LESSONS LEARNED OF 10+ YEARS ETHERNET DEVELOPMENT.
Ways on keeping the complexity under control.
TABLE OF CONTENTS

• Communication Stack Design.
• Gateways connecting Worlds.
• Network Design: Link Speeds.
• The Power Cycle.
• Safety, Redundancy, and Ethernet.
• Application vs. Network Developers.
• Testing: No Time to waste.
• Recording Communication.
• Open Logging Formats.
• The appropriate Analysis Toolchain.
• Summary.

TECHNICA ENGINEERING

LESSONS LEARNED OF 10+ YEARS ETHERNET DEVELOPMENT.
• 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.
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 today’s ECUs, simplify the Gateway:
- Transport PDUs over UDP.
- Leave E2E protected messages as they are.

Lesson learned: Be aware of the scalability of your gateways concept! Prefer simple concepts!
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.
Lesson learned of 10+ years of Automotive Ethernet Development. Ways on keeping the complexity under control.

• 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!
“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!
• 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!
• 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.

• 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.
Lesson learned of 10+ years of Automotive Ethernet Development. Ways on keeping the complexity under control.