technica engineering

LESSONS LEARNED OF 10+ YEARS ETHERNET DEVELOPMENT.

Ways on keeping the complexity under control.

Thomas Königseder
Dr. Lars Völker

2020-09-17

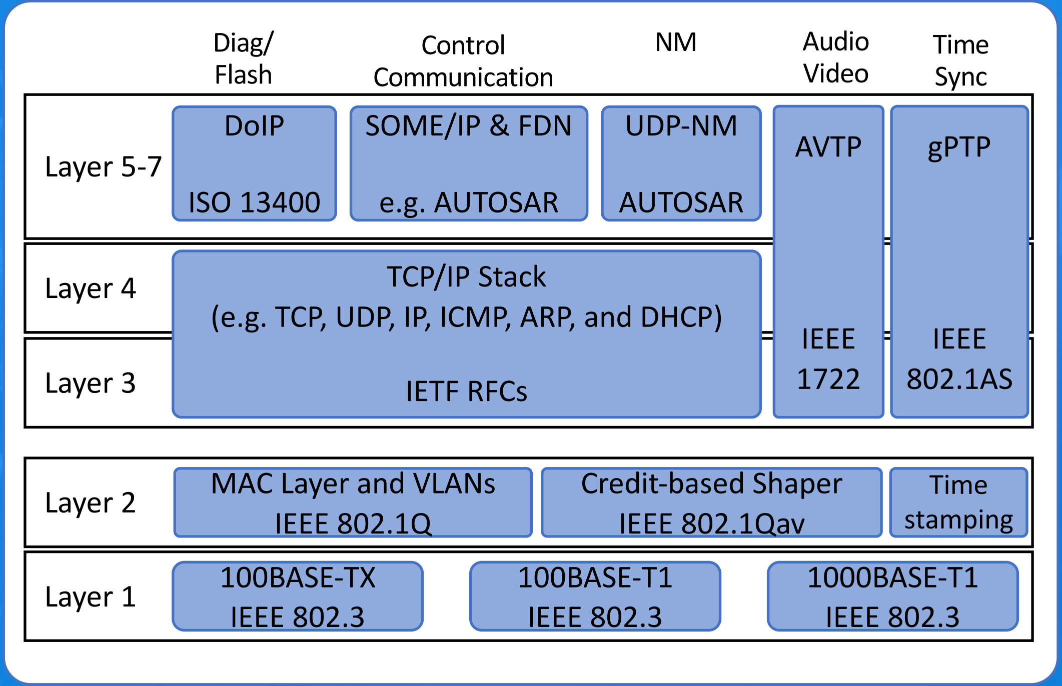
TECHNICA ENGINEERING

LESSONS LEARNED OF 10+ YEARS ETHERNET DEVELOPMENT.

TABLE OF CONTENTS

COMMUNICATION STACK DESIGN

- Does your stack look like this?
- Variations of the stack between OEMs lead to increased complexity and more bugs.
- If the stack vendor and Tier-1 must do something new, this can take more time, cost more money, and lower the quality. Is this worth it?
- If you want a smooth SOP, try to stick to what has already been proven by other OEMs.
- Compete on apps and not on the stack!

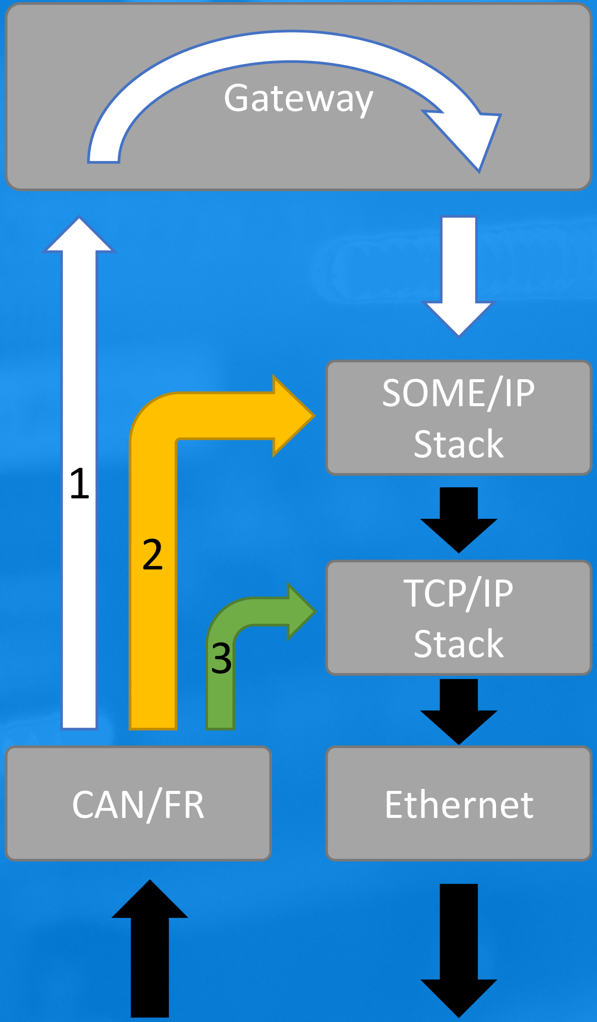


- Lesson learned: Lower the complexity of the stack! Stay with the mainstream, when possible!

GATEWAYS CONNECTING WORLDS

- Different options for connecting classic bus system with Ethernet exist:
  1. ⬜ Create an Application Layer Gateway that translates between both.
  2. 🟨 Transport legacy PDUs over SOME/IP and optionally wrap them into services.
  3. 🟩 Transport legacy PDUs (e.g. CAN) over UDP.

- What approach do you want to take?

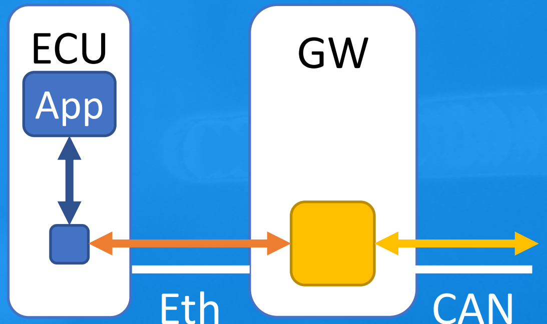


(*) SOME/IP is just an example here. You could use something else as well.

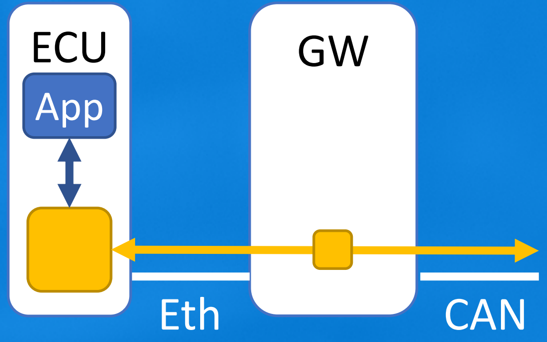
GATEWAYS CONNECTING WORLDS (2)

- It turns out that this is mainly a distribution problem!
  - Gateway translates vs. ECU translates.
  - This limits the scalability of the overall system.
- But: With increasing number of translations, the Gateway becomes the bottleneck and limits scalability!
  - Translating data is not a good fit for embedded gateways.
- With the power of todays ECUs, simplify the Gateway:
  - Transport PDUs over UDP.
  - Leave E2E protected messages as they are.

a) Big Gateway



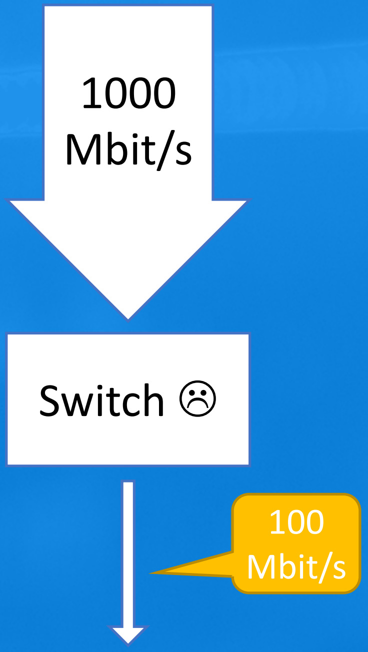
b) Scalable Gateway



- Lesson learned: Be aware of the scalability of your gatewaying concept! Prefer simple concepts!

NETWORK DESIGN: LINK SPEEDS

- Ethernet has the wonderful capability that you can mix and match different link speeds:
  - e.g. 100 Mbit/s and 1000 Mbit/s
- Let's imagine you have an 800 Mbit/s LIDAR stream that hits a small ECU on a 100 Mbit/s link!
  - This might not end well for the ECU and the network.
- What happens if you start mixing 10, 100, 1000, and 10000 Mbit/s?
- Your Automotive Network is in great danger since you Switches could start dropping packets soon.



NETWORK DESIGN: LINK SPEEDS (2)

- When mixing link speeds the following mechanisms may not be optional anymore:
  - Separate your network into smaller pieces using VLANs, Multicast groups, etc.!
    - You do not want a video stream to hit a radar, if something goes wrong.
  - Shape traffic on all senders and switches!
    - Give the switches and receivers room to breath.
  - Police traffic on incoming switch ports!
    - Make sure that nobody disturbs you network.
  - Design your topology right!
    - Don’t daisy chain your ECUs, use less switches by using switches with more ports!

- Open problem: too many SoCs still don’t support traffic shaping in hardware!

- Lesson learned: Mixing link speeds can kill your network faster! Engineer your network!

THE POWER CYCLE

- The Automotive startup and its time is critical.
  - Semiconductors startup times needed to be validated.
  - Startup of the software stack is critical.
  - Timings of the protocols (e.g. SOME/IP-SD) need to be designed.
  - Keep your Security Protocols in mind.

- However, Shutdown and Restart are even more important!
  - An ECU shutting down, needs to make sure that its peers can cleanup state.
  - When canceling a shutdown or restarting, weird timings effects can occur, and your protocol stack might get into trouble.
  - And yes: sleep and suspend to RAM must be handled too!

- Lesson learned: Make sure you have a detailed design and complete requirements for the Startup, Shutdown, and Restart! Test this extensively!

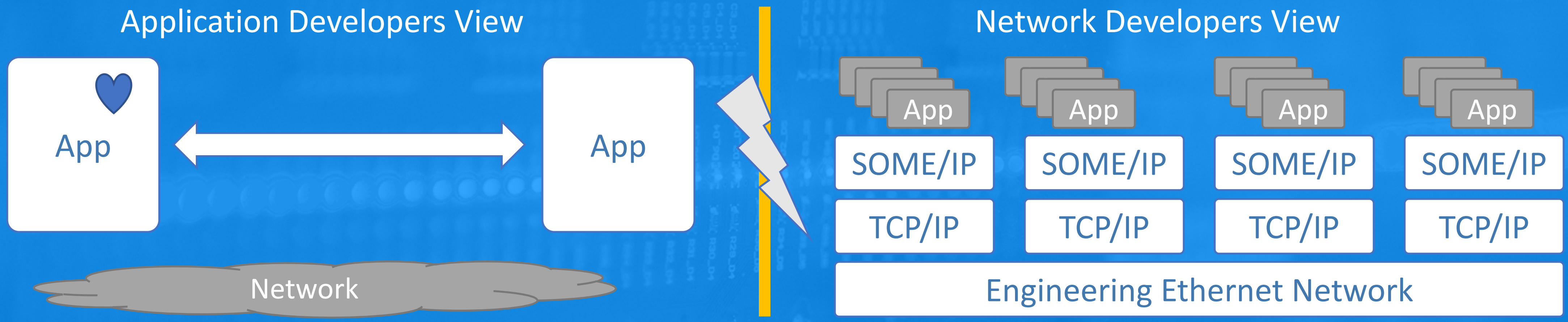
SAFETY, REDUNDANCY, AND ETHERNET.

- In the Automotive world assuming the communication systems as a “grey channel” and using the End-to-End Communication Protection.
  - You can detect failure of the communication system and react to it. Fail Safe.

- For Autonomous Driving you might need a bit more than Fail Safe.
  - The goal is ”system level redundancy” and Fail Operational.
  - Redundancy concepts for different communication system layers in discussion.
  - However, redundancy on Ethernet does not help much, when your sensor gets hit by a stone. And if you add a sensor, you get another Ethernet link anyway.
  - “If the only tool you have is a hammer, you treat everything as if it were a nail…”
  - Many networking guys seem to only have a hammer.

- Lesson learned: Don’t try to solve redundancy locally! Think globally!

APPLICATION VS NETWORK DEVELOPER



- “No limitations” vs. “stable and predictable network”.
- Description of cyclic and/or Audio-Video traffic is rather easy. Events-based messages might get difficult.
- A clear interface between both worlds allows both to focus their work.
  - Using VLANs and traffic classes allows to construct different “virtual pipes”.

- Lesson learned: Divide the realm of the applications and the network! Cleverly constructed traffic aggregates/classes can help!

TESTING: NO TIME TO WASTE

- Do you have time to waste in your development cycle?
- With the increasing complexity of the system and the faster integration cycles, you want to be more agile!
- Don’t let the test process slow you down!
- Wasting time means:
  - Completely testing ECU Software the first time at the OEM.
  - Integrating your Ethernet cluster for the first time in a vehicle.

- Lesson learned: Frontload testing to the supplier! Control the testing! Don’t mix implementer and tester!
- Lesson learned: Frontload integration of your Ethernet cluster! Create a setup to focus on Ethernet-specific integration before your ECUs go into the vehicle!

RECORDING COMMUNICATION

- Every mile driven is important, so make sure that everything is recorded in adequate depth and quality. But how to record the data?
- Make sure that you can log Ethernet, CAN-FD, CAN, and others:
  - Record meta information too: link, direction, and precise and coordinated timestamp
- Make sure you don’t loose data after on startup or in stressful situations!
- State-of-the-art solutions offer:
  - Logger implements high-precision time sync to sync all external capture modules.
  - Capture Modules collect data and meta data for the logger to record.
  - Capture Modules or Loggers buffer the startup.

- Lesson learned: Specify your vehicle logging setup based on state-of-the-art!

OPEN LOGGING FORMATS

- How to store the data recorded in the vehicle?

- ASAM MDF?
  - Current standard has large gaps for bus loggings. In checker tool too.
  - Interoperable implementations almost impossible.

- Proprietary formats in “ascii text” or “binary”?
  - When the specification is not open, you get a single vendor lock…

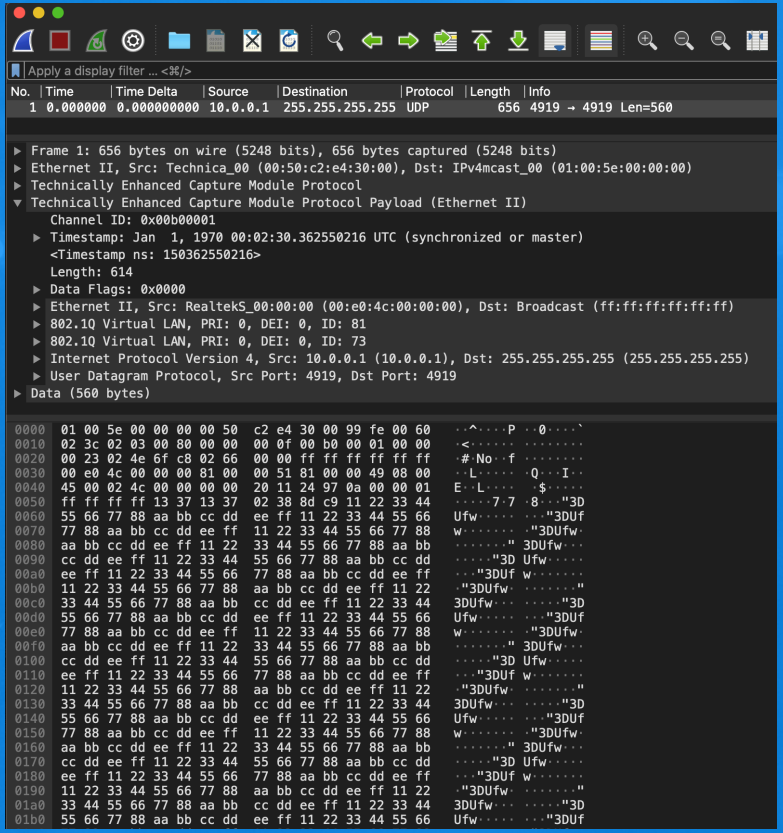
- pcap/pcapng are open specifications and can carry Ethernet, CAN, FlexRay, and more.

- Encountering bugs in closed-source converters is really bad.

- Lesson Learned: Support well specified open formats (e.g. pcap and pcapng)!

THE APPROPRIATE ANALYSIS TOOLCHAIN

- When analyzing problems in the stack, you get lost without the right tools. Not all tools work well for every problem.
- Tools should be available, when first bugs occur.
- If a developer develops a Linux-based ECU, he probably prefers a Linux-based tool for analysis!
- For stack developers: the open source Wireshark is still the reference. We use it for SOME/IP, NM, DLT, DoIP, TECMP, MQTT, IPsec, TLS, MACsec, and others.
- For application developer, other tools exist too.



```
Apply a display filter ... <⌘/>
No. |Time     |Time Delta  |Source   |Destination     |Protocol |Length |Info
  1 0.000000 0.000000000  10.0.0.1  255.255.255.255  UDP          656  4919 → 4919 Len=560

▶ Frame 1: 656 bytes on wire (5248 bits), 656 bytes captured (5248 bits)
▶ Ethernet II, Src: Technica_00 (00:50:c2:e4:30:00), Dst: IPv4mcast_00 (01:00:5e:00:00:00)
▶ Technically Enhanced Capture Module Protocol
▼ Technically Enhanced Capture Module Protocol Payload (Ethernet II)
    Channel ID: 0x00b00001
  ▶ Timestamp: Jan  1, 1970 00:02:30.362550216 UTC (synchronized or master)
    <Timestamp ns: 150362550216>
    Length: 614
  ▶ Data Flags: 0x0000
  ▶ Ethernet II, Src: RealtekS_00:00:00 (00:e0:4c:00:00:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 81
  ▶ 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 73
  ▶ Internet Protocol Version 4, Src: 10.0.0.1 (10.0.0.1), Dst: 255.255.255.255 (255.255.255.255)
  ▶ User Datagram Protocol, Src Port: 4919, Dst Port: 4919
▶ Data (560 bytes)

0000  01 00 5e 00 00 00 00 50  c2 e4 30 00 99 fe 00 60   ··^····P ··0····`
0010  02 3c 02 03 00 80 00 00  00 0f 00 b0 00 01 00 00   ·<······ ········
0020  00 23 02 4e 6f c8 02 66  00 00 ff ff ff ff ff ff   ·#·No··f ········
0030  00 e0 4c 00 00 00 81 00  00 51 81 00 00 49 08 00   ··L····· ·Q···I··
0040  45 00 02 4c 00 00 00 00  20 11 24 97 0a 00 00 01   E··L····  ·$·····
0050  ff ff ff ff 13 37 13 37  02 38 8d c9 11 22 33 44   ·····7·7 ·8···"3D
0060  55 66 77 88 aa bb cc dd  ee ff 11 22 33 44 55 66   Ufw····· ···"3DUf
0070  77 88 aa bb cc dd ee ff  11 22 33 44 55 66 77 88   w······· ·"3DUfw·
0080  aa bb cc dd ee ff 11 22  33 44 55 66 77 88 aa bb   ·······" 3DUfw···
0090  cc dd ee ff 11 22 33 44  55 66 77 88 aa bb cc dd   ·····"3D Ufw·····
00a0  ee ff 11 22 33 44 55 66  77 88 aa bb cc dd ee ff   ···"3DUf w·······
00b0  11 22 33 44 55 66 77 88  aa bb cc dd ee ff 11 22   ·"3DUfw· ·······"
00c0  33 44 55 66 77 88 aa bb  cc dd ee ff 11 22 33 44   3DUfw··· ·····"3D
00d0  55 66 77 88 aa bb cc dd  ee ff 11 22 33 44 55 66   Ufw····· ···"3DUf
00e0  77 88 aa bb cc dd ee ff  11 22 33 44 55 66 77 88   w······· ·"3DUfw·
00f0  aa bb cc dd ee ff 11 22  33 44 55 66 77 88 aa bb   ·······" 3DUfw···
0100  cc dd ee ff 11 22 33 44  55 66 77 88 aa bb cc dd   ·····"3D Ufw·····
0110  ee ff 11 22 33 44 55 66  77 88 aa bb cc dd ee ff   ···"3DUf w·······
0120  11 22 33 44 55 66 77 88  aa bb cc dd ee ff 11 22   ·"3DUfw· ·······"
0130  33 44 55 66 77 88 aa bb  cc dd ee ff 11 22 33 44   3DUfw··· ·····"3D
0140  55 66 77 88 aa bb cc dd  ee ff 11 22 33 44 55 66   Ufw····· ···"3DUf
0150  77 88 aa bb cc dd ee ff  11 22 33 44 55 66 77 88   w······· ·"3DUfw·
0160  aa bb cc dd ee ff 11 22  33 44 55 66 77 88 aa bb   ·······" 3DUfw···
0170  cc dd ee ff 11 22 33 44  55 66 77 88 aa bb cc dd   ·····"3D Ufw·····
0180  ee ff 11 22 33 44 55 66  77 88 aa bb cc dd ee ff   ···"3DUf w·······
0190  11 22 33 44 55 66 77 88  aa bb cc dd ee ff 11 22   ·"3DUfw· ·······"
01a0  33 44 55 66 77 88 aa bb  cc dd ee ff 11 22 33 44   3DUfw··· ·····"3D
01b0  55 66 77 88 aa bb cc dd  ee ff 11 22 33 44 55 66   Ufw····· ···"3DUf
```

Wireshark on MacOS with Automotive protocols.

- Lesson learned: Never trust a single tool! Support different operating systems!
- Lesson learned: Protocol support and configs ready, when the project starts!

SUMMARY

- There are multiple pitfalls on the way to your SOP.

- If you minimize complexity and plan ahead, you are on the right track:
  1. Design your protocol stack wisely and reduce risk!
  2. Choose your gatewaying strategy with scalability in mind!
  3. Make sure that your layer 2 is robust by turning shaping, policing, etc. on!
  4. Don't underestimate Startup, Shutdown, and Restart!
  5. Don't solve redundancy locally! Think globally!
  6. Find the right split between Application and Network Designers!
  7. Frontload your testing to minimize time lost!
  8. Have the right logging and recording setup!
  9. Don't lock yourself in with incompatible formats!
  10. Get your open analysis toolchain ready early!

- After all: enjoy your relaxed Automotive Ethernet SOPs.

STAY CONNECTED

Thomas Königseder
CEO & CTO
Thomas.Koenigseder@technica-engineering.de
+49 (0) 176 30735403

Dr. Lars Völker
Technical Fellow
Lars.Voelker@technica-engineering.de
+49 (0) 175 1140982

Technica Engineering GmbH / Leopoldstraße 236 / 80807 Munich / Germany