ERICSSON

A LIGHTWEIGHT AND SECURE BOOTSTRAPPING MECHANISM FOR THE INTERNET OF THINGS

MASTER'S THESIS WORK

Adriano, Jaime

THESIS OVERVIEW I

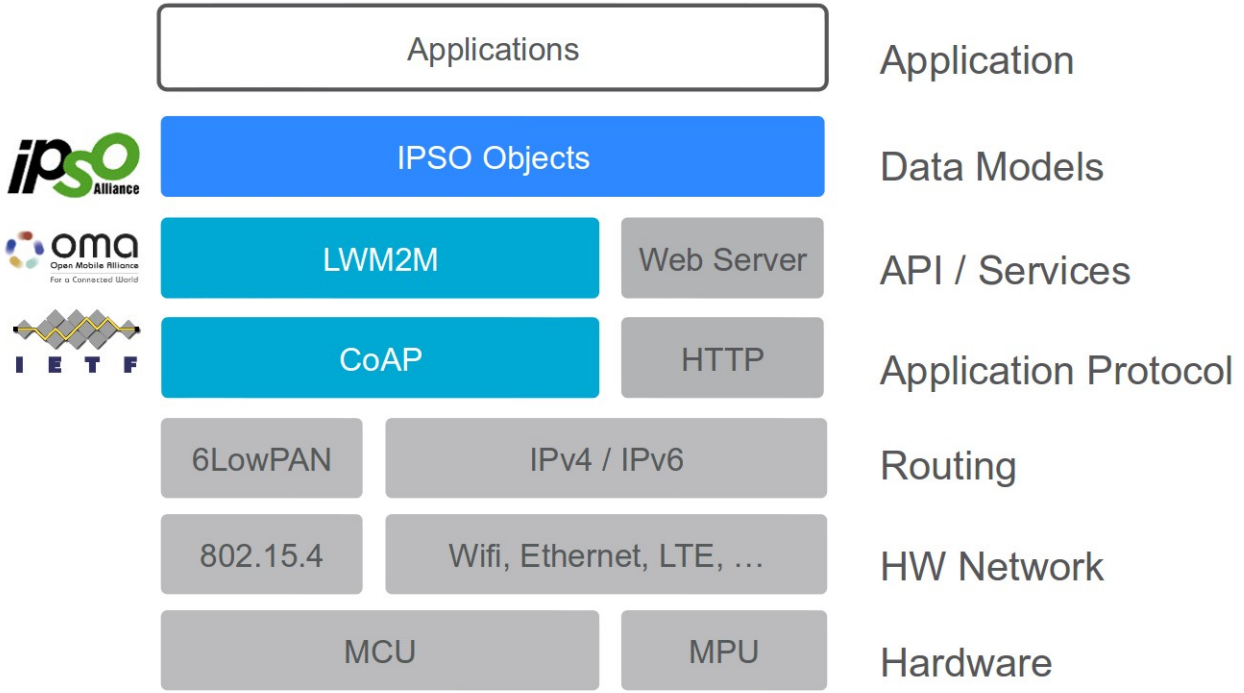
- Background
  - **Internet of Things** to connect to the Internet any real world object to exchange data between them.
  - **Machine-to-Machine** to automate the process of gathering information from the environment and act accordingly to the values read, e.g. from a sensor network.
- Problem
  - **Device Bootstrapping** to provision a new deployed device (or gateway) with initial information for connectivity and to enable a specific service.
  - **Security** to protect the information flowing over the network, adding Confidentiality, Integrity and Authentication.
  - **Semantic Interoperability** to allow different implementations to share a common interface and communicate one with another.

THESIS OVERVIEW II

- Objective
  - Enable device bootstrapping for IoT in an automatic way.
  - Encipher the information exchanged during the bootstrap and the device management phase.
  - Provide a user interface for device management (Mert's Thesis).
- Requirements
  - **Automatic and selective bootstrap**: not every device can be bootstrapped.
  - **Semantic interoperability**: sharing a common data model between devices so that every services and applications can understand the content.
  - **Protection against network attacks**: the information should not be disclosed to an unauthorized third party of the communication.

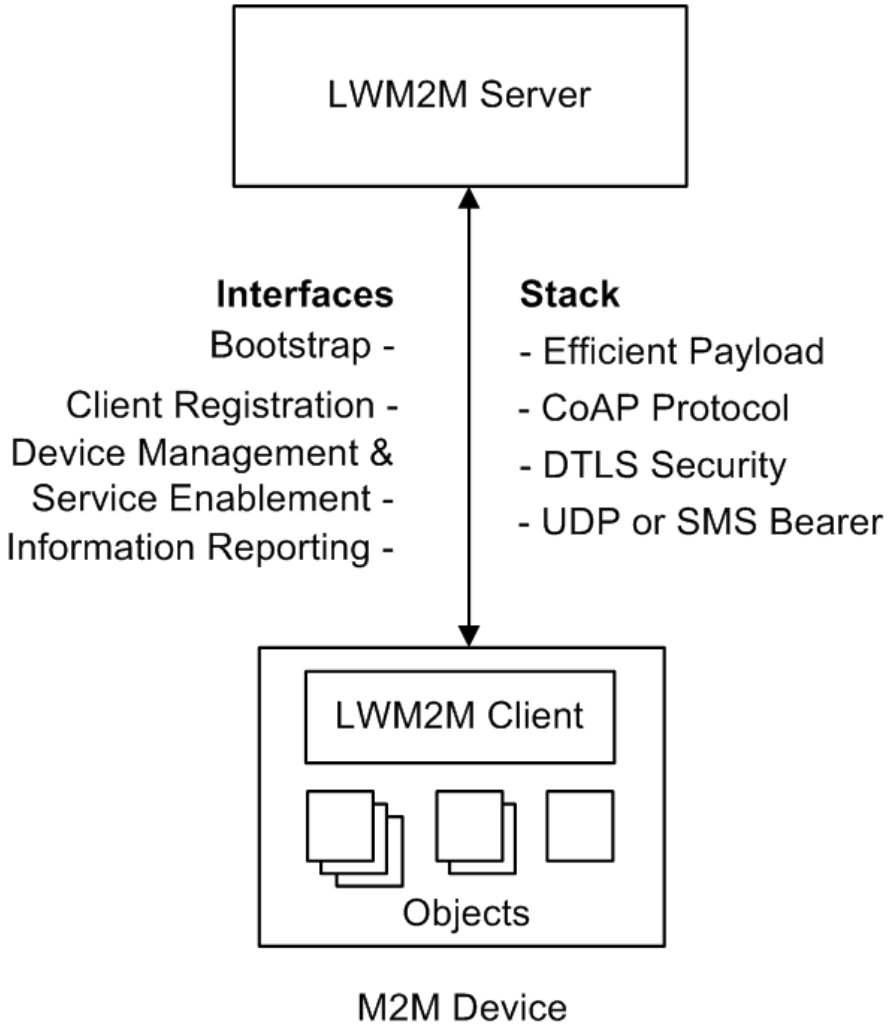
PROTOCOLS & TOOLS

- Communication patterns used:
  - Pub/Sub
  - REST: Representational State Transfer
- Protocols used:
  - CoAP: Constrained Application Protocol
  - LWM2M: Lightweight M2M
  - IPSO Smart Objects
  - DTLS: Datagram Transport Layer Security



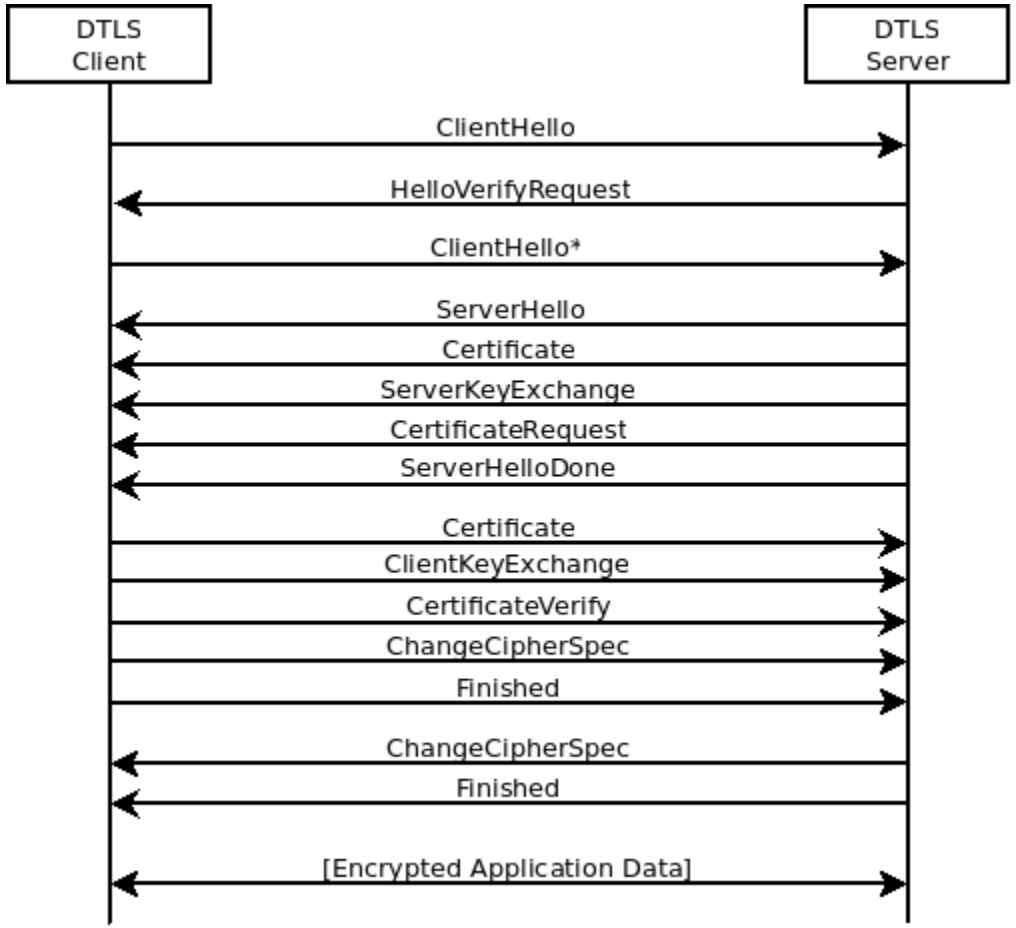
OMA LIGHTWEIGHT M2M

- Device Management protocol
  - Built on top of CoAP
  - Provides REST API for Device Management
  - Standard objects for security, location, firmware, etc...
- 4 Interfaces
  - Bootstrap
  - Client Registration
  - Device Management (Read, Write, Execute)
  - Information Reporting
- Resource Model
  - Client's data is divided into Resources
  - Resources are organized into Objects with different instances



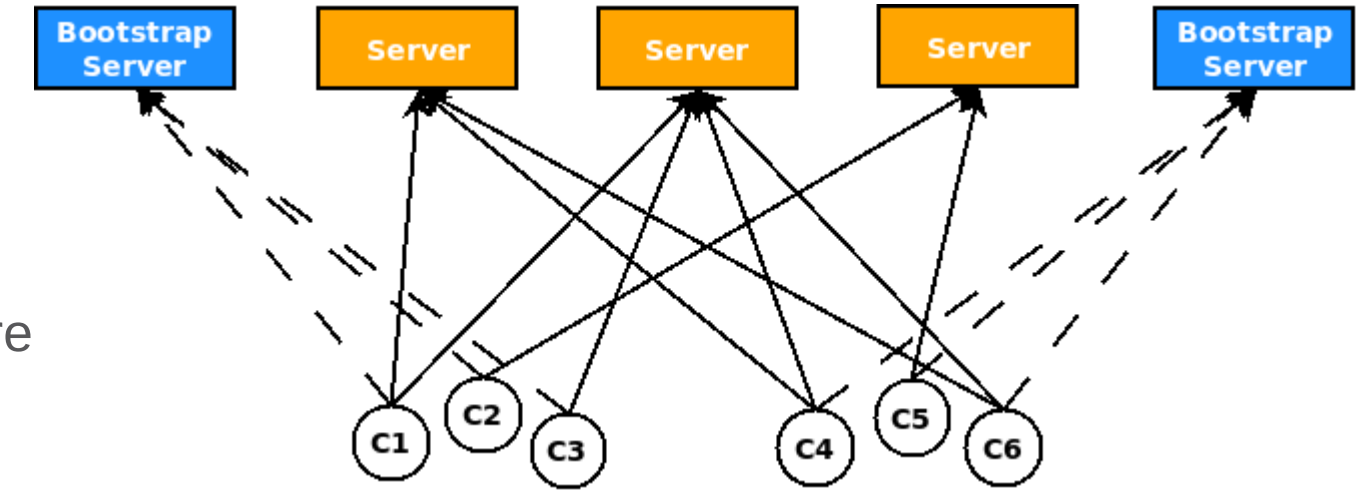
DATAGRAM TRANSPORT LAYER SECURITY

- Datagram Transport Layer Security
  - TLS adaptation for UDP transport protocol
  - Handshake Protocol for security parameters negotiation
  - Provides confidentiality, integrity and authentication.
  - Introduces protocol overhead:
    - 13 bytes for DTLS header
    - 16 bytes for DTLS signature
- Raw Public Key
  - Asymmetric cryptographic keys without certificate
  - Need an out-of-band mechanism for public key validation



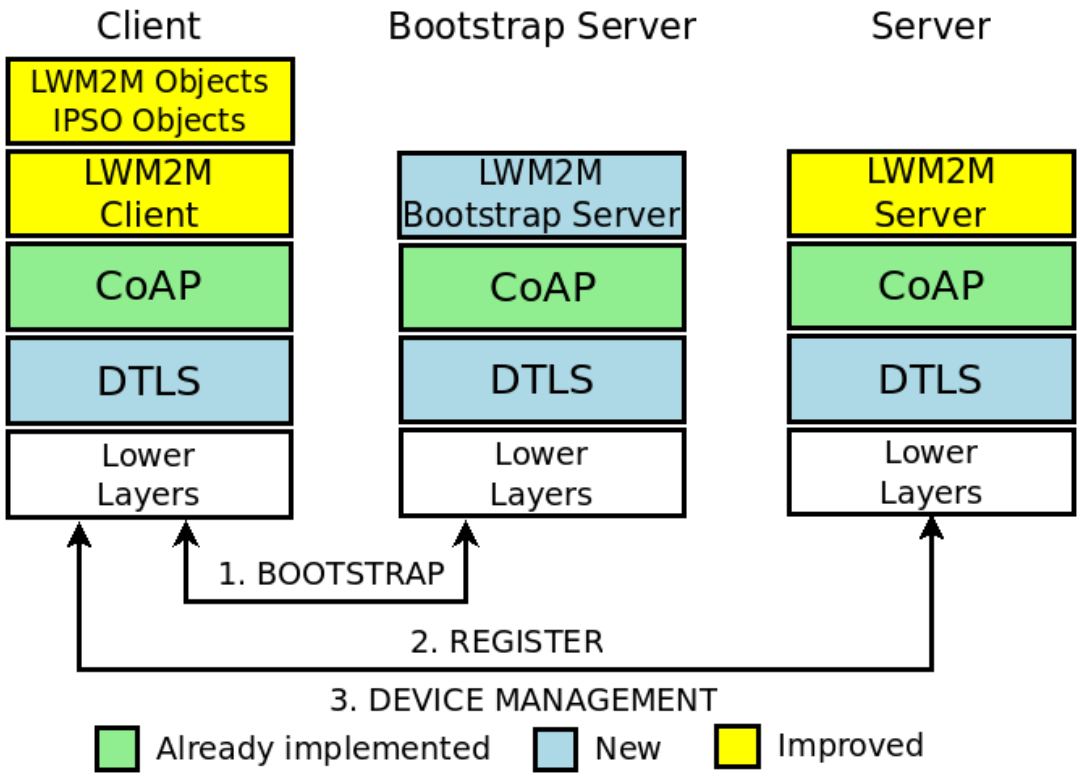
LWM2M BOOTSTRAP SERVER

- Functions
  - Particular server with limited responsibilities
  - Answers only to a bootstrap request (addressed to the ”/bs” resource)
  - Erases information about the LWM2M Bootstrap Server in the Client
  - Writes new information about one or more LWM2M Servers in the Client
  - Uses two LWM2M Objects:
    - LWM2M Security Object (key material)
    - LWM2M Server Object



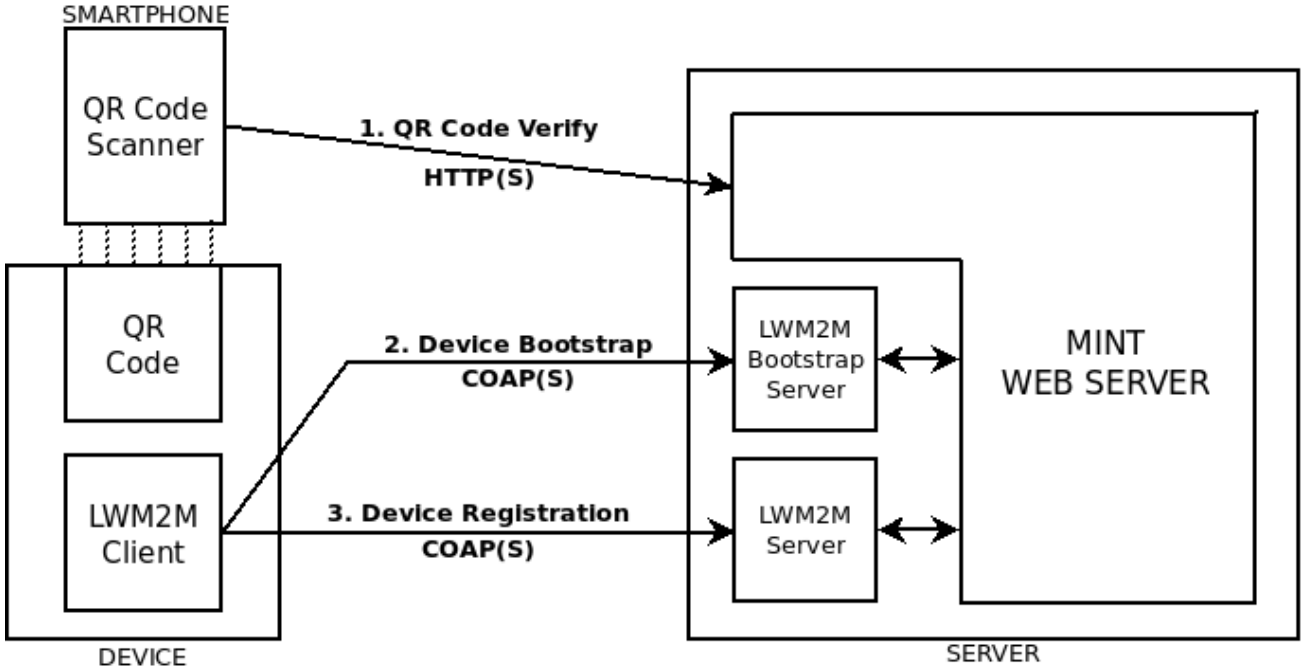
IMPLEMENTATION I

- Implemented stack (**C** language)
  - Lower layers: IPv4/IPv6 + UDP
  - DTLS with **TinyDTLS** library
  - CoAP with **ErbiumCoap**
  - LWM2M with **Wakaama** library
  - **IPSO Objects** as data model
- Entities
  - LWM2M Client (running on the device)
  - LWM2M Bootstrap Server
  - LWM2M Server



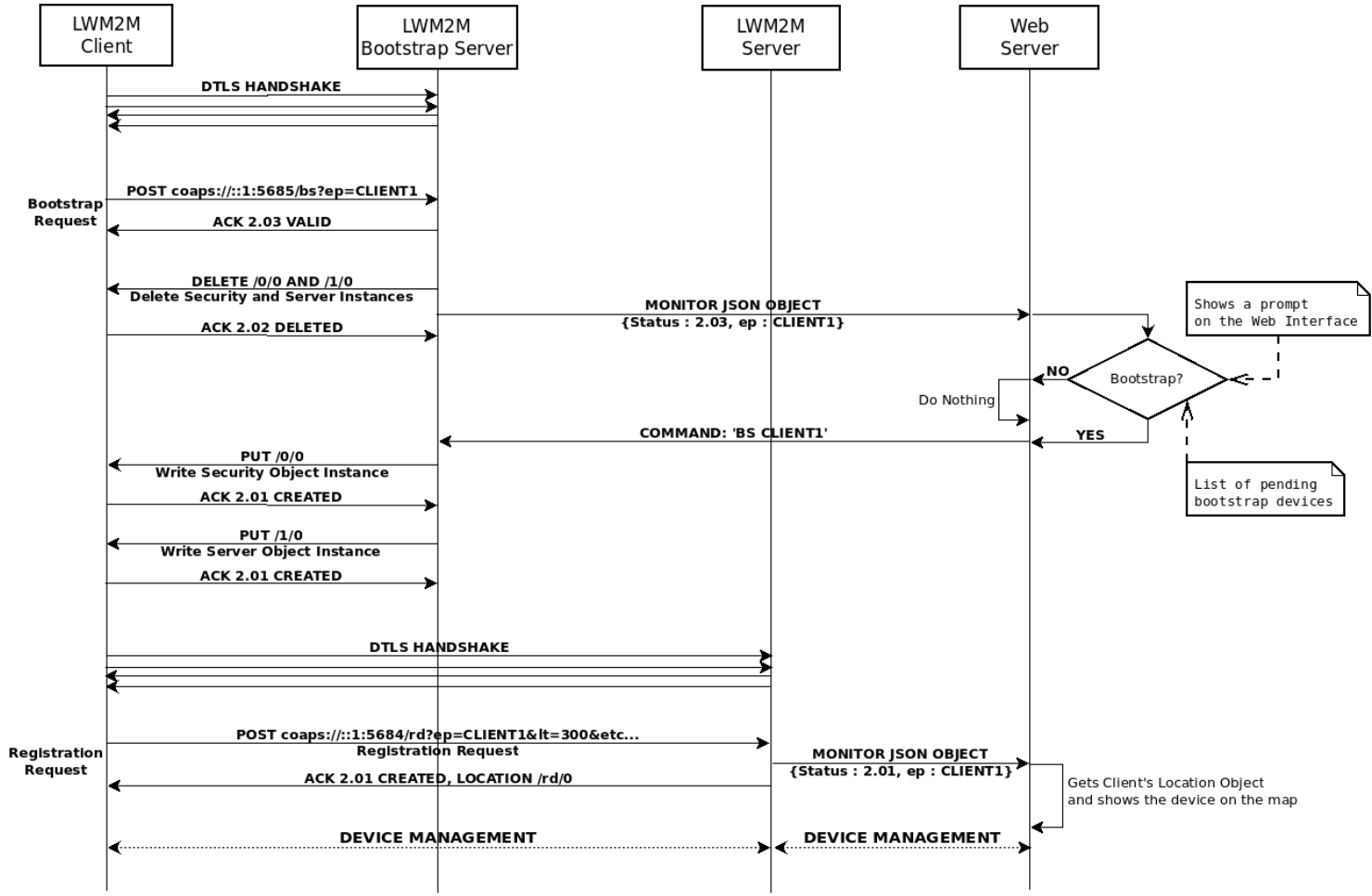
IMPLEMENTATION II

- Raw Public Key validation
  - Out-of-band mechanism
  - The user scans a QR code printed onto the device
  - A notification is sent via HTTP(S) to the Web Server
  - The device is turned on and tries to bootstrap via CoAP(S)
  - The Bootstrap Server notifies the request to the Web Server
  - The user approves the new device (if the information of the QR code matches the request)
  - The device registers to the Server



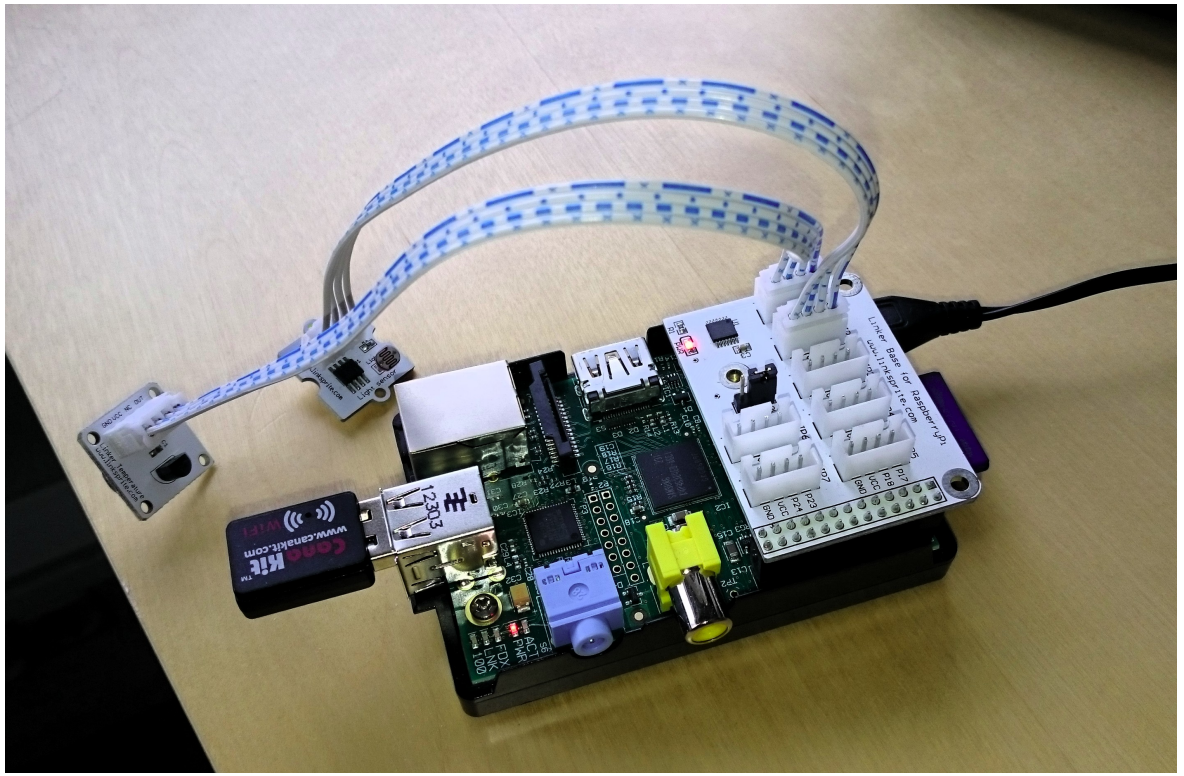
IMPLEMENTATION III

- Communication diagram
  - LWM2M Client opens a DTLS session with the LWM2M Bootstrap Server
  - The BS notifies the Web Server of the bootstrap request
  - The user chooses whether to accept or deny the request
  - If accepted, the BS writes new information in the Client
  - The Client opens a new DTLS session with the LWM2M Server and registers to it
  - The Client is showed on the map



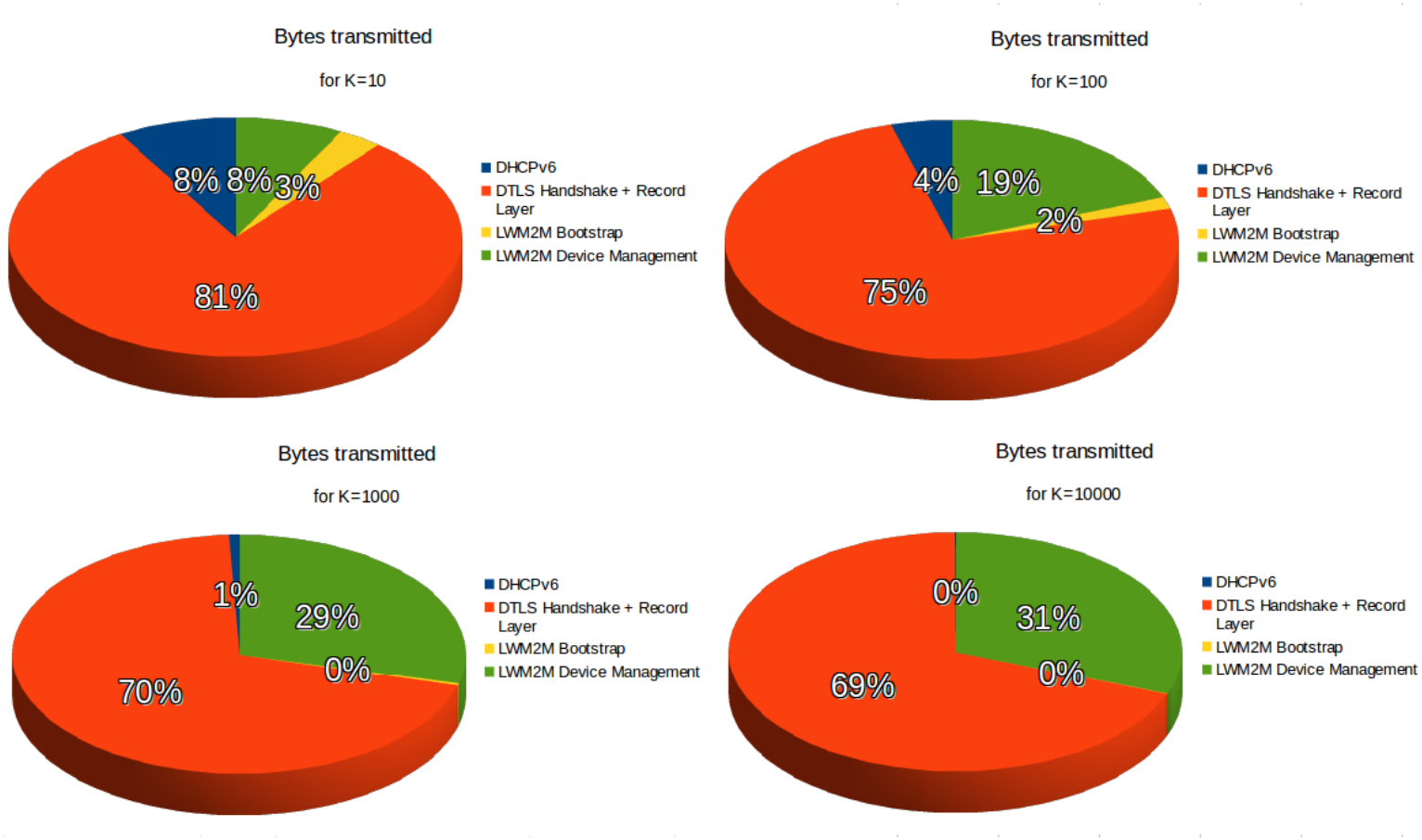
TESTBED AND DEMO

- Configuration
  - LWM2M Client
    - Running on a RaspPi
    - Temperature and Light sensors
    - GPIOs and SPI
  - LWM2M Bootstrap Server
    - Running on a laptop
  - LWM2M Server
    - Running on a laptop



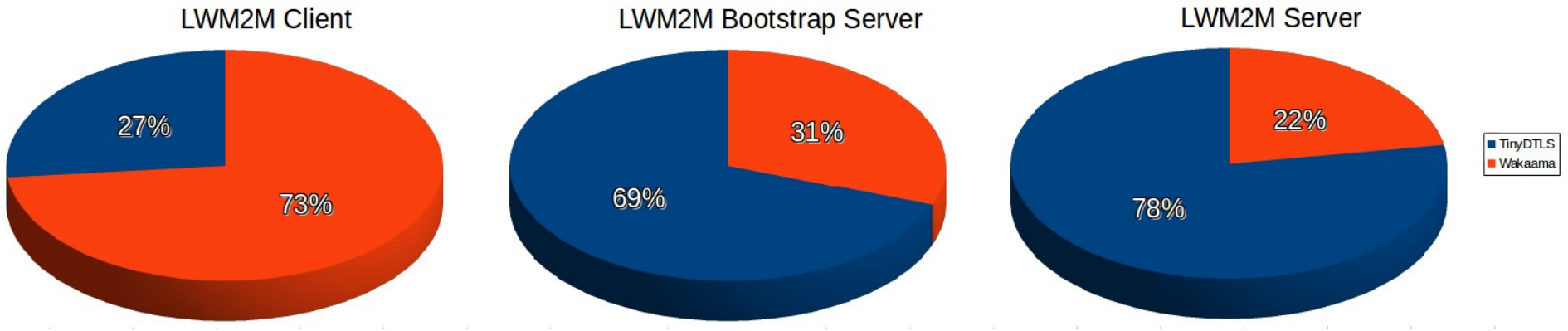
EVALUATION RESULTS I – PROTOCOL OVERHEAD

- DTLS Protocol Overhead
  - 29 bytes for each Application Data packet
  - 1128 bytes for a single DTLS Handshake Protocol session
  - The overhead introduced by the bootstrap phase is negligible over long device management sessions



EVALUATION RESULTS II – MEMORY

› Memory Footprint



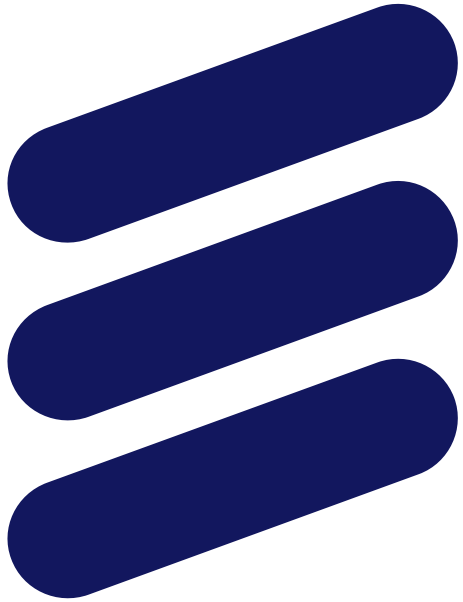
› Memory Usage Peaks

- 4.5 KB difference by average

Endpoint	NoSec Mode	DTLS Mode
LWM2M Client	18.1 KB	22.9 KB
LWM2M BS	16.7 KB	21.1 KB
LWM2M Server	16.2 KB	20.4 KB

CONCLUSIONS AND FUTURE WORK

- Conclusions
  - A working prototype has been developed
  - The prototype shows a significant protocol overhead with the introduction of DTLS layer
  - The prototype shows small memory occupation both with and without DTLS
- Future Work
  - Wakaama improvement
    - Taking care of fragmentation in the Application Layer (CoAP Block Options)
    - Multiple instance resources for the Web Server Interface
  - DTLS stateless compression to reduce the overhead
  - Testing on other devices
    - Power consumption considerations and comparisons



ERICSSON