TECHNISCHE HOCHSCHULE DEGGENDORF

# IAS Spezielle Protokolle des IoT

## Constrained Application Protocol

# Motivation

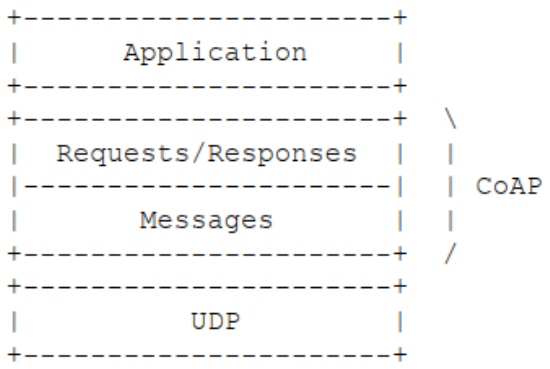
**Constrained Application Protocol (CoAP) ermöglicht:**

- Eine REST ähnliche Architektur mit URI und Content-Type
- Request-Response Mitteilungen mit niedrigem Header-Overhead und niedriger Parsing-Komplexität
- UDP statt TCP wegen geringerem Overhead, aber mit einem eigenen, einfachen Reliability/Acknowledgement layer
- Asynchroner Austausch von Meldungen
- Einfache Implementierung von Proxy und Cache durch Stateless-Design

# CoAP

**Überblick CoAP:**

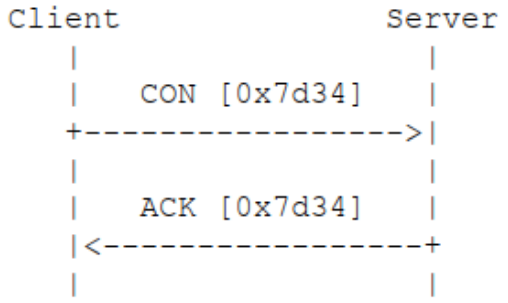
- IETF Standard seit Juni 2014: RFC7252
- Blockweiser Transfer für CoAP Messages: RFC7959
- CoAP ist Teil des Constrained RESTful Environmens (CoRE)
- CoAP liegt zwischen UDP und der Applikation
- CoAP lehnt sich sehr stark an das REST Protokoll an, verwendet aber UDP für die Datenübertragung
- Unterstützung von Proxy Funktionalitäten
  - innerhalb von CoAP (CoAP-to-CoAP Proxy mit Caching oder Namenspace-Translation)
  - Von anderen Protokollen zu CoAP (Cross-Proxy, z.B. CoAP-to-HTTP)
- Unterstützung von Resource Discovery (Service Discovery)
- COAP nutzt URIs ähnlich zu REST:
  "coap:" "//" host [ ":" port ] path-abempty [ "?" query ]
  diese URIs können im Protokoll aufgespalten werden in: Host, Port, Path (Pfad zur Ressource) und Query (Parameter)



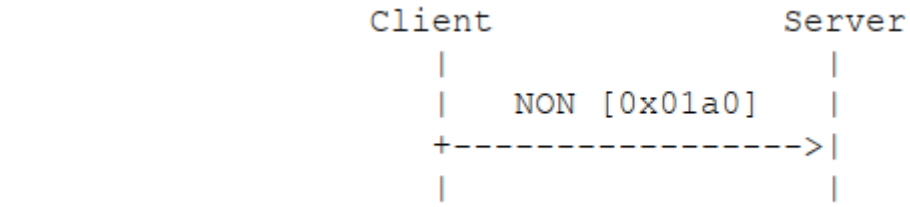
# CoAP

**Message Typen**

- Reliable Messages (Confirmable, CON): Messages, die ein Acknowledge erfordern (oder nach einem Timeout wiederholt werden), der Server kann statt ACK mit RST (Reset) antworten, um Ablehnung des Pakets zu signalisieren



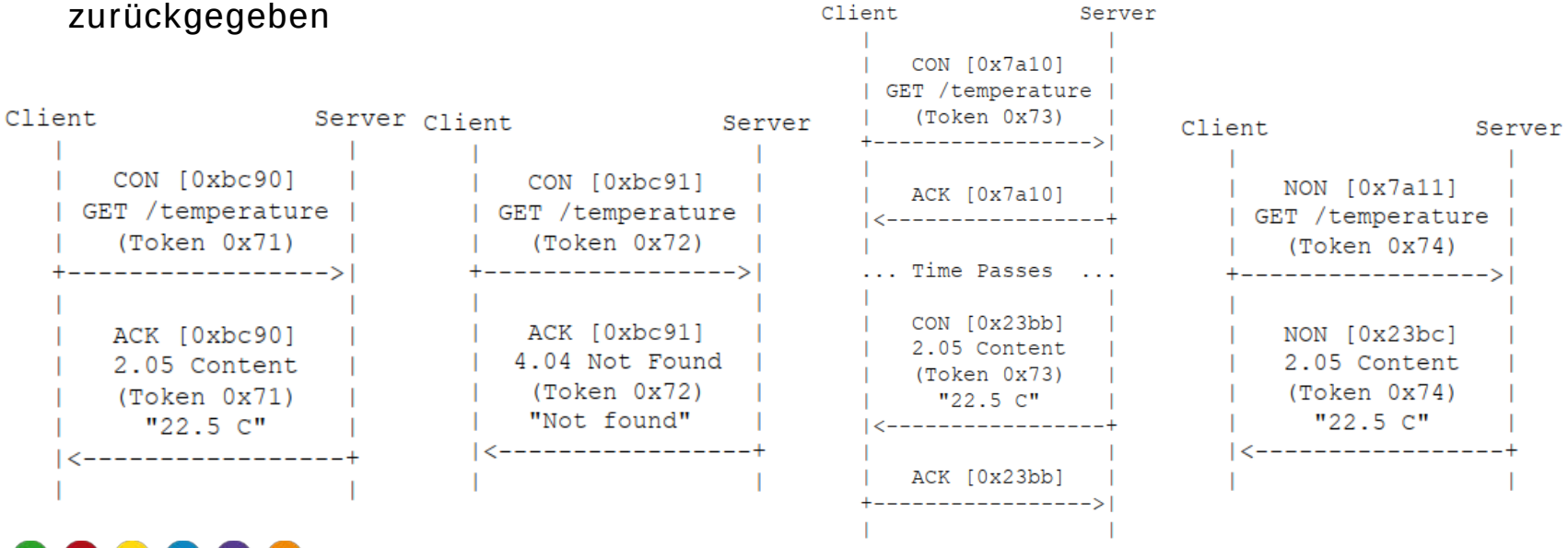

- Unreliable Messages (Non-Confirmable, NON):Nachrichten, die kein Acknowledgement erfordern



# CoAP

**Request / Response (Anfrage / Antwort) Verhalten**

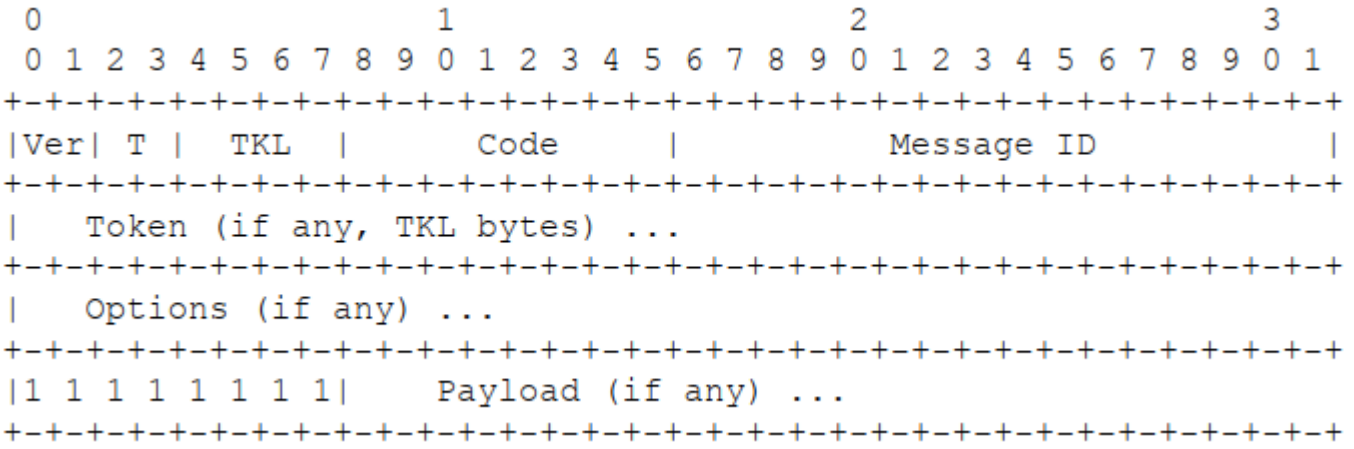
- Es gibt 3 Arten von Request/Response Möglichkeiten
  - Confirmable GET gesendet, kombiniertes ACK/RESPONSE empfangen
  - Confirmable GET gesendet, erst ACK, dann REPSONSE empfangen und mit ACK bestätigt
  - Non-Confirmable GET gesendet, RESPONSE empfangen
- Bei einer nicht vorhandenen Ressource werden HTTP-typische Fehlercodes zurückgegeben



# CoAP

**Paket-Header bestehend aus:**

- 2 Bit Version (aktuell: 01)
- 2 Bit Typ: Confirmable (0), Non-confirmable (1), Acknowledgement (2), Reset (3)
- 4 Bit Token Länge: Das Token ordnet Anfragen und Antworten einander zu und kann 0-8 Byte lang sein
- 8 Bit Request/Response Code: 3Bit für Hauptthema der Anfrage, 5 Bit für Unterthema, z.B. 4.04 für eine « Resource not found » Antwort
- 16 Bit Message-ID: Erkennung von Duplizierung von Messages (z.B. durch verlorengeganges ACK) und Zuordnung von ACK/RST zu CON/NON



# CoAP

**Header-Informationen (Fortsetzung):**

- Optionen können genutzt werden, um spezielle Protokolleinstellungen vorzunehmen, z.B. um URI-Host, URI-Port, URI-PATH und URI-Query Parameter für eine Anfrage an einen HTTP-Host zu übertragen oder einen zwischengeschalteten Proxy zu konfigurieren
  - Die Übertragung von Optionen erfolgt mit einem effizienten differentiellen Signalisierungsformat

# CoAP

**Anfragen (Requests):**

- Eine CoAP Anfrage enthält
  - eine Methode (GET, POST, PUT, DELETE)
  - die Nennung einer Ressource auf die die Methode angewendet werden soll (URI)
  - eine Payload
- Optional kann eine Anfrage enthalten:
  - einen Internet Medientyp (z.B. « application/XML »)
  - Metadaten für die Anfragen

**Antworten (Responses):**

- Eine Antwort enthält:
  - Einen Antwort Code (4.04, 2.00, …), 3 Bit Major-Code, 5 Bit Minor-Code
  - Die Antwort auf die Anfrage

# CoAP

**Proxy**

- Unterscheidung nach Art des Protokolls: CoAP-to-CoAP oder Cross-Protocol
- Unterscheidung, ob Informationen zwischengespeichert werden oder nicht: Caching Proxy, Non-Caching Proxy
- Unterscheidung nach Selbstständigkeit des Proxy: Weiterleitung zum spezifizierten Server (Forward-Proxy) oder eigene Resource-Datenbank, vorkonfiguriert oder durch Resource-Discovery (Reverse-Proxy)

- Ein Caching-Proxy reduziert die Netzwerklast, indem er bei der ersten Anfrage eines Clients, den zugehörigen Server anfragt, dann aber (bis zum Erreichen der max-age der Server-Antwort) aus seinem Speicher antwortet

# CoAP

**Service und Ressource Discovery:**

- CoAP Server können mit dedizierter Adressierung (Unicast) oder über Multicast gefunden werden
  Default Port-Nummern: 5683 bzw. 5684 (für DTLS-Verschlüsselung)
- Der Pfad zur Resource Beschreibung ist « /.well-knwon/core » (GET)
- Query Strings können genutzt werden, um die gewünschte Auswahl einzuschränken
- Die Antwort beinhaltet eine Liste von Ressourcen mit:
  - Pfad
  - Titel
  - Medientyp
  - Ressourcen-Typ rt
    (z.B. Outdoor-Temperature)
  - Interface-Typ if
- Mehr Informationen in RFC6690 und RFC2045

```
REQ: GET /.well-known/core

RES: 2.05 Content
</sensors>;ct=40

REQ: GET /sensors

RES: 2.05 Content
</sensors/temp>;rt="temperature-c";if="sensor",
</sensors/light>;rt="light-lux";if="sensor"

An example query filter may look like:

REQ: GET /.well-known/core?rt=light-lux

RES: 2.05 Content
</sensors/light>;rt="light-lux";if="sensor"
```

# CoAP

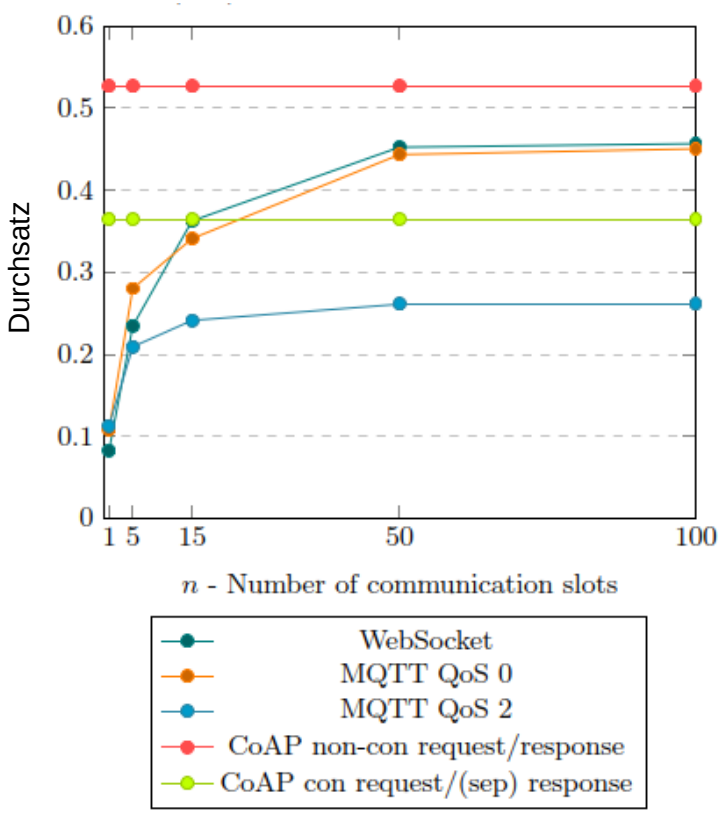
**Vergleich CoAP / MQTT / Websockets**

- Unterschiedliche Szenarien:
  - CoAP: Request/Response (Pull), Server (Sensor) muss immer antworten können
  - MQTT: Publish/Subscribe (Push), Entkopplung des Sensors von der Verarbeitung aber stark asynchron
  - Websockets: Pull-Push Austausch, aber Sensor dauerhaft verbunden

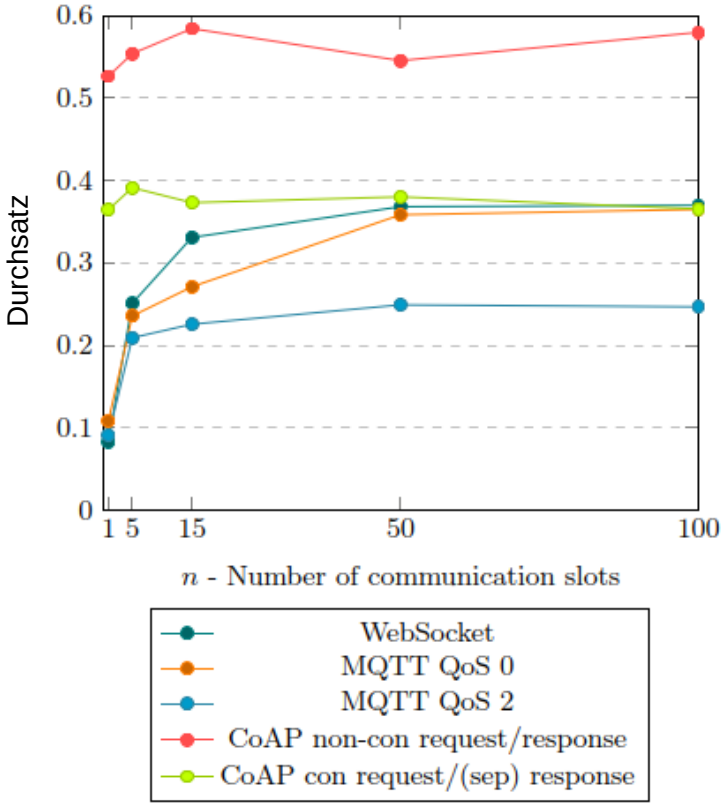
# CoAP

**Vergleich CoAP / MQTT / Websockets: Performance**

- Ohne Paketverlust




mit 20% Paketverlust




[Sarafov, Seeger, « Comparison of IOT Data Protocol Overhead », Seminars FI / IITM WS 17/18, Technical University of Munich]

# CoAP

**Zusammenfassung:**

- CoAP ist eine Request/Response Infrastruktur für IoT
- Stark an REST orientiert, weniger Protokoll-Overhead durch Verzicht auf TCP und HTTP
- Durch Proxy kann Anbindung an REST-Umgebung erfolgen
- Service- und Ressource-Discovery spezifiziert

**Quellen:**

- Soweit nicht anders angegeben:
  Z. Shelby, K. Hartke, C. Bormann, « The Constrained Application Protocol (CoAP) »,
  Request for Comments 7252, Internet Engineering Task Force, Juni 2014,
  ISSN: 2070-1721

TECHNISCHE HOCHSCHULE DEGGENDORF

**Danke für Ihre Aufmerksamkeit!**