Californium

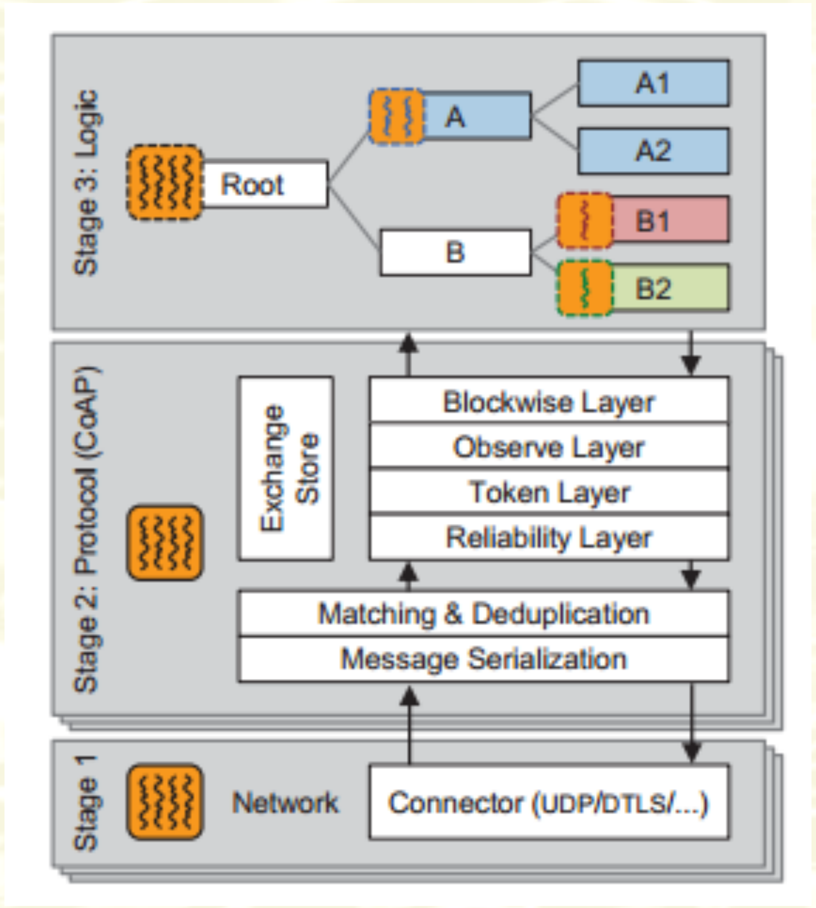
Giacomo Tanganelli
PhD student @ University of Pisa
g.tanganelli@iet.unipi.it

Californium

- Californium is a Java CoAP library.
- Download with:
  - git clone https://github.com/eclipse/californium.core.git
  - Compile using Maven:
  - mvn install

  or
  - Eclipse:
    - Download last eclipse

Architecture overview



- Network:
  - Receiving and sending byte arrays over the network. UDP or DTLS for CoAP
- Protocol:
  - Executes the CoAP protocol with a thread pool. Exchange object holds all data needed by a request/ response. The exchange is passed layer by layer.
- Logic:
  - Host resources structured in a logical tree.

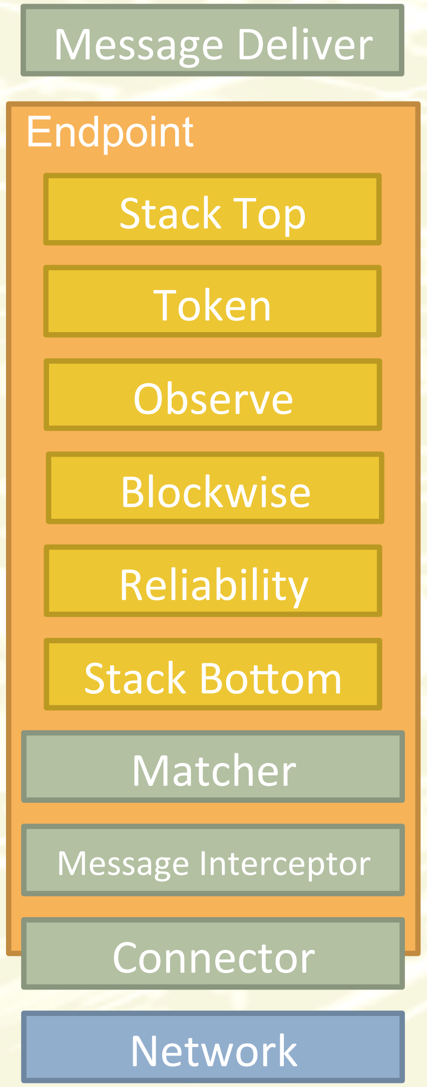
Package coap - Classes

- Message: core entities for the message exchange between CoAP endpoints. All communication layers process objects of this class.
  - Users will instantiate request and response subclasses rather than directly create a new message object.
- Option: A message can have several Options with different or same option numbers. Every option is associated with a value of implicit type.
- Request: provides the functionality of a CoAP request as a subclass of Message.
- Response: provides the functionality of a CoAP response as a subclass of Message.

Package network

- An Endpoint is bound to a particular IP address and port.
- Clients use an Endpoint to send a request to a server. Servers bind resources to one or more Endpoints.
- Generalization of client and server.

Package network – CoAPEndpoint

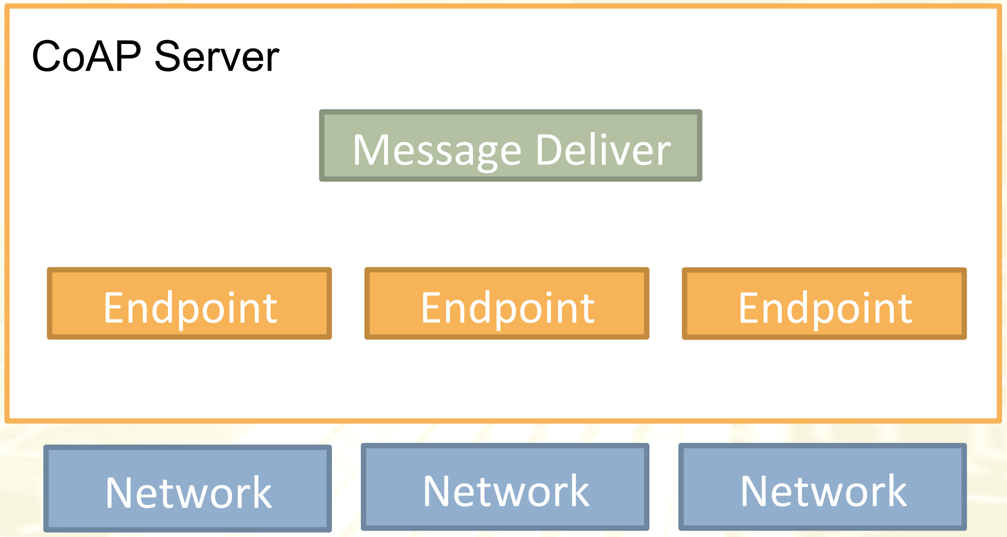


- Endpoint encapsulates the stack that executes the CoAP protocol and forwards incoming messages to a MessageDeliverer.
- The deliverer will deliver requests to its destination resource.
- The endpoint sends outgoing messages over a connector.
- The connector encapsulates the transport protocol.

Package server

- A server hosts a tree Resources exposed to clients.
- Resources can be added and removed dynamically.
- Resource is an element on the resource tree of a server.
- A resource must have a unique URI.
- A resource is able to respond to CoAP requests.

```
CoapServer server = new CoapServer(port);
server.add(new CoapResourceExample(“hello-world”);
server.start();
```



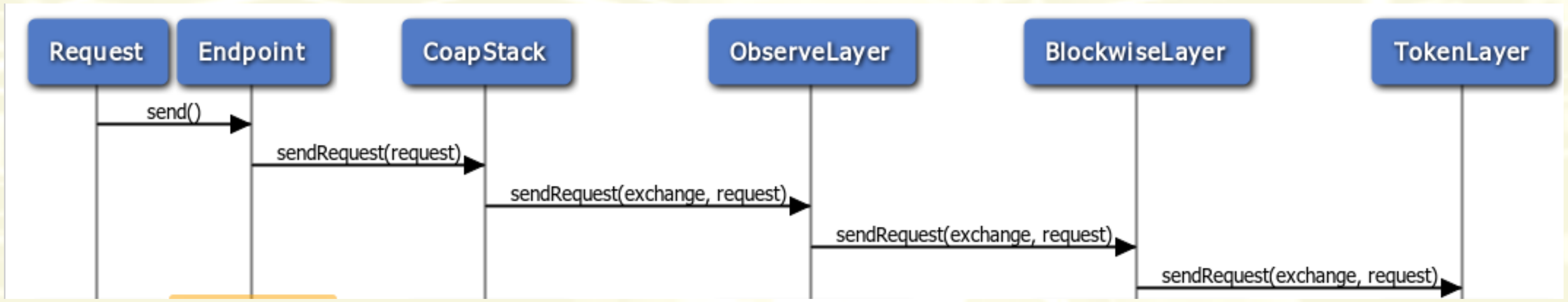
CoAPResource

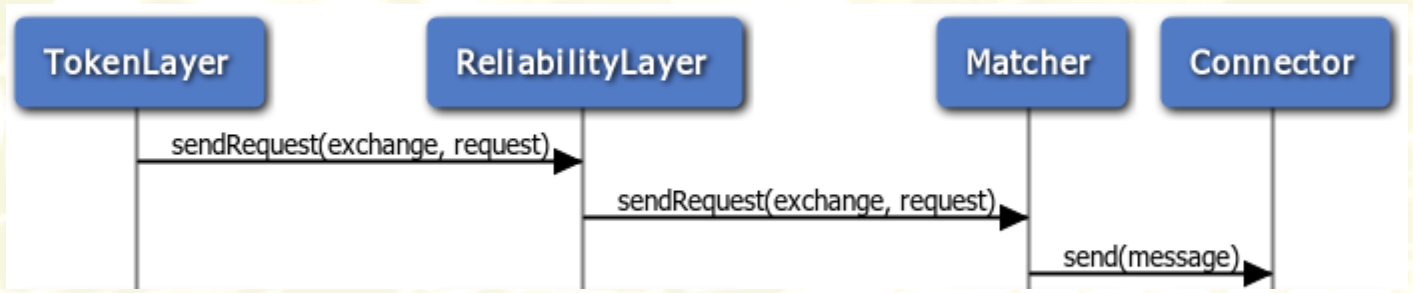
```
public class CoAPResourceExample extends CoapResource {
		public CoAPResourceExample(String name) {
				super(name);
		}
		public void handleGET(CoapExchange exchange) {
				exchange.respond("hello world");
		}
		public void handlePOST(CoapExchange exchange) {
				exchange.accept();
				/* your stuff */
				exchange.respond(ResponseCode.CREATED);
		}
….
}
```

Package stack

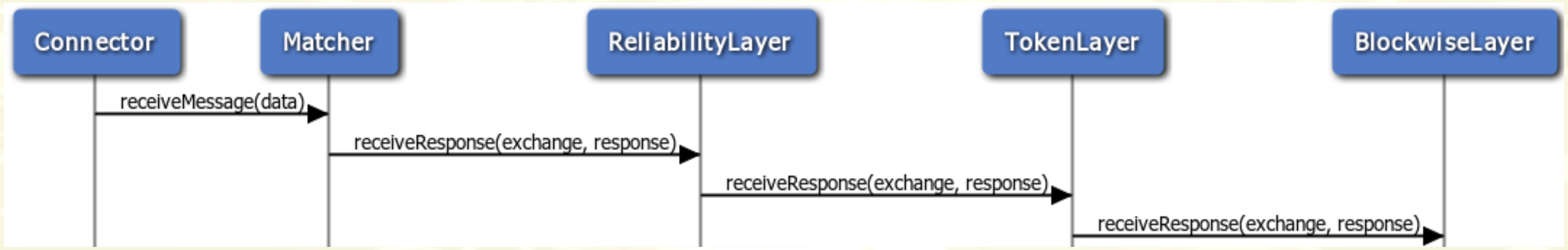
- A layer processes requests, responses and empty messages.
- Receive a message:
  - Endpoint forwards it to the bottom layer by calling the corresponding receive-method. Each layer processes the message and either forwards it to its upper layer or decides not to.
- Send a message:
  - Endpoint forwards it to the uppermost layer by calling the corresponding send-method. Each layer forwards the message to its lower layer.

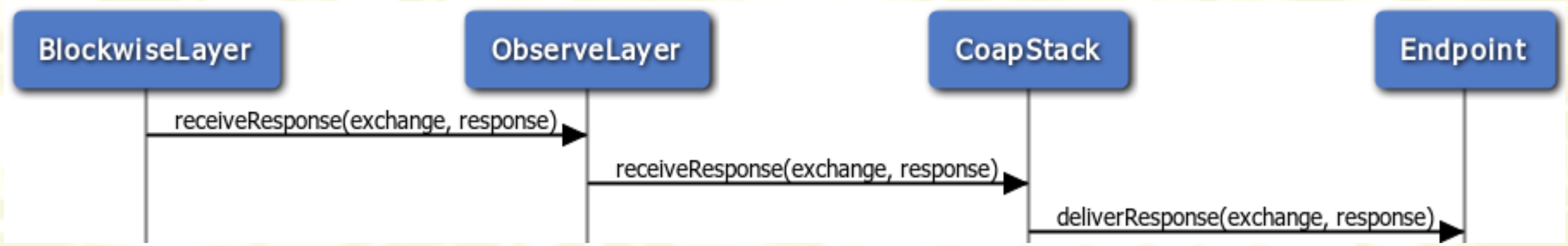
Send


Request
Endpoint
CoapStack
ObserveLayer
BlockwiseLayer
TokenLayer
send()
sendRequest(request)
sendRequest(exchange, request)
sendRequest(exchange, request)
sendRequest(exchange, request)


TokenLayer
ReliabilityLayer
Matcher
Connector
sendRequest(exchange, request)
sendRequest(exchange, request)
send(message)

Receive


Connector
Matcher
ReliabilityLayer
TokenLayer
BlockwiseLayer
receiveMessage(data)
receiveResponse(exchange, response)
receiveResponse(exchange, response)
receiveResponse(exchange, response)


BlockwiseLayer
ObserveLayer
CoapStack
Endpoint
receiveResponse(exchange, response)
receiveResponse(exchange, response)
deliverResponse(exchange, response)

Exercise 1

- Write a CoAP client which interacts with a CoAP server. The server must have a resource which allows GET and POST.
- Modify the previous server in order to send separate responses.
  - exchange.accept();
- Create a resource on the server that read a number from a query parameter and reply back with the square of the input number.

Exercise 2

- Write a client which interacts with a server deployed in cooja with contiki.
- The server must have a resource with GET and POST method allowed.
- The client must send request to get the value and to set the value.

Proxy

- Californium provides proxies:
  - Http2Coap
  - Coap2Coap
- Coap2Coap implements the forward proxy as described in the CoAP RFC.

Why use proxy?

- Sleeping devices
- Cache
- Observing
- Unreachable sensors

Forward v. Reverse

- Forward proxy:
  - A client is aware of the presence of a forward proxy
  - Client must add a proxy-uri option in the message send to the forward proxy with the path to the real CoAP server.
- Reverse proxy:
  - A client does not know about the reverse proxy.
  - Client sends normal messages, the reverse proxy forwards requests to the real CoAP server.
  - Reverse proxy must change namespaces.

Exercise 3

- Write a forward proxy which acts as an interface between the previous CoAP client and the server running in cooja.