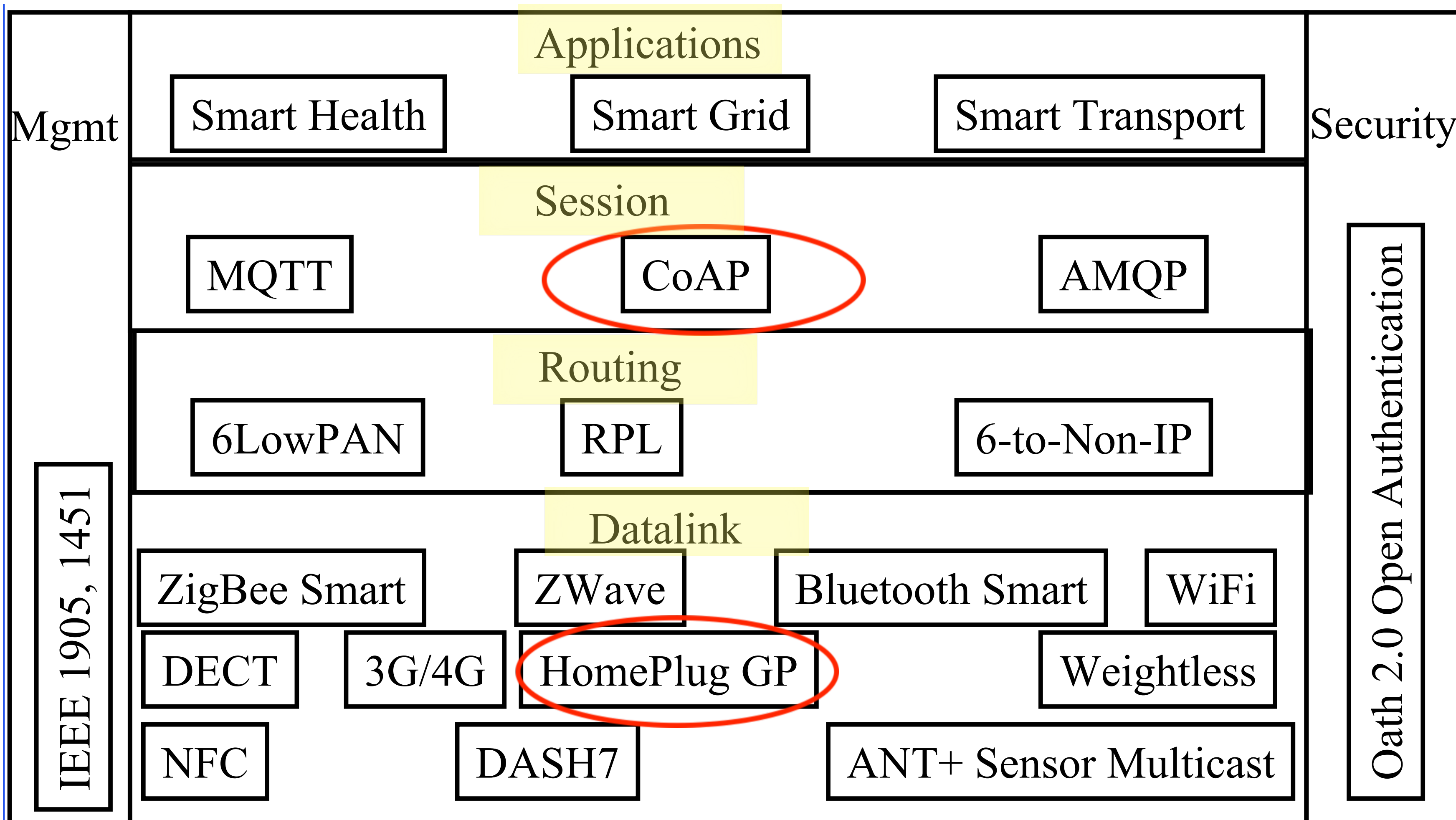
IoT: HP and CoAP

Gaia Maselli
Dept. of Computer Science

Recent protocols for IoT

SAPIENZA
UNIVERSITÀ DI ROMA



HomePlug: standard for PowerLine communication

Power Line Communication (PLC)

- The existing electrical wiring and outlets are used as the medium for data communication within the home
- Using power lines as the network infrastructure has many advantages over other technologies.
  - no new wires are needed
  - there are many access points (power sockets) in a home/ building (four or more per room)
  - the cost to build a power line network is low compared to that of other technologies
- *power line communication* (PLC) as specified by the *HomePlug* 1.0 standard provides a 14 Mb/s raw data rate

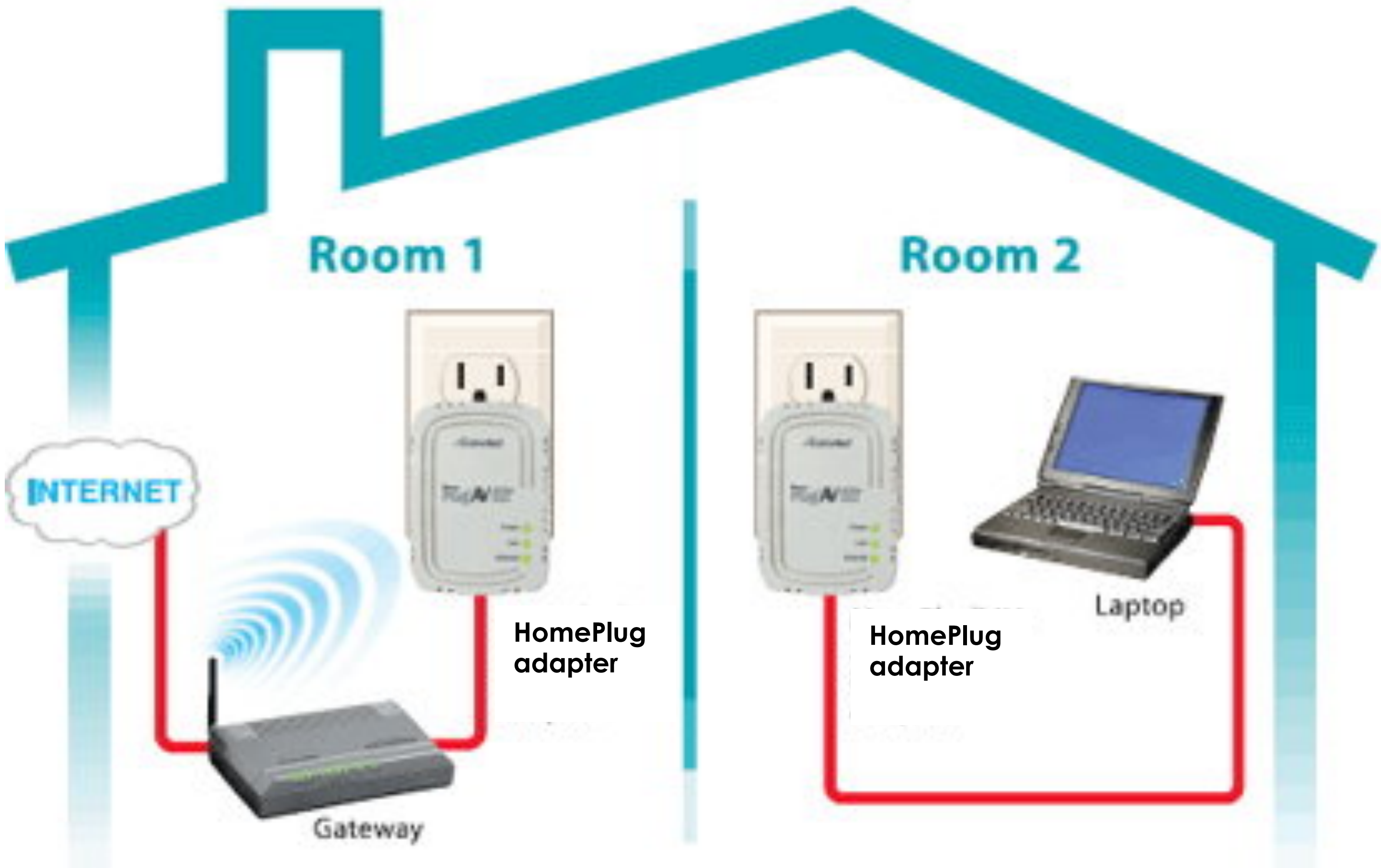
PLC in the past

- In the past, power lines were considered unacceptable for signal transmission, since the channel is subject to a lot of noise, interference, and fading.
- Power lines were not designed for delivering high-frequency signals
- The poor quality of a power line is not ideal for signal transmission because the channel contains noise and interference.
- The appeal of using the existing power line as a transmission medium for data exchange was too great to be ignored.
- The advancement of signal modulation technologies (Orthogonal Frequency Division Multiplexing - **OFDM**), digital signal processing, and error control coding **has minimized the restrictions of channel imperfections**, and high-speed signal transmission through power lines is now feasible.

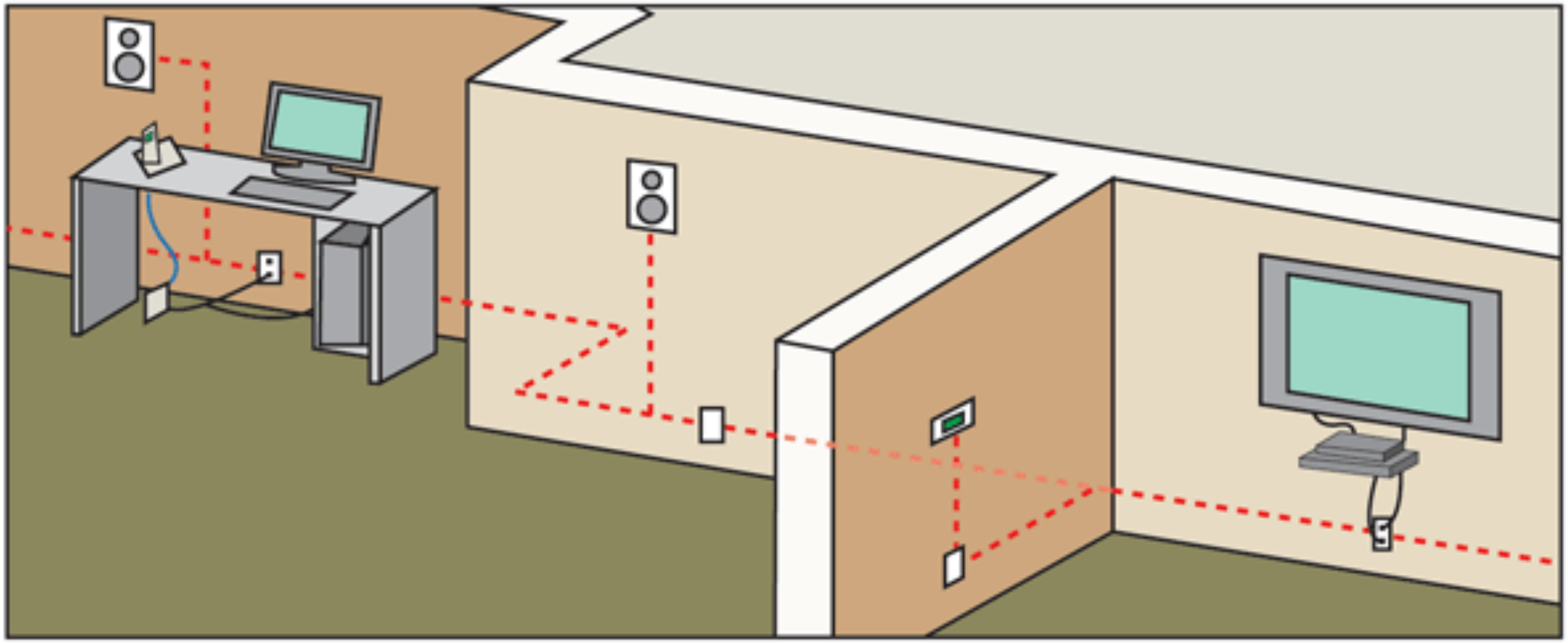
HomePlug

- HomePlug 1.0
- HomePlug AV: Audio/Video - 20- 200 Mbps
- HomePlug AV2: Gigabit networking
- HomePlug GP: Green PHY greater distance coverage and lower power consumption than HomePlug AV

Example

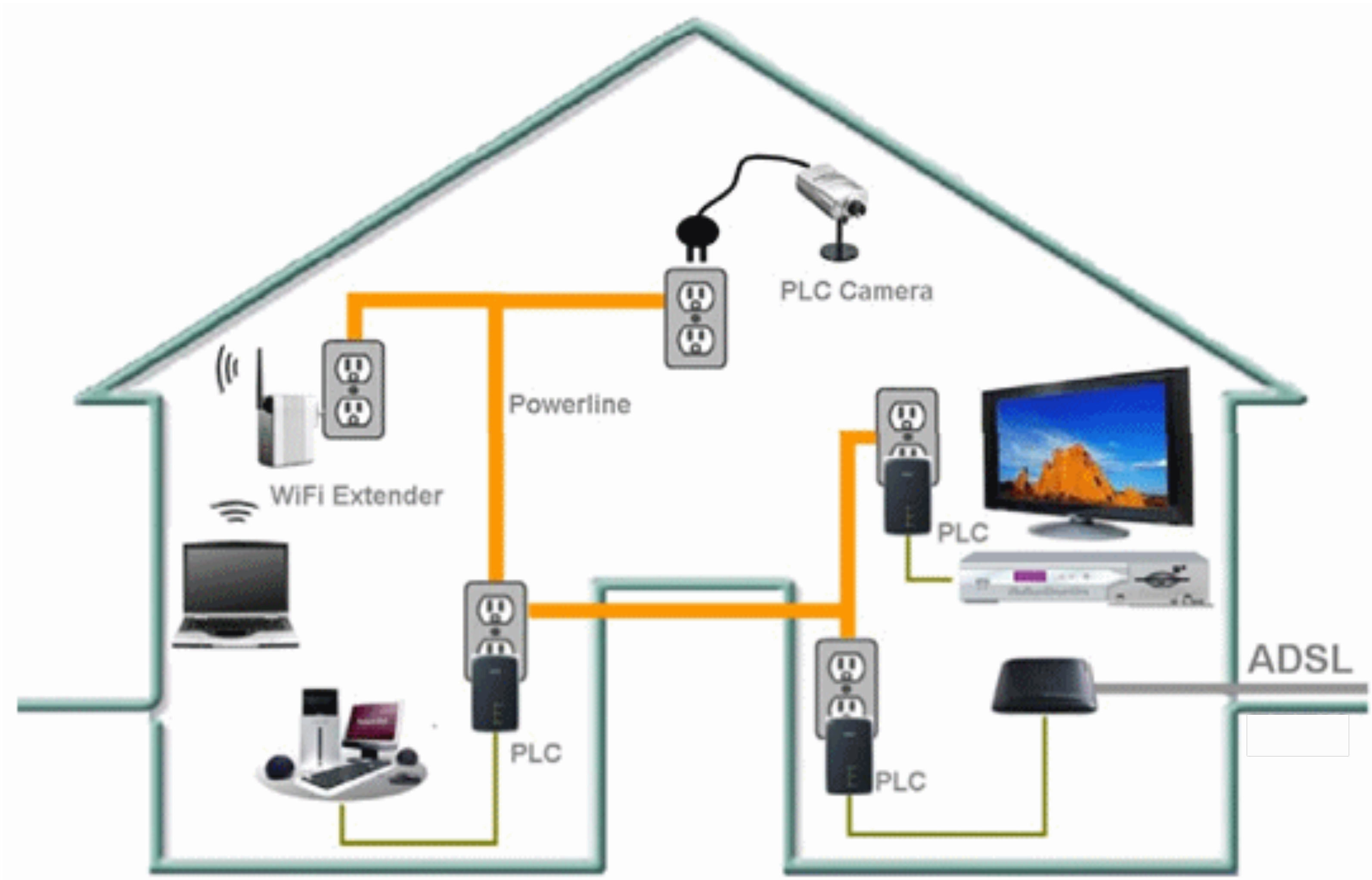

Room 1
Room 2
INTERNET
HomePlug adapter
HomePlug adapter
Laptop
Gateway

HomePlug AV



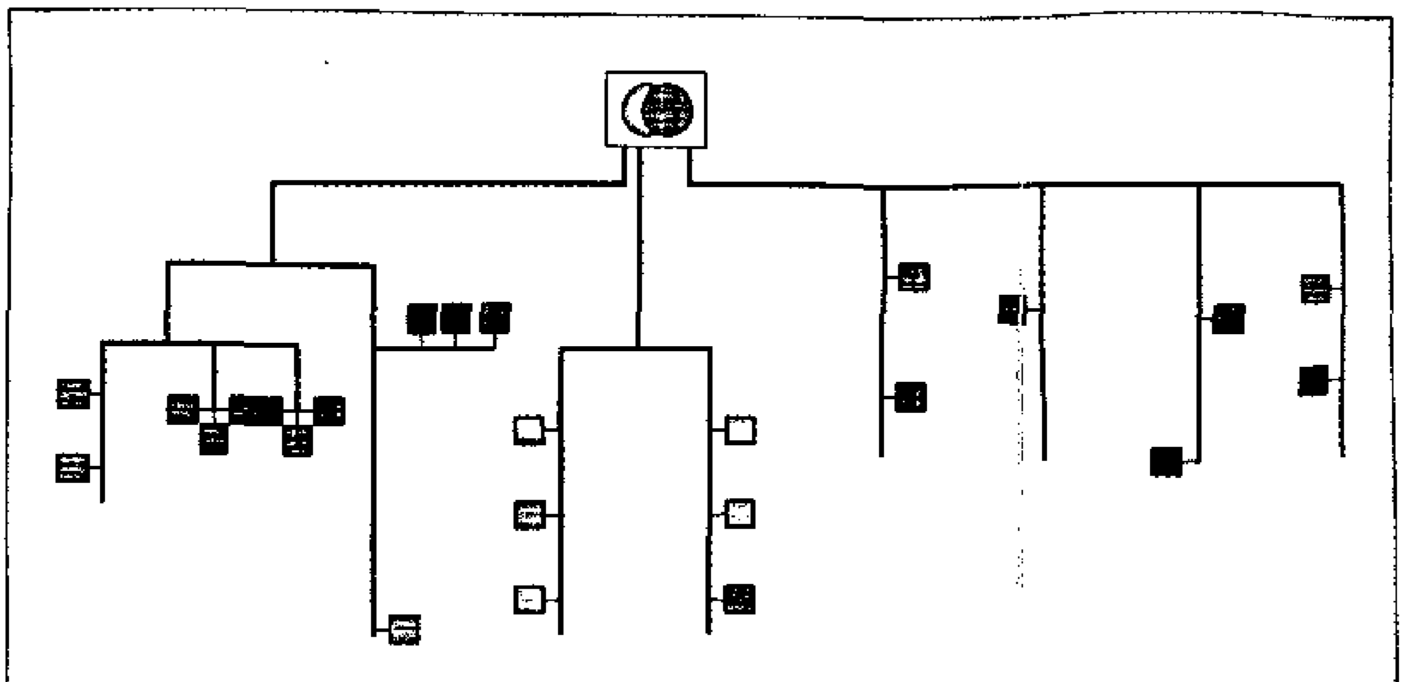
- With HomePlug technology the electrical wires in a home can distribute broadband Internet, HD video, digital music, and smart energy applications

Fully connected home



MAC layer

- Power grid networks usually form a bus or tree topology
- Communication between any pair of terminals is possible
- However, most traffic is expected to be from and to a terminal serving as the network gateway



MAC layer issues

- CSMA/CD cannot be applied on power line networks as the wide variation of the received signal and noise levels makes collision detection difficult and unreliable.
- An alternative to collision detection that can he easily employed in cases of PLC is collision avoidance (**CSMA/CA**), a technique that uses random back-offs to further reduce the collision probability
- Time-division multiple access (**TDMA**) can also be used

HPAV: MAC protocols

- Connection-oriented **Contention Free** (CF) service
  - to support the QoS requirements (guaranteed bandwidth, latency and jitter requirements) of demanding AV and IP applications.
  - This Contention Free service is based on periodic **Time Division Multiple Access (TDMA)** allocations of adequate duration to support the QoS requirements of a connection.
- Connectionless **prioritized Contention** based service
  - to support both best-effort applications and applications that rely on prioritized QoS.
  - This service is based on **Collision Sense Multiple Access/Collision Avoidance (CSMA/CA)** technology which is applied to only traffic at the highest pending priority level after the pending traffic with lower priority levels has been eliminated during a brief Priority Resolution phase at the beginning of the contention window.

TDMA + CSMA/CA: coordination

- To efficiently provide both kinds of communication service, HPAV implements a **flexible, centrally-managed architecture**.
- The central manager is called a **Central Coordinator (CCo)**. The CCo establishes a **Beacon Period** and a schedule which accommodates both the Contention Free allocations and the time allotted for Contention-based traffic.
- the Beacon Period is divided into 3 regions:

1. Beacon Region
2. CSMA Region
3. Contention-Free Region

- Messaging in HPAV is direct from station to station; however, the CCo monitors the messages. The header of each message contains information about how much data is pending for transmission on the connection

Reading material

- HomePlug Alliance, “HomePlug AV White Paper”, http://www.homeplug.org/media/filer_public/b8/68/b86828d9-7e8a-486f-aa82-179e6e95cab5/hpav-white-paper_050818.pdf
- “A Power Line Communication Network Infrastructure For The Smart Home”, *IEEE Wireless Communications*, December 2002

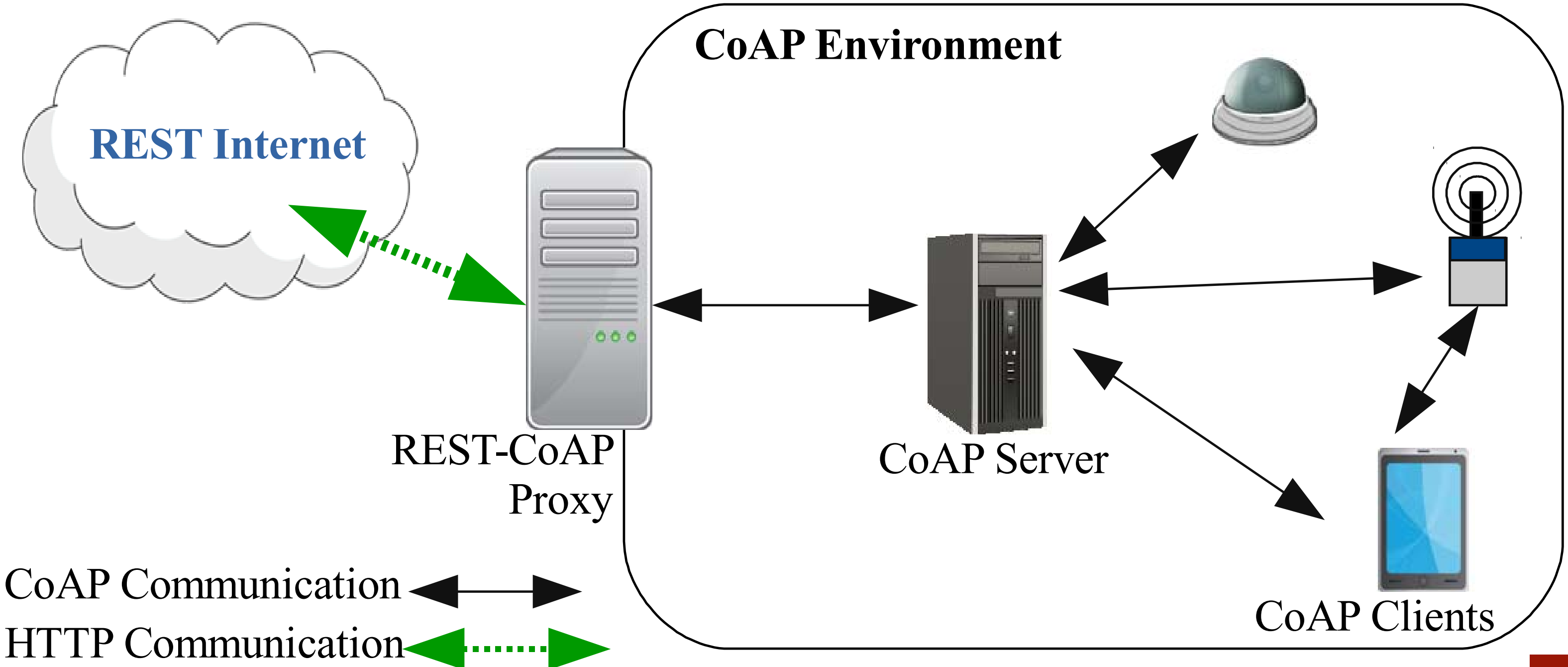
CoAP:
Constrained Application Protocol

Constrained Application Protocol (CoAP)

- The IETF Constrained RESTful Environments (CoRE) working group created CoAP, which is an application layer protocol for IoT applications
- Specialized **web transfer protocol** for use with **constrained nodes** and constrained (e.g., low-power, lossy) **networks**.
- The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.
- Provides **a request/response interaction** model between application endpoints
- Supports built-in discovery of services and resources
- Includes key concepts of the Web such as URIs and Internet media types.
- CoAP is designed to **easily interface with HTTP** for integration with the Web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments.

CoAP functionality

- CoAP modifies some HTTP functionalities to meet the IoT requirements such as low power consumption and operation in the presence of lossy and noisy links
- Conversion between these two protocols in REST-CoAP proxies



CoAP operation

- The interaction model of CoAP is similar to the **client/server** model of HTTP
- machine-to-machine (M2M) interactions typically result in a CoAP implementation acting in both client and server roles
- CoAP request is equivalent to that of HTTP and is sent by a client to **request an action** (using a Method Code) on a resource (identified by a URI) on a server.
- The server then sends a response with a Response Code; this response may include a resource representation.
- Unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport such as UDP.

Abstract Layering of CoAP

- CoAP logically uses a two-layer approach
- CoAP messaging layer is used to deal with UDP and the asynchronous nature of the interactions
- the request/response interactions using Method and Response Codes
- CoAP is however a single protocol, with messaging and request/response as just features of the CoAP header.

```
+----------------------+
|      Application     |
+----------------------+
+----------------------+  \
|  Requests/Responses  |  |
|----------------------|  | CoAP
|       Messages       |  |
+----------------------+  /
+----------------------+
|          UDP         |
+----------------------+
```

CoAP messages

- CoAP defines four types of messages
  1. **Confirmable**
  2. **Non-confirmable**
  3. **Acknowledgement**
  4. **Reset**
- **Method Codes** and **Response Codes** included in some of these messages make them carry requests or responses.
- Requests can be carried in Confirmable and Non-confirmable messages
- Responses can be carried in Confirmable and Non-confirmable messages as well as piggybacked in Acknowledgement messages
- Each message contains a Message ID used to detect duplicates and for optional reliability

Messaging model: reliable

Reliable Message Transmission

- Reliability is provided by marking a message as Confirmable (CON).
- A Confirmable message is **retransmitted** using a default **timeout and exponential back-off** between retransmissions, until the recipient sends an Acknowledgement message (ACK) with the same Message ID (in this example, 0x7d34) from the corresponding endpoint

```
Client                 Server
   |                     |
   |   CON [0x7d34]      |
   +-------------------->|
   |                     |
   |   ACK [0x7d34]      |
   |<--------------------+
   |                     |
```

Messaging model: unreliable

Unreliable Message Transmission

- A message that does not require reliable transmission (for example, each single measurement out of a stream of sensor data) can be sent as a Non-confirmable message (NON).
- These are not acknowledged, but still have a Message ID for duplicate detection (in this example, 0x01a0)

```
Client              Server
   |                   |
   |   NON [0x01a0]    |
   +------------------>|
   |                   |
```

Messaging model: reset

- When a recipient **is not at all able to process** a Confirmable or a Non Confirmable message (i.e., not even able to provide a suitable error response), it replies with a **Reset message (RST)**

CoAP request/response model

- CoAP request and response semantics are carried in CoAP messages, which include either a Method Code or Response Code, respectively

Immediate response to a CON request

- A request can be carried in a Confirmable (CON) message, and, if immediately available, the response to the request can be carried in the resulting Acknowledgement (ACK) message.

```
Client              Server       Client              Server
   |                  |             |                  |
   |   CON [0xbc90]   |             |   CON [0xbc91]   |
   | GET /temperature |             | GET /temperature |
   |   (Token 0x71)   |             |   (Token 0x72)   |
   +----------------->|             +----------------->|
   |                  |             |                  |
   |   ACK [0xbc90]   |             |   ACK [0xbc91]   |
   |   2.05 Content   |             |  4.04 Not Found  |
   |   (Token 0x71)   |             |   (Token 0x72)   |
   |     "22.5 C"     |             |   "Not found"    |
   |<-----------------+             |<-----------------+
   |                  |             |                  |
```

Two GET Requests with Piggybacked Responses

CON message with separate response

- If the server is not able to respond immediately to a request carried in a Confirmable message, it simply responds with an Empty Acknowledgement message so that the client can stop retransmitting the request.
- When the response is ready, the server sends it in a new Confirmable message (which then in turn needs to be acknowledged by the client).

```
Client                Server
   |                     |
   |   CON [0x7a10]      |
   | GET /temperature    |
   |   (Token 0x73)      |
   +-------------------->|
   |                     |
   |   ACK [0x7a10]      |
   |<--------------------+
   |                     |
```

A GET Request with a Separate Response

Non-confirmable request/ response

- If a request is sent in a Non-confirmable message, then the response is sent using a new Non-confirmable message, although the server may instead send a Confirmable message.

```
Client              Server
   |                  |
   |  NON [0x7a11]    |
   | GET /temperature |
   |  (Token 0x74)    |
   +----------------->|
   |                  |
   |  NON [0x23bc]    |
   |  2.05 Content    |
   |  (Token 0x74)    |
   |    "22.5 C"      |
   |<-----------------+
   |                  |
```

A Request and a Response Carried in Non-confirmable Messages

CoAP methods

- CoAP, as in HTTP, utilizes methods such as **GET**, **PUT**, **POST** and **DELETE** to achieve Create, Retrieve, Update and Delete (CRUD) operations.
- Example: the GET method can be used by a server to inquire the client's temperature using the piggybacked response mode. The client sends back the temperature if it exists; otherwise, it replies with a status code to indicate that the requested data is not found.

CoAP messages

- CoAP uses a simple and small format to encode messages.
- 4 bytes of header + token (0-8 bytes) + options + payload
- The token value is used for correlating requests and responses. The options and payload are the next optional fields.
- A typical CoAP message can be between 10 to 20 bytes

0 1	2 3	4 5 6 7	8	16 31
Ver	T	OC	Code	Message ID
Token (if any)				
Options (if any)				
Payload (if any)				

Message header

- Version (Ver): 2-bit unsigned integer - Indicates the CoAP version number.
- Type (T): 2-bit unsigned integer. Indicates if this message is of type Confirmable (0), Non-confirmable (1), Acknowledgement (2), or Reset (3).
- OC: 4-bit unsigned integer. Indicates the length of the variable-length Token field (0-8 bytes).
- Code: 8-bit unsigned integer, Code represents the request method (1–10) or response code (40–255). For example the code for GET, POST, PUT, and DELETE is 1, 2, 3, and 4, respectively

0 1	2 3	4 5 6 7	8	16 31
Ver	T	OC	Code	Message ID

Reading material

- RFC 7252: The Constrained Application Protocol (CoAP)