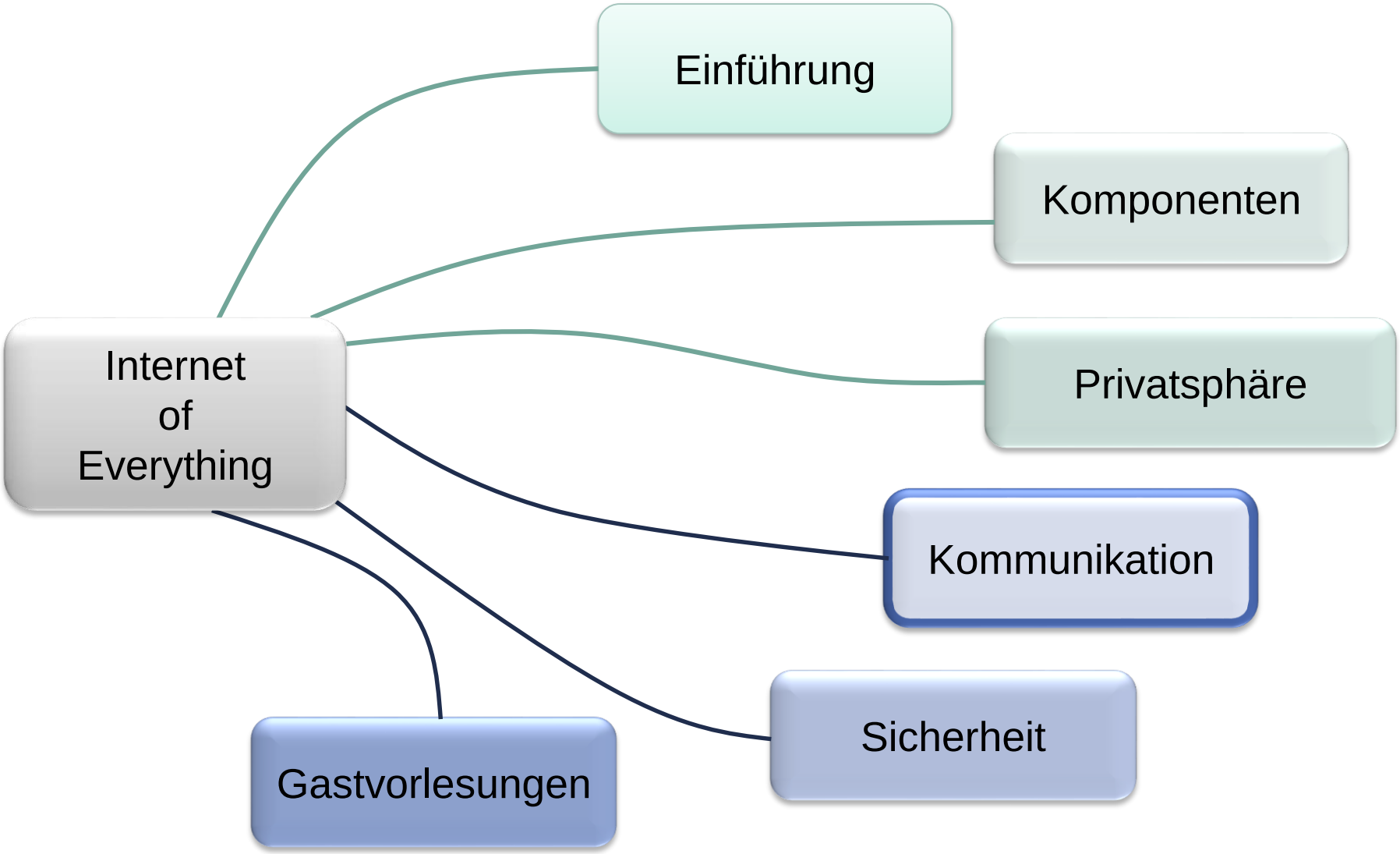
Karlsruher Institut für Technologie

Vorlesung Internet of Everything
Wintersemester 2017/18

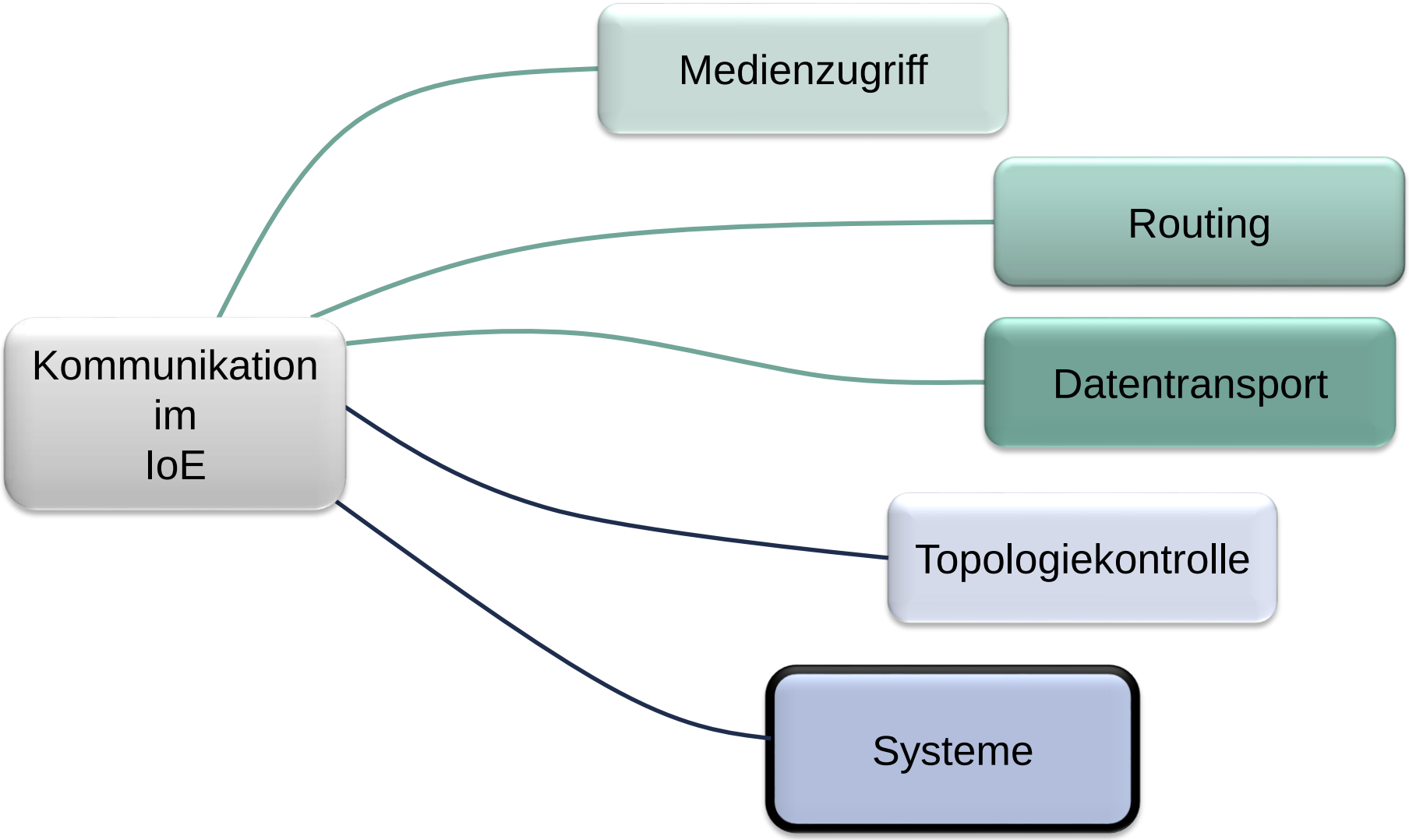
Kapitel 4.5 – Systeme

Institut für Telematik, Prof. Zitterbart

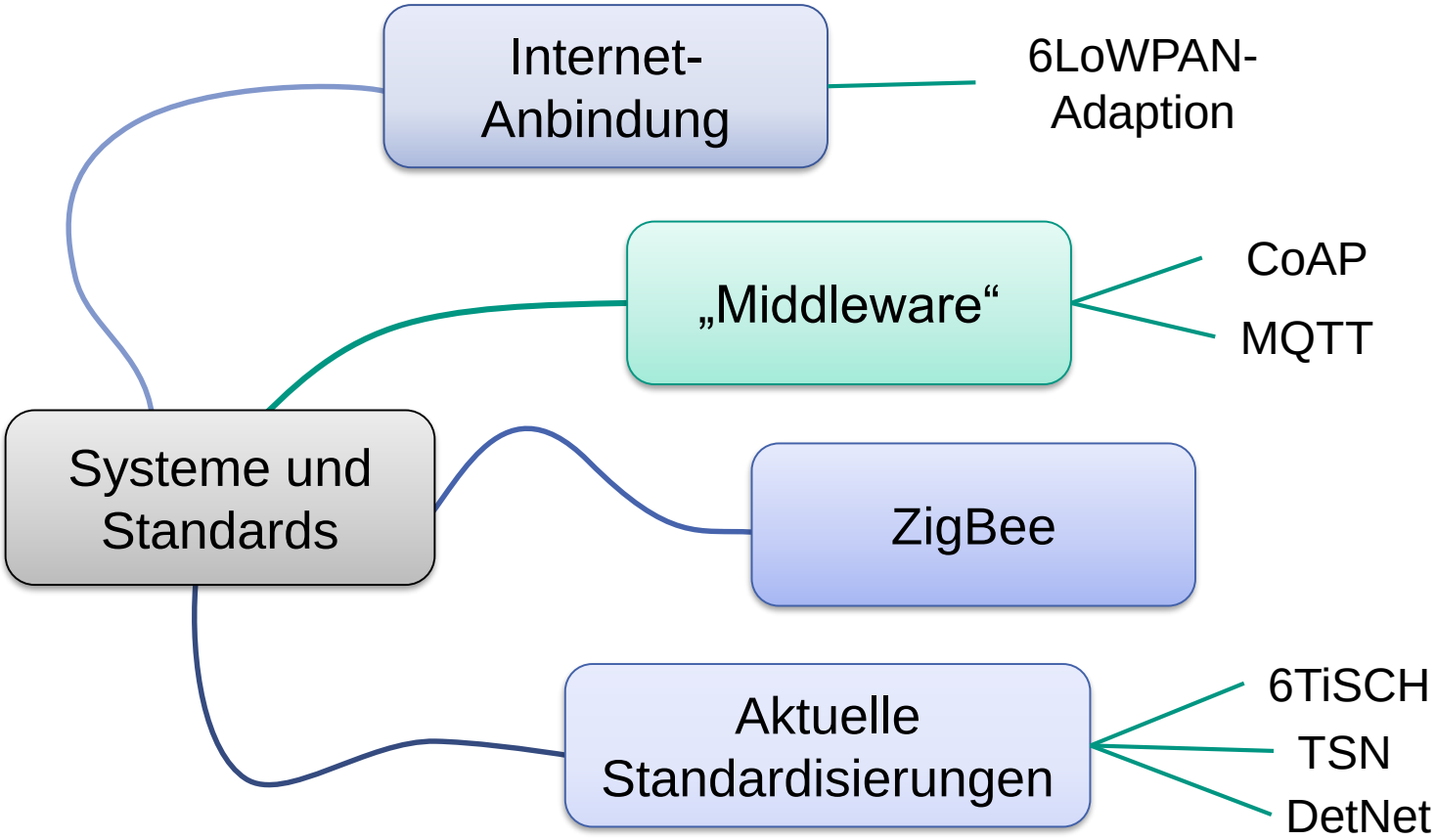
Inhalte der Vorlesung



Kapitel 4.5: Systeme



Schwerpunkte des Kapitels im Überblick

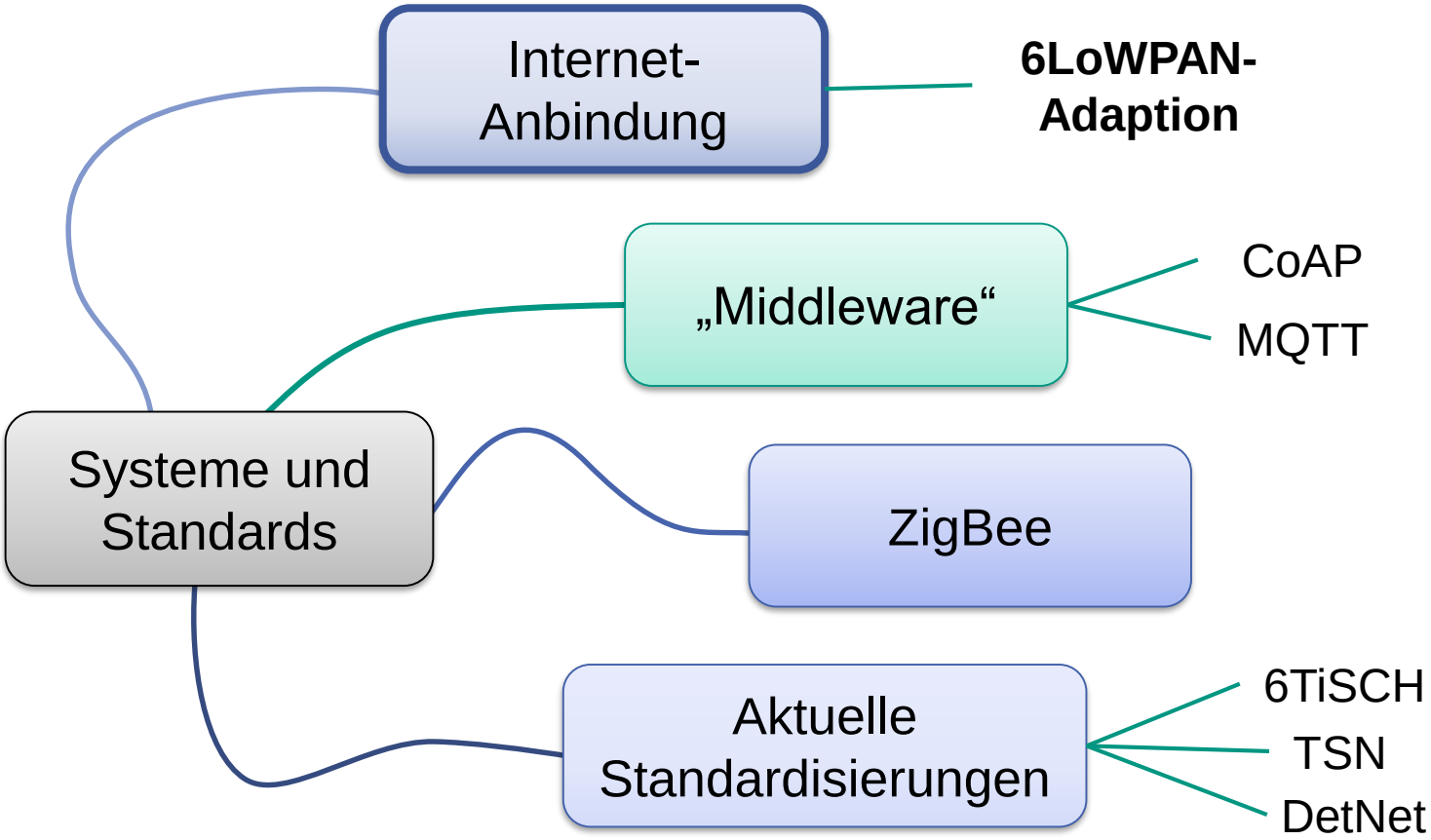


Systeme / Standardisierung

- Im Folgenden Fokus eher auf Systemen und Standardisierung und den damit assoziierten Protokolle
  - (1) Verbindung mit dem Internet → Internet of Things
    - 6LoWPAN zur IPv6-Konnektivität
  - (2) Anwendungsprotokolle/Middleware für das Internet der Dinge
    - CoAP zur Unterstützung für Client/Server-Anwendungen
    - MQTT für Publish/Subscribe-Dienste
  - (3) ZigBee: Protokollstapel und Geräteprofile u.A. für Hausautomation
  - (4) Automatisierungsanwendungen und Industrial Internet (Industrie 4.0)
    - OPC UA/IEC 61850: Herstellerübergreifende Protokollstapel für industrielle Anwendungen/Smart Grids, selbstbeschreibende Geräte und Daten
  - (5) Aktuelle Standardisierung
    a) WPANs (Wireless Personal Area Networks)
      - 6TiSCH: Zusammenarbeit von IEEE 802.15.4e mit 6LoWPAN

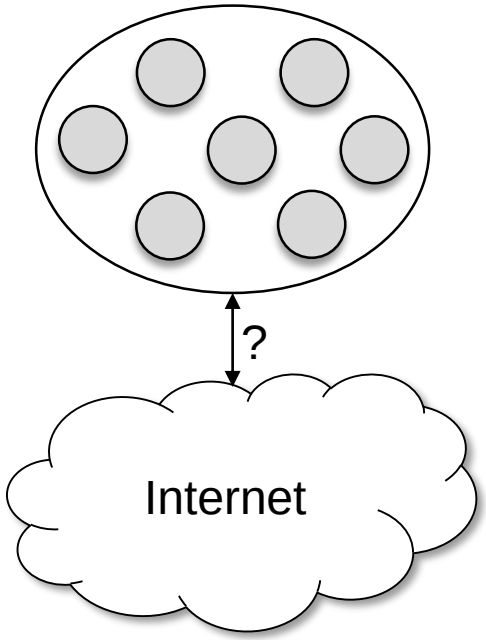
    b) LANs (Local Area Networks)
      - Time-Sensitive Networking
      - DetNet Arbeitsgruppe der IETF

Schwerpunkte des Kapitels im Überblick



Verbindung mit dem Internet

- Wie finden „Things“ ins Internet?



- Bisher diskutiert: Kommunikation im Funknetz
  - Medienzugriff
  - Routing
  - Transport

- Und weiter?
  - Globale Adressierung?
  - Interoperabilität mit dem „normalen“ Internet?
  - Interaktion mit anderen, entfernten, Diensten und Systemen?
    - Smartphone-Apps, Cloud-Services …

Verbindung mit dem Internet

- Drei wesentliche Standards der IETF

  - RPL
    - Routing im LoWPAN (Low-Power Wireless Personal Area Network)
  - 6LoWPAN
    - Konnektivität von LoWPANs zum Internet
    - Komprimierung und Fragmentierung von IPv6-Dateneinheiten
  - CoAP
    - Nutzung Client/Server-basierter Systeme

[Pal+13]

6LoWPAN

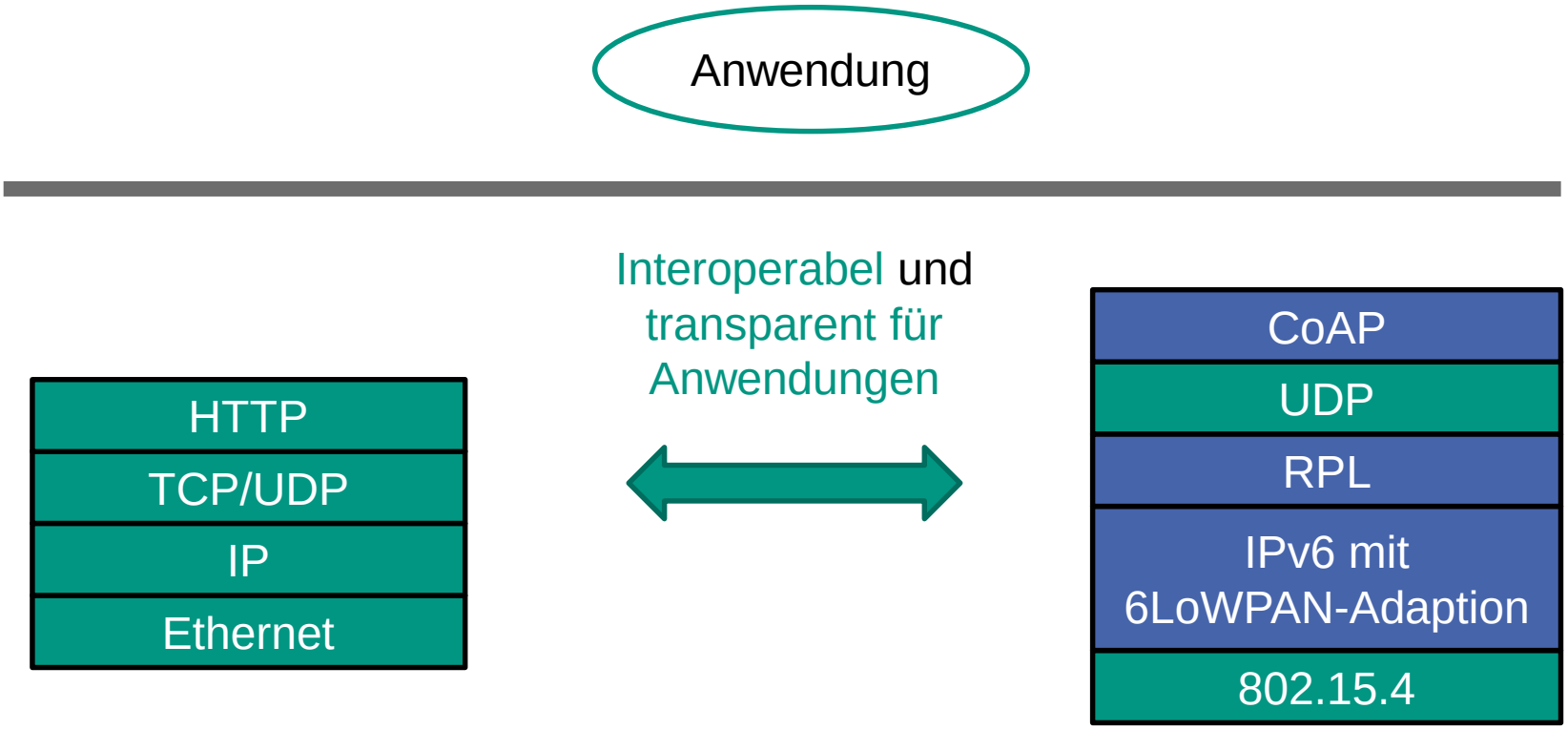
- 6LoWPAN: IPv6 over Low-Power Wireless Personal Area Networks
  - Nutzung von IPv6 in Umgebungen mit stark eingeschränkten Ressourcen
- Potenziell sehr viele vernetzte Systeme
  - IPv4 Adressraum reicht nicht aus
  - Nutzung von IPv6 notwendig
- Systeme haben global eindeutige IPv6 Adresse
  - Damit alle Systeme global erreichbar
  - Kein Gateway erforderlich
  - Ende-zu-Ende Kontext bleibt erhalten
- Nutzung von IPv6 Autokonfigurationsmechanismen
  - Erspart Konfigurationsoverhead
- Nutzung von etablierten Managementprotokollen
- Kein Anpassen von bestehenden Anwendungen erforderlich
  - UDP kann als Transportprotokoll verwendet werden

[RFC4944, RFC6282]

Schichtenmodell im Kontext von 6LoWPAN

„Klassisches“ Schichtenmodell im Internet

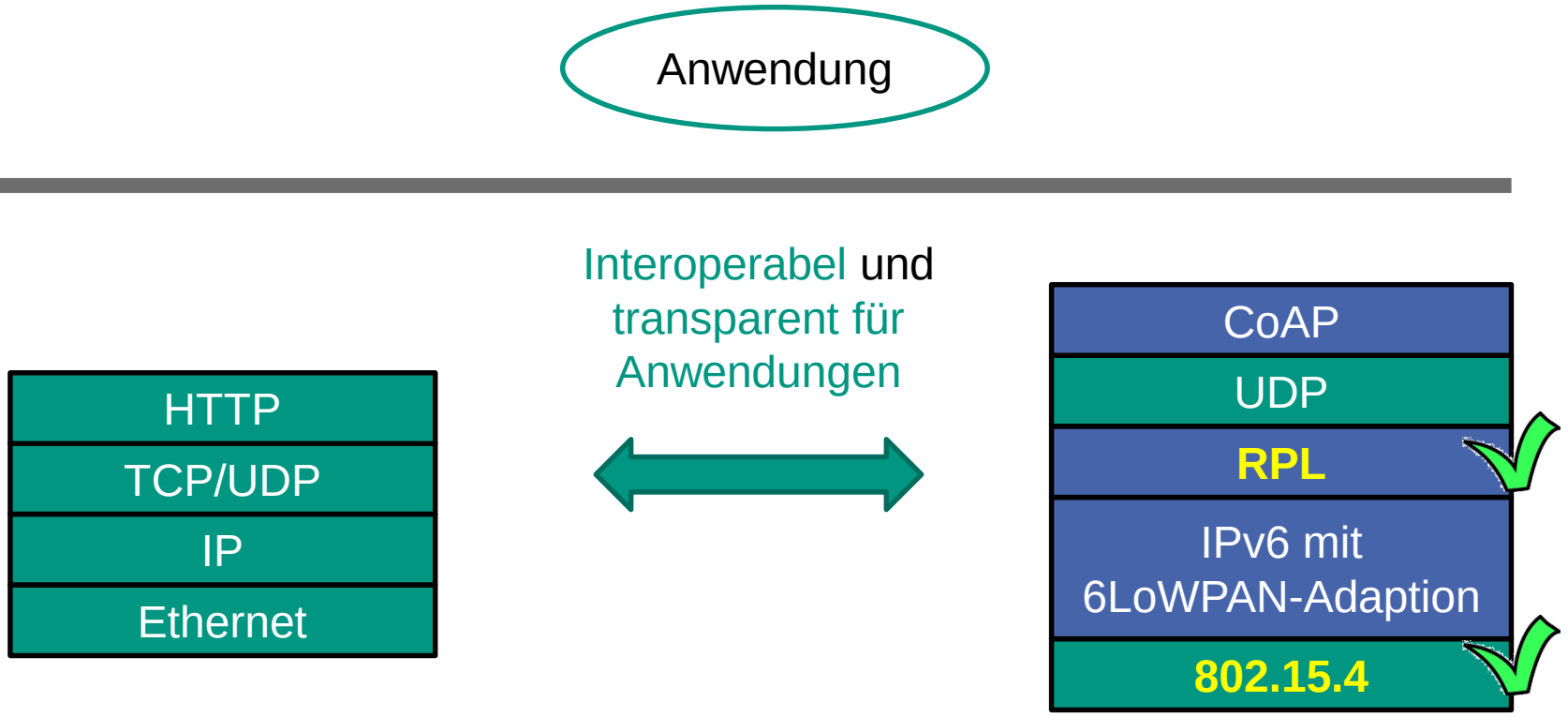
Schichtenmodell von 6LoWPAN



… bereits betrachtet

„Klassisches“ Schichtenmodell im Internet

Schichtenmodell von 6LoWPAN



Aufgaben im Protokollstapel: Überblick

- 802.15.4
  - Medienzugriffsprotokoll
  - Verschiedene Betriebsmodi möglich
  - *… andere Protokolle sind möglich, im Folgenden aber nicht betrachtet*
    - Bluetooth Low Energy, NFC …
- RPL
  - Routing, Distanz-Vektor-basiert
  - Fokus: Concast und Multicast; Punkt-zu-Punkt möglich
- 6LoWPAN-Adaption
  - Komprimierung der Paketgröße, Fragmentierung …
- CoAP
  - Constrained Application Protocol
  - Transferprotokoll für Umgebungen mit stark limitierten Ressourcen
  - RESTful-basiertes Protokoll, vergleichbar mit HTTP

Annahmen und Anforderungen

- Anwendungen senden kleine Datenmengen
  - Typischerweise < 100 Byte
- 6LoWPANs basieren auf 802.15.4
  - Inzwischen auch: IPv6 over Bluetooth Low Energy, NFC und andere
- Geräte sind in Leistungsfähigkeit extrem limitiert und günstig
- Geräte sind normalerweise batteriebetrieben
  - Daher häufiges Schlafen um Energie zu sparen
- Ausbringung der Netze ad-hoc, Lokation nicht vordefiniert
  - Verschiedene Topologien möglich: Mesh, Star, …
  - Anzahl der Geräte in 6LoWPANs tendenziell hoch

  →Selbstorganisation notwendig
- Konfigurations- und Managementbedarf muss minimal sein
  - Systeme u.U. schwer zu erreichen

[RFC4919]

Aufgaben der 6LoWPAN-Adaption

- Ziel: Herstellen von IP-(Inter-)Konnektivität
  - Dabei: Nutzung bereits etablierter IPv6-Mechanismen

- Adressierung → IPv6
- Adaption an *link MTU* (IEEE 802.15.4: < 127 Byte) → 6LoWPAN
  - Fragmentierung und Reassemblierung
  - Header-Compression reduziert Overhead
- Autokonfiguration → 6LoWPAN-ND
  - Protokolle müssen einfach zu konfigurieren sein
  - Bootstrapping von neuen Netzen soll möglichst einfach sein
  - „Selbstheilung“ muss unterstützt werden

Überblick Standardisierung

- Relevante RFCs
  - Überblick über Anforderungen und Annahmen aus RFC4919
    - *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*

[RFC4919]

  - 6LoWPAN Adaptionsschicht → RFC4944
    - *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*

[RFC4944]

  - Header Compression → RFC6282
    - *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*

[RFC6282]

- Im Folgenden nicht behandelt
  - Neighbor Discovery (ND) → RFC6775
    - *Neighbor Discovery Optimization for Low Power and Lossy Networks (6LoWPAN)*

[RFC6775]

Standardisierung

- 6LoWPAN wird in RFC 4944 standardisiert
  - Transmission of IPv6 Packets over 802.15.4 Networks
    - Transparent für darüber liegende Protokolle
    - Anpassung und Optimierung ohne Verlust an Funktionalität

  - *“This document describes*
    - *The frame format for transmission of IPv6 packets*
    - *The method of forming IPv6 link-local addresses and statelessly autoconfigured addresses on IEEE 802.15.4 networks*
    - *Additional specifications include a simple header compression scheme using shared context and provisions for packet delivery in IEEE 802.15.4 meshes”*

[RFC4944]

Überblick 6LoWPAN-Adaption

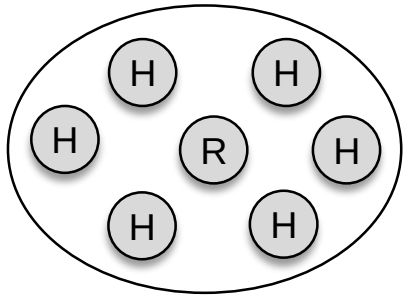
- Architektur

- Adressierung
  - Autokonfiguration für IPv6 über IEEE 802.15.4

- 6LoWPAN-Frame-Format
  - Fragmentierung
  - Header-Compression
    - IPv6 Header Compression
    - UDP Header Compression
    - …

6LoWPAN Architektur

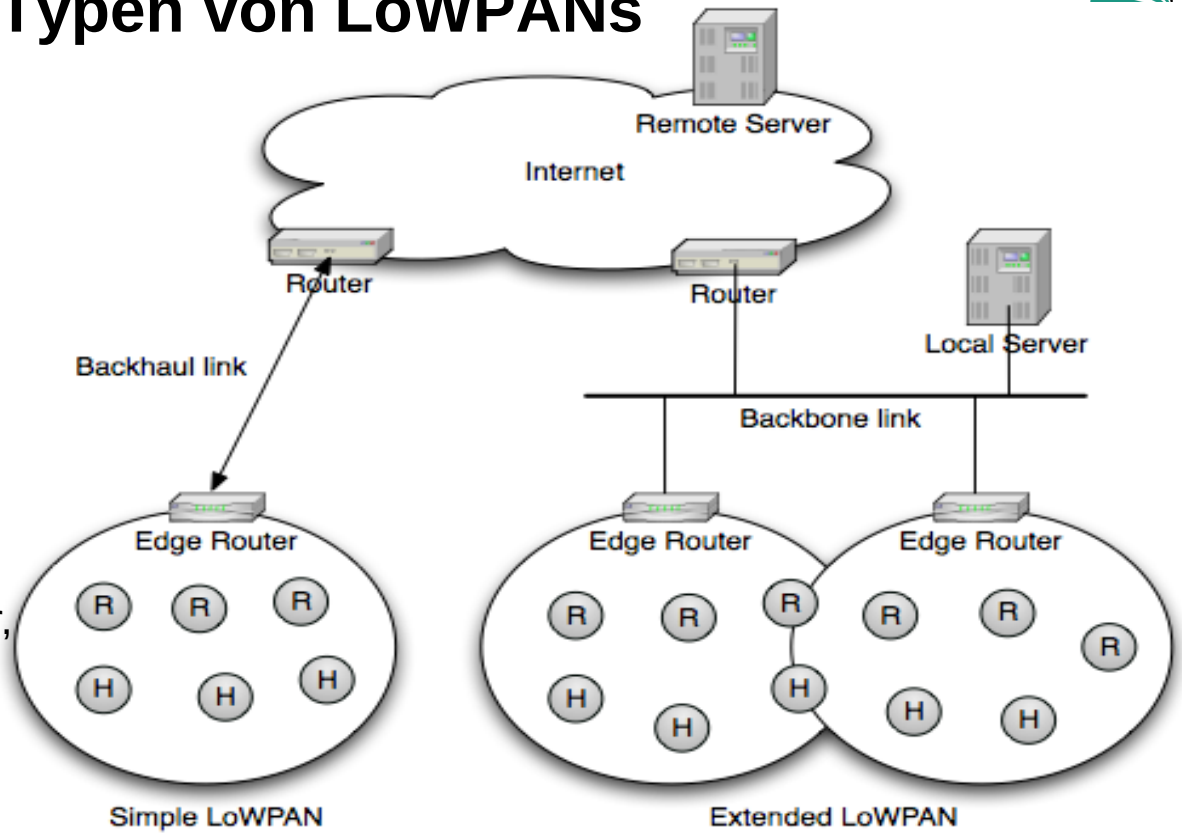
- Grundlegendes Konzept
  - Verbindung zwischen „Inseln“ mit drahtlosen eingebetteten Systeme herstellen
  - Jede „Insel“ ist Stub-Netz im Internet
    - Enthält Quellen und Senken von Daten
    - Fungiert nicht als Transit-Netz für andere Quellen und Senken



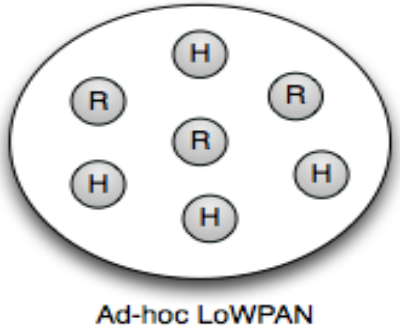
- LoWPAN
  - „Insel“ mit Geräten mit gemeinsamem IP-Addresspräfix

- Zwei Typen von Geräten in einem LoWPAN werden unterschieden
  - Router
    - Führen auch Routingaufgaben aus
  - Host
    - Quellen und/oder Senken von Daten
    - Sind nicht in das Routing involviert

Verschiedene Typen von LoWPANs

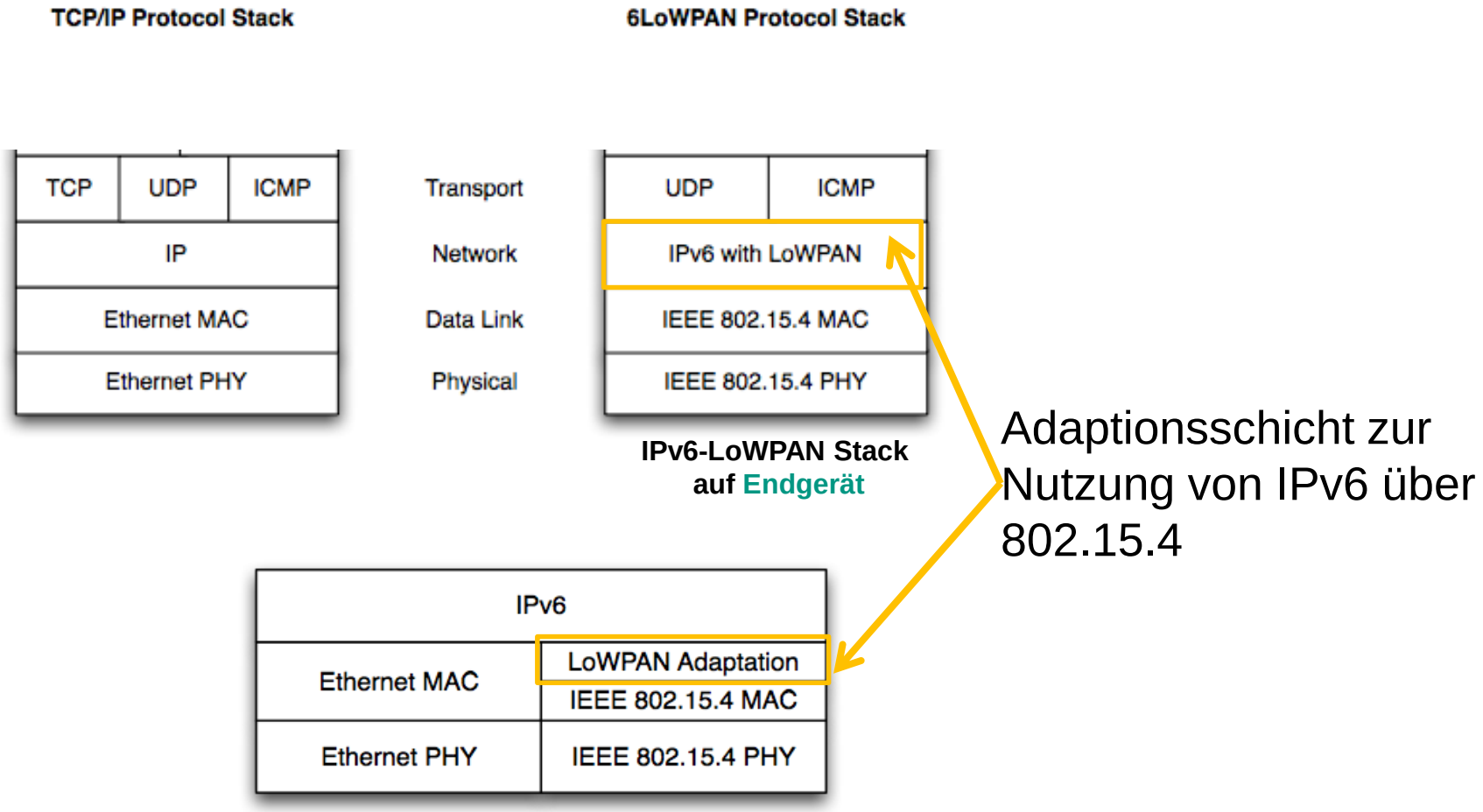


- Simple LoWPAN
  - Ein Edge Router, ein Link zu anderen IP-Netzen
- Extended LoWPAN
  - Mehrere Edge Router, gemeinsamer Backbone Link
- Ad-hoc LoWPAN
  - Kein Edge Router, keine Infrastruktur



[ShBo09]

Einordnung der 6LoWPAN-Adaptionsschicht



[ShBo09]

Überblick 6LoWPAN-Adaption

- Architektur
- Adressierung
  - Autokonfiguration für IPv6 über IEEE 802.15.4
- 6LoWPAN-Frame-Format
  - Fragmentierung
  - Header-Compression
    - IPv6 Header Compression
    - UDP Header Compression
    - …

Adressierung

- Aufgabe
  - Abbildung von IPv6 Adressen auf IEEE 802.15.4 Adressen
- Annahme in RFC4944
  - LoWPAN wird auf einen IPv6 Link abgebildet
    - Alle Knoten eines LoWPANs bilden also einen IPv6 Link
    - Alle Knoten eines LoWPANs haben gleichen Adresspräfix
  - Unterstützung von Link-Layer Broadcast
  - Abbildung von IPv6 Multicast Adressen auf Link-Layer Broadcast Adresse
    - In jedem MAC-Frame PAN-ID und Link-Layer Broadcast-Adresse (0xffff) enthalten

IPv6-Adressen

- Folgende Typen von Adressen bei IPv6 möglich
  - Unicast
    - Identifikator für ein einzelnes Interface
  - Anycast
    - Identifikator für eine Menge von Interfaces
    - Dateneinheit mit einer solchen Adresse wird an ein Interface aus dieser Menge ausgeliefert (üblicherweise das „nächstgelegene“)
  - Multicast
    - Identifikator für eine Menge von Interfaces
    - Dateneinheit mit solcher Adresse wird an alle Interfaces aus dieser Menge ausgeliefert

[RFC4291]

IPv6-Adressen

- Adresstyp wird durch führende Bits einer Adresse festgelegt
- Beispiele

	Führende Bits	als Präfix
Multicast	11111111	ff00::/8
Link-Local Unicast	1111111010	fe80::/10
Global Unicast	(alles übrige)	

  - Adresse innerhalb abgeschlossener Netzwerksegmente
  - Beispielsweise 0:0:0:0:0:ffff::/96 für IPv4 Adressen abgebildet auf IPv6 Adressen

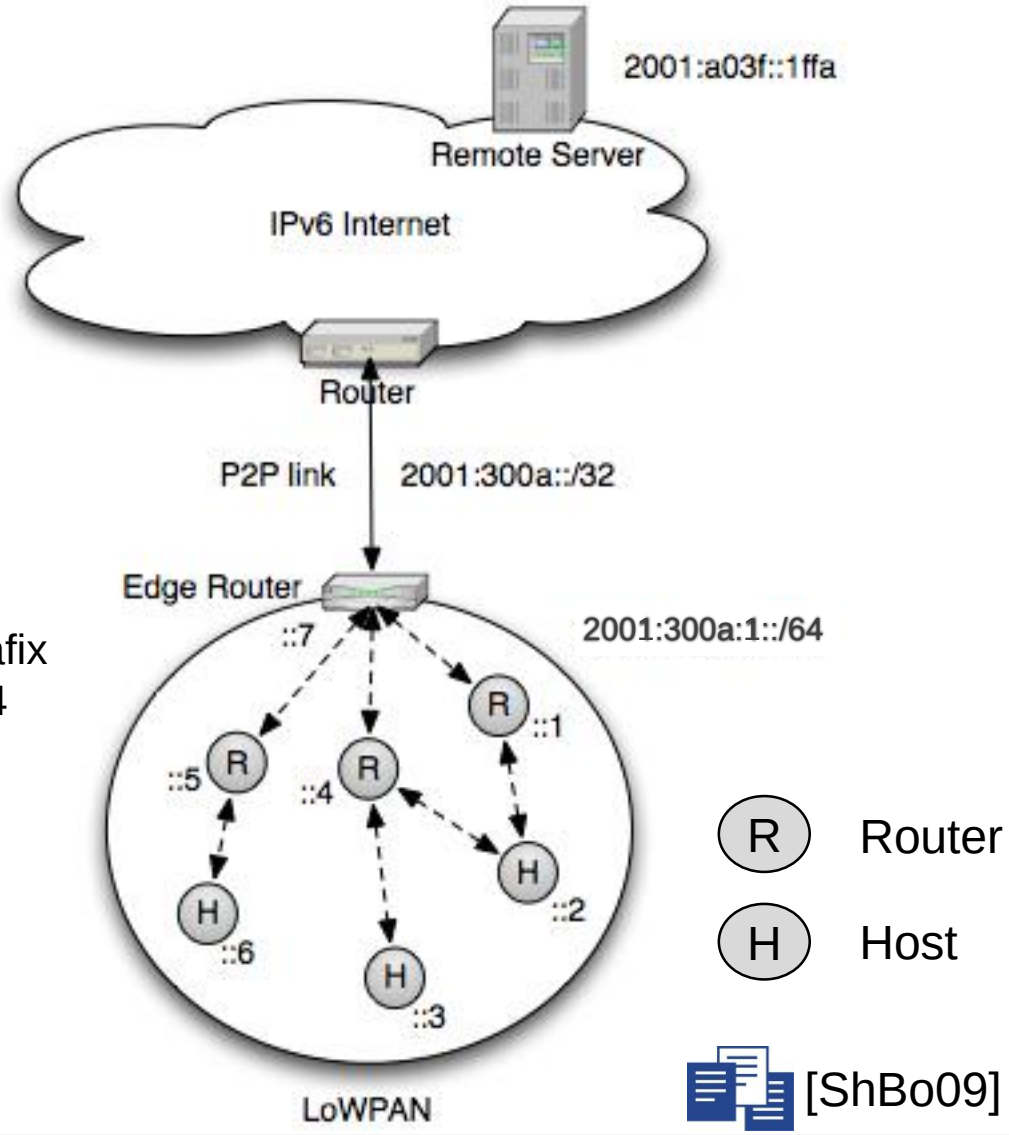
Adressen bei Nutzung von IEEE 802.15.4

- 802.15.4 bietet unterschiedliche Adressierungsmodi
  - IEEE 64-bit Adressen
    - MAC-Adresse des drahtlos angebundenen Geräts
  - 16-bit (short) Adressen, eindeutig im PAN
    - Vergeben durch PAN Koordinator
    - Vergleiche 802.15.4 im Non-Beacon Mode

- Berechnung der IPv6 Adressen in 6LoWPAN
  - Hier relevant: IPv6-Adressen mit 64 Bit Präfix und 64 Bit Interface Identifier
  - 6LoWPAN nutzt *Stateless Address Autoconfiguration*
    - Berechnung der IPv6 Adresse aus
      - Präfix des LoWPAN und
      - Interface Identifier des drahtlos angebundenen Geräts

Beispiel für Simple LoWPAN Netzkonfiguration

- Simple LoWPAN Netz
  - Ein Edge Router
  - Drei LoWPAN Router
  - Drei LoWPAN Hosts
  - Nutzt 802.15.4
- Adresskonfiguration
  - Router gibt IPv6 Präfix 2001:300a::/32 bekannt
  - Edge Router konfiguriert IPv6 Präfix 2001:300a:1::/64 für das 802.15.4 Interface
  - Edge Router gibt dieses Präfix im LoWPAN bekannt
    - Router konfigurieren Adressen
    - z.B. Addresse ::1 (Interface ID)
    - Komplette IPv6 Adresse: 2001:300a:1::1

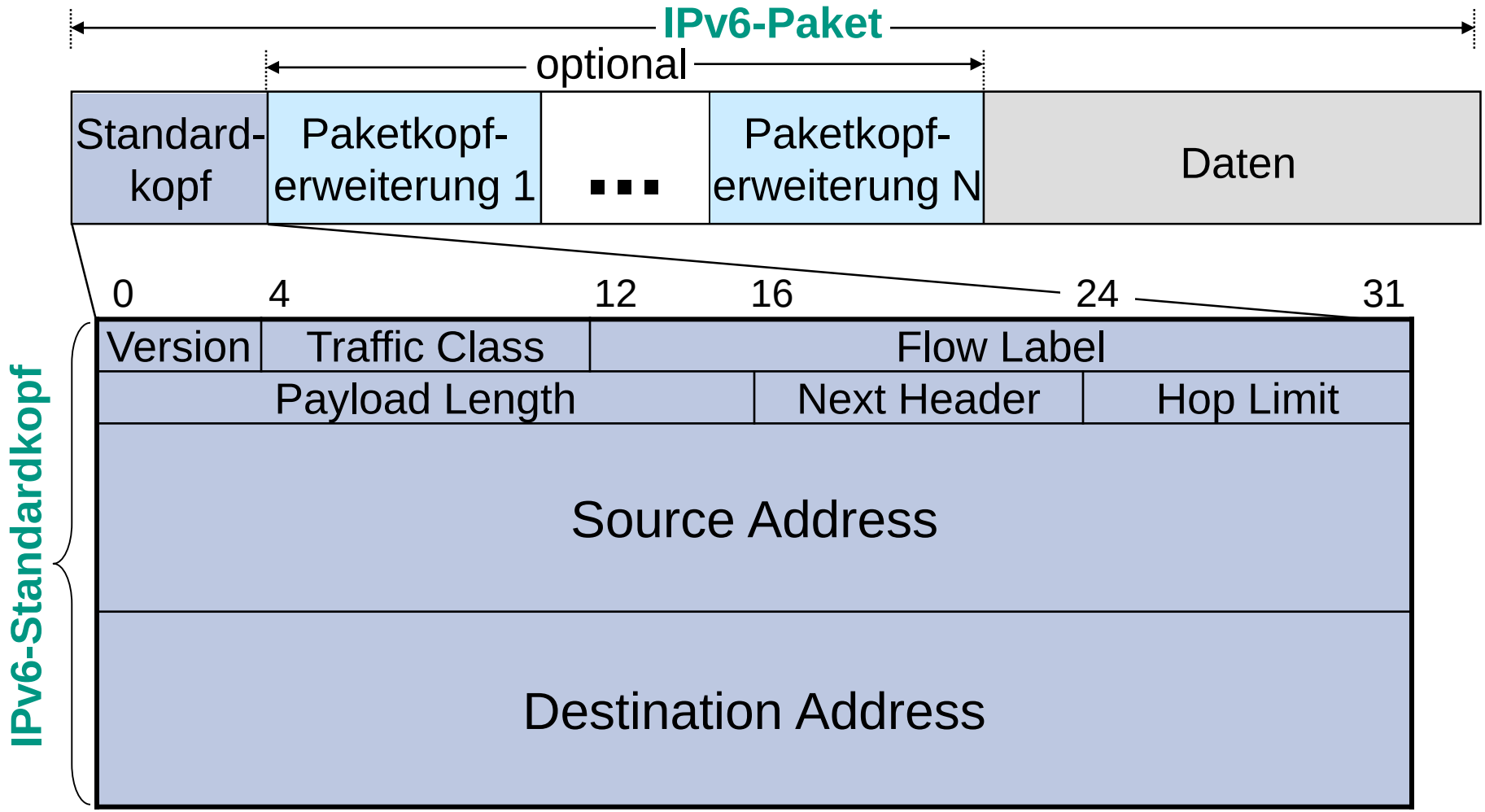


[ShBo09]

Überblick 6LoWPAN-Adaption

- Architektur
- Adressierung
  - Autokonfiguration für IPv6 über IEEE 802.15.4
- 6LoWPAN-Frame-Format
  - Fragmentierung
  - Header-Compression
    - IPv6 Header Compression
    - UDP Header Compression
    - …

IPv6-Dateneinheit



[RFC2460]

Randbedingungen

- Anforderung von IPv6
  - Minimale MTU auf einem Link beträgt 1280 Byte
- Aber bei 802.15.4
  - Maximale Größe einer Dateneinheit: 127 Byte

→Fragmentierung in der Adaptionsschicht notwendig

- 802.15.4 Dateneinheit beinhaltet maximal 102 Byte Nutzdaten
  - Rest für Header notwendig
- 802.15.4 bietet keine Fragmentierung an

→Header Compression wünschenswert

- IPv6-Kopf benötigt 40 Byte, UDP-Kopf 8 Byte
  - Also im schlechtesten Fall 32 Byte für Nutzdaten übrig
  - Teilweise Informationen redundant in Paketköpfen enthalten
  - Effizientere Codierung wünschenswert

6LoWPAN Frame-Format und Paketköpfe

- 6LoWPAN spezifiziert eigene Paketköpfe für LoWPAN-spezifische Metadaten und die Einbettung der IPv6-Dateneinheit in 802.15.4-Rahmen

```
 0 1
+-+-+------------------------+---------+
|A B| 6LoWPAN AB-type header | Payload |
+-+-+------------------------+---------+
```

- Werte für A B bestimmen den Paketkopf-Typ, unterteilt in 4 Klassen
  - 00 → Kein LoWPAN Paketkopf
  - 01 → Dispatch-Header, siehe Komprimierung
  - 10 → Mesh Paketkopf, für Schicht-2-Routing
  - 11 → Fragmentierungsheader, siehe Fragmentierung

- 6LoWPAN-Paketköpfe können verkettet werden

```
+-----+----------+-----+--------------+----+----------+---------+---------+
| 1 0 | Mesh-Hdr | 1 1 | Fragment-Hdr | 01 | HC1-Disp | HC1-Hdr | Payload |
+-----+----------+-----+--------------+----+----------+---------+---------+
```

  - Mesh-Header + Fragment-Header + Header-Compression (HC1)

Fragmentierung

- Fragmentierungskopf

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1|0 0 0| Datagram_size       | Datagram_tag                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Erstes Fragment

  - *Datagram_tag* zur Unterscheidung der Dateneinheiten bei Reassemblierung
    - Nutzung in Kombination mit 802.15.4 Sender/Empfänger Adresse und der *Datagram_size*
  - Datagram_size: Größe der gesamten Dateneinheit nach Reassemblierung

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1|1 0 0| Datagram_size       | Datagram_tag                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Datagram_offset|
+-+-+-+-+-+-+-+-+
```

Weitere Fragmente

  - *Datagram_offset* zur Berechnung der restlichen Daten
    - Kann nur Positionen die Vielfache von 8 Bytes sind beschreiben

Header Compression

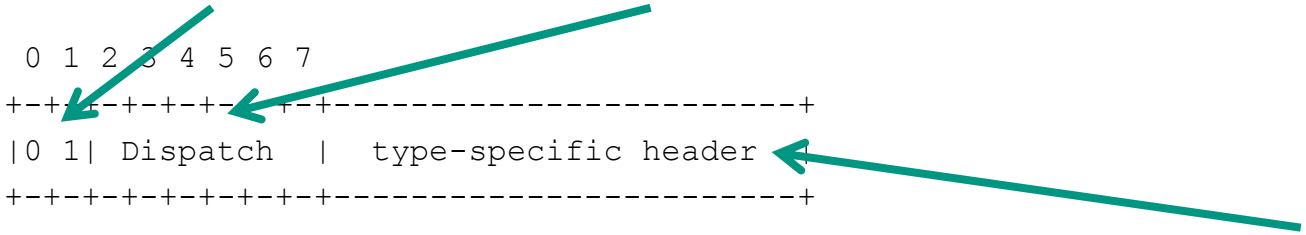
- Grundlegende Techniken zur 6LoWPAN Header Compression
  - Unnütze, weil redundante Informationen vermeiden
  - Felder mit allgemeinen Informationen weglassen
  - Hop-by-Hop Header Compression

- RFC4944 definiert zwei Techniken zur Header Compression
  - LOWPAN_HC1: Komprimierung des IPv6-Kopfs
  - LOWPAN_HC2: Komprimierung des UDP-Kopfs
  - Zustandslose Komprimierung
    - Sender und Empfänger benötigen keinen gemeinsamen Kontext

- Erweiterungen, um effizientere, zustandsbehaftete Komprimierungsmechanismen existieren
  - *RFC6282: Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*

Dispatch-Header und Header Compression

- Dispatch-Header gibt nachfolgenden Paketkopftyp an
  - Kodierungsmöglichkeit für Protokollerweiterungen
- Unterscheidung der Paketköpfe anhand „Dispatch Byte“
  - 2 Bit Dispatch-Header-Präfix, 6 Bit Selektor

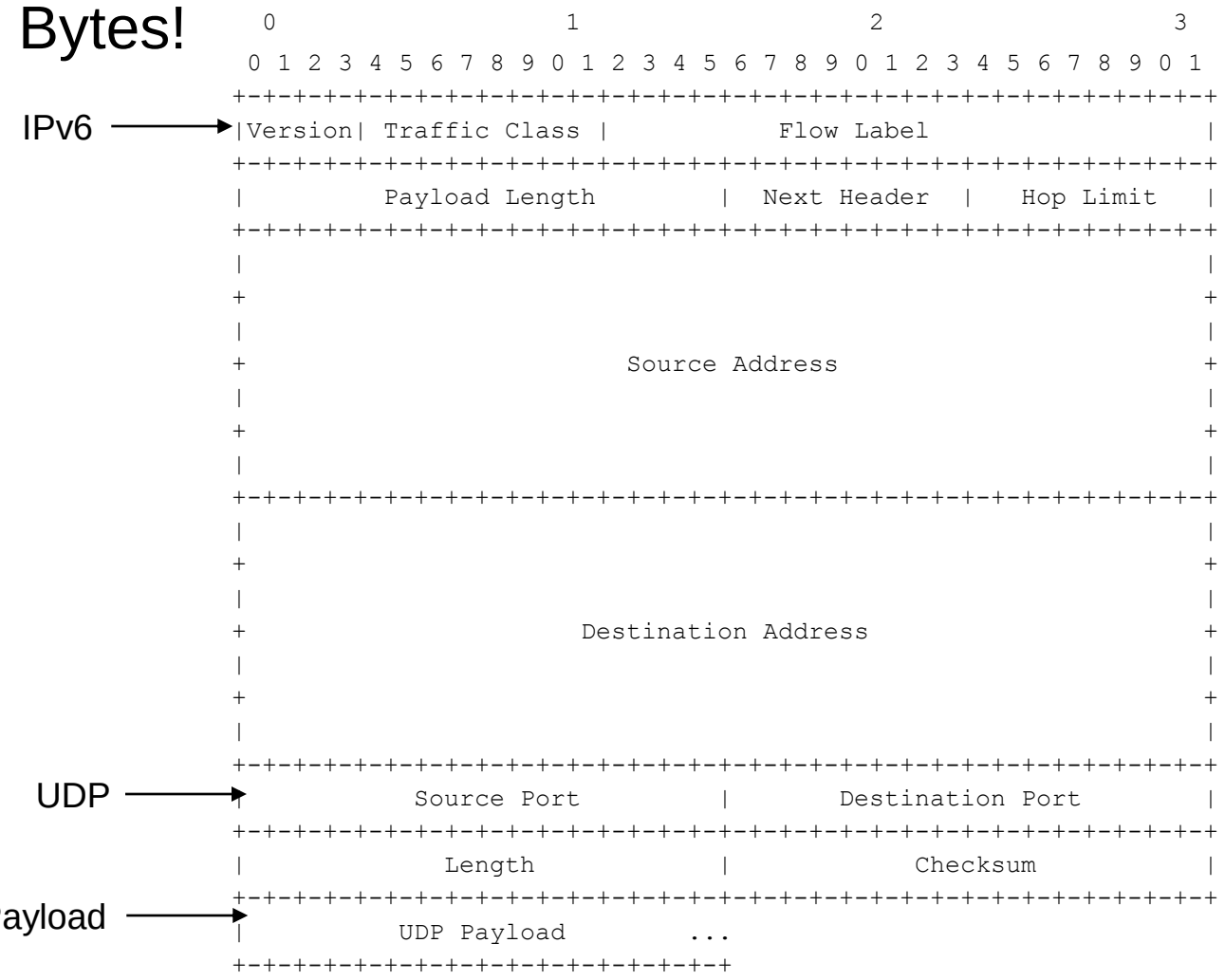


  - Im Anschluss folgt direkt der durch den Selektor beschriebene Paketkopf

  - Beispiele
    - 01 000001 → Unkomprimierter IPv6 Header
    - 01 000010 → Mit LOWPAN_HC1 komprimierte IPv6 Dateneinheit (mit vorangestelltem LOWPAN_HC1-Header)

Wiederholung: UDP/IPv6-Kopf

48 Bytes!



[ShBo09]

Komprimierung im IPv6-Kopf

- Was kann überhaupt im IPv6-Kopf komprimiert werden?
  - Beispiel: Kommunikation zweier Geräte im gleichen LoWPAN
  - Versionsnummer immer 6
  - IPv6 Ursprungs- und Zieladresse sind link-lokale Unicast-Adressen
    - → Interface IDs (untere 64 Bits) können aus Schicht-2-Adresse abgeleitet werden
  - Payload Length
    - → Kann u.U. aus Schicht 2 Payload Length oder aus Datagram_size im Fragmentierungskopf berechnet werden
  - Traffic Class und Flow Label häufig 0
  - Next Header ist UDP, ICMP oder TCP

→ Kann auf 2 Byte Kopf komprimiert werden !

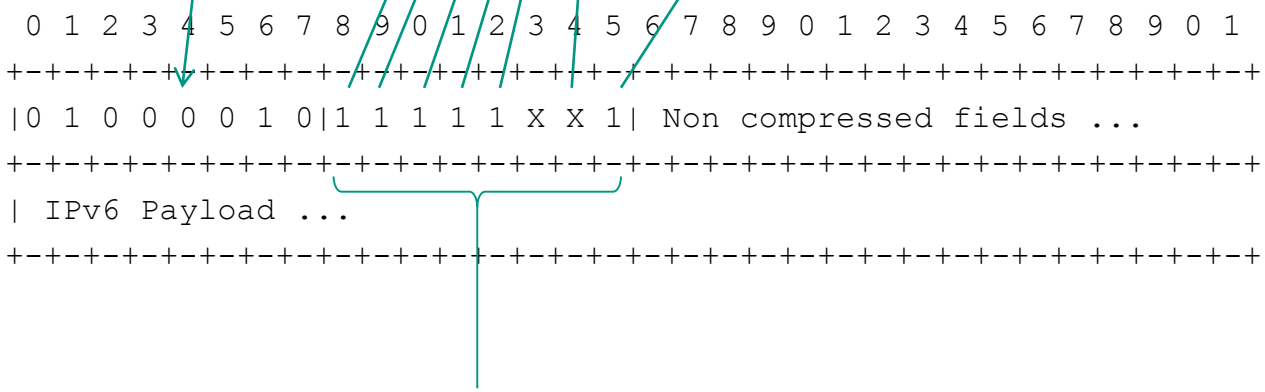
- Plus 1 Byte für das Hop-Limit Feld

Komprimierter IPv6-Kopf

LOWPAN_HC1

Dispatch Byte für LOWPAN_HC1

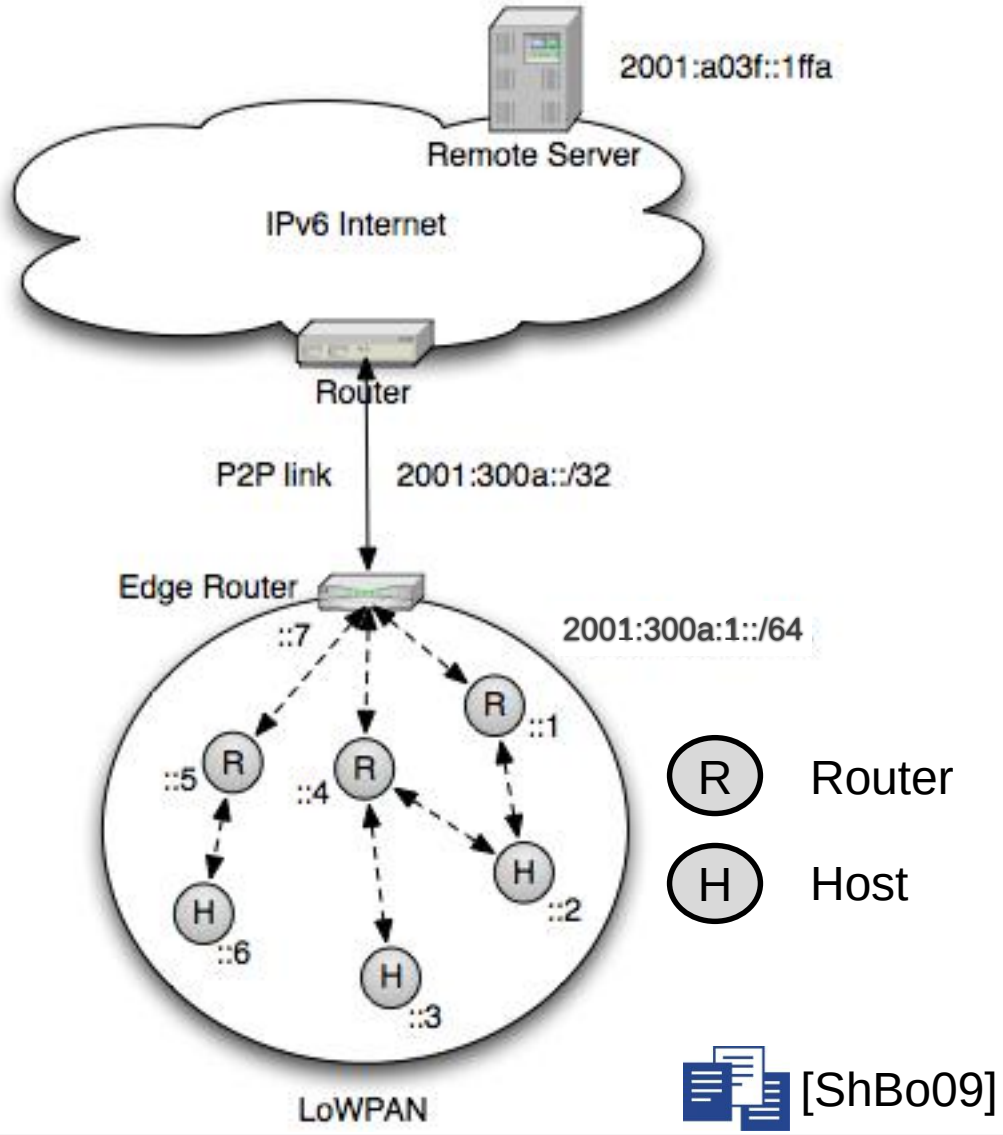
1: Source Prefix komprimiert	0: sonst
1: Source Interface ID komprimiert	0: sonst
1: Destination Prefix komprimiert	0: sonst
1: Destination Interface ID komprimiert	0: sonst
1: Traffic Class and Flow Laber komprimiert	0: sonst
XX: Identifiziert Next Header	(00) unkompr., (01) UDP, (10) TCP, (11) ICMP
Nutzung von HC2 Header Komprimierung für Transport Layer	0: sonst



LOWPAN_HC1 Header

Beispiel: Komprimierter IPv6-Kopf

- IPv6 Absender- und Zieladresse werden komprimiert
- Kommunikation auf dem selben Link → benötigt keine IPv6 Adressen da Adressen schon in Schicht-2-Rahmen vorhanden
  - Beispiel: ::5 nach ::6
- Kommunikation über mehrere Hops → verwende Schicht-2-Adresse als Quelle/Ziel im LoWPAN-Kopf
  - Beispiel: ::3 nach ::7
    - Bei erstem/letztem Hop kann Quell- bzw. Zieladresse aus Schicht-2-Rahmen abgeleitet werden, dazwischen sind beide im LoWPAN-Kopf erforderlich
  - Benötigt nur 16 Bit je Adresse statt 128 Bit
- Kommunikation nach außen → volle IPv6 Zieladresse notwendig
  - Beispiel: ::6 nach 2001:a03f::1fffa
  - Absenderadresse jedoch komprimiert, Edge Router ersetzt LoWPAN Kopf mit korrektem IPv6/UDP Kopf



[ShBo09]

Komprimierter UDP-Kopf

- Was kann überhaupt im UDP-Kopf komprimiert werden?
  - Port Nummer
    - Benötigen wir im IoE 2^{16} Ports ? → Eventuell weniger ausreichend
  - UDP Payload Length

    →Kann u.U. aus Schicht 3 Payload Length berechnet werden

- Was kann nicht komprimiert werden?
  - Prüfsummenkomprimierung schwierig

→ IPv6-Kopf + UDP-Kopf können auf 6 Byte komprimiert werden!

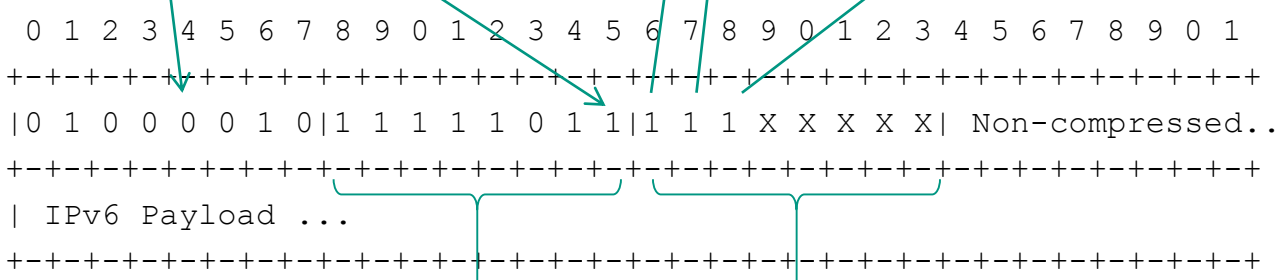
Komprimierter UDP-Kopf

LOWPAN_HC2

Dispatch Byte für LOWPAN_HC1

LOWPAN_HC2 verwendet

1: UDP Source Port komprimiert und In-Line übertragen	0: Unkomprimiert
1: UDP Source Port komprimiert und In-Line übertragen	0: Unkomprimiert
1: Länge komprimiert	0: Unkomprimiert
XXXX: Reserviert	



LOWPAN_HC1 Header

LOWPAN_HC2 Header

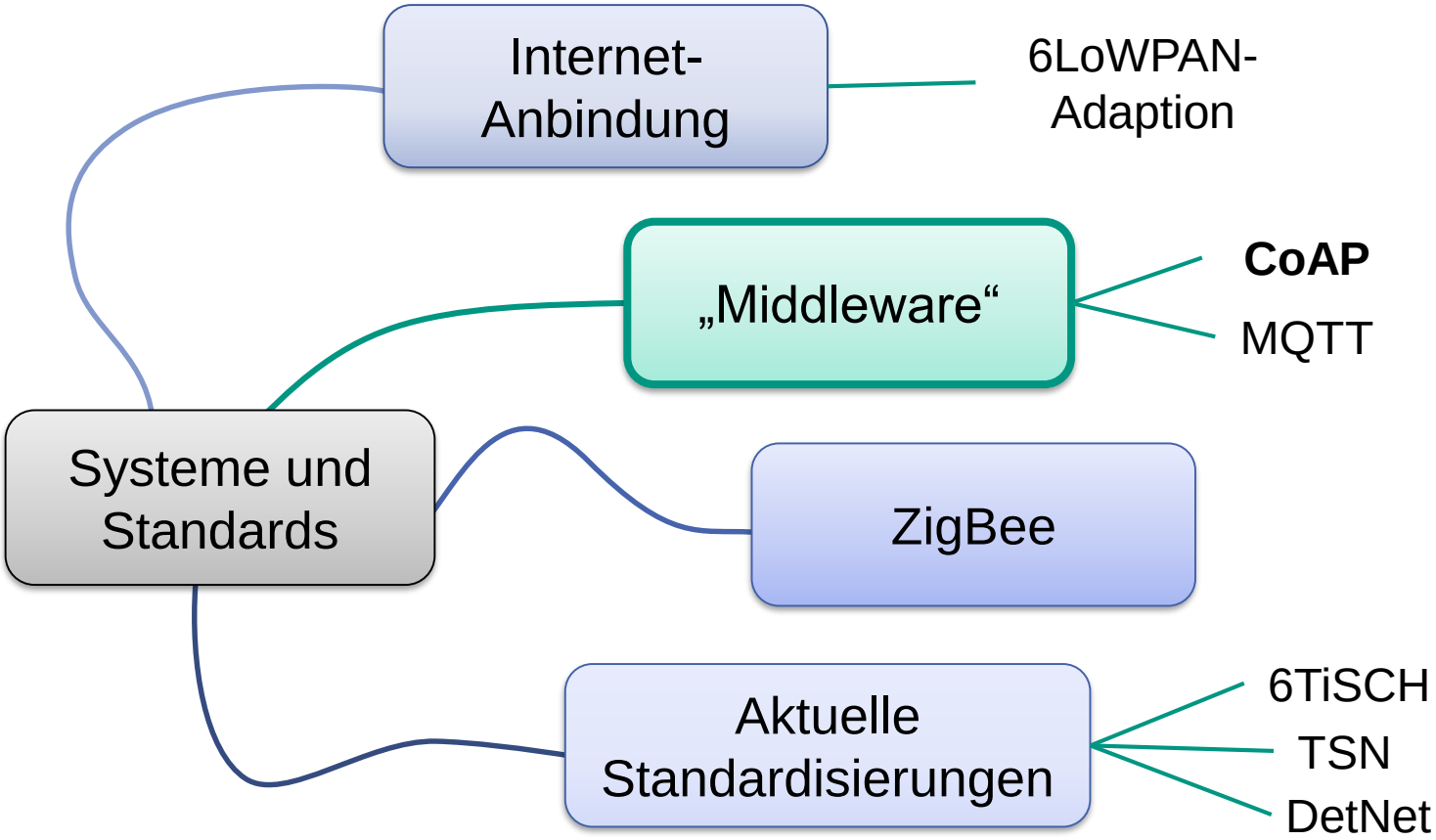
Komprimierung bei anderen Adresstypen

- RFC6282 definiert zwei weitere Komprimierungstechniken
  - LOWPAN_IPHC
    - Komprimierung von globalen Unicast Adressen unter Ausnutzung von Kontextinformationen
      - Es wird nicht beschrieben, wie Kontextinformationen ausgetauscht werden
      - Eine Möglichkeit wäre z.B. während der Neighbor Discovery
    - Komprimierung von Multicast-Adressen
  - LOWPAN_NHC
    - Komprimierung der Next-Header in IPv6
      - Sowohl UPD als auch TCP nutzbar

Zusammenfassung 6LoWPAN

- Ziel: IPv6-Konnektivität für Systeme mit eingeschränkten Ressourcen
- Adaption von IPv6 u.a. auf IEEE 802.15.4
  - Dabei Ausnutzung etablierter Protokolle und Mechanismen
  - Adaptionsschicht spezifiziert eigene Frame-Formate
    - Gateway für Übersetzung 6LoWPAN ←→ IPv6 erforderlich
- Eingeschränkte *link MTU* erfordert Fragmentierung
- Reduktion des IPv6-Overheads durch Header-Compression
  - Aber: Kompression von Adressen nur für Quelle/Ziel im LoWPAN
  - Zusätzliche Adresse entspricht ~8% der möglichen Datagrammgröße

Schwerpunkte des Kapitels im Überblick



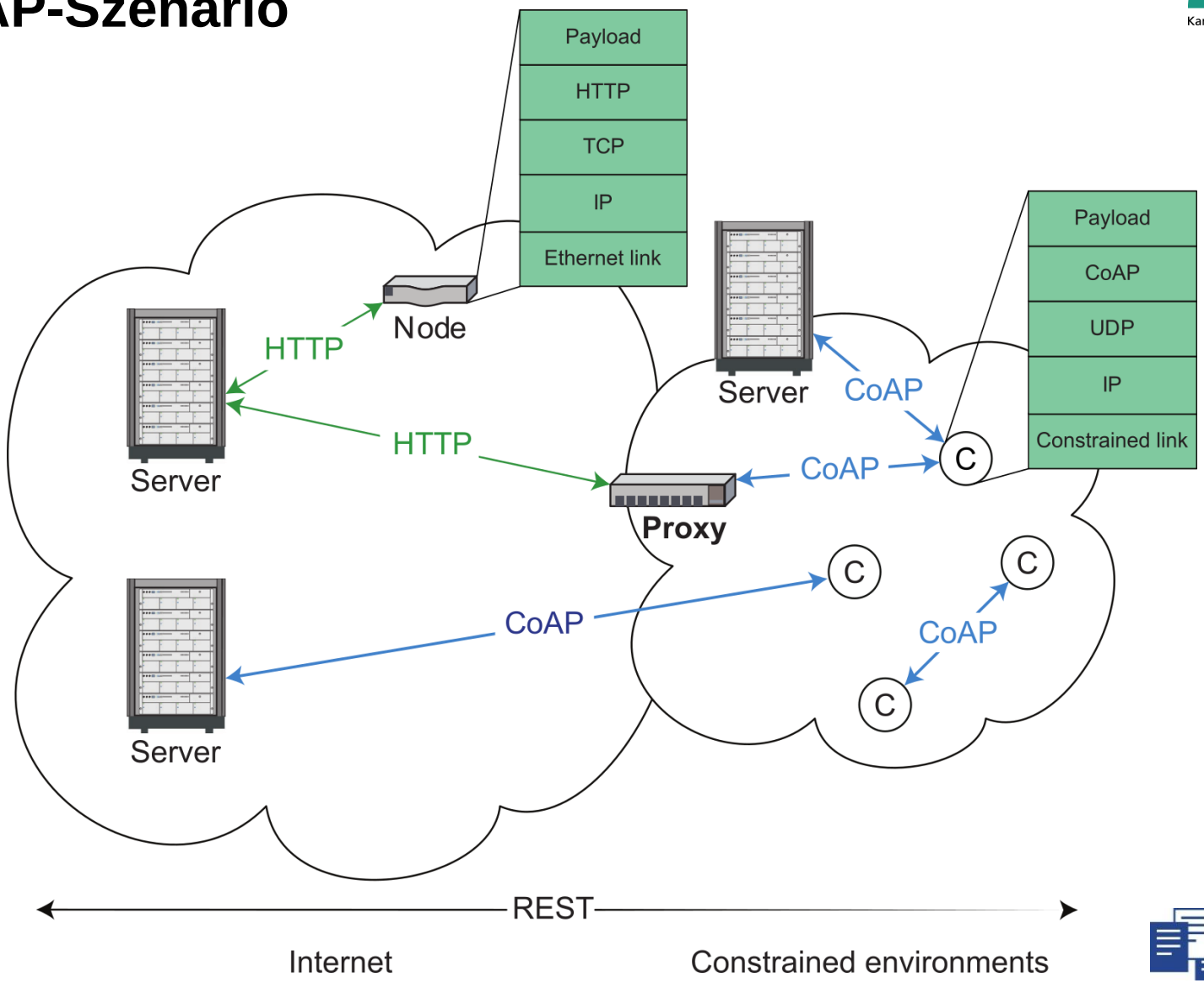
CoAP: Grundlegende Eigenschaften

- CoAP (**Co**nstrained **A**pplication **P**rotocol)
  - Transfer-Protokoll für eingebettete bzw. ressourcenbeschränkte Maschinen-zu-Maschinen Kommunikation
  - Leichtgewichtige Alternative zu HTTP

- Eigenschaften
  - Nutzung von UDP
    - Einfache Message-Schicht für Sendewiederholungen
  - Kompaktes Format
  - RESTful (Representational State Transfer)
    - GET, PUT, POST und DELETE
    - Unterstützung von URIs (Uniform resource identifier)
  - Interworking mit HTTP
    - Einfache Proxy und Caching Umsetzung
  - Unterstützung von DTLS (Datagram Transport Layer Security)

[RFC7252]

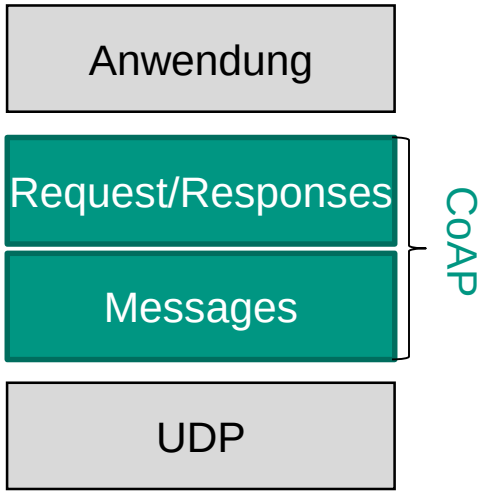
CoAP-Szenario



[BoCS12]

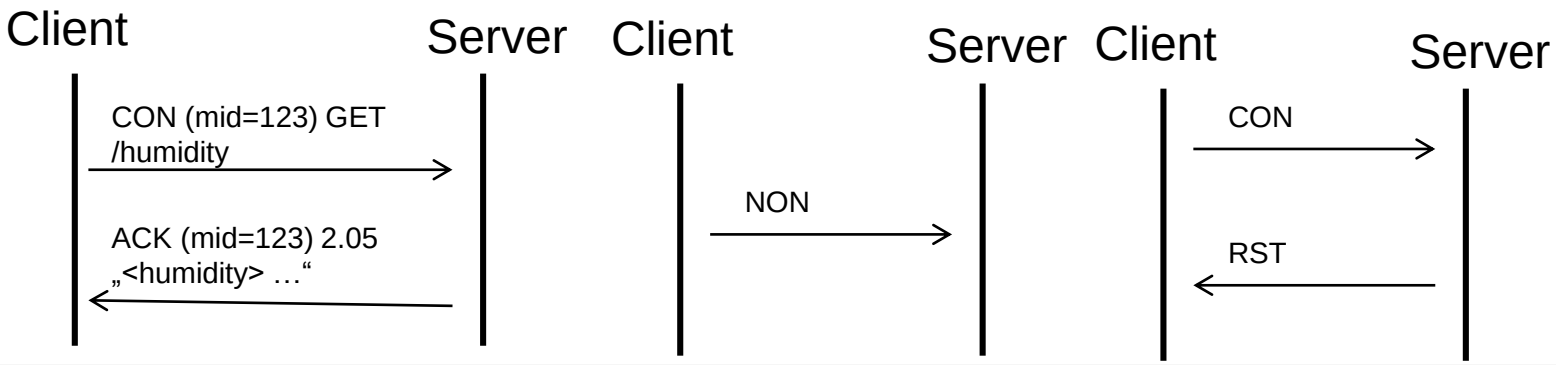
CoAP-Schichten

- Message-Schicht
  - Mechanismen zur zuverlässigen Nachrichtenübertragung
  - 4 Typen von Nachrichten
    - Confirmable, Non-confirmable, Acknowledgement, Reset
  - Übertragung von Request/Responses
- Requests/Responses
  - Enthalten jeweils einen Methodenaufruf auf ein Objekt
  - Vier Methoden: GET, PUT, POST und DELETE
  - Ein Objekt wird durch einen Objektindentifikator gekennzeichnet (URIs)



Messaging Modell

- Eindeutige Nachrichten-ID
  - Duplikaterkennung, Zuordnung von Quittungen
- Unzuverlässige Übertragung
  - Nachrichtentyp Non-Confirmable (NON)
- Zuverlässige Übertragung
  - Quittungen und exponentieller Backoff
  - Nachrichtentyp Confirmable (CON)
  - Quittungen haben den Nachrichtentyp Acknowledgement (ACK)
- Signalisierung von Fehlern
  - Nachrichtentyp Reset (RST)
- Beispiele



Request/Response Modell

- CoAP basiert wie HTTP auf einem Client/Server Modell
  - …allerdings agieren beide Kommunikationspartner häufig sowohl als Client als auch als Server
  - CoAP-Request vergleichbar mit HTTP-Request
  - Client sendet Request an Server, um Aktion auf einer Ressource auszuführen

- CoAP Request besteht aus
  - Auszuführender Methode auf der Ressource
  - Identifikator der Ressource
  - Nutzdaten
  - Optional: Meta-Daten zum Request

- Client erzeugt zur eindeutigen Identifikaton des Requests einen sogenannten Token

Request/Response Modell

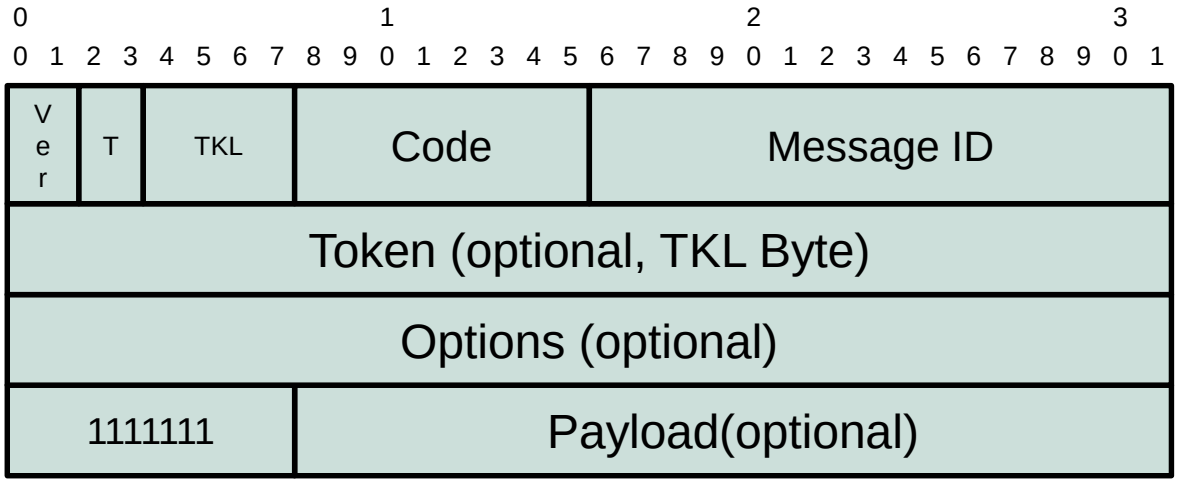
- CoAP Methoden in Requests
  - GET: Aufruf eines Objekts
  - PUT: Speichern eines Objekts
  - POST: Übermittlung neuer Informationen
  - DELETE: Löschen eines Objekts

- Server antwortet mit Response und dazugehörigem Response-Code
  - CoAP Response-Code vergleichbar mit HTTP Status Code
  - Es existieren drei Klassen an Responses
    - 2 - Success: Request wurde erfolgreich empfangen, interpretiert und ausgeführt
    - 4 - Client Error: Request enthält falsche Syntax oder ist nicht ausführbar
    - 5 - Server Error: Server konnte Request nicht ausführen

- Responses können sowohl Piggy-backed (im ACK) als auch in separaten Nachrichten übertragen werden

Nachrichtenformat

- CoAP nutzt einfaches und kompaktes Nachrichtenformat



Ver : Versionsnummer
T : Nachrichtentyp (CON,NON,ACK,RST)
TKL : Token Länge
Code : Identifiziert Methodenaufruf und Fehlercode
Token : Wird genutzt, um zusammengehörende Requests und Responses zu identifizieren
Options : CoAP definiert verschiedene Optionen, beispielsweise URI-Path: "temperature„
Payload : Nach dem Payload Marker (0xFF) folgt der eigentliche Inhalt

CoAP URIs

- CoAP nutzt „coap“ und „coaps“ URI-Schema um Ressourcen und deren Lokation zu identifizieren
  - „coaps“ bei der Verwendung von DTLS, vergleiche https

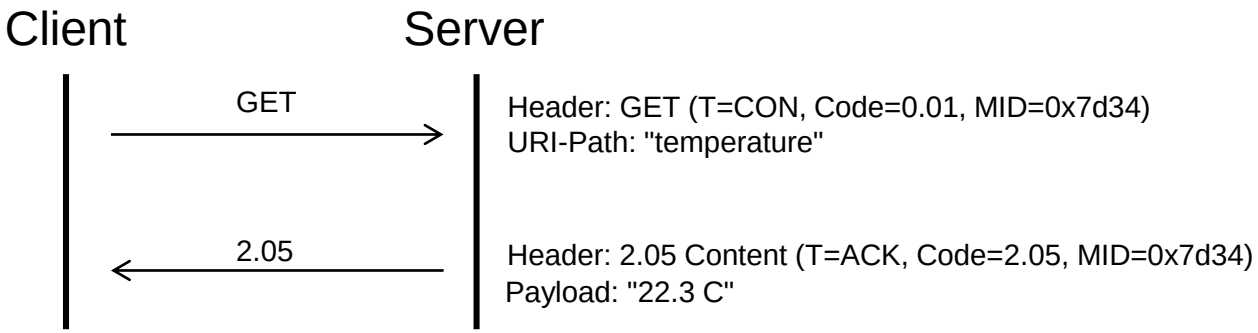
- Syntax für Erstellung von CoAP-URIs
  - Coap-URI = "coap:" "//" host [":" port] path-abempty ["?" query]

- Beispiel zur Erzeugung von URIs

Input: Destination IP Address = [2001:db8::2:1]
Destination UDP Port = 61616
URI-Host = "example.net"
URI-Path = ".well-known"
URI-Path = "core"

Output: coap://example.net:61616/.well-known/core

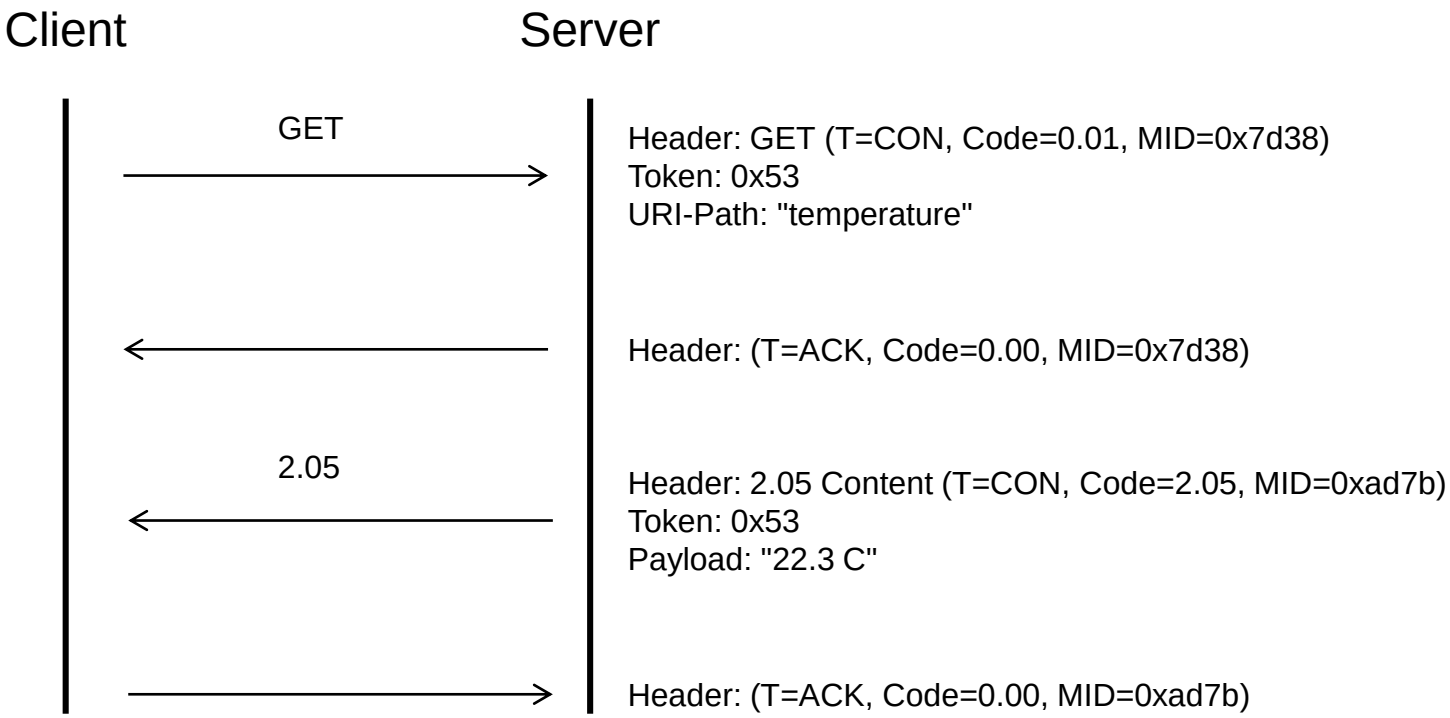
Beispiel: CON Request, Piggy-backed Response



Nachrichteninhalte:

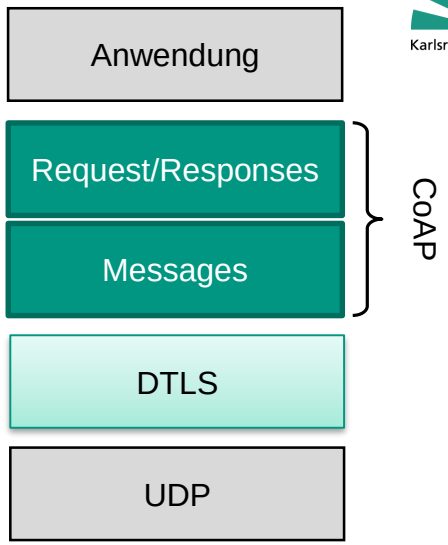
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 0 |   0   |  GET(0.01)=1  |          MID=0x7d34           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  11   |  11   |      "temperature" (11 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 2 |   0   | CONT(2.05)=69 |          MID=0x7d34           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|      "22.3 C" (6 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Beispiel: CON Request, Separate Response



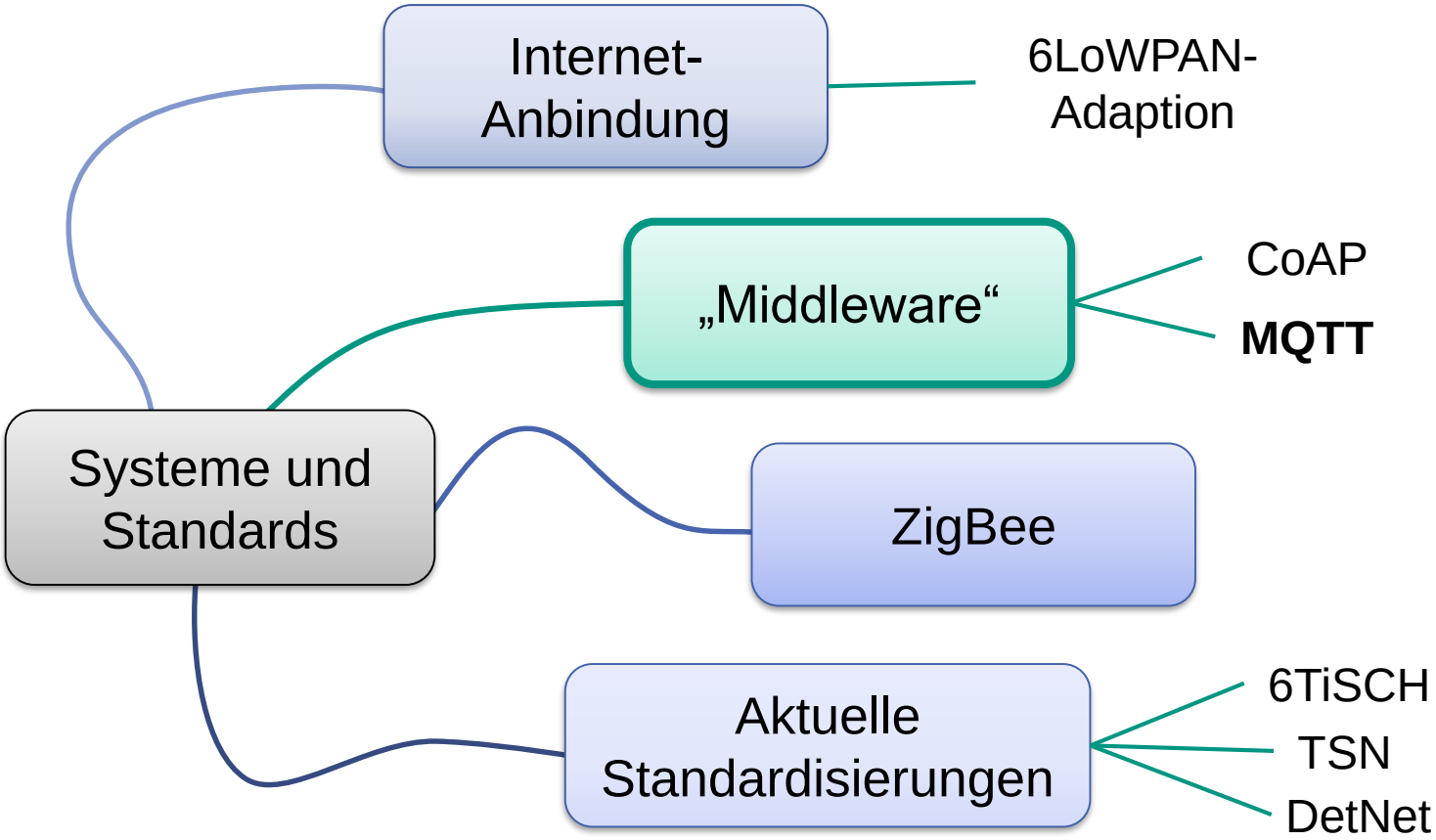
- Token stellt Bezug zwischen Request und Response her
  - Message ID nur zwischen Nachricht und Quittung

Sicherheit und CoAP



- CoAP unterstützt die Nutzung von DTLS
  - Somit Integritätsschutz, Vertraulichkeit und Authentifizierung für Anwendungsdaten
- Es exisitieren vier Sicherheitsmodi
  - NoSec: DTLS wird nicht genutzt
  - PreSharedKey: DTLS, vorverteilte Schlüssel
  - RawPublicKey: DTLS, asymmetrische Schlüsselpaare
  - Certificate: DTLS, X.509 Zertifikate
- Für den Einsatz auf ressourcenschwachen Geräten kommen spezielle, angepasste Cipher-Suites zum Einsatz
  - → Siehe Kapitel Sicherheit

Schwerpunkte des Kapitels im Überblick



MQTT: Grundlegende Eigenschaften

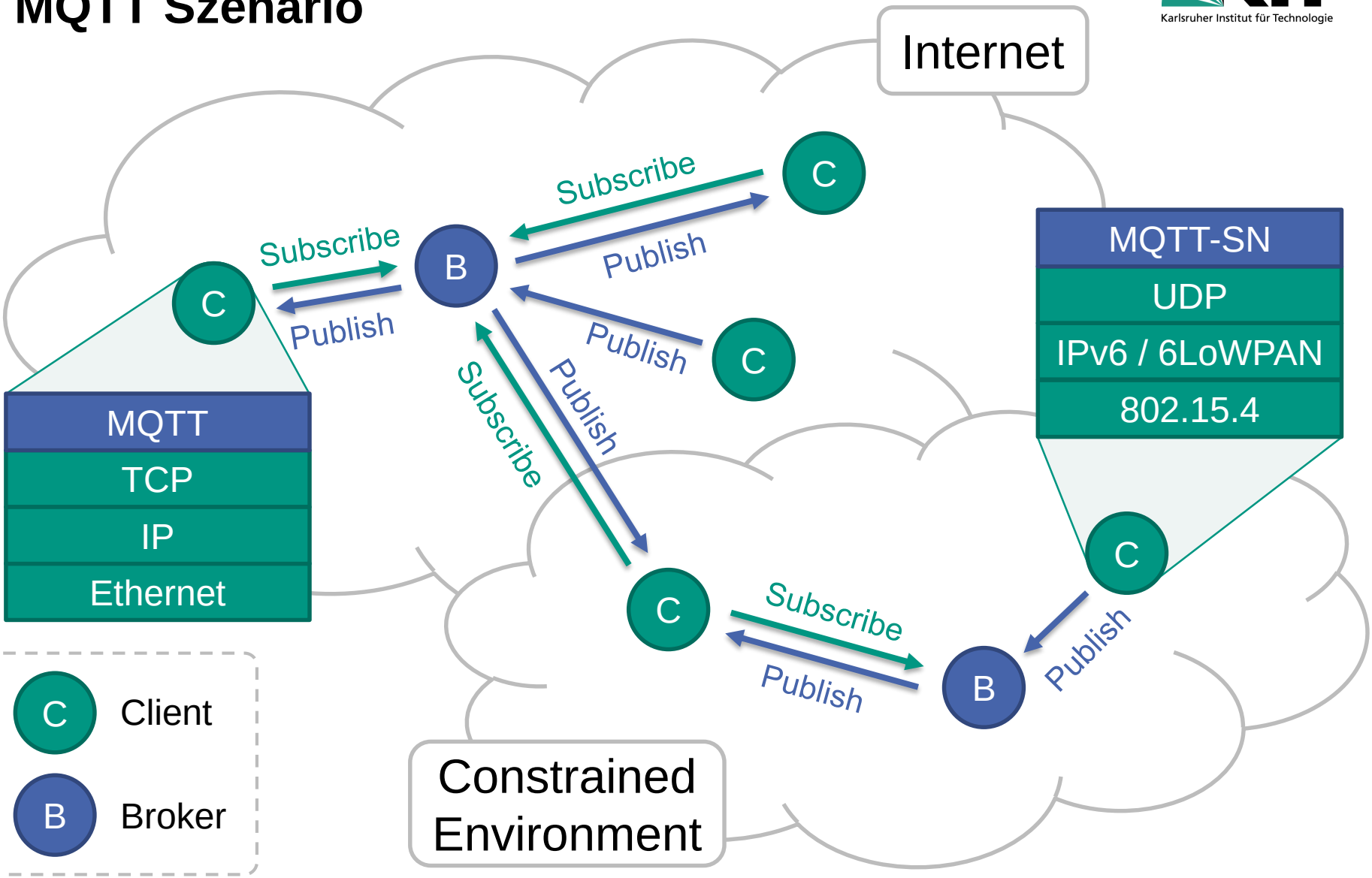
- MQTT (**M**essage **Q**ueue **T**elemetry **T**ransport)
  - Wie CoAP: Transfer-Protokoll für eingebettete bzw. ressourcenbeschränkte Maschinen-zu-Maschinen Kommunikation
  - Fokus auf Many-to-Many-Kommunikation
  - ISO/OASIS-Standard, geht u.a. auf IBM zurück


[MQTT]

- Eigenschaften
  - Nutzung von TCP
    - MQTT setzt (im Gegensatz zu CoAP) zuverlässigen Transportdienst voraus
    - MQTT-SN: Auf UDP aufbauende MQTT-Variante für Sensornetze
  - Einfaches Protokoll, leichtgewichtige Implementierung
  - Publish-Subscribe-Paradigma
    - Zuordnung von Inhalten/Nachrichten zu Topics
    - Nachrichtenspeicherung und -weiterleitung durch zentralen Broker
  - Sicherheit über TLS/DTLS

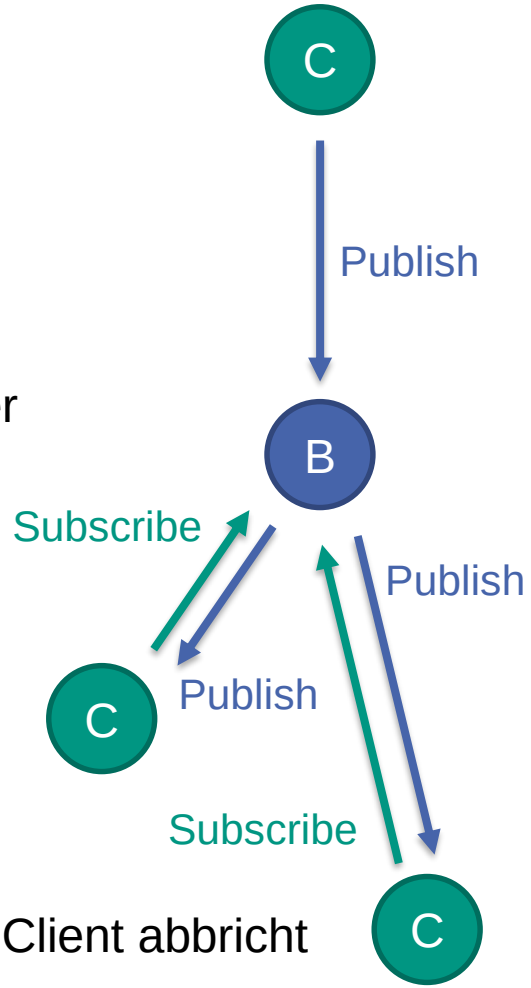

[MQTT-SN]

MQTT Szenario



Publish/Subscribe Modell

- Ereignisorientiertes Modell
- Nachrichtenweiterleitung über zentralen Broker
  - Datenquellen senden Nachricht an Broker
    - Einem Topic (Thema/Betreff) zugeordnet
    - `PUBLISH „Home/Fridge/Temperature“ „7°C“`
  - Datensenken (Clients) abonnieren Topics bei Broker
    - `SUBSCRIBE „Home/Fridge/Temperature“`
- Broker kann letzte Nachricht im Topic puffern
  - Weiterleitung an neue Clients
- „Letzter Wille“
  - Spezielle Nachricht, bei Broker hinterlegt
  - Veröffentlichung durch Broker wenn Verbindung zu Client abbricht

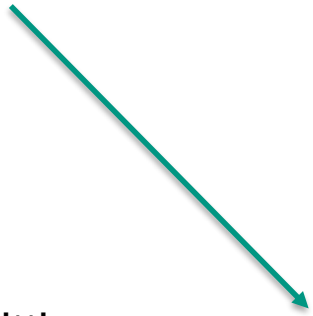


Quality of Service

- Abgestufte Quality of Service (Zuverlässigkeit)
  - Level 0: Keine Garantien
  - Level 1: Clients empfangen Nachricht mindestens einmal
  - Level 2: Clients empfangen Nachricht genau einmal

- QoS-Aushandlung
  - Client bestimmt maximales QoS-Level (`SUBSCRIBE`)
  - Broker kann verfügbares QoS-Level begrenzen
  - Absender legt gewünschtes QoS-Level fest (`PUBLISH`)
  - → Tatsächliche QoS kann geringer sein

- Paket-ID in Nachrichten
  - Nur für Level 1 und 2
  - Eindeutig zum Sendezeitpunkt
  - Wiederverwendung erst nach Quittierung



	QoS-Level
Client	≤1
Server	≤0
Angefordert	≤2
Effektiv	**0**

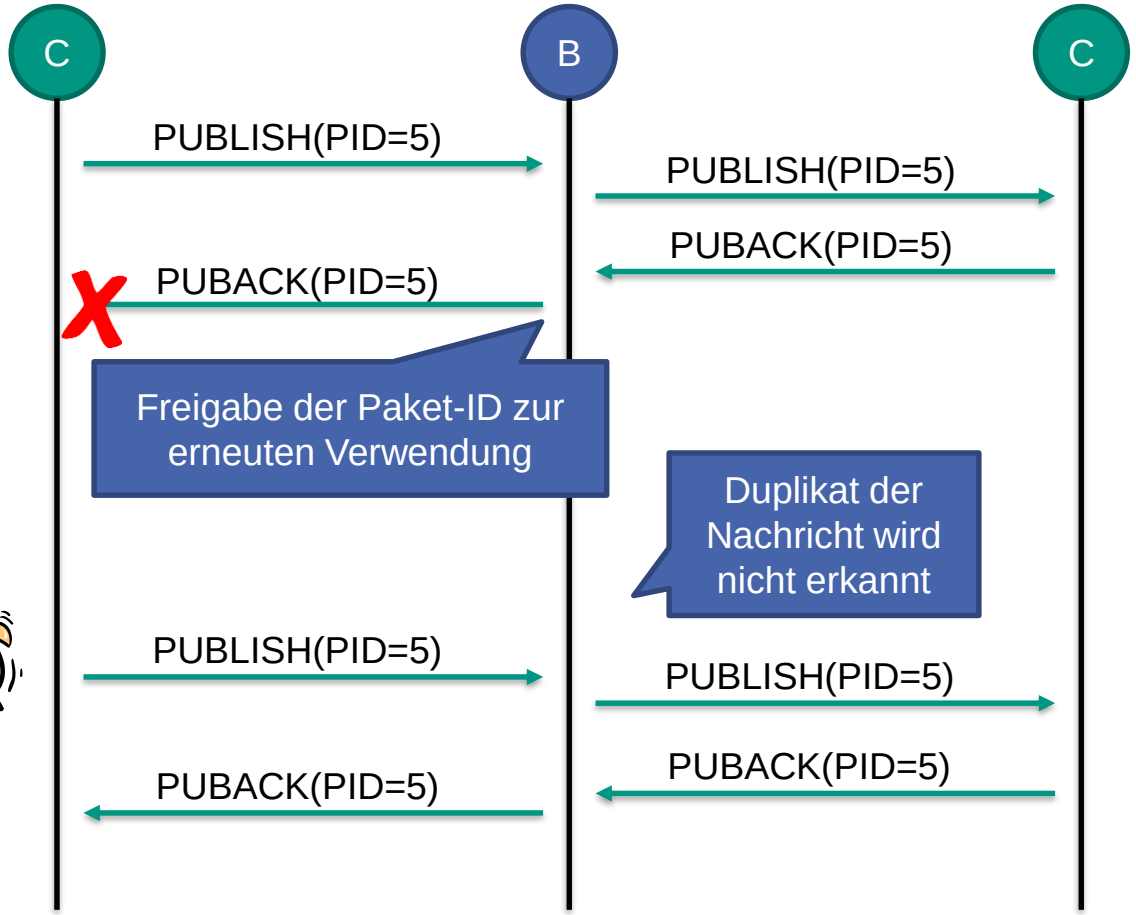
MQTT QoS Level 1

- Duplikate bei Empfänger möglich

- Ursache:
  PUBACK hat Doppelfunktion
  - Quittung
  - Freigabe Paket-ID

Empfänger „vergisst“ Nachricht nach Quittierung

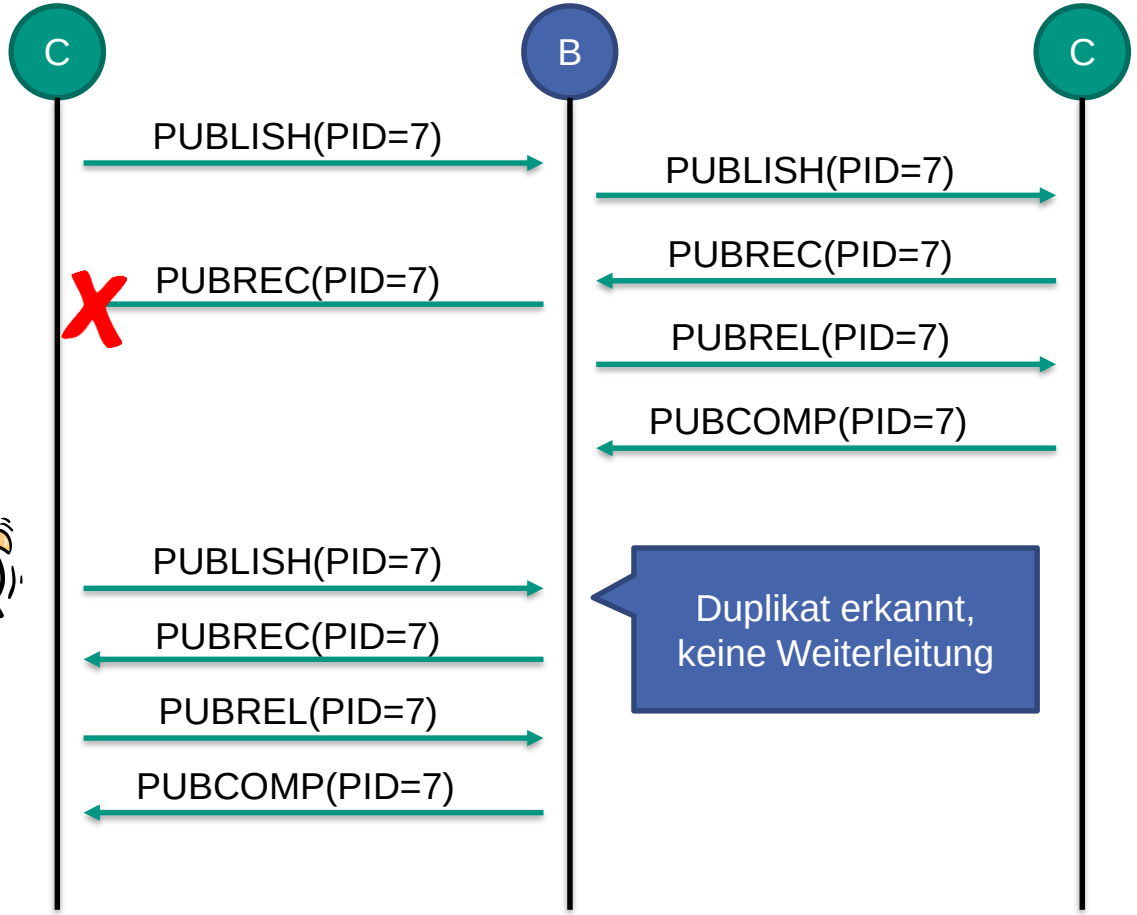
- Sendewiederholung nach verlorener Quittung wird nicht erkannt



MQTT QoS Level 2

- Keine Duplikate bei Empfänger

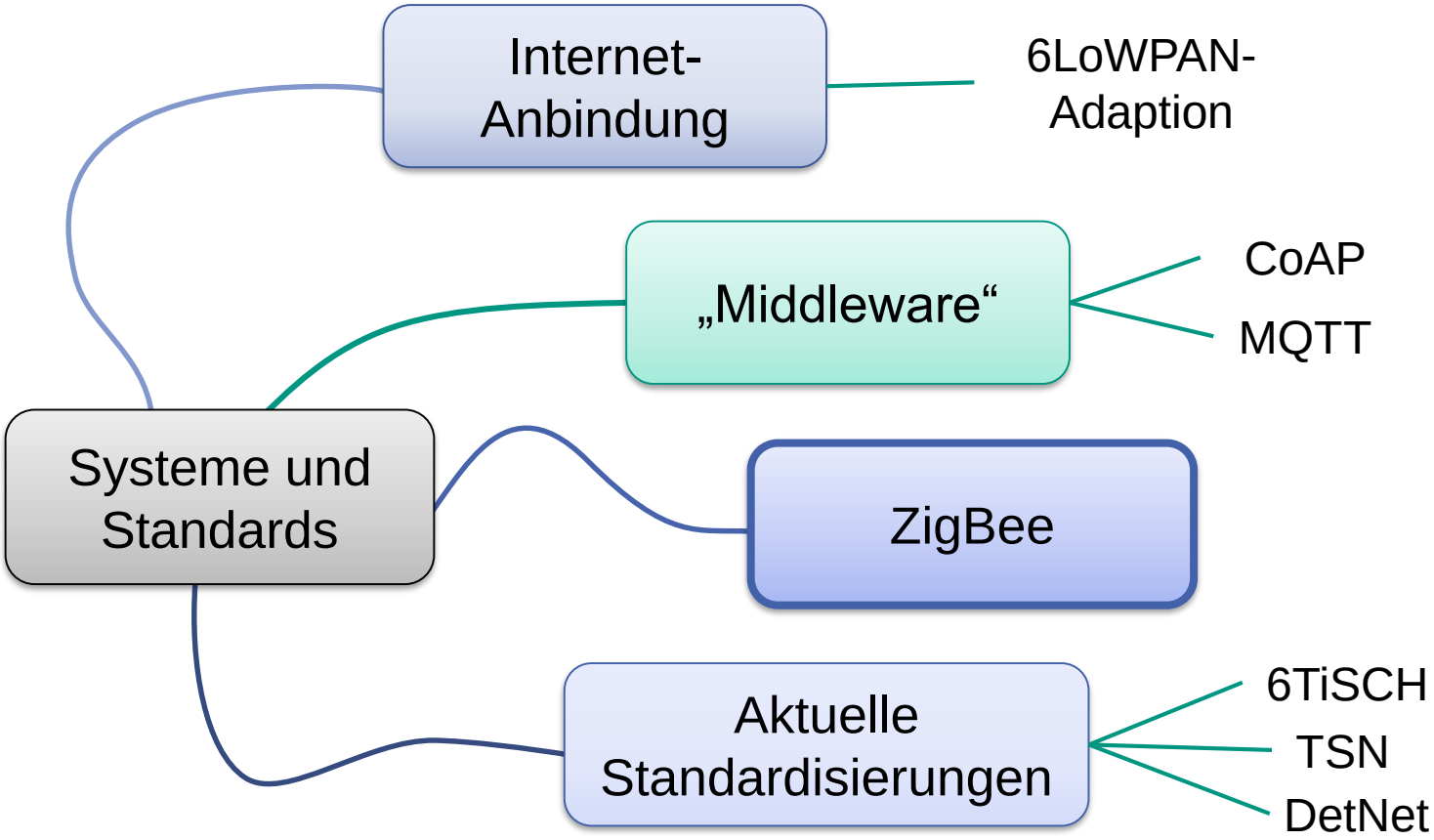
- Aufgabenteilung gegenüber Level 1:
  - PUBREC: Quittierung
  - PUBREL: Freigabe Paket-ID
  - PUBCOMP: Quittiert PUBREL



Gegenüberstellung MQTT/CoAP

- Grundsätzlich anderer Ansatz als bei CoAP
  - Publish/Subscribe-Paradigma mit zentralem Broker
  - → Keine Ende-zu-Ende-Verbindung zwischen Datenquelle und Senke
- MQTT kann Datenquellen entlasten …
  - Jede Information muss nur einmal veröffentlicht werden
  - Broker kann Nachrichten zwischenspeichern
  - CoAP benötigt einmal Request/Response für jeden Client
    -  [RFC7390] Experimentelle Multicast-Unterstützung für CoAP
    -  [RFC7641] Ereignisbasierte Benachrichtigung für Clients (statt Polling)
- … oder belasten
  - Wenige Senken und Aktualisierungsrate ≫ Bedarf
  - → Weniger Aufwand für bedarfsorientiertes Request/Response-Modell
- Abgestufte Zuverlässigkeit bei MQTT
  - MQTT QoS 2 ≙ CoAP CON

Schwerpunkte des Kapitels im Überblick



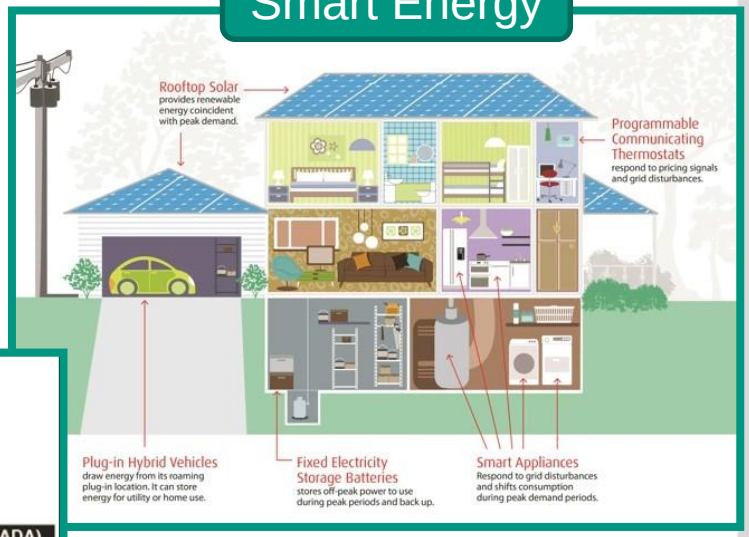
Zwei verschiedene Sichten

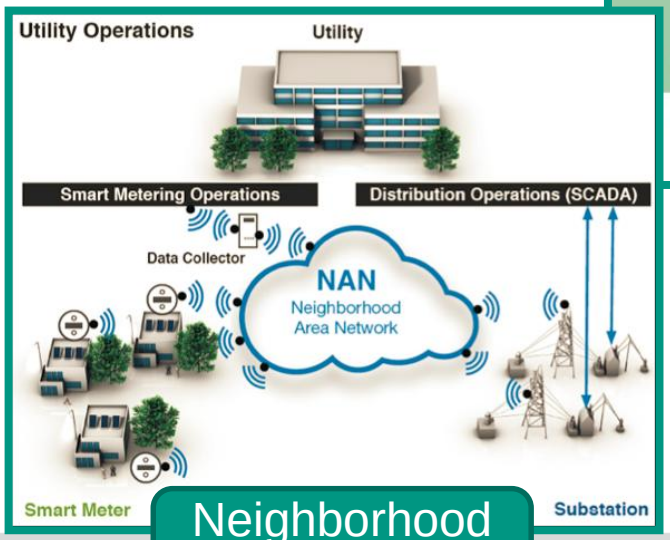
- 6LoWPAN
  - Von der IETF getrieben
  - Anbindung an das Internet steht im Mittelpunkt, basierend auf IPv6
  - Anwendungsneutral

- ZigBee
  - *“The Open, Global Wireless Standard for Connecting Everyday Devices”*
    - *„ZigBee is the language that allows you to control the everyday devices around you“*
  - Standards für das Internet der Dinge
  - Von der ZigBee Alliance getrieben
    - Zusammenschluss von Industrieunternehmen
  - Andere Sichtweise: Sehr stark anwendungsorientiert
  - Kleinere autonome Netze stehen im Mittelpunkt
    - Keine Interoperabilität mit etablierter Infrastruktur

ZigBee Anwendungsbereiche

- Ziel
  - Standards zur Realisierung von Anwendungen mit Überwachungs- und Steuerungsaufgaben; einheitliche, herstellerübergreifende Schnittstellen
    - Annahme: Geringes Datenaufkommen
    - Ziel: Geringer Energieverbrauch
- Beispiele
  - Haus- und Gebäudeautomation
  - Beleuchtungs-steuerung
  - Gesundheits-pflege
  - Einzelhandel und Logistik
  - Smart Energy

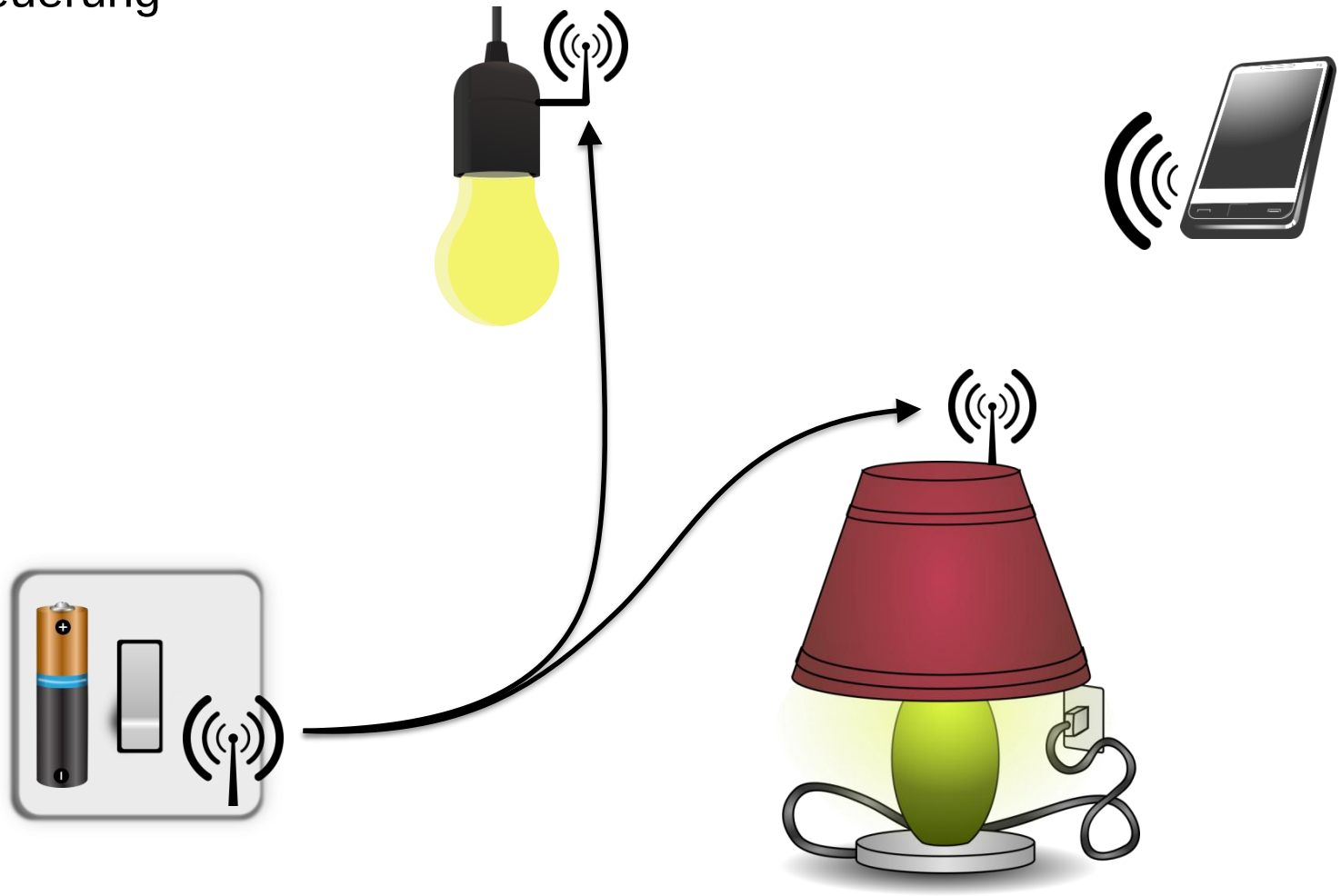




[http://www.zigbee.org/what-is-zigbee/utility-industry/]

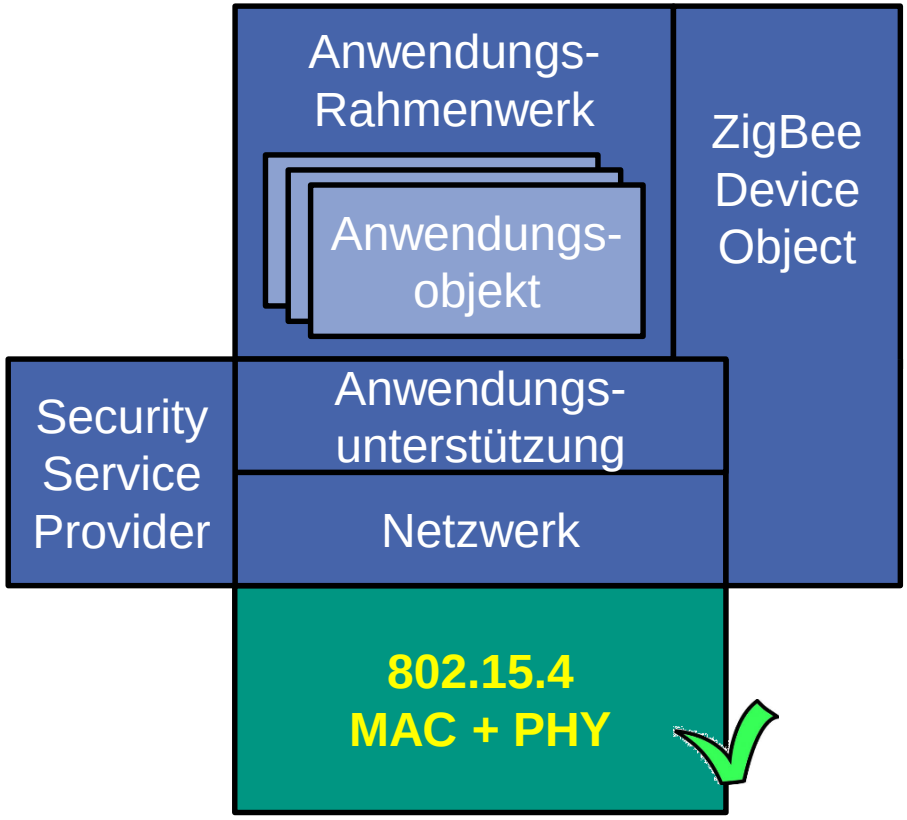
Einfaches Anwendungsbeispiel

- Lichtsteuerung



Architektur – Schichtenmodell

- 802.15.4
  - Medienzugriffsprotokoll
- Netzwerk-Schicht
  - Routing
  - Adressierung
- Anwendungsschicht
  - Anwendungsunterstützung
    - Zuverlässiger Transportdienst
    - Binding (Kopplung von Geräten)
    - Gruppenverwaltung
  - Anwendungs-Rahmenwerk
    - Anwendungsobjekte (Anwendungsfunktionalität)
  - ZigBee Device Object
    - Management-Dienste, „Hypervisor“



[ZigBee] [KrKo14]

Anwendungs-Rahmenwerk

- Teil der Anwendungsschicht
- Einbettung von Anwendungsobjekten
  - Unterteilung der Anwendung in bis zu 240 Anwendungsobjekte
    - Je Anwendungsobjekt eine eigene Aufgabe
    - Eindeutige ID (1 … 240), entspricht einem Endpunkt (EP)
- Definition von Anwendungsprofilen
  - Standards für die Kommunikation zwischen Geräten in einem Anwendungsbereich, etwa Hausautomation
  - Identifiziert über Profile-Identifier
  - Standardprofile
    - Entwickelt von der ZigBee Alliance
    - Allgemeine nützliche Anwendungsbereiche
  - Private Profile
    - Von individuellen Herstellern entwickelt
    - Benötigen einen von der ZigBee Alliance zugeordneten Profile-Identifier

Anwendungsprofile

- Bestehen aus
  - Menge von Geräten (Devices) die für die Anwendung benötigt werden
  - Menge von Clustern um die Funktionalität zu implementieren
  - Menge von Attributen, die den Zustand von Geräten repräsentieren
  - Menge von Befehlen die Kommunikation ermöglichen
  - Beschreibung welche Cluster von welchen Geräten benötigt werden
  - Funktionale Beschreibung für jedes Gerät
- Beispiel
  - ZigBee Home Automation Profile (Profile ID: 0x0104)

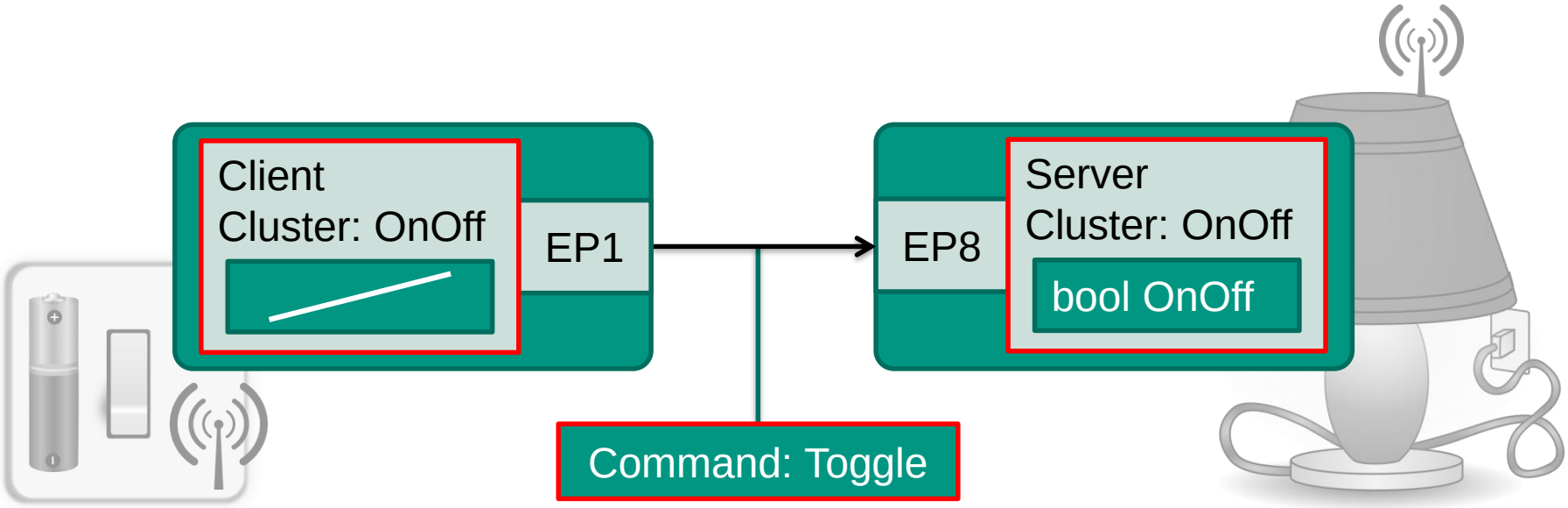
Device	Device ID
Door Lock	0x000A
On/Off Light Switch	0x0103
Dimmer Switch	0x0104
Thermostat	0x0301

Cluster	Cluster ID
Door Lock	0x0101
Window Covering	0x0102


[ZigBHA]

Client/Server-Modell

- Anwendungen basieren auf Client-Server Modell
  - Attribute sind (typischerweise) Servern zugeordnet
    - Beispiel: Status einer Lampe (Cluster OnOff)
  - Clients nutzen Attribute über standardisierte Cluster-Befehle
    - Beispiel OnOff-Cluster: Ein/Aus/Umschalten



Cluster

- Menge von Attributen und Befehlen
- Jeweils einer Rolle (Client oder Server) zugeordnet

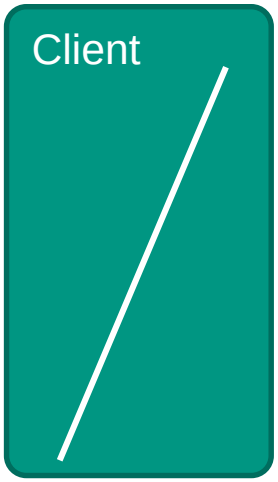
- ZigBee Standards spezifiziert keine Cluster
  - Cluster werden in der ZigBee Cluster Library zusammengefasst
  - Beispiel
    - On/Off-Cluster (Cluster ID: 0x0006)


[ZCLib]

Server

Attribut ID	Name	Typ	Zugriff
0x0000	OnOff	boolean	read-only

Command ID	Name
0x00	Off
0x01	On
0x02	Toggle



Geräte

- Device Description
  - Funktionale Beschreibung eines Gerätetyps
  - Spezifiziert welche Cluster die Funktionalität bereitstellen
  - Beispiel
    - *Dimmer Switch* (Device Description ID: 0x0104)

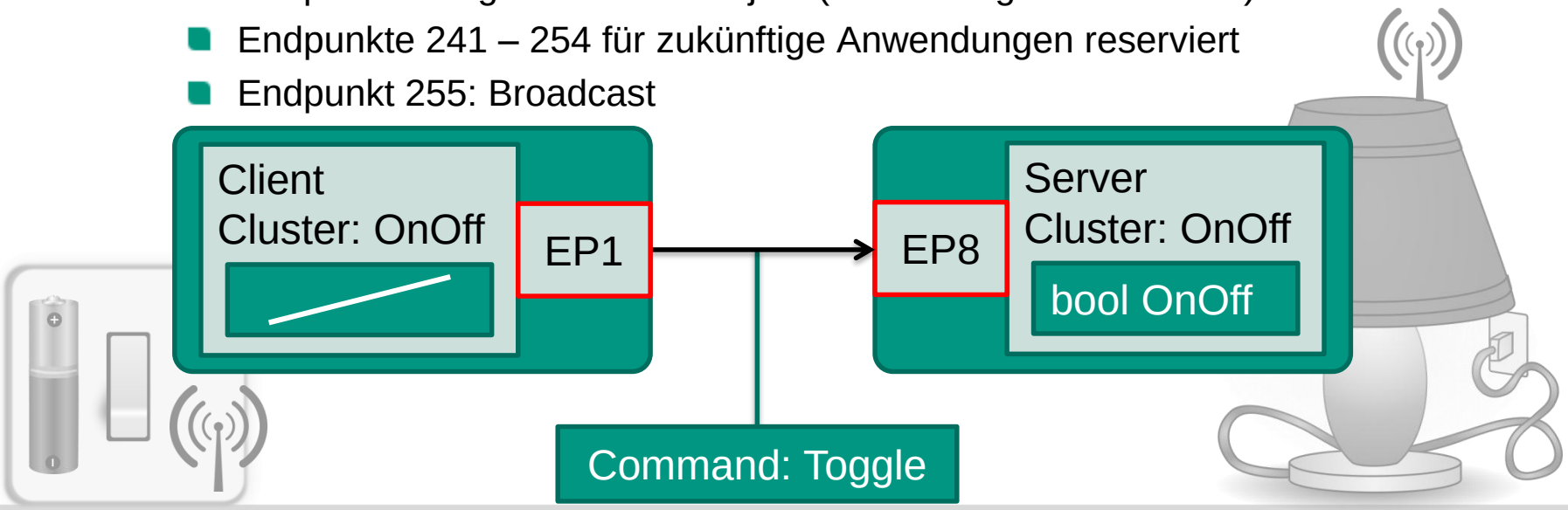
Server-Funktionalität	Client-Funktionalität
	On/Off (0x0006)
	Level Control (0x0008)
Optional	
On/Off-Switch Configuration (0x007)	Scenes (0x005)
	Groups (0x004)

Cluster IDs

[ZigBHA]

Endpunkt

- Entspricht konzeptionell einem Port im Internet-Modell
  - Kommunikation erfolgt immer zwischen zwei Endpunkten
  - Endpunkt ist genau einer Device Description zugeordnet
    - Gerät kann über mehrere Endpunkte verfügen
      - Z.B. Smartphone mit EP2 zum Steuern einer Lampe und EP5 zum Steuern einer Klimaanlage
  - Maximal 256 Endpunkte möglich
    - Endpunkt 0: ZigBee Device Object (Verwaltungsfunktionalität)
    - Endpunkte 241 – 254 für zukünftige Anwendungen reserviert
    - Endpunkt 255: Broadcast



Anwendungsunterstützung

- Teil der Anwendungsschicht
  - „Schnittstelle zur Netzwerkschicht“
- Wesentliche Aufgaben
  - Zuverlässiger Transportdienst
  - Binding (Kopplung von Geräten)
  - Gruppenverwaltung

Zuverlässiger Transportdienst

- Nutzt 802.15.4
  - Beschränkte Größe der Dateneinheiten
  - Unzuverlässiger Dienst, hohe Bitfehlerwahrscheinlichkeit

- Datentransport mit ZigBee
  - Keine eigene Schicht
  - Verbindungslos
  - Fragmentierung für große Dateneinheiten
  - Keine „TCP-like“ Zuverlässigkeit
    - ⊕ Quittierung (Ende-zu-Ende), nur bei Fragmentierung verpflichtend
    - ⊕ Duplikaterkennung (also Sequenznummern)
    - ⊖ Fehlererkennung (Schicht 2 hat aber Prüfsummen)
    - ⊖ Reihenfolgetreue (außer bei Fragmentierung …)
    - ⊖ Staukontrolle
    - ⊖ Flusskontrolle (außer bei Fragmentierung …)

Nutzung von Quittungen

- Quittungen
  - Erfolgen pro übertragene Dateneinheit
  - Werden als spezielle Dateneinheit übertragen
    - Keine piggy-back Quittungen wie z.B. bei TCP

- Sendewiederholung
  - Nach Ablauf eines Timers
    - Maximal drei Sendewiederholungen
  - Bei Empfang einer „unerwarteten“ Quittung
    - Endpunkt, Cluster ID oder Sequenznummer stimmen nicht überein

Fragmentierung

- Hier: Quittungen verpflichtend

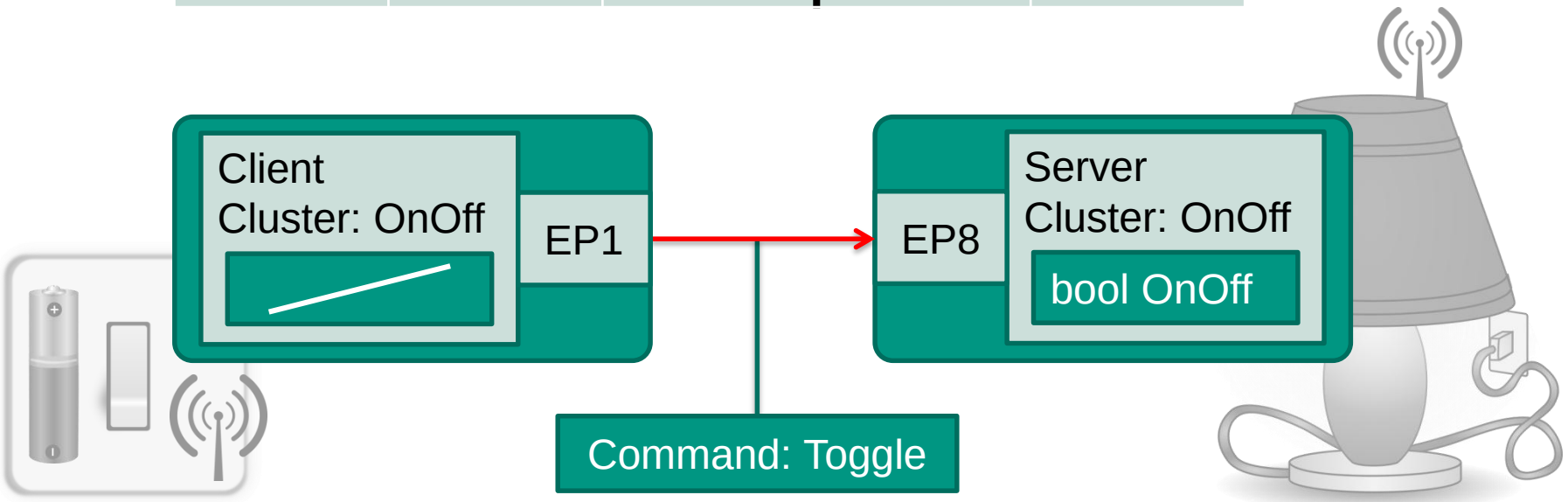
- Randbedingungen von 802.15.4
  - Max. 127 Bytes stehen auf der physikalischen Schicht zur Verfügung

- MAC-Datenfeld nicht ausreichend, dann Fragmentierung
  - Quittungen nicht pro Fragment sondern pro Fenster
    - Fenstergröße: 1-8 Fragmente
    - Quittung beinhaltet Sequenznummer des ersten Fragments und Bitmaske
      - Selektives ACK/NACK durch Bitmaske
    - Quittung gesendet bei
      - Erhalt des letzten Blocks im Sendefenster oder
      - vollständigem Empfang des Sendefensters oder
      - (Optional) Timeout
    - Nicht quittierte Blöcke müssen erneut gesendet werden
  - Kein Versand neuer Fragmente vor vollständiger Quittierung des Fensters

Binding

- Verknüpfung zwischen den Endpunkten zweier Geräte
  - Wird in Binding-Tabelle gespeichert
    - Im Netz stellt ein Gerät (Cache-Server) komplette Binding-Tabelle bereit
    - Bei der Kommunikation von Befehlen keine Adressen erforderlich
      - Quelle muss Zielgeräte nicht kennen

SrcAddr	SrcEndp	Cluster	DstAddr	DstEndp
0x1234	EP1	0x0006	0x2341	EP8



Gruppenverwaltung (Multicast)

- Gruppen
  - Identifiziert über eine Gruppen-ID
  - Gruppentabelle: Zuordnung Gruppen-ID → Lokale Endpunkte

- Endpunkte können mehreren Gruppen zugeordnet werden

Netzwerk-Schicht

- Routing
- Adressierung
  - Zuweisung und Verwaltung von 16-Bit Kurzadressen
  - Keine Verwendung von IP-Adressen!
- Konfiguration der Geräte
  - Rollenzuweisung
    - Koordinator, Endgerät, Router
- Netzwerkmanagement

Geräteklassen bzw. Rollen

- Reduced-Function Devices (RFD)
  - Implementieren nur nötigste Teile von IEEE 802.15.4
  - Keine direkte Kommunikation zwischen RFDs möglich
- Full-Function Devices (FFD)
  - Können auch als 802.15.4-PAN-Koordinator agieren
  - RFD-RFD-Kommunikation via FFDs (store-and-forward)

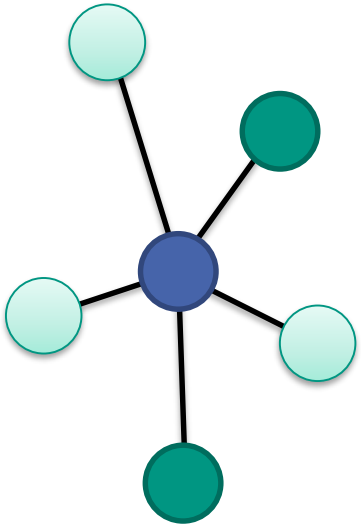
IEEE 802.15.4

ZigBee

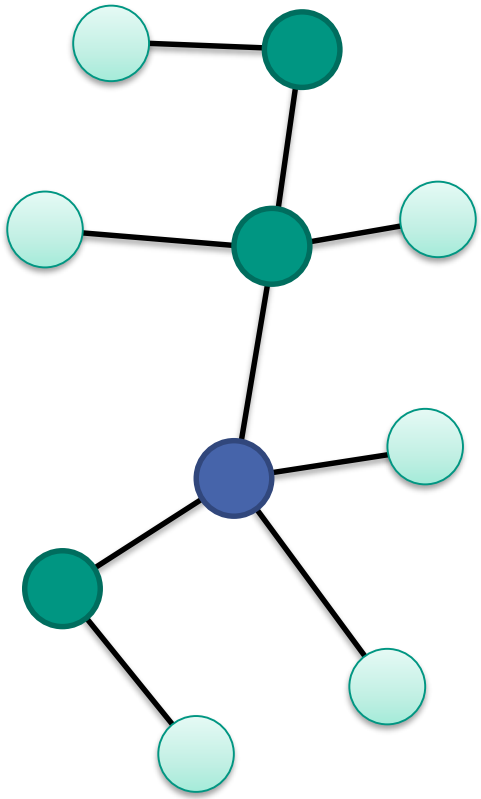
- Koordinator (FFD)
  - Übernimmt die Verwaltung des ZigBee-Netzwerks
  - Beeinflusst Netztopologie, Funkkanal etc.
- Router (FFD)
  - Kann Dateneinheiten im Netzwerk weiterleiten
- Endgerät (RFD oder FFD)
  - Gerät ohne Weiterleitungsfähigkeiten

Unterstützte Netztopologien

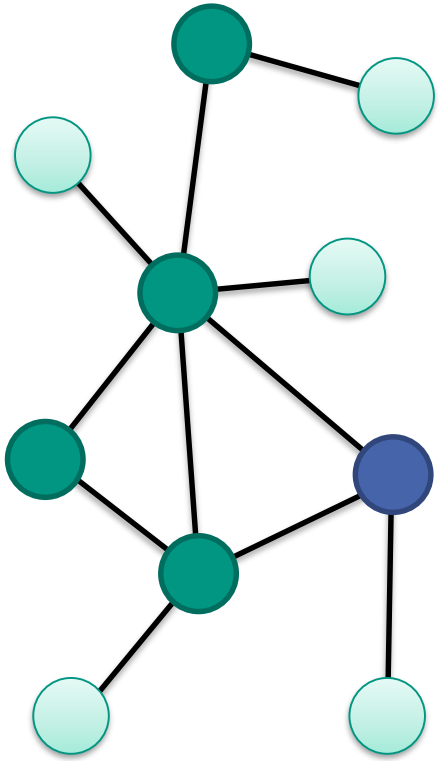
Stern
(auch bei 802.15.4)



Baum



Mesh
(802.15.4: Ohne Routing)



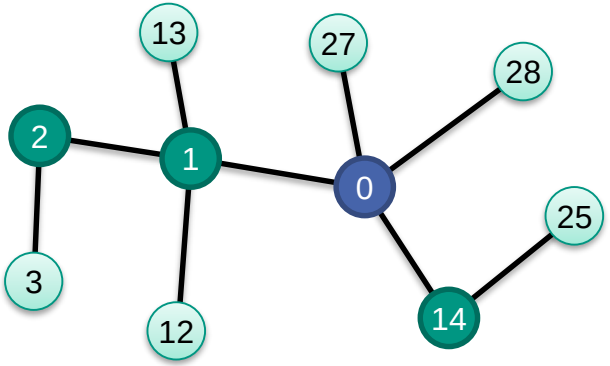
Koordinator (FFD)

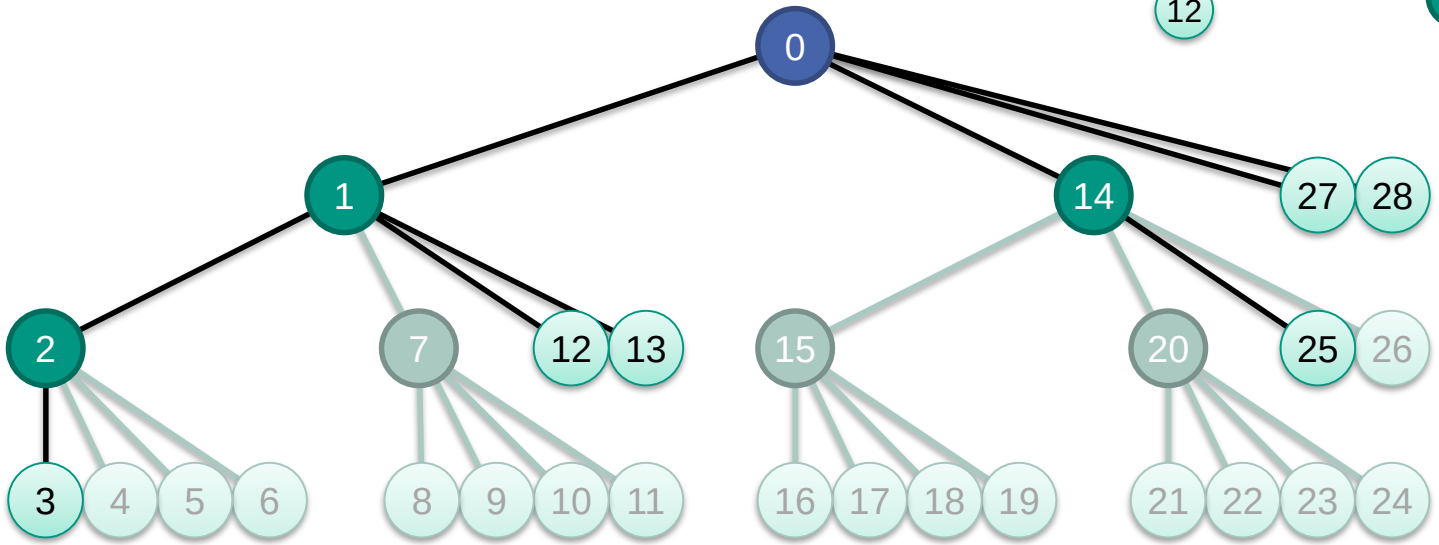
Router (FFD)

Endgerät (RFD/FFD)

Routing – Baumtopologie

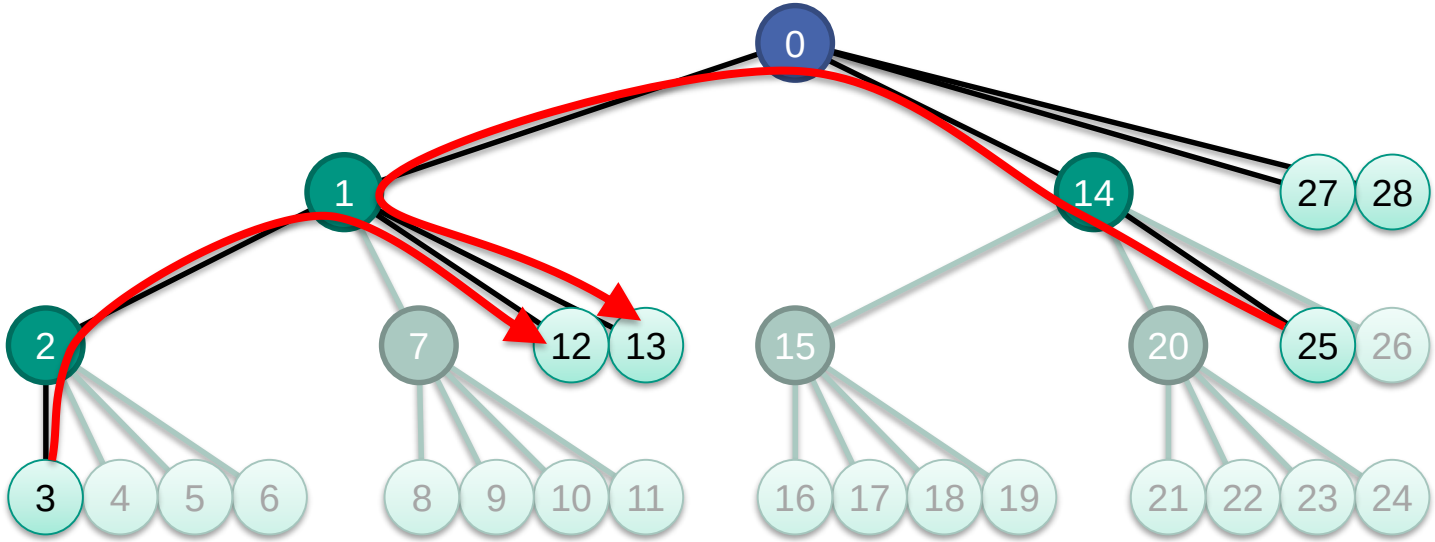
- Statisches Adressierungsschema
  - Sieht für jeden Router einen eigenen Adressbereich vor
    - → Implizite Baumstruktur im Adressraum
- Parameter
  - Max. Tiefe des Baums – Hier 3
  - Max. Anzahl an Kindknoten – Hier: 4
  - Max. Anzahl an Kind-Routern – Hier 2





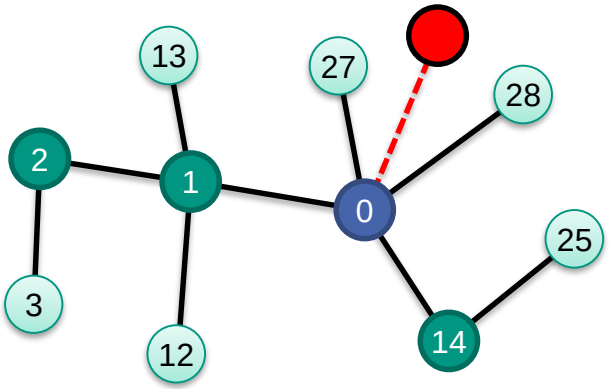
Routing – Baumtopologie

- Adressen werden direkt zum Routing genutzt
  - Router kennen Distanz zum Koordinator und Parameter des Adressbaums
    - Routingentscheidung lokal mit wenigen Rechenoperationen möglich
  - Falscher Teilbaum: Weiterleitung aufwärts Richtung Koordinator
  - Richtiger Teilbaum: Weiterleitung abwärts Richtung Ziel

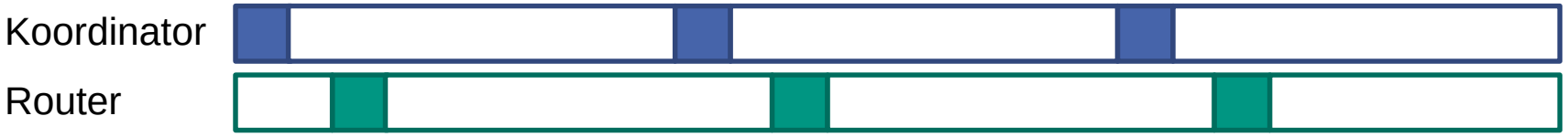


Routing – Baumtopologie

- Vorteil
  - Keine Pfadsuche
  - Keine Zustandshaltung (Routingtabellen!)
- Nachteile
  - Ausfallanfällig, keine redundanten Pfade
  - Suboptimale Pfade
  - Keine Router mit freien Adressen in Reichweite → Kein Beitritt zum Netz möglich



- Weitere Eigenschaften
  - 802.15.4 Beacon-Mode möglich (duty-cycling für alle Knoten)
    - Koordinator gibt Beacon-Takt vor
    - Abhängige Router senden ihr Beacon zeitversetzt

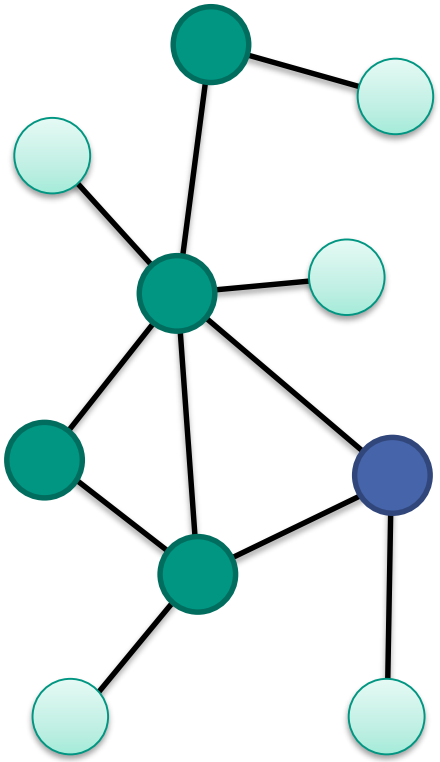


Routing – Meshtopologie

- Adressen werden zufällig gewählt
  - Koordinator stellt sicher dass keine Adresskollisionen auftreten
- Adressen haben keinen Bezug zur Topologie
  → Routingverfahren erforderlich
  - AODV (Ad-hoc on Demand Distance Vector Routing)

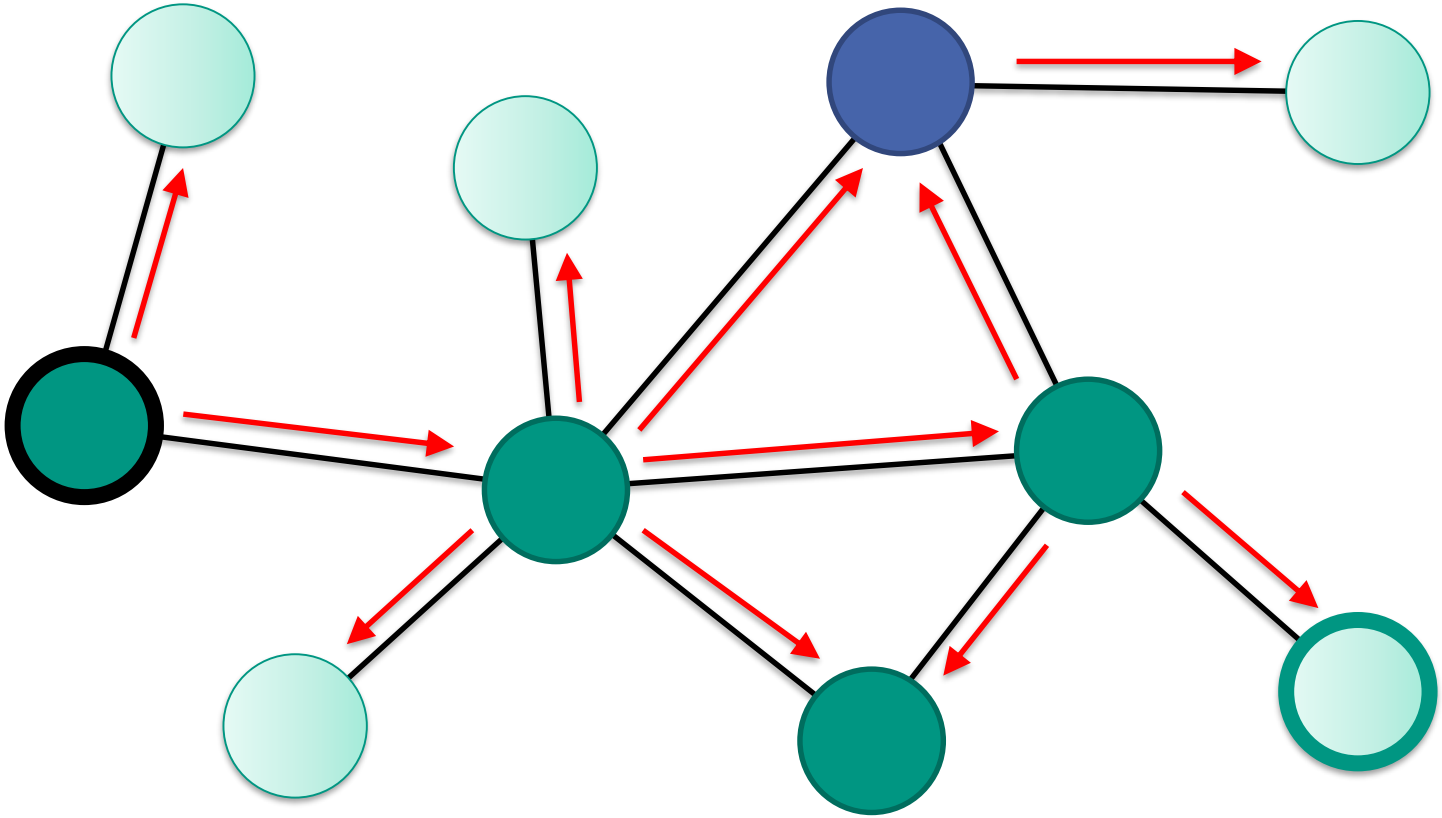
MK

- AODV ist vergleichsweise einfaches Protokoll
  - Routenanfrage wird durch das Netz geflutet und etabliert zur Quelle führende Pfade
  - Zielsystem beantwortet Anfrage und baut so einen bidirektionalen Pfad auf
  - Endgeräte beteiligen sich nicht am Algorithmus



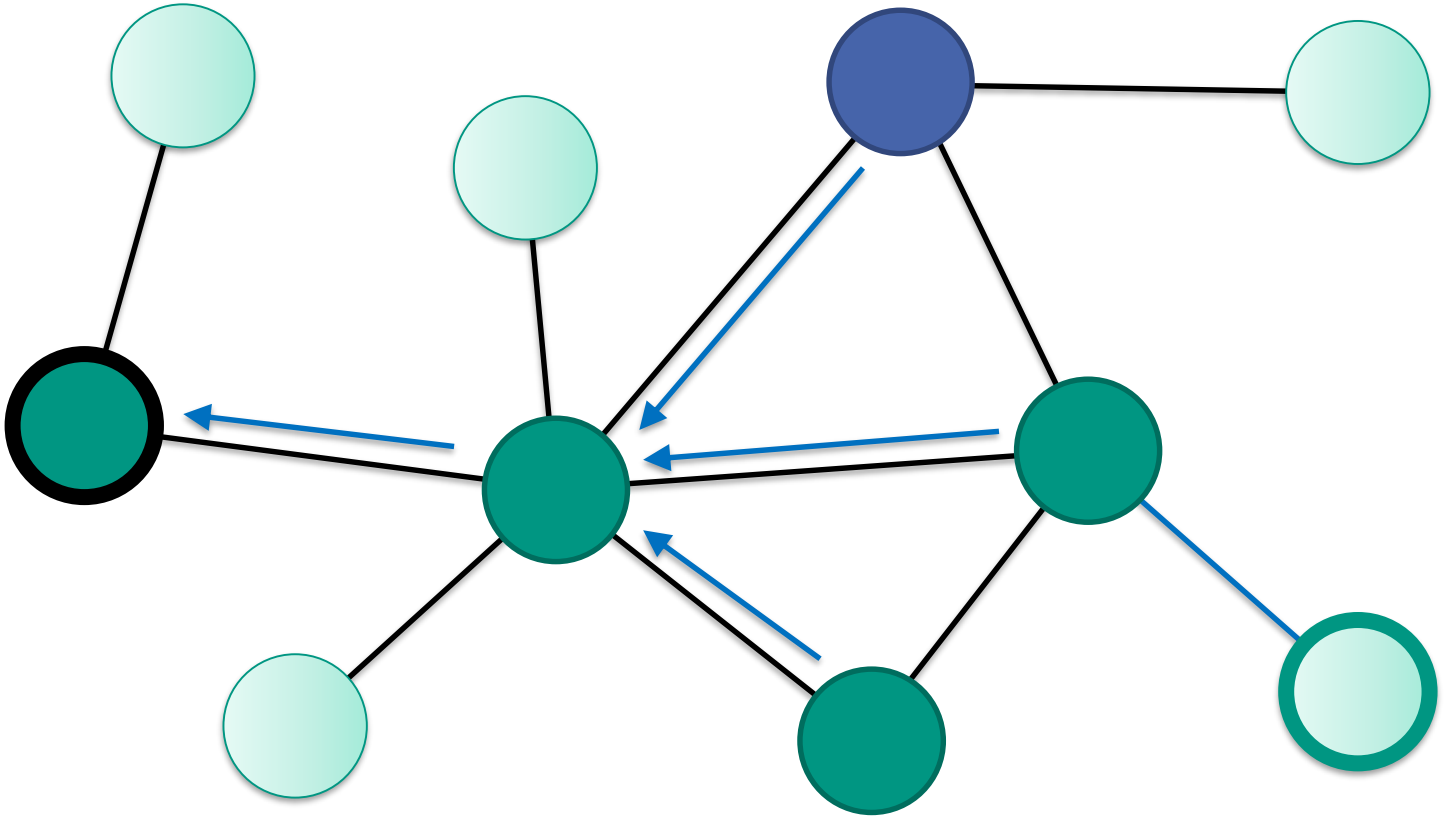
Routing – Meshtopologie

- Fluten der Routenanfrage



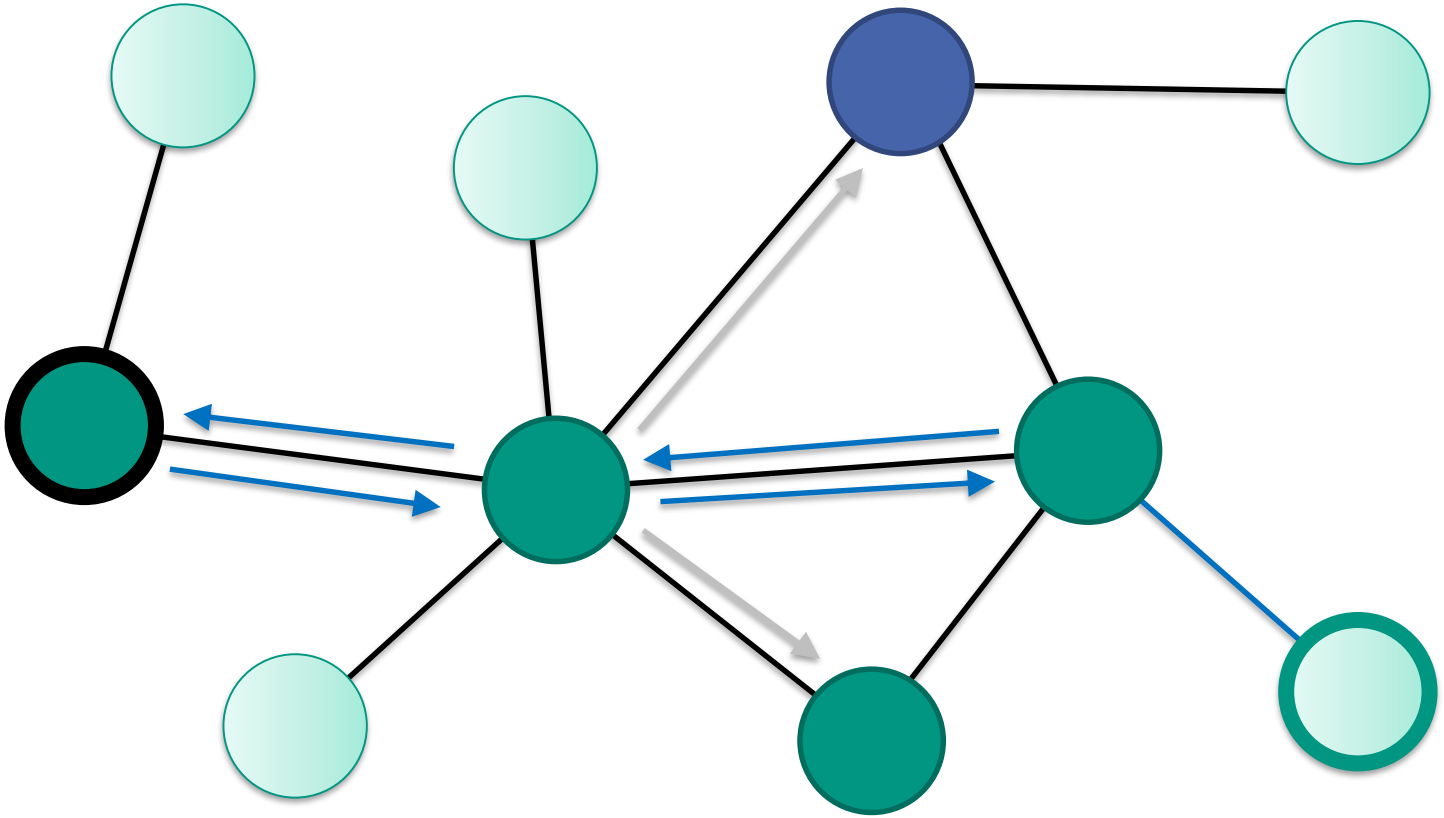
Routing – Meshtopologie

- Fluten der Routenanfrage
  - Ergebnis: Zur Quelle zeigende Pfade



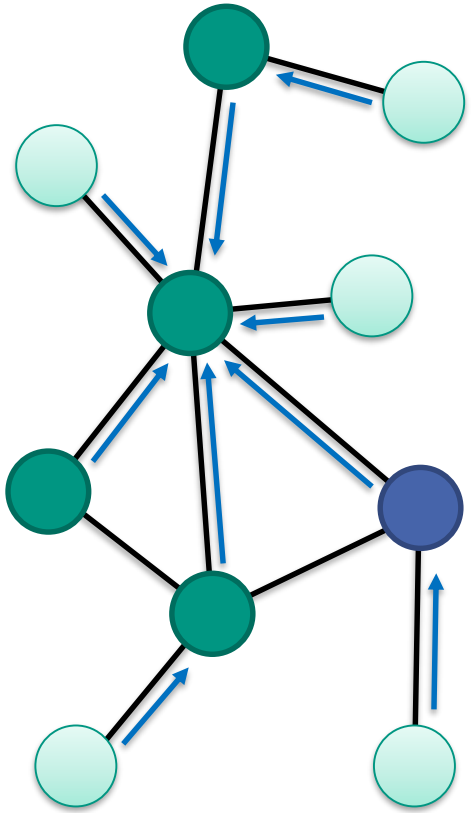
Routing – Meshtopologie

- Antwort vom Zielsystem etabliert einen Pfad
  - Hier: Zielsystem ist Endgerät → Router antwortet stellvertretend



ZigBee-Routing – Meshtopologie

- Vorteile
  - Robust/Selbstheilend
  - Bessere Pfade?
- Nachteile
  - Kein IEEE 802.15.4 Beacon-Mode
    - Koordinator und Router müssen permanent aktiv sein
    - → Ungeeignet für batteriebetriebene Geräte
  - Ressourcenbedarf
    - Zustandshaltung und Kommunikation
- Weitere Eigenschaften
  - Für Many-to-One-Routing (Concast) nutzbar
    - Senke etabliert zu ihr führende Pfade
    - Effizienter als viele einzelne Pfadsuchen
  - Pfadinformationen für Source-Routing nutzbar
    - Weniger Zustand in Zwischensystemen



Multicast-Routing

- Schicht der Anwendungsunterstützung: Gruppenverwaltung
- Netzwerk-Schicht: Weiterleiten der Dateneinheiten an eine Gruppe

- Sender ist Mitglied der Gruppe
  - Broadcast mit begrenzter Reichweite
    - Zieladresse ist Broadcastadresse
    - Gruppenmitglieder werten Dateneinheit aus
    - Nicht-Mitglieder leiten Dateneinheit weiter
      - Begrenzter Broadcast-Radius … Zähler wird erniedrigt

- Sender ist kein Mitglied der Gruppe
  - Routing zu Mitglied der Gruppe
    - Wie finden? Weganfrage wird durch Router geflutet
  - Von diesem Mitglied dann Weiterleitung wie oben

Varianten von ZigBee

- Standardisierung einer Reihe von Varianten, u.a.
  - ZigBee 2006
  - ZigBee 2007: Zwei Profile, ZigBee und ZigBee Pro
  - ZigBee 3.0: Weiterentwicklung auf Basis von ZigBee Pro
    - u.a. bessere Interoperabilität, zusätzliche Energiesparmechanismen
  - ZigBee IP: Basiert auf 6LoWPAN

ZigBee 2007 – Profile

- ZigBee
  - Netztopologie: Baum mit eingebetteter Adressierung
  - Statisches Routing
  - „Normale Sicherheit“
  - Reduzierte Funktionalität, einfache Verfahren
  - → Geringer Ressourcenbedarf

- ZigBee Pro
  - Netztopologie: Mesh, freie Adressvergabe
  - Beherrscht Concast, Multicast und Source-Routing
  - „Hohe Sicherheit“
  - → Erweiterter Funktionsumfang

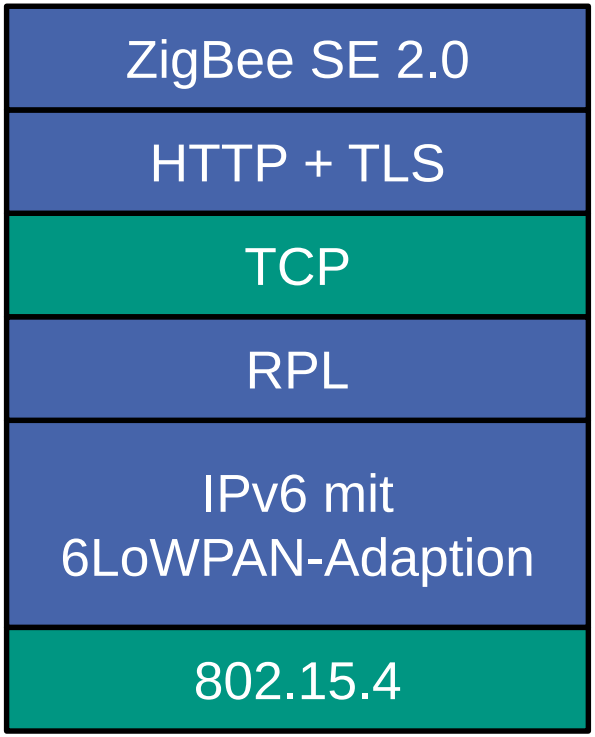
[ZigBee]

ZigBee IP

- Entstand im Zuge der Entwicklung des ZigBee Smart Energy 2.0 Profils
  - NIST-Vorgabe: Offener Standard, basierend auf IEEE/IETF-Standards

- Definiert einen Protokollstapel auf Basis etablierter Standards
  - 6LoWPAN und RPL
  - HTTP als Anwendungsschichtprotokoll erzwingt TCP als Transportprotokoll
  - TLS zur Absicherung

→ ZigBee Smart Energy 2.0 spezifiziert im wesentlichen Datenformate, Schnittstellen und Anwendungsverhalten



[ZigBIP]

ZigBee Security

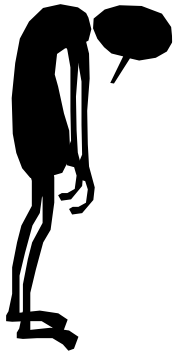
- Absicherung auf zwei Schichten möglich

[ZigBee]

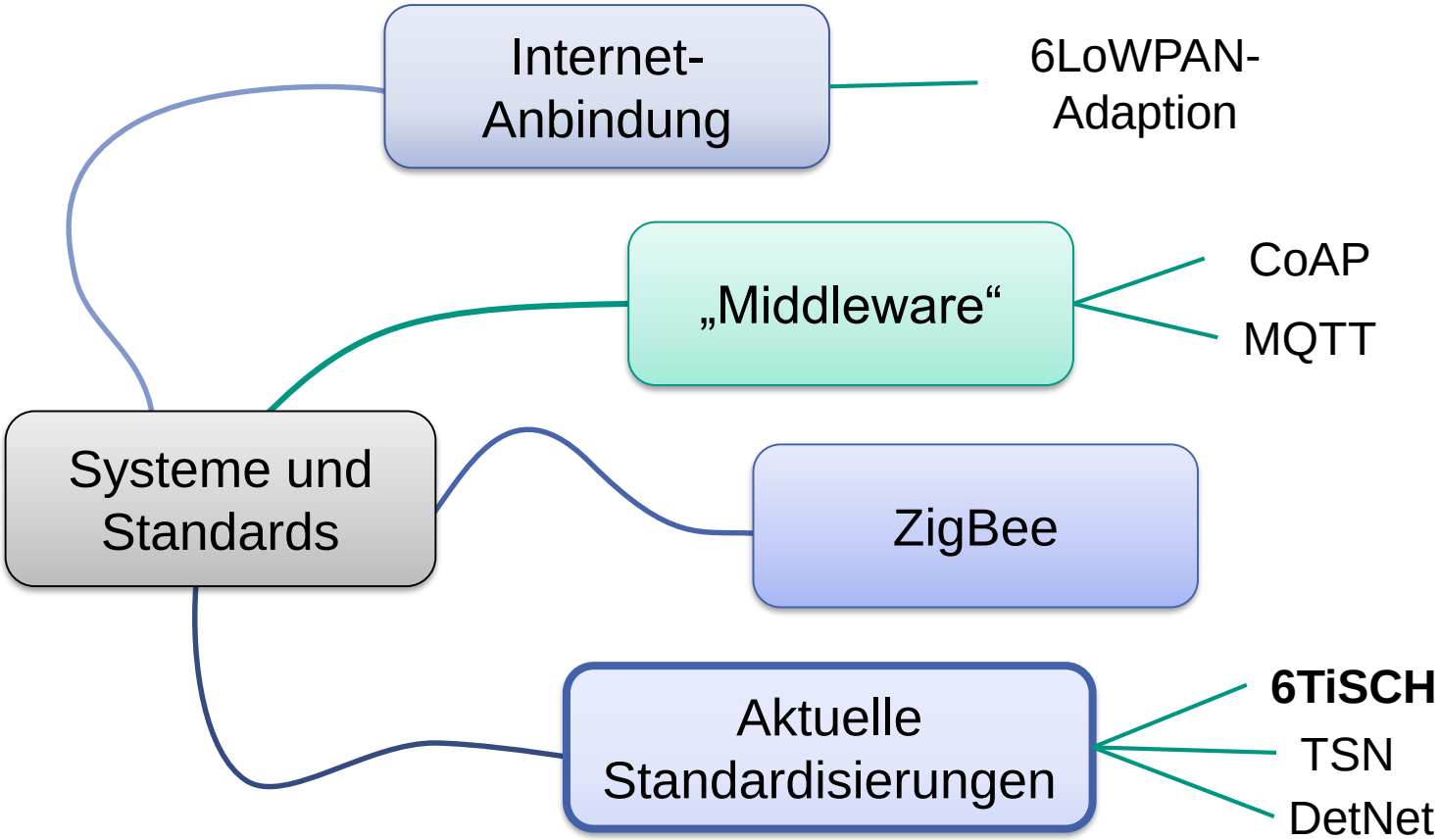
- Netzwerkschicht
  - Ein geteilter Schlüssel für das gesamte Netzwerk (Single Mission Key)
  - Schützt alle Übertragungen, insbesondere auch Broadcasts
- Anwendungsunterstützungsschicht
  - Eigene Schlüssel für jede Ende-zu-Ende-Verbindung (Linkschlüssel)
  - Nur auf Unicast-Kommunikation anwendbar

[Zill15]

- Schlüsselverwaltung erfolgt durch ein Trustcenter
  - Problem: Schlüsselverteilung bei initialem Verbindungsaufbau
    - Übertragung erfolgt im Klartext
  - Home Automation/Light Link Profile: Definieren Standardschlüssel
    - Bieten nicht mehr Sicherheit als die Übertragung im Klartext



Schwerpunkte des Kapitels im Überblick

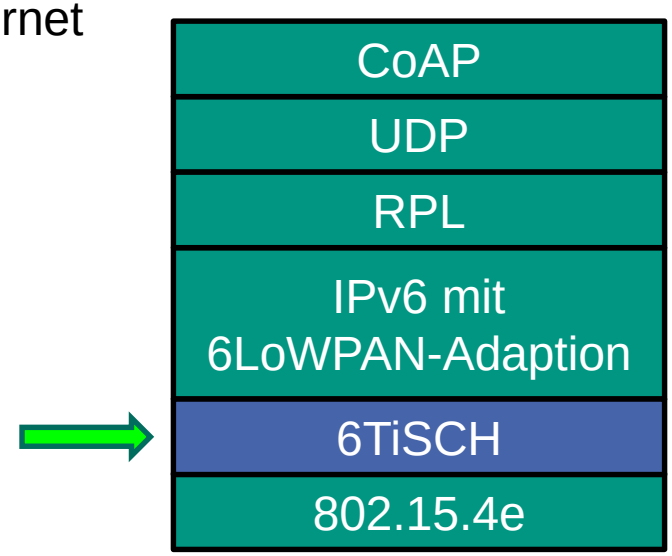


IEEE 802.15.4e TSCH im Gesamtkontext

- Medienzugriff mit IEEE 802.15.4e Time Slotted Channel Hopping
  - Betrachtet Kommunikation auf Schicht 2

Wie passt das zusammen?

- „Upper Layers“, also 6LoWPAN
  - … hier oberhalb von Schicht 2 gemeint
  - Konnektivität mit und über IP-basiertes Internet
- 6TiSCH im Kontext von 6LoWPAN
  - 802.15.4e auf Schicht 2
  - Umsetzung von Scheduling



6TiSCH

- 802.15.4e …
  - … beschreibt, wie Schicht 2 einem gegebenen Zeitplan folgt …
  - … nicht aber, wie Zeitplan erstellt und im Betrieb angepasst wird

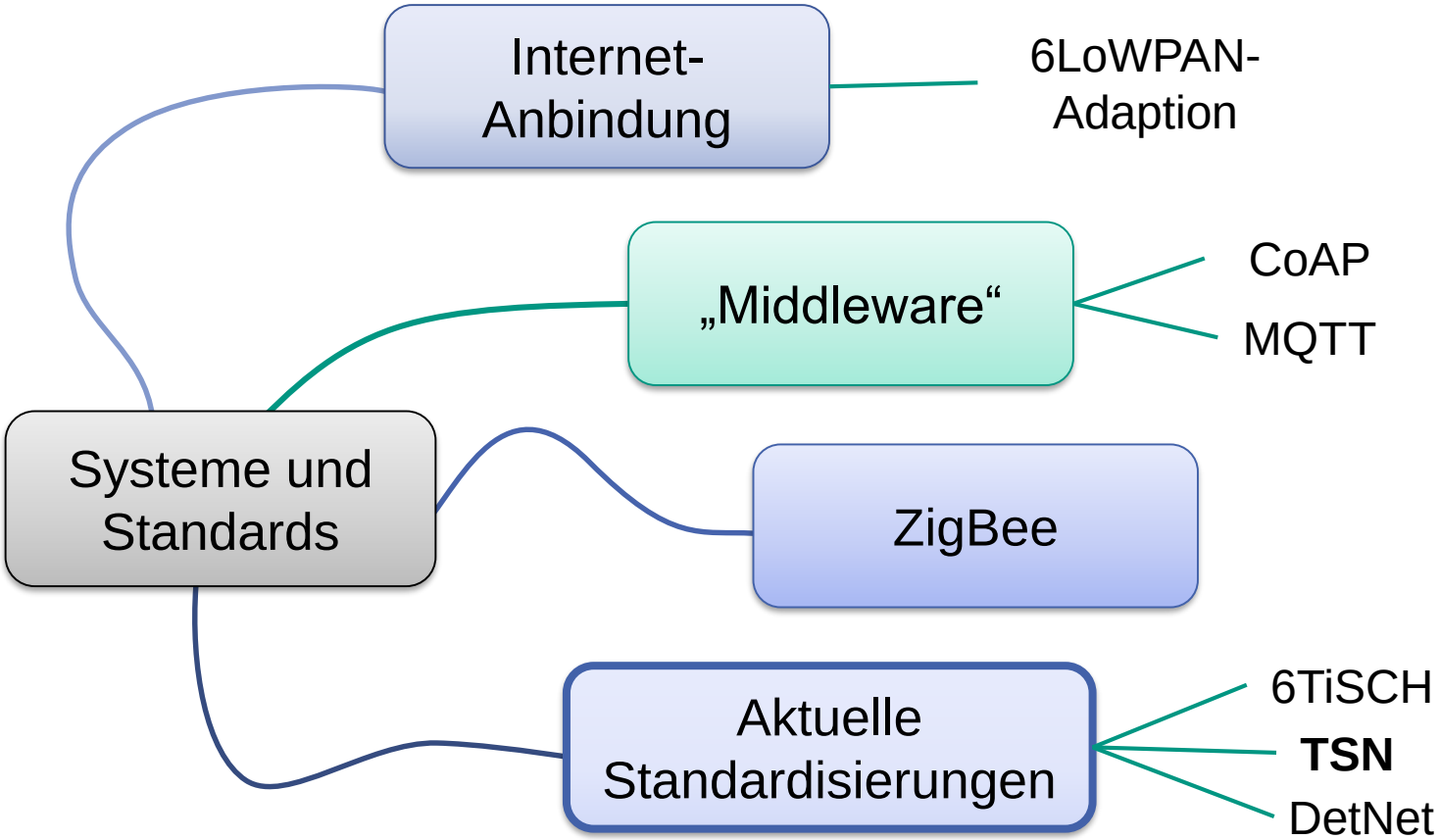
- IETF Working Group um 6LoWPAN über 802.15.4e einzusetzen
  - 6TiSCH: IPv6 over the TSCH mode of IEEE 802.15.4e
    - Anpassung der 6LoWPAN-Architektur an Annahmen in 802.15.4e
      - Unter anderem Synchronisierung der Geräte, Signalisierung …
    - Bisher noch kein RFC fertiggestellt, lediglich im draft-Stadium


[Thub15]

6TiSCH

- Aufgaben aus verschiedenen Teilbereichen
  - Nachbarn erkennen (Erweiterungen zu RPL und 6LoWPAN-ND)
  - Zeitpläne erstellen und anpassen
    - 6TiSCH betrachtet aktuell nur statische Zeitpläne
  - Multi-Hop Pfade von RPL auf Zeitpläne abbilden
  - Auf Toplogieänderungen reagieren
- 6TiSCH-Netz
  - Kleine drahtlose Mesh-Netze, verbunden über schnelles Backbone-Netz
- Scheduling
  - Nicht von 6TiSCH standardisiert
  - Scheduling-Instanz kann zentral oder dezentral sein
- Hohe Anforderungen an Synchronisation

Schwerpunkte des Kapitels im Überblick



Time-Sensitive Networking

- 6TiSCH adressiert robuste und planbare Medienzugriffsmechanismen für WPANs

- IEEE Time-Sensitive Networking verfolgt anderen Ansatz
  - Echtzeit- und Verfügbarkeitsgarantien mit IEEE 802.x Technologie
  - Schwerpunkt: Ethernet (802.3)
  - Ansätze sind auch auf WLAN (802.11) etc. übertragbar

- Keine Änderung des Medienzugriffsverfahrens
  - Best-Effort-Kommunikation neben Echtzeitkommunikation möglich
  - Einhaltung von Garantien durch geeignetes Scheduling
  - Übergeordnete Planung/Koordination der Mediennutzung

  → Gemeinsamkeit zu 6TiSCH

Grundlegendes Konzept

- Geringe Latenzen und hohe Zuverlässigkeit
  - → Keine Paketverluste in Zwischensystemen zulässig
  - → Keine großen Puffer in Zwischensystemen möglich

- Explizite Reservierung von Ressourcen (Bandbreite)
  - Keine Überlastsituationen → keine Pufferung nötig
  - Leistungsgarantien nur für akzeptierte Datenströme
  - Datenströme mit definierter Charakteristik (max. Datenrate, Burstfreiheit)
  - Abgeschlossenes Teilnetz, kooperierende Zwischen- und Endsysteme
  - Anderer Datenverkehr nur mit Best-Effort-Garantien

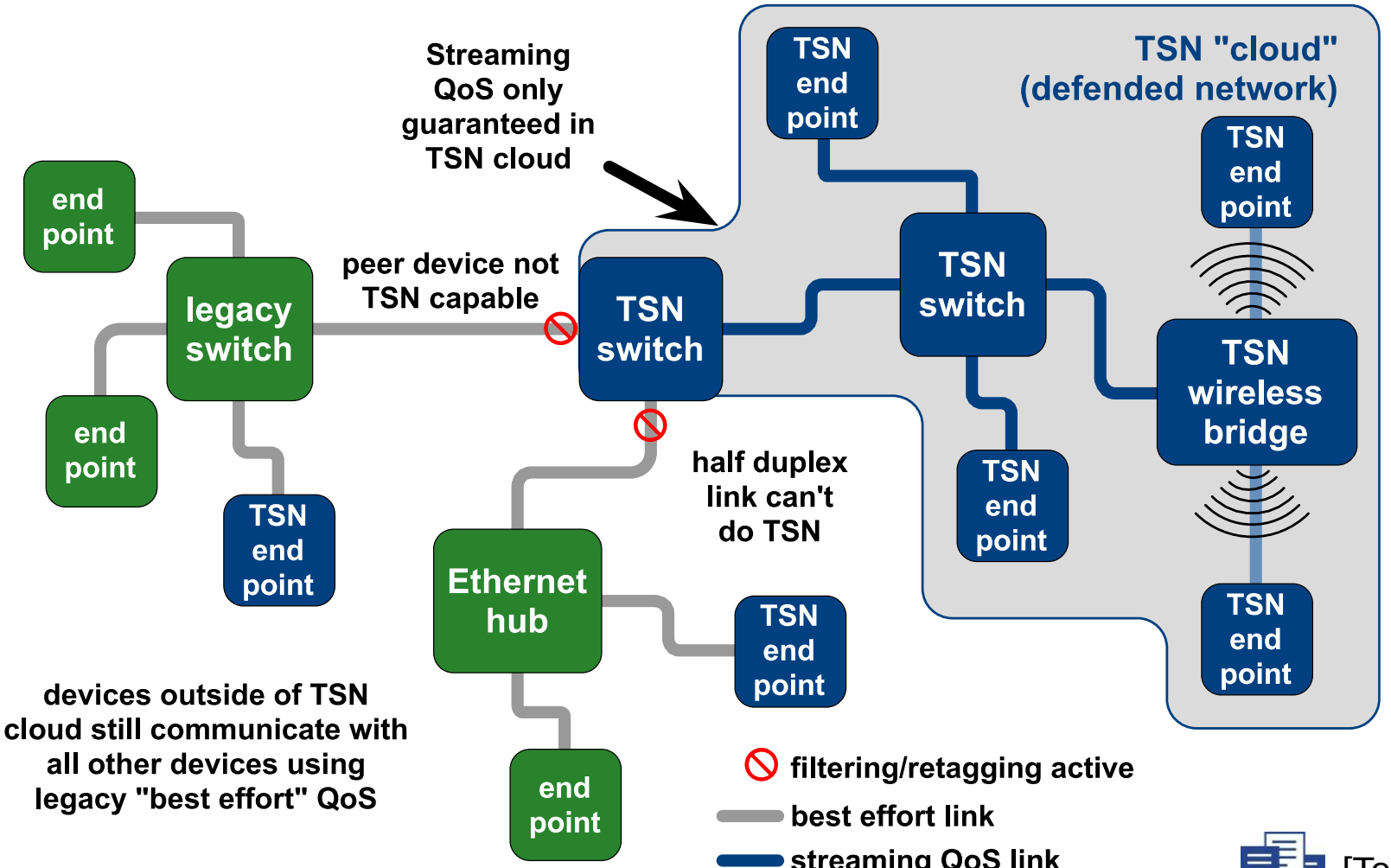
- Die Vermeidung von Überlastsituationen ermöglicht deterministische Latenzen von unter 50 ms bzw. 2 ms über 7 Hops.
  - Ansätze für Latenzen deutlich unter 1 ms existieren

[Tee+13]

Streams

- TSN basiert auf dem Konzept von Streams
  - „A unidirectional flow of data (e.g., audio and/or video) from a Talker to one or more Listeners”
- Stream-Eigenschaften
  - Maximale Datenrate
  - Anforderungen (Latenz, Jitter, Abwesenheit von Verlusten)
  - …

TSN innerhalb eines LAN



[Tee+13]

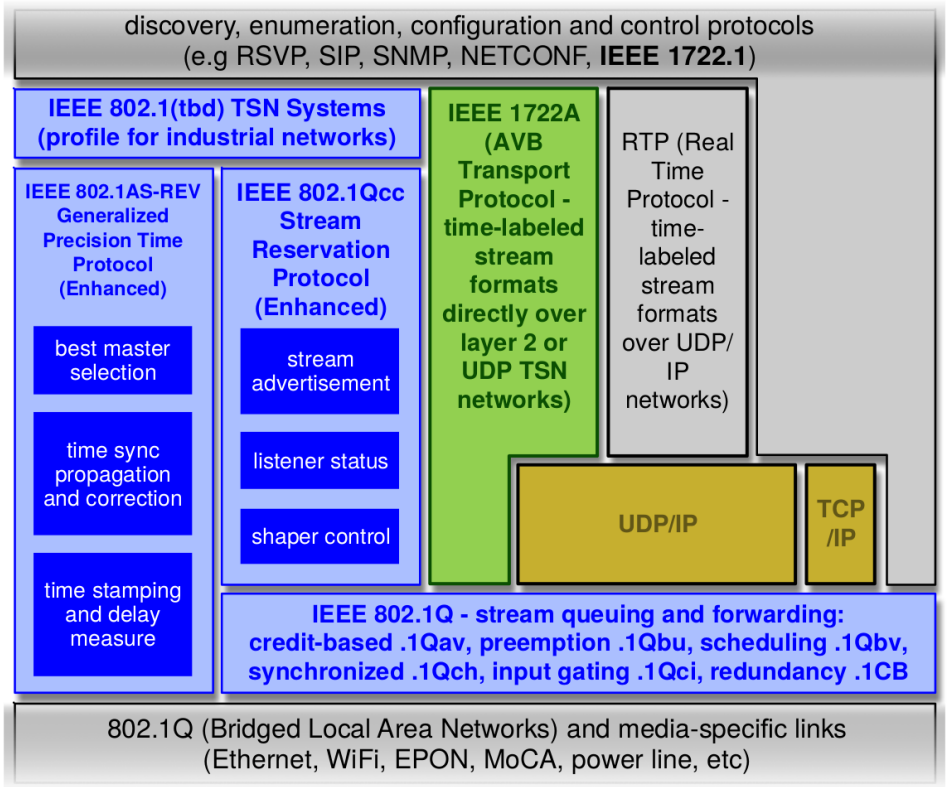
TSN: Hintergrund

- Spezifikation in IEEE-Standard 802.1Q
  - Viele Entwürfe, welche noch nicht Teil des Haupt-Standards sind
- Ursprünglich als „AVB“ (Audio-Video-Bridging) zur Wiedergabe von interaktiven Videos auf verteilten Systemen
  - Zeitsynchronisation zwischen Endgeräten zum gleichzeitigen Abspielen
  - Maximale Verzögerung zur Interaktion

- Mittlerweile Kandidat für industrielle Anlagen mit Echtzeitanforderungen
  - Latenzgarantien
  - Verlustfreiheit

TSN: Bausteine

- Grundlage: Weiterleitungsmechanismen auf Schicht 2
- Verschiedene Bausteine
  - Zeitsynchronisation (802.1AS)
  - Stream Reservation Protocol (802.1Qat)
  - Forwarding/Queueuing (802.1Qav)
  - Multipath-Routing uvm.
- Nichtdeterminismus von Medien/-zugriffsverfahren wird nicht berücksichtigt!
  - Zeitliche Koordinierung bei drahtlosen (geteilten) Medien erforderlich → TSCH!



[Teen15]

TSN: Stream Reservation Protocol (SRP)

- Protokoll, zur Reservierung von Bandbreite zwischen Sender („Talker“) und Empfänger („Listener“)

- Grundlegende Idee
  - Talker sendet Talker-Advertisement-Nachricht für einen Stream
    - Beinhaltet Stream-Spezifikation (Bandbreite etc.) und „akkumulierte Latenz“
      - jeder Switch auf dem Weg addiert seine maximale Latenz
    - Jeder Router schickt erste Talker-Aadvertisement-Nachricht an jeden Link, der noch genug Bandbreite frei hat
    - Jedes Gerät empfängt das Advertisement, wenn der Stream möglich ist, und sieht die Latenzgarantie
  - Listener kann nun Listener-Nachricht senden, um den Stream zu empfangen
    - Jeder Hop leitet sie in die Richtung weiter, aus der die entsprechende Talker-Advertisement-Nachricht kam, reserviert die Bandbreite, und merkt sich, in welche Richtung Rahmen des Streams weitergeleitet werden müssen
    - Talker beginnt zu senden, sobald eine Listener-Nachricht empfangen wurde

  - Zusätzlich Nachrichten für Reservierungsfehler, Abmelden von Streams, …

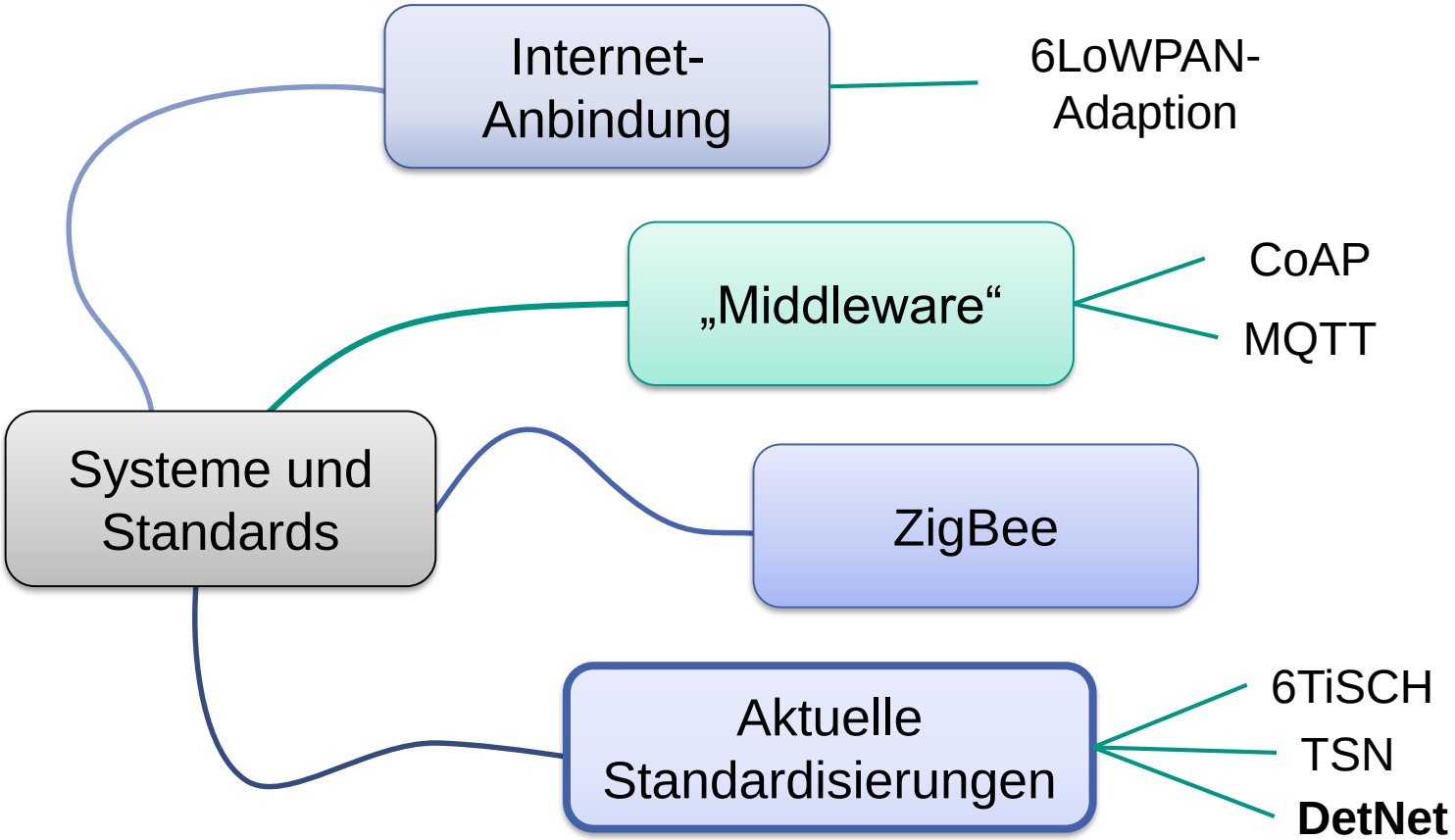
TSN: Ingress Policing und Traffic Shaping

- Ingress Policing: Verwerfen von nicht reserviertem Traffic
  - Traffic wird auf „Anzahl Rahmen oder Bytes pro Zeiteinheit“ reserviert
    - Erlaubt Bursts in geringem Rahmen
  - Verwerfen von eingehenden Rahmen bei Verletzung der Vereinbarung

- Traffic Shaping: Priorisieren und Formen von Echtzeittraffic
  - Viele Warteschlangen in Switches pro Ausgangsport
  - Prioritäten zwischen Warteschlangen
  - Zusätzliche Eigenschaften von Warteschlangen:
    - Credit-Based Shaper: Warteschlange sendet nach Leaky-Bucket-Prinzip
      - Wandelt Bursts in gleichförmige Streams, aber kann bei Bedarf ohne Einbußen höher-priorisiertem Verkehr Vorrang geben
    - Time-Aware Shaper: Warteschlange sendet immer zu festen Zeiten
      - Deterministische Latenz
      - Synchronisation mit anderen Geräten erlaubt sehr schnelle Durchleitung

- Frame Preemption: Pausieren von Best-Effort-Rahmen
  - Verhindert, dass große Best-Effort-Rahmen stören

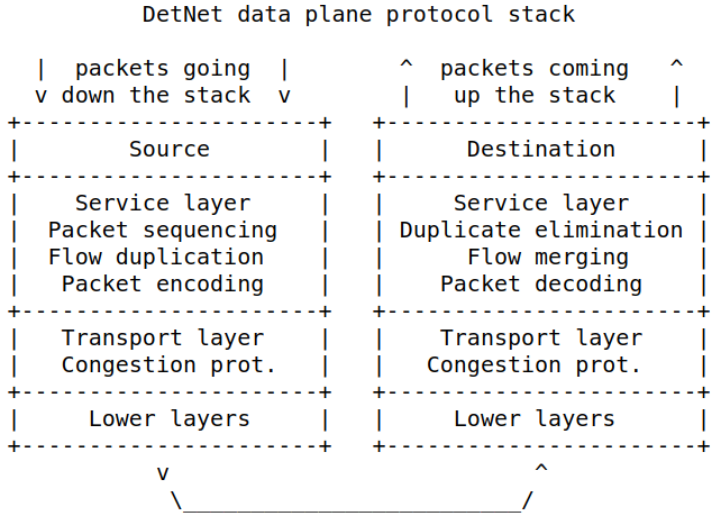
Schwerpunkte des Kapitels im Überblick



IETF Deterministic Networks (DetNet)

- TSN: Layer-2-Mechanismen für lokale Netze
- Benötigt: Layer-3-Mechanismen für verbundene, heterogene Netze

DetNet Services

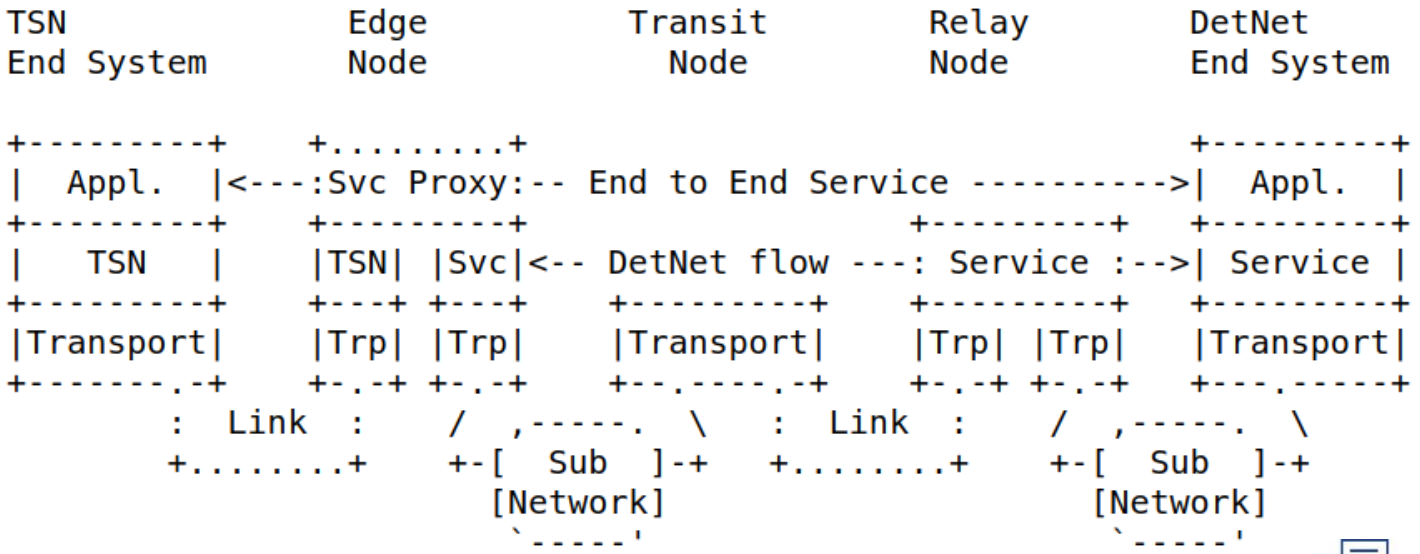


- Service Layer: Fehlertoleranz durch Replikation/Elimination
  - Pakete werden gleichzeitig über mehrere Pfade geleitet
  - Vermeidet Paketverlust oder Verzögerung, wenn einer der Pfade ausfällt
- Transport Layer: Latenz und Überlast-Verluste
  - Ermöglicht Garantien für Latenz und Jitter
  - Vermeidet Paketverluste durch Überlastsituationen
- Flows (= TSN-Streams) können beide Service und/oder Transport Layer nutzen

[DNArch]

DetNet Nodes

- Transit Node: Nur DetNet Transport Layer
- Relay Node: Transport- and Service-Layers
  - Kann also Pakete Replizieren/Eliminieren
- Edge Node: Kann als Proxy agieren
  - z.B. zwischen TSN und DetNet übersetzen

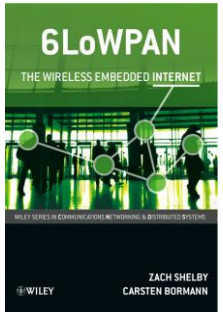


[DNArch]

Literatur

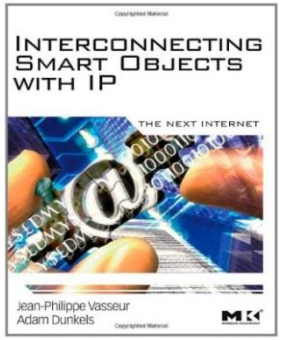
- *6LoWPAN: The Wireless Embedded Internet*
  - Z. Shelby, C. Bormann; Wiley & Sons

[ShBo09]



- Interconnecting Smart Objects with IP: The Next Internet
  - J-P. Vasseur und Adam Dunkels; Morgan Kaufmann

[VaDu10]



- RFCs sind zu finden unter www.ietf.org
  - 6LoWPAN: http://tools.ietf.org/wg/6lowpan/
  - IPv6 over Networks of Resource-constrained Nodes (6lo): https://datatracker.ietf.org/wg/6lo/
  - RPL: http://tools.ietf.org/wg/roll/
  - CoAP: http://tools.ietf.org/wg/core
  - 6TISCH: https://datatracker.ietf.org/wg/6tisch
  - DetNet: https://tools.ietf.org/wg/detnet/

Literatur

[BoCS12] C. Bormann, A.P. Castellani, Z. Shelby; **CoAP: An Application Protocol for Billions of Tiny Internet Nodes**, Internet Computing, IEEE, vol.16, no.2, pp.62-67, März-April 2012

[DNArch] Finn, N., Thubert, P., Varga, B. and Farkas, J., **Deterministic networking architecture**, draft-ietf-detnet-architecture-03. Internet-Draft, IETF.

[DWVT14] D. Dujovne, T. Watteyne; X. Vilajosana, P. Thubert; **6TiSCH: deterministic IP-enabled industrial internet (of things),** Communications Magazine, IEEE, vol.52, no.12, pp.36-41, Dezember 2014

[EnOcean] **EnOcean Radio Protocol 2 v1.0**, EnOcean GmbH, September 2013

[KrKo14] M. Krause M., R. Konrad; **Drahtlose ZigBee-Netzwerke**, Springer Vieweg, 2014

[MQTT] A. Banks, R. Gupta; **MQTT Version 3.1.1 Errata 01**, OASIS Standard, Dezember 2015

[MQTT-SN] A. Stanford-Clark and H.L. Truong; **MQTT For Sensor Networks (MQTT-SN) Protocol Specification Version 1.2**, IBM, November 2013

[Pal+13] M.R. Palattella et al; **Standardized protocol stack for the internet of (important) things**, Communications Surveys & Tutorials, IEEE, vol.15, no.3, pp.1389-1406, 3. Quartal 2013

[ShBo09] Z. Shelby, C. Bormann; ***6LoWPAN: The Wireless Embedded Internet***, Kapitel 4.2. Wiley, 2009, ISBN 978-0-470-74799-5

Literatur

[Teen15] M. Teener; **A Time-Sensitive Networking Primer: Putting It All Together**. ISPCS 2015 Keynote

[Tee+13] M. Teener et al; **Heterogeneous Networks for Audio and Video: Using IEEE 802.1 Audio Video Bridging**, Proceedings of the IEEE, vol.101, no.11, pp.2339-2354, November 2013

[Thub15] P. Thubert; **An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4**, draft-ietf-6tisch-architecture-09, November 2015. Work in Progress

[VaDu10] J.-P. Vasseur, A. Dunkels; ***Interconnecting Smart Objects with IP – the next internet***, Kapitel 17. Morgan Kaufmann 2010, ISBN 978-0-12-375165-2

[ZCLib] ZigBee Standards Organization; **ZigBee Cluster Library Specification**, 2008

[ZigBee] ZigBee Standards Organization; **ZigBee Specification**, 2012

[ZigBHA] ZigBee Alliance; **Home Automation Public Application Profile**, 2010

[ZigBIP] ZigBee IP; **IEEE Adoption of Smart Energy Profile 2.0 Application Protocol Standard**, 2013

[Zill15] T. Zillner; **ZigBee Exploited – The good, the bad and the ugly**, 2015