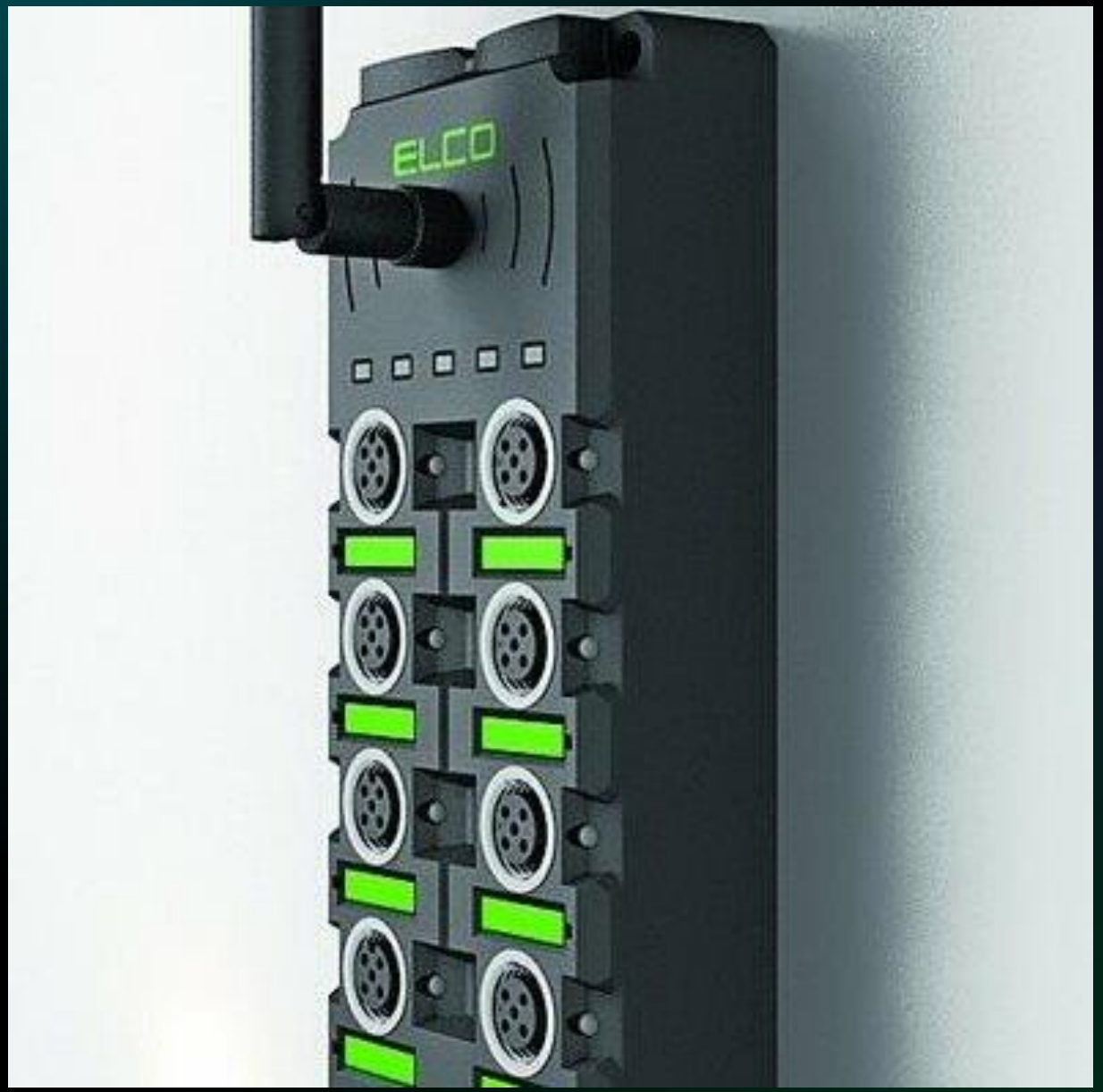
ELCO INDUSTRIAL AUTOMATION

MQTT & CoAP

KEINE NEUE COMPANY



Für alle, die das Promovideo gesehen haben ...

Kleine Werbeeinblendung

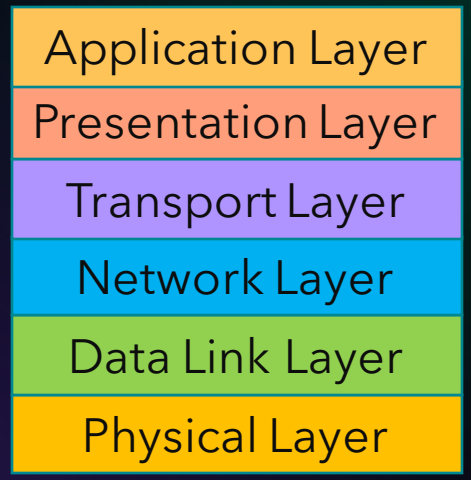
DER REFERENT

- Gelernter Informatiker
- Software Paläontologe
- Software Architekt
- Seit 2018 Arbeit am IoTHub

ELCO INDUSTRIE AUTOMATION GMBH

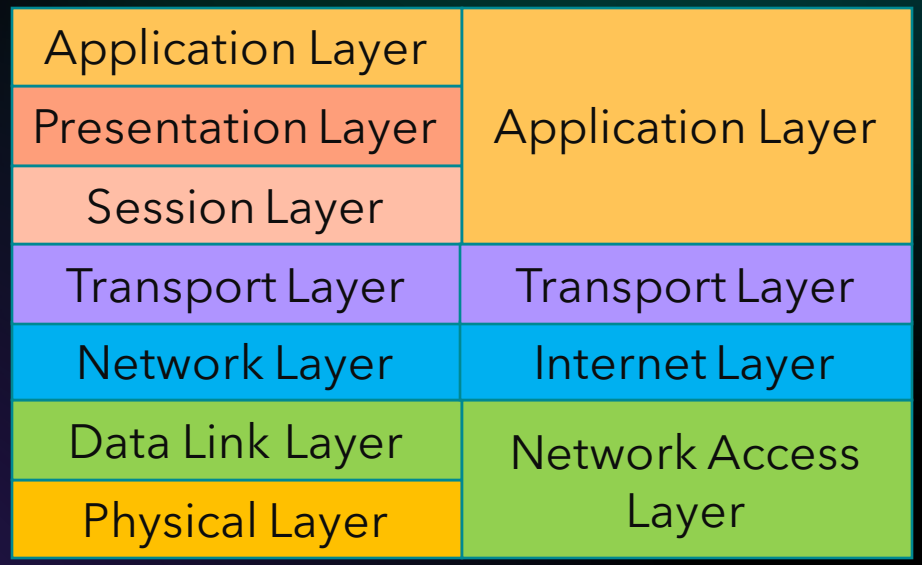
- Automation Produkte
- (Maschinen-) Daten sammeln
- Geräte steuern
- Visualisieren

OSI 7 Schichtmodell



TCP/IP Referenzmodell

- OSI 7 Schichtmodell



Web Stack Protokolle

- OSI 7 Schichtmodell
- TCP/IP Referenzmodell

Application Layer	Application Layer	Web Application
Presentation Layer		HTML, JSON, XML
Session Layer		HTTP(S), DNS, TLS, DHCP
Transport Layer	Transport Layer	TCP, UDP
Network Layer	Internet Layer	IPv6, IPv4, IPsec
Data Link Layer	Network Access Layer	Ethernet, WiFi, WLan, DSL, ISDN
Physical Layer		

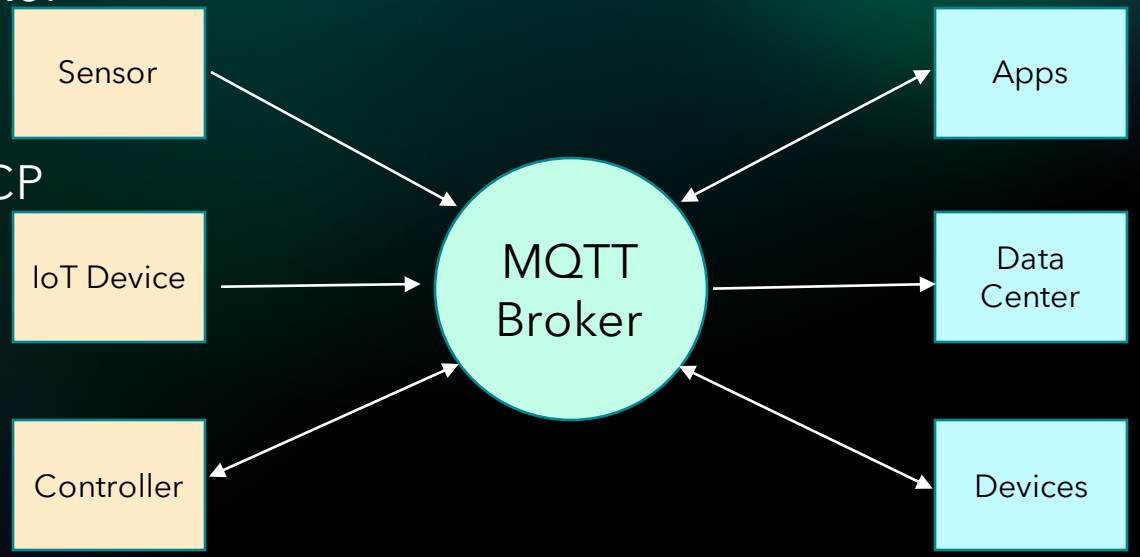
IoT Protokolle

- OSI 7 Schichtmodell
- TCP/IP Referenzmodell
- Web Stack Protokolle

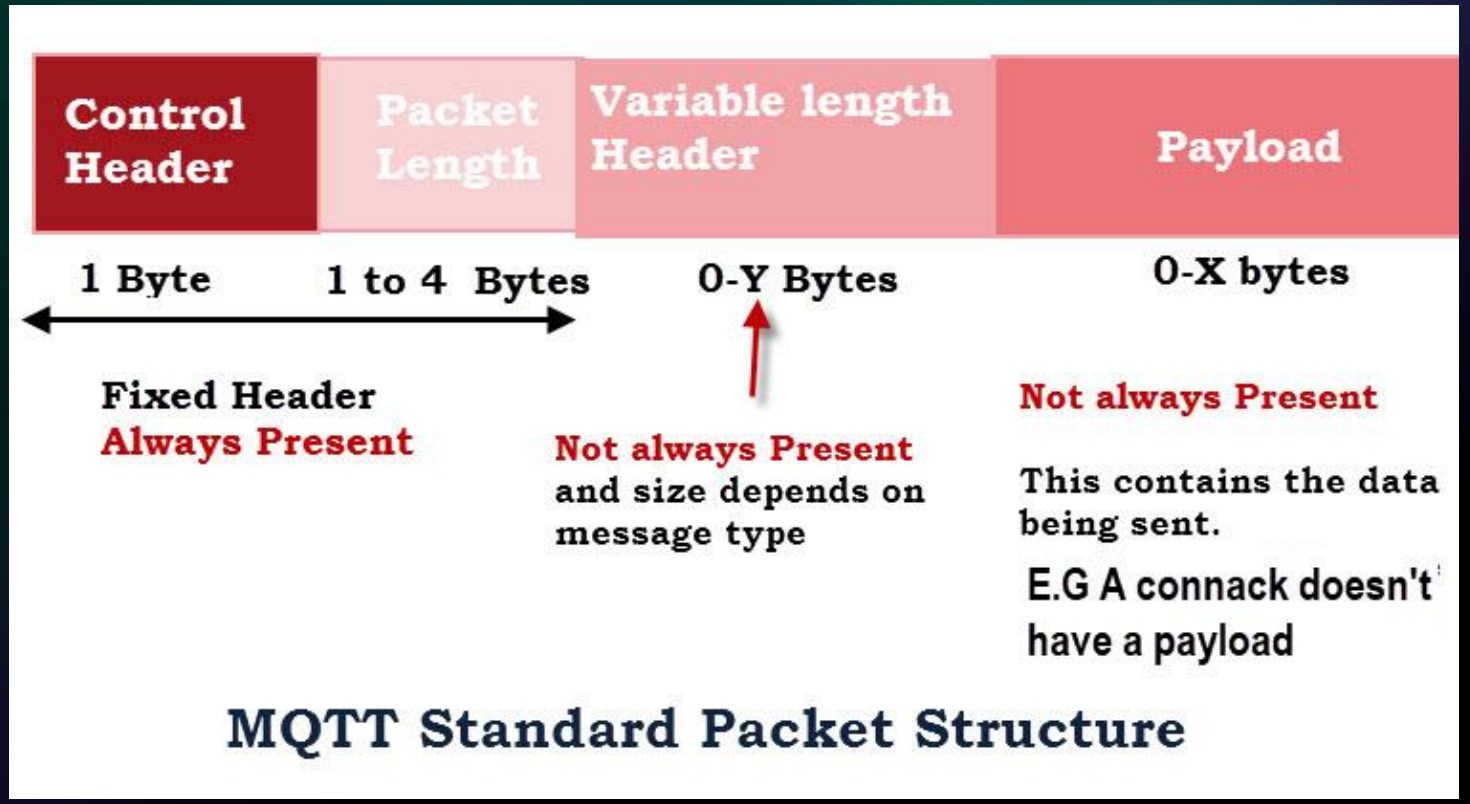
Application Layer	Application Layer	Web Application	Application
Presentation Layer		HTML, JSON, XML	Binary, JSON, CBOR
Session Layer		HTTP(S), DNS, TLS, DHCP	MQTT, SMQTT, AMQP, CoAP, XMPP, DDS
Transport Layer	Transport Layer	TCP, UDP	TCP, UDP, DTLS
Network Layer	Internet Layer	IPv6, IPv4, IPsec	6LoWPAN, 6TiSCH, 6Lo, IP..., RPL, CORPL, CARP
Data Link Layer	Network Access Layer	Ethernet, WiFi, WLan, DSL, ISDN	IEEE 802.15.4e, Z-Wave, Bluetooth, Zigbee, Dash7, HomePlug, G.9959, LTE-A, LoRa
Physical Layer			

MQTT

- Für kleine Geräte
- Verwendet einen Broker
- Austausch via Topics
- Kommunication via TCP
- Sicherheit via TLS, Username/Password
- DoS Level 0-2
- Last Will Nachricht



MQTT Message



Code Beispiel github.com/eclipse/paho.mqtt.golang

```
var messagePubHandler mqtt.MessageHandler = func(client mqtt.Client, msg mqtt.Message) {
    fmt.Printf("Received message: %s from topic: %s\n", msg.Payload(), msg.Topic())
}

var connectHandler mqtt.OnConnectHandler = func(client mqtt.Client) {
    fmt.Println("Connected")
}

var connectLostHandler mqtt.ConnectionLostHandler = func(client mqtt.Client, err error) {
    fmt.Printf("Connect lost: %v", err)
}
```

Code Beispiel github.com/eclipse/paho.mqtt.golang

```
func main() {
    var broker = "broker.emqx.io"
    var port = 1883
    opts := mqtt.NewClientOptions()
    opts.AddBroker(fmt.Sprintf("tcp://%s:%d", broker,
   port))
    opts.SetClientID("go_mqtt_client")
    opts.SetUsername("emqx")
    opts.SetPassword("public")
    opts.SetDefaultPublishHandler(messagePubHandler)
       opts.OnConnect = connectHandler
       opts.OnConnectionLost = connectLostHandler
       client := mqtt.NewClient(opts)
       if token := client.Connect(); token.Wait()
   && token.Error() != nil {
           panic(token.Error())
       }
       sub(client)
       publish(client)
       client.Disconnect(250)
}
```

```
func publish(client mqtt.Client) {
    num := 10
    for i := 0; i < num; i++ {
        text := fmt.Sprintf("Message %d", i)
        token := client.Publish("topic/test", 0, false,
   text)
        token.Wait()
        time.Sleep(time.Second)
    }
}

func sub(client mqtt.Client) {
    topic := "topic/test"
    token := client.Subscribe(topic, 1, nil)
    token.Wait()
    fmt.Printf("Subscribed to topic: %s", topic)
}
```

Code Beispiel org.eclipse.paho.client.mqttv3

```
public class SimpleMqttCallBack implements MqttCallback {

  public void connectionLost(Throwable throwable) {

    System.out.println("Connection to MQTT broker lost!");

  }

  public void messageArrived(String s, MqttMessage mqttMessage) throws Exception {

    System.out.println("Message received:\t"+ new String(mqttMessage.getPayload()) );

  }



  public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {

  }

}
```

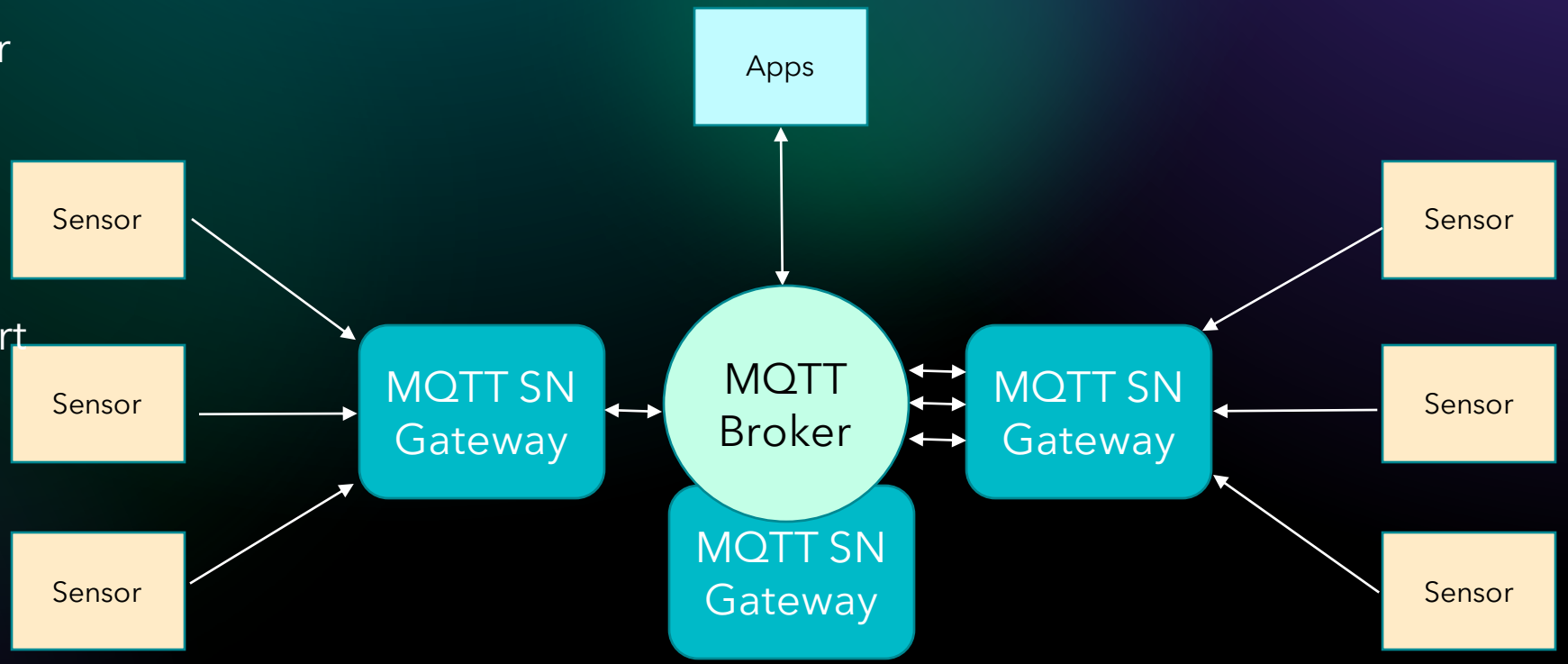
Code Beispiel org.eclipse.paho.client.mqttv3

```
public class Subscriber {

  public static void main(String[] args) throws
   MqttException {

    System.out.println("== START SUBSCRIBER ==");

    MqttClient client=new
   MqttClient("tcp://localhost:1883",
   MqttClient.generateClientId());

    client.setCallback( new SimpleMqttCallBack() );

    client.connect();

    client.subscribe("iot_data");

  }

}
```

```
public class Publisher {

  public static void main(String[] args) throws
   MqttException {

    String messageString = "Hello World from Java!";

    MqttClient client = new
   MqttClient("tcp://localhost:1883",
   MqttClient.generateClientId());

    client.connect();

    MqttMessage message = new MqttMessage();

    message.setPayload(messageString.getBytes());

    client.publish("iot_data", message);

    client.disconnect();

  }

}
```

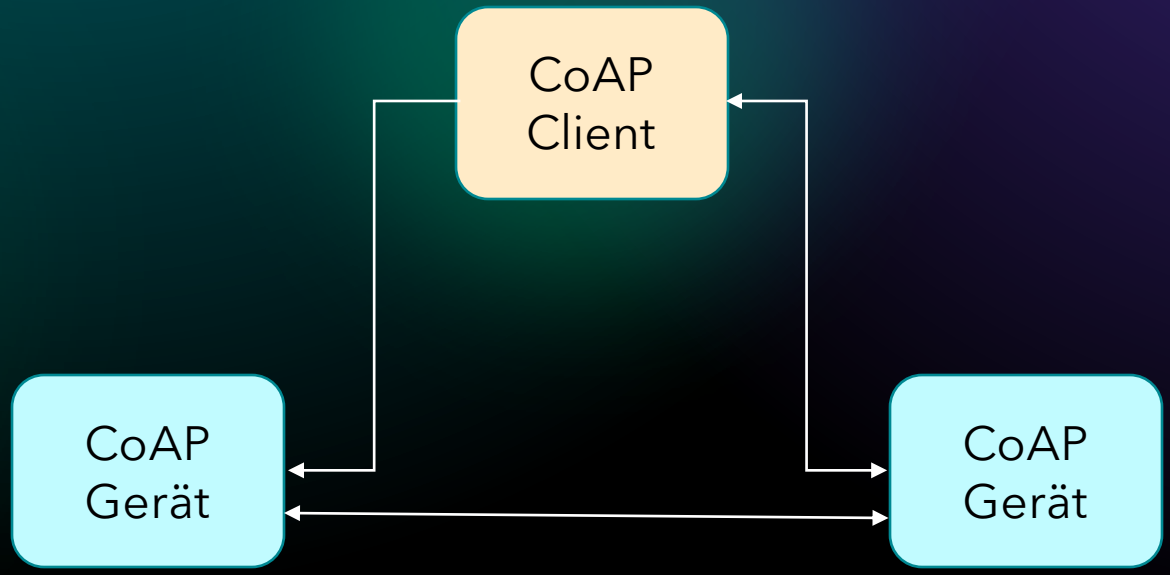
MQTT SN

- Sensoren verteilt über eine große Fläche
- Verbindung zum Broker via Gateways
- UDP und ein reduziertes Protokoll
- 3 Arten von Gateways:
  - Integrated
  - Transparent
  - Aggregating

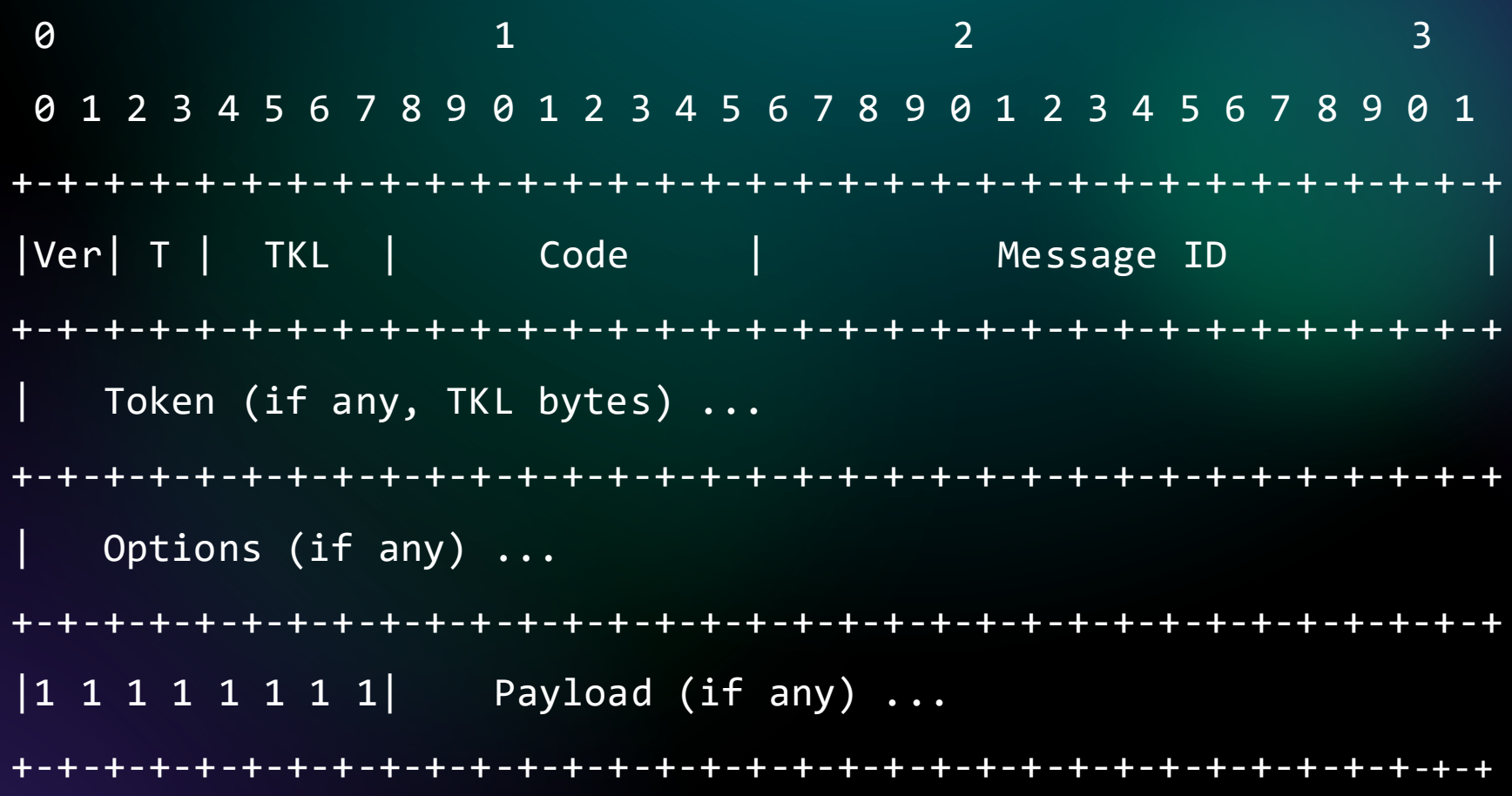


CoAP

- Für kleine Geräte mit instabiler Verbindung
- Verwendet UDP und ein reduziertes Protokoll ähnlich wie HTTP(s)
- Subscriptions (Abonnements)
- Auffinden von Diensten und Ressourcen



CoAP Nachricht



Code Beispiel github.com/plgd-dev/go-coap/v2

```
func communicate() context.CancelFunc{
    co, err := udp.Dial("192.168.188.50:5684")
    if err != nil {
        log.Fatalf("Error dialing: %v", err)
    }
    getPaths := []string{"/", "/config"}
    observePaths := []string{"/module_1/channel_1"}
    ctx, cancel := context.WithCancel(context.Background())
    for _, path := range getPaths {
        resp, _ := co.Get(ctx, path)
        printMessage(resp, path)
    }
    for _, path := range observePaths {
        go observe(path, co, ctx)
    }
    return cancel
}
```

```
func observe(path string, co *client.ClientConn, ctx
context.Context) {
    _, err := co.Observe(ctx, path, func(req *pool.Message) {
        printMessage(req, path)
    })
    if err != nil {
      log.Fatalf("Unexpected error '%v'", err)
    }
}
```

Code Beispiel de.uzl.itm.ncoap

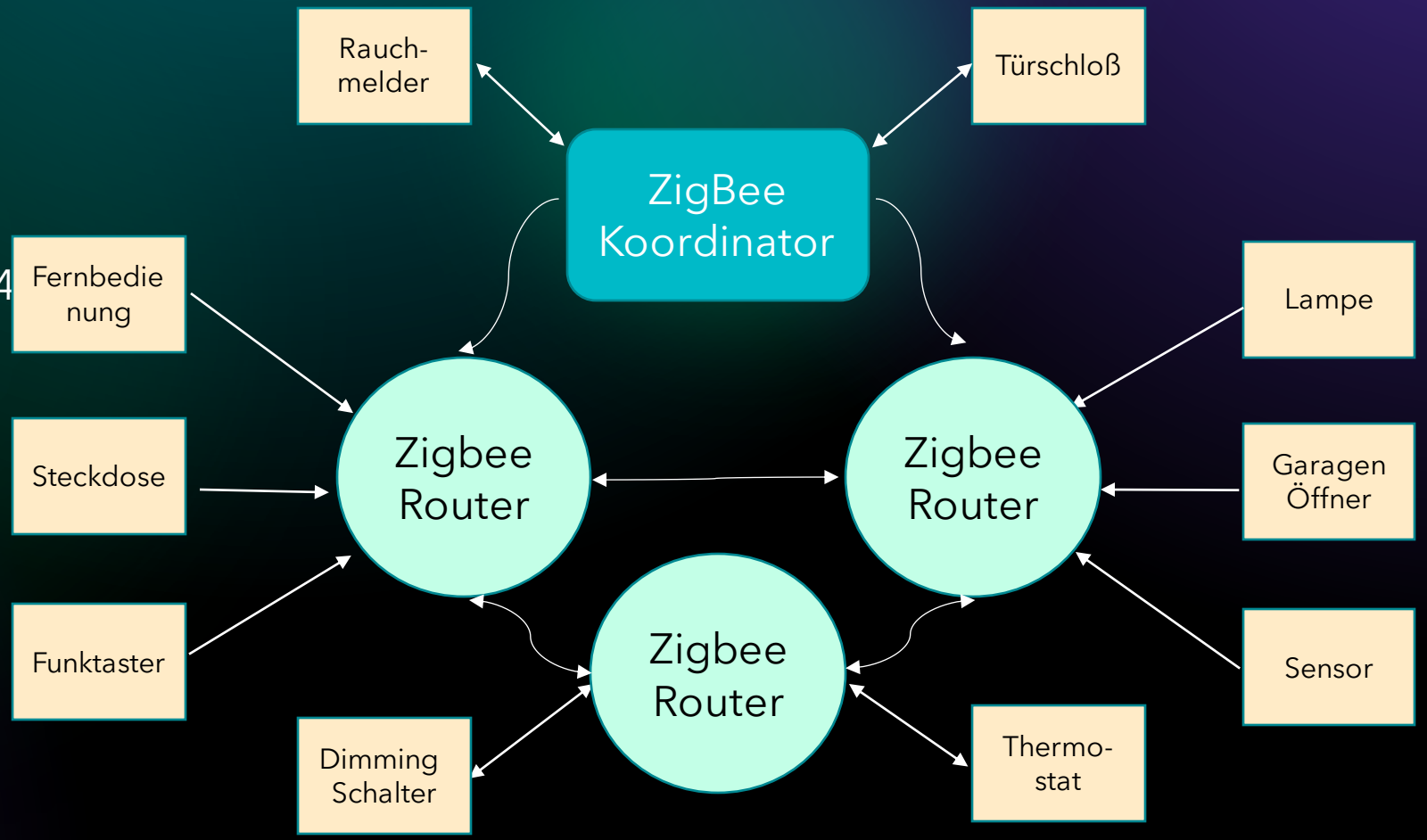
```
private void sendCoapRequest() throws URISyntaxException,
   UnknownHostException {

    String host = arguments.getUriHost();

    int port = arguments.getUriPort();

    String path = arguments.getUriPath();

    String query = arguments.getUriQuery();

    URI resourceURI = new URI ("coap", null, host, port, path,
   query, null);

    boolean useProxy = arguments.getProxyAddress() != null;

    int messageType = arguments.isNon() ? MessageType.NON :
   MessageType.CON;

    CoapRequest coapRequest = new CoapRequest(messageType,
   MessageCode.GET, resourceURI, useProxy);

    if (arguments.isObserve()) {

        coapRequest.setObserve(0);

    }
```

```
    InetSocketAddress remoteSocket;
    if (useProxy) {
        InetAddress proxyAddress =
InetAddress.getByName(arguments.getProxyAddress());
        int proxyPort = arguments.getProxyPort();
        remoteSocket = new InetSocketAddress(proxyAddress,
proxyPort);
    } else {
        InetAddress serverAddress
= InetAddress.getByName(arguments.getUriHost());
        int serverPort = arguments.getUriPort();
        remoteSocket = new InetSocketAddress(serverAddress,
serverPort);
    }
    if (arguments.isObserve()) {
        callback =
new  SimpleObservationCallback(arguments.getMaxUpdates());
    } else {
        callback = new SimpleCallback();
    }
    this.sendCoapRequest(coapRequest, remoteSocket,
callback);
}
```

Zigbee
ZigZag Dance of Bees

ZigBee

- Netzwerk mit Koordinator, Routern, Endgeräten
- Basierend auf IEEE-802.15.4 Standard (WPAN)
- Mehrer Profile
  - Home Automation
  - Light Link
- Reichweiter bis zu 100m
- Sicherheit via AES-128 Netzwermschlüssel



Code Beispiel

```
func toggleIkeaBulb(stewie *steward.Steward, message *model.DeviceIncomingMessage) {
    if isXiaomiButtonSingleClick(message) {
        if ikeaBulb, registered := devices["TRADFRI bulb E27 W opal 1000lm"]; registered {
            toggleTarget(stewie, ikeaBulb.NetworkAddress)
        } else {
            fmt.Println("IKEA bulb is not available")
        }
    }
}
func toggleTarget(stewie *steward.Steward, networkAddress string) {
    go func() {
        stewie.Functions().Cluster().Local().OnOff().Toggle(networkAddress, 0xFF)
    }()
}
```

Code Example

```
XBee xbee = new XBee();
xbee.open("COM5", 9600);

// this is the Serial High (SH) + Serial Low (SL) of the remote XBee
XBeeAddress64 addr64 = new XBeeAddress64(0xa, 0xb, 0xc, 0xd, 0xe, 0xf, 0, 1);

// Turn on DIO0 (Pin 20)
RemoteAtRequest request = new RemoteAtRequest(addr64, "D0", new int[{XBeePin.Capability.DIGITAL_OUTPUT_HIGH.getValue()});

xbee.sendAsynchronous(request);

RemoteAtResponse response = (RemoteAtResponse) xbee.getResponse();

if (response.isOk()) {
    System.out.println("Successfully turned on DIO0");
} else {
    System.out.println("Attempt to turn on DIO0 failed.  Status: " + response.getStatus());
}

// shutdown the serial port and associated threads
xbee.close();
```

Fazit

- MQTT
  - Pro: Einfaches Handling
  - Pro: Topics
  - Pro: Unidirektional
  - Con: Broker
  - Con: externe Metaprotokolle
  - Con: Dedizierte Rollen
- CoAP
  - Pro: Spezifiziertes Protokoll
  - Pro: Discovery und Multicast
  - Pro: Bidirektional
  - Con: Komplexeres Handling
  - Con: Gerät ist Server
  - Con: Strikterer Funktionsumfang
- ZigBee
  - Pro: Höhere Ausfallsicherheit
  - Pro: Vernetzung
  - Pro: Resourcennutzung
  - Con: Komplexes Handling
  - Con: Viel Hardware nötig
  - Con: Reduzierter Funktionsumfang

Fragen?

Quellen

- OSI Modell und TCP/IP Referenz:
  - https://de.wikipedia.org/wiki/OSI-Modell
- IoT Protokolle:
  - http://revolutionofthings.com/de/iot-protokolle/
  - https://nicolaswindpassinger.com/osi-reference-model
  - https://www.industry-of-things.de/http-xmpp-coapp-und-mqtt-iot-protokolle-fuer-die-kommunikation-a-813079/

Quellen

- MQTT
  - http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/
- CoAP:
  - https://datatracker.ietf.org/doc/html/rfc7252
- ZigBee
  - https://de.wikipedia.org/wiki/Tanzsprache
  - https://de.wikipedia.org/wiki/ZigBee
  - https://www.homeandsmart.de/zigbee-funkprotokoll-hausautomation

Quellen

- Code Examples:
  - https://www.emqx.com/en/blog/how-to-use-mqtt-in-golang
  - https://github.com/tgrall/mqtt-sample-java
  - https://github.com/okleine/nCoAP
  - https://github.com/dyrkin/zigbee-steward
  - https://github.com/andrewrapp/xbee-api