eSync Architecture and Programming Model for OTA and Diagnostics
Reaching Non-Ethernet Devices Over an Ethernet Backbone

Presented by: Shrikant Acharya
Chief Technology Officer, Excelfore Corp.

Contributing Authors: Anoop Balakrishnan, Excelfore Corp.
Rema Balaraman, Excelfore Corp.
Agenda

1. Considerations – Objectives and Constraints
2. Architecture Review
3. Protocols, System Requirements, Security
4. Use Case Examples
eSync System Design Objectives

• Reach
  • From Cloud to End Device – Across Various Automotive Sub-Networks

• Bi-Directional
  • Pipeline for Data Push and Data Pull
  • Push Over-the-Air (OTA) Updates to the Vehicle
  • Pull Diagnostic and Telematics Data from the Vehicle

• Highly Secure
  • Vehicles can not be “Spoofed” or Compromised with Spurious Updates
  • Cloud Server can not be “Spoofed” with Spurious Vehicle Data

• Scalable
  • Scales to Many Devices in One Vehicle
  • Scales to Many Different Vehicle Configurations
  • Scales to Millions of Vehicles

2017 ©Excelfore
Important Design Constraints

• Downtime
  • Full Vehicle Update Cycle Must Minimize Vehicle Downtime

• Resilience
  • Must be Resilient Against Errors / Interruptions in Over-the-Air Transmissions

• Efficient
  • Must Be Flexible for Different Processing and Memory Resources in Legacy ECUs

• Safe
  • Functional Safety Considerations, as Defined in ISO 26262 (ASIL levels)
Important Considerations on Safety and Robustness

• ISO26262 Requirements:
  1. Non-Critical: The OTA Update System Does Not Reach Critical Elements at All
     - or -
  2. All Critical: The OTA Update System, and the Entire In-Vehicle Network, Operate Entirely as a Critical System
     - or -
  3. Isolate Critical: The In-Vehicle Network and the OTA Update System Isolate Critical and Non-Critical Elements of the Separate ASIL domains
     • Requires Parallel, Separate OTA Paths

• Robustness
  • Design for Modular Component Integration
  • Keep Up with Current Techniques by Using Latest Standards on Security and Network Protocols
Agenda

1. Considerations – Objectives and Constraints

2. Architecture Review

3. Protocols, System Requirements, Security

4. Use Case Examples
Representative Approach to Next-Gen Vehicle Network

Vehicle Gateway

- Ethernet or OBD Diagnostic Port
- High-Speed Ethernet
- High-Speed Ethernet
- High-Speed Ethernet

ASIL D
- Powertrain Controller/Gateway
  - CAN
  - Ethernet TSN
  - Ethernet

ASIL B
- IVI Head Unit/Gateway
  - CAN
  - Ethernet AVB

ASIL D
- ADAS Controller/Gateways
  - Ethernet TSN
  - LVDS

ASIL B
- Body Controller/Gateways
  - LIN
  - CAN

Powertrain

Infotainment

ADAS

Body/Chassis

Cloud Server
The eSync System Architecture

Security Check Point

Encryption & Authentication

Cloud

OTA Server

Vehicle

DM Client
HMI Service
Status Agent

Message Server

Update Agent
Update Agent
Update Agent
Update Agent

2017 ©Excelfore
Update Agent

- eSync Client
- Message Server
- eSync Message Server Interface
- Decryption
- Delta Reconstruction
- Signature Verification
- Rollback Management
- Error Monitoring
- UDS Client
- ECU
- UDS Server
- Sessions Layer + App Layer

2017 ©Excelfore
Representative Approach to Next-Gen Vehicle Network
(Single Client, Multiple Agents)
Operational Modes of OTA

Data Push:
- Cloud to DM Client
- DM Client to Message Server
- Message Server to Update Agent
- Status Agent to Update Agent
- Update Agent Re-flash of ECU
- Run Diagnostic Scripts

Data Pull:
- ECU Update Agent to Message Server
- Status Agent to DM Client
- DM Client to Cloud
- DM Client to Cloud
Agenda

1. Considerations – Objectives and Constraints
2. Architecture Review
3. Protocols, System Requirements, Security
4. Use Case Examples
## UDS Server Command Sequences

<table>
<thead>
<tr>
<th>UDS Sessions Layer</th>
<th>UDS Application Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Set the UDS server into program mode</td>
<td>1. Transfer Data</td>
</tr>
<tr>
<td>2. Reset to new mode</td>
<td>2. Read Data ID (even reading DTC codes)</td>
</tr>
<tr>
<td>3. Request Seed*</td>
<td>3. Write Data ID</td>
</tr>
<tr>
<td>4. Send Key*</td>
<td>4. Upload Data</td>
</tr>
<tr>
<td>5. Transfer Data† (multiple data transfers)</td>
<td>5. Erase</td>
</tr>
<tr>
<td>6. Erase Memory†</td>
<td>6. Verify</td>
</tr>
<tr>
<td>7. Verify Memory</td>
<td></td>
</tr>
<tr>
<td>8. Set to Normal Mode</td>
<td></td>
</tr>
<tr>
<td>9. Reset to Normal Mode</td>
<td></td>
</tr>
<tr>
<td>10. End of Procedure</td>
<td></td>
</tr>
</tbody>
</table>

* May not be available on all ECUs
† Sequence may differ between UDS servers
Ethernet Based ECUs

• Newer ECUs May have Ethernet Interface

• Security Protocols can be Embedded into ECUs

• End to End Authentication can Go to the ECUs Directly
  • Payload can Remain Encrypted

• Simplifies the Security Architecture and Layout of Devices
  • Clear Segmentation of Functional Domains (using Ports and VLAN)

• No Change to UDS Client / UDS Server Handshake
  • Same as CAN-based ECU Transactions
Security Considerations

• DM Client Acts as Gate Keeper for Authentication
  • Preferred Location: In TCU
  • Can be in Gateway Switch – all External Connection are Authenticated
  • DM Client in a HeadUnit (Infotainment Gateway) Presents a Security Risk

• For ECUs located on FlexRay, CAN, LIN – Update Agents Can Reside in Gateways
  • Each ECU Authenticates with its Update Agent

• Newer ECUs on IP Networks can Host Update Agent within their Code Space
  • Isolate Legacy ECUs from Direct Connection to OBD Port
  • Use ECU Arbitration to Authenticate Legacy ECU Connections

• DM Client and each ECU have their own Unique Digital Certificates
  • Establish Bi-Directional Authentication
  • Difficult for Attackers to ‘Spoof’ or Impersonate Any Element, Difficult to Gain Access to the System
    • Removes “man in the middle” Attacks
    • Impact on Cost and Performance
System Resource Requirements for eSync Client

- Operating System with Secure Non-Volatile File System
- Enough File System Memory for the Largest Expected Combination of Software Update Images, Plus Approximately 10%
- Enough Non-Volatile File System Memory to Buffer Diagnostic and Telematics Data
  - To Prevent Loss of Data when Connection is Interrupted
- Less than 500KB for eSync Client Code
- Typical: about 500KB for RAM
  - Additional RAM May be Needed for Many Update/Diagnostic Agents in the System
Agenda

1. Considerations – Objectives and Constraints

2. Architecture Review

3. Protocols, System Requirements, Security

4. Use Case Examples
Demonstrated Use Case Environments

eSync Client:

- Operating Systems: Linux, QNX, Integrity and Android
  - Other OS and File Systems are Possible
- Processors: Intel *Apollo Lake*; NXP *i.MX6*; Qualcomm *Snapdragon 820*; Renesas *R-Car3*

eSync Agent:
All OSs and Processors Used for the eSync Client, Plus:

- Operating Systems: AUTOSAR, Erika, FreeRTOS
- Processors and Controllers: NXP MPC5777 / 5648; Cortex R4 / Cortex M
- Bus / Networks: Ethernet (Broad-R Reach, AVB/TSN), CAN, LIN, FlexRay, USB
Use Case 1: Basic Vehicle System

Telematic Unit

eSync Client

Gateway/Switch

Agent

Infotainment Headunit

Agent

Camera

ECU

Agent

Camera

1000base-T1 Ethernet
100base-T1 Ethernet
CANbus
Use Case 2: Vehicle Interconnect Using Ethernet for New Vehicle Platforms
Use Case 3: Multi-Domain eSync OTA System With Secure Gateway for Critical Domain
Summary of eSync System

• Bi-Directional and Transaction Based Information Transfer
• Modular Design with Update Agents for All Electronic Devices (ECUs, Sensors, etc.)
  • In the Device for IP Addressable Ethernet Devices
  • In the IP Addressable Port of the Gateway Switch for CAN, LIN Devices
  • Ensures System Reaches All Electronic Devices
• Layered Authentication and Encryption Between All Modules
  • Robust Security against Hackers
• Any Number of Update Agents, Update Any Number of ECUs in Parallel
  • Minimizes Vehicle Downtime during Updates
• Modular Design for Optimal Use of Limited CPU and Memory Resources