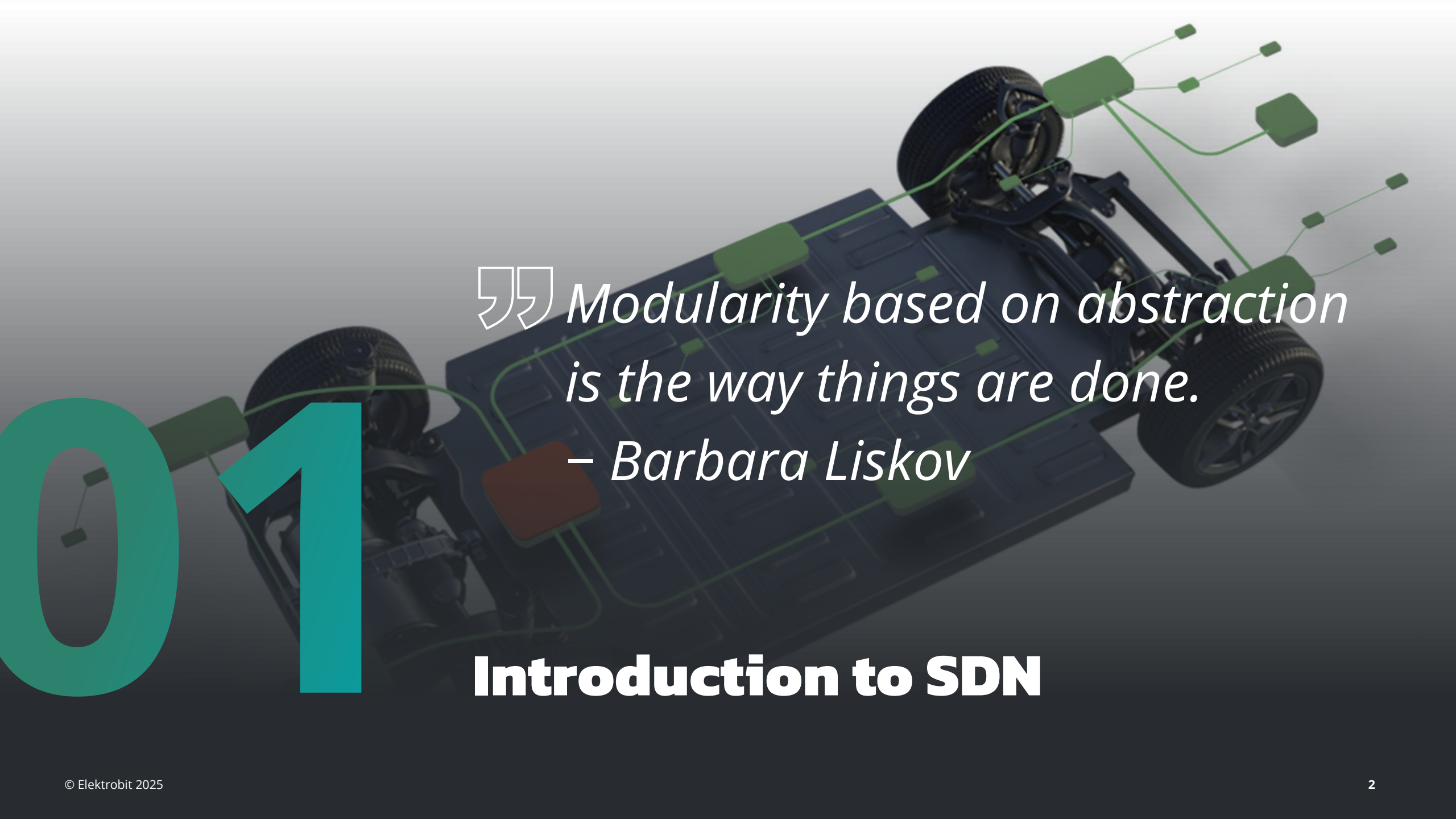
Software Defined Network (SDN) works for automotive

Getting automotive TSN-networks ready for SDV

Carsten Weich, Daniel Thiele

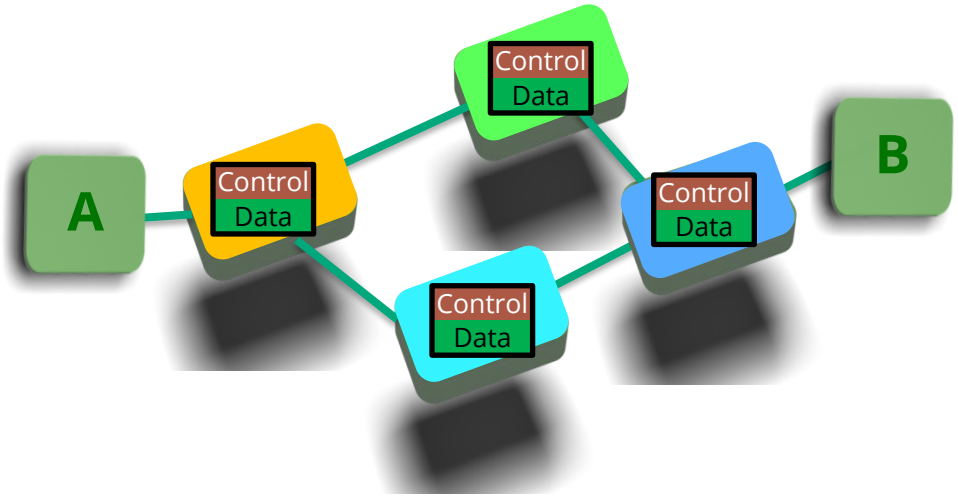
IEEE Ethernet & IP @ Automotive Technology Day
October 16, 2025



Modularity based on abstraction is the way things are done.
– Barbara Liskov

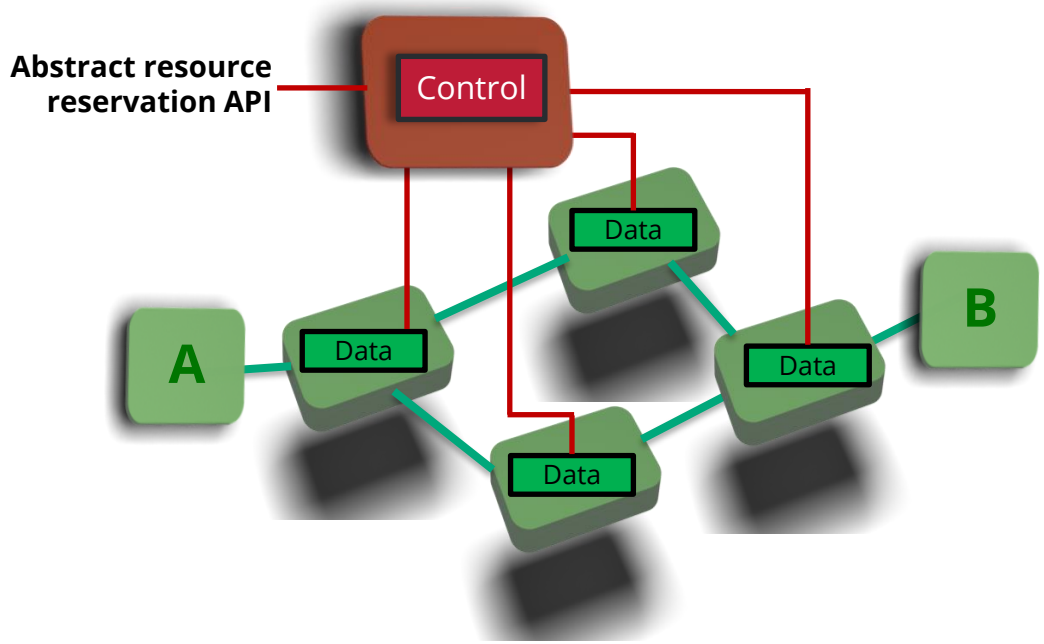
01 Introduction to SDN

Introduction to SDN



Non-SDN networks

- **Coupled** control and data planes
- **High reconfiguration effort** due to high vertical integration (i.e. bundled and device-specific)
  - Update requires **individual updates** of all devices along a path
- **Local view**: independent control instances on each device



SDN networks

- **Decoupled** control and data planes
  - **Abstraction** of network devices
- **Abstraction** of communication resource reservation
  - **Updates** are consistently **managed through the controller**
- **Global view** allows **efficient network usage** (e.g. alternative paths)

There is no true software-defined vehicle (SDV) without automotive software-defined networking (SDN)!

02 Automotive requirements

Software Defined Vehicle (SDV) vision

A Software-Defined Vehicle is any vehicle that manages its operations, adds functionality, and enables new features primarily or entirely through software.

Requirements

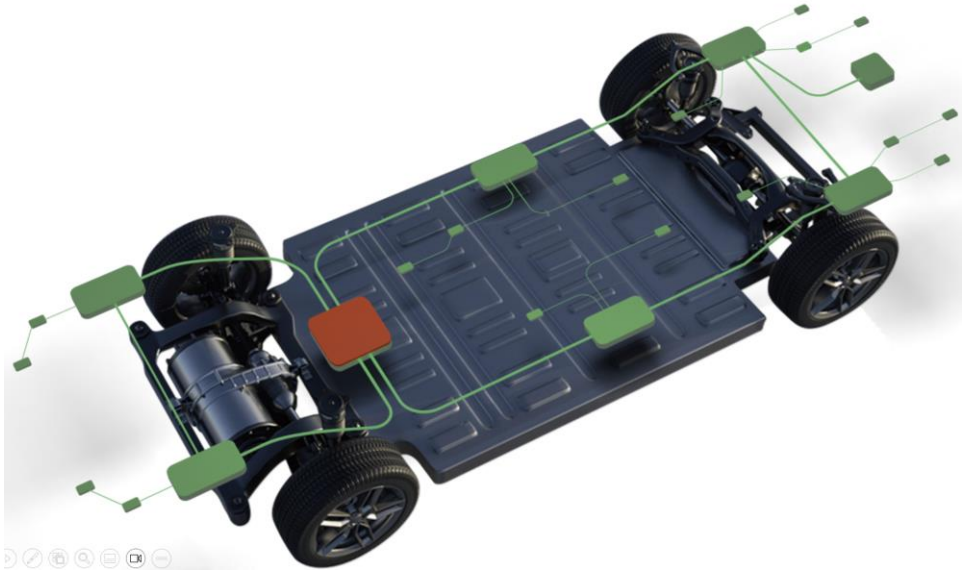
Increasing user expectations in infotainment as well as assisted and automated driving

- **Massively increases the requirements for in-vehicle communication**
  - Bandwidth,
  - QoS,
  - Safety, security,
  - Number of subscribers, …
- **Requires much more flexibility and adaptability** of the vehicle network to apply feature updates and new features via over-the-air software updates for any ECU of the vehicle.

Software Defined Vehicle (SDV) vision

Status quo

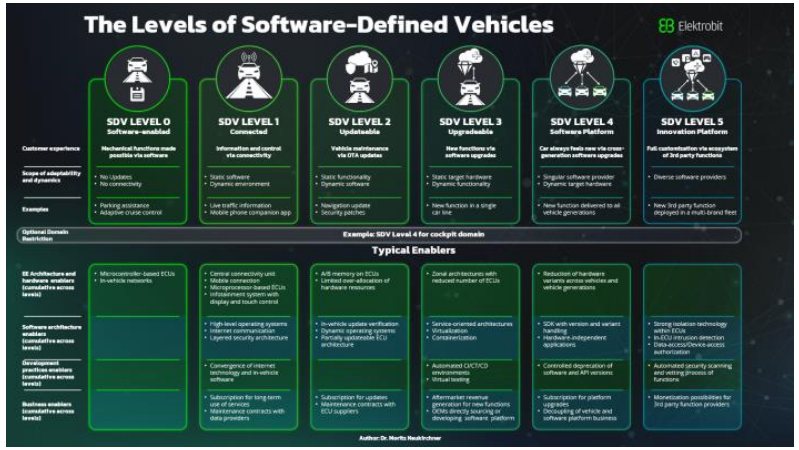
- **Static configuration** of the network and its components, **completely specified for each vehicle variant** and ECU.
- **Network switches are attached to host controller**, not accessible via the network. Any change of an automotive Ethernet switch requires an update of the host controller software.
- Changes of the network configuration are quite complex, **dynamic adaptations** as needed by application changes are **not supported**.



Software-Defined Vehicles & SDN for Automotive

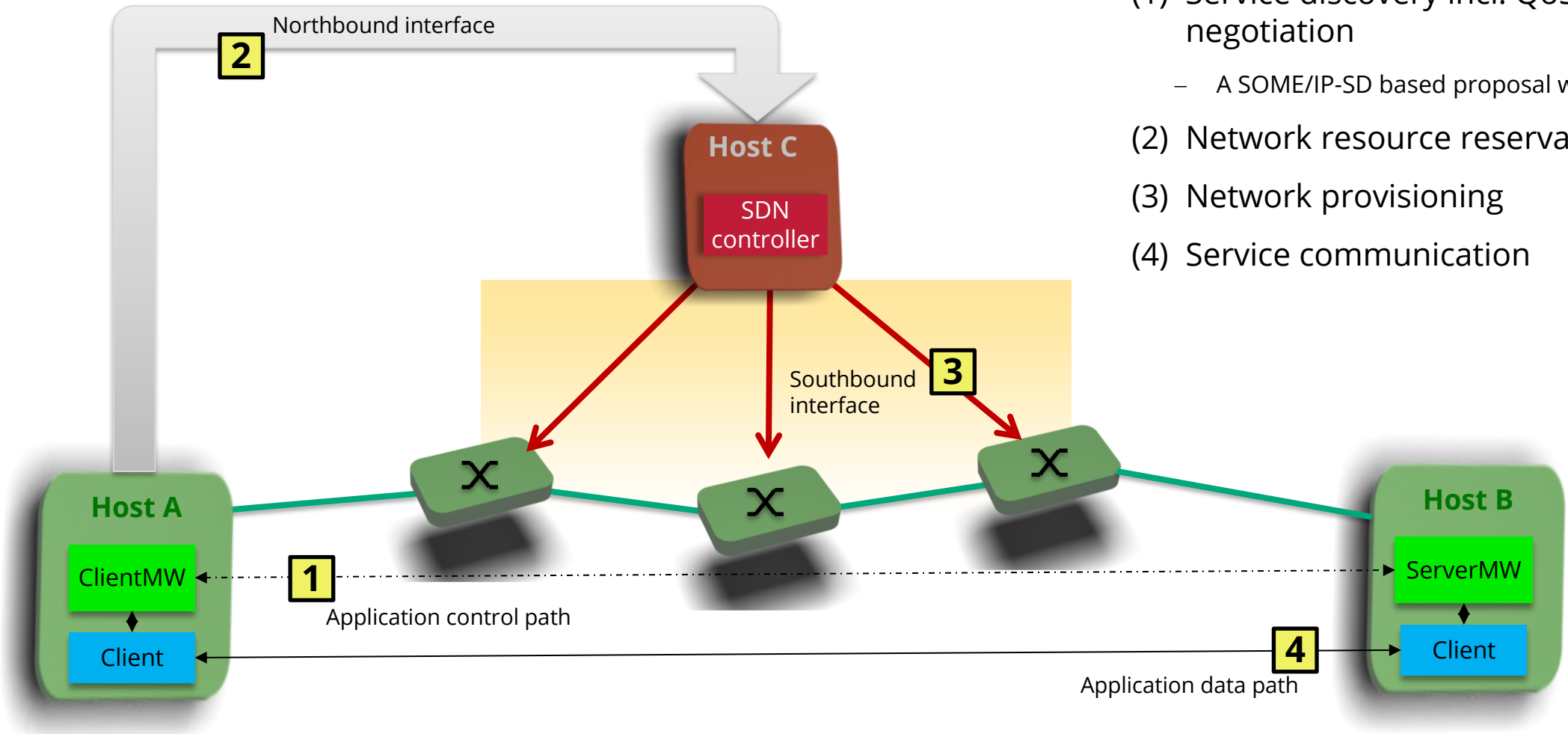
Software-defined vehicles (SDVs) levels

- Level 0: Software is used to run the vehicle's functions
- Level 1: Vehicle is connected to internet to get information like traffic updates, news maps
- Level 2: OTA update of vehicle's SW possible
- **Level 3: Upgrade of functionality via OTA possible – using OEM frameworks**
- **Level 4: SW platform. Forward and backward compatibility: Unplanned upgrading of older vehicle generation possible, as well as using older SW generation in newer vehicles.**
- **Level 5: Open SW platform: Standardized interfaces, contribution of third-party SW providers possible without OEM interaction**



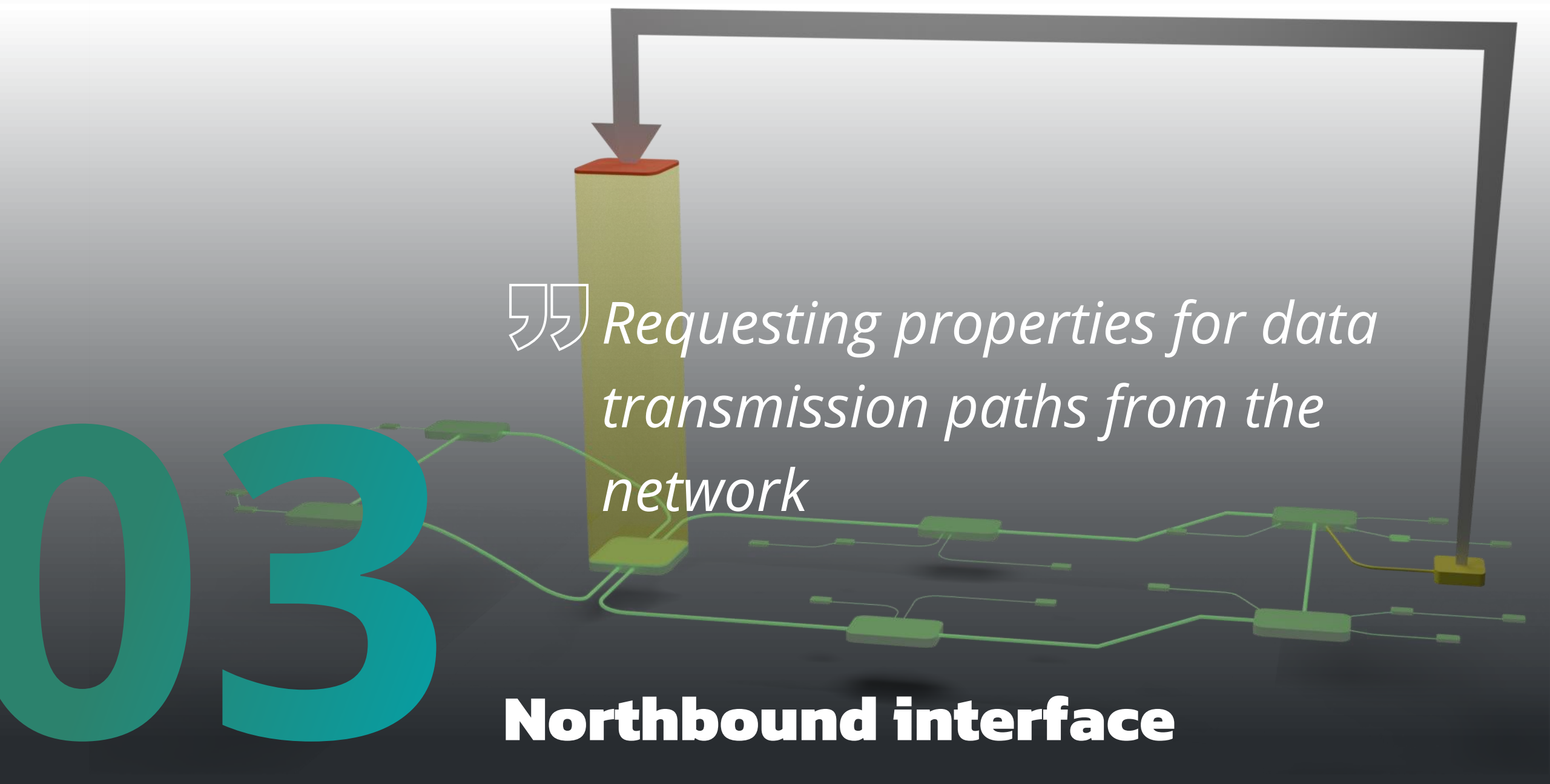
> *Communication needs from unplanned function upgrades must be fulfilled for any vehicle variant.*
>
> *It will be hard, if not impossible, to reach level-4 and -5 with a statically defined network*

SDN architecture for dynamic service integration



(1) Service discovery incl. QoS parameter negotiation
  - A SOME/IP-SD based proposal was presented at [1]
(2) Network resource reservation request
(3) Network provisioning
(4) Service communication

[1] T. Galla, F. Lucut, M. Pleil, D. Thiele, “SDV challenge accepted - The case for automotive software-defined networking (SDN)”, AUTOSAR Open Conference, May, 2025



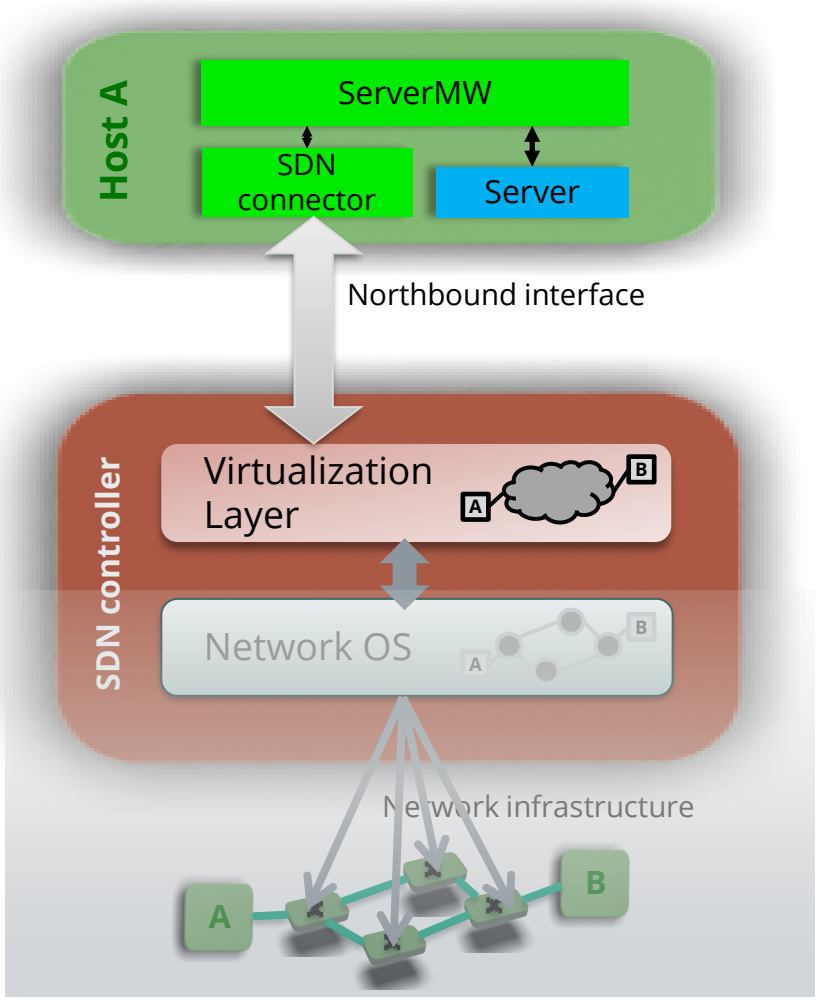
03

Requesting properties for data transmission paths from the network

Northbound interface

Northbound interface

SDN specification abstraction via specification of intent



Offers **high-level configuration abstraction** to managing applications

Example usage

- Request end-to-end connection with a given bandwidth
- Request QoS guarantees (e.g., maximum latency) for this connection

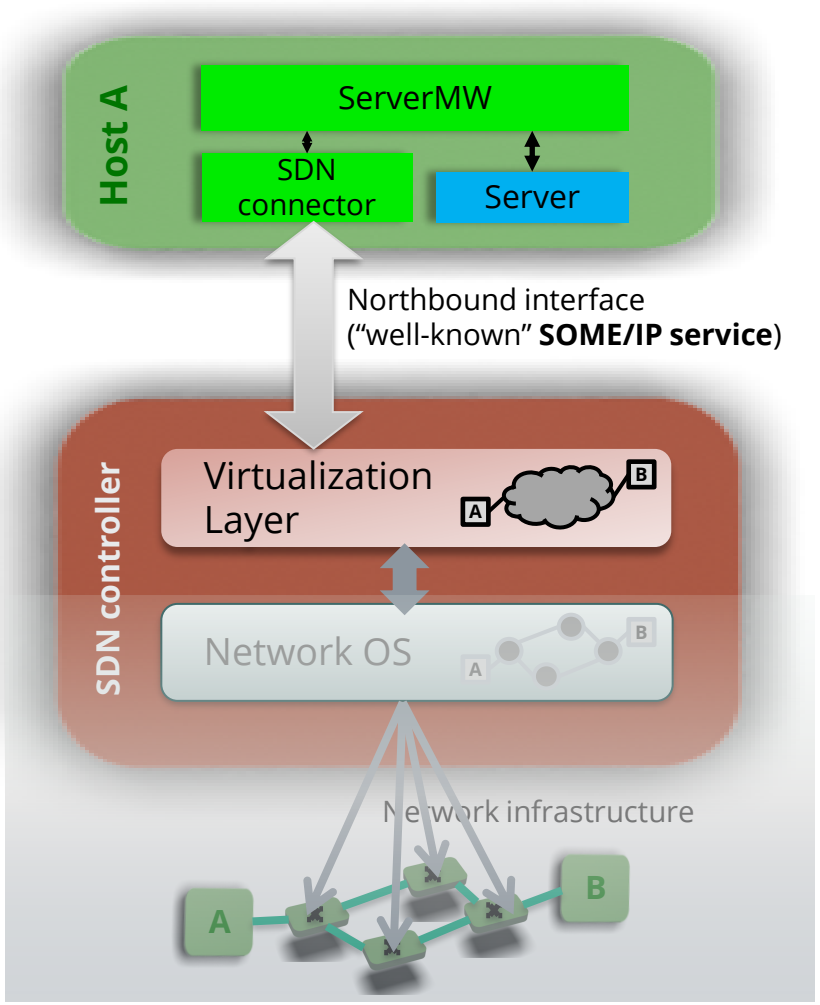
Currently there is no standard for northbound interfaces

- Implementations typically expose functions for
  - Topology information (at different levels of abstraction)
  - Resource reservation requests

Challenge:
Define a “well-known” automotive northbound interface

Northbound interface

SDN specification abstraction via specification of intent

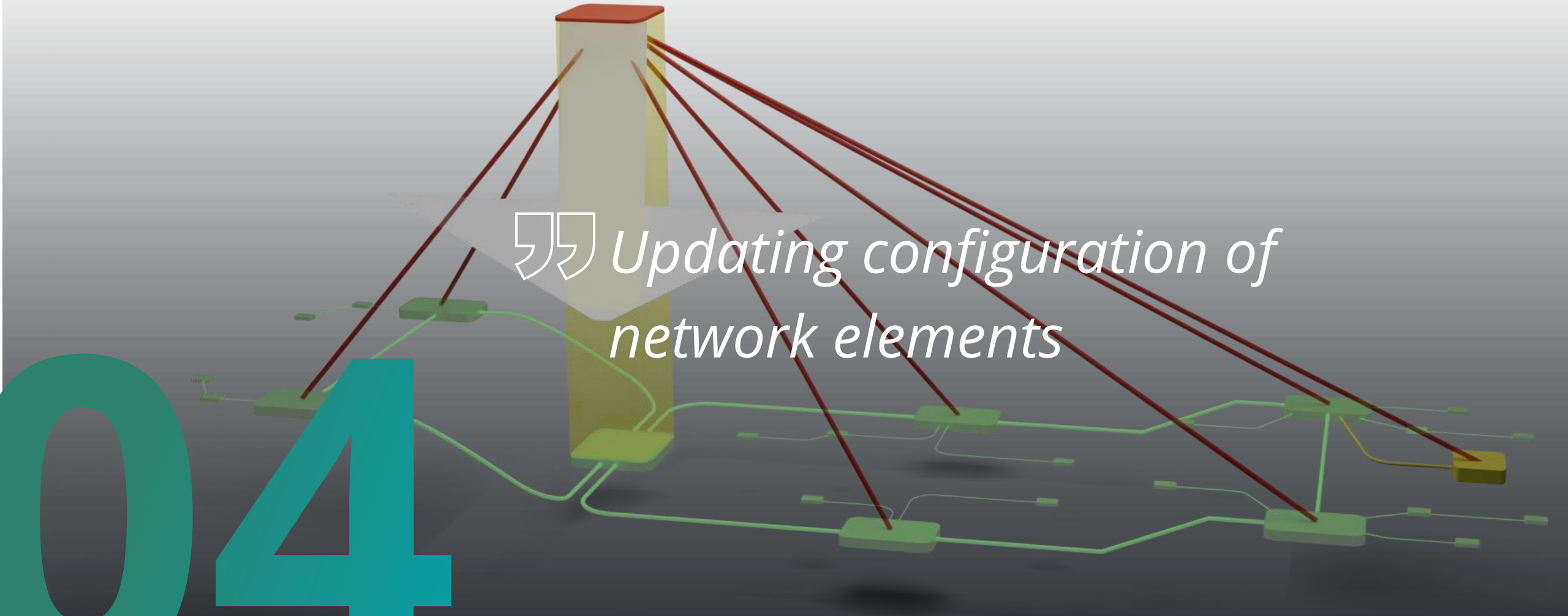


Proposal: Introduce a “well-known” **northbound SOME/IP service with remote methods** to announce intent to

- Send event and/or call methods
- Request resource/bandwidth allocation

Relevant parameters as method arguments

- Traffic flow identification
  - L2/L3/L4 address information and protocol
- Traffic flow specification (“bandwidth”)
  - Maximum payload size (derived from service interface, data types, and address information)
  - Intended update/call cycle
- Traffic flow requirements
  - Desired maximum latency

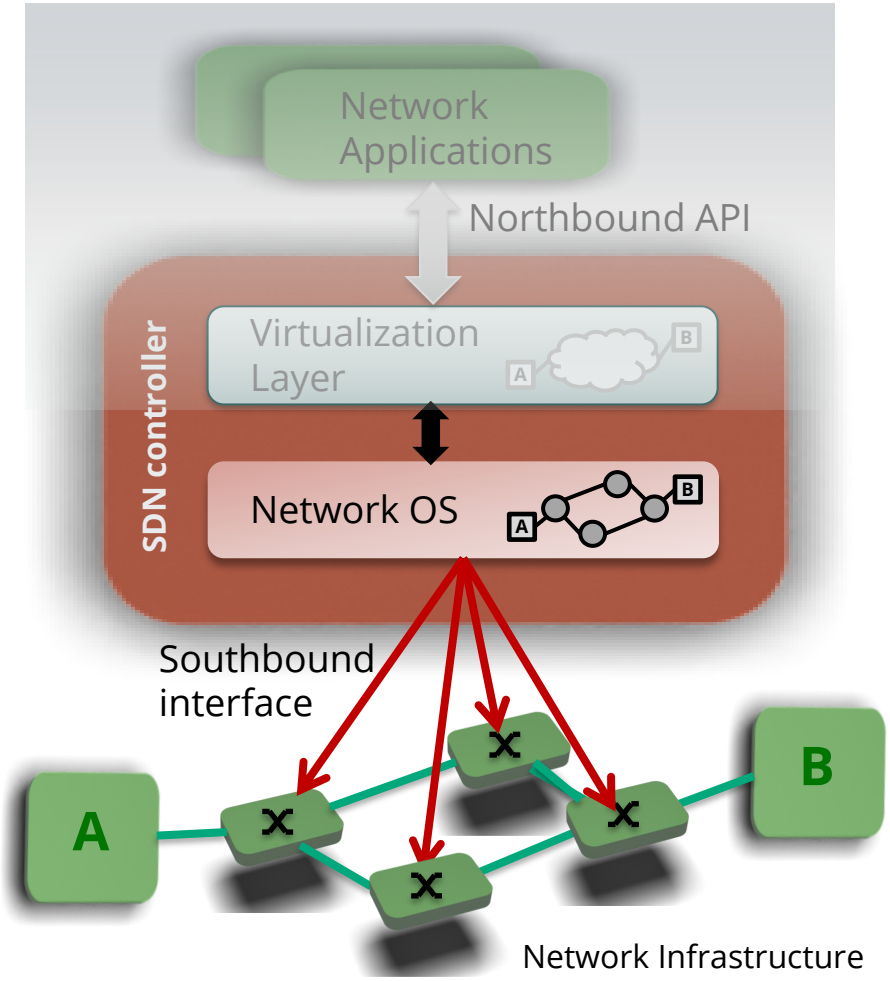


Updating configuration of network elements

04 Southbound interface

Southbound interface

SDN forwarding abstraction



Abstracts implementation details of network devices through data models and management protocols

Example usage

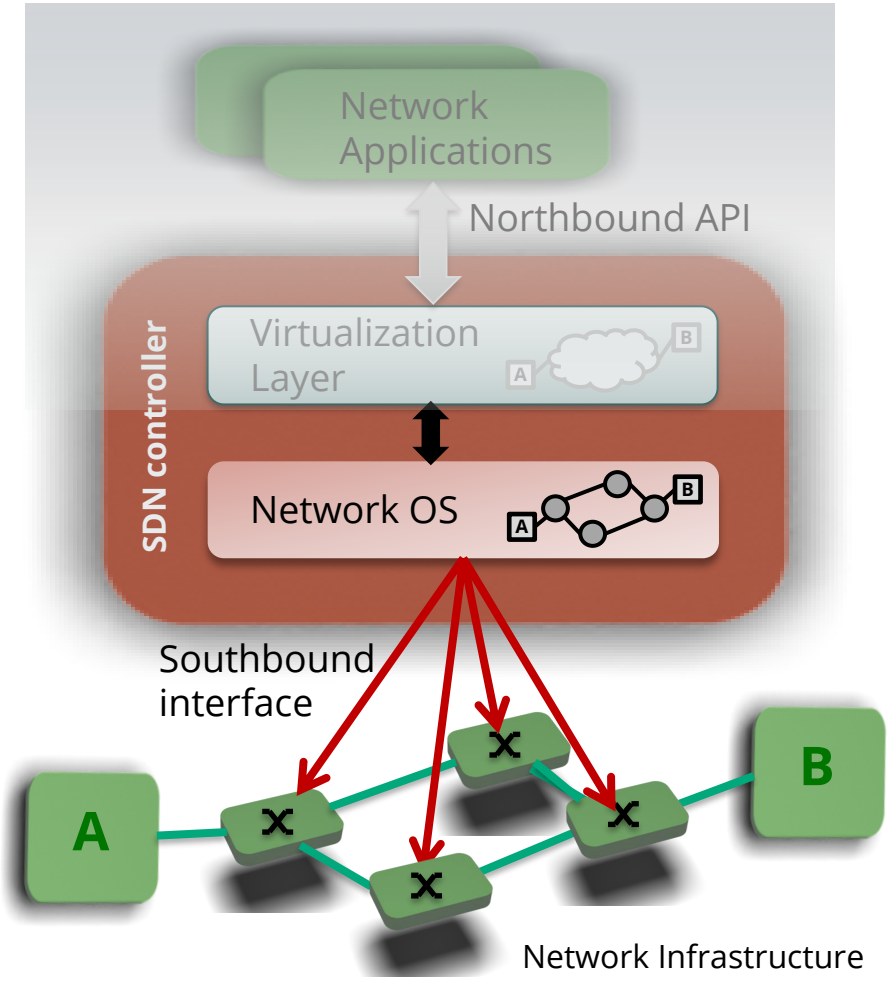
- Establish an end-to-end communication path for a given bandwidth
- Reserve QoS guarantees (e.g., maximum latency) along this path

Data models

- Model configuration data, operational data, remote procedure calls, and notifications
  - E.g. standardized parameter identifiers – no proprietary registers addresses
- Prominent example: YANG
  - Many YANG models available for TSN standards from IETF and IEEE
    - .1Qav (CBS), .1Qbv (TAS), .1Qci (PSFP), .1Qcr (AST), etc.

Southbound interface

SDN forwarding abstraction



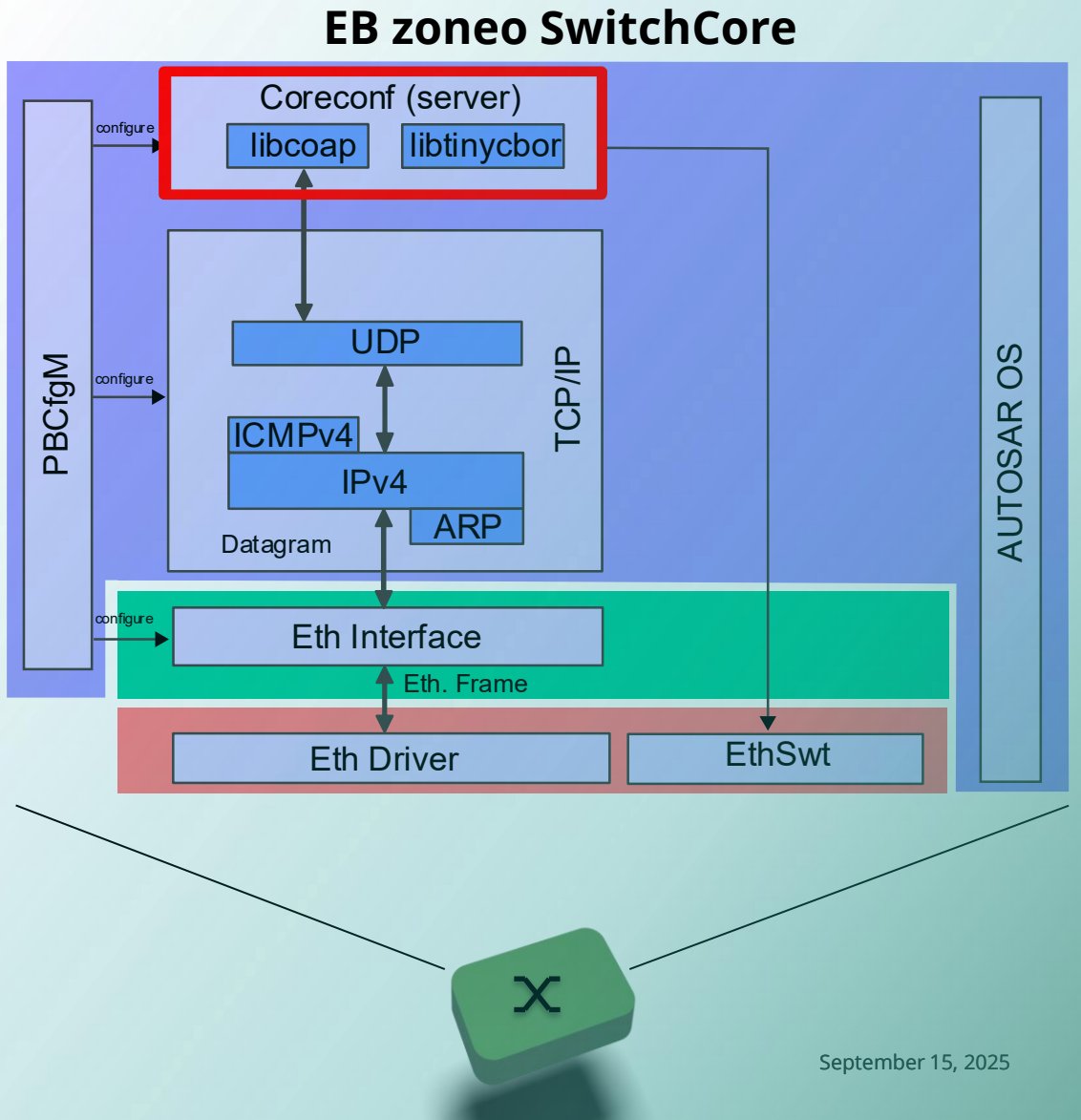
Management protocols

- Mechanisms to install, manipulate, and delete the configuration of network devices
- NETCONF (XML serialized YANG over TCP)
  - Integrated on HPCs
- RESTCONF (XML or JSON serialized YANG over TCP)
- CORECONF (CBOR (binary) serialized YANG over UDP)
  - Integrated into EB SwitchCore firmware for automotive switches

CORECONF implementation

Implementation as EB zoneo SwitchCore extension

- Demo available for Infineon (Marvell) 88Q5152
- 512KB RAM, 512KB ROM
- Resource usage of CORECONF implementation
  - ca. 10% ROM
  - ca. 5% RAM
    (2% statically assigned RAM, 3% heap)



A central authority in the network to consistently update all elements

05

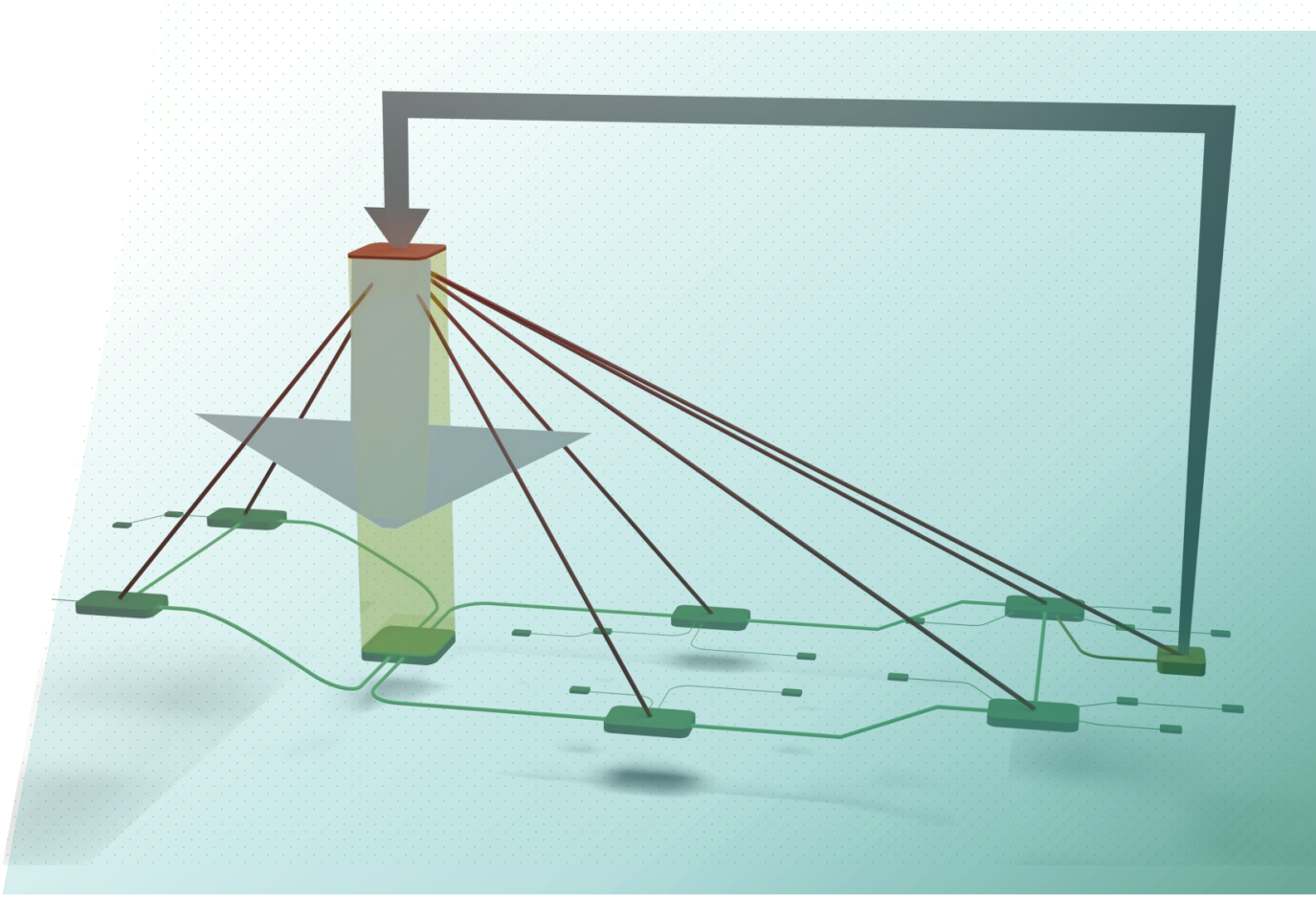
SDN Controller

SDN Network Controller

A SDN controller having full control at runtime over the complete network configuration is the “classic interpretation” following the SDN concept

However, for automotive use-case, this is questionable due to:

- High resource needs
- Difficult timing, leaving unclear network behavior during re-configuration operation
- Risky and hard-to-verify while interacting with safety-critical functions



SDN Network Controller

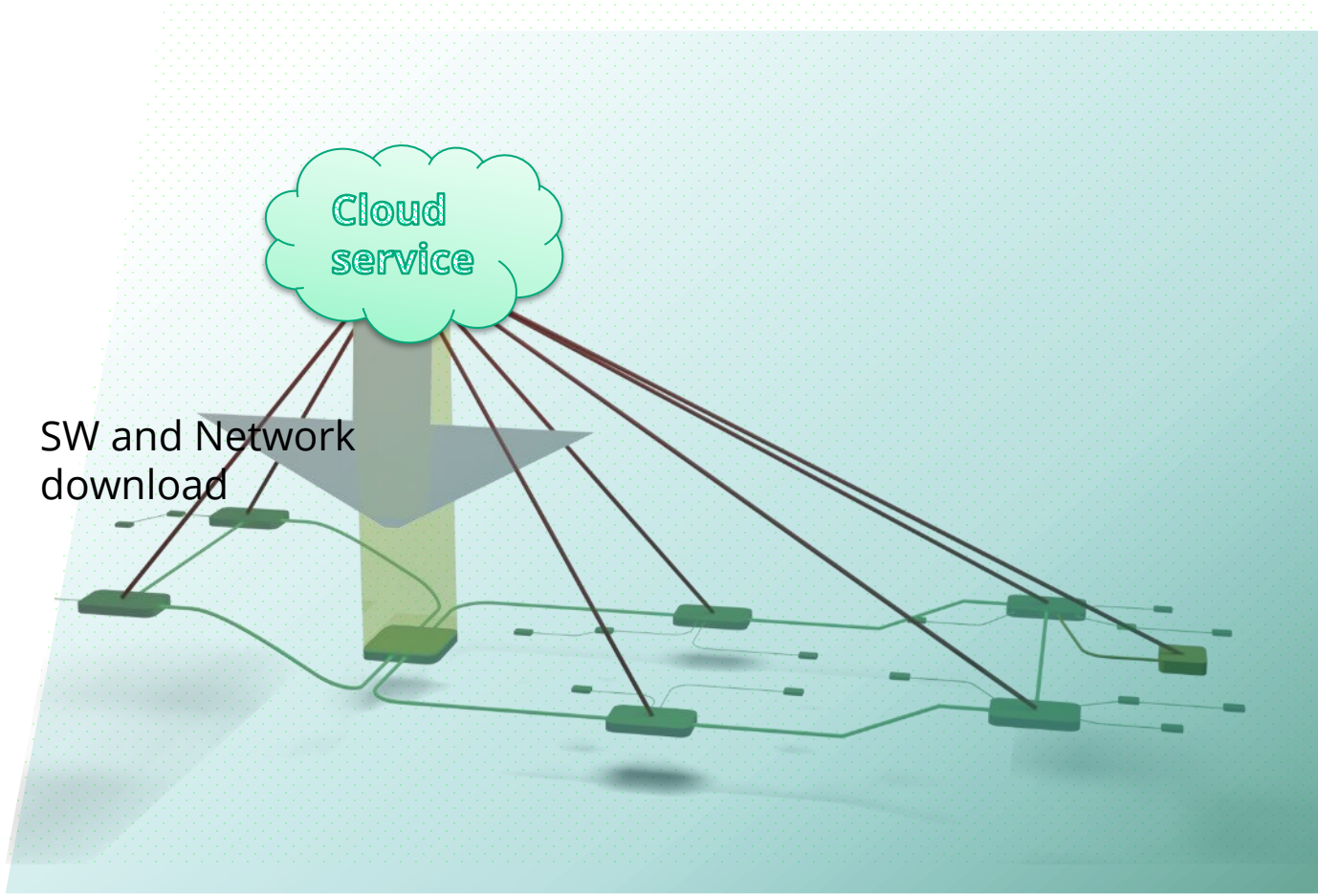
Automotive interpretation proposal

Scenario 1: Major function update

- New HW added
- Significant upgrade of vehicle functionality (offered by OEM)

SDN controller functionality moves into cloud

- New network configuration will be generated based on actual vehicle specification
- Tested and verified in the cloud in a simulation setting
- Downloaded when vehicle is in a safe state (in a garage) together with the SW upgrade



SDN Network Controller

Automotive interpretation proposal

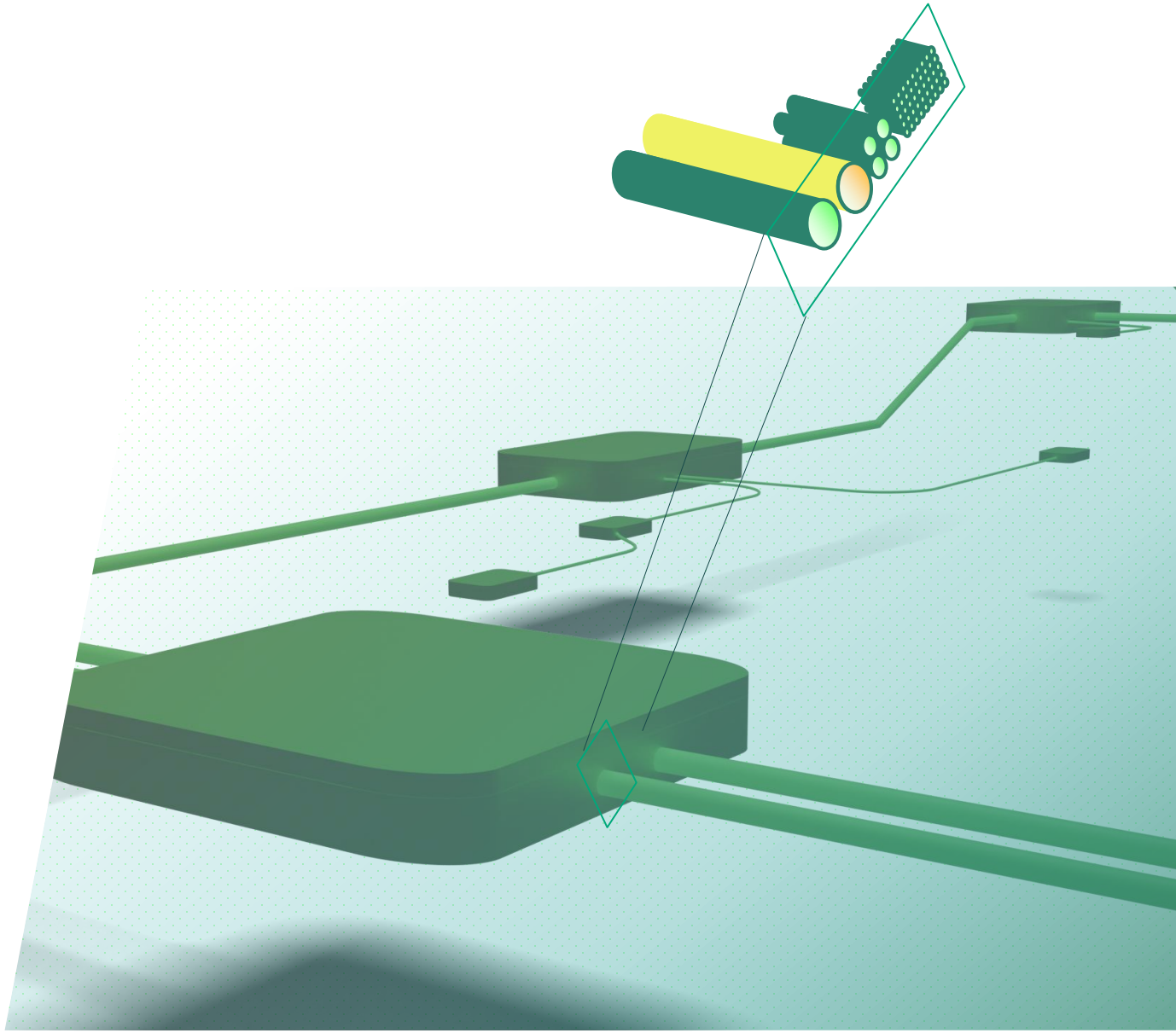
Scenario 2: Initialization of a new service

Network configuration contains reserve for allocation of bandwidth at run-time

- Bandwidth is split into forwarding classes
- Separation of safety-relevant traffic from other forwarding classes can be statically verified – it is never changed at run-time

Service is offered as possible OTA update

- A manifest states the communication requirements
- Before allowing the download, the SDN controller can check if the requirements are met
- At run-time, SDN controller keeps track of resource usage per forwarding class by just counting the number of services in each class

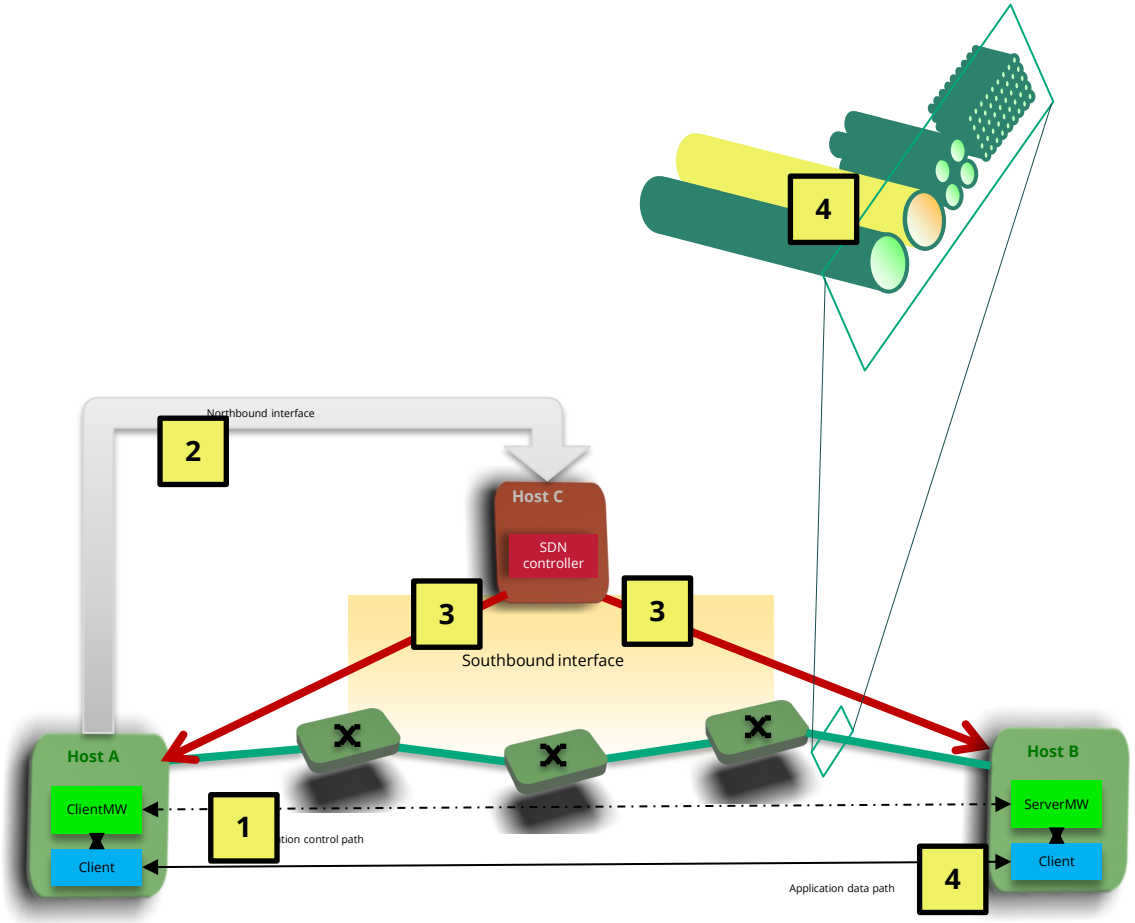


Forwarding classes

Automotive proposal

How this can work:

- As result of a service discovery (1), the SDN controller is requested to assign bandwidth for the service (2).
- Main difference is in step (3): SDN controller instructs the end-nodes to use a certain forwarding class.
- Actual traffic runs over one of the already-defined forwarding classes (4), the network nodes are not touched.



Forwarding classes

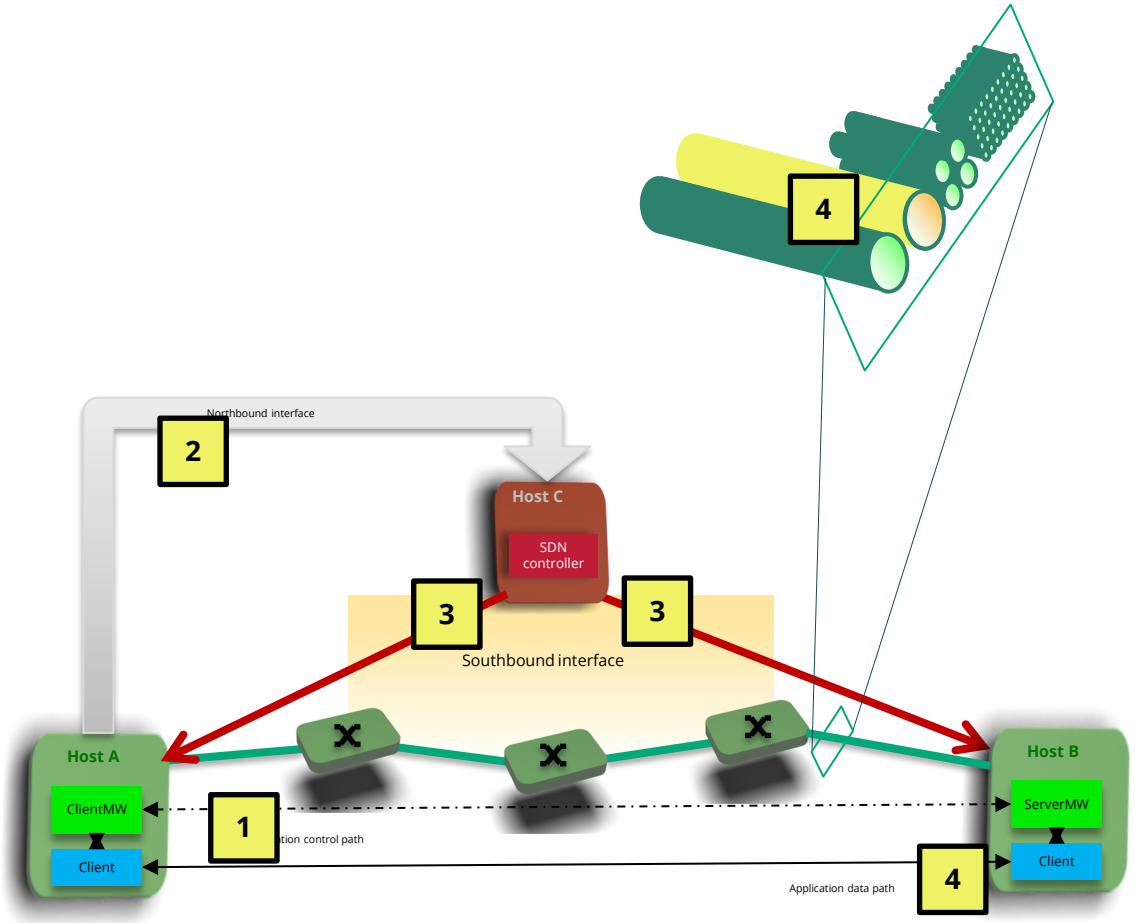
Automotive proposal

How this can work:

- As result of a service discovery (1), the SDN controller is requested to assign bandwidth for the service (2).
- Main difference is in step (3): SDN controller instructs the end-nodes to use a certain forwarding class.
- Actual traffic runs over one of the already-defined forwarding classes (4), the network nodes are not touched.

On the end-station

- Safety-relevant applications (typically running on Classic AUTOSAR cores) have statically defined network configuration
- Infotainment and user-downloadable applications (typically running on AUTOSAR Adaptive or Linux cores) organize a shaper to split up the local senders according to forwarding classes and ensure fairness among several senders of the same forwarding class.



Forwarding classes

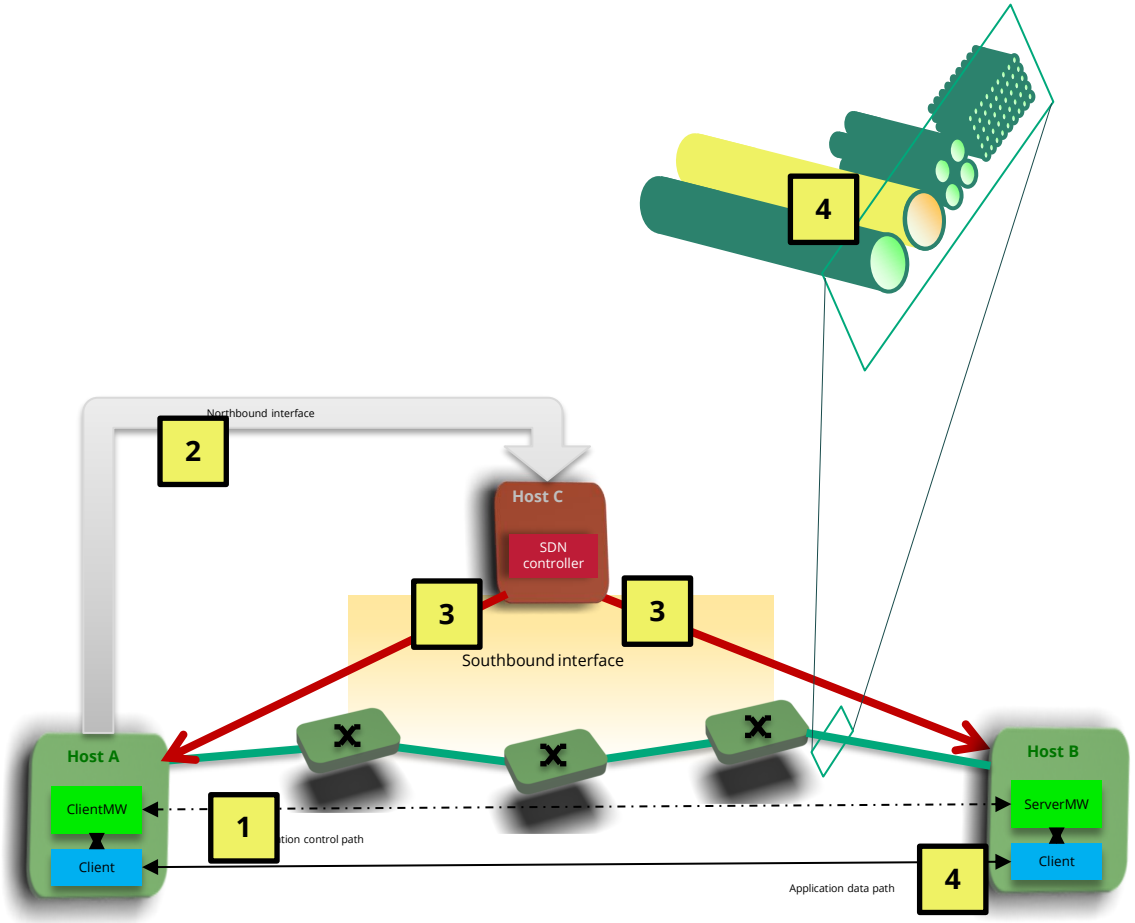
Automotive proposal

How this can work:

- As result of a service discovery (1), the SDN controller is requested to assign bandwidth for the service (2).
- Main difference is in step (3): SDN controller instructs the end-nodes to use a certain forwarding class.
- Actual traffic runs over one of the already-defined forwarding classes (4), the network nodes are not touched.

On the end-station

- Safety-relevant applications (typically running on Classic AUTOSAR cores) have statically defined network configuration
- Infotainment and user-downloadable applications (typically running on AUTOSAR Adaptive or Linux cores) organize a shaper to split up the local senders according to forwarding classes and ensure fairness among several senders of the same forwarding class.



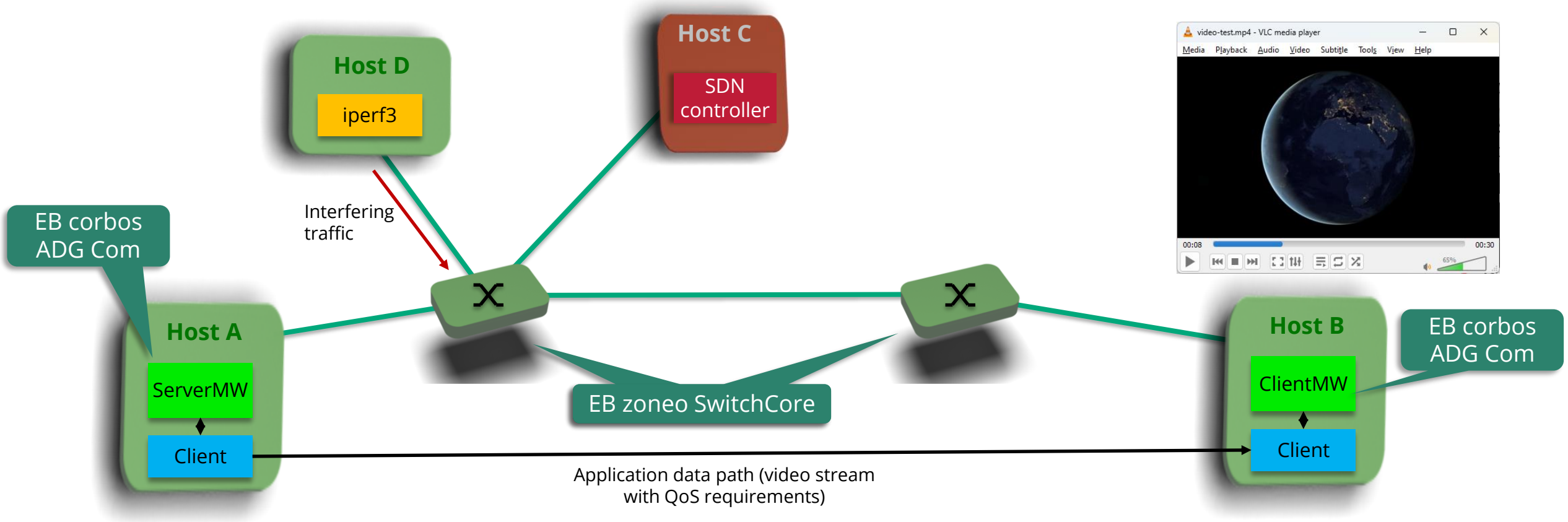
On switches

- Forwarding classes are pre-configured
- Implementation using time-aware shapers and/or credit-based shapers are possible
- Ingress policers (pre-configured) can be added to ensure that end-nodes do not exceed the bandwidth allocated to the respective forwarding class

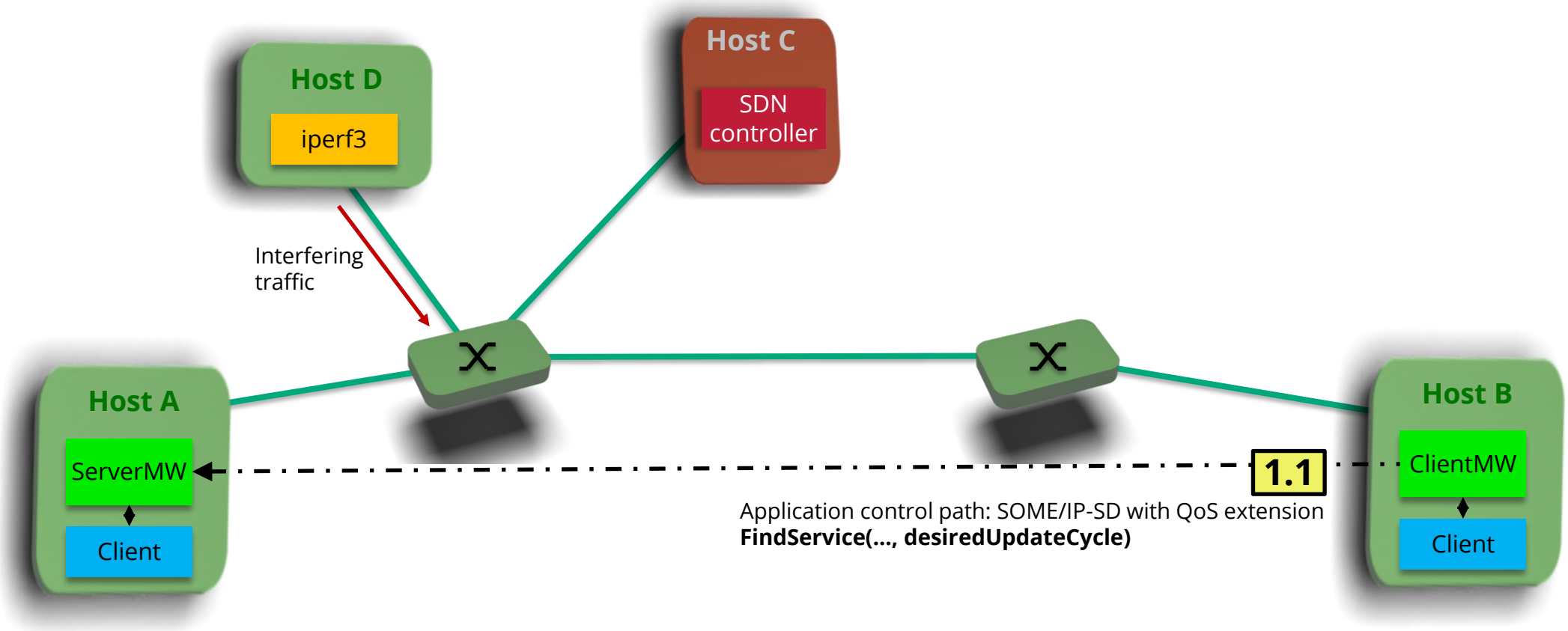
06 Demo

Demonstrator

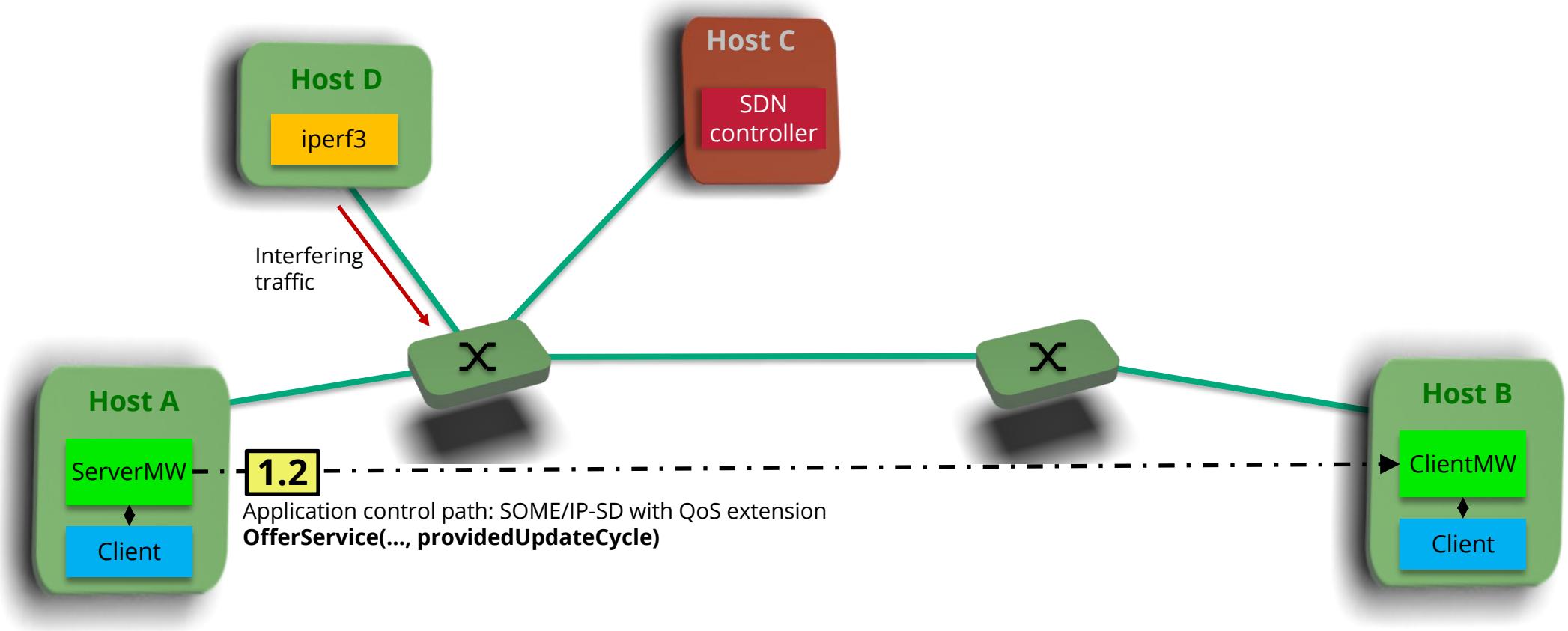
Transport time-critical video stream from host A to host B in the presence of interfering traffic from host D



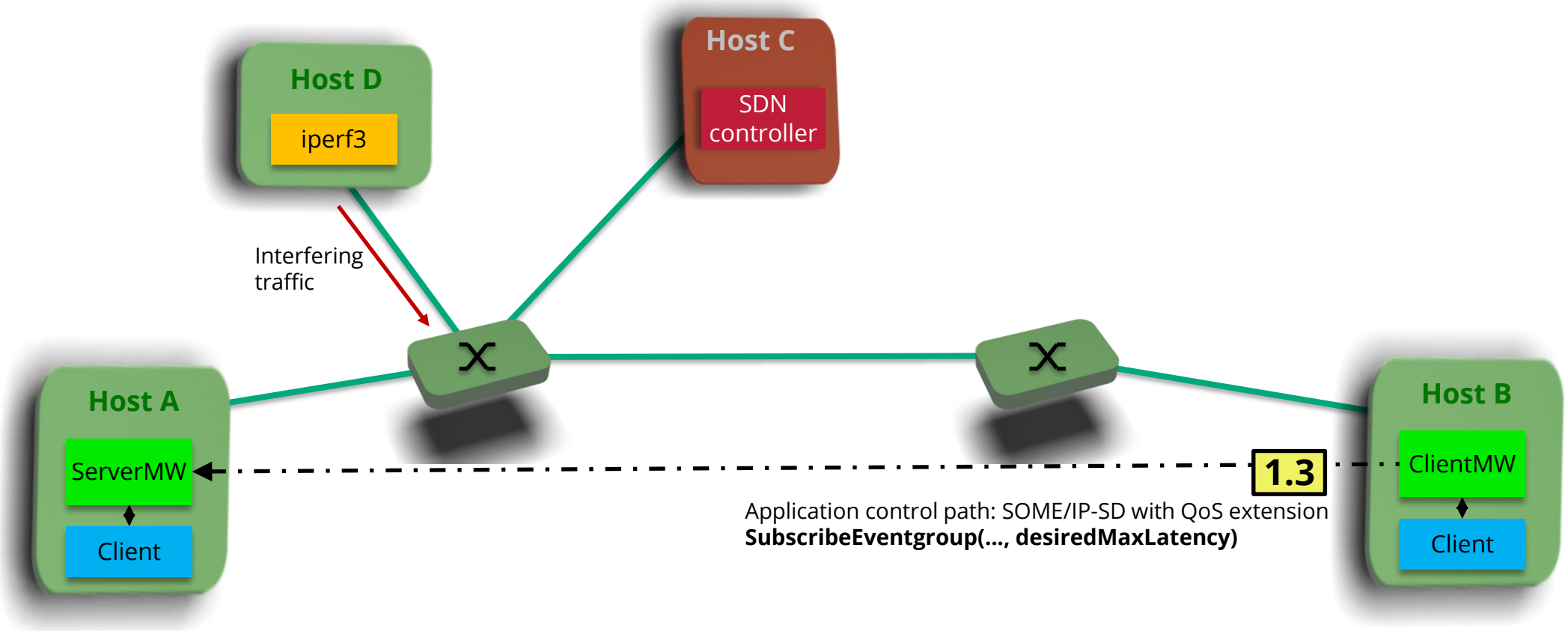
Demonstrator



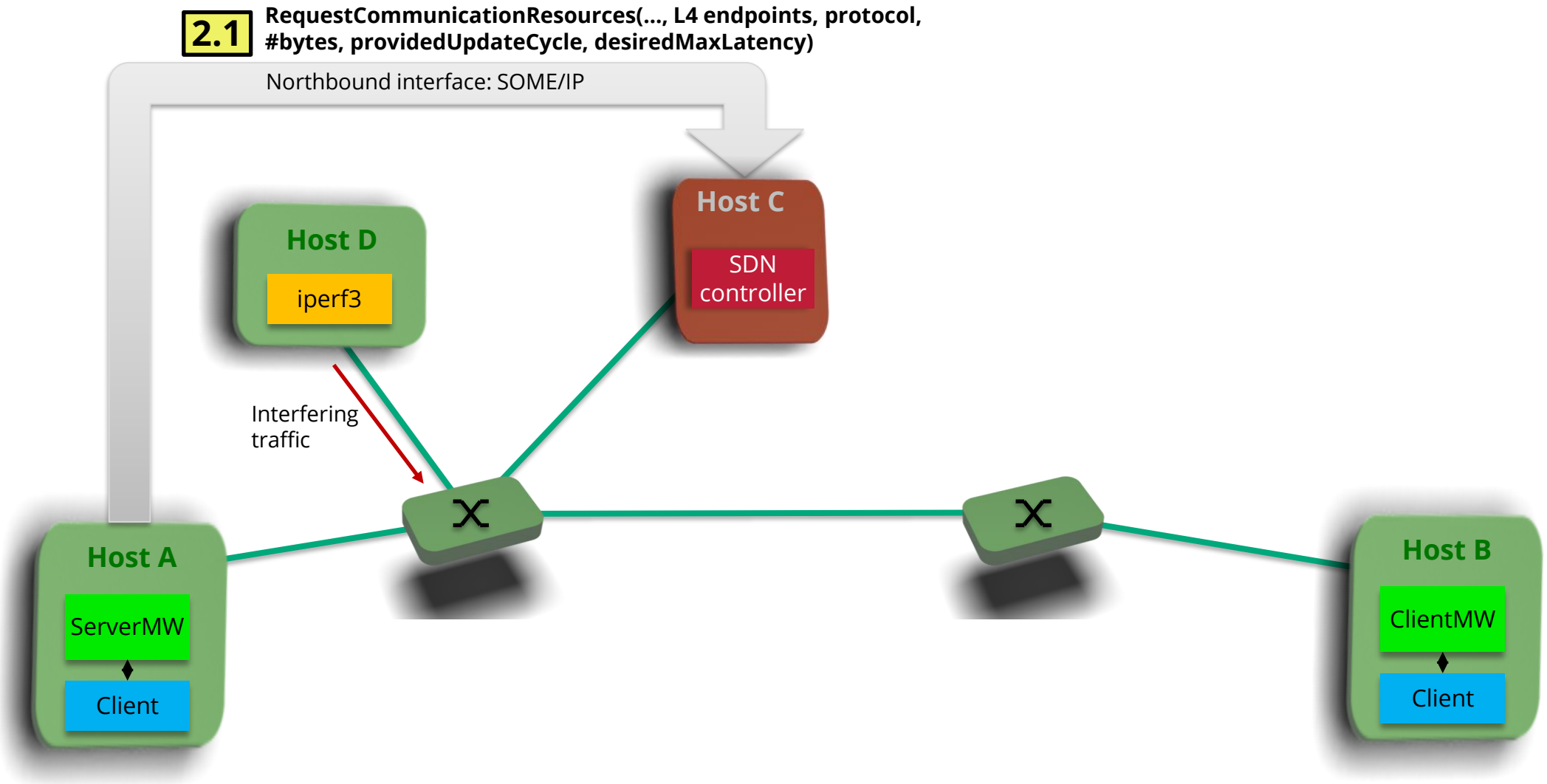
Demonstrator



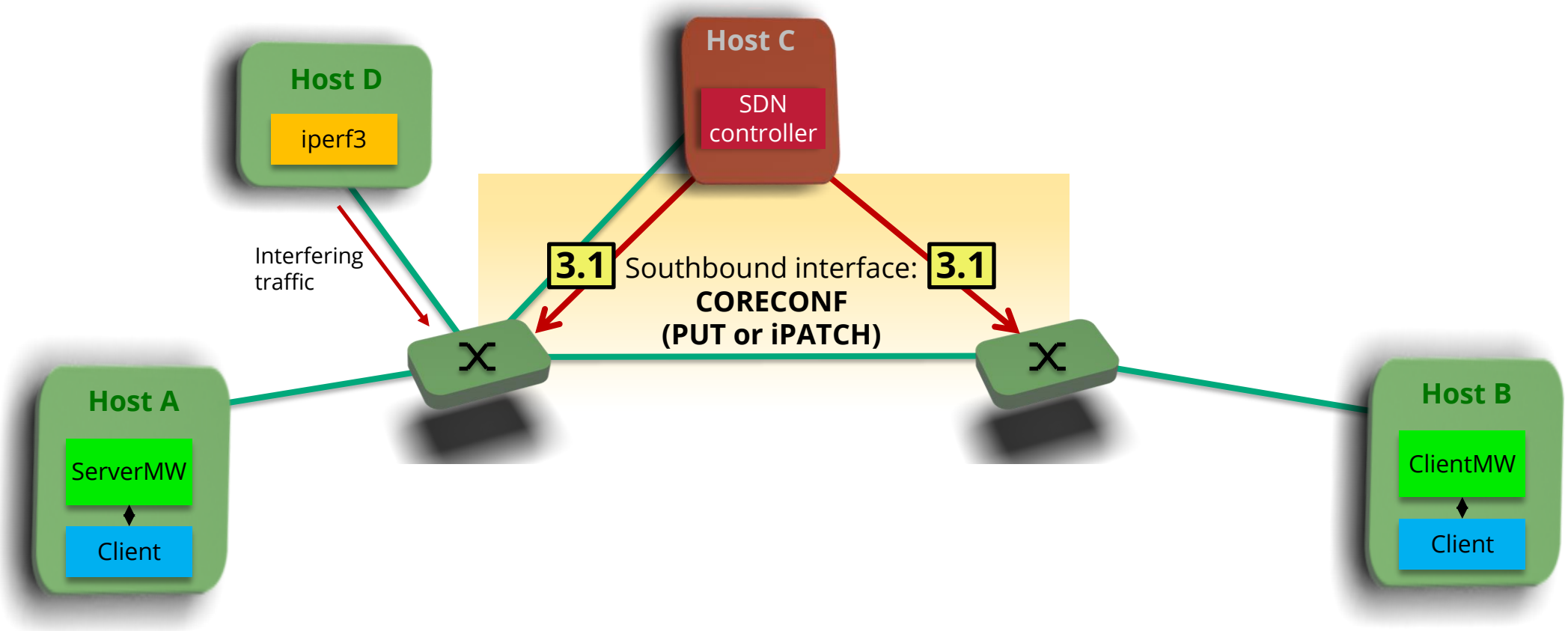
Demonstrator



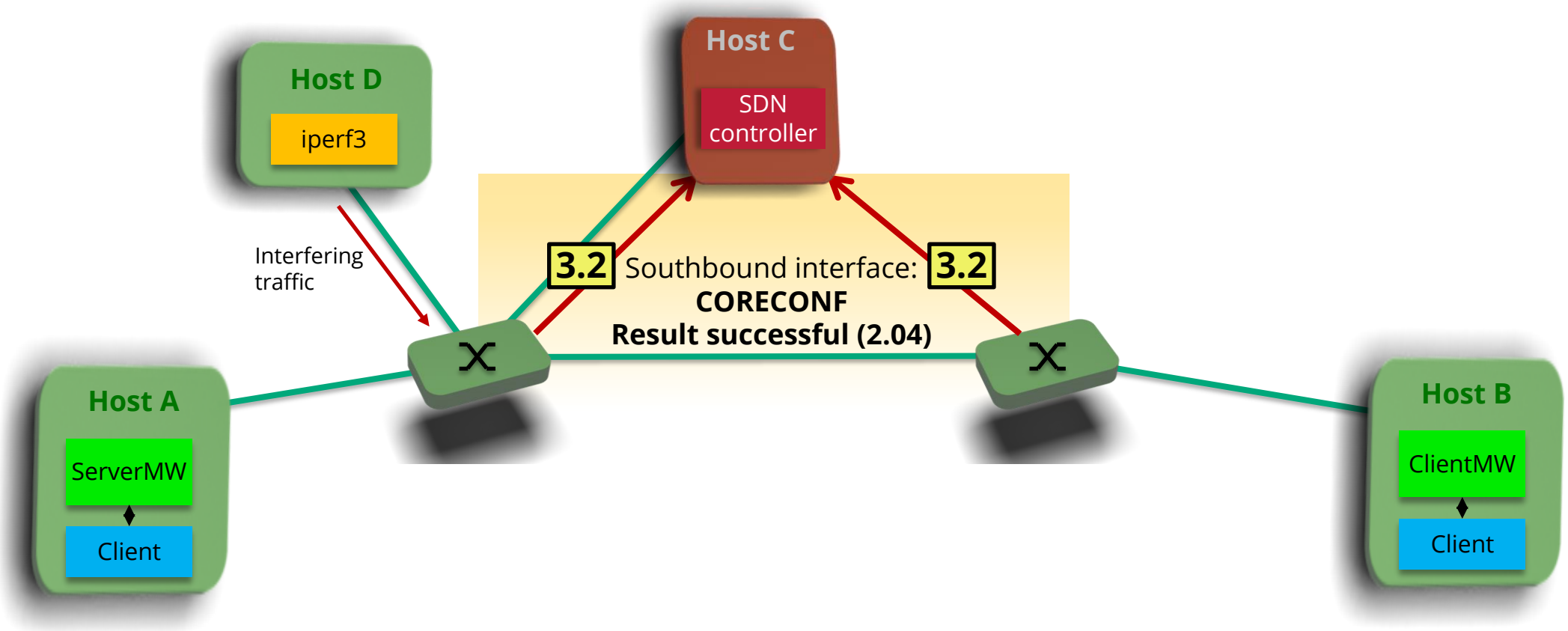
Demonstrator



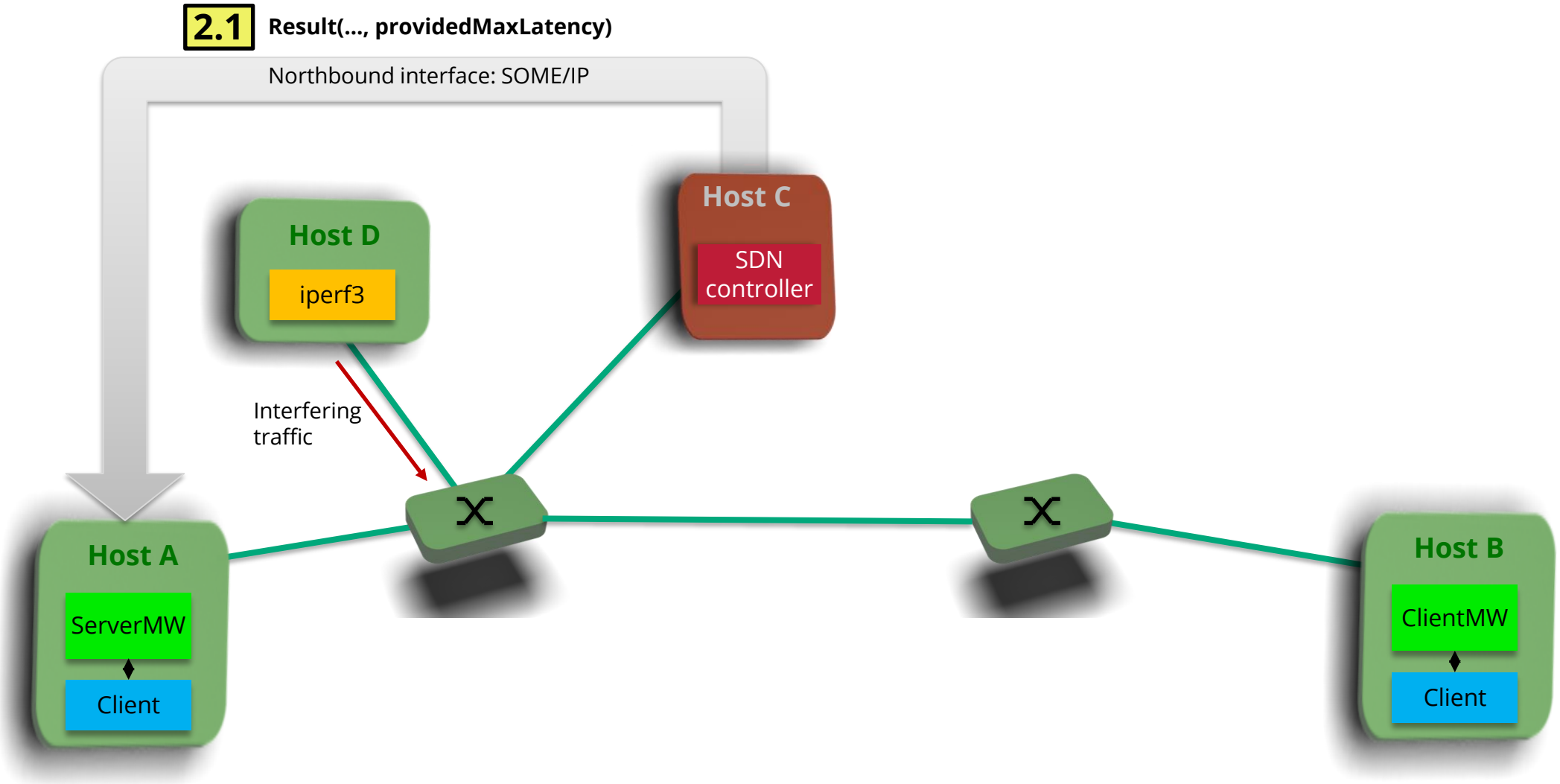
Demonstrator



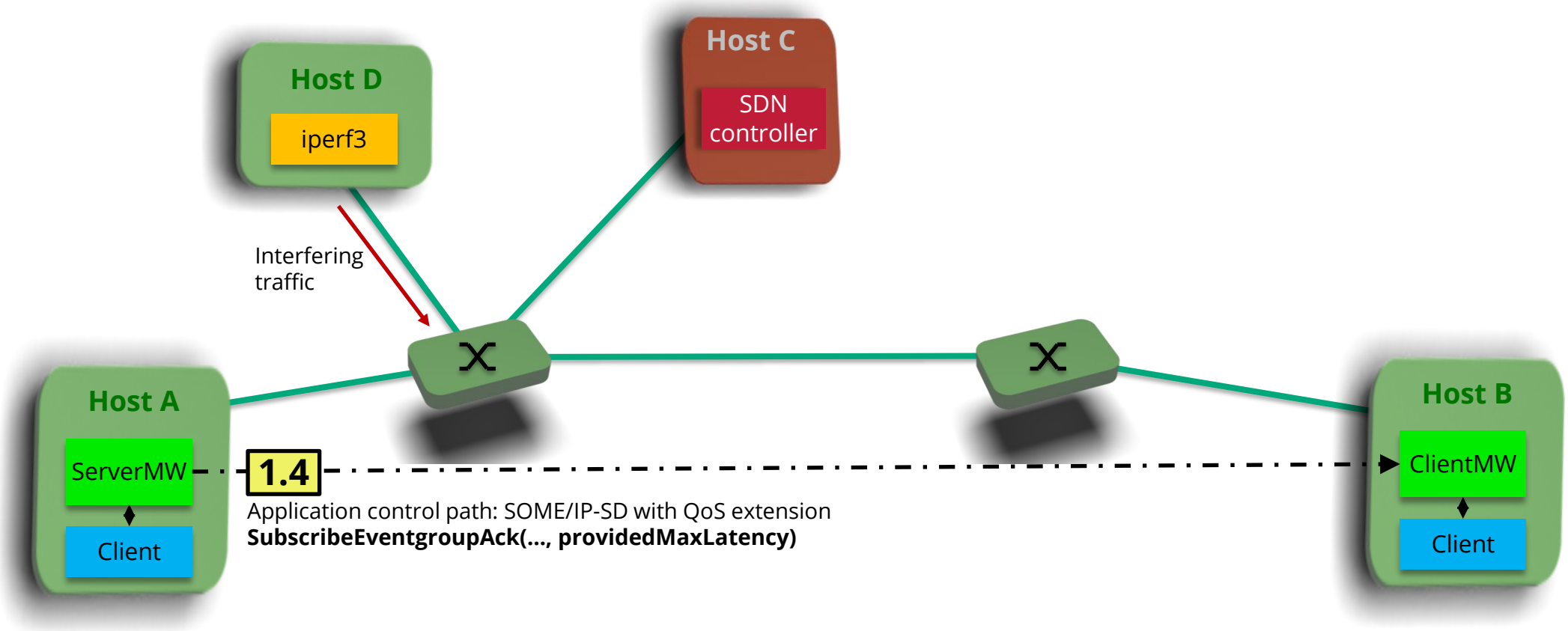
Demonstrator



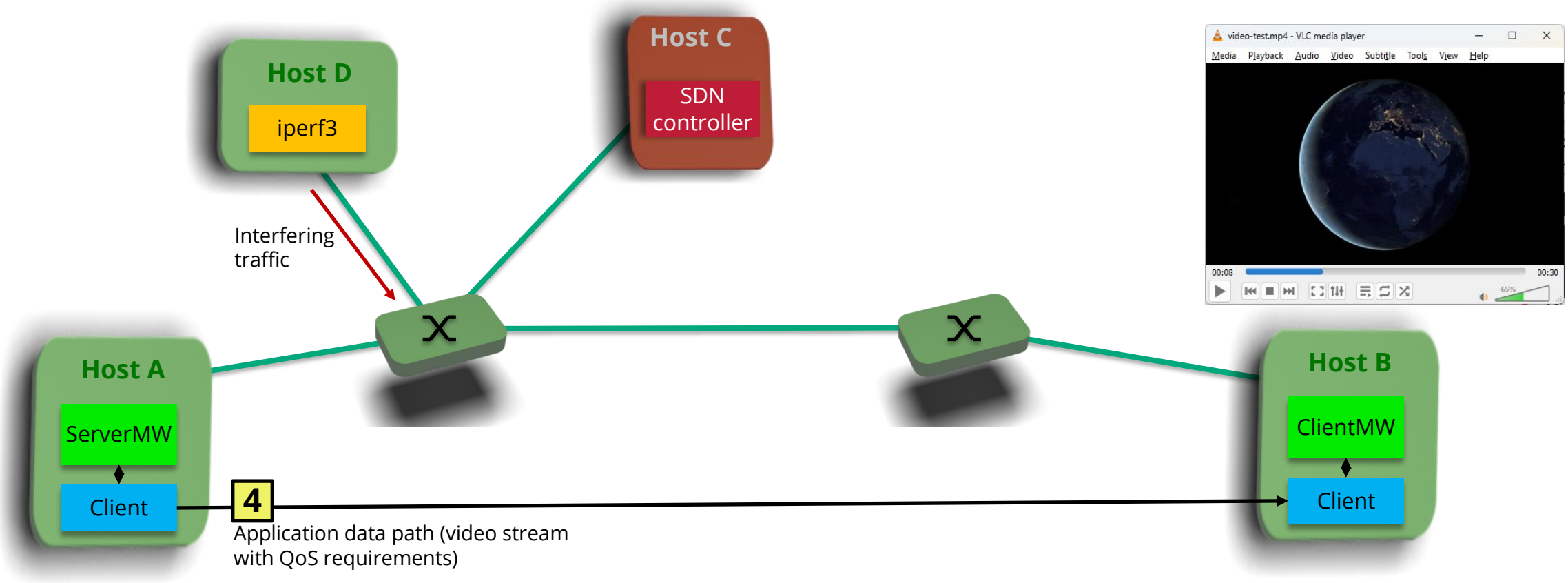
Demonstrator



Demonstrator



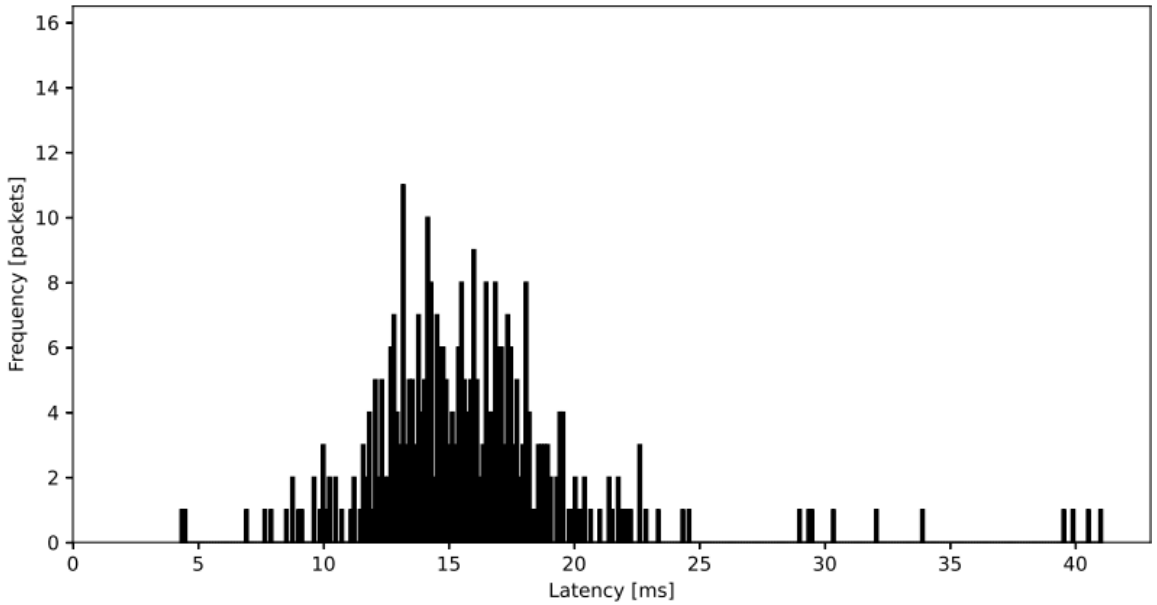
Demonstrator



Demonstrator

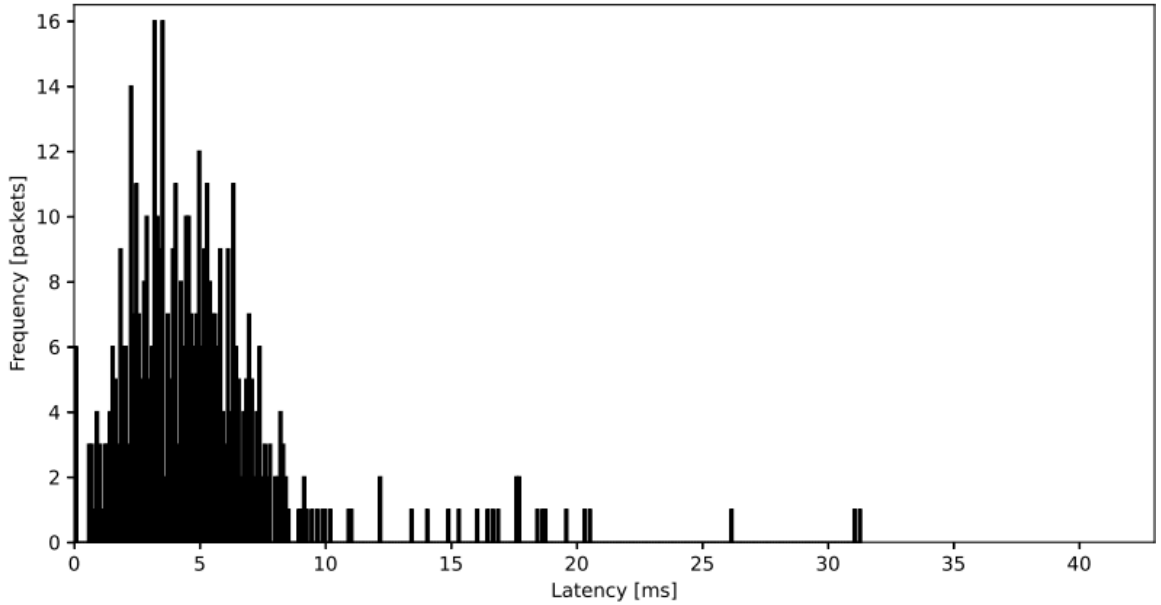
Negotiating and Enforcing Quality of Service (QoS) Attributes - Results

Baseline without QoS



- All traffic at priority 0
- Critical and non-critical traffic share the same egress queues in the switches
- About 10% frame loss of the critical traffic in the first switch

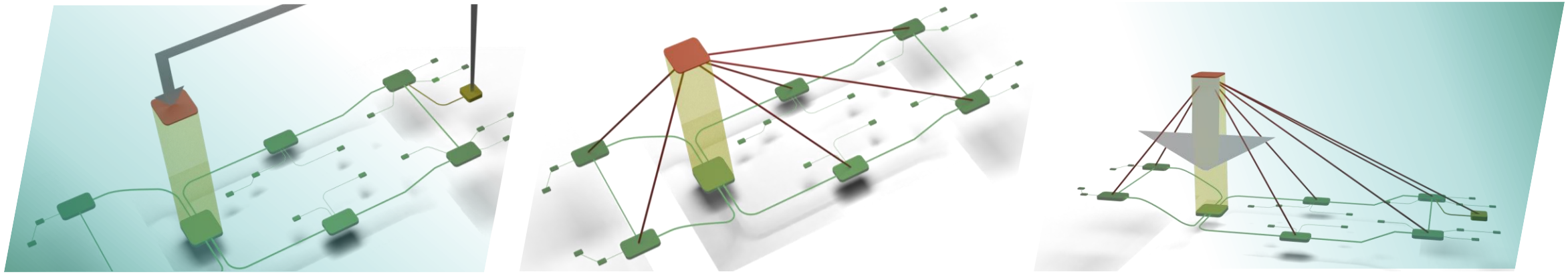
SDN controlled with privileged forwarding class



- Critical traffic at priority 7, interfering traffic at priority 0
- Each traffic flow uses a dedicated queue in the switches
- 0% frame loss of critical traffic in the first switch

Summary

Automotive SDN is possible



Northbound interface

Request network services without dealing with the “how”.

→ Extension of SOME/IP Service-Discovery

SDN controller

Control the network parametrization from a central authority

→ A TSN-based network pre-configuration with fixed forwarding classes enables a simple automotive interpretation of the SDN controller

→ A central contact point to only adapt network configuration greatly decreases SDN complexity

Southbound interface

Connect to all network nodes to enable central parametrization

→ Established protocols (i.e. CORECONF/YANG) that fulfill automotive requirements exist

Contact us

Carsten Weich

Product Owner
Automotive Networks

Elektrobit – Our software moves the world

Carsten.Weich@elektrobit.com
elektrobit.com

Daniel Thiele

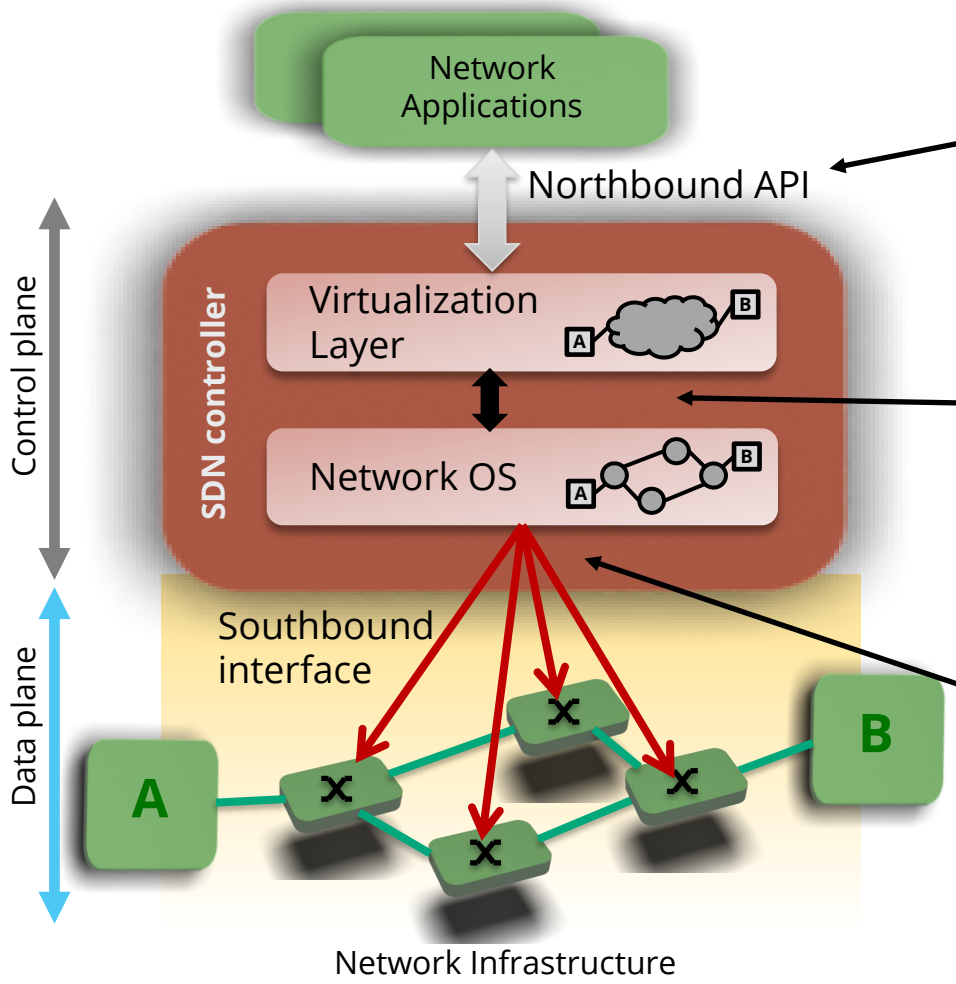
Senior Expert
Automotive Networks

Elektrobit – Our software moves the world

Daniel.Thiele@elektrobit.com
elektrobit.com

BACKUP

SDN architecture and its fundamental abstractions



SDN is the abstraction of the control plane

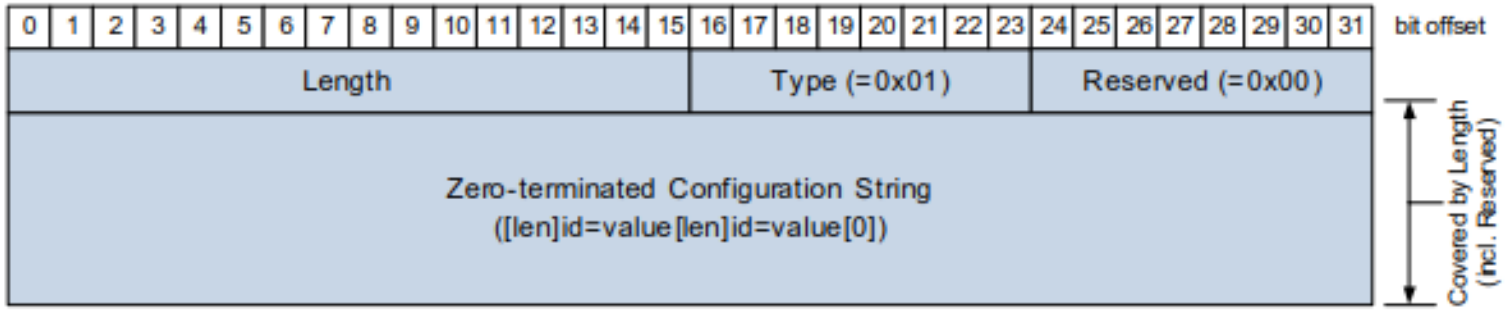
- Specification abstraction
  - Express communication intent (not implementation)
  - Abstract graph with just enough detail to specify communication goals
    - Example: Let A talk to B with a given QoS
- Network state abstraction
  - Transform distributed control into logically centralized control
  - Abstract network and its distributed state as a simple data structure (e.g., annotated graph of the network)
    - Example: Direct use graph algorithms
- Forwarding abstraction
  - Abstract details of underlying hardware (e.g., vendor-specific low-level forwarding configuration)

SOME/IP-SD with QoS parameter negotiation

QoS parameter encoding

Goal: embed QoS parameter negotiation into existing SOME/IP-SD protocol

- Leverage an existing SOME/IP-SD feature: configuration options (a.k.a. capability records)
- May carry arbitrary additional information as key-value pairs 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 (desiredMaxLatency, DesiredUpdateCycle, etc.)

```
▾ SOME/IP Service Discovery Protocol
  ▸ Flags: 0xc0, Reboot Flag, Unicast Flag
    Reserved: 0x000000
    Length of Entries Array: 16
  ▸ Entries Array
    Length of Options Array: 73
  ▾ Options Array
    ▸ 0: IPv4 Endpoint Option (192.168.88.77:60000 (UDP))
    ▾ 1: Configuration Option
        Length: 58
        Type: 1
        Reserved: 00
      ▾ Configuration String: \vevent=32769\x15desiredUpdateCycle=10\x15desiredMaxLatency=100
          Configuration String Element: event=32769
          Configuration String Element: desiredUpdateCycle=10
          Configuration String Element: desiredMaxLatency=100
```

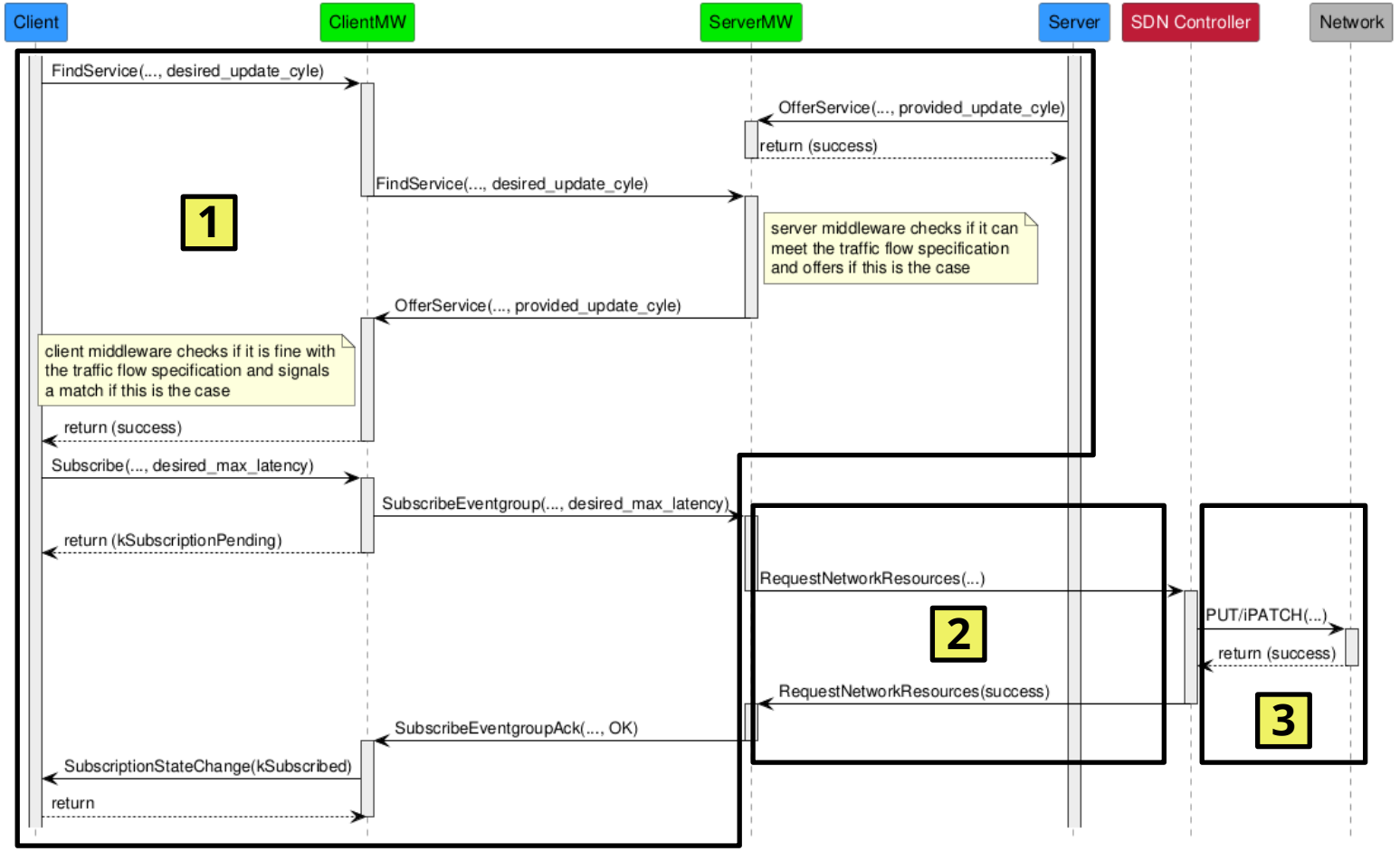
SDN based dynamic service integration in detail

(1) QoS parameter negotiation via SOME/IP-SD extension [1]
- Configuration options
- Find/Offer: Match communication cycles
- Subscription: Maximum communication latency

(2) Network resource reservation request via SOME/IP service
- Service for abstract resource reservation request: Path with QoS

(3) Network provisioning via CORECONF
- SDN controller translates abstract request to device configurations and provisions devices across the network

(4) Service communication (not shown)
- Between client and server



[1] T. Galla, F. Lucut, M. Pleil, D. Thiele, “SDV challenge accepted - The case for automotive software-defined networking (SDN)”, AUTOSAR Open Conference, May, 2025

Southbound Interface

Management protocols

Network Configuration Protocol (NETCONF) [IETF RFC 6241]

– XML encoded YANG instance as data

RESTCONF [IETF RFC 8040]

– XML or JSON encoded YANG instance as data

Constrained Application Protocol (CoAP) Management Interface (CORECONF) [draft-ietf-core-comi-20]

– CBOR encoded YANG instance as data

Exchange Protocol	Standard	Transport Protocol	Transaction Support	Constrained Devices	Fine Grained Update	Data Model	Encoding
NETCONF	IEEE/IETF/W3C	TCP	✓	×	✓	YANG	XML
RESTCONF	IEEE/IETF/ISO/W3C	TCP	×	×	✓	YANG	XML & JSON
CORECONF	IEEE/IETF	**UDP**/TCP/WebSockets	×	✓	✓	YANG	CBOR (binary)

The Levels of Software-Defined Vehicles

Elektrobit

SDV LEVEL 0
Software-enabled

Customer experience: Mechanical functions made possible via software

Scope of adaptability and dynamics:
- No Updates
- No connectivity

Examples:
- Parking assistance
- Adaptive cruise control

SDV LEVEL 1
Connected

Customer experience: Information and control via connectivity

Scope of adaptability and dynamics:
- Static software
- Dynamic environment

Examples:
- Live traffic information
- Mobile phone companion app

SDV LEVEL 2
Updateable

Customer experience: Vehicle maintenance via OTA updates

Scope of adaptability and dynamics:
- Static functionality
- Dynamic software

Examples:
- Navigation update
- Security patches

SDV LEVEL 3
Upgradeable

Customer experience: New functions via software upgrades

Scope of adaptability and dynamics:
- Static target hardware
- Dynamic functionality

Examples:
- New function in a single car line

SDV LEVEL 4
Software Platform

Customer experience: Car always feels new via cross-generation software upgrades

Scope of adaptability and dynamics:
- Singular software provider
- Dynamic target hardware

Examples:
- New function delivered to all vehicle generations

SDV LEVEL 5
Innovation Platform

Customer experience: Full customization via ecosystem of 3rd party functions

Scope of adaptability and dynamics:
- Diverse software providers

Examples:
- New 3rd party function deployed in a multi-brand fleet

Optional Domain Restriction: Example: SDV Level 4 for cockpit domain

Typical Enablers

	SDV LEVEL 0	SDV LEVEL 1	SDV LEVEL 2	SDV LEVEL 3	SDV LEVEL 4	SDV LEVEL 5
EE Architecture and hardware enablers (cumulative across levels)	• Microcontroller-based ECUs • In-vehicle networks	• Central connectivity unit • Mobile connection • Microprocessor-based ECUs • Infotainment system with display and touch control	• A/B memory on ECUs • Limited over-allocation of hardware resources	• Zonal architectures with reduced number of ECUs	• Reduction of hardware variants across vehicles and vehicle generations	
Software architecture enablers (cumulative across levels)		• High-level operating systems • Internet communication • Layered security architecture	• In-vehicle update verification • Dynamic operating systems • Partially updateable ECU architecture	• Service-oriented architectures • Virtualization • Containerization	• SDK with version and variant handling • Hardware-independent applications	• Strong isolation technology within ECUs • In-ECU intrusion detection • Data-access/Device-access authorization
Development practices enablers (cumulative across levels)		• Convergence of internet technology and in-vehicle software		• Automated CI/CT/CD environments • Virtual testing	• Controlled deprecation of software and API versions	• Automated security scanning and vetting process of functions
Business enablers (cumulative across levels)		• Subscription for long-term use of services • Maintenance contracts with data providers	• Subscription for updates • Maintenance contracts with ECU suppliers	• Aftermarket revenue generation for new functions • OEMs directly sourcing or developing software platform	• Subscription for platform upgrades • Decoupling of vehicle and software platform business	• Monetization possibilities for 3rd party function providers

Author: Dr. Moritz Neukirchner