Constrained Application Protocol (CoAP)

Mohit P. Tahiliani

Assistant Professor,

Department of Computer Science & Engineering,

National Institute of Technology Karnataka, Surathkal

Homepage: cse.nitk.ac.in/faculty/mohit-p-tahiliani

Inbox: tahiliani@nitk.edu.in

Blog: mohittahiliani.blogspot.com

{github, gitlab}: mohittahiliani

Outline of the presentation

- Overview of CoAP
  - Motivation
  - Features
  - Working of CoAP
- Implementations of CoAP
  - Openly available implementations
  - Commercial implementations
- Experiment 1: Emulating CoAP using network namespaces
  - Creating network namespaces and virtual NICs
  - Setting up a topology
  - Running CoAP on a virtual topology
  - Capturing CoAP packets and tracing using Wireshark
- Experiment 2: Emulating CoAP using NeST
  - Creating network namespaces, virtual NICs and setting up topology
  - Running CoAP on NeST topology

Overview of CoAP

Motivation

- HTTP + REST (Representational State Transfer)
  - commonly used for communication over Internet
  - Depends on TCP to provide reliability and congestion control
  - Leverages features of HTTP such as persistent connections
  - Not suitable for communication in IoT
- IoT networks: characterised as Low power and Lossy Networks (LLNs)
  - Low memory and computing power (constrained node)
  - Low bandwidth and lossy wireless channels (constrained network)
  - TCP maintains states for every open connection; becomes too heavy!
  - IoT traffic is event based, persistent connections not required.
  - Can we have a lightweight application protocol for IoT? Yes: CoAP

CoAP and its features

- Overview of CoAP
  - Application layer protocol for constrained nodes and networks
  - Standardised by IETF in RFC 7252 (CoRE Working Group)
  - Developed for machine-to-machine communication
  - Provides Request/Response model similar to HTTP
  - Runs atop UDP (latest work on CoAP makes it work with TCP too!)
- Features
  - Asynchronous message exchanges
  - Low header overhead and parsing complexity
  - Request Methods: GET, POST, PUT, DELETE
  - Message types: Confirmable, Non Confirmable, ACK and Reset

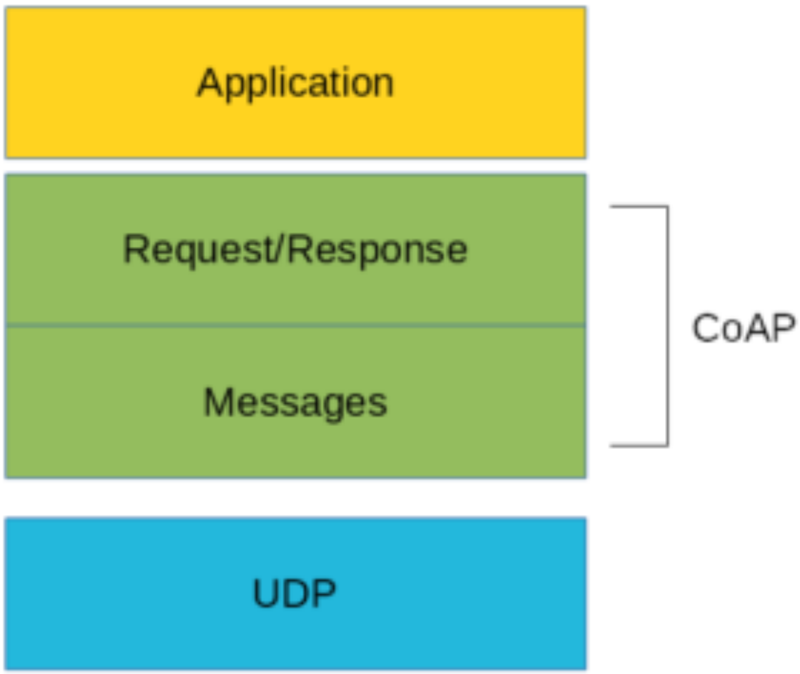
Working of CoAP: Layers

- ❑ Two layers of CoAP
  - ❑ Request/Response Layer:

    - manages interaction between devices

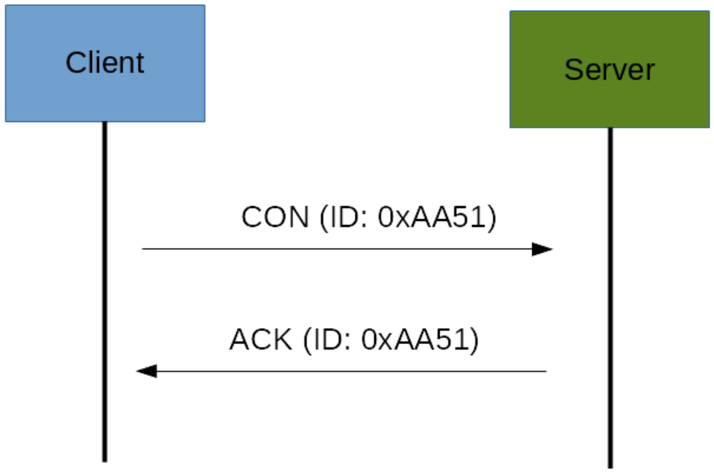
  - ❑ Messaging Layer:

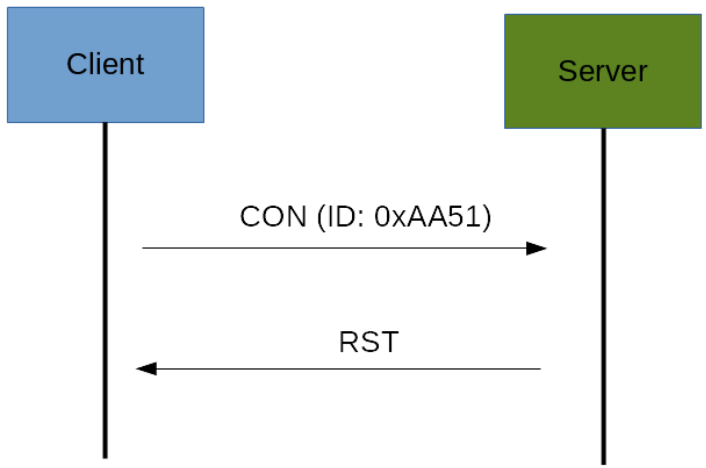
    - deals with UDP and manages the asynchronous messages (e.g., the server may respond to requests from clients later when it is ready)



Ref.: CoAP Protocol: Step-by-Step Guide - https://dzone.com/articles/coap-protocol-step-by-step-guide

Working of CoAP: Confirmable messages

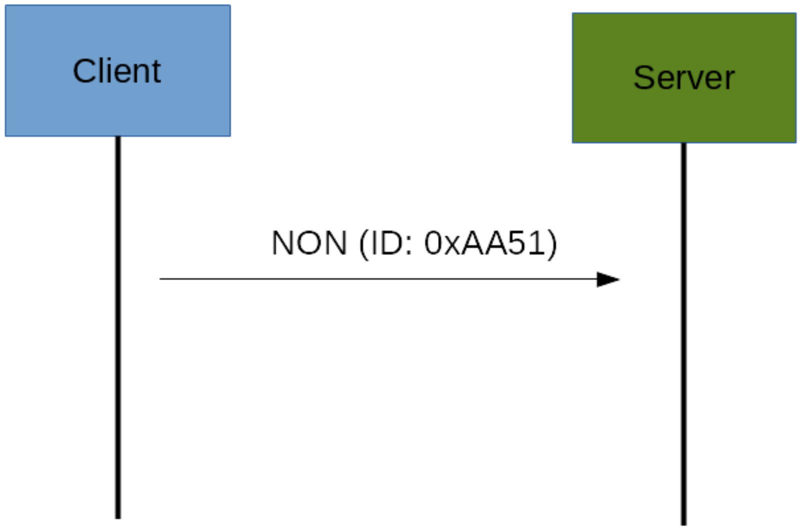




- CoAP messages (1)
  - Confirmable (CON) messages
    - used when reliability is needed
    - Server replies with an ACK message to every CON message
    - Server replies with a RST message if there is a problem

Ref.: CoAP Protocol: Step-by-Step Guide - https://dzone.com/articles/coap-protocol-step-by-step-guide

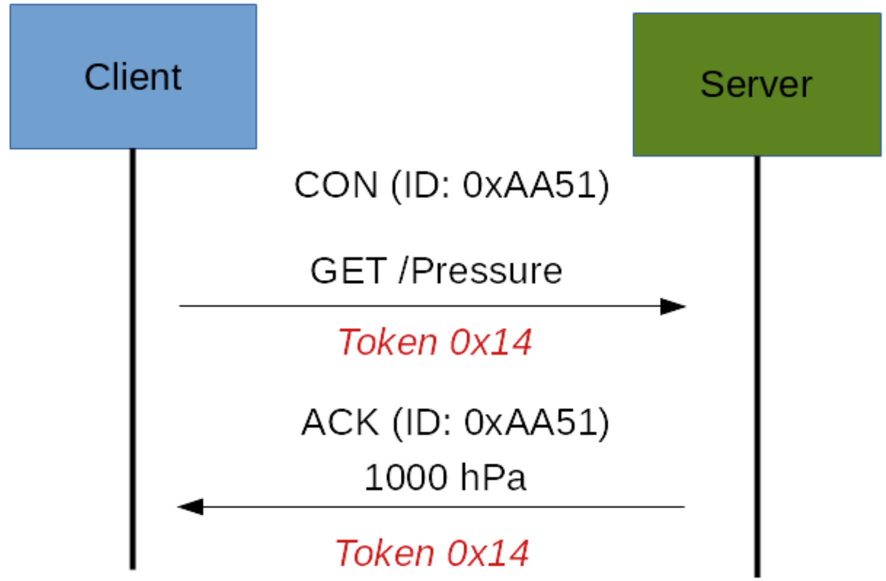
Working of CoAP: Non-Confirmable messages



- CoAP messages (2)
  - Non-Confirmable (NON) messages
    - used when reliability is not needed
    - Server does not reply to NON messages
    - Not recommended for applications that need a bit of reliability

Ref.: CoAP Protocol: Step-by-Step Guide - https://dzone.com/articles/coap-protocol-step-by-step-guide

Working of CoAP: Request/Response

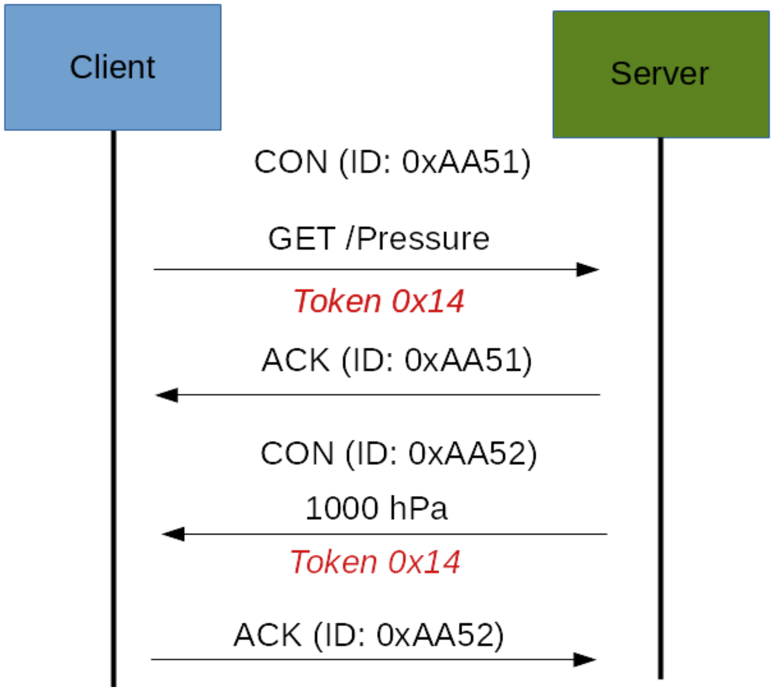


- CoAP Request/Response models
  - If the server can reply immediately to the CON message
    - ACK is sent
  - Otherwise
    - The server sends an empty ACK, later follows up with the client

Ref.: CoAP Protocol: Step-by-Step Guide - https://dzone.com/articles/coap-protocol-step-by-step-guide

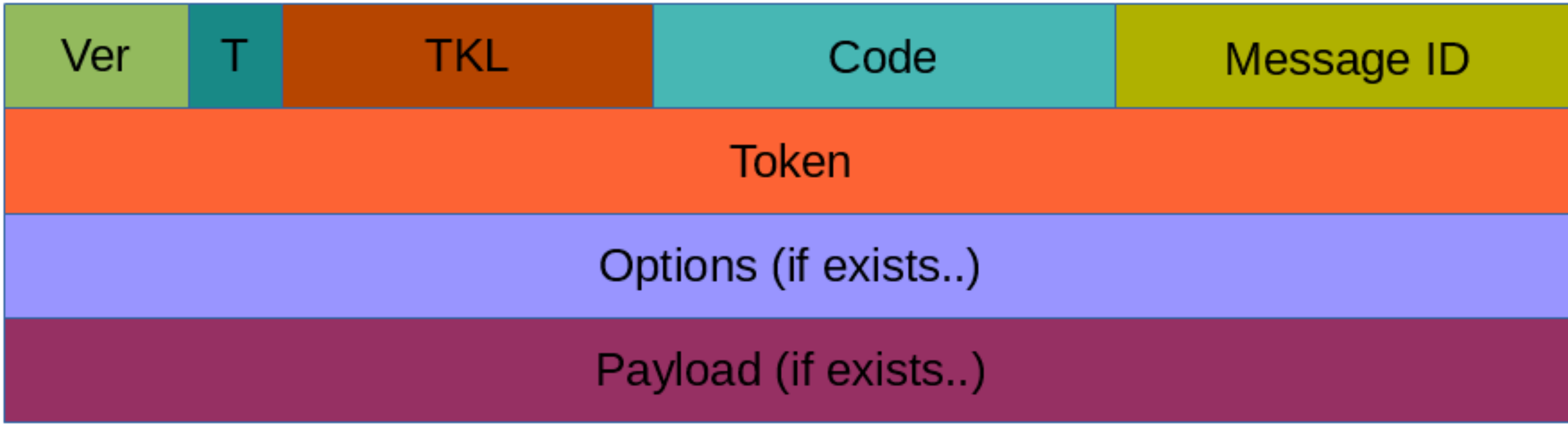
Working of CoAP: Request/Response (contd ...)

- CoAP Request/Response models
  - How does the server follow up with the client later?
    - It sends a CON with the reply, if client requested using CON
    - Client then ACKs the reply from the server
    - If client requested using a NON, server follows up with a NON



Ref.: CoAP Protocol: Step-by-Step Guide - https://dzone.com/articles/coap-protocol-step-by-step-guide

CoAP: Message Format



- ❑ Ver: 2-bit unsigned integer indicating the version of CoAP
- ❑ T: 2-bit unsigned integer indicating message type (CON: 0, NON: 1)
- ❑ TKL: Token Length (4 bit field)
- ❑ Code: 8-bit field indicating the code response
- ❑ Message ID: 16-bit field indicating the message identifier
- ❑ Token: carries the actual token

Ref.: CoAP Protocol: Step-by-Step Guide - https://dzone.com/articles/coap-protocol-step-by-step-guide

Our work on CoAP

- Rathod, Vishal, Natasha Jeppu, Samanvita Sastry, Shruti Singala, and Mohit P. Tahiliani. "CoCoA++: Delay Gradient based Congestion Control for Internet of Things." *Future Generation Computer Systems* 100 (2019): 1053-1072.
- Rathod, Vishal J., Sanjana Krishnam, Ayush Kumar, Gauri Baraskar, and Mohit P. Tahiliani. "Effective RTO Estimation using Eifel Retransmission Timer in CoAP." In *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1-6. IEEE, 2020.
- Rathod, Vishal J., and Mohit P. Tahiliani. "Geometric Sequence Technique for Effective RTO Estimation in CoAP." In *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1-6. IEEE, 2020.
- Rathod, Vishal J., and Mohit P. Tahiliani. "Geometric Series based Effective RTO Estimation Technique for CoCoA." *Ad hoc Networks* (2022)

Implementations of CoAP

Several implementations of CoAP

Implementations

CoAP is simple enough to implement from scratch for a simple application.
For applications where that is not desirable, generic implementations are becoming available for a variety of platforms. Many of these implementations stay private, but some are published with liberal open-source licenses such as the Apache 2.0 or the MIT license.

Constrained devices

Implementations for constrained devices are typically written in C.

Erbium

Contiki is a widely used operating system for constrained nodes, being employed for research and product development. Erbium is a full-fledged REST Engine and CoAP Implementation for Contiki.

View details »

libcoap

A C implementation of CoAP that can be used both on constrained devices (running operating systems such as Contiki or LWIP) and on a larger

CoAP is not only used between constrained devices, but also between them and more powerful systems such as cloud servers, home centrals, smartphones:

Server-side

Java

One significant Java-based implementation of CoAP is **Californium**.

View details »

nCoAP is a Java implementation of the CoAP protocol using the Netty NIO client server framework:

View details »

Leshan is an OMA Lightweight M2M (LWM2M) server-side implementation, on top of Californium.

Browser-based

Copper is an extension for Firefox to enable direct access to CoAP resources from a browser.

View details »

Smartphones

Some implementations are specifically targeting mobile devices such as smartphones and tablets. These tend to differ between platforms:

iOS, OSX

A simple iOS client implementation has been written by Wojtek Kordylewski in Objective-C.

View details »

CoAP client and server libraries are also available

Ref.: https://coap.technology/impls.html

Experiment 1:
Emulating CoAP using network namespaces

Experiment 2:
Emulating CoAP using NeST

Thank you!