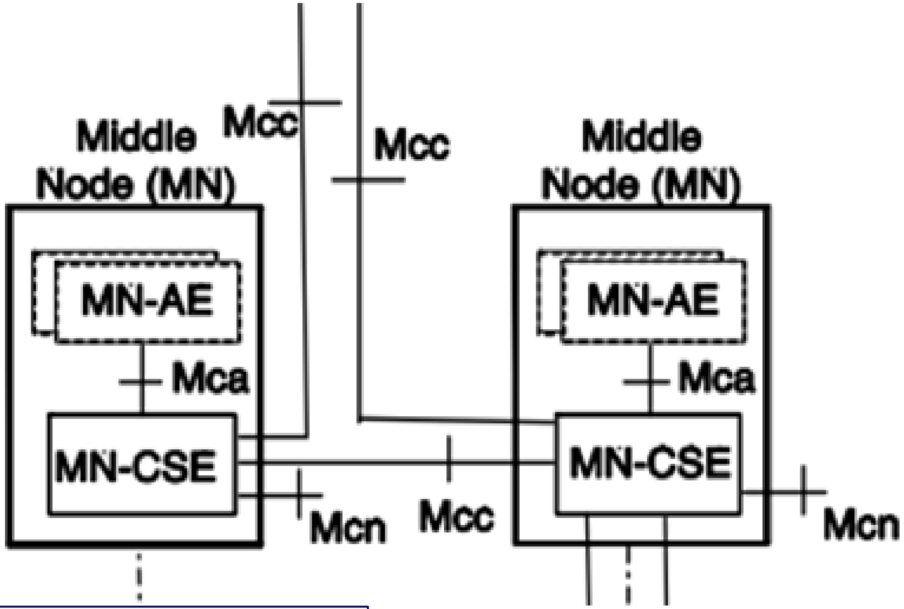
Only one logical IN in the Infrastructure Domain per oneM2M Service Provider of an M2M System

Field Domain (1)



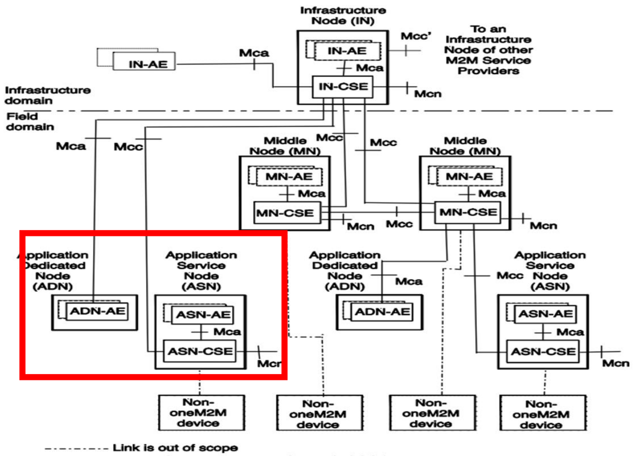
Source: oneM2M TS-0001

IN @ Infrastructure Domain

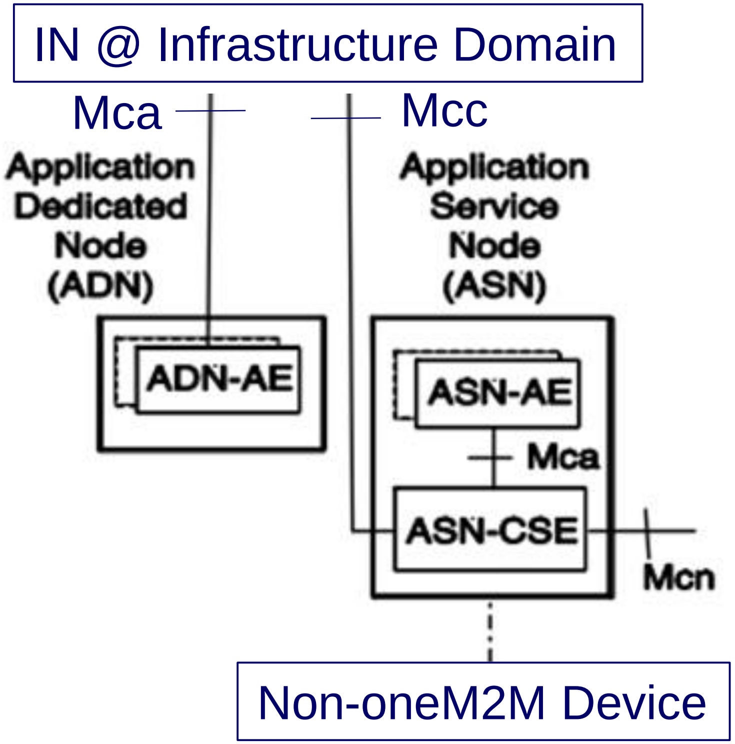


Non-oneM2M Device

Field Domain (2)



Source: oneM2M TS-0001


Application Dedicated Node (ADN)
ADN-AE
Application Service Node (ASN)
ASN-AE
Mca
ASN-CSE
Mcn

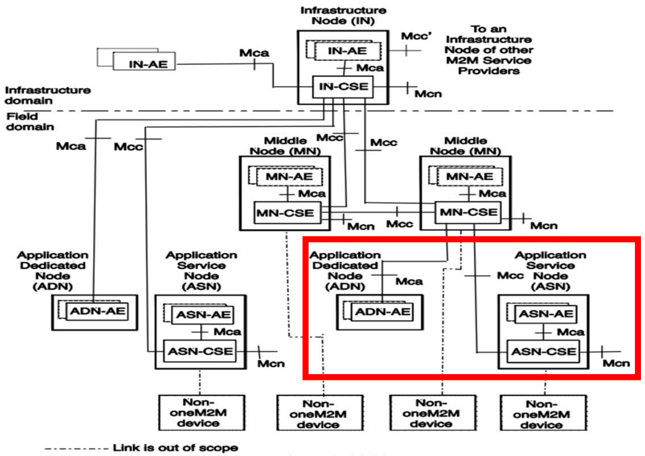
IN @ Infrastructure Domain

Mca

Mcc

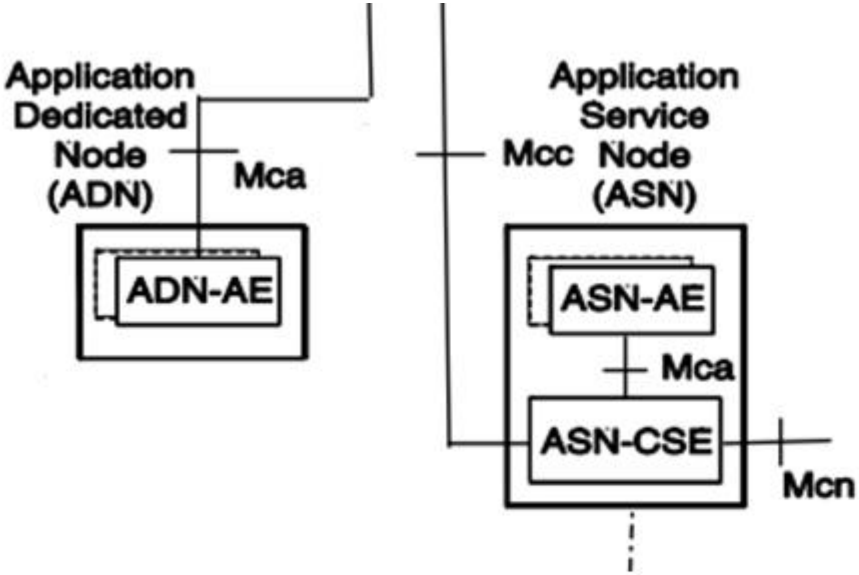
Non-oneM2M Device

Field Domain (3)



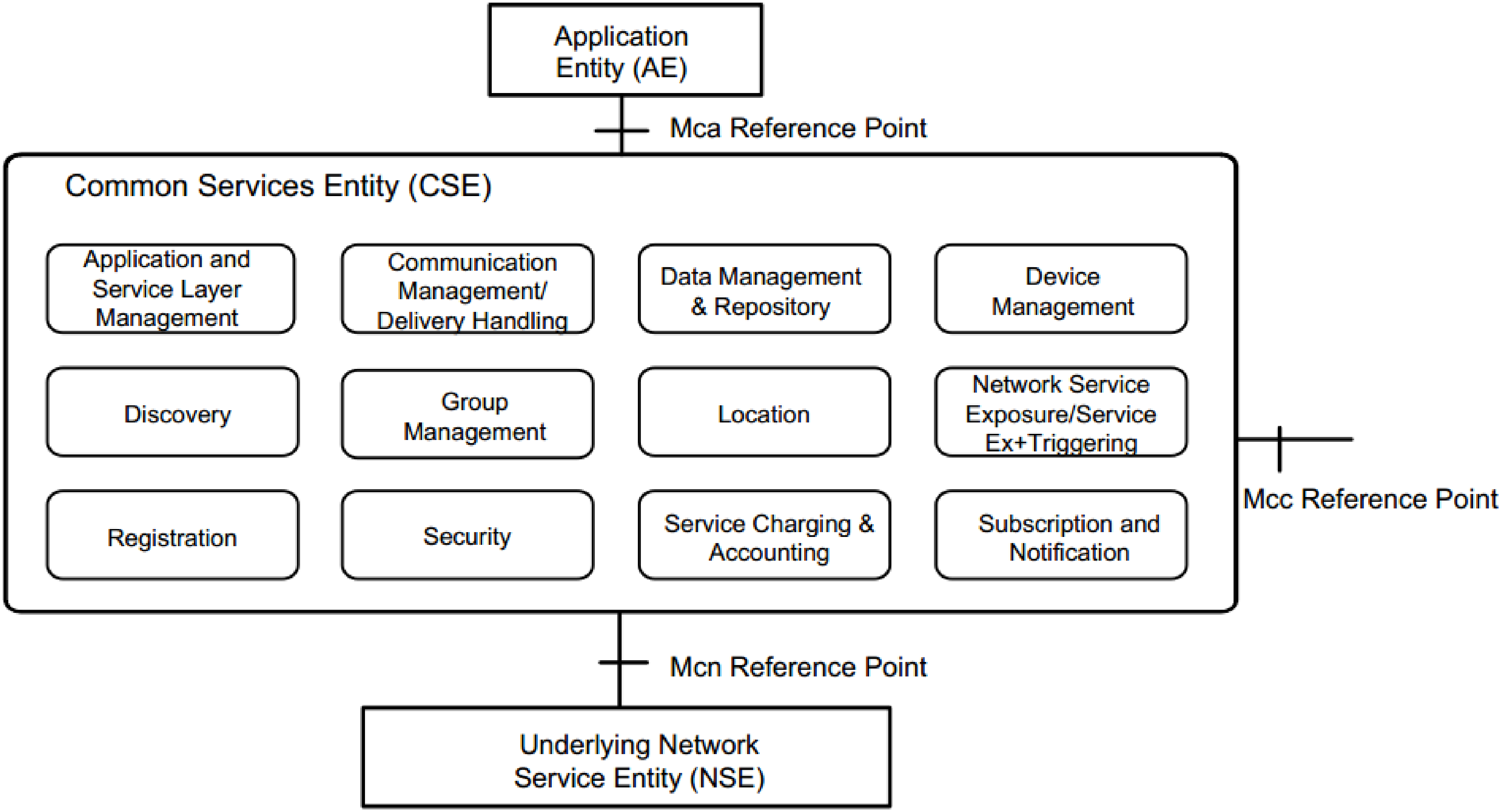
Source: oneM2M TS-0001

MN @ Field Domain


Application Dedicated Node (ADN)
Mca
ADN-AE
Application Service Node (ASN)
Mcc
ASN-AE
Mca
ASN-CSE
Mcn

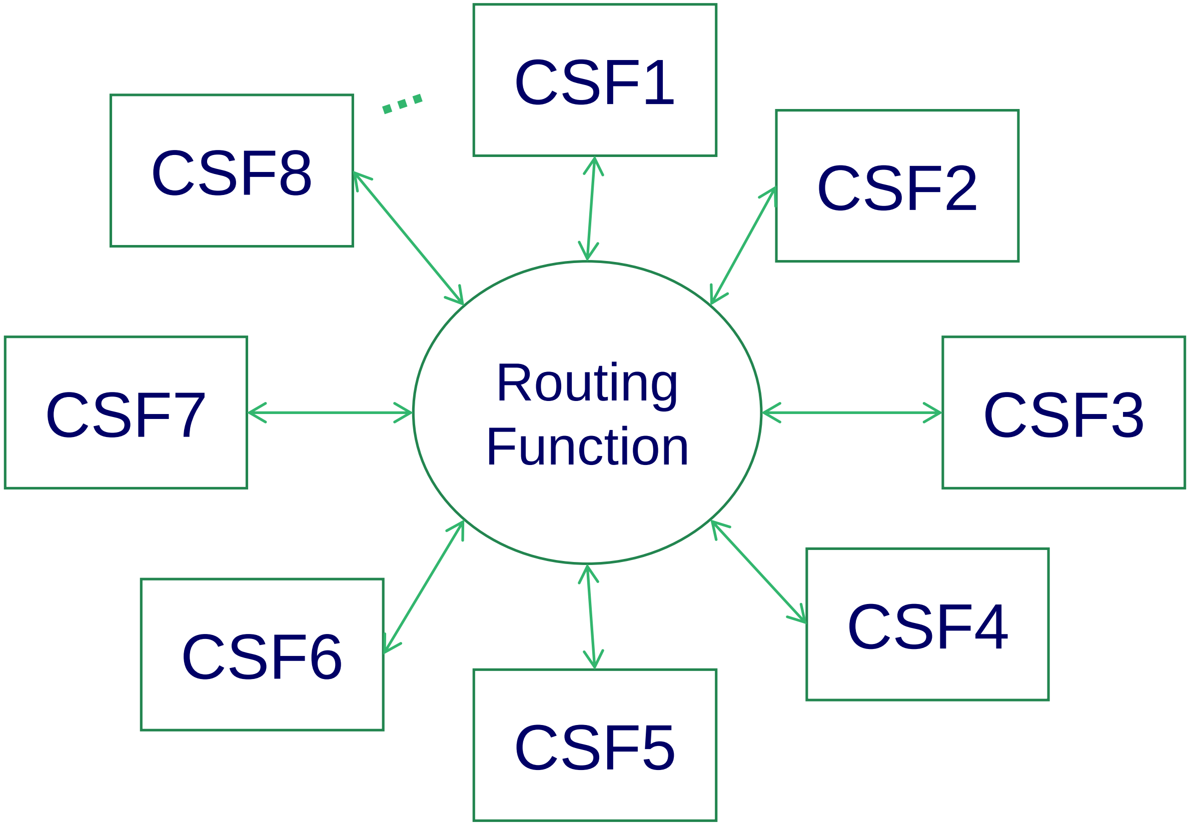
Non-oneM2M Device

Common Service Functions (1)


Application Entity (AE)
Mca Reference Point
Common Services Entity (CSE)
Application and Service Layer Management
Communication Management/ Delivery Handling
Data Management & Repository
Device Management
Discovery
Group Management
Location
Network Service Exposure/Service Ex+Triggering
Registration
Security
Service Charging & Accounting
Subscription and Notification
Mcc Reference Point
Mcn Reference Point
Underlying Network Service Entity (NSE)

Source: oneM2M TS-0001

Common Service Functions (2)



Abbreviation of CSFs

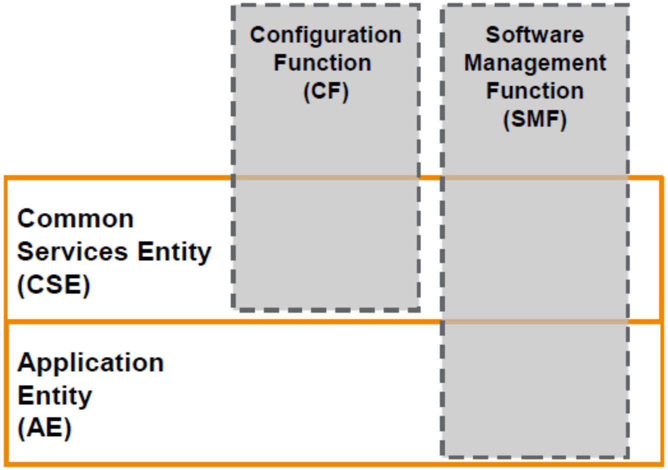
1. ASM – Application and Service Layer Management CSF
2. CMDH – Communication Management and Delivery Handling CSF
3. DMR – Data Management and Repository CSF
4. DMG – Device Management CSF
5. DIS – Discovery CSF
6. GMG – Group Management CSF

Abbreviation of CSFs

7. LOC – Location CSF
8. NSSE – Network Service Exposure, Service Execution and Triggering CSF
9. REG – Registration CSF
10. SEC – Security CSF
11. SCA – Service Charging and Accounting CSF
12. SUB – Subscription and Notification CSF

Application and Service Layer Management (ASM)

- Management of AEs and CSEs on the
  - Application Dedicated Nodes,
  - Application Service Nodes,
  - Middle Nodes,
  - Infrastructure Nodes
- It is divided into two functions:
  - Configuration Function (CF)
  - Software Management Function (SMF)


Configuration Function (CF)
Software Management Function (SMF)
Common Services Entity (CSE)
Application Entity (AE)

Application and Service Layer Management (ASM)

- CF only applies to CSE, while SMF works on both CSE and AE.
- SMF manages software component for application and CSE including the state of the software package
- Management is done through role based permissions.

Communication Management and Delivery Handling (CMDH)

- It enables the communication among CSE, AE, NSE.
- It determines which communication connection to use, and when to buffer request for later (Delivery Handling).

Communication Management and Delivery Handling (CMDH)

- Characteristics
  - Need to access to provisioned delivery handling policies
  - Allow to store requests
  - Processing based on provisioned policies and delivery handling parameters
  - Transparent data delivery
  - Management through role based permissions

IP based protocols

- HTTP (TS-0008)
  - Method is mapped to the oneM2M operation parameter
  - Request-URI is derived from the oneM2M to parameter
  - Status-Code and Reason-Phrase are derived from the oneM2M
  - responseStatusCode parameter of the response primitive
  - This specification supports binding to HTTP 1.1

IP based protocols

- CoAP (TS-0009)
  - Support of blockwise transfers
  - Caching is supported using freshness and validity information
  - carried with CoAP responses
  - Due to limited code field in CoAP, some oneM2M response code
  - shall be carried via CoAP payload field

IP based protocols

- MQTT (TS-0010)
  - Two scenarios for MQTT server location: MQTT server co-located within a node (MN, IN), and MQTT server located independently from nodes

- The received resource representation shall be in in plain XML, binary XML or JSON

Data Management and Repository (DMR)

- It provides data storage and mediation to the M2M entities.
- Characteristics:
  - It grants access based on access control policies.
  - It is able to aggregate data from different entities.
  - Data conversion into specific format for analysis and semantic processing.
  - Data can be raw or processed by intermediate node.

Data Management and Repository (DMR)

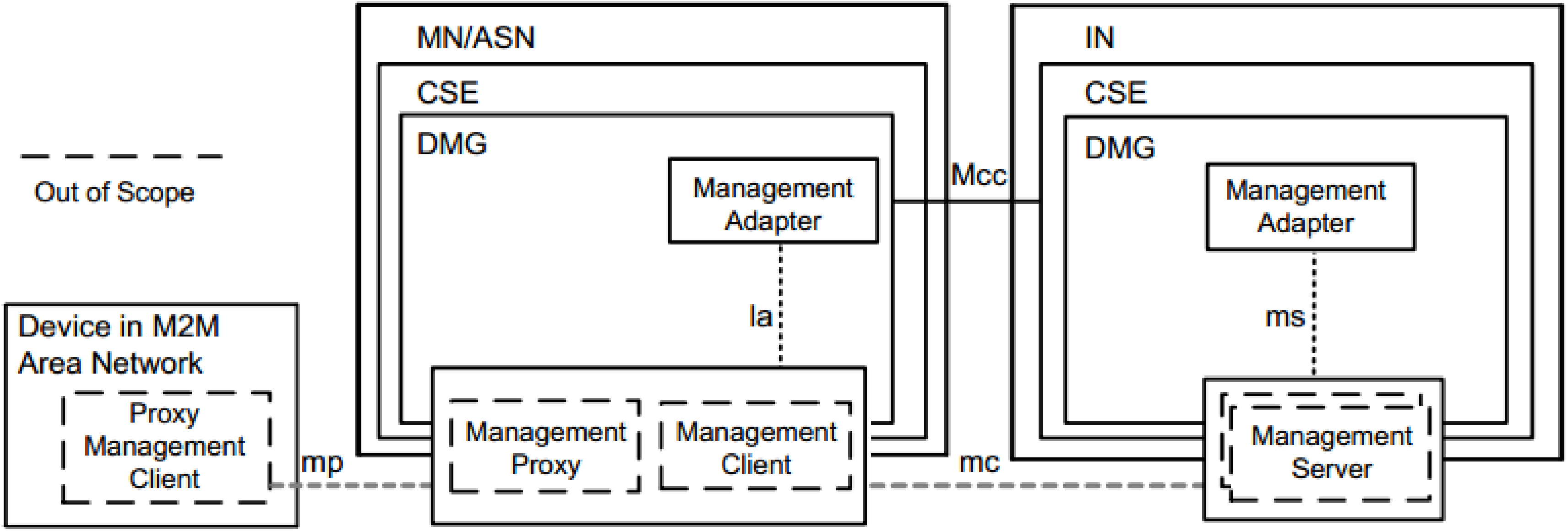
- Support transfer of data to/from the AEs and other CSEs
- Provide a means to perform analytics on large amount of data.
- Provide semantic information and enable functions for annotation
  - Expose M2M resources based on semantic information.
  - Enable application to discover and interpret M2M data from different sources.

Device Management and Repository (DMG)

- To manage the device capabilities on
  - Middle Nodes (M2M Gateways),
  - Application Service Nodes
  - Application Dedicated Nodes reside
  - within an M2M Area network
- Utilize existing device management technologies
  - TR-069, OMA-DM, and LWM2M

Device Management and Repository (DMG)

❖ Device Management Architecture



Source: oneM2M TS-0001

Device Management and Repository (DMG)

- Management Adapter performs translation and adaptation between the DMG and the management technology
  - Interacting with the management server over ms or la interfaces
  - Through this CSF, an AE does not need to know the device management technology of the devices since DMG will handle the command translation accordingly.

Device Management and Repository (DMG)

- DMG capabilities are divided into:
- Device Configuration Function
  - Discovery of a device's management objects and attributes
  - Ability to enable or disable a device capability
  - Provisioning configuration parameters of a device

Device Management and Repository (DMG)

- Device Topology Management Function
  - Management of the topology of the M2M Area Network
- Device Diagnostics and Monitoring Function
  - Troubleshooting through the use of diagnostic tests, alert generation
  - Retrieval of operational status and statistics associated
- Device Firmware Management Function
  - Software lifecycle management for firmware components

Device Management and Repository (DMG)

- Reuse device management capabilities defined by OMA (Open Mobile Alliance) and BBF (Broadband Forum)
  - oneM2M-TS-0005 specifies the usage of OMA DM and OMA LWM2M resources.
  - oneM2M-TS-0006 specifies the usage of the BBF TR-069 protocol.

Discovery (DIS)

- Provides discovery of information/resources residing in a local or multiple remote CSE(s).
  - Supported attributes and resources
  - Permission required
- Supports various discovery methods through the use of "filter criteria"
  - Keyword, identifiers, location, semantic information
  - Subject to access control policies and filter criteria specified by the Originator in the discovery request

Discovery (DIS)

- Response includes actual discovered information or resource address(es)
- Discovery is based on the capabilities defined through the device configuration function

Group Management (GMG)

- Responsible for handling Group related requests.
- Bulk operations includes read, write, subscribe, notify, device management, etc.
- Access control is facilitated by grouping.
- May rely on broadcast and multicast network capabilities.
- Support subscriptions to individual Groups.

Group Management (GMG)

- Management of a Group and its membership
  - Allow to create, query, update, and delete a Group
  - Allow to retrieve the information (e.g. URI, metadata, etc.) of a Group and its associated members
  - Allow to add or remove members to and from a Group's member list
- Only M2M Applications or CSEs with common role shall be included in the same Group
  - role based access control

Location (LOC)

- It allows AEs to obtain location information of M2M entities such a ASN, MN, etc.
- Location information could be more than latitude and longitude coordinates.
- To obtain location information, this CSF interacts with:
  - GPS module on devices
  - Location technology from the Underlying Network (if available)
  - Location information existent in the node themselves.
- It is also responsible of protecting confidentiality of geographical location information.

Network Service Exposure, Service Execution and Triggering (NSSE)

- Manages communications with the Underlying Networks for accessing network service functions over the Mcn reference point.
- The network service functions provided by the Underlying Network include service functions such as:
  - device triggering, small data transmission, location notification, policy rules setting, location queries, IMS services, device management, etc.
  - Such services do not include the general transport services

Network Service Exposure, Service Execution and Triggering (NSSE)

- For example, for the 3GPP networks, to provide the network services based on control plane signaling mechanisms
  - It may have interface to 3GPP components such as PCRF, ANDSF and HSS.

- Hide the underlying network technology from the AN and CSF

Registration (REG)

- Handling an Application or another CSE to register with a CSE in order to allow the registered entities to use the services offered by the registered-with CSE.
- CSE in the field domain shall register with the CSE in the infrastructure domain.
  - Enabling peering relationship
- Application in both domains shall register with local CSE.
- Registration information include
  - ID
  - Reachability schedule
- If the later is not present, it will assume the resource is always available.

Security (SEC)

- Sensitive Data Handling functionality
  - Protects the local credentials on which security relies during storage and manipulation.
  - Several cryptographically separated security environments.
- Security Administration functionality
  - Creation and administration of dedicated security environment
  - Post-provisioning of a root credential
  - Provisioning and administration of subscriptions related to M2M Services

Security (SEC)

- Security Association Establishment functionality
  - Between corresponding M2M nodes
- Authorization and Access Control functionality
  - According to provisioned security policies and assigned roles
- Identity Protection Functionality
  - Unique identifier
  - Support pseudonym identifier without linking to true identity

Service Charging and Accounting (SCA)

- It provides charging functions for the Service Layer.
- Support online and offline charging
- Capturing chargeable events, recording of information, generating charging records and charging information
- Multiple charging models: subscription based, event based charging (e.g. based on create, update and retrieve), session based charging, service based charging

Service Charging and Accounting (SCA)

- It supports:
  - Independent Service Layer Charging (M2M SL), and
  - Correlated charging (M2M SL and UN)
- It is divided into three functions:
  - Charging Management Function
  - Charging Triggering Function
  - Offline Charging Function

Subscription and Notification (SUB)

- Manage subscriptions to resources, subject to access rights, and send corresponding notifications to the watcher address(s).
- Watcher address(s) represents the resource subscribers that want to receive the notification.
- An AE subscribes to resources on a local or remote CSE.
- A CSE subscribes to resources on multiple CSEs.

Subscription and Notification (SUB)

- Subscription scope includes the subscribed resource, and its attributes and child nodes (however it does not include child nodes attributes)
- It supports filter criteria and group subscription.
- Subscription request includes Subscription ID, Hosting CSE-ID, Subscribed-to resource addresses, Criteria (optional), the address(s) for sending the notification.

What’s REST?

- REST (REpresentational State Transfer)
  - The term representational state transfer was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation.
  - Conforming to the REST constraints is generally referred to as being "RESTful”.

Representational State Transfer

- In REST, requests and responses are built around the transfer of representations of resources. A representation of a resource is typically a document that captures the current or intended state of a resource.

REST Architectural Style

- The REST architectural style imposes certain constraints, while leaving the implementation of the individual components free to design.
- The key goals are
  - Scalability of component interactions
  - Generality of interfaces
  - Independent deployment of components
  - Intermediary components to reduce latency, enforce security and encapsulate legacy systems

REST Constraints (1)

- Client–server with a uniform interface separating clients from servers.
  - Clients are not concerned with data storage, which remains internal to each server, so that the portability of client code is improved.
  - Servers are not concerned with the user interface or user state, so that servers can be simpler and more scalable.
  - Servers and clients may also be replaced and developed independently, as long as the interface between them is not altered.

Source: Wikipedia

REST Constraints (1)

- Stateless:
  - The client–server communication is further constrained by no client context being stored on the server between requests.
  - Each request from any client contains all of the information necessary to service the request, and any session state is held in the client

Source: Wikipedia

REST Constraints (2)

- Cacheable:
  - In REST, clients can cache responses.
  - Responses from the servers must therefore, implicitly or explicitly, define themselves as cacheable or not, to prevent clients reusing stale or inappropriate data in response to further requests.
  - Well-managed caching partially or completely eliminates some client–server interactions, further improving scalability and performance.

Source: Wikipedia

REST Constraints (2)

- Layered system
  - A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way. Intermediary servers may improve system scalability by enabling load-balancing and by providing shared caches.
  - They may also enforce security policies.

Source: Wikipedia

REST Resources (1)

- An important concept in REST is the existence of resources (sources of specific information), each of which is referenced with a global identifier (e.g., a URI in HTTP).
- In order to manipulate these resources, components of the network (clients, proxies, servers etc.) communicate via a standardized interface (e.g., HTTP) and exchange representations of these resources (the actual documents conveying the information).

REST Resources (2)

- For example, a resource that represents a circle (as a logical object) may accept and return a representation that specifies a center point and radius, but may also accept and return a representation that specifies any three distinct points along the curve (since this also uniquely identifies a circle) as a comma-separated list.
- Any number of connectors (e.g., clients, servers, proxies, etc.) can mediate the request, but each does so without "seeing past" its own request (based on the REST constraint of "layering“ discussed before).

REST Resources (3)

- As a result, an application can interact with a resource by knowing two things: (1) the identifier of the resource and (2) the action required—it does not need to know whether there are caches, proxies, gateways, firewalls, tunnels, or anything else between it and the server actually holding the information. The application does, however, need to understand the format of the information (representation) returned, which is typically an HTML, XML or JSON document of some kind, although it may be an image, plain text, or any other content.

REST Interfaces (1)

- The uniform interface any REST interface must provide:
  - Identification and representations of resources: Individual resources are identified in requests, for example using URIs in web-based REST systems. The resources themselves are conceptually separate from the representations that are returned to the client. For example, the server does not send its database, but rather, some HTML, XML or JSON that represents a few database records expressed in language and encoding, depending on the details of the request and the server implementation.

REST Interfaces (2)

- Manipulation of resources via their representations: When a client holds a representation of a resource, including any metadata attached, it has enough information to modify or delete the resource on the server, provided it has permission to do so.
- Self-descriptive messages: Each client-server message includes enough information to describe how to process the message. For example, which parser to invoke may be specified by an Internet media type (previously known as a MIME type). Responses also explicitly indicate their cacheability.

Manipulation of Resources in REST

- There are four basic operations:
  - CREATE
  - UPDATE
  - DELETE
  - READ
- Each operation is composed of request and response messages
- Both UPDATE and READ operations are idempotent.

Idempotent Operations

- The end result of the operation is unchanged no matter how many times the operation is repeated.
- Due to no side effects,
  - Resources can be freely distributed.
  - Proxy functions can be freely inserted.
- Implementation of REST in HTTP
  - CREATE => HTTP POST; READ => HTTP GET; UPDATE => HTTP PUT; DELETE => HTTP DELETE

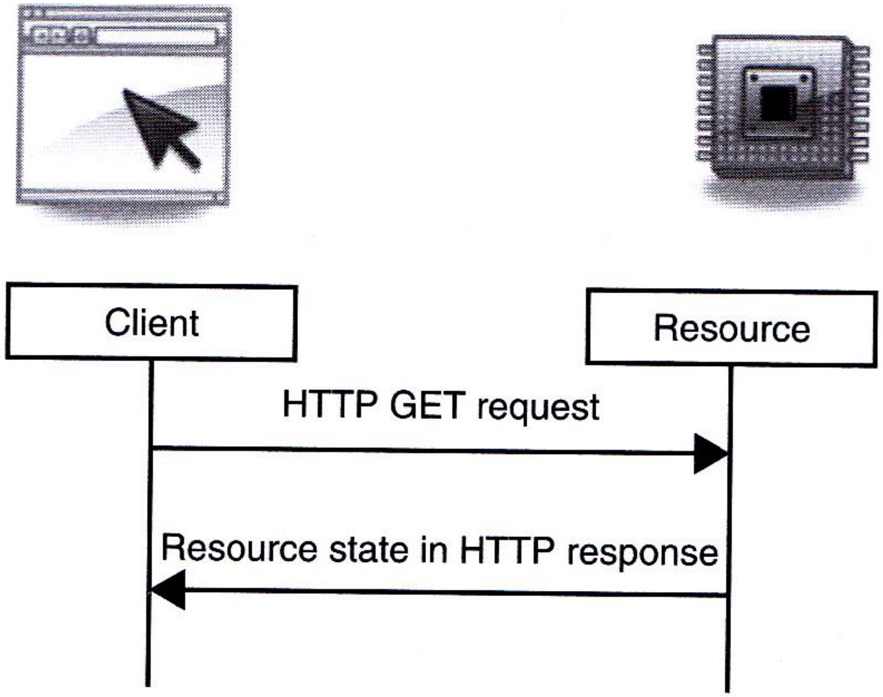
Why REST for M2M?

- REST mapped well to M2M by modeling M2M sensor and devices as REST resources that have particular states to be manipulated.
- The REST-based architecture allows visualization and manipulation of M2M sensors and devices using a web browser. It also allows web mash-ups of information from multiple M2M sensors and devices.

Why REST for M2M?

- Furthermore, scalability is greatly improved because the states transferred to the client can be cached and each communication with the server is stateless. This is particularly true when a proxy is used to serve many M2M sensors or devices.
- Large M2M applications can be developed using simple HTML and Javascript.

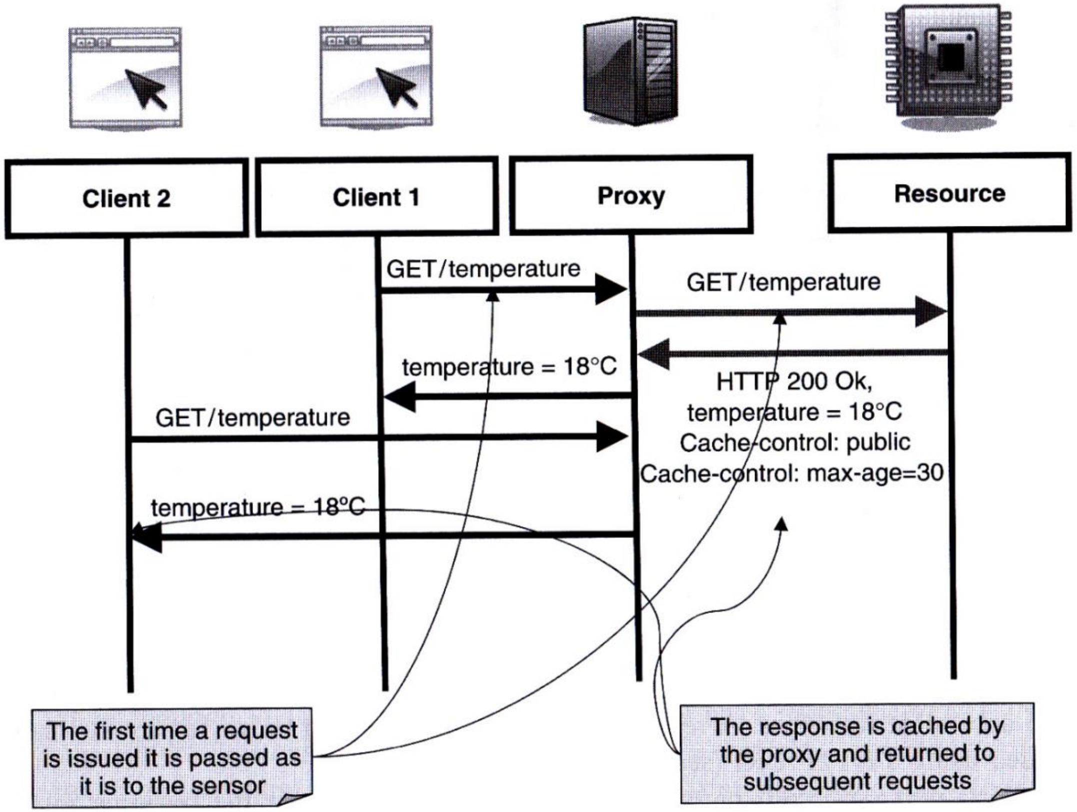
M2M REST – READ via HTTP GET



Source: M2M Communications: A Systems Approach, Wiley, 2012

This shows that HTTP GET request is used to read the state of an M2M resource. The result is returned in an HTTP response.

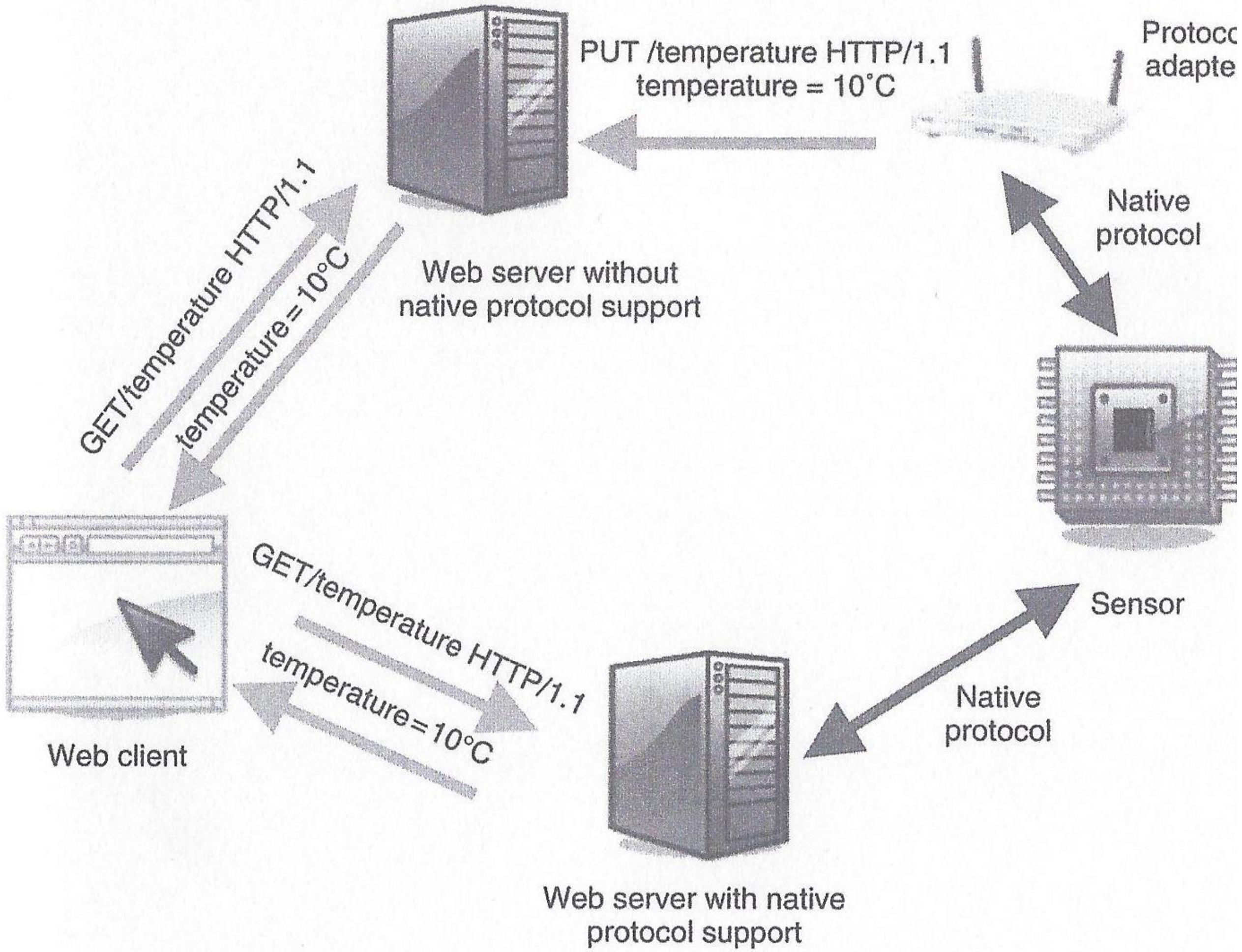
M2M REST – Use of Proxy to READ


Client 2
Client 1
Proxy
Resource
GET/temperature
GET/temperature
temperature = 18°C
HTTP 200 Ok,
temperature = 18°C
Cache-control: public
Cache-control: max-age=30
GET/temperature
temperature = 18°C
The first time a request is issued it is passed as it is to the sensor
The response is cached by the proxy and returned to subsequent requests

Source: M2M Communications: A Systems Approach, Wiley, 2012

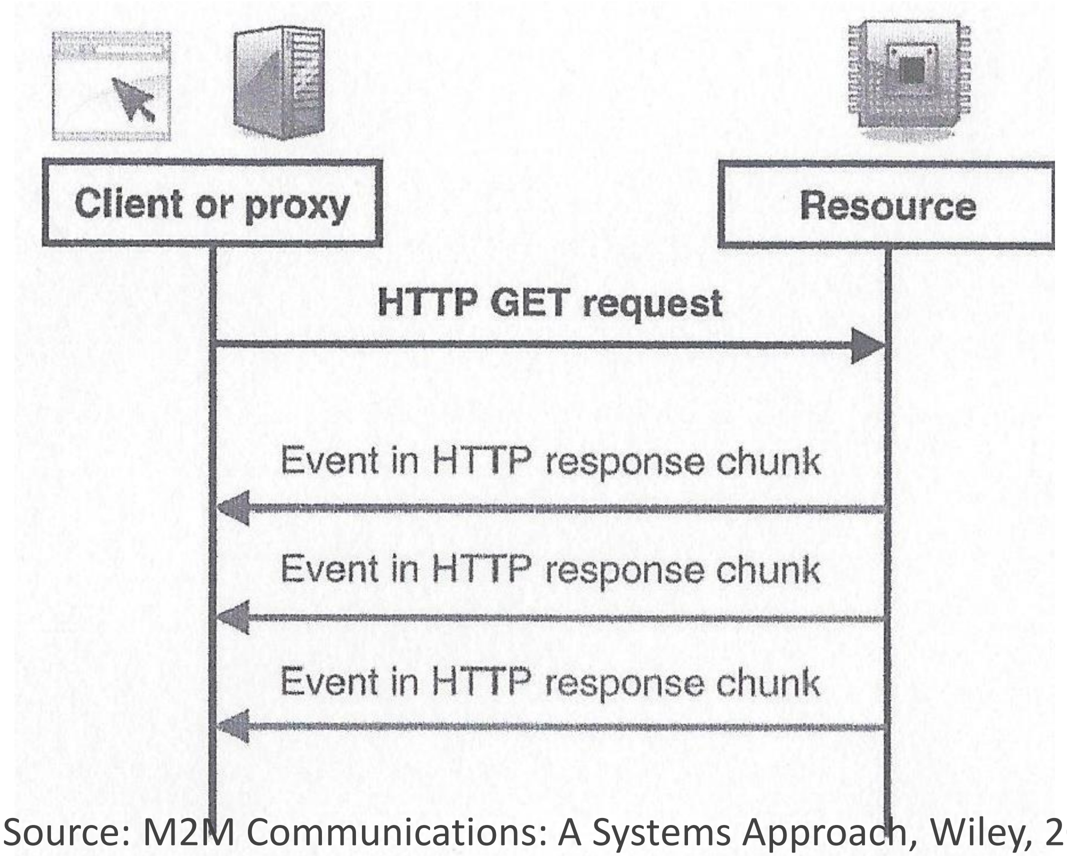
This shows that a proxy is used to cache the result of a GET of Client 1
so that the request from Client 2 can get the result form the cache in the proxy.

M2M REST – Read from Sensors with Native Protocols



Source: M2M Communications: A Systems Approach, Wiley, 2012

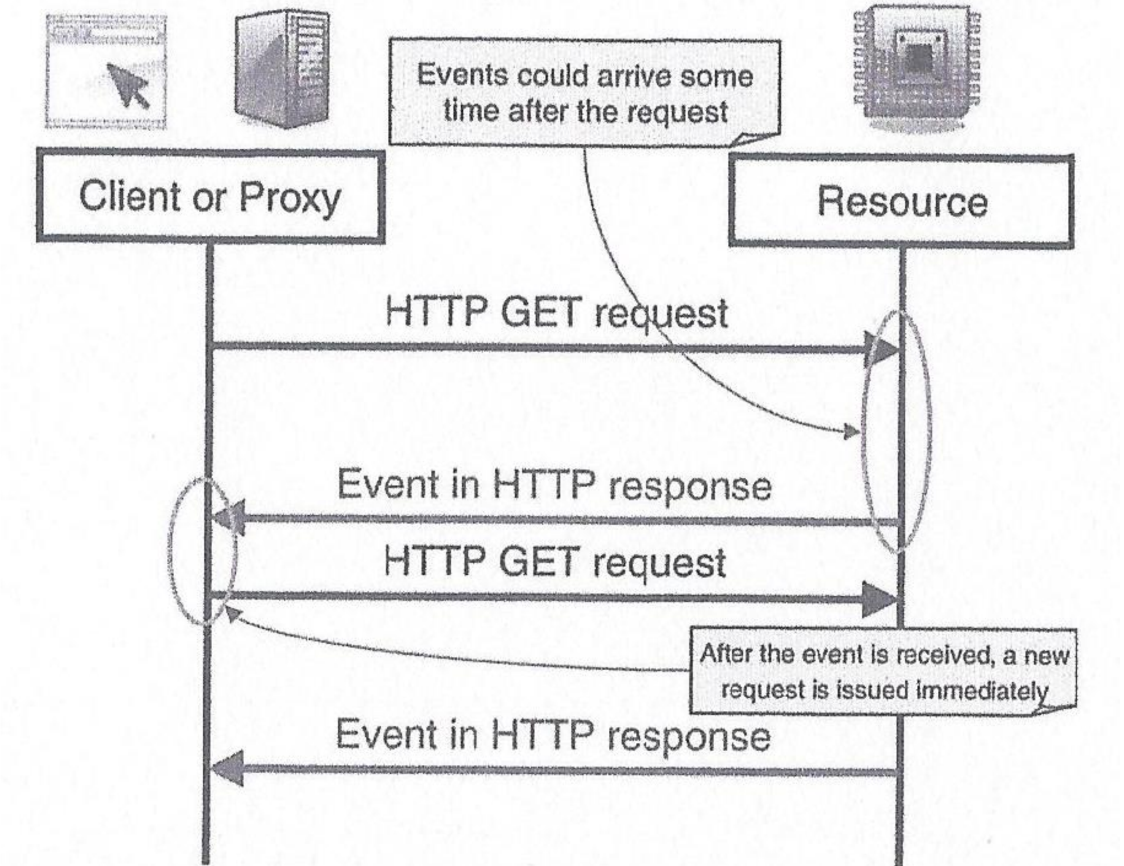
M2M REST – Get Steaming Events from Sensors via Multipart HTTP Responses


Client or proxy
Resource
HTTP GET request
Event in HTTP response chunk
Event in HTTP response chunk
Event in HTTP response chunk

Source: M2M Communications: A Systems Approach, Wiley, 2012

HTTP "Server Push" Technique for sending streaming events (requires support of special MIME type) from M2M resource to client or proxy

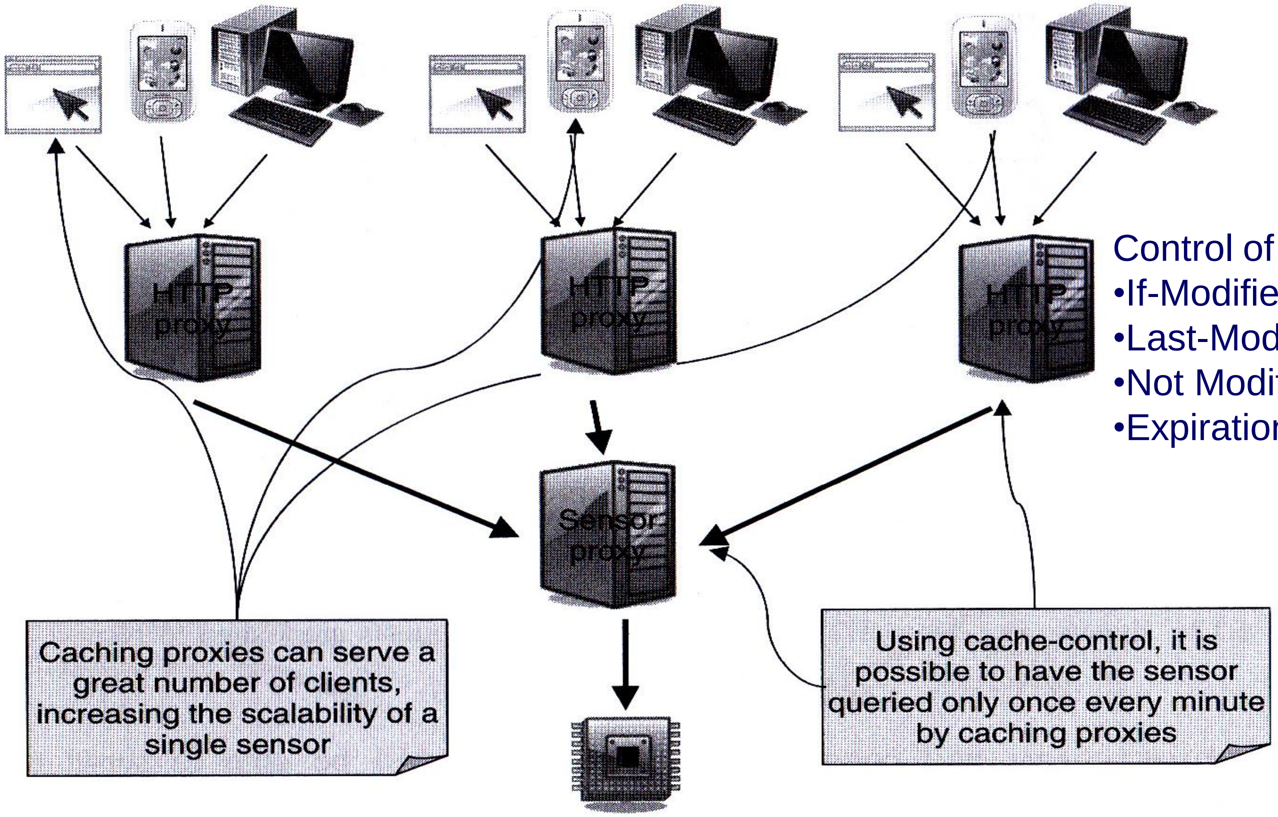
M2M REST – Get Continuous Responses from Sensors via HTTP


Events could arrive some time after the request
Client or Proxy
Resource
HTTP GET request
Event in HTTP response
HTTP GET request
After the event is received, a new request is issued immediately
Event in HTTP response

Source: M2M Communications: A Systems Approach, Wiley, 2012

HTTP "Long Polling" Technique for sending continuous responses (doesn't need support of special MIME type) from M2M resource to client or proxy

M2M REST – Use of Caching Proxies to Increase Scalability of Sensor Reading



Caching proxies can serve a great number of clients, increasing the scalability of a single sensor

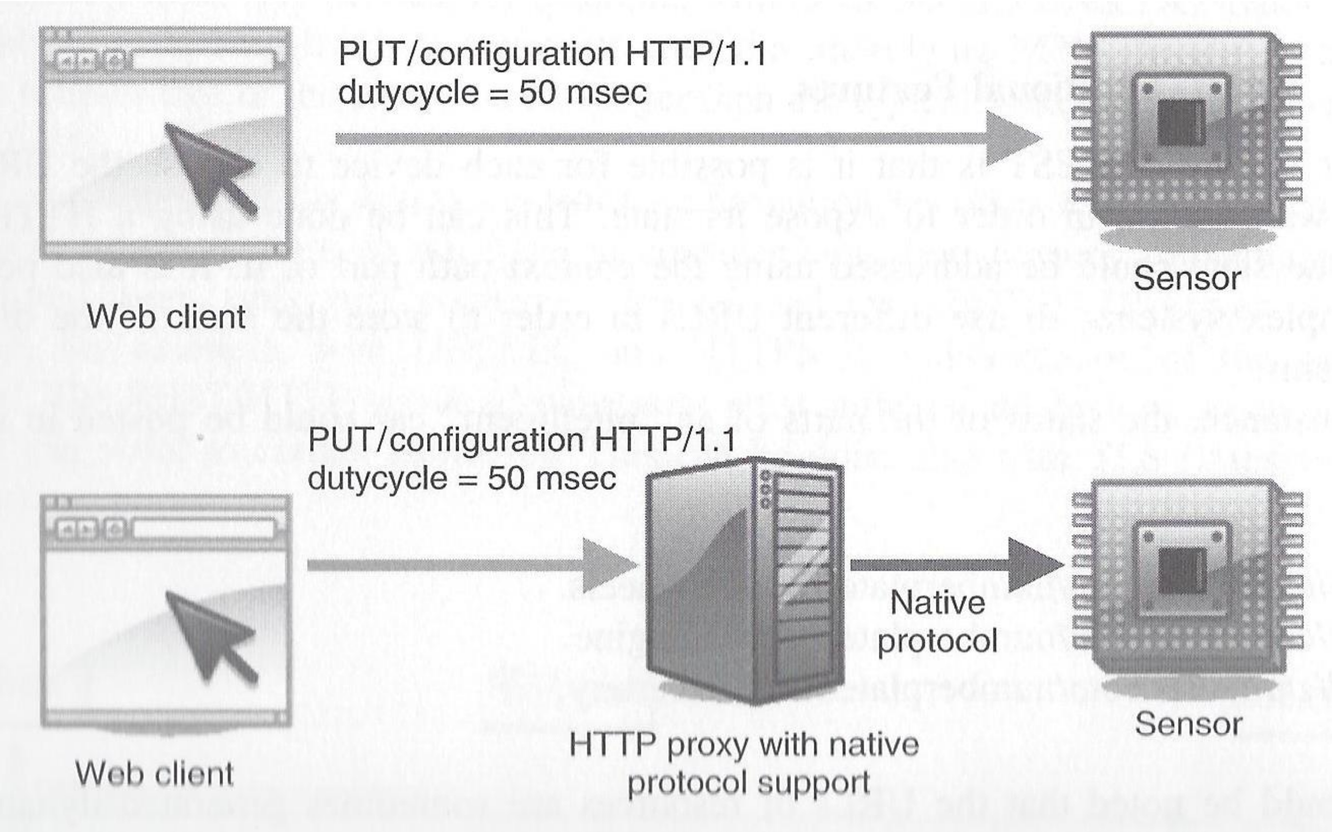
Using cache-control, it is possible to have the sensor queried only once every minute by caching proxies

Control of Caching using HTTP
- •If-Modified-Since Header(Request)
- •Last-Modified Header(Response)
- •Not Modified(Response)
- •Expiration Time(Response)

Source: M2M Communications: A Systems Approach, Wiley, 2012

Two layers of HTTP proxies to increase scalability of access to an M2M sensor

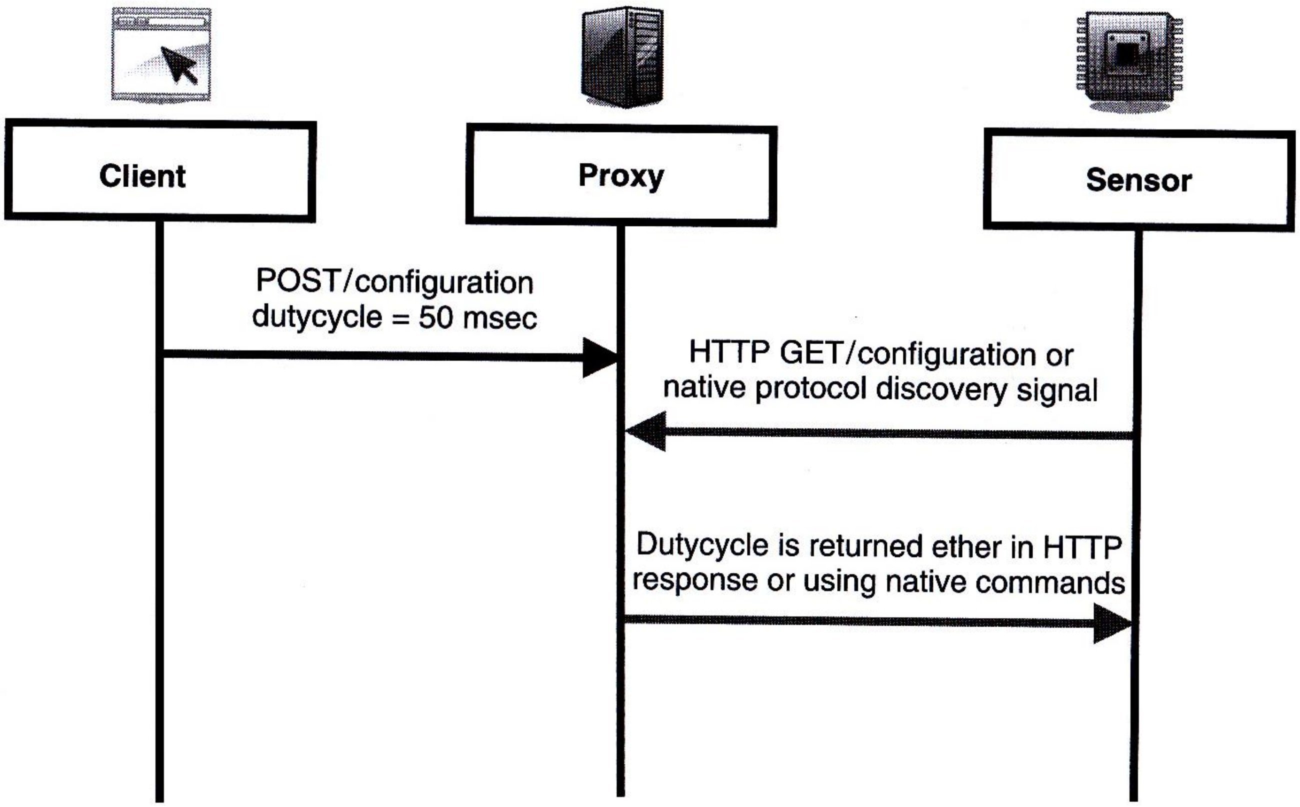
M2M REST – Configure Device via HTTP (1)



Source: M2M Communications: A Systems Approach, Wiley, 2012

Use of HTTP PUT to configure sensor devices:
(1) directly or (2) via device native protocol

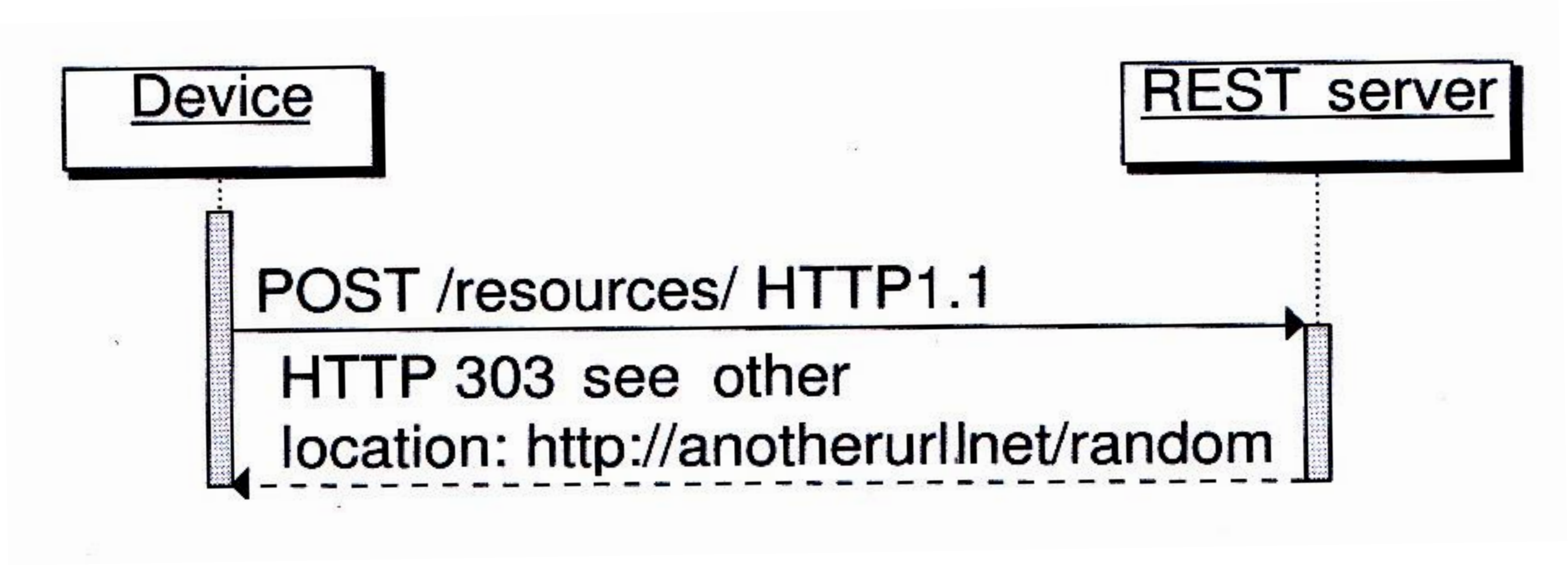
M2M REST – Configure Device via HTTP (2)


Client
Proxy
Sensor
POST/configuration
dutycycle = 50 msec
HTTP GET/configuration or
native protocol discovery signal
Dutycycle is returned ether in HTTP
response or using native commands

Source: M2M Communications: A Systems Approach, Wiley, 2012

Use of HTTP POST to configure sensor devices via a Proxy

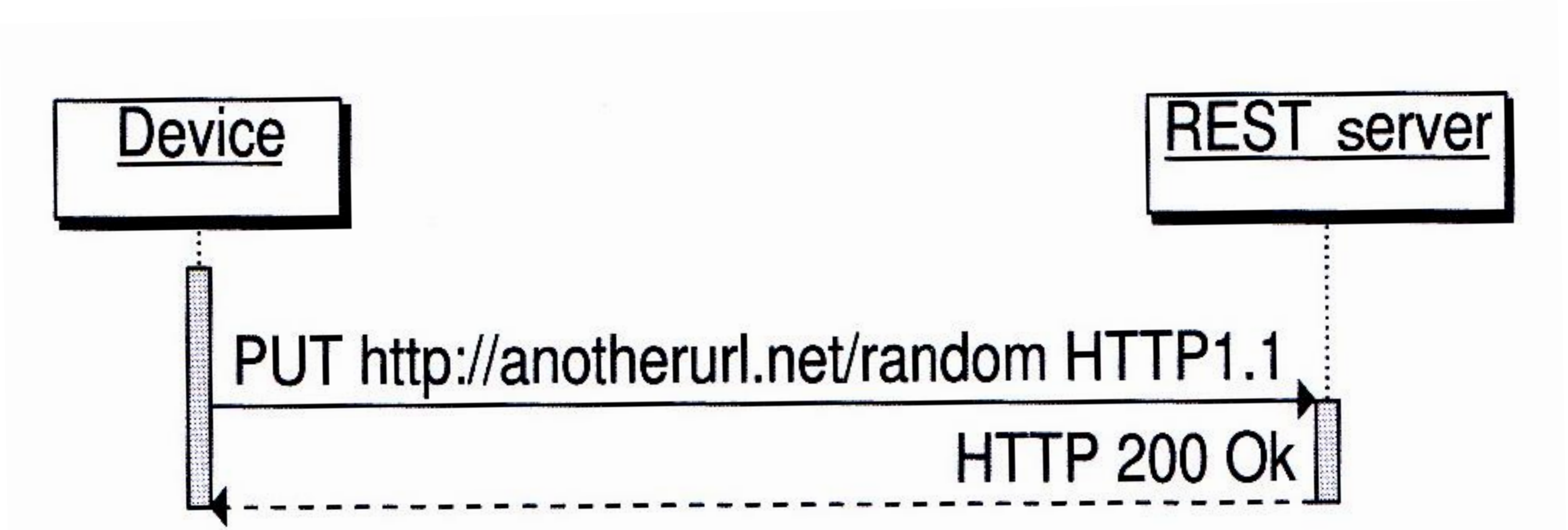
Store M2M Device States in the REST Server (1)



Source: M2M Communications: A Systems Approach, Wiley, 2012

Step 1. Create a resource representation in the REST Server
by sending a POST to the REST Server.
The REST Server returns with the URL of the newly created resource.

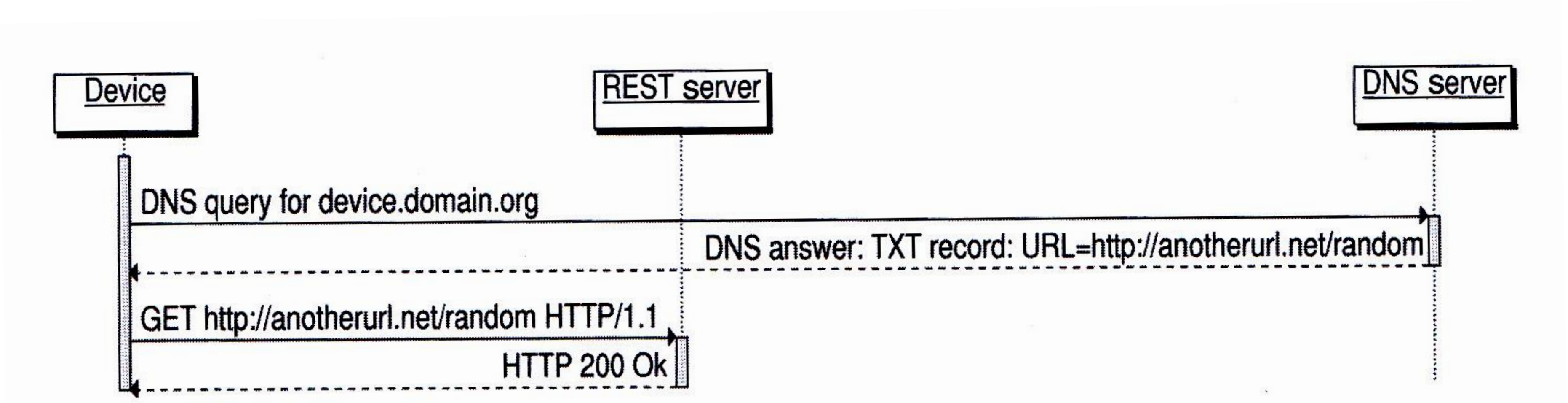
Store M2M Device States in the REST Server (2)



Source: M2M Communications: A Systems Approach, Wiley, 2012

Step 2. Device then stores its state to the newly created resource by sending a PUT to the REST Server.

Retrieve M2M Device States in the REST Server


Device
REST server
DNS server
DNS query for device.domain.org
DNS answer: TXT record: URL=http://anotherurl.net/random
GET http://anotherurl.net/random HTTP/1.1
HTTP 200 Ok

Source: M2M Communications: A Systems Approach, Wiley, 2012

To retrieve the state of an M2M device, the requester first needs to discover the address of the REST server for the device from the DNS server.
It then sends a GET to retrieve the state of the device.

總結

- In this section, we first give an overview of oneM2M Common Service Functions (CSFs) framework.
- We go over each of 12 M2M CSFs.
- Next, a detailed discussion on REST architectural style is given including topics on REST constraints, REST resources, REST interfaces and REST basic operations.
- Then, REST architectural style specific to M2M is elaborated.