Lithe: Lightweight Secure CoAP for the Internet of Things

S. Raza, H. Shafagh, etc.

IEEE Sensors 2013, Volume 13

Speaker: Renato Iida, Le Wang

Outline

- Introduction
- Background
  - CoAP and DTLS
  - 6LoWPAN
- DTLS Compression
  - DTLS-6LoWPAN Integration
  - 6LoWPAN-NHC for the Record and Handshake Headers
  - 6LoWPAN-NHC for ClientHello / ServerHello
  - 6LoWPAN-NHC for other Handshake Messages
- Implementation
- Evaluation
  - Packet Size Reduction
  - RAM and ROM Requirement
  - Run-Time Performance
- Conclusion

Outline

- *Introduction*
- Background
  - CoAP and DTLS
  - 6LoWPAN
- DTLS Compression
  - DTLS-6LoWPAN Integration
  - 6LoWPAN-NHC for the Record and Handshake Headers
  - 6LoWPAN-NHC for ClientHello / ServerHello
  - 6LoWPAN-NHC for other Handshake Messages
- Implementation
- Evaluation
  - Packet Size Reduction
  - RAM and ROM Requirement
  - Run-Time Performance
- Conclusion

Introduction

- 6LoWPAN (IPv6 over Low power Wireless Personal Area Network) enables IPv6 in low-power and lossy wireless networks such as WSNs.
  - 6LoWPAN defines *header compression mechanisms.*
- CoAP (Constrained Application Protocol) is designed for simplicity, low overhead and multicast support in resource-constrained environments.

Introduction

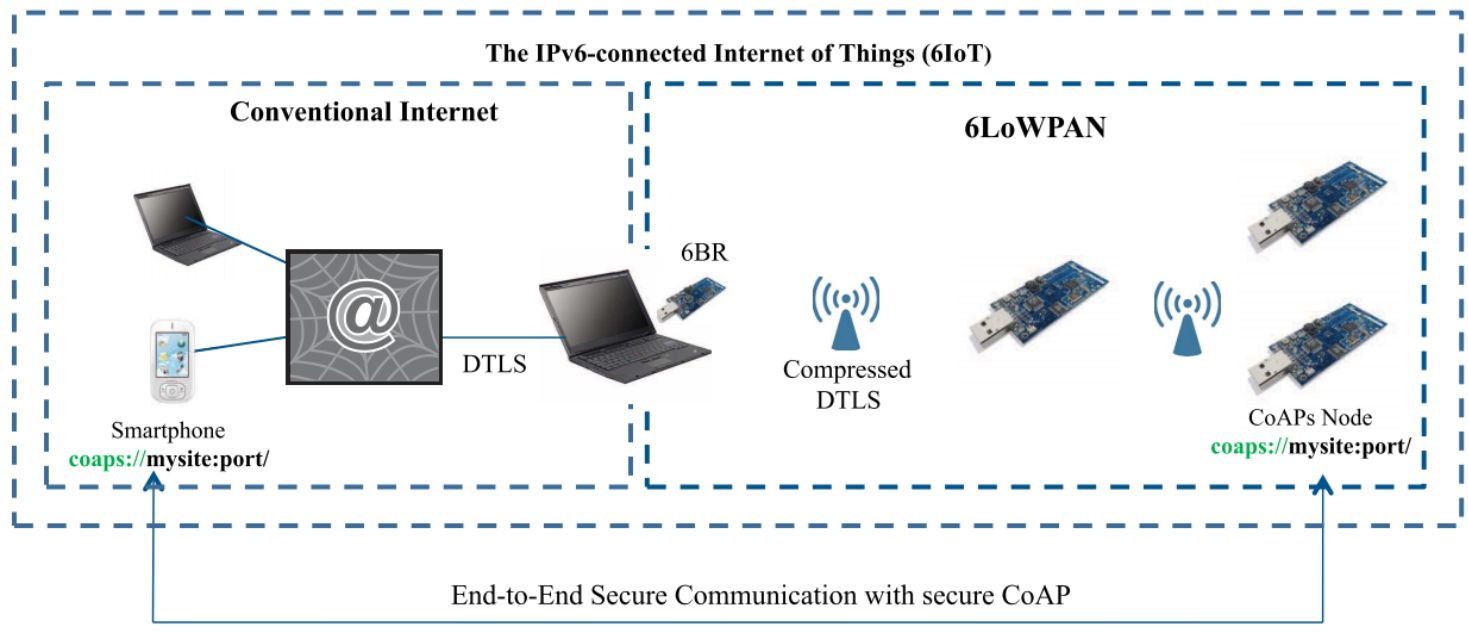
- DTLS (Datagram Transport Layer Security) is used by CoAP as the security protocol
  - For key management and data encryption and integrity protection.
- CoAPs is CoAP with DTLS support, similar to HTTPs.
  - Problem: DTLS is inefficient for constrained IoT devices.
  - Solution: Apply the 6LoWPAN header compression mechanisms to compress DTLS header.

Introduction: Lithe

- Lithe: a lightweight CoAPs by compressing the underneath DTLS protocol with 6LoWPAN header compression mechanisms.
  - To achieve energy efficiency by reducing the message size;
  - To avoid 6LoWPAN fragmentation as 6LoWPAN protocol is vulnerable to fragmentation attaches.
- Lithe is the proposal solution in this paper.

E2E Communication with CoAPs

- **6BR:** 6LoWPAN Border Router is used between 6LoWPAN networks and the Internet to compress/decompress or/and fragment/reassemble messages before forwarding between the two realms.



Outline

- Introduction
- *Background*
  - CoAP and DTLS
  - 6LoWPAN
- DTLS Compression
  - DTLS-6LoWPAN Integration
  - 6LoWPAN-NHC for the Record and Handshake Headers
  - 6LoWPAN-NHC for ClientHello / ServerHello
  - 6LoWPAN-NHC for other Handshake Messages
- Implementation
- Evaluation
  - Packet Size Reduction
  - RAM and ROM Requirement
  - Run-Time Performance
- Conclusion

Background

- Goal: To enable secure yet efficient communication among IoT devices that utilize the CoAP protocol.
- CoAP and DTLS
- 6LoWPAN

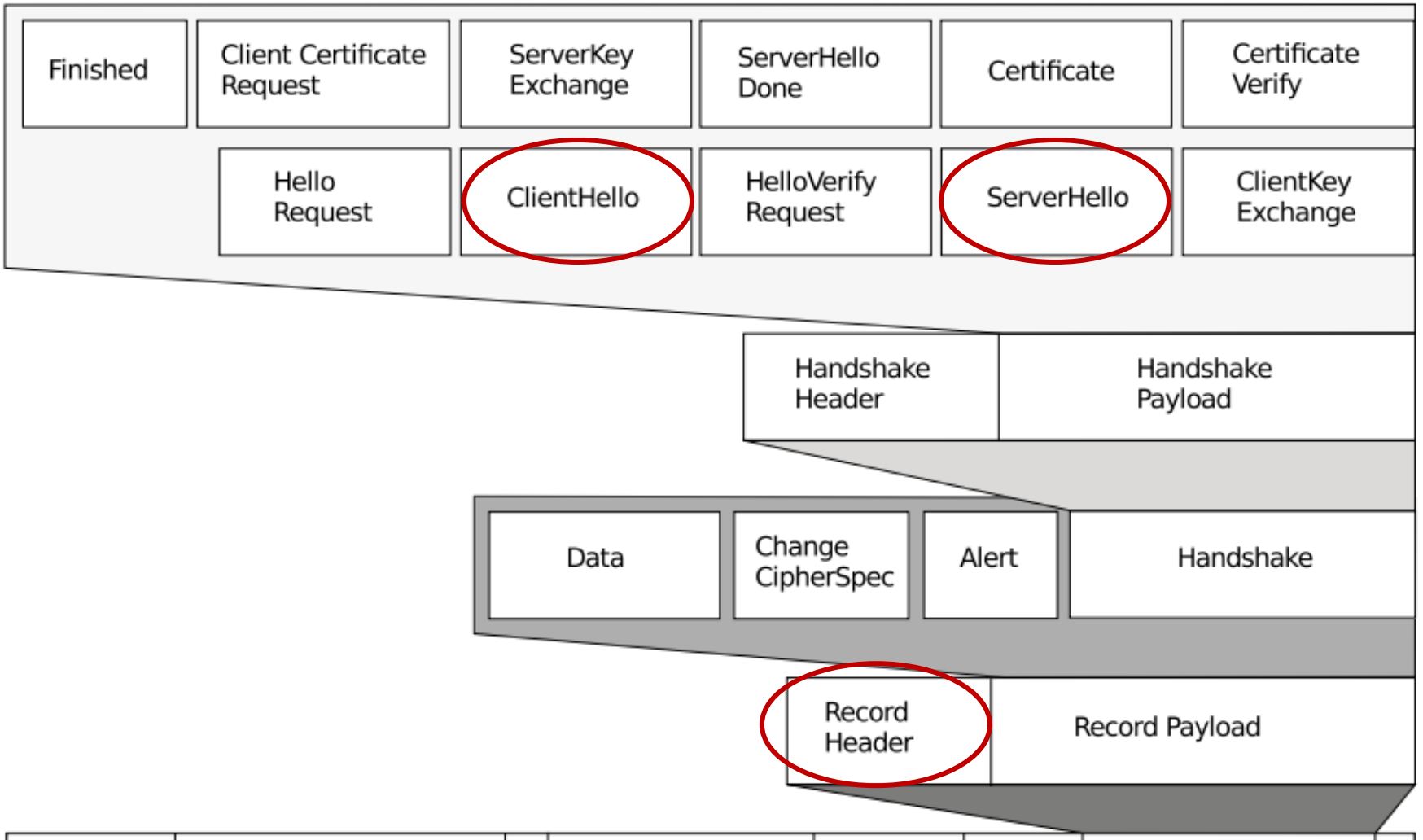
CoAP

- CoAP is a web protocol that runs over the UDP for IoT
  - A variant of HTTP
- Datagram Transport Layer Security (DTLS) is used to protect CoAP transmission.
- Similar to HTTPs (TLS-secured HTTP), CoAPs is DTLS-secured CoAP.
  - *Coaps://myIPv6Address:port/MyResource*

DTLS

- DTLS consists of two sublayers:
  - Upper layer contains:
    - ***Handshake***, *Alert* and *ChangeCipherSpec* protocols
    - Or ***application data***.
  - Lower layer contains the Record protocol
    - Carrier for the upper layer protocols
    - Record header contains content type and fragment fields.
- DTLS is between Application layer and Transport Layer

Layout of a packet secured with DTLS



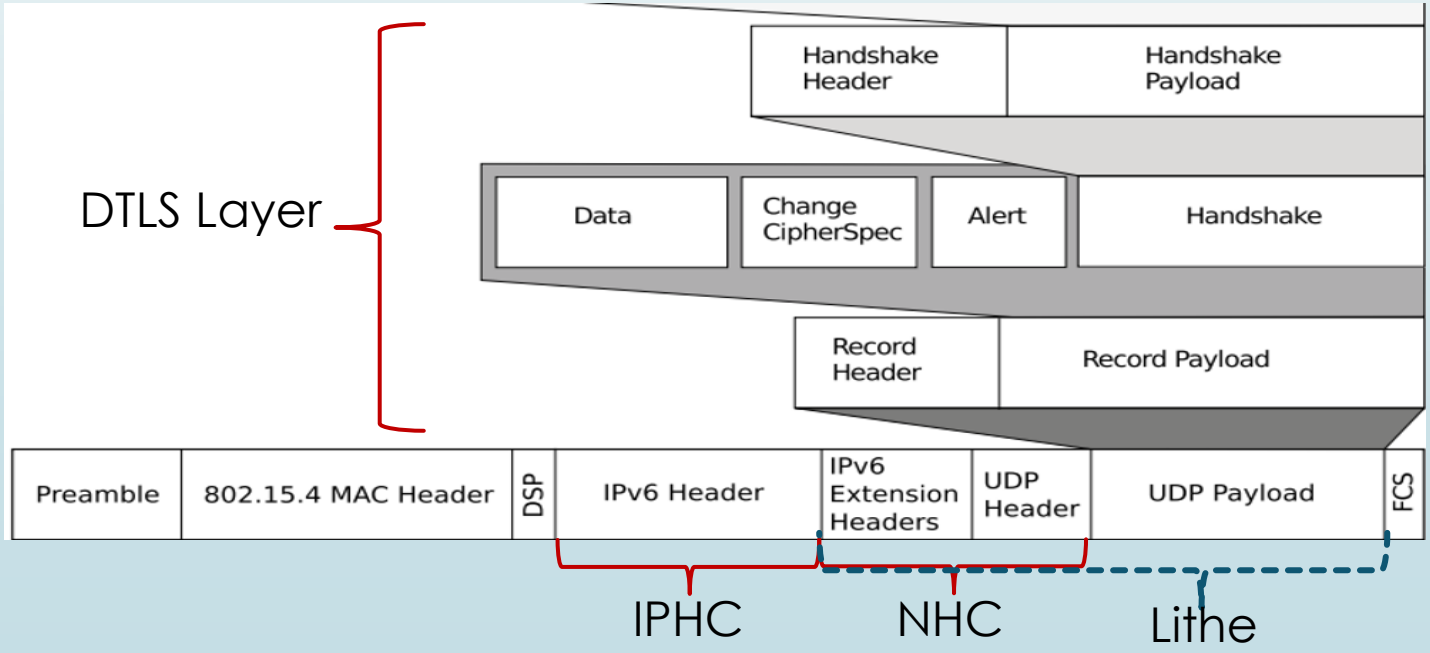
DTLS-Handshake Process

- The handshake messages are used to negotiate security keys, cipher suites and compressing methods.
- This paper is limited to the header compression process only.
- During the handshake process the ClientHello message is sent twice.
  - Without cookie
  - With the server's cookie

DTLS handshake protocol. * means optional.

6LoWPAN

- Header compression
  - IP Header Compression (IPHC)
    - Compress Header to 2 bytes for a single hop network
    - Or 7 bytes for a multi-hop networks *(1-byte IPHC, 1-byte dispatch, 1-byte Hop Limit, 2-byte Source address and 2-byte Destination Address)*
  - Next Header Compression (NHC)
    - Used to encode the IPv6 extension headers and UDP header.
    - Lithe extends the NHC range to UDP payload.



Outline

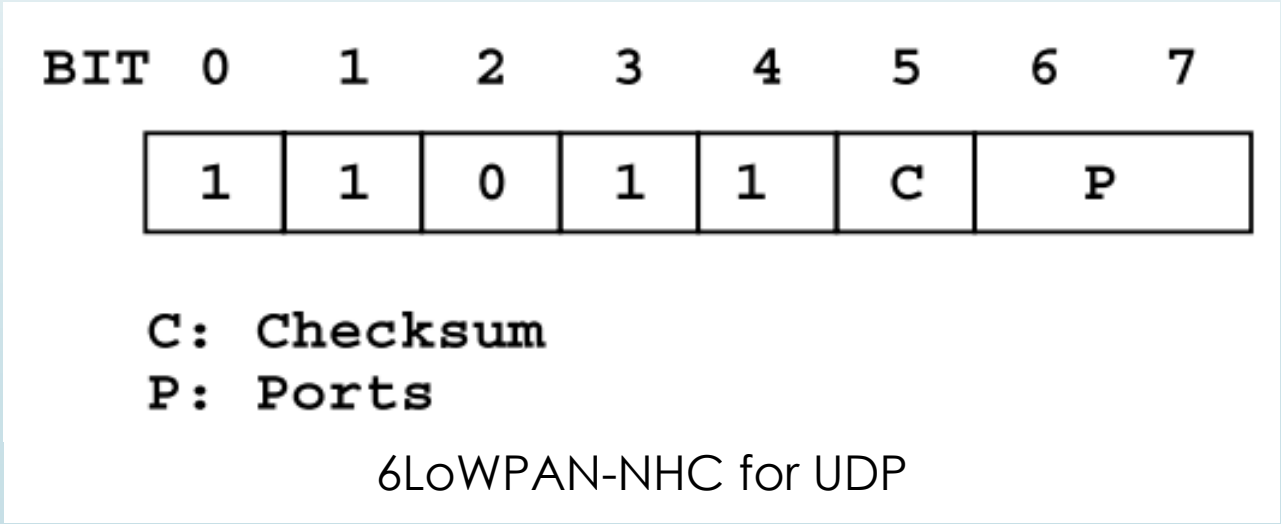
- Introduction
- Background
  - CoAP and DTLS
  - 6LoWPAN
- *DTLS Compression*
  - DTLS-6LoWPAN Integration
  - 6LoWPAN-NHC for the Record and Handshake Headers
  - 6LoWPAN-NHC for ClientHello / ServerHello
  - 6LoWPAN-NHC for other Handshake Messages
- Implementation
- Evaluation
  - Packet Size Reduction
  - RAM and ROM Requirement
  - Run-Time Performance
- Conclusion

DTLS Compression

- DTLS header compression is applied only within 6LoWPAN networks, i.e., between sensor nodes and the 6BR.
  - DTLS-6LoWPAN Integration
  - 6LoWPAN-NHC for the Record and Handshake Headers
  - 6LoWPAN-NHC for ClientHello / ServerHello
  - 6LoWPAN-NHC for other Handshake Messages

DTLS-6LoWPAN Integration

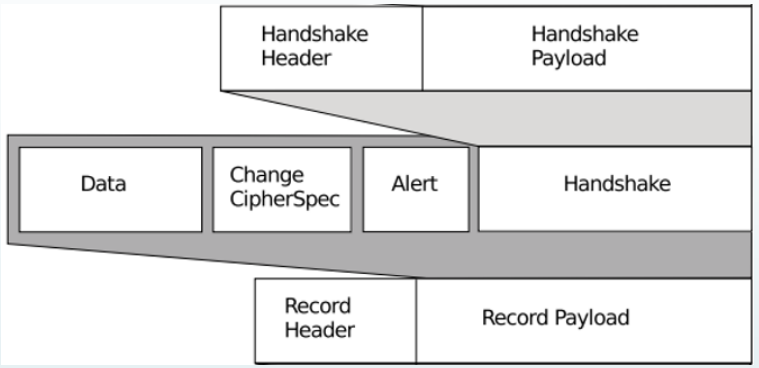
- Apply 6LoWPAN header compression mechanism to compress headers in the UDP payload.
- The ID bits in the NHC for UDP defined in 6LoWPAN:
  - 11110 means the UDP payload is not compressed;
  - 11011 means the UDP payload is compressed with 6LoWPAN-NHC.



6LoWPAN-NHC for UDP

6LoWPAN-NHC for the Record and Handshake Headers

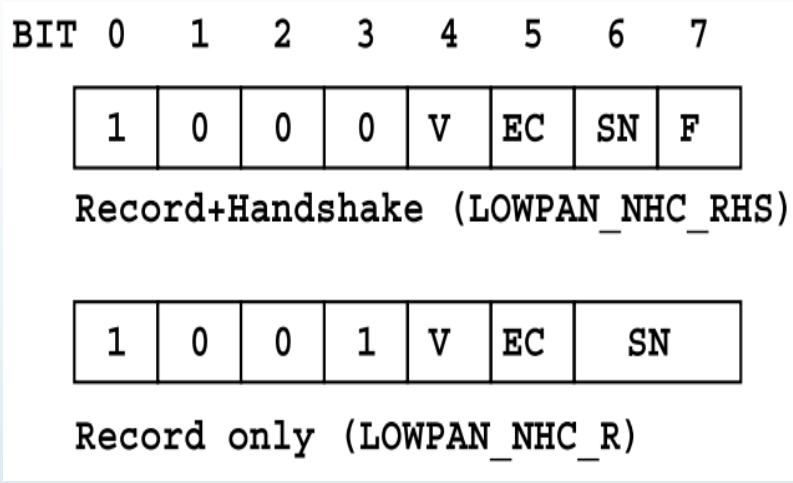
- After compression, the Handshake header can decrease from 12 to 5 bytes and the Record header can decrease from 13 to 3 bytes.



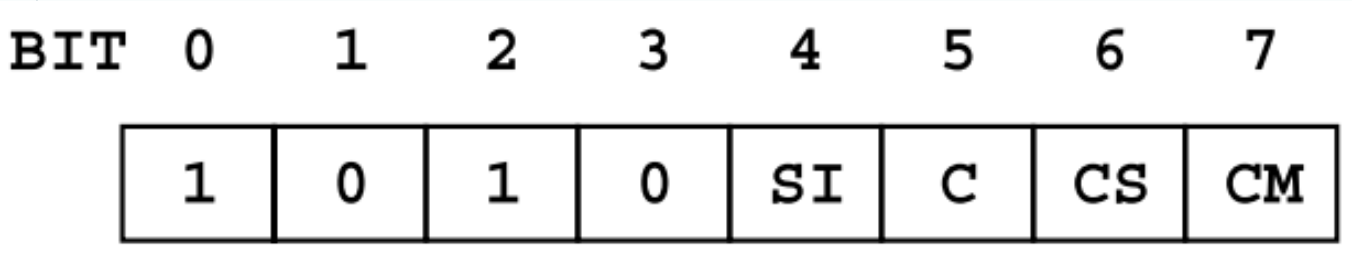
- 6LoWPAN-NHC-RHS
  - 6LoWPAN-NHC for Record + Handshake
  - For Handshake messages
- 6LoWPAN-NHC-R
  - 6LoWPAN-NHC for Record
  - Applied after the DTLS handshake has been performed successfully
  - For application data.

6LoWPAN-NHC-R and RHS

- First 4 bits represent the ID field:
  - 1000 – 6LoWPAN-NHC-RHS
  - 1001 – 6LoWPAN-NHC-R
- Version (v): DTLS version
  - 0 – omit version field (16 bits)
- Epoch (EC):
  - 0, 8 bit epoch is used and the left most 8 bits are omitted.
  - 1, all16 bit epoch is used.
- Sequence Number (SN):
  - 0, 16 bit SN, omit 32 bits
  - 1, 48 bit SN
- Fragment (F):
  - 0, not fragment.
    - Omit 2 x *(offset + length)* 6 bytes.
  - 1, fragment applied.

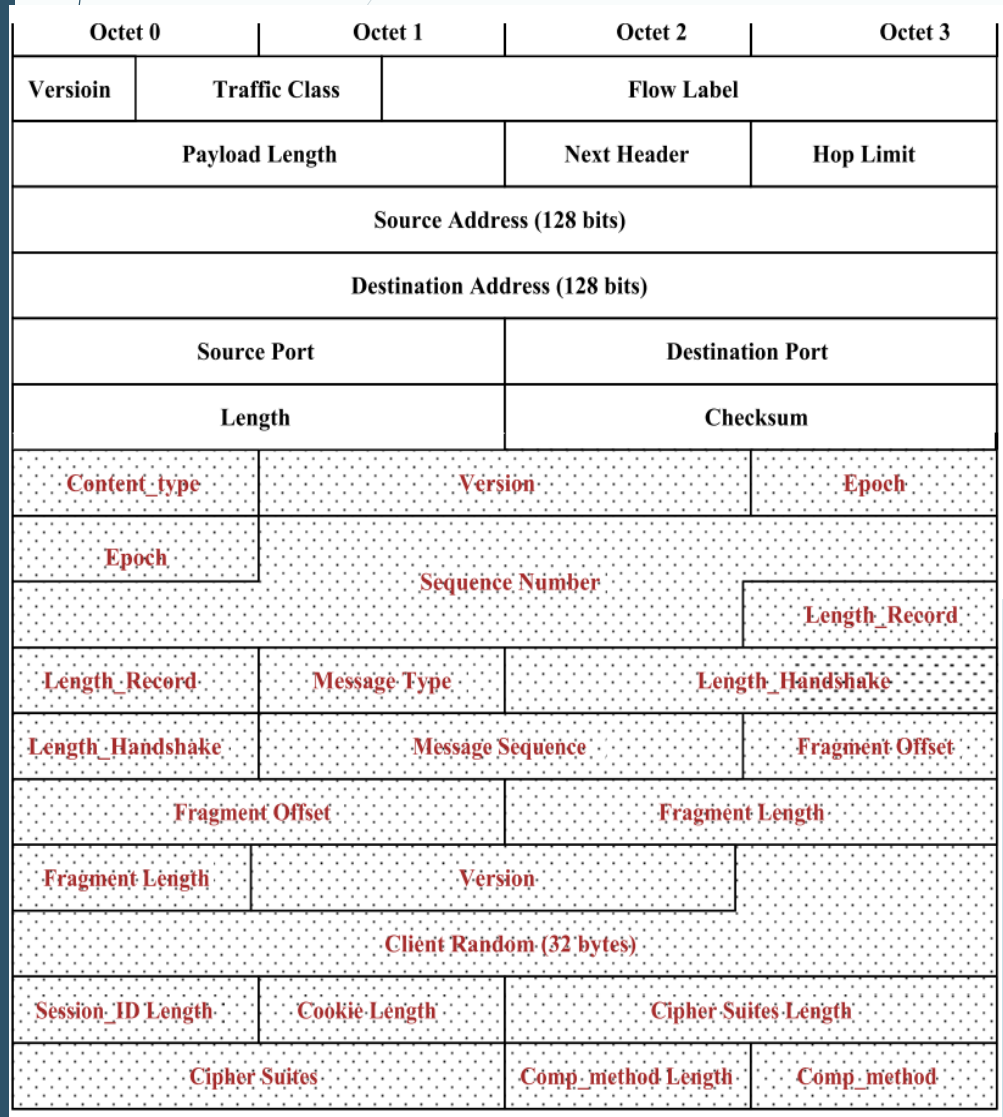


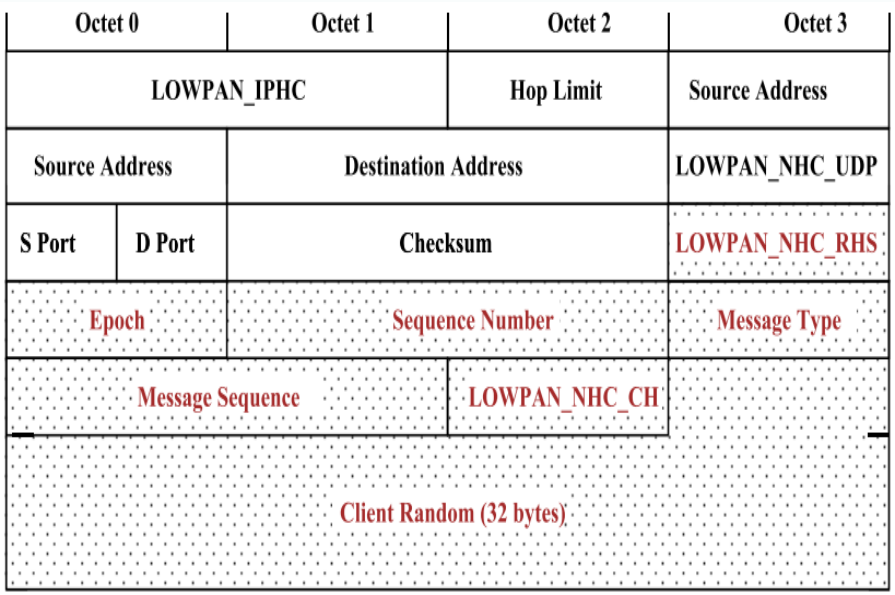
6LoWPAN-NHC-CH



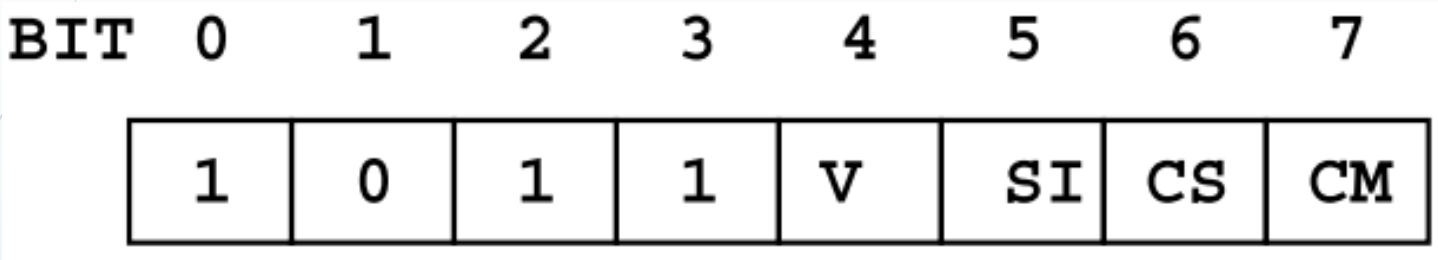
- First 4 bits is ID, 1010
- When the parameter is set to 0, the corresponding field is omitted.
  - Session ID (SI): omit 8 bits
  - Cookie (C): omit 16 bits
  - Cipher Suites (CS): omit 16bits
  - Compression Method (CM): Omit 8 bits

6LoWPAN-NHC for ClientHello





6LoWPAN-NHC-SH



- Similar to ClientHello except:
  - ID field is 1011
  - V (Server DTLS Version): 0 - DTLS 1.0, omit 16 bits

6LoWPAN-NHC for other Handshake Messages

- The remaining mandatory handshake messages:
  - *ServerHelloDone,ClientKeyExchange, Finish* have no fields that could be compressed.

Outline

- Introduction
- Background
  - CoAP and DTLS
  - 6LoWPAN
- DTLS Compression
  - DTLS-6LoWPAN Integration
  - 6LoWPAN-NHC for the Record and Handshake Headers
  - 6LoWPAN-NHC for ClientHello / ServerHello
  - 6LoWPAN-NHC for other Handshake Messages
- *Implementation*
- Evaluation
  - Packet Size Reduction
  - RAM and ROM Requirement
  - Run-Time Performance
- Conclusion

Implementation

- Extension to the 6LoWPAN in the Contiki OS;
- Hardware platform: WiSMote.
- Lithe implementation consists of four components:
  - DTLS: open source tinyDTLS;
  - CoAP: default CoAP in Contiki;
  - CoAP-DTLS integration module: Connects the CoAP and DTLS to enable CoAPs.
  - DTLS header compression.

Implementation

- The 6LoWPAN layer resides between the IP and MAC layers.
- While applying header compression, the End-to-End security of DTLS is not compromised. .

Outline

- Introduction
- Background
  - CoAP and DTLS
  - 6LoWPAN
- DTLS Compression
  - DTLS-6LoWPAN Integration
  - 6LoWPAN-NHC for the Record and Handshake Headers
  - 6LoWPAN-NHC for ClientHello / ServerHello
  - 6LoWPAN-NHC for other Handshake Messages
- Implementation
- *Evaluation*
  - Packet Size Reduction
  - RAM and ROM Requirement
  - Run-Time Performance
- Conclusion

Evaluation

- Packet Size Reduction
- RAM and ROM Requirement
- Run-Time Performance
  - DTLS Compression Overhead
  - CoAPs Initialization
  - CoAPs Request-Response

Evaluation - Packet Size Reduction

Number of Bits Sent and Space Saving

DTLS Header	Without Comp. [Bit]	With Comp. [Bit]	Space Saving
Record	104	40[1]	62%
Handshake	96	24[1]	75%
`ClientHello`	336[2]	264[2]	23%
`ServerHello`	304	264[3]	14%
`CertificateRequest`	40	0	100%

Evaluation – RAM/ROM Requirement

ROM AND STATIC RAM REQUIREMENTS FOR LITHE

Feature	ROM [Byte]	RAM [Byte]
DTLS Crypto (SHA-256, CCM, AES)	6590	2868
DTLS	10662	989
Contiki OS	32145	4979
CoAP	8632	582
DTLS Compression	2820	1
Total	60849	9419

Evaluation - Run-Time Performance

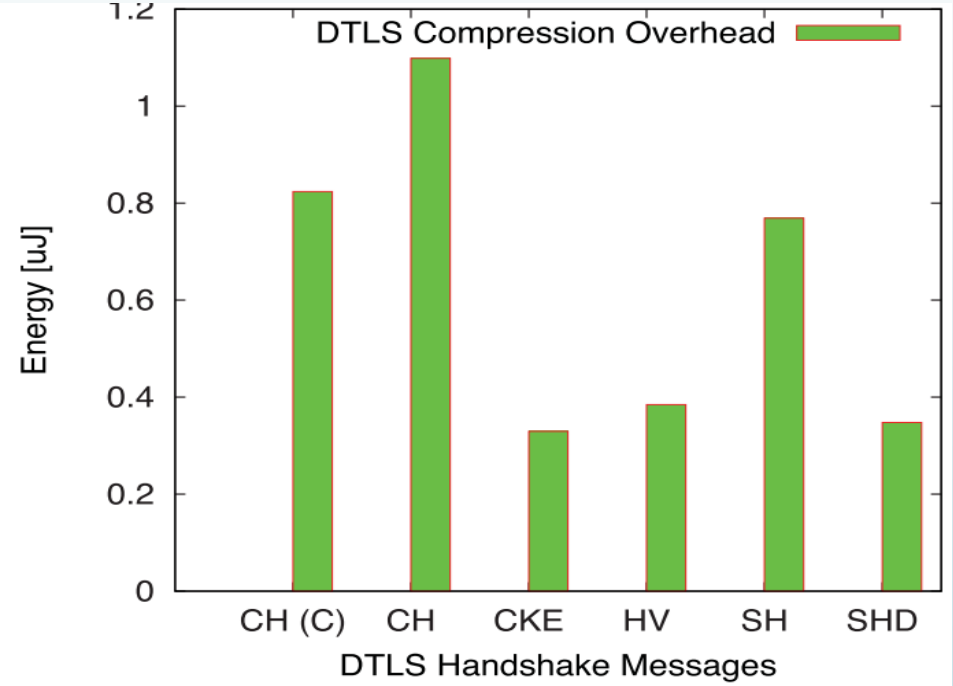
- Radio Duty Cycling (RDC)
  - With RDC, the radio is off most of the time and is turned on either in certain intervals to check the medium for incoming packets or to transmit packets.
- Duty cycled MAC protocol, X-MAC
- Metrics:
  - Energy consumption
    - Energy estimation module in Contiki OS
    - Conversion from absolute timer values to energy:

$$\text{Energy } [mJ] = \frac{\text{ticks} \times \text{I } [mA] \times \text{Voltage } [V]}{\text{ticks per second}}$$

  - Network-wide round trip time (RTT)

Evaluation - Run-Time Performance

- DTLS Compression Overhead
  - The overhead caused through in-node computation for compression and decompression of DTLS headers is almost negligible.



Additional Energy Consumption for Compression of the Handshake Messages.

- CH – ClientHello
- CH(C) – ClientHello with Cookie
- CKE – ClientKeyExchange
- HV – HelloVerify
- SH – ServerHello
- SHD - ServerHelloDone

- For a DTLS handshake based on pre-shared keys, 4.2uJ of energy is consumed for compression

Evaluation - Run-Time Performance

- CoAPs Initialization
  - The tradeoff between additional in-node computation vs. reduced packet sizes shows itself in the energy consumption for packet transmission in a DTLS handshake.

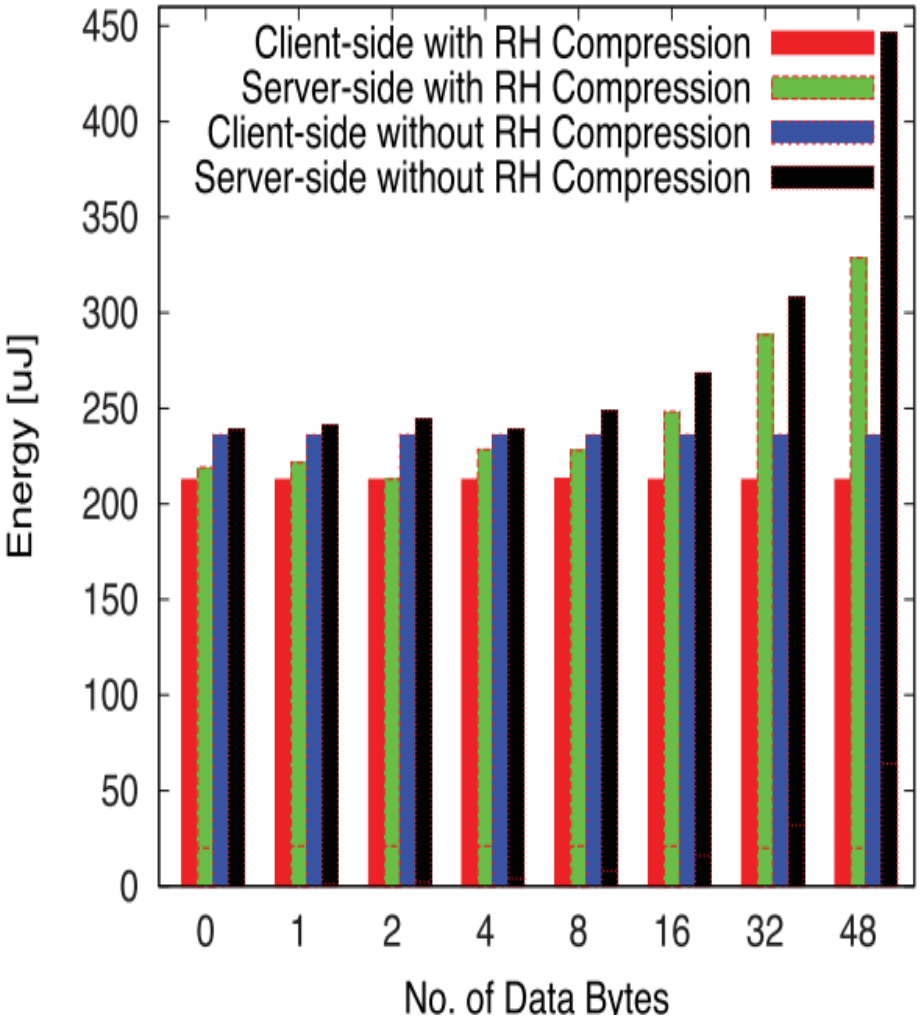
Compression	Client-side [uJ]	Server-side [uJ]	Total [uJ]
Without	1756.66	1311.65	3068.31
With	1467.54	1143.47	2611.01

- 15% less energy is used transmit/receive compressed packets.

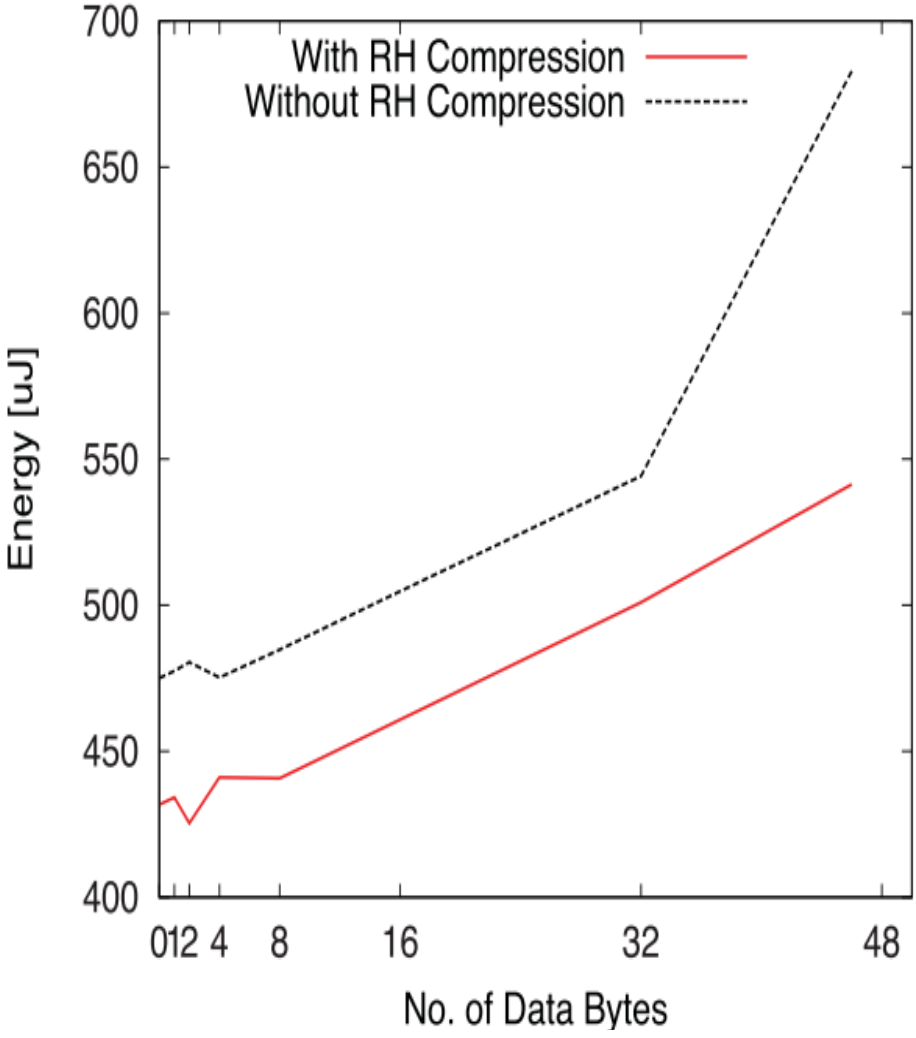
Evaluation - Run-Time Performance

- CoAPs Request-Response
  - Once the CoAPs initialization phase is completed, i.e., the handshake has been performed, a sensor node can send/receive secure CoAP messages using the DTLS Record protocol.
- Metrics
  - Energy consumption
  - RTT

Evaluation – Energy Consumption

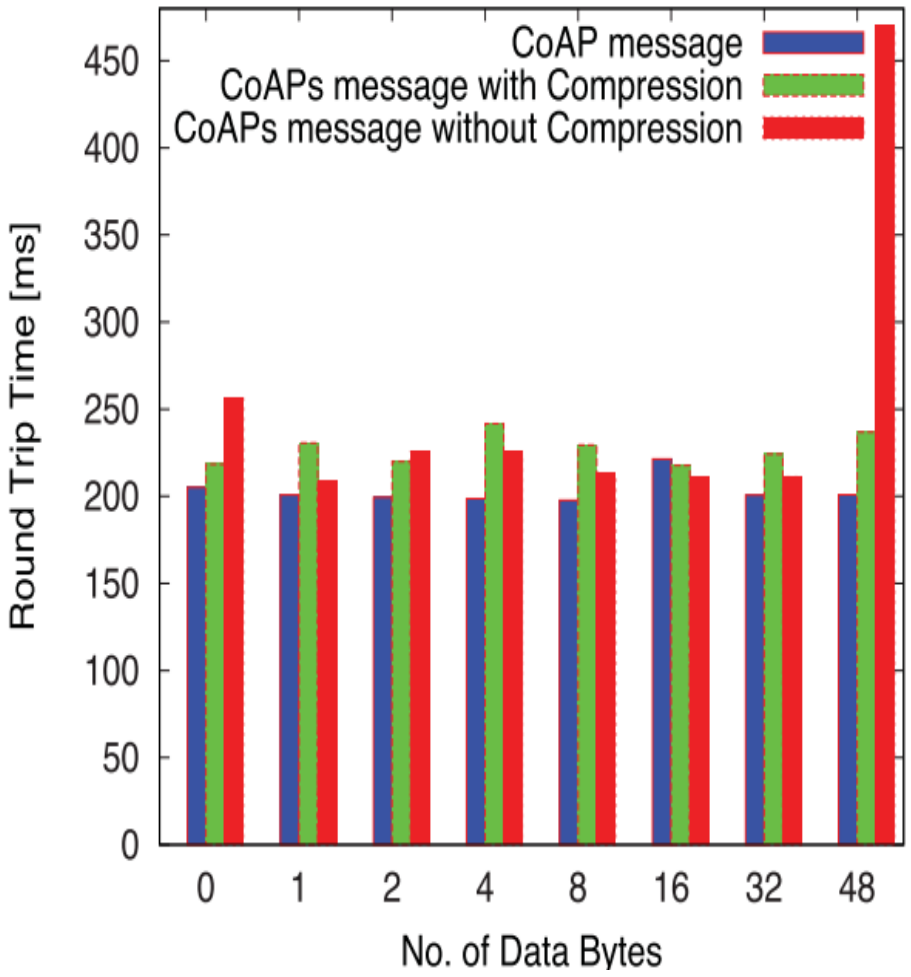


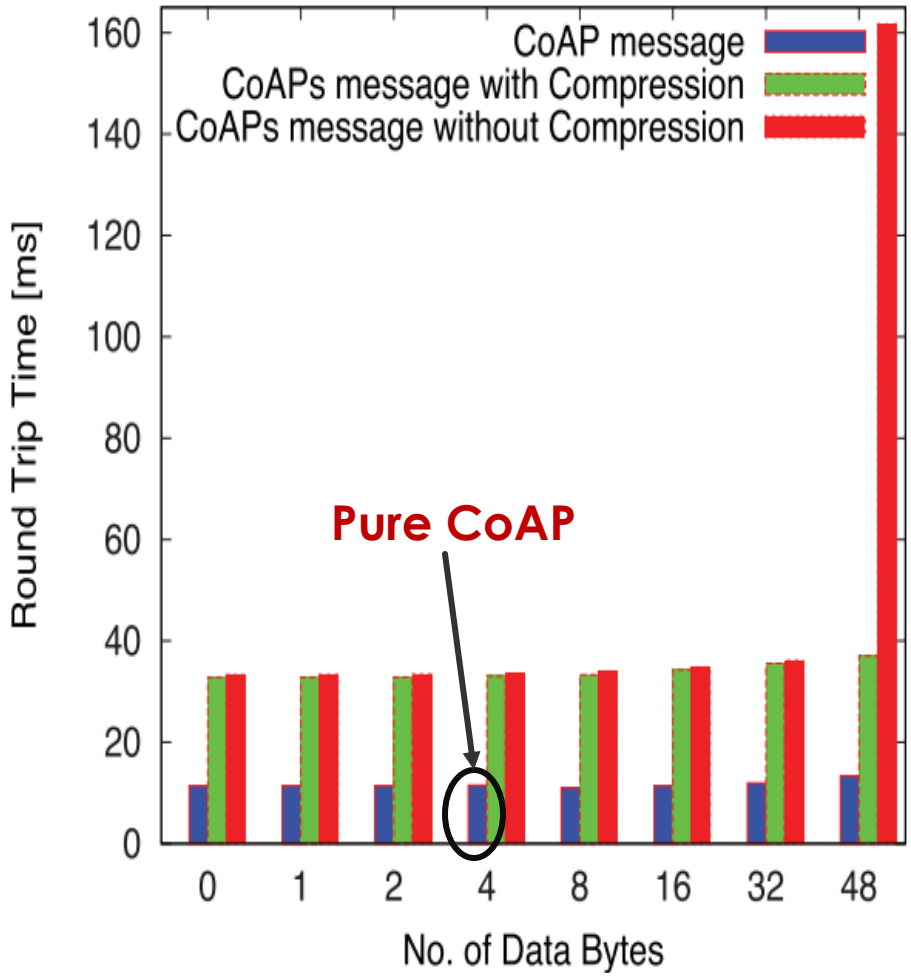
The Energy Consumption from Client/Server w/out RH Compression



The Energy Consumption from the sum of Client/Server w/out RH Compression

Evaluation – Round Time Trip (RTT)





Comparison of RTT for Lithe, CoAPs and CoAP

Outline

- Introduction
- Background
  - CoAP and DTLS
  - 6LoWPAN
- DTLS Compression
  - DTLS-6LoWPAN Integration
  - 6LoWPAN-NHC for the Record and Handshake Headers
  - 6LoWPAN-NHC for ClientHello / ServerHello
  - 6LoWPAN-NHC for other Handshake Messages
- Implementation
- Evaluation
  - Packet Size Reduction
  - RAM and ROM Requirement
  - Run-Time Performance
- *Conclusion*

Contribution

- The first paper to propose 6LoWPAN compressed DTLS and enable lightweight CoAPs support for the IoT.
- Provide novel and standard compliant DTLS compression mechanisms that aim to increase the applicability of DTLS and, thus, CoAPs for constrained devices.
- Implement the compressed DTLS in an OS for the IoT and evaluate it on real headware;
- Lithe is more efficient compared to uncompressed CoAP/DTLS.