Elektrobit

Time-sensitive control data streams in service-oriented architecture based on Automotive Ethernet

Dr. Thomas Galla, Dr. Michael Ziehensack

November 2021 | IEEE SA Ethernet & IP @ Automotive Technology Week, Munich

Rettungsgasse – ad hoc emergency lane in case of a traffic jam

Mandatory on highways in Germany and Austria

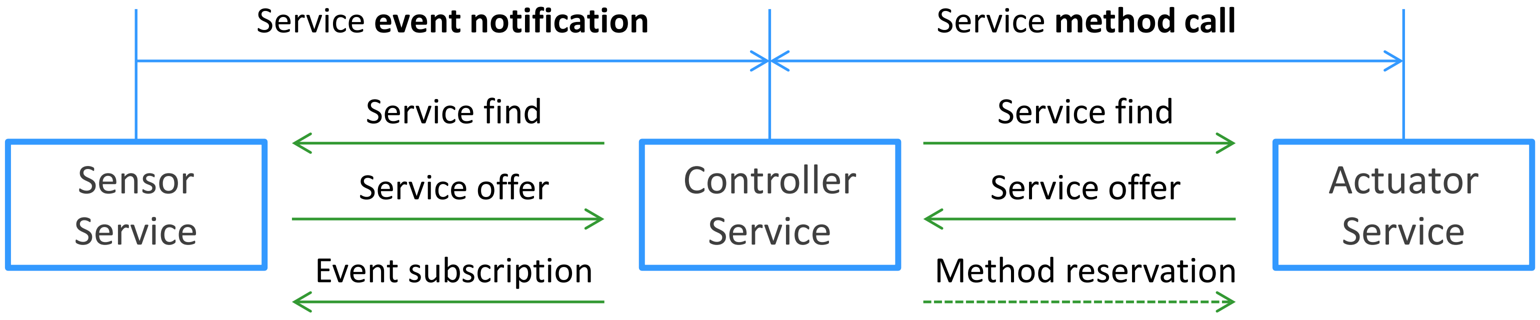
Give way to important things –
that´s what we also need for time-critical applications …

Service-Oriented Architecture (SOA)

Brief overview

- **Service** = discrete unit of functionality that can be accessed remotely and updated independently
- Applications are built by combining services, e.g. control loop built out of Sensor, Controller, Actuator

Server offers and provides service events and methods



Client finds services and
- subscribes/consumes events
- reserves/calls methods

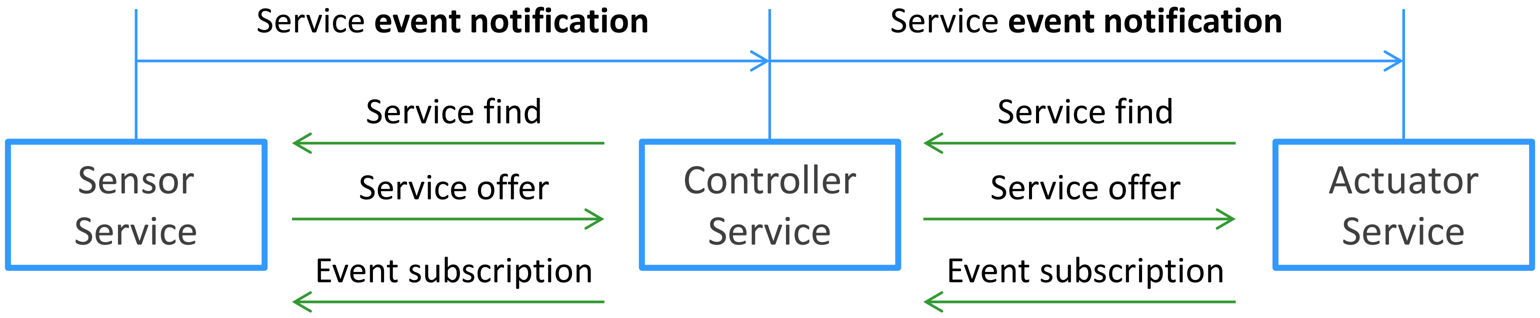
- **Binding** between **server** (service provider) and **client** (service user) is established dynamically via **service discovery**
- Provides scalability and allows adding new service users without any adaptation of existing service providers

Service-Oriented Architecture (SOA)

Brief overview

- Interaction between Controller and Actuator Service can also be realized via service events (instead of methods)
- In this case client/server roles are swapped between Controller and Actuator

Server offers and provides service events



Client finds services and subscribes and consumes service events

Note:

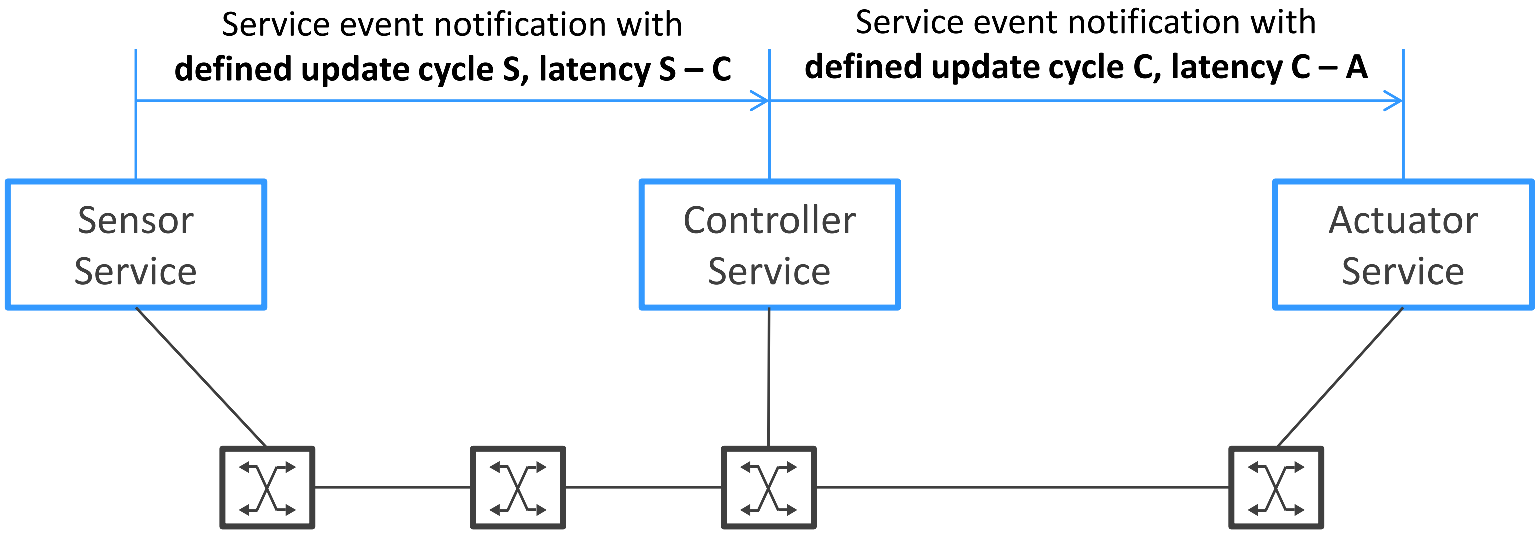
- The examples in this presentation focus on the scenario with service events
- The concept shown also applies to service methods (with some adaptations)

Time-Sensitive Control Data Streams in SOA

Motivation and overview

- Advanced driver assistance and automated driving functions as well as chassis applications are time-critical systems using time-sensitive control data streams.

Server provides service events with a defined **update cycle**



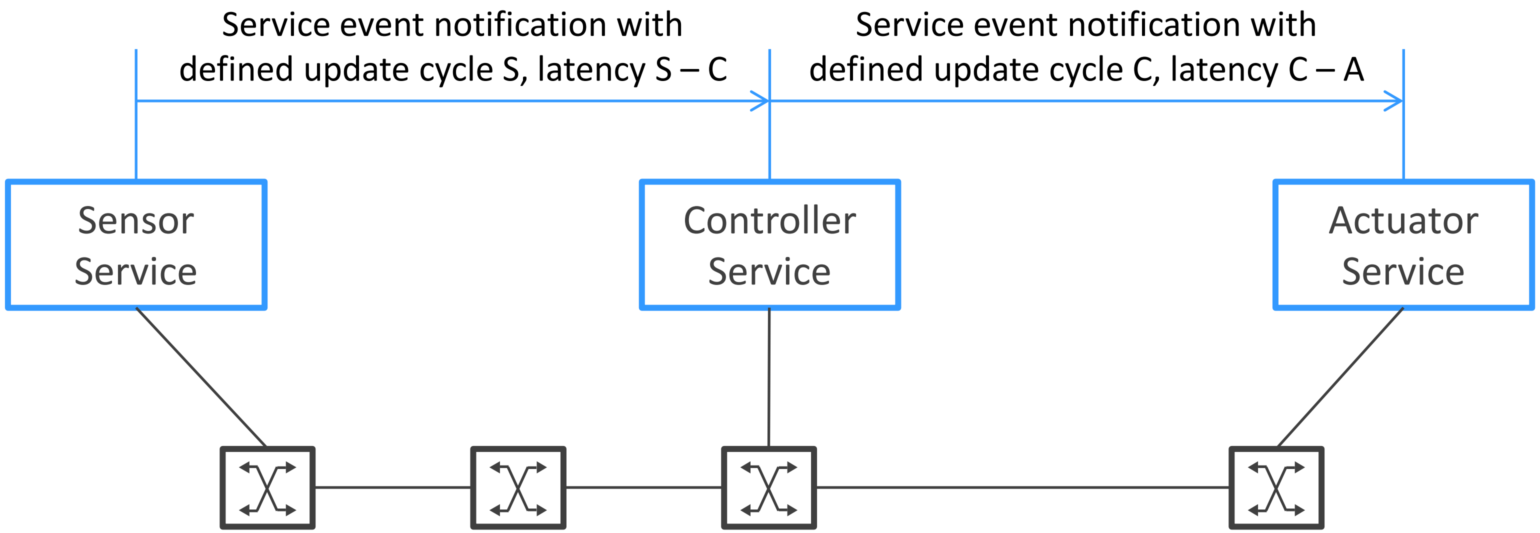
Client needs service events with a defined **update cycle** and **max bounded latency**

Automotive Ethernet network with TSN can **guarantee maximum bounded latencies** for control data streams by separating the streams into up to 8 traffic classes and applying traffic shaping

… Ethernet switch

Time-Sensitive Control Data Streams in Service-Oriented Architecture

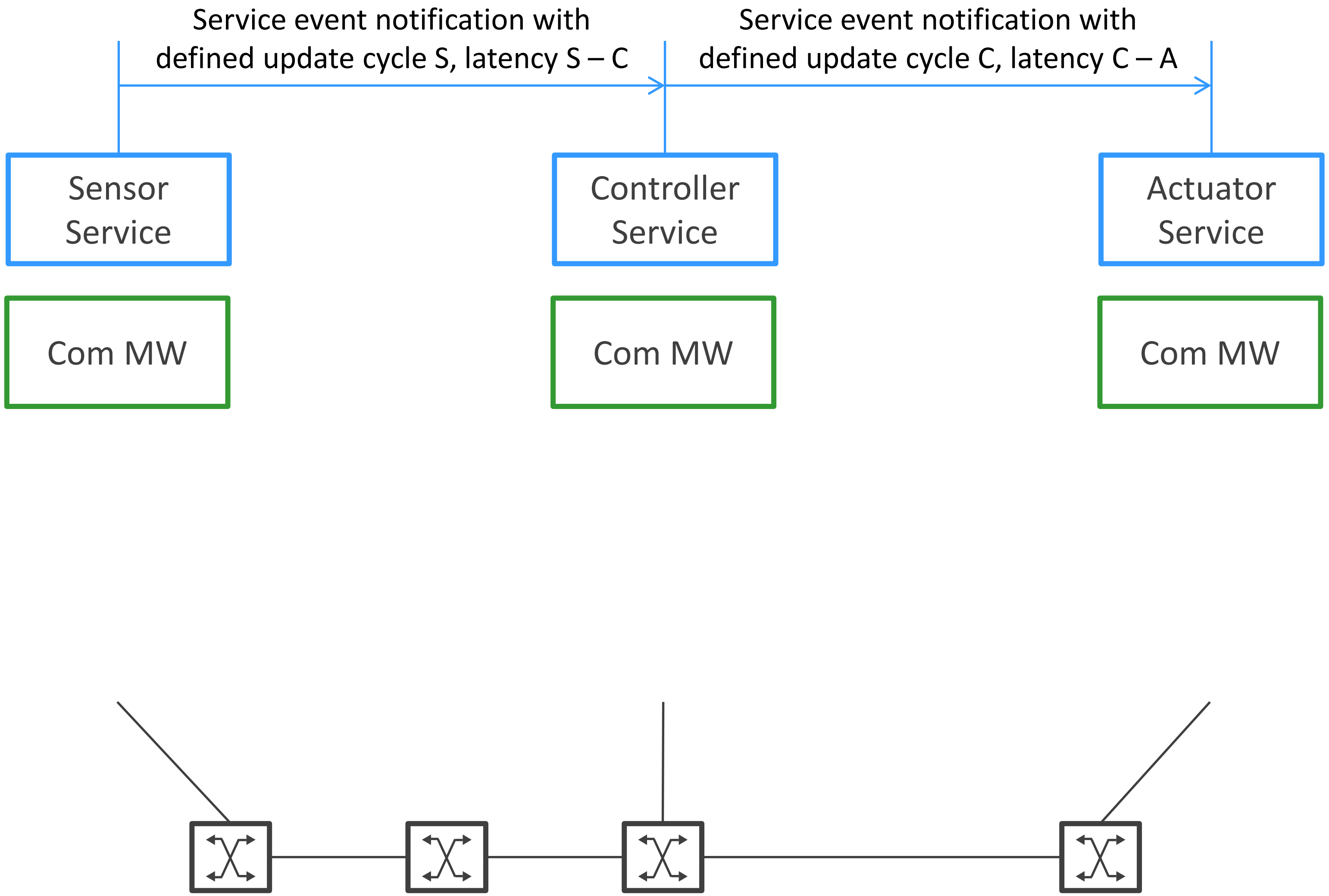
Introducing further layers



Time-Sensitive Control Data Streams in Service-Oriented Architecture

Elektrobit

Introducing further layers – Communication Middleware

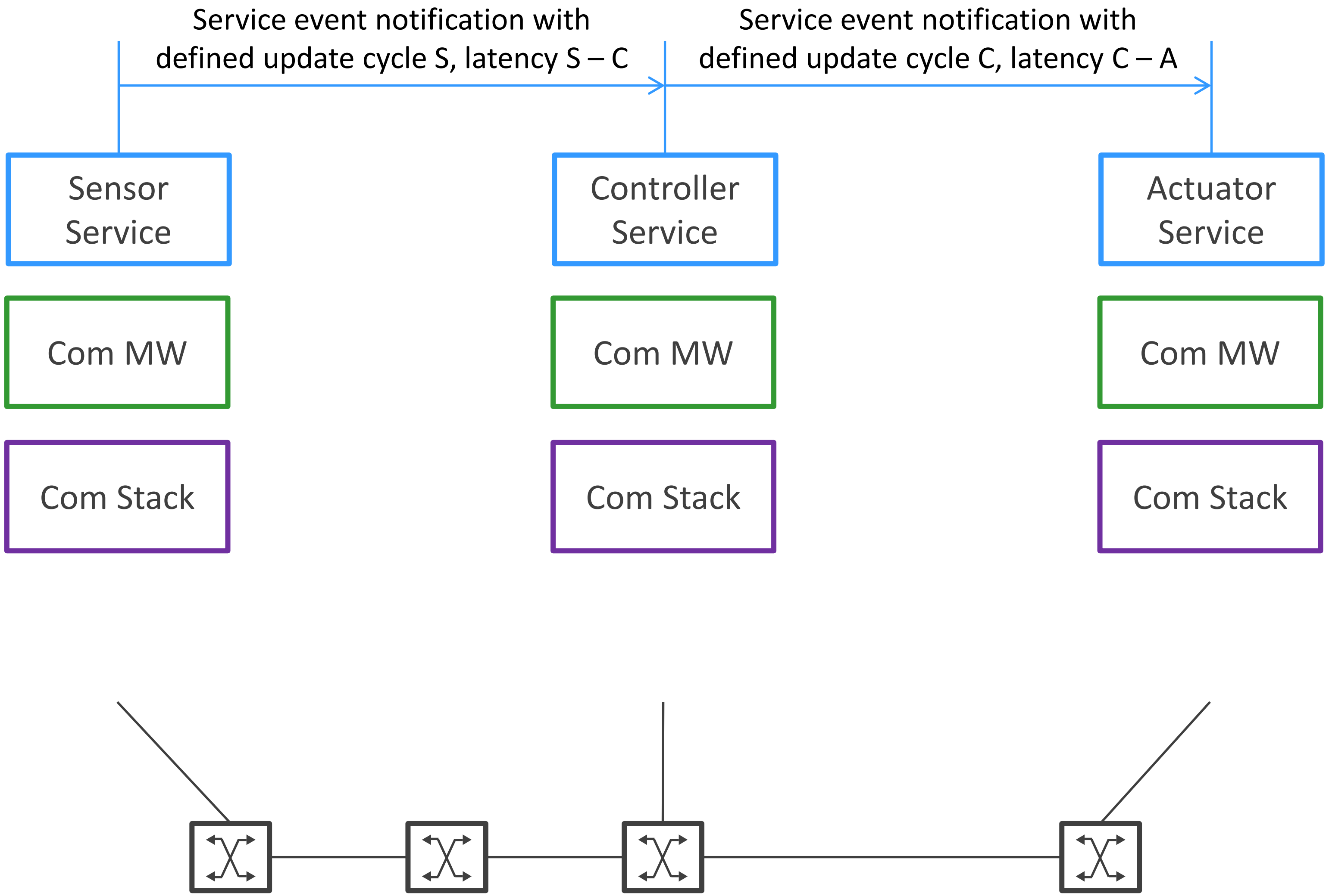


Communication Middleware

- **implements** OSI Layer 5 – 7
- **negotiates** service event update cycle and latency dynamically
- **provides** environment for prioritized channels
- **selects** Tx/Rx priority based on max latency
- **checks** if selected prio is supported
- **requests** bandwidth reservation at all involved network elements
- **reports** success/failure to Client

Time-Sensitive Control Data Streams in Service-Oriented Architecture

Introducing further layers – Communication Stack

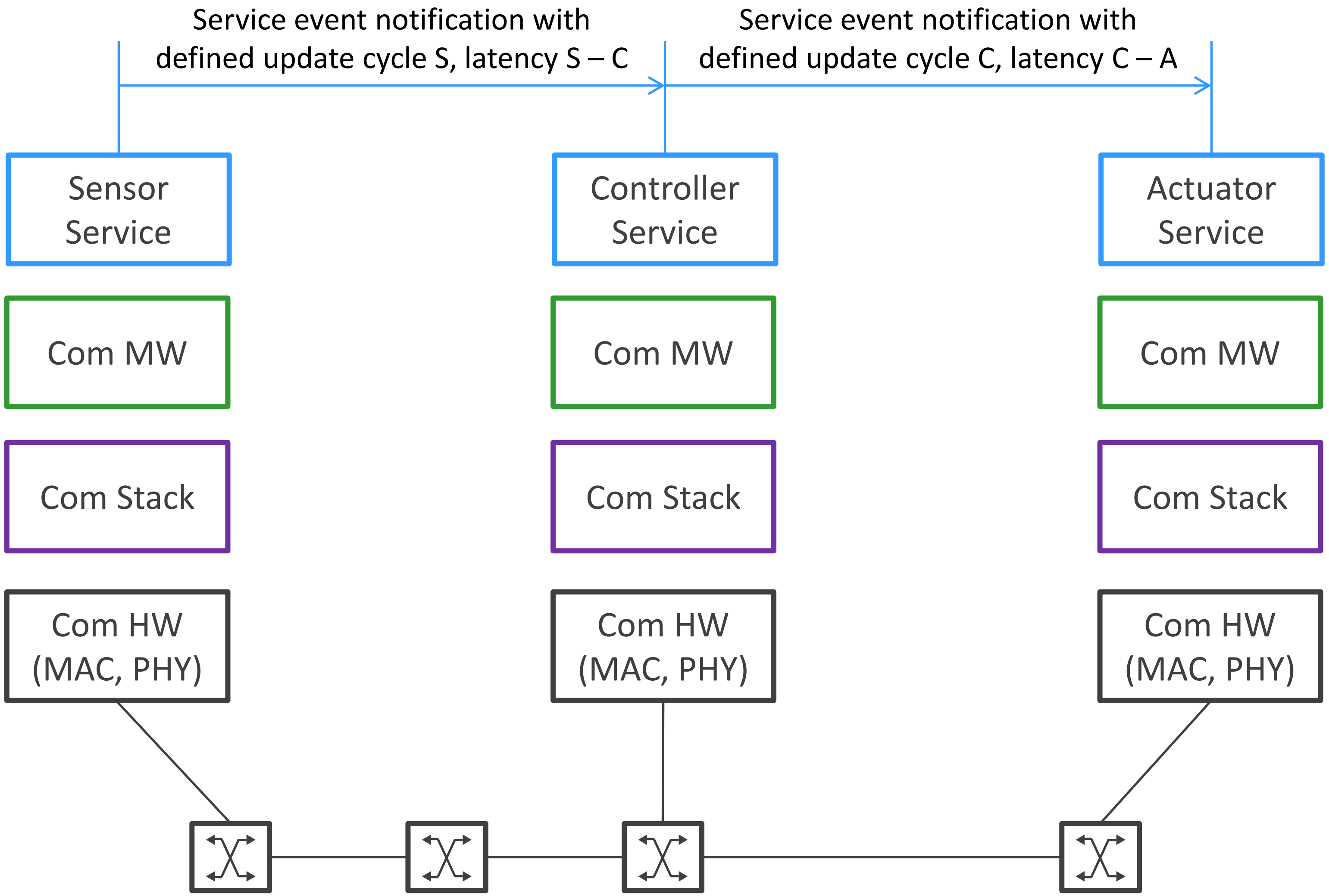


Communication Stack

- **implements** OSI Layer (2), 3, 4
- **consists** of Network stack and drivers
- **provides** environment for prio channels
- **sets** VLAN PCP field based on priority
- **selects** Tx/Rx priority queue of Com HW

Time-Sensitive Control Data Streams in Service-Oriented Architecture

Introducing further layers – Communication Hardware

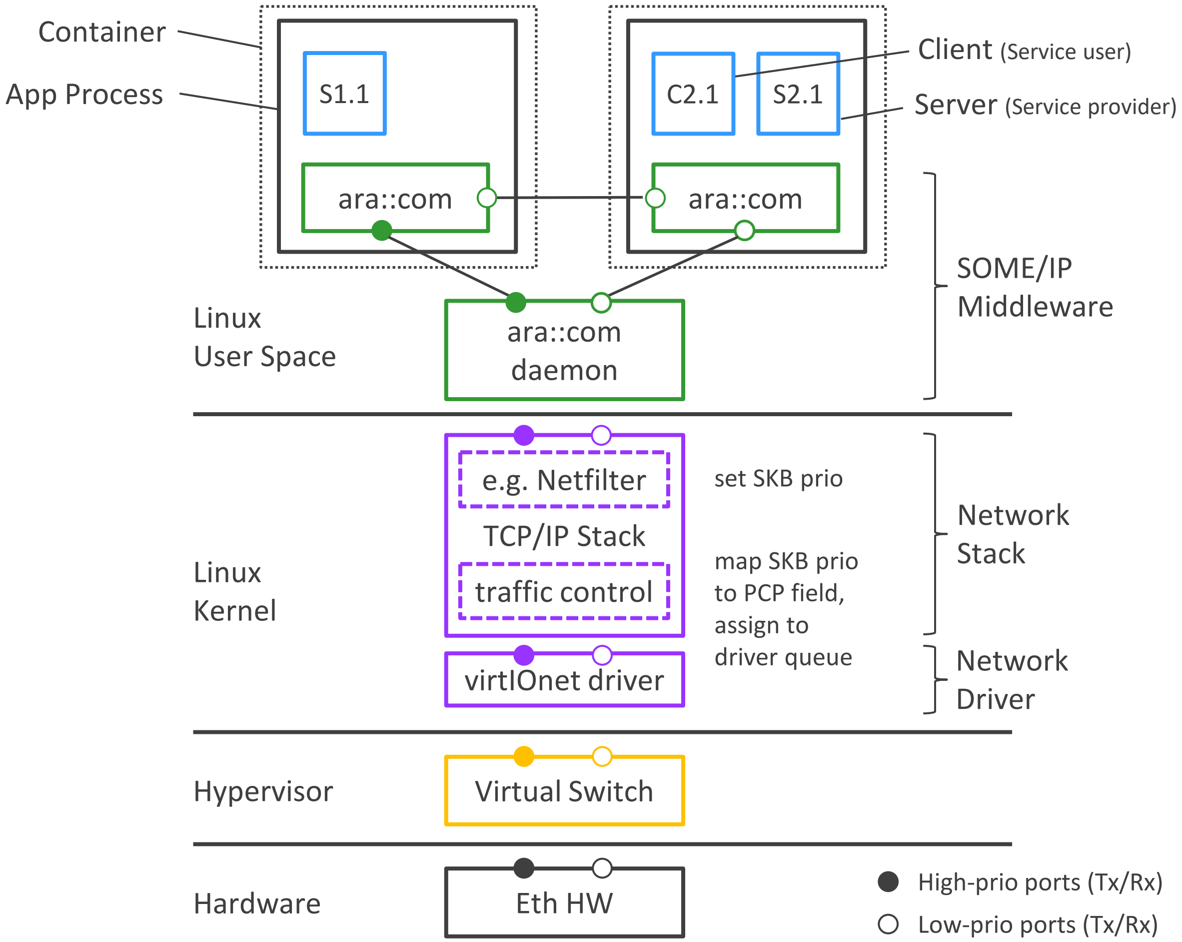


Com HW

- **implements** OSI Layer 1, 2
- **consists** of multiple network elements: switches, source and destination nodes
- related TSN standards:
  - 802.1AS Timing and Synchronization
  - 802.1Q - Forwarding and Queuing Enhancements for Time-Sensitive Streams (FQTSS): Priority, CBS
  - 802.1Q - Enhancements for Scheduled Traffic: TAS
  - 802.1Q - Asynchronous Traffic Shaping
  - 802.1Q - Stream Reservation Protocol (SRP)
  - 802.1Q - Per-Stream Filtering and Policing
  - 802.1Q - Cyclic Queuing and Forwarding
  - 802.1Q - Frame Preemption

Communication SW Architecture of a Vehicle Server (HPC)

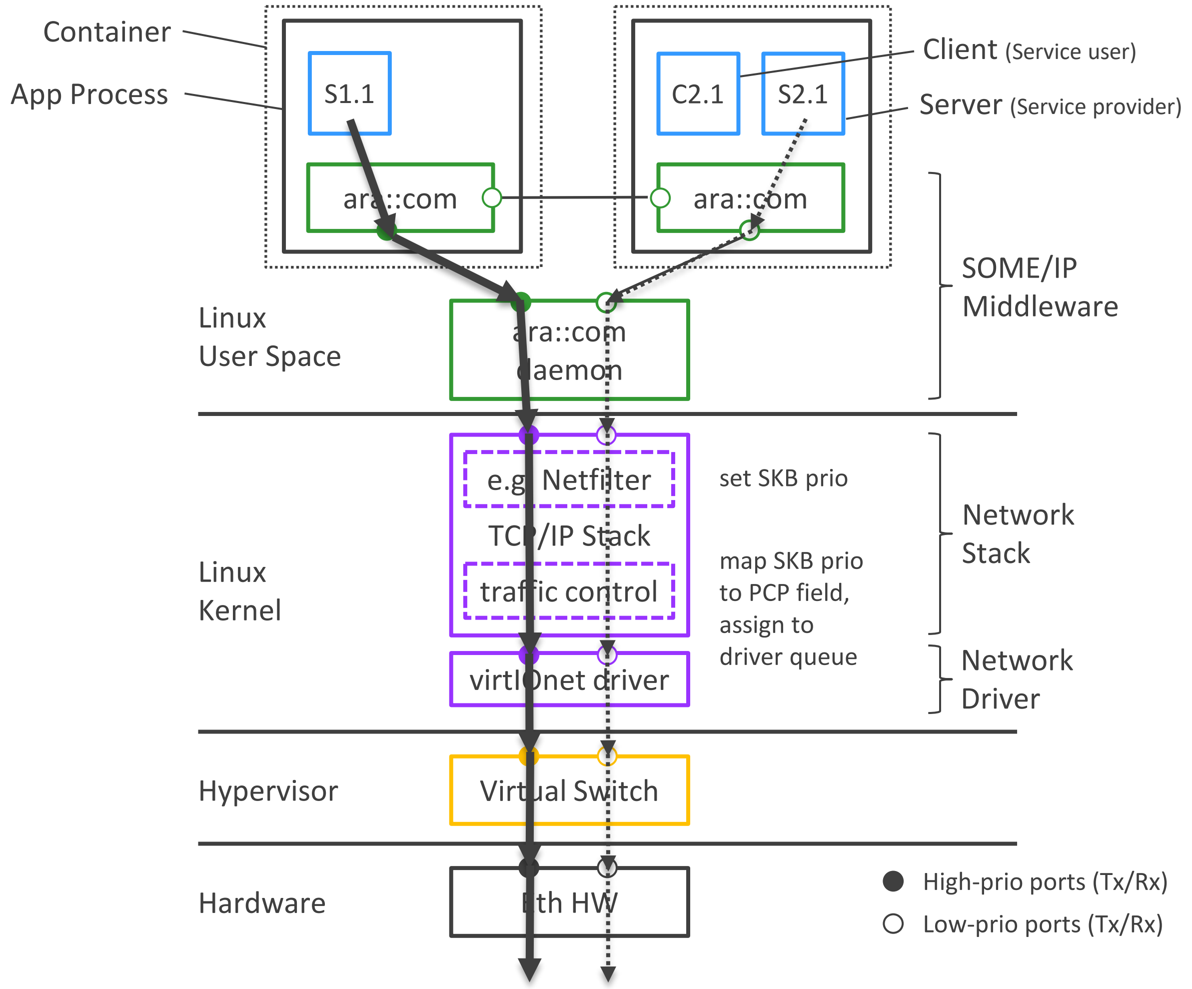
Latency and prioritization measures on the different software layers



- **Com Middleware**
  - ara::com daemon providing a priority and a best effort channel
  - separate IPC sockets and processing threads with different priority (within same prio: weighted round-robin, between prios: strict priority)
- **Network stack**
  - Maps MW prio channel to related UDP socket; set SKB prio accordingly (`setsockopt(…, SO_PRIORITY, …)`)
  - traffic control sets PCP field and selects driver queue (congestion resolution: Qdisc MQPRIO (multiqueue priority))
- **Network driver**
  - Maps driver queue of a VM to Virtual Switch queue
- **Virtual Switch (Hypervisor)**
  - Maps Virtual Switch queue to Eth HW queue
  - Frame processing acc. to strict priority and via interleaved weighted round robin algorithm within the same priority
- **Ethernet hardware** queues
  - Two queues: priority and non-priority

Communication SW Architecture of a Vehicle Server (HPC)

Latency and prioritization measures on the different software layers



Test Setup:

- Server S1.1 transmits data with high priority (bold line)
- Server S2.1 transmits data with low priority (dotted line) and might be blocked by interfering traffic
- Traffic generator that can be activated to produce a lot of interfering traffic on a low priority socket
- Used Software: EB corbos AdaptiveCore, EB corbos Linux, EB corbos Hypervisor, EB corbos virtual Switch

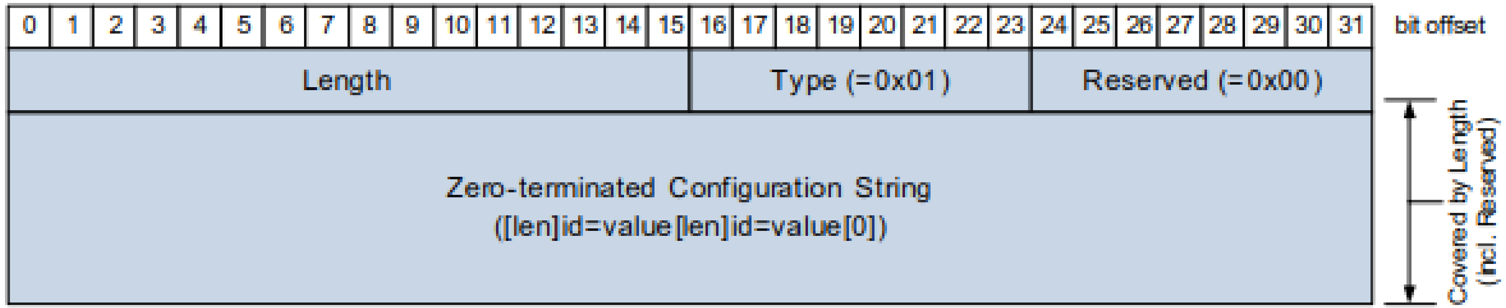
Results:

- Service events from S1.1 are always transmitted without any frame drops or latency violations (independent of interfering traffic)
- Service events from S2.1 are transmitted incompletely (frame drops, increased latency depending on interfering traffic)

AUTOSAR SOME/IP QoS Extension (Update Cycle, Latency)

COM Middleware mechanism to negotiate Update Cycle and Latency - Overview

- Negotiation can be done via existing SOME/IP-SD feature
  - Configuration options (a.k.a. capability records) – may carry arbitrary additional information for a service interface



Ref. AUTOSAR

Key/value pairs (like DNS-SD TXT records according to IETF RFC 6763)

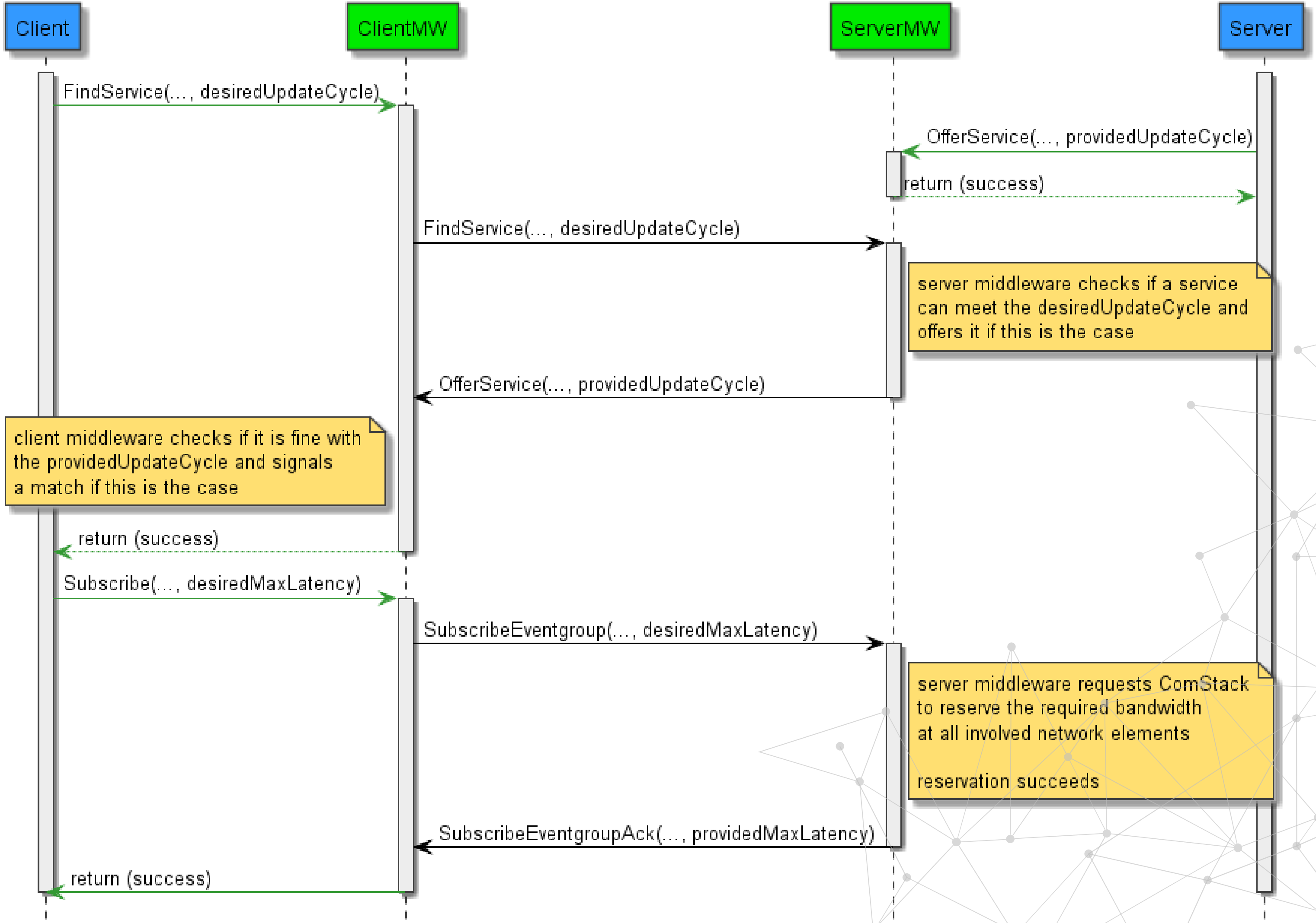
  - Use these configuration options in *OfferService*, *FindService*, *SubscribeEventGroup*, *SubscribeEventGroupAck* SOME/IP-SD message entries to transport QoS attributes, introduce standardized keys (**`max_latency`**, **`update_cycle`**)
- Interface extension for using these config options
  - Variant 1: QoS attributes are configured statically per service interface, e.g. AP ara::com Manifest (configuration)
  - Variant 2: QoS attributes are provided by the application during run-time, e.g. AP ara::com API (additional arguments)

AUTOSAR SOME/IP QoS Extension (Update Cycle, Latency)

Negotiation of Update Cycle and Latency – Sequence Diagram: Success Scenario

Scenario description

- Client looks for a service event with a desired
  - update cycle and
  - maximum latency
- Server offers a service event with a provided update cycle that is **lower** than the desired one of the Client
- Server middleware requests ComStack to reserve bandwidth to guarantee the desired maximum latency and reservation **succeeds**

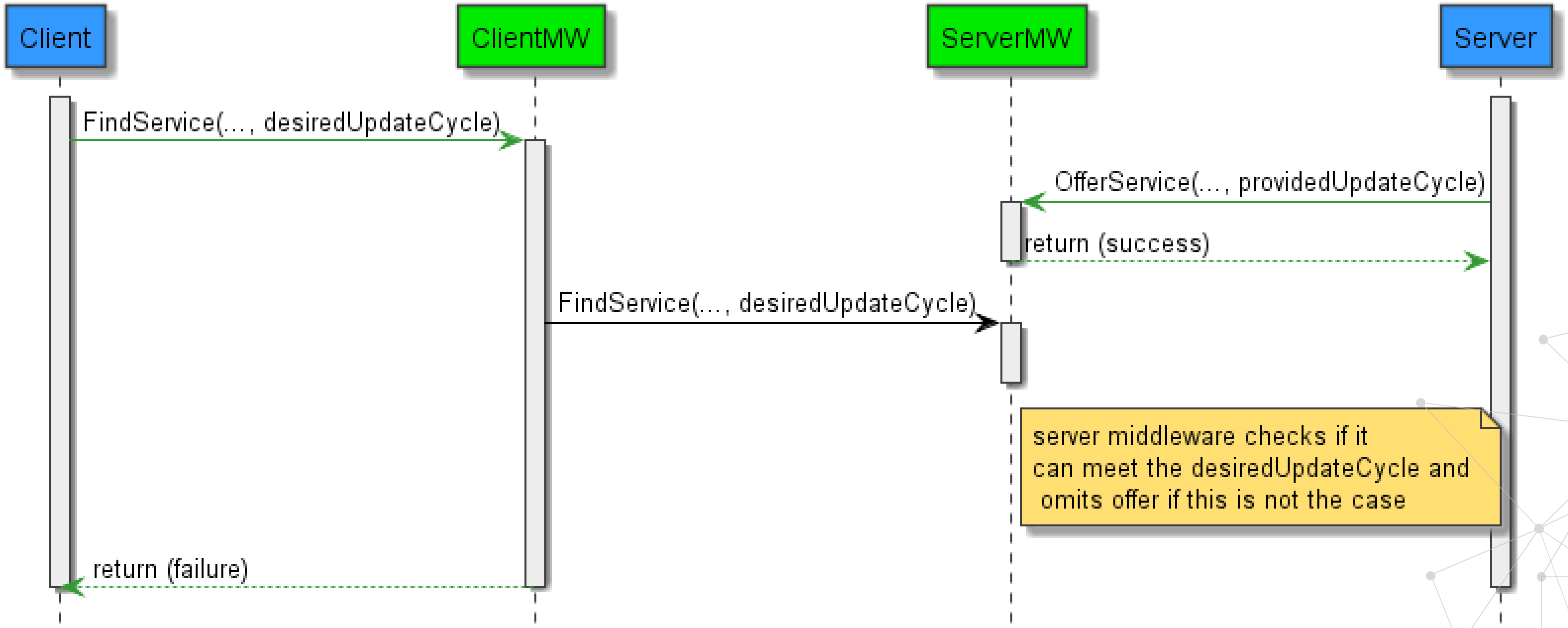


AUTOSAR SOME/IP QoS Extension (Update Cycle, Latency)

Negotiation of Update Cycle and Latency – Sequence Diagram: Failure Scenario (Update Cycle Mismatch)

Scenario description

- Client looks for a service event with a desired
  - update cycle and
  - maximum latency
- Server offers a service event with a provided update cycle that is **larger** than the desired one of the Client

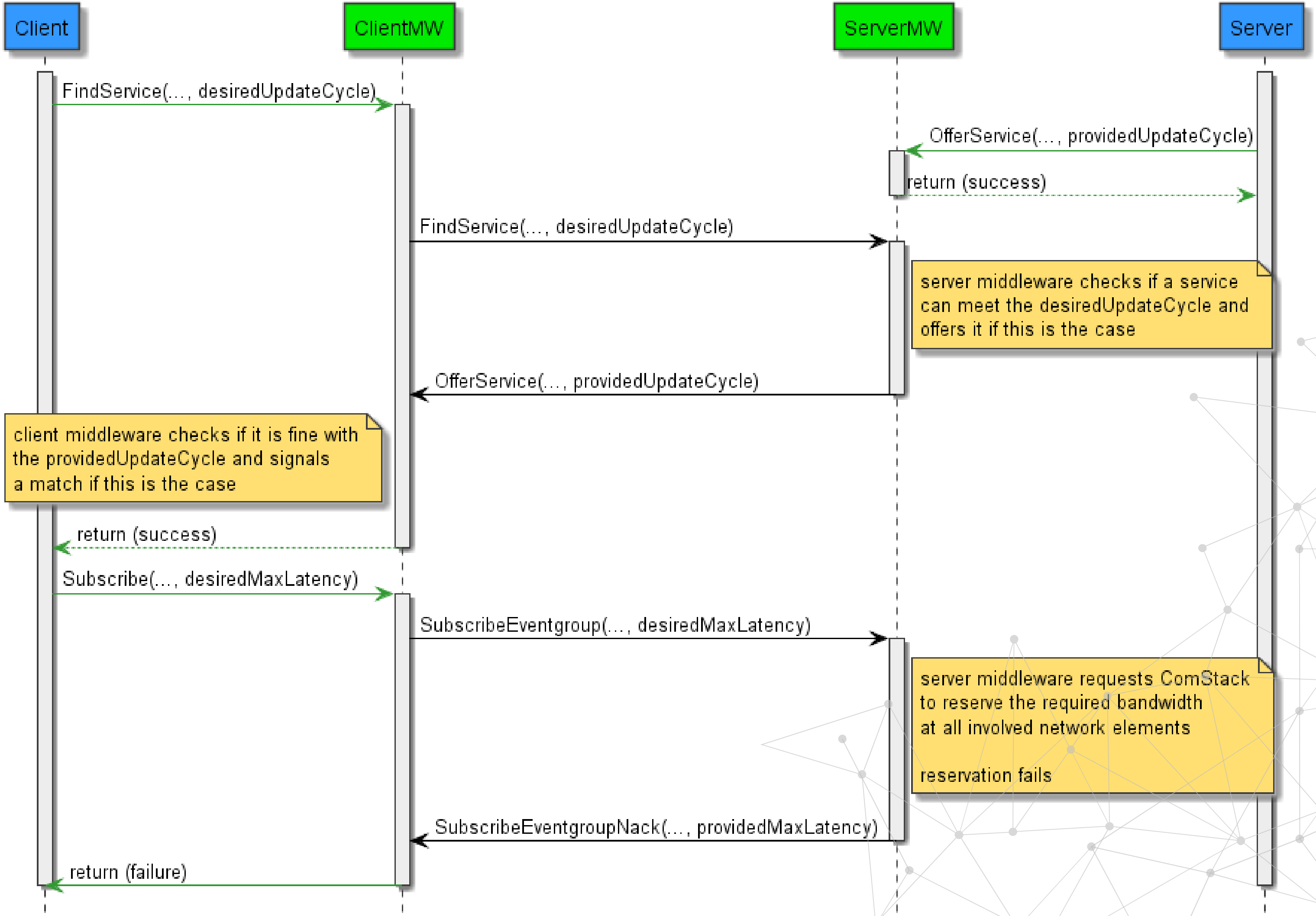


AUTOSAR SOME/IP QoS Extension (Update Cycle, Latency)

Negotiation of Update Cycle and Latency – Sequence Diagram: Failure Scenario (Latency Violation)

Scenario description

- Client looks for a service event with a desired
  - update cycle and
  - maximum latency
- Server offers a service event with a provided update cycle that is **lower** than the desired one of the Client
- Server middleware requests ComStack to reserve bandwidth to guarantee the desired maximum latency and reservation **fails**



Configuration and Bandwidth Reservation of Network Elements

COM Stack mechanism for configuration and bandwidth reservation – some alternative options

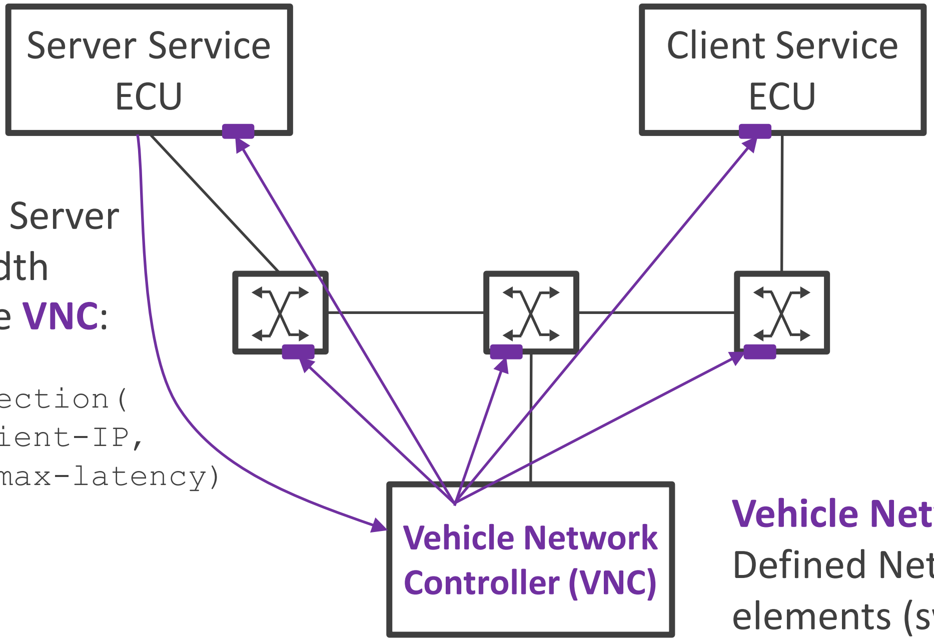
(1) **Static configuration** using virtual reservations based on a-prio known communication relationships

(2) **Stream Reservation Protocol (SRP)** according to IEEE 802.1Q

(3) Central **Vehicle Network Controller (VNC)** based on Software-Defined Network (SDN) Architecture as shown below

Com Stack at the Server requests bandwidth reservation at the **VNC**:

```
requestL3Connection(
server-IP, client-IP,
data-length, max-latency)
```



- Standardized remote configuration interface (e.g. NETCONF/YANG) to switches and end nodes

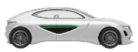
Vehicle Network Controller (VNC) is basically a Software-Defined Network (SDN) controller that manages all network elements (switches, end nodes)

Summary

Time-sensitive control data streams in service-oriented architecture are **service events with a defined update cycle and maximum bounded latency**.

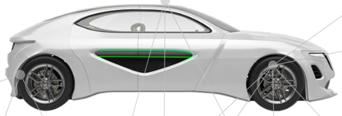
Communication middleware needs to support dynamic negotiation of update cycle and latency.
Related **extension for AUTOSAR SOME/IP QoS support** has been shown.

For QoS the whole data path on the network and from network interface to the application need to be covered.
Concrete measures have been described **for the different communication layers** (e.g., priority and a best effort channels, dedicated processing threads, virtual switch priority queues, interleaved weighted round robin)



Experiences from the **QoS implementation of a Vehicle Server (HPC)** based on EB´s products have been shared.
Time-sensitive control data streams in SOA are fully supported along the complete communication path.

Thank you for your attention!!



Contact us

Elektrobit

Dr. Thomas Galla, Elektrobit
Chief Architect, Automotive Networks
thomas.galla@elektrobit.com

Dr. Michael Ziehensack, Elektrobit
VP, Automotive Networks
michael.ziehensack@elektrobit.com