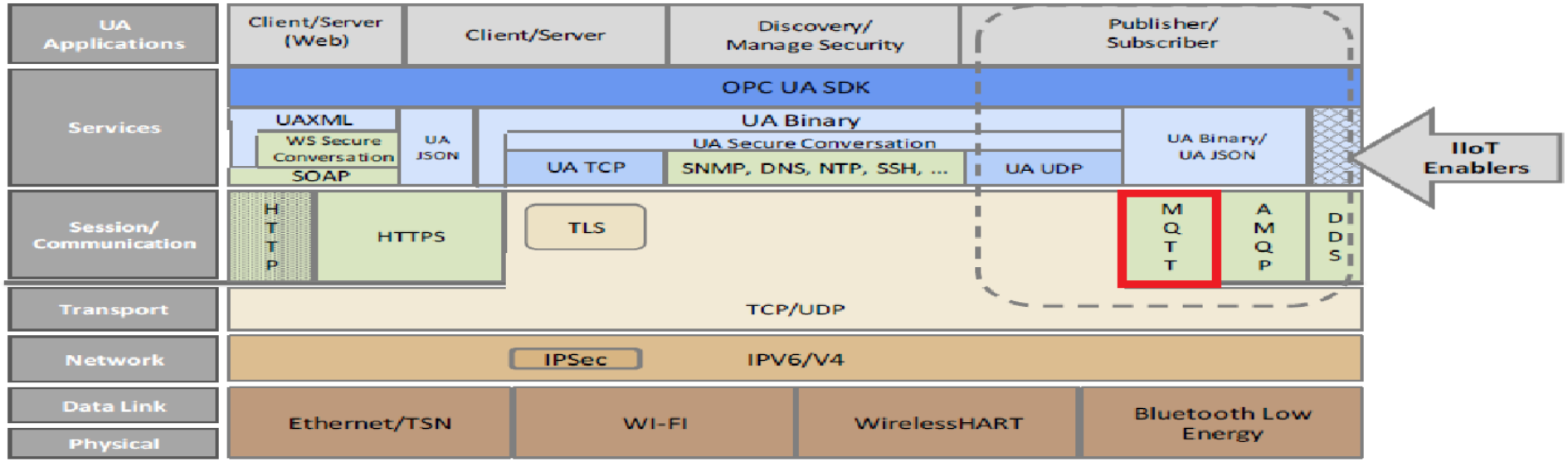
Transporte MQTT do SAGE-OPCUA para conexão direta com o WebSocket do Grafana-8 Live



6.5 Publish-Subscribe

With *PubSub, OPC UA Applications* do not directly exchange requests and responses. Instead, *Publishers* send messages to a *Message Oriented Middleware*, without knowledge of what, if any, *Subscribers* there may be. Similarly, *Subscribers* express interest in specific types of data, and process messages that contain this data, without knowledge of what *Publishers* there are.

Message Oriented Middleware is software or hardware infrastructure supporting sending and receiving messages between distributed systems. It depends on the *Message Oriented Middleware* how this distribution is implemented.

To cover a large number of use cases, OPC UA *PubSub* supports two largely different *Message Oriented Middleware* variants. These are:

- A broker-less form, where the *Message Oriented Middleware* is the network infrastructure that is able to route datagram-based messages. *Subscribers* and *Publishers* use datagram protocols like UDP multicast.
- A broker-based form, where the *Message Oriented Middleware* is a *Broker*. *Subscribers* and *Publishers* use standard messaging protocols like AMQP or MQTT to communicate with the *Broker*. All messages are published to specific queues (e.g. topics, nodes) that the *Broker* exposes and *Subscribers* can listen to these queues. The *Broker* may translate messages from the formal messaging protocol of the *Publisher* to the formal messaging protocol of the *Subscriber*.

PubSub is used to communicate messages between different system components without these components having to know each other's identity.

A *Publisher* is pre-configured with what data to send. There is no connection establishment between *Publisher* and *Subscriber*.

The knowledge about who *Subscribers* are and the forwarding of published data to the *Subscribers* is off-loaded to the *Message Oriented Middleware*. The *Publisher* does not know or even care if there is one or many *Subscribers*. Effort and resource requirements for the *Publisher* are predictable and do not depend on the number of *Subscribers*.

Part 14 describes the details of the OPC UA *PubSub* model.

7.3.5 MQTT

7.3.5.1 General

The Message Queue Telemetry Transport (MQTT) is an open standard application layer protocol for *Message Oriented Middleware*. MQTT is often used with a *Broker* that relays messages between applications that cannot communicate directly.

Publishers send MQTT messages to MQTT brokers. Subscribers subscribe to MQTT brokers for messages. A *Broker* may persist messages so they can be delivered even if the subscriber is not online. *Brokers* may also allow messages to be sent to multiple *Subscribers*.

The MQTT protocol defines a binary protocol used to send and receive messages from and to topics. The body is an opaque binary blob that can contain any data serialized using an encoding chosen by the application.

This specification defines two possible encodings for the message body: the binary encoded *DataSetMessage* defined in 7.2.2 and a JSON encoded *DataSetMessage* defined in 7.2.3. MQTT does not provide a mechanism for specifying the encoding of the MQTT message which means the *Subscribers* shall be configured in advance with knowledge of the expected encoding. *Publishers* should only publish *NetworkMessages* using a single encoding to a unique MQTT topic name.

Security with MQTT is primary provided by a TLS connection between the *Publisher* or *Subscriber* and the MQTT server, however, this requires that the MQTT server be trusted. For that reason, it may be necessary to provide end-to-end security. Applications that require end-to-end security with MQTT need to use the UADP *NetworkMessages* and binary message encoding defined in 7.2.2. JSON encoded message bodies must rely on the security mechanisms provided by MQTT and the MQTT server.