CoAP - Constrained Application Protocol

Dheryta Jaisinghani (PhD1211) and Parikshit Maini (MT1214)

April 5, 2013

Agenda

- Internet of Things
- REST framework
- CoRE framework & CoAP protocol
- CoAP working example
- An interconnection framework - Californium (Cf)
- Open research problems

The Internet of Things (IoT): What and Why

- All devices are uniquely identified as objects to integrate and communicate seamlessly.
- The huge address space of ipv6 has lead to the speculation and belief that internet would become the internet of things and we would have a multitude of smart devices around us in the near future, all intelligently communicating with each other and interacting with the environment.

The Internet of Things: How

- The current network and data communication protocols are more suited for computer systems with relatively powerful processors and abundant storage space.
- The IoT model would include a majority of embedded resource constrained devices with limited processing power and resources.
- These devices would require new protocols more suited for the embedded devices. Here, IoT may also be referred to as *Web of Things*.

Protocol Stack

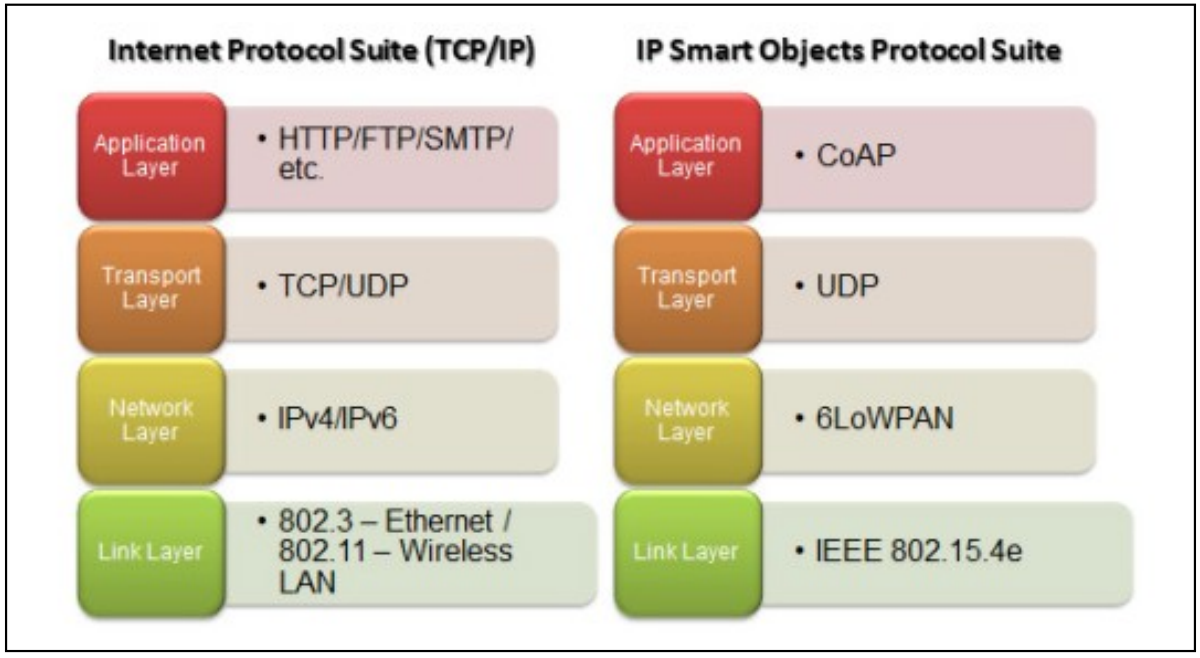

Figure : Normal TCP/IP Stack and IoT Stack, Image Courtesy [1]

How elements communicate over internet?

- HTTP (Hyper Text Transfer Protocol) over REST (Representaional State Transfer) [6].
- CoAP (Constrained Application Protocol) over CoRE (Constrained RESTful Environment)

RESTful Architecture [7]

Tim Berners Lee [1996] said "Webs major goal was to be a shared information space through which people and machines could communicate."

- Provides architectural constraints to provide independence and scalability, as well as minimize communication and latency.
- Launched in October 1994.
- *WWW* Requirements: Low entry barrier like hypermedia to have simplicity of use, Extensibility, Distributed Hypermedia, Internet scale, Evolving requirements.
- An abstraction of architectural elements(Processing elements/components, Data elements, and connecting elements/connectors) of distributed hypermedia system is REST.

RESTful Architecture - Constraints [8]

- Client-Server Model
- Stateless
- Cache driven
- Uniformity
- Layered architecture
- Code-on-demand

RESTful Architecture - Elements [7]

- Data elements
  - Resources & URIs
  - Representations
- Processing elements
  - User Agent
  - Client Agent
  - Proxies
  - Gateways
- Connecting elements
  - Client
  - Server
  - Cache
  - Resolver
  - Tunnel

RESTful Architecture - Views [8]

Views describe how architectural elements collaborate.

- Process View - Path of data is revealed.
- Connector View - Mechanics of communication between components.
- Data View - Application state as the information flows through the system.

RESTful Architecture - Advantanges [7]

- System performance depends on communication between components in web and other distributed network applications due to REST web based applications could scale from 100,000 requests per day to 600,000,000 requests per day.
- Modern web applications could be designed and deployed in a standardized way.
- Robustness of the system is also enhanced.
- Due to its simplicity information could be accessed on any browser.
- Does not require higher bandwidth.

CoRE - Constrained RESTful Environment Architecture

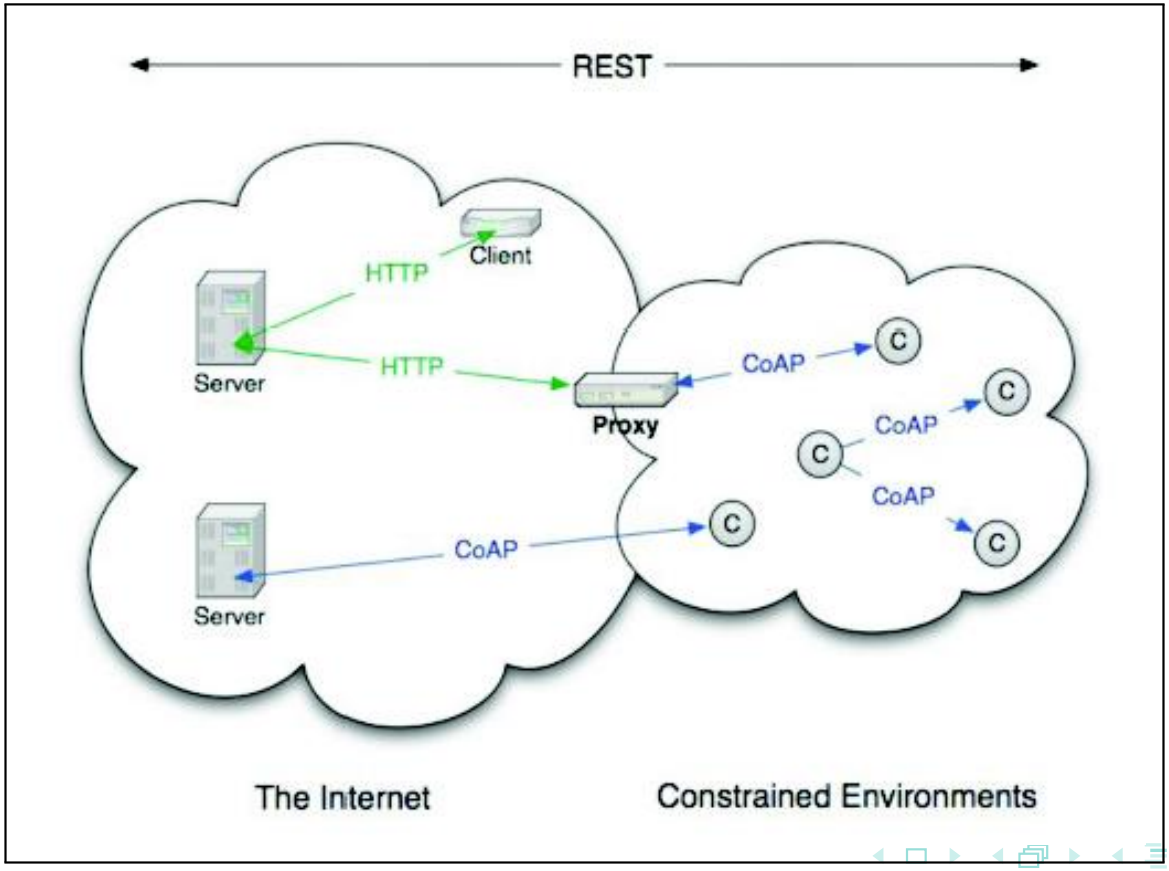

Figure : CoRE Architecture Image Courtesy [9]

Why CoAP is over CoRE & not REST [2, 4]

- CoRE - Constrained RESTful Environments like 6LoWPAN.
- Required due to complex HTTP headers.
- TCP performance issues.
- Sensor do not work on pull model.
- Complex XML parsing in native REST.
- Embedded devices are contrained in terms of memory, computation and energy. REST is not aimed for such devices.
- Fragmentation of IPv6 needs to be minimized.
- Header and parsing complexity needs to be reduced.

What is CoAP ? [2]

- CoAP - Constrained Application Protocol.
- Specialized web transfer protocol.
- Devised for contrained and low power networks.

Why CoAP [3, 4]

- Current web technologies do not consider memory, energy and computation constraints of embedded devices.
- CoRE group from IETF works on developing RESTful application layer protocol - CoAP[1] .
- No consensus on a common application layer due to huge variety of manufacturers of these embedded devices is one of the reasons for this.
- Need for a common application layer for resource constrained devices formed the motivation for CoAP.

[1] Details on next slide

Introduction to CoAP[2, 4]

- CoAP provides request/response interaction model as in *WWW - HTTP*.
- CoAP helps in integration with existing web along with meeting special needs of constrained devices.
- CoAP is based on UDP, supports asynchronous messages, low overheads, URI & content type support and provides simple proxy and caching possibilities.

CoAP: Abstract Layer [2]

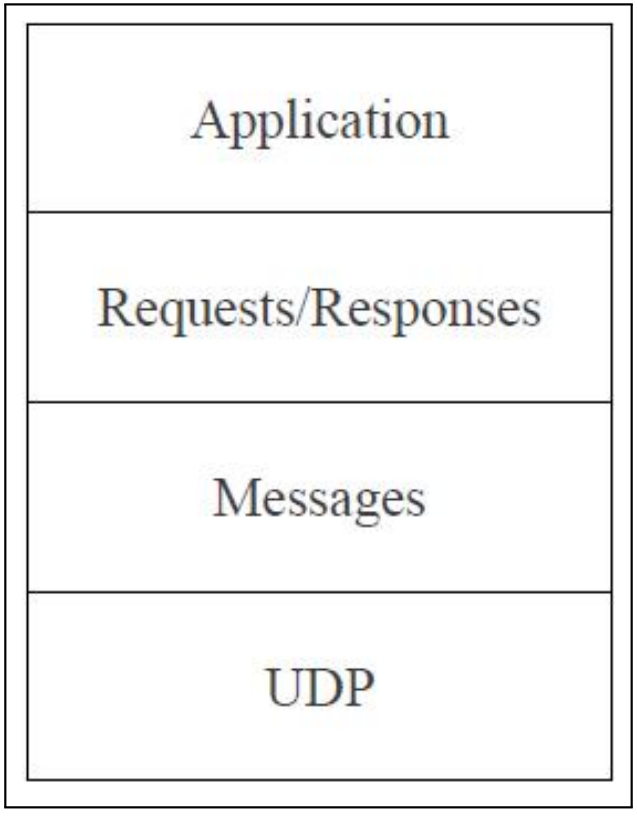

Figure : Abstract Layering

CoAP: Terminology [2]

- Endpoint - Partcipating entity.
- Sender - Orginating entity.
- Recipient - Terminating entity.
- Client - Originating endpoint of request.
- Server - Destination endpoint of response.
- Origin Server - Resource resides here.
- Intermediary - Common endpoint which acts as both client and server.
- Proxy - Forwards requests and performs caching. Forward - endpoint selected by client, Reverse - endpoint stands in for other servers , Cross - proxy to translate between protocols.

CoAP: Terminology ... Contd. [2]

- Message - Confirmable - messages which require acknowledgemet, Non-Confirmable - messages do not require acknowledgement, Acknowledgement - acknowledges reciept of confirmable message, Reset - acknowledges reciept of confirmable and non-confirmable messages.
- Response - Piggy-backed, Separate.
- Options - Critical - endpoint should undertand for proper decoding, Elective - option could be ignored by endpoint, Safe - proxy can forward it even if it does not understand , Unsafe - proxy would not forward unless it understands.
- Resource Discovery - Getting list of associated resources from server.
- Content Format - Format of all packets.

CoAP: Outline [2]

- Messaging Model - Messages are exchanged over UDP endpoints, a confirmable message adds to the reliability.
- Request/Response Model - Requests and Responses are carried in CoAP messages. Method Codes for Requests and Response Codes for Responses. GET, PUT, POST and DELETE methods are used(similar to HTTP). Client parses the URI to get host, port, path and query components, thus URI support is simplied.
- Intermediaries and Caching - Responses are cached for faster reply to requests, proxies may be used which help in reducing network traffic.
- Resource Discovery - Machine to machine interactions require resource discovery, which is done using CoRE link format.

CoAP: Features [2, 4]

- TCP complexities are reduced by using UDP.
- Request Methods: GET, POST, PUT, DELETE.
- Response Methods: 2.xx (success), 4.xx (client error), 5.xx (servererror).
- Message types: Confirmable, Non Confirmable, Acknowledgement and Reset.
- Unicast and multicast requests.
- Resource discovery capability.
- URI representations for resources.
  coap-URI= “coap:” “//” host [“:” port] path-abempty [“?” query]
- Block transfers for large files.

CoAP: Communication Example [2]

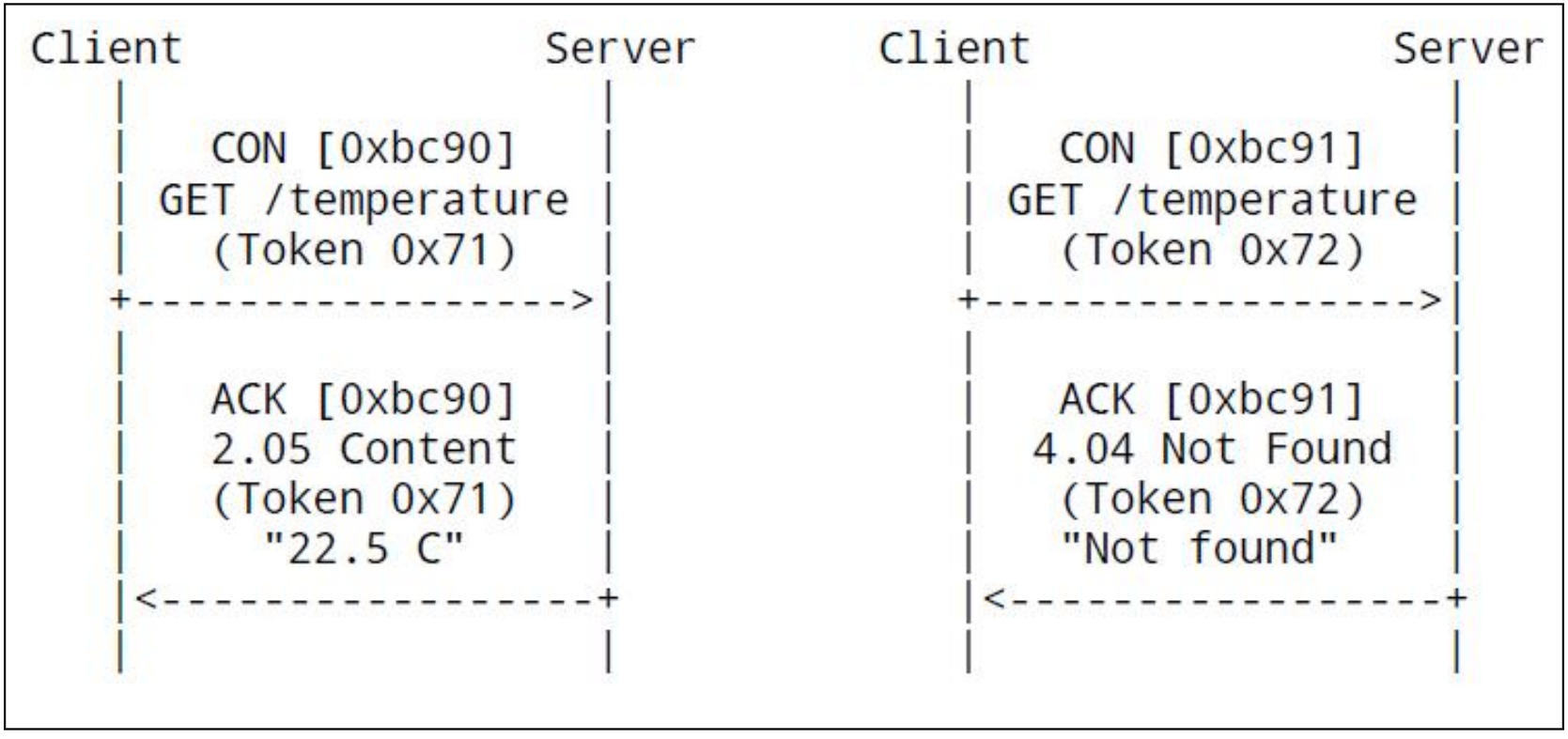

Figure : Request/Response Model

CoAP: Message Transmission [2]

CoAP requests and responses are transferred asychnronously wrapped in messages. Due to UDP, messages could be out of order, duplicate or get lost. Thus, it also introduces a reliable lightweight protocol like TCP.

- Stop-and-wait protocol
- Binary exponential back-off for Confirmable messages.
- Duplicate detection for both Confirmable and Non-confirmable messages.

CoAP: Message Transmission ... Contd. [2]

- Message Transmission is asynchronous between the endpoints.
- A CoAP endpoint is a source or destination of a message.
- Without security endpoints are identified by IP and Port number.
- With security its the security mode: NoSec, PreSharedKey, RawPublicKey and Certificate.

CoAP: Message Transmission ... Contd. [2]

- For reliable message transmission, it should be marked as Confirmable in the CoAP header.
- Confirmable message is transmitted at exponentially increasing intervals, until an acknowledgement (or Reset message) is recieved, or attempts get over.
- Therefore, timer and counter are required to handle retransmissions.

CoAP: Message Transmission ... Contd. [2]

- For unreliable message transmission, acknowledgement is not required for example repeated readings from sensor.
- A message is marked Non-confirmable in this case.
- Reciept of such message cannot be tracked.

CoAP: Message Transmission ... Contd. [2]

- Message deduplication is required for multiple messages recieved.
- In case of reliable messages, each duplicate copy is to be acknowledged by the reciever, but request can be processed only once.
- In case of unreliable messages, each duplicate copy is to be silently ignored by the reciever.

CoAP: Transmission Parameters [2]

- ACK TIMEOUT - 2 seconds
- ACK RANDOM FACTOR - 1.5
- MAX RETRANSMIT - 4
- NSTART - 1
- DEFAULT LEISURE - 5 seconds
- PROBING RATE - 1 Byte per second

CoAP: Method Definitions [2]

- GET: Retrieves the information corresponding to the resource in request URI. It is safe and idempotent.
- POST: Requests processing of representation in the response. Neither safe nor idempotent.
- PUT: Resource identified by the request URI be updated or created with the enclosed representation. Not safe but idempotent.
- DELETE: Requests to delete the resource identified by URI. Not safe but idempotent.

CoAP: Response Codes [2]

- Success 2.xx: 2.01 Created, 2.02 Deleted, 2.03 Valid, 2.04 Changed and 2.05 Content.
- Client Error 4.xx: 4.00 Bad Request, 4.01 Unauthorized, 4.02 Bad Option, 4.03 Forbidden,4.04 Not Found, 4.05 Method Not Allowed, 4.06 Not Acceptable, 4.12 Precondition Failed, 4.13 Request Entity Too Large and 4.15 Unsupported Content-Format.
- Server Error 5.xx: 5.00 Internal Server Error, 5.01 Not Implemented, 5.02 Bad Gateway, 5.03 Service Unavailable, 5.04 Gateway Timeout and 5.05 Proxying Not Supported.

CoAP: Discovery [2]

- Service Discovery: Client discovers its server from learning the URI. This can be a multicast operation as well.
- Resource Discovery: Discovery of resources by CoAP endpoints.

CoAP: Security [2]

DTLS: Datagram Transport Layer Security

- NoSec: DTLS is disabled.
- PreSharedKey: DTLS is enabled and pre-shared keys for security.
- RawPublicKey: DTLS is enabled and a raw public key is used for security.
- Certificate: DTLS is enabled and the device has an asymmetric key pair with an X.509 certificate.

CoAP: Applications [4]

- Smart Grid and Building Automations
- Legacy protocols like BACnet may be mapped as CoAP resources and respective communication data may be mapped to CoAP messages to have automated building applications.
- CoAP multicasts may be used for effective group communications like every sensor of a type in a room.

CoAP: Limitations [4]

- End to end secure connection required for CoAP/HTTP mapping at a proxy using DTLS/TLS.
- Securing multicast communications.
- Semantics should be standardized.
- Caching of requests should also be allowed.

A working example: WSN over CoAP

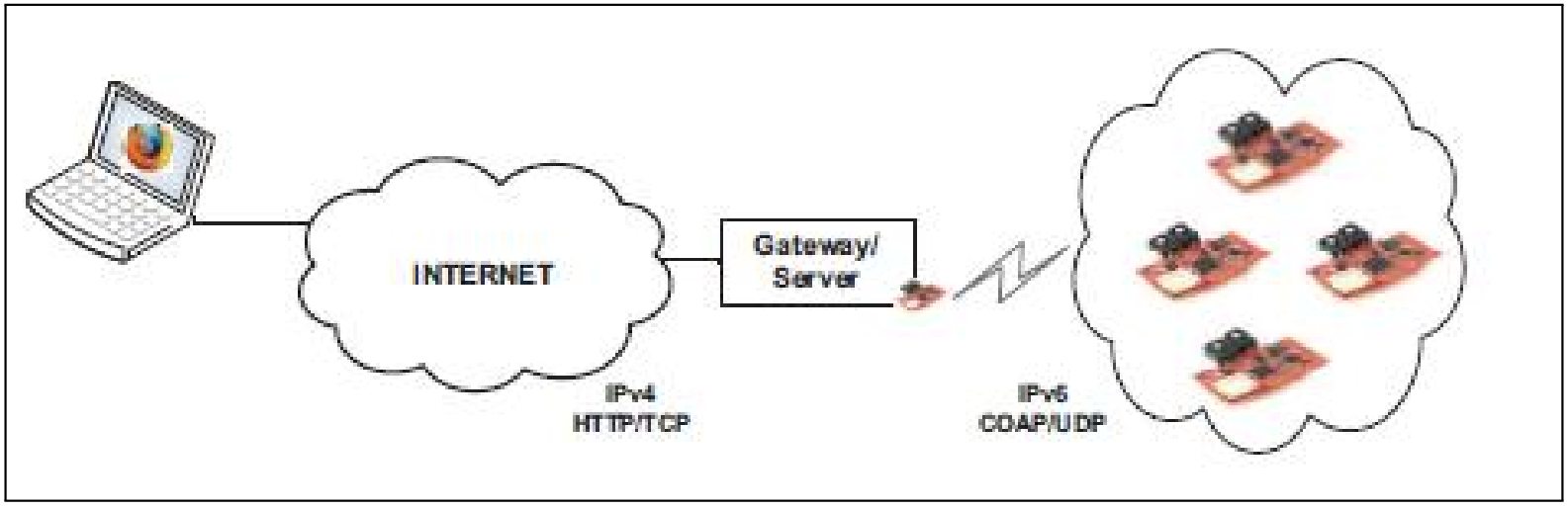

Figure : Integration of web and WSN, Image Courtesy [10]

A working example: WSN over CoAP

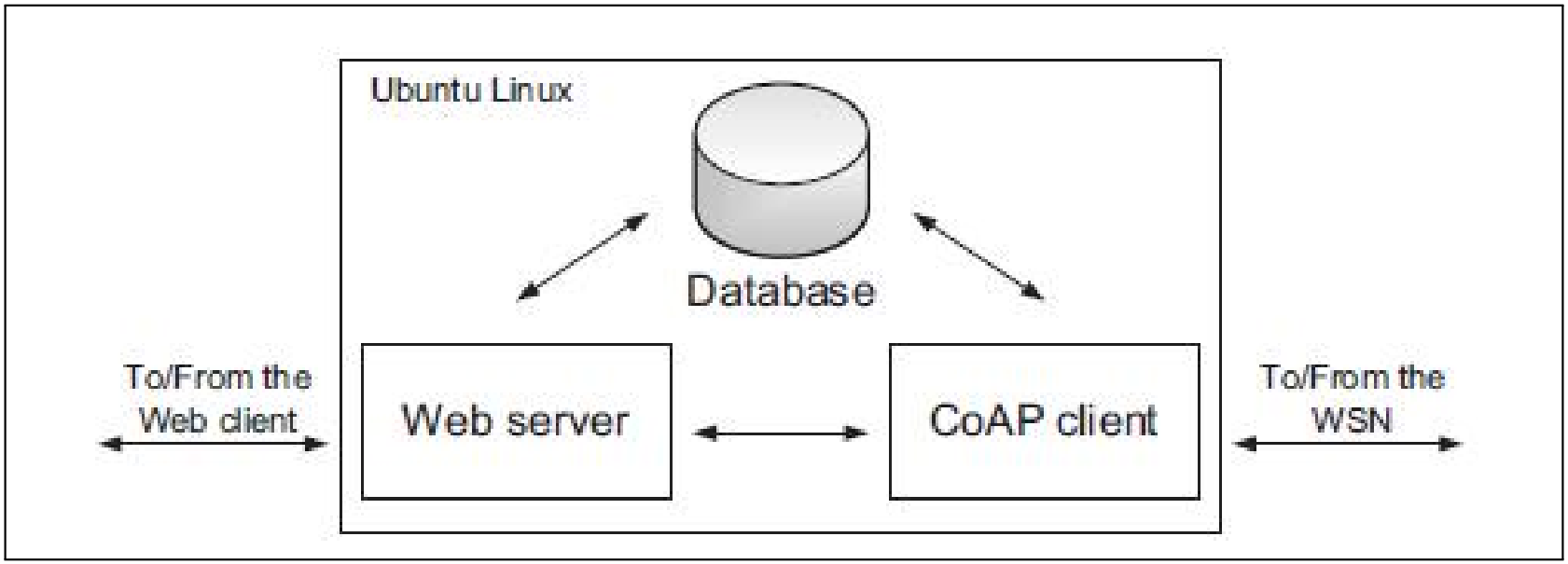

Figure : Gateway Building Blocks, Image Courtesy [10]

A working example: WSN over CoAP

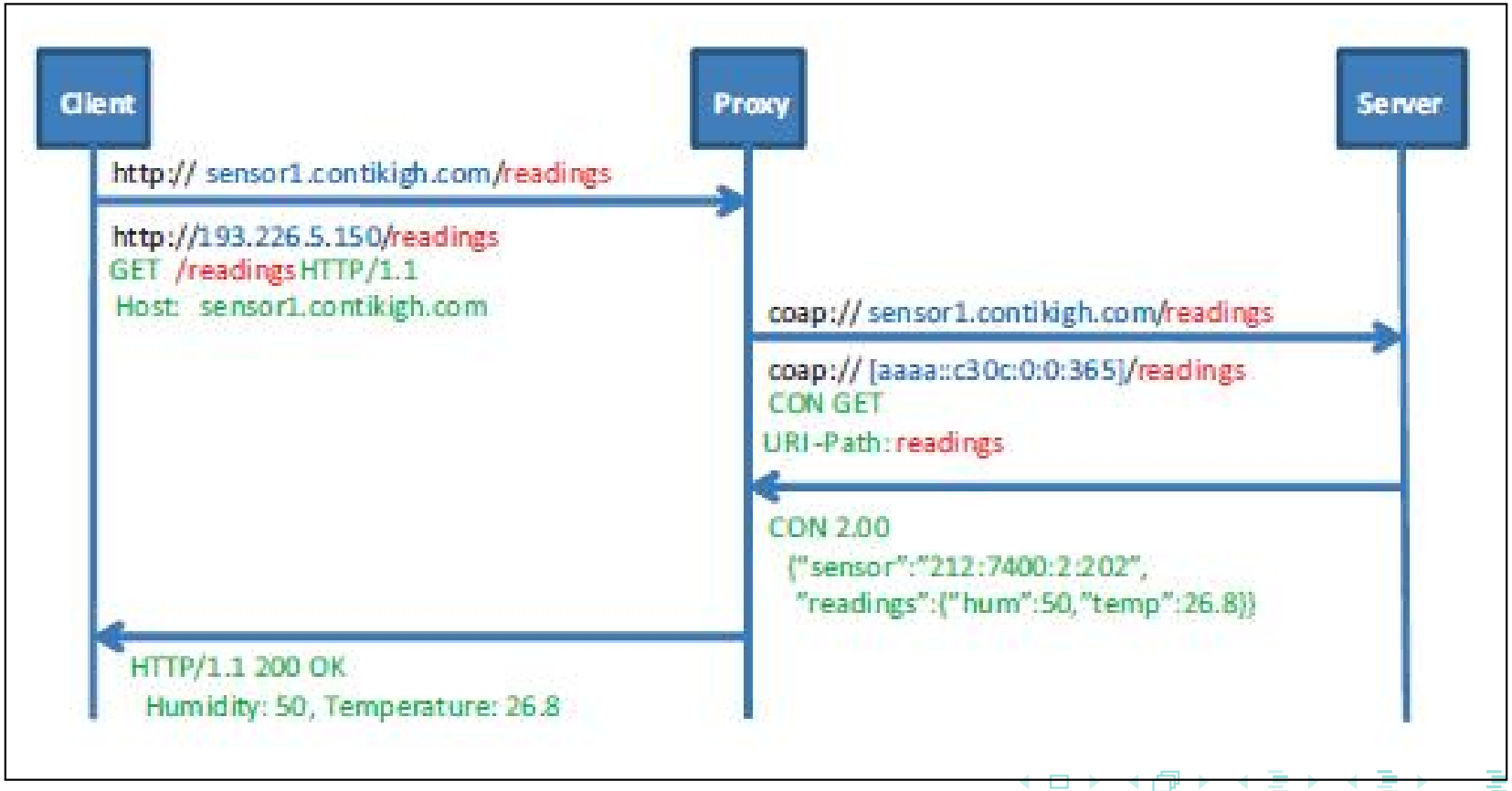

Figure : Message Exchange, Image Courtesy [10]

Frameworks for interconnection of devices

- Web of Things plug and play experience
- WebPlug
- Simple Measurement and Actuation Profile (sMAP)
- Californium (Cf)

What do users get with such interconnections

- Smart devices with user need specific application program.
- Heat/Cool office/home effectively.
- Intelligent room lighting as per user specifications.
- Health, patient monitoring.

Californium (Cf)

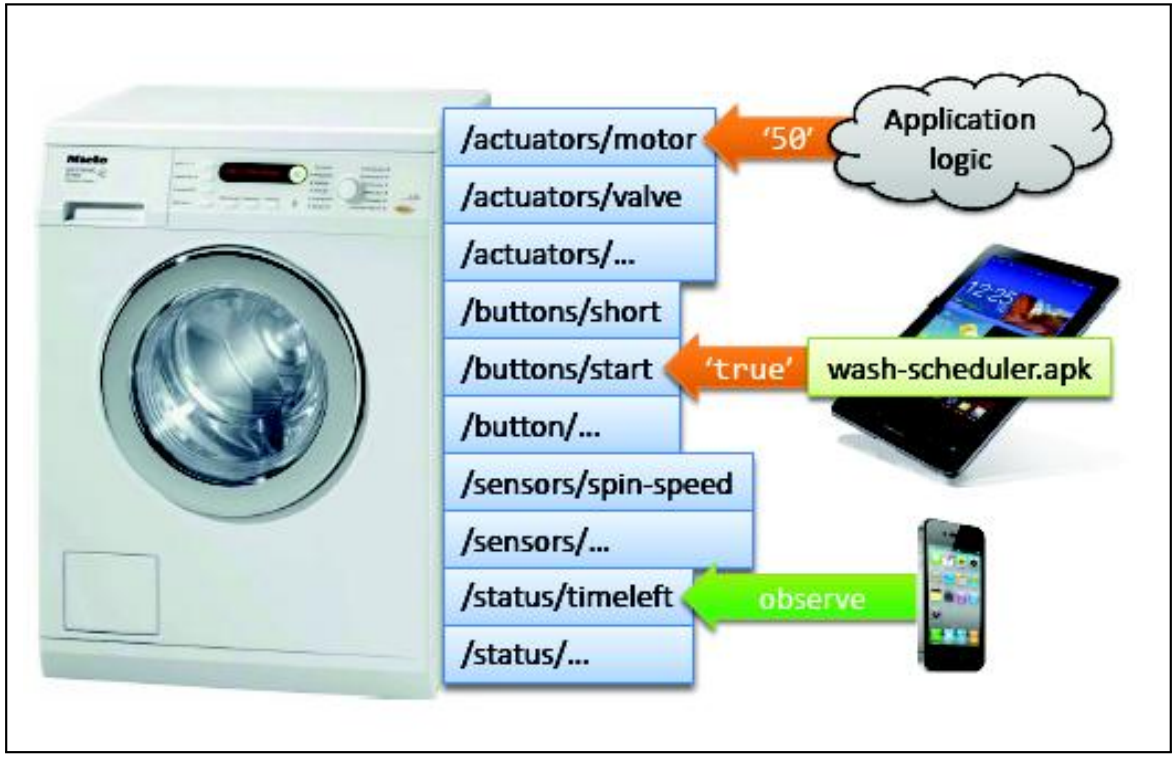

Figure : Thin architecture Californium, Image Courtesy [3]

Californium (Cf): The Architecture

- *Thin server architecture* - Do not host the application logic on the devices, rather act as a wrapper and support native API for access to device's sensors and actuators.
- *Separate Application Server* - Application logic is hosted on separate *Application Server* which is on the web, hence seperated from the devices.
- Provide an API to access the device's resources using CoAP/HTTP and are RESTful.
- Same devices can be used for different applications without changing the firmware.
- New devices are discovered and integrated in the infrastructure robustly.

Californium (Cf): Features

- Full Web Integration: The embedded devices can exploit the vast array of resources available on the internet.
- Intuitive APIs: API interfaces follow the RESTful architecture constraints. Thus, the integration costs are lowered and complex distributed systems co-ordinate efficiently.

Californium (Cf): Features ... Contd.

- Decoupling of Infrastructure and Applications: Upgrades of the application logic can be carried out without any firmware upgrades. So the capabilities of a huge number of nodes can be altered just by altering the logic on the web server hosting the application logic.
- End-User Programming: Developers with no or little embedded programming background can develop applications for embedded devices on the IoT. Separating the device action from the device, allows the users to customize the device behavior to their specific needs.

Implemention CoAP on Raspberry Pi

- Essentially a CoAP client & CoAP server is to be implemented.
- Install Linux on Raspberry Pi.
- Follow the instructions on
  `http://docs.tinyos.net/tinywiki/index.php/CoAP`
  to enable CoAP client and server.

Open research problems: Food for thought

- Where is the security ?
- What if internet connectivity is gone ?
- What are safety measures taken ?
- What about path loss and signal power needs to have efficient communications ?

Thank You

Thank You

References I

[1] Web, "Internet of things protocols," 2013, [Online; accessed 20-March-2013]. [Online]. Available: http://postscapes.com/internet-of-things-protocols

[2] Z. Shelby, Sensinode, K. Hartke, C. Bormann, and U. B. TZI, "Internet-draft - constrained application protocol (coap)," 2013, [Online; accessed 20-March-2013]. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-core-coap/?include_text=1

[3] M. Kovatsch, S. Mayer, and B. Ostermaier, "Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, 2012, pp. 751–756.

[4] B. Villaverde, D. Pesch, R. De Paz Alberola, S. Fedor, and M. Boubekeur, "Constrained application protocol for low power embedded networks: A survey," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, 2012, pp. 702–707.

[5] G. Goth, "Critics say web services need a rest," *Distributed Systems Online, IEEE*, vol. 5, no. 12, pp. 1–1, 2004.

[6] Web, "A brief introduction to rest," 2013, [Online; accessed 20-March-2013]. [Online]. Available: http://www.infoq.com/articles/rest-introduction

[7] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002. [Online]. Available: http://doi.acm.org/10.1145/514183.514185

References II

[8] Web, "Fielding dissertation: Chapter 5: Representational state transfer (rest)," 2013, [Online; accessed 20-March-2013]. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[9] M. Laine, "Restful web services for the internet of things."

[10] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, "Rest enabled wireless sensor networks for seamless integration with web applications," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, 2011, pp. 867–872.