

[Oct., 2025]

ETHERNOVIA

Why TCP/IP and ISO/OSI layering is essential for IVNs

TCP/IP

Ruling computer networks since the 1970's

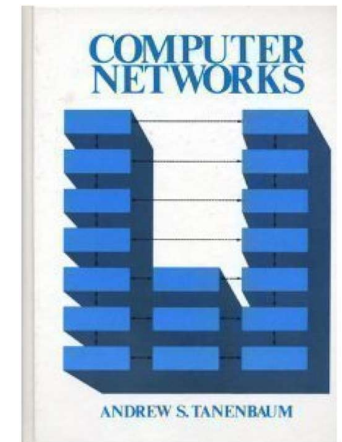


More than 50 Years of History

The **internet protocol suite**, or TCP/IP for short¹⁾, is a framework for organizing the communication protocols used in computer networks according to functional criteria

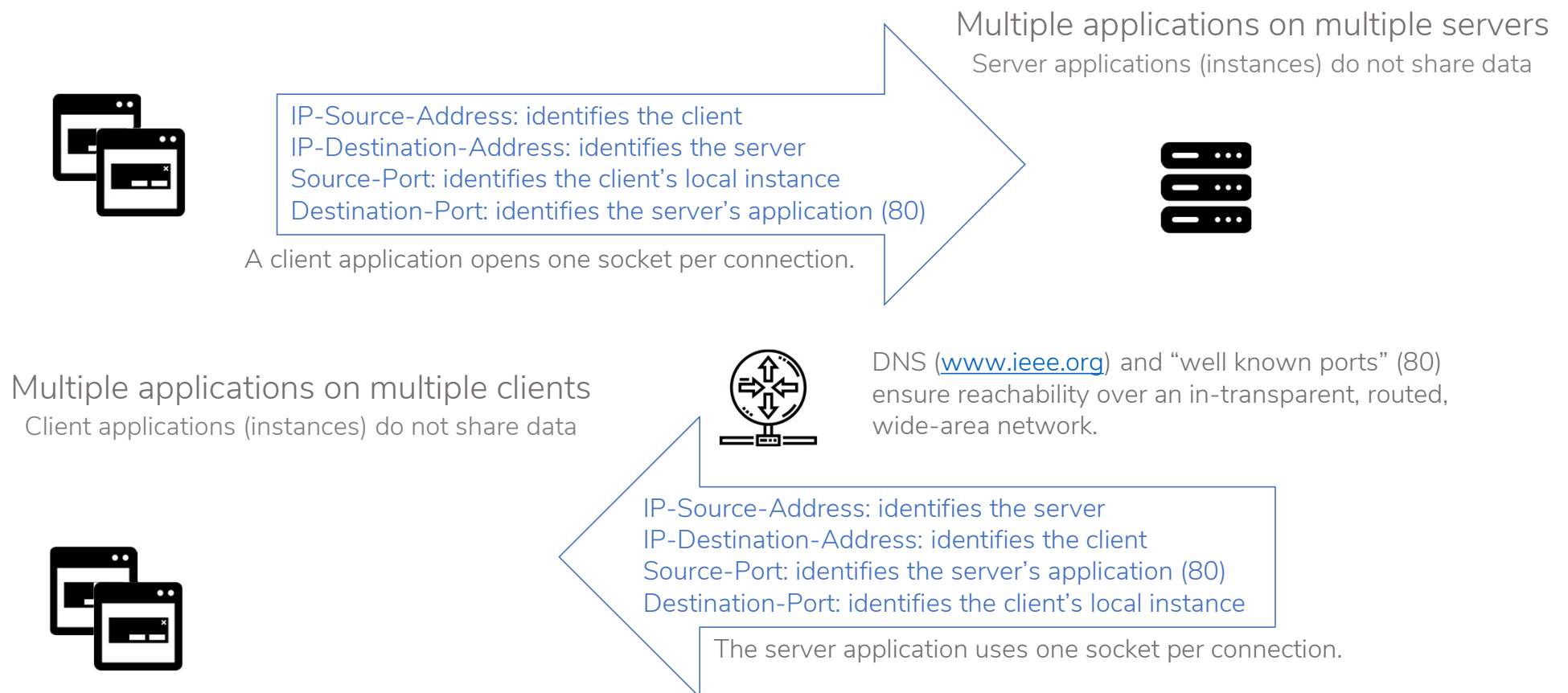
- The first TCP/IP stack was developed by DARPA in the 1970s, for use in ARPANET
- The ISO/OSI (Open Systems Interconnection) reference model was introduced in 1983
- The first IEEE 802.3 standard for Ethernet was also published in 1983
- “Triple-Play” converged Telephone, Cable TV, and Internet access from the early 2000s (VoIP replaced SDH/SONET)
- The (Classic) Autosar communication stack, since 2003, resembles the ISO/OSI model rather closely
- ISO 13400 (Diagnostics over IP) in 2011 introduces TCP/IP to connect the vehicle to the internet

¹⁾ my excuses to Lou Berger at IETF!

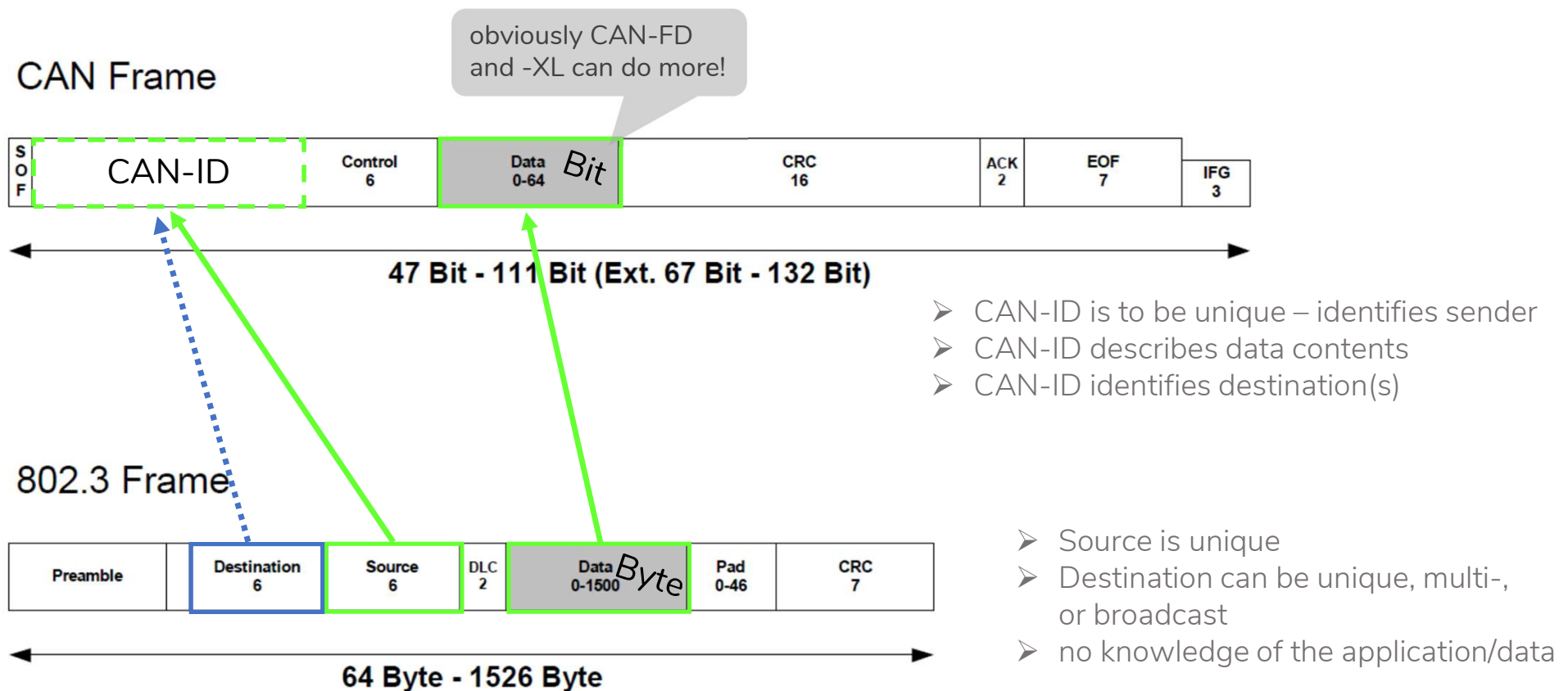


1981

The (over-simplified) Web-Server example



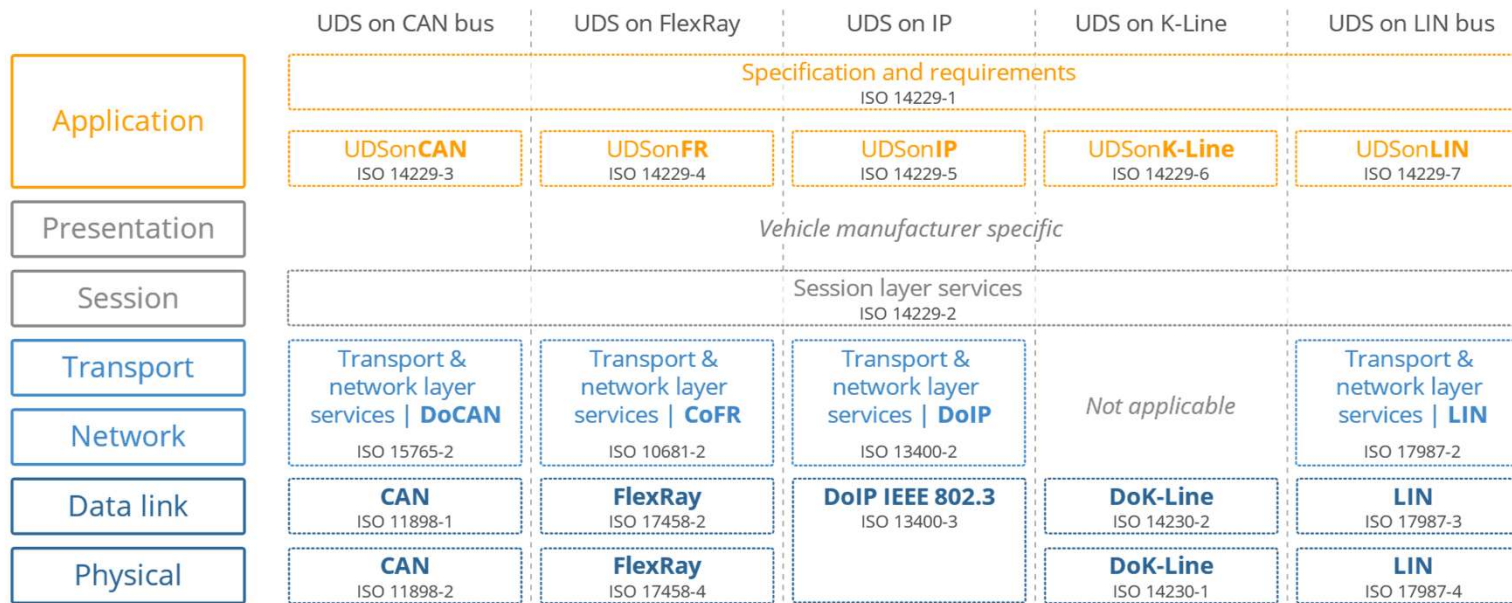
From CAN to TCP/IP: Data centric to Address centric



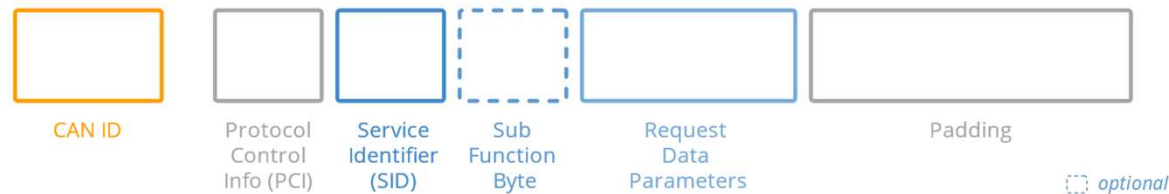
Based on a BMW internal slide from Feb. 2008.

UDS over CAN is based on the ISO/OSI model

7 layer OSI model | **Unified Diagnostic Services (UDS)**



UDS request message structure (UDS on CAN)



<https://www.csselectronics.com/pages/uds-protocol-tutorial-unified-diagnostic-services>

Why should things be different in 2025?

@AEC2025:
P. Ancel (BMW) and
S. James (Acsia)



Acsia

There is a lot we can do with the Ethernet Frame



- Reduced protocol overhead and reduced software impact
- Optimal for confined networks (like in vehicles)
- Direct access to Layer 2 capabilities
- Time synchronization
- Protocol flexibility
- High performance
- Bandwidth reservation
- ...

We have all we
need at Layer 2

AUTOMOTIVE
ETHERNET
CONGRESS

Data Aggregation and Edge Node Control using ASA-ML and RCP for Zonal Architecture

8

<https://events.componeers.net/automotive-ethernet-congress/>

beautification and highlighting added

An abstract graphic in the top right corner consisting of a complex network of blue lines and dots, resembling a map or a data visualization of a network.

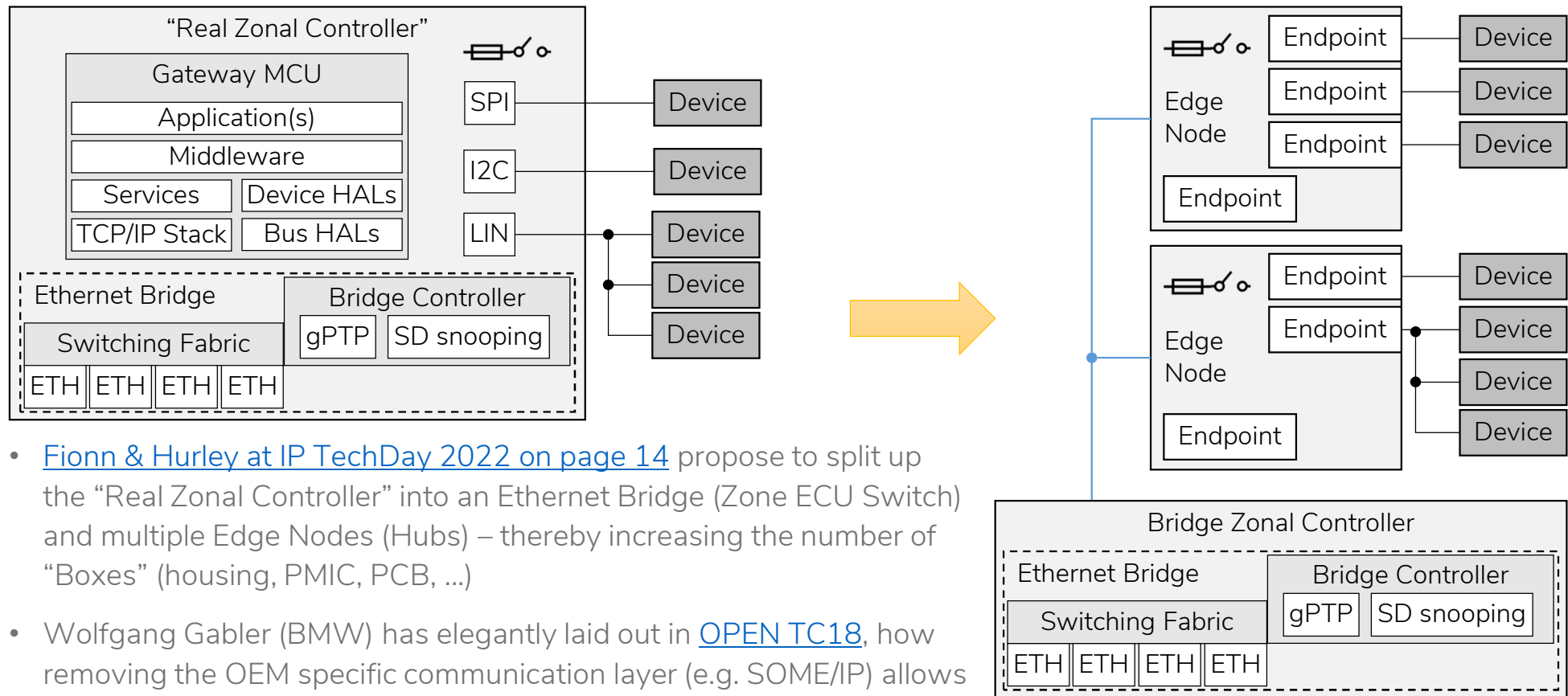
Challenges of Zonal Architecture

Two-way communication with legacy devices from highly integrated high-performance compute platforms

“Real Zonal Controllers” are (too) expensive

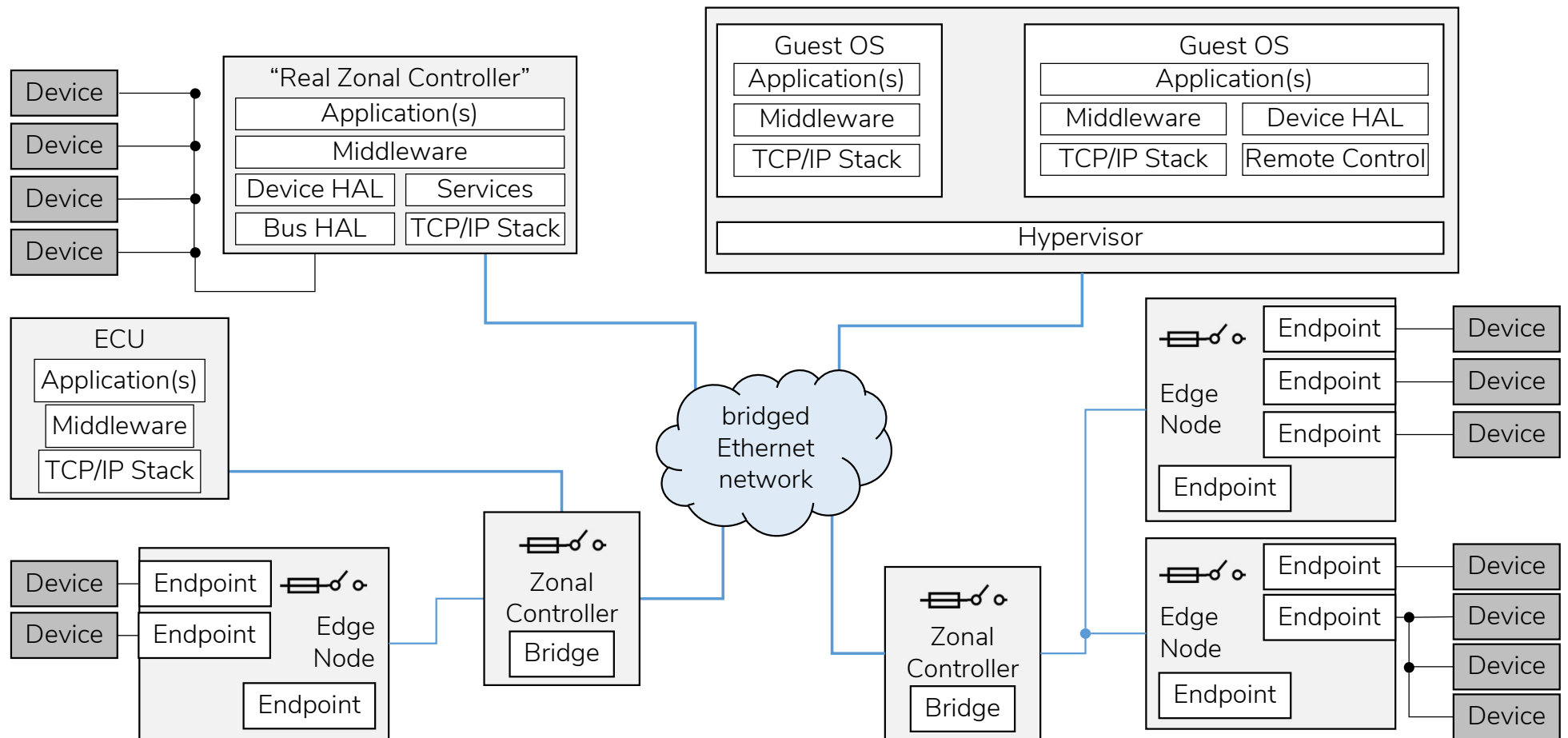
- Def.: “Real Zonal Controller”:
 - “Backbone” (high-speed) Ethernet connection
 - Connects to simple devices via CSI2, LIN, analog, PWM, I2C, PSI5, SPI, ... (potentially CAN, multidrop Ethernet, ...)
 - Abstracts the simple devices towards an HPC application through a Service API
 - Supplies connected devices with power and enables power management through a Service API
 - Manages SW updates to connected devices
- Few (~ 6) “Real Zonal Controllers” per vehicle is costly:
 - Too many connectors: ~ 500 wires each
 - Need to be adapted for (almost) every vehicle option, type, model, and generation - if we uphold the European OEM’s model of options and vehicle lines
 - Software integration for each “Real Zonal Controller”, making them an ECU
 - Energy management is tricky, if door-access, alarm, and online access are centralized

Alternative Zonal-Architecture approach



- [Fionn & Hurley at IP TechDay 2022 on page 14](#) propose to split up the "Real Zonal Controller" into an Ethernet Bridge (Zone ECU Switch) and multiple Edge Nodes (Hubs) – thereby increasing the number of "Boxes" (housing, PMIC, PCB, ...)
- Wolfgang Gabler (BMW) has elegantly laid out in [OPEN TC18](#), how removing the OEM specific communication layer (e.g. SOME/IP) allows to move the application and device specific software to the HPC and standardize the Edge Nodes and the communication with them

A centralized Zonal-Architecture mix



An abstract network diagram consisting of numerous blue dots (nodes) connected by thin blue lines (edges). The nodes are distributed across the upper right portion of the slide, with a higher density of connections on the right side, creating a complex, web-like structure that suggests a network topology.

Choosing a Communication Concept

(Micro-)Services vs. Data centric

What protocols could we (re-)use?

- “SOME/IP” has joined such terms a “TSN” or “TCP/IP” in the sense that everyone you ask will have a different view on what it is or means.
 - SOME/IP is very expressive (ServiceID, MessageID, ClientID, SessionID, versioning, ...) + the IP and UDP headers (SOME/IP over TCP is usually not a good idea)
 - Two-way communication can easily be identified in the messages
 - Basically no standardized IDs or data messages, but with a set of rules for compatibility
 - NOT one protocol, but per application/service messages, which justifies the UDP port proliferation
 - Brings along a Service Discovery
- IEEE Std 1722-xxxx is actually NOT one protocol. It is very(!) many protocols transporting different communication channels.
 - The StreamID (per 1722 frame) is potentially less expressive than a CAN-ID, as no rules are given
 - Some 1722 sub-protocols are literally a collection of user defined fields, defying any interoperability (almost like SOME/IP in a way) – a huge step away from AVB days
 - The semantics of the BusID (per acf message) are basically unspecified
 - Has IEEE Std 1722.1 as a management protocol
- Proprietary solutions are just that – proprietary!

Support of the TCP/IP protocol suite

- Both UDP and TCP are part of the so called TCP/IP protocol suite ([RFC 1180](#))
- Using TCP between ECUs is tricky, even DoIP implementations are struggling – not useful for sensor/actuator communication
- If you can configure a StreamID, you can configure a UDP port number
- Copying a source address or port number from a request into a response is done in HW easily and helps hugely in trace analysis
- Responses to Echo/Ping and ARP can easily be implemented in HW
- UDP checksum calculation can be offloaded in HW
- Sending out a pre-defined SD message can be implemented in HW
- No full TCP/IP stack is required on the Edge Nodes

What would be “strong semantics”?

- Unique Layer 2 MAC addresses enable the well known source-address learning in switches/bridges
 - Service Discovery fills those ARL tables before the first application message is sent
 - Unique addresses allow mixing devices in test and experimental setups
 - Reversing source/destination in request/response establishes a clear two-way communication relation on a network topology level
- IP addresses, even if not used for routing, abstract the Layer 2 MAC addresses
 - Ring-swaps of HW are possible while HW can be identified in traces
 - Service Discovery can communicate a more abstract host, i.e. in a Hypervisor environment
 - Reversing source/destination in request/response establishes a clear two-way communication relation on an abstract host level

“Strong semantics” – continued ...

- UDP ports can identify protocols and/or instances
 - Reversing source/destination in request/response is the common way to differentiate (client) instances
- Where a single protocol is used to communicate between different applications, the concept of a Service can be used to define a Service-Contract and thereby capabilities, message serialization, and “QoS”
- Service-Server and -Client instance differentiation can be done in different ways, but both is needed (SOME/IP’s UDP port proliferation)
- Identification of a Session allows associating responses with requests, even if the order is not preserved
- Separation of Protocol- and Service-Version gives flexibility in implementing forward/backward compatible solutions and evolving both independently

Why is (was) IEEE Std 1722 Layer 2 based?

- While Annex J describes a UDP encapsulation over IP, most of us remember AVB to be Layer 2
- IEEE Std 1722 in AVB was designed as a Direct Memory Access (DMA) capable transport layer
 - A receiving node would have to look at few fields (likely dst. address, Ethertype, and StreamID) to locally determine the memory address (in a ring buffer) where the (audio sample) data received would have to be delivered to, for an application to use it
 - This is very similar to the concept of send/receive in RoCE(v2)
(<https://snia.org/sites/default/files/ESF/Everything-You-Wanted-to-Know-About-RDMA-But-Were-Too-Proud-to-Ask-Final%20v2.pdf>)

An abstract graphic in the top right corner consisting of a complex network of blue lines and dots, resembling a mesh or a data network, extending from the right edge towards the center.

Different approaches to Direct Memory Access

RoCEv2 and IEEE Std 1722

IP and UDP do not prevent DMA

- RoCEv2 uses IP and UDP along with the InfiniBand headers
- Conceptually there is a large similarity with the 1722 ideas
- In comparison to SOME/IP, both solutions lack semantic expressiveness:
 - Single “well known” server port
 - QueueIDs and StreamID have little semantic power, more like PDU-ID or CAN-ID
 - No versioning of the actual data transported (compared with interfaceVersion)
 - Lack of a client and session identifier in case of a multi application environment
- RoCEv2 and IEEE 1722 require an out of band configuration protocol to set up the data transfer
 - or need to rely on fixed configuration and constant (blind) transmissions – very much like CAN in the past



Where does this all leave us?

Conclusion

- Do we need IP routing inside an IVN? – hopefully not!
- UDP and IP add significant semantic information in a multi-to-multi application communication environment distributed over different deployments (bare metal, hypervisor, container)
- Most Service Discovery protocols are based on IP, not on MAC addresses
- In the SDV, building an “efficient” protocol to support low-BW 10BASE-T1S is not the ideal way forward
- Fewer Protocols also simplifies testing
- The challenge of receiving a multitude of small frames with small gaps can be solved by DMA on the HPC, no need for an “efficient” protocol aggregating data
- IP and UDP do not prevent the use of DMA concepts and HW implementations
- Every technical student (engineering, physics, computer science, ...) will know how to open an AF_INET socket (on the HPC) in at least in one programming language
- Removing the IP and UDP headers from in vehicle communication for “efficiency” is like “throwing the baby out with the bath water”



Max Turner

Utrechtseweg 75

NL-3702AA Zeist

The Netherlands

+49 177 863 7804

max.turner@ethernovia.com