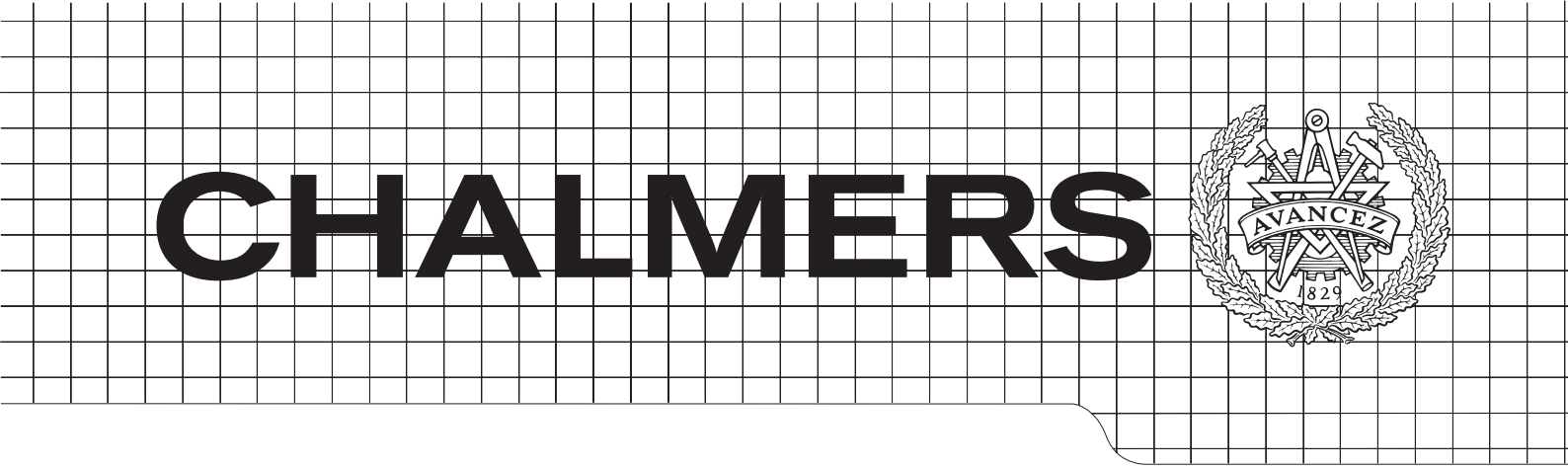

Remotely access real-time system in a modern car

Master of Science Thesis in Systems, Control and Mechatronics

JONAS HELLBERG
MIKAEL PETTERSSON

Department of Signals and Systems
Division of Automatic control, Automation and Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2013
Master's Thesis 2013:27

Remotely access real-time system in a modern car

Master of Science Thesis in Systems, Control and Mechatronics

In corporation with

Diadrom Systems AB

Jonas Hellberg
Mikael Pettersson
Gothenburg, Sweden 2013

Supervisor:
Henrik Fagrell
PhD Informatics
Diadrom Holding AB

Examiner:
Bengt Lennartsson
Professor
Department of Signals and Systems
Chalmers University of Technology

August 13, 2013

Abstract

A modern car requires regular service and maintenance to retain a long operational lifetime. Advanced diagnostic applications, which often are expensive, brand specific and require high electronic skills, are used for tracing and solving errors. If all independent repair facilities need their own experts for each brand, the services will be significantly more expensive for the car owners. This cost could be decreased by running advanced diagnostics remotely, e.g. car and expert are located at different places. The different places of the two introduces latency into the communication. The arising problem is the time critical functions, the login diagnostic service and the recurrent keep-alive message, which need to be run to gain full access to the Electrical Control Units (ECU). Connecting an intelligent Pass-Thru device between the car and the advanced diagnostic application could enable the possibility to run time critical functions on the real-time system of a car when high latency is present.

This thesis results in the Artificial Remote Networker (ARN) concept, which is intended to be implemented in an ARN Pass-Thru device, developed to use the SAE J2534 standard. The developed ARN concept includes three phases; unlocking-, keep-alive- and compatibility phase. The unlocking phase manages the stream of messages in the login diagnostic service to unlock the ECU and the keep-alive phase keeps it unlocked. The compatibility phase indicates whether the ECUs are compatible with the ARN concept or not, and can be used by the advanced diagnostic application to examine the possibility to run time critical functions with high latency present before the expert tries to unlock an ECU.

The ARN concept was implemented as a proof of concept on a subnetwork of a real-time system from a car running General Motors in vehicle Local Area Network (GMLAN) over Controller Area Network (CAN). Tests for evaluation demonstrates that the ARN concept manages time critical functions when high latency is present for ECUs with static seeds, or static long enough. The thesis also results in a proposal of an extension of the SAE J2534 standard to cover ECUs with dynamic seed as well. The way of implementation has some drawbacks. The first is that the advanced diagnostic applications have their own internal timers to decide when they should stop expecting a response message due to e.g. timeouts, which cause problems when trying to read messages as high latency is present. The second drawback is that the message buffer on the vehicle side is filled up quickly and the advanced diagnostic application does not read the messages fast enough due to the latency. Both this observations results in suggestions of how to solve them in future work.

Acknowledgements

We would like to express our appreciation to Diadrom for providing us the opportunity to perform our Master's thesis at their company in an inspiring environment. A special thanks to our supervisor Henrik Fagrell, Oscar Lund and Lars Persson for their help and support throughout the process. We would also like to thank all persons concerned at Diadrom for participating in interviews and workshops. Your help has been of great importance to us.

Jonas Hellberg & Mikael Pettersson, Gothenburg, 30/5/13

Abbreviations

API	Application Program Interface
ARN	Artificial Remote Networker
CAN	Controller Area Network
CARB	California Air Resources Board
CDMA	Code Division Multiple Access
DLL	Dynamic-Link Library
DoIP	Diagnostics over IP
D-PDU	Diagnostic Protocol Data Unit
DTC	Diagnostic Trouble Codes
ECC	Electronic Climate Control
ECU	Electrical Control Unit
EPA	Environmental Protection Agency
GMLAN	General Motors in vehicle Local Area Network
GPS	Global Positioning System
GUI	Graphical User Interface
ISO	International Organization for Standardization
LAN	Local Area Network
MVCI	Modular Vehicle Communication Interface
OEM	Original Equipment Manufacturers
OSI	Open Systems Interconnection
RTT	Round Trip Time
SAE	Society of Automotive Engineers
WLAN	Wireless Local Area Network

Contents

1

Introduction

Modern cars are evolving towards higher complexity fast. Thirty years back in time the car was a pure mechanical product, but around that time the first software was introduced into the car [1]. This computer-based solution controlled the ignition and was the start of a remarkable evolution. When the cars evolved into a software based product instead of a mechanical, the cost of correcting errors decreased considerably. To correct an error in older cars a new hardware product had to be developed, produced, kept in stock and distributed around the world. When the errors lies in the source code of the software all the physical handling of the update can be eliminated. All that has to be done is an update of the software at a repair facility. This also resulted in a more open mind to update the vehicles by the manufacturers, causing the rate of how often new functionality were implemented in a car to increase [2]. Today a modern car includes more than 75 independent computers, so-called Electrical Control Units (ECU), which control its features, e.g. engine, gearbox and brakes. The increasing number of ECUs in cars is shown in Figure 1.1. The software code which runs on the ECUs in a modern car consists of tens of millions of lines and over 80 % of the innovations in automotives are software based [1]. In many cases an ECU used in one model of a car could be included in another car of a different model and by only changing the software the functionality can be of a different kind. The possibility to use standardized hardware in different contexts is appealing to manufacturers since purchasing big quantities of a product reduces the cost per unit [2].

As the complexity of the software in cars grew, a way for optimizing the electrical wiring was necessitated. The solution to be adopted was bus systems, e.g. the Controller Area Network (CAN) protocol, which today is the most common in the automotive industry [1]. This eased the possibility to extract diagnostic data and change parameters in vehicles. It also made it possible for the ECUs of the vehicles to share sensors. For example a sensor for measuring the outside temperature could be needed for both the engine and the Electronic Climate Control (ECC). Instead of constructing a vehicle which

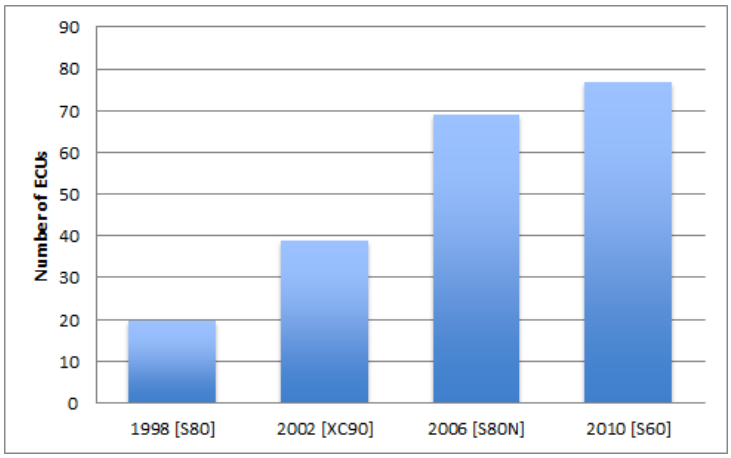

Figure 1.1: The number of ECUs in different models of Volvo cars over the years 1998-2010 [3].

has two sensors which do the same thing, a single sensor could be connected to the bus system where the units could share the functionality by invoking it. Introducing a bus system also enables the ECUs to share information between each other. This allowed new functionality to be implemented by only changing the software, e.g. the instrument panel could warn the driver if the engine was shut-down, the key was removed and the door had been opened *but* the parking brake was not set [2].

To provide a long operational lifetime for a car, regular service and maintenance are required. It is of importance to distinguish between basic- and advanced diagnostics. Basic diagnostics intend the process of reading error codes and readout configurations from the real-time system of a car. It can be performed with simple diagnostic devices communicating with e.g. a smartphone by Bluetooth [4]. Advanced diagnostics includes the functionality of basic diagnostics, but are extended with the possibility to update software (for newer versions or functionalities), alter parameters and trigger built-in tests of the ECUs. Due to complexity of the on-board real-time systems, advanced diagnostic applications are needed to trace and solve errors. Since these applications often are brand specific, expensive and require high electronic skills to trace a specific error, the procedure becomes difficult for the independent mechanics [5]. If all repair facilities need their own expert for each brand without having a major customer base for them, the repairs would be significantly more expensive for the car owners.

When ECUs were introduced, multiple hardware and software devices were necessary to reprogram the real-time systems of cars by different manufacturers. All Original Equipment Manufacturers (OEM) developed their own interfaces towards the cars, which increased the cost to run an independent repair facility. This resulted in a situation where the end customer could not choose where to repair its car and the OEMs could charge a big fee. As this grew, U.S Environmental Protection Agency (EPA) together with the California Air Resources Board (CARB) developed vehicle network requirements which all manufacturers that sell vehicles in North America had to implement for every

emission-related ECU in models produced beyond 2004, namely the Society of Automotive Engineers (SAE) J2534 standard [6]. The SAE J2534 specifies a standardized system for programming the ECUs of a car. This includes a PC, standard interface to a software device driver and an interface for connection between the PC and a reprogrammable ECU [7]. The European Commission also adopted a constitution stating that all vehicles manufactured after 31st of August 2010 should incorporate the SAE J2534 or the newer International Organization for Standardization (ISO) 22900 [8] for all their control units. ISO 22900 is in a similar way to the SAE J2534 basically independent of the underlying device as long as the physical interface is supported and no tool vendor-specific functions are required. In order to preserve the diagnostic applications based on SAE J2534 the ISO 22900 specifies that a connection device shall contain a compatibility layer or wrapper for converting to SAE J2534 [9]. The principle flow of data in diagnostic systems of today is illustrated in Figure 1.2.

Connecting an intelligent Pass-Thru device could enable experts to access and run advanced diagnostics remotely, and eliminate the need of experts located at the independent repair facility. However, the on-board real-time systems cannot handle too much bus-latency, since it requires rapid communication when login-procedures are conducted to enable execution of advanced diagnostic functions and for the keep-alive function which holds the communication link open during the diagnostic. If the wired connection between the PC and car is replaced with e.g. an Internet connection, the low latency can not be guaranteed any more.

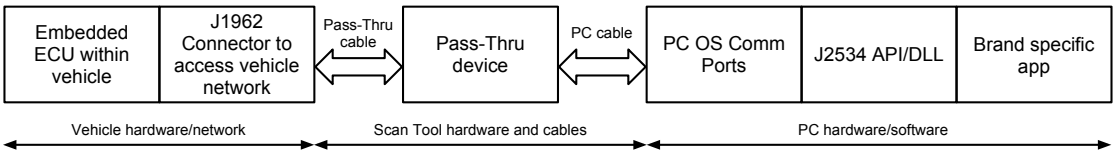

Figure 1.2: Overview of the communication between a car and a diagnostic application [6].

1.1 Purpose

The purpose of this thesis is to extend the existing field of application for Pass-Thru devices to handle communication with advanced diagnostic applications. This require rapid answers when high latency is present, e.g. caused by different locations of the expert and the car without making the solution brand specific. Further, ways of implementing a learning mechanism will be studied and evaluated.

The desired result of this thesis is a concept which allows a user to perform time critical functions in a real-time system of a car when high latency is present. The concept provides the advanced diagnostic application with information about whether remote advanced diagnostic is possible or not. Further, propositions of what future standards should include to make it possible to solve the cases not handled by the developed concept are stated.

1.2 Delimitations

- The aim of the thesis is to make a proof of concept, thus a full scale implementation is not of priority. Hence, only some of the time critical functions are implemented and evaluated.
- The implemented concept is only evaluated for one specific protocol, GMLAN over CAN.
- No data security aspect, such as encryption, is taken into account when designing the system.
- Safety issues raised by running advanced diagnostics remotely is not taken into account.
- The transferring protocol used between the advanced diagnostic application and the Pass-Thru device is not discussed in detail.
- The thesis only focuses on the SAE J2534 standard and not ISO 22900. This since the latter is backward compatible with the first.

1.3 Research question

The main research question is:

- Can time critical functions in advanced diagnostic applications be operational on a car which is located elsewhere geographically, as high latency is present?

To answer the research question the following challenges are addressed:

- There are many different brands on the market.
- There are legislated standards on the market today.
- The solution must be functional with already existing advanced diagnostic applications according to legislated standards.

Summary of chapter

In this chapter an introduction and background to the problem is given. The scenario stated results in an explanation of the purpose of the thesis, which is compressed into a research question formulation. The issues that need to be treated are pinpointed and the delimitations are stated as well. In order to solve the stated problems and to answer the research question the methodology used is specified in Chapter 2.

2

Research methodology

This chapter describes the processes of collecting knowledge about the area of vehicle diagnostics and which methods that have been used. Further, it also explains how they have been applied in the thesis.

2.1 Design science research

The process of stating the problem and to solve it demanded acquisition of the right knowledge, which was done in an iterative process as the *design science research*, illustrated in Figure 2.1, proposed in [10]. This indicates that most knowledge is found during a design effort and the awareness of the problem is increasing as the process proceeds.

The method begins with the *Awareness of problem* phase, where the developer becomes aware of an interesting problem. This results in a proposal of a new research effort.

The *Suggestion* phase of the model follows the proposal. It suggests a tentative design which would solve the problem. This design is built by the use of the current knowledge base. This phase results in a plan for how the problem shall be solved in the current iteration.

In the *Development* phase the tentative design is implemented. It is possible to return to the *Awareness of problem* phase if a circumscription is discovered. This means that the new knowledge can be used when the problem formulation is updated in the *Awareness of problem* phase and the tentative design is proposed.

The *Evaluation* phase is used to evaluate the implemented design according to the criteria stated in the *Awareness of problem* phase. If the results from the evaluation deviates from the expected outcome, it must be carefully noted and explained. It is also possible to return to the *Awareness of problem* phase if a circumscription is encountered here, to redefine the proposal and tentative design.

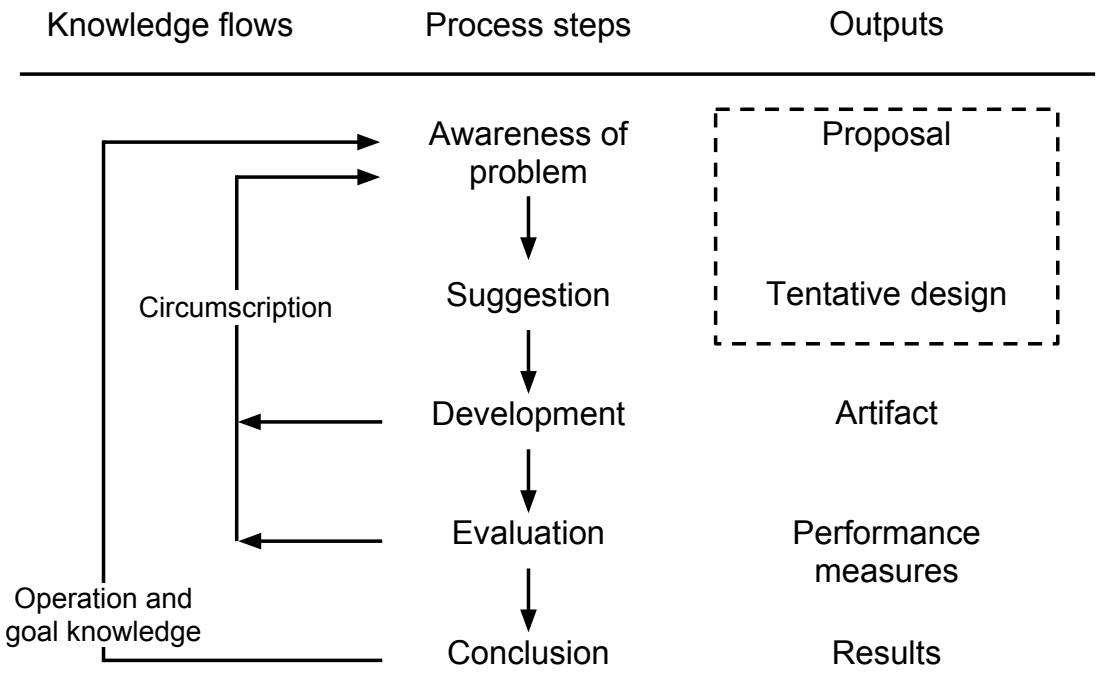

Figure 2.1: The general methodology of design science research [10].

As the *Evaluation* phase is finished the *Conclusion* phase is initiated. It is the last phase of the specific research effort. Often, the result achieved in the earlier phases is *satisfactory*. This means that even though the result still deviates a little from its intended behaviour, it is considered *good enough*. The knowledge collected may lead to proposal of future work which leads to that a new research effort is initiated, but the current effort is ended and the result is analyzed to see whether it is applicable in a general and similar situation or not [10].

2.1.1 Literature studies

To get a basic understanding of the area regarding advanced diagnostics to be used in the *Awareness of problem* phase, the initial phase of the thesis consisted of literature studies concerning standard protocols used in automotive industry and brand specific solutions for running time critical advanced diagnostics. It gave an initial insight of the problems and difficulties in the area.

The published information about how the future market is evolving is rather limited, since the manufacturers are careful not to expose their possible advantages. Hence, the major part of this information consists of the solutions that are already released to the market today.

2.1.2 Interviews and workshops

To obtain a deeper understanding of advanced diagnostics that the literature studies covered, a number of interviews and workshops were made.

Interviews with experts in the field of diagnostics were conducted. The interview contents covered different manufacturers of Pass-Thru devices, vehicles and advanced diagnostic applications and extended the knowledge of the latest technology available on the market today. The possibilities and difficulties according to the current legislated standards were discussed and investigated and the solutions different OEMs use were discussed.

Workshops were arranged in order to supplement the understanding of how advanced diagnostics is performed today and to emphasize the different important parts and problems. These parts were experienced through a hands-on perspective by a walk-through of an advanced diagnostic application which further clarified the relevant facts.

The analyzed information from the literature studies, interviews and workshops stated the parts to consider during the development of the concept, and are explained in Chapter 4.

Summary of chapter

The chapter introduces the design science research methodology, which is used to answer the research question. An explanation of how the knowledge to develop a solution was collected is given. This includes a study of adequate literature and a description of the how interviews and workshops are executed. This investigation results in the technical information presented in Chapter 3.

3

Technical background

In this chapter the technical background within the field of remote vehicle diagnostics is briefly presented. It treats the basic technologies and standards used today and intends to grant the reader enough knowledge to grasp the following chapters. A short insight in related work is also presented.

3.1 Electronic control unit

An ECU is an embedded system in automotives which controls the real-time system of a vehicle, in e.g. engine, brakes and emission control. It uses sensors of different kinds to gather data from the vehicle to determine its behaviour. The data is used by the ECUs to perform necessary actions to achieve a desired behaviour [11].

One of the first advanced ECUs was the engine control unit. It controlled parameters such as fuel injection and ignition timing which resulted in both improved engine-performance and reliability. Since the software in the ECUs have the possibility to compensate for fault tolerances, the high precision needed to produce engine parts was not as important anymore and the manufacturing costs was reduced [2].

The ECUs generate Diagnostic Trouble Codes (DTCs) when the outcome of parameter measurements differ from expected values. The DTCs are stored in the memory of the ECUs and can be read and/or deleted with a diagnostic application [11].

The number of ECUs in modern vehicles are increasing every year and the need for reprogramming the software of the ECUs and calibrate their settings by using external programming applications also increases and is expected to continue to do so in the future [7, 12].

3.2 Open systems interconnection-model

To help developers organize the communication of the ECUs in the real-time system of cars the open Open Systems Interconnection (OSI) model can be used. The basic structuring technique in the model is layering. The data sent over a network must go through layers, called entities, as it is sent or received [13]. The model consists of N subsystems ordered in a vertical sequence, shown in Figure 3.1. The layers are connected through their borders, which means that the layer N is only connected to layer $N-1$ and $N+1$ (except the highest- and lowest layer). The seven main layers included in the OSI-model are:

- application layer
- presentation layer
- session layer
- transport layer
- network layer
- data link layer
- physical layer

where the application layer is the highest and the physical layer is the lowest [14]. Often, actual protocol stacks combines layers, shrinking the OSI-model into fewer layers [15].

3.3 Controller area network

Until the early 1990s the ECUs of a car were connected through cables. This meant that the sensors needed to be directly connected to the ECU using it and that a new sensor had to be connected for every ECU who needed it, for instance, a sensor measuring the outside temperature might be needed by both the ECC and the engine ECU. By connecting many sensors which have the same functions redundancy between them arises and the numbers of cables in the car is increased. To solve this problem, the sensors and ECUs were connected over a network on the same bus. Hence the same sensor could be used by multiple ECUs, making it possible to reduce the amount of sensors and cables [2].

One of these bus systems is called CAN. CAN is a system with multi-master capabilities, which means that several of the ECUs connected can make requests of the bus and transmit data at the same time. The CAN bus system covers the two lowest layers in the OSI-model; the data link- and the physical layer. In a CAN bus system the message a user wants to send is packed in a CAN frame. A CAN frame includes a CAN identifier and user data with a size between 0 and 8 bytes, but the data can be segmented to be able to send longer data using multiple frames. The CAN bus system sends a prioritized

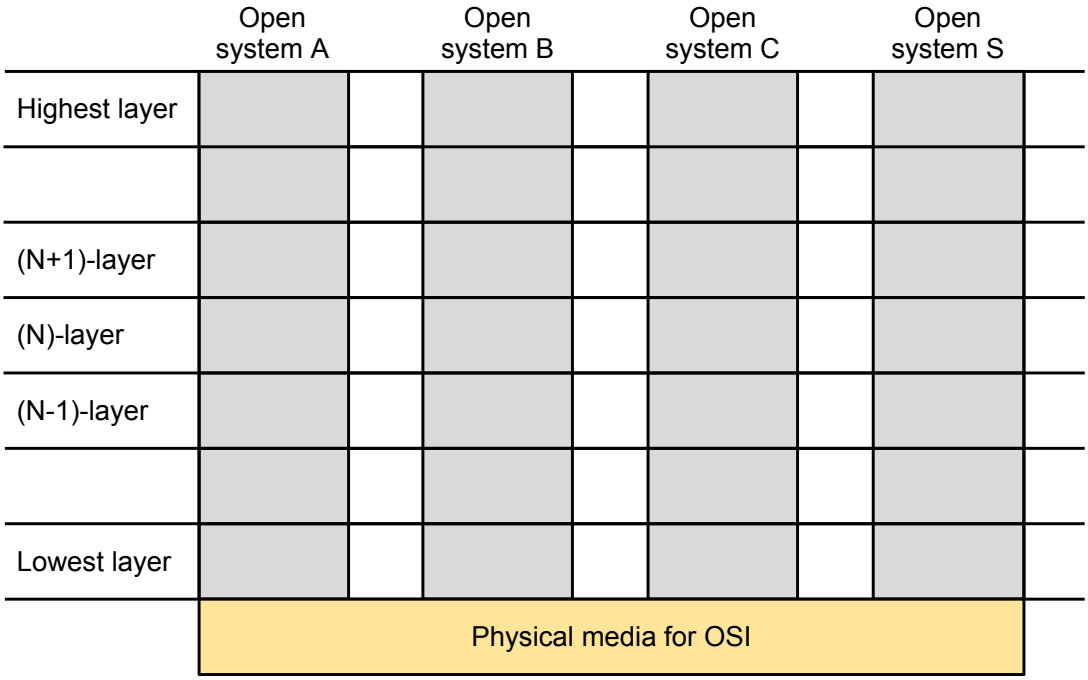

Figure 3.1: Structure of layering in cooperating open systems. The grey columns represents systems and each row represents a layer [16].

message to all ECUs (broadcasting) and the ECUs decide if it is intended for them by reading the CAN identifier. The identifier also includes information about the priority of the CAN message in case of that two messages engage the bus at the same time [17].

3.4 General Motors in vehicle Local Area Network

To standardize how the data between ECUs and diagnostic applications are sent, General Motors in vehicle Local Area Network (GMLAN) is used in several vehicles. GMLAN is a family of serial communication buses, developed by the car manufacturer General Motors, which uses the Controller Area Network (CAN) protocol. It supports three buses; dual wire high speed bus (500k bps), dual wire mid speed bus (approximately 95.2k bps) and single wire low speed bus (33.33k bps) and operates on a transport- and application layer level.

GMLAN defines a number of diagnostic services used to diagnose an ECU. The messages included in the services are packed into CAN frames for transferring. The requested diagnostic service is specified in the service identifier parameter. If the diagnostic service supports multiple levels of functionality, a sub-function parameter is used to specify which level is requested. It shall always occupy the first byte after the service identifier when used. A data parameter is included in the cases where an indication of which data

CAN Id	#1	#2	#3	#4	#5	#6	#7	#8
$241	$02	$27	$01	—	—	—	—	—

Table 3.1: CAN frame of a *requestSeed* service.

is requested from the ECU is needed. It does only affect which data is returned and not the functionality of the diagnostic service e.g. if the diagnostic service is used to provide ECU output statuses, the data parameter can be used to choose which outputs data are returned.

The *SecurityAccess* is used as an example of a GMLAN diagnostic service, see Table 3.1. The table shows how the CAN frame is marshalled. It starts by including the CAN Id for the specific ECU, in this case $241. The first byte of the CAN messages defines how many bytes the message contains. The second byte is the service identifier parameter which is defined as $27 for *SecurityAccess*. Directly after the service identifier parameter, the sub-function parameter is specified, here $01 for the functionality *requestSeed*, which tells the ECU to send a security seed [18].

3.5 SAE J2534

To reduce the emissions of cars, EPA and the SAE developed the SAE J2534 standard which became mandatory for car manufacturers that sell vehicles in North America to implement for all emission-related ECUs in every car manufactured after the year of 2004. This made it easier for independent repair facilities to run diagnostics [6]. The European Commission followed this initiative by adopting a constitution in 2010, which stated that all cars manufactured after the 31st of August 2010 should support either SAE J2534 or ISO 22900 for all their ECUs [8].

Before the SAE J2534 specification was adopted, all hard- and software needed to handle the reprogramming of ECUs were specific to each car manufacturer. This made the diagnostic process complicated and expensive for the aftermarket businesses, which handled cars from a wide range of vehicle manufacturers, since they needed a lot of hard- and software which could cost as much as the car itself [6].

SAE J2534 specifies a hardware device which enables connectivity between a PC and the network of ECUs of a vehicle and a software Application Program Interface (API) which controls the hardware device. An application is run on a PC which contains software steps, proprietary to the manufacturers, for reprogramming, calibrating and monitoring ECUs of a vehicle. The user handles the application through a Graphical User Interface (GUI) which parses the register files of the PC to find the Dynamic-Link Library (DLL) file of the manufacturer. The advanced diagnostic application provides the user access to the downloadable software and calibration files. It maintains and enforces security policies to control user access and provides programming algorithms for controlling the ECU reflashing process for the specific vehicle by calling the API

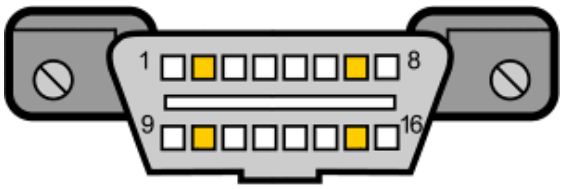

Figure 3.2: The pinout of the SAE J1962 connector [19].

functions. This software must be sold *'for a reasonable price'* [6].

A fully compatible SAE J2534 interface must support all the following communication protocols:

- ISO 9141
- ISO 14230 (KWP2000)
- SAE J1850 41.6 KBPS PWM (Pulse Width Modulation)
- SAE J1850 10.4 KBPS VPW (Variable Pulse Width)
- ISO 11898 (CAN)
- ISO 15765 [1]
- SAE J2610 [6]

3.5.1 Pass-Thru device

The diagnostic application uses references to the SAE J2534 API functions, see Appendix A for a list of all functions, to control the message stream between a Pass-Thru device and ECUs of the vehicle. A Pass-Thru device does not interpret, examine or alter the information that flows through it to the network of the vehicle, which allows the manufacturers to protect their proprietary data. In other words, a Pass-Thru device works as a connection between the application layer and the lower layers of the vehicle. The manufacturer of a Pass-Thru device supplies the hardware appliance and cables needed to connect to the vehicle network. The physical interface to the vehicle is the SAE J1962 connector, see Figure 3.2. As a result, a SAE J2534 compatible Pass-Thru device works with any vehicle network and any user application and the hardware dependencies of the different components are kept to a minimum [6].

An illustration of how the treated protocols are connected is shown in Figure 3.3. The PC calls the SAE J2534 API functions to communicate with the Pass-Thru device. The device sends e.g. GMLAN messages to the CAN bus packaged as CAN frames, which are guided to the right ECU by the CAN Id.

[1] Almost identical to GMLAN [7, 13]

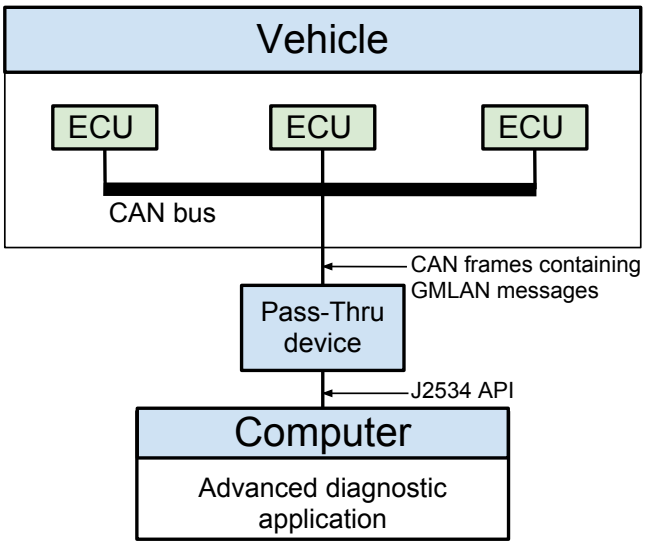

Figure 3.3: An explanatory illustration of how SAE J2534, GMLAN and CAN is connected.

3.5.2 Establishment of connection

To clarify how the functions in SAE J2534 is used by an advanced diagnostic application, a brief example of how a general communication flow between an application and an ECU can be composed is given. The functions need to be called in the order of appearance. The asterisk (*) in the function parameters represents a pointer, which holds an address to a memory location.

1. **PassThruOpen(*pName, *pDeviceID*)**
   The PassThruOpen-function is a prerequisite for further calls to be recognized. The function establishes a connection to and initialize the Pass-Thru device. *pName* must be NULL and is reserved for use of multiple devices. *pDeviceID* is a pointer to the location of the assigned device Id.

2. **PassThruConnect(*DeviceID, ProtocolID, Flags, BaudRate, *pChannelID*)**
   The function is used to set up a logical connection with a protocol channel on the device. A specific combination of parameters is set depending on the ECU which is to be communicated with. *pChannelID* is a pointer to the location for the assigned channel Id, used for further communication.

3. **PassThruStartMsgFilter(*ChannelID, FilterType, *pMaskMsg, *pPatternMsg, *pFlowControlMsg, *pFilterID*)**
   The function must be called before any messages can be read. Prior to this all incoming messages are blocked. This decides i.e. what type of filter to be used and how the mask should be composed.

4. **PassThruIoctl(*ChannelID, IoctlID, *pInput, *pOutput*)**
   The function reads and writes all the protocol hardware and software configuration parameters. *IoctlID* specifies the type of configuration to be performed, e.g. SET_CONFIG, GET_CONFIG and CLEAR_RX_BUFFER. *pInput* is a pointer to an input structure with configurations. *pOutput* is a pointer to the location for the output structure.

After the *PassThruIoctl* is run everything is configured to read and write messages from the vehicle. The structure of the messages depends on the used protocol. How the message functions are called is seen below.

- **PassThruReadMsgs(*ChannelID, *pMsg, *pNumMsgs, Timeout*)**
  The function reads messages and indications from the receive buffer. *pMsg* is a pointer to the structure of the messages, *pNumMsgs* is a pointer to the location of the specified number of messages to read and *Timeout* specifies the time limit before a timeout occur when not receiving the expected number of messages.

- **PassThruWriteMsgs(*ChannelID, *pMsg, *pNumMsgs, Timeout*)**
  The function is used to place messages in the transmit buffer. *pMsg* is a pointer to the structure of the messages, *pNumMsgs* is a pointer to the location of the specified number of messages to write. *Timeout* specifies the time to wait for that an error has occurred or that all messages have been transmitted.

After a successful session and the connection to the ECU must be terminated, the two following SAE J2534 functions have to be called.

- **PassThruDisconnect(*ChannelID*)**
  The function is used to terminate the connection with a protocol channel, the channel corresponding to *ChannelID*.

- **PassThruClose(*DeviceID*)**
  The function is used to close the connection to the Pass-Thru device with the Id of *DeviceID*.

If the functions run succesfully when called, they return the value STATUS_NOERROR.

3.6 Related work

There are solutions on the market today for running diagnostics remotely. They are briefly described in this section.

3.6.1 Volvo Trucks' Remote Diagnostics

Volvo Trucks' *Remote Diagnostics* helps maximize the uptime of a truck. By a telematic solution, they have enabled the possibility for Volvo Action Service, Volvo's 24/7 support

team, to run basic diagnostics remotely. When the vehicle arrive at the repair facility the service is prepared, thanks to the information sent in advance. Since the problem is already located, spare parts are ordered and in stock. All models in North America has this as standard feature since mid-2012 [20, 21]. This implicates an additional production cost for each individual truck, since a telematic unit needs to be installed.

3.6.2 General Motors' OnStar

General Motors is one of the companies behind a service called *OnStar* and in 1997 the first unit was available for cars. It uses Code Division Multiple Access (CDMA) cellular technology for data transmission and voice communication. *OnStar* transmits data from Global Positioning System (GPS) and the on-board diagnostics system, such as DTCs, which is used by the driver for health reports or *OnStar's* call center. The call center can e.g. be notified when an airbag deploys in case of an accident or assist the driver with help by a hands-free call [22]. This solution implicates, just as for Volvo Trucks' *Remote Diagnostics*, an additional production cost for each individual car.

3.6.3 Diagnostics over IP

The ISO 13400 standard, more known as the Diagnostics over IP (DoIP) protocol, enables communication using several different data link layer technologies which eases the work when implementing new communication devices. It will simplify the communication between real-time systems of vehicles and external diagnostic application.

The DoIP protocol consists of four parts. The first part covers general information and concentrates on communication between the real-time system of a vehicle and external diagnostic application through Ethernet, Local Area Network (LAN) or Wireless Local Area Network (WLAN). The second part outlines the transport protocol and network layer services. It contains IP address assignment, vehicle- announcement and discovery, connection establishment, communication protocol message format, data routing to vehicle ECUs, status information and error handling. When following the given specification of DoIP, data exchange can be done without being on the same local network. The third and fourth part defines the data link layer and physical layer requirements [5]. Since DoIP is an onboard solution it will result in an additional production cost for each individual vehicle, but on the other hand it does not require a Pass-Thru device for connectivity.

3.6.4 ISO 22900

The ISO 22900 is a standard which specifies a Diagnostic Protocol Data Unit (D-PDU) API as a Modular Vehicle Communication Interface (MVCI) protocol module software interface, which is generic and protocol independent. This forms a common basis for diagnostics and advanced diagnostic applications which is intended to provide a "plug-and-play" concept for the access to the MVCI protocol modules from different device

manufacturers. The MVCI protocol module is the key component to exchange implementations of the diagnostic protocols used by different vehicle manufacturers and device suppliers without re-engineering the software which is already implemented. The MVCI protocol module translates the calls made from the diagnostic application using the D-PDU API into a predefined protocol and passes it to the vehicle connected.

It also covers how to migrate existing standards, such as SAE J2534, to the ISO 22900 standard by the use of compatibility layers and wrappers. This is described in order to avoid older SAE J2534 based diagnostic applications to be obsolete [9].

Summary of chapter

The chapter provides the technical background which is needed to understand why and how the solution is of importance. It treats the ECUs of a car, the OSI-model, the protocols and the standards used. Some related work is presented and explained. The presented technical background is used to develop a proof of concept which is presented in Chapter 4.

4

The ARN concept

This chapter presents how the research question and the stated problems specified in Section 1.3 are solved by the developed concept named *Artificial Remote Networker* (ARN). The challenging parts are explained in the following sections.

4.1 Unlocking of ECU

Advanced diagnostics requires a login diagnostic service to be performed to gain access to all of the functions of the ECUs. In commonly used standards, this diagnostic service is called *SecurityAccess* which uses a seed/key algorithm for granting access. If the ECU is unlocked, all of its functionality will be available. The general procedure is carried out in the following sequence:

1. The advanced diagnostic application requests a seed from the ECU.
2. The ECU sends a seed back to the advanced diagnostic application.
3. The seed is run through an algorithm in the advanced diagnostic application and the answer, the key, is sent to the ECU.
4. If the received key corresponds to the expected key the ECU will unlock and grant access.

The design of the seed/key algorithm in the advanced diagnostic application is proprietary to the OEMs. There are two types of seed: *static* and *dynamic*. *Static* means that the same seed is returned irrespective of when or how many times it is requested and *dynamic* is stated to be updated at every single request. The latter can be dependent of e.g. the current time or a random parameter. There also exists a combination of the different types of seeds which is static over some time interval or a number of requests.

In order to prevent unauthorized connections, ECUs are designed only to unlock if the key is received within a specified time interval after sending the seed, else an internal timer will timeout and not grant access. This interval depends on the specification set by the manufacturer and sometimes even on the specific ECU. There are ECUs on the market today that demands an answer within as little as 50 ms. When high latency is introduced into the system, the ECU will not receive the key in time and hence timeout.

When the ECU has been unlocked all functionality will be available. This includes the advanced diagnostic functions, which hence are not time critical themselves. The only requirements for the functions to stay executable are that the ECU has an uninterrupted connection and receives a recurrent keep-alive message within a specified time interval. This is further explained in Section 4.2.

The ARN concept has an *unlocking* phase illustrated in Figure 4.1. In Figure 4.2 the interaction of the ARN concept towards the advanced diagnostic application and the ECU is shown. It is defined to start when the ECU sends a seed to the advanced diagnostic application. The ARN concept stores the received seed while it is transferred through the ARN Pass-Thru device. The advanced diagnostic application reads the seed, calculates a key and sends it back to the ECU. In the same manner as for the seed, the key is stored. If the key is correct and delivered in time the ECU unlocks and returns a positive response message. If the ECU times out before receiving the key it remains locked and returns a negative response message. The message is detected by the ARN concept and a *requestSeed* is called again, without any action taken by the advanced diagnostic application. If the new seed from the ECU is the same as the stored seed, the ARN concept replies by sending the stored key to the ECU. This is performed on the vehicle side and hence without the high latency present. If the key is correct the ECU unlocks and sends a positive response message to the diagnostic application. This solution, from the diagnostic application view, looks like a normal advanced diagnostic procedure because of the ARN concept's ability to handle communication without the knowledge of the application.

4.1.1 Storage of seed/key pairs

An alternative solution to the login diagnostic service problem investigated is to save all seed/key pairs in lookup tables locally. This solution removes the latency caused by the transferring media, e.g. Internet, in a remote solution, since the whole login diagnostic service is handled at the vehicle side.

If the amount of ECUs used by the vehicle manufacturers is estimated to circa 200 and there are approximately 20 relatively large manufacturers, there are a total of 4000 different ECUs which needs to be handled. Each of these units have their own component Id. To each of the component Ids, a table of seed/key pair is assigned. The seeds and keys are assumed to be 2 bytes long respectively. This means that there are 2^{16} different pairs in the tables in a worst case scenario. Each of these pairs occupies 4 bytes and 4000 tables would contain 0,98 GB (1 048 576 000 bytes) of data. In addition to this, it would need another table containing the component Ids with corresponding address to the correct seed/key table. If it is assumed that a component Id is 2 bytes long the

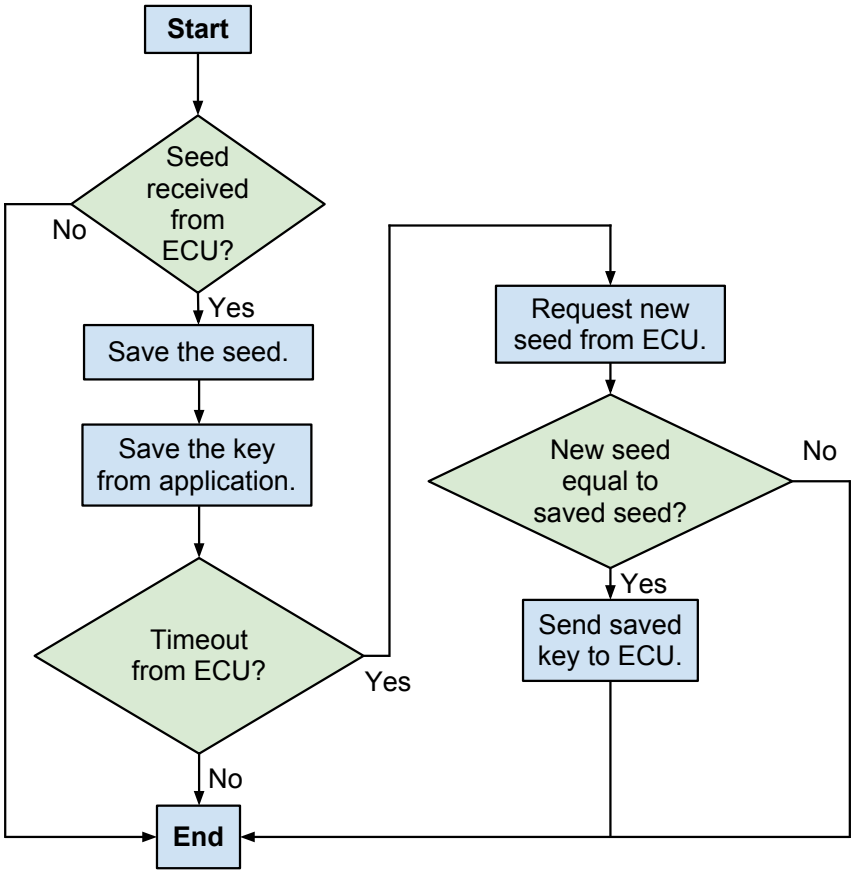

Figure 4.1: Flow chart for the *unlocking* phase of the ARN concept.

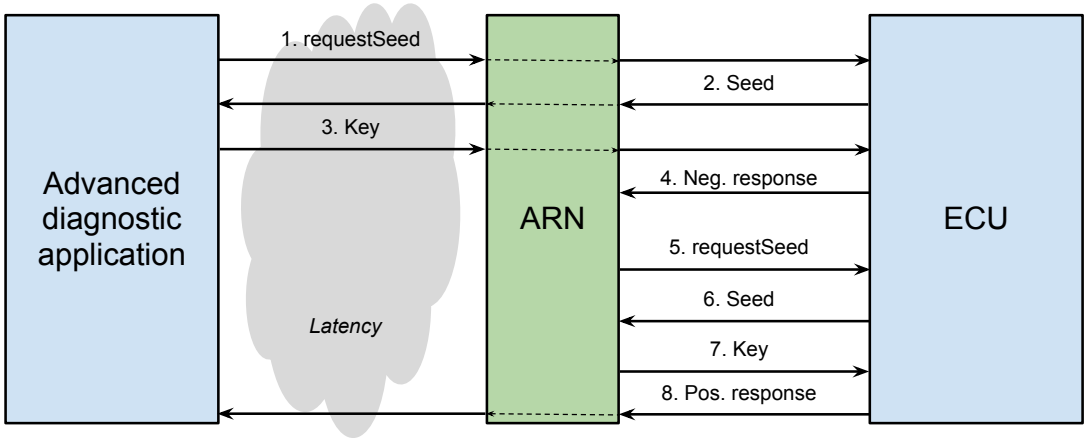

Figure 4.2: Overview of an ARN Pass-Thru device handling a *SecurityAccess* service from the advanced diagnostic application to the ECU.

amount of data generated by this can be neglected. The total size of required storage will not be a problem.

A problem which is not handled when using this solution is that it is not brand unspecific in a general way. It only handles the vehicles which are already incorporated into the system. Every time a new model of an ECU is introduced in a vehicle, the look-up tables need to be updated and therefore hard to keep up-to-date. To solve this the ARN Pass-Thru device can be reading from an external database continuously where new seed/key pairs are added as new models are introduced to the market. A problem with this solution is that the vehicle manufacturers are not willing to provide a third party company with lookup tables with all their seed/key pairs, since it is proprietary information. If the locally stored tables leak out to the general public and spread on Internet, the security which prevents unauthorized persons from updating the ECUs is spoiled. Hence the solution is not developed any further.

4.2 Keep-alive message

When an ECU unlocks it requires a received recurrent keep-alive message within a specific time interval, specified for each ECU by the manufacturer, in order stay unlocked. In commonly used standards this is achieved by a diagnostic service called *TesterPresent*, which is called from the advanced diagnostic application to guarantee that a tester is present. When introducing high latency into the system, the advanced diagnostic application will try to start sending *TesterPresent* to the ECU but it will not receive these messages in time after the login phase is finished and hence directly return to a locked state and a new login is needed to execute advanced diagnostics.

This problem is solved by a phase called the *keep-alive* phase, shown in Figure 4.3. The phase supports the ECUs which is connected and unlocked with the required *TesterPresent* messages. The ARN concept detects when an ECU unlocks by reading its answer to the *SecurityAccess* service and automatically starts to send keep-alive messages. This is performed at the vehicle side, in the ARN Pass-Thru device, thus the problem with too much latency to keep the ECU unlocked is eliminated.

4.3 Blocking mode

The manufacturers usually protect their ECUs from unauthorized connections with a security mode, which means that it can be blocked for a predefined time. This counteracts acquisition of seed/key pairs by *brute-force* attacks. If an ECU receives a faulty key it will increment an internal counter. If the counter exceeds a predefined calibration value, the ECU will enter a blocking mode for a specified time decided by the manufacturer. If this scenario is repeated, the blocking time can e.g. be increased by some predefined factor. It is noteworthy that the counter does not increase when sending a seed, but only when receiving a faulty key.

To avoid that the ECUs enter the blocking mode, the ARN concept is designed not to try to unlock them if the acquired seeds differs. If the seeds are different the solution

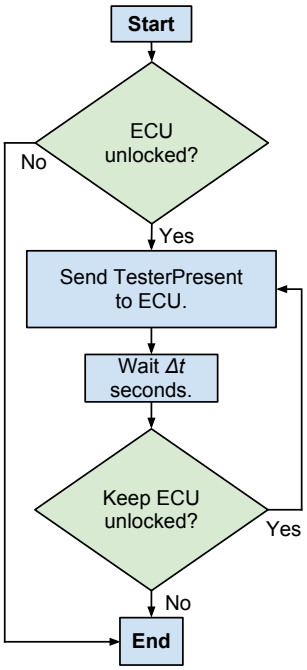

Figure 4.3: Flow chart for the *keep-alive* phase of the ARN concept.

for dealing with high latency of the ARN concept is not applicable and a login attempt is not of any use.

4.4 Examination of compatibility

The ARN concept is developed to have the possibility to enter a *compatibility* phase according to Figure 4.4 as soon as the ARN Pass-Thru device is connected to an ECU. In Figure 4.5 the interaction of the ARN concept towards the advanced diagnostic application and ECU is shown. Without any action taken from the advanced diagnostic application, the ARN Pass-Thru device examines the possibility of running time critical diagnostic services at the present ECU.

When the ARN Pass-Thru device is connected and setup for communication, it sends a *requestSeed* function call to receive a seed from the ECU. The procedure is run at least once more, where each iteration is delayed by some time. The received seeds are compared to each other to decide whether the seed is static or not. Since the seeds are requested without answering with a key, it will not increment the internal counter of the ECU and will not risk that the ECU enters a blocking mode. By the same reason, the ARN concept can request more seeds over a longer time span for a better knowledge about the ECU. If the seeds are static over a time greater than the Round Trip Time (RTT) plus the time it takes for the advanced diagnostic application to generate the key and for the ARN concept to save the key and request for a new seed, the concept works.

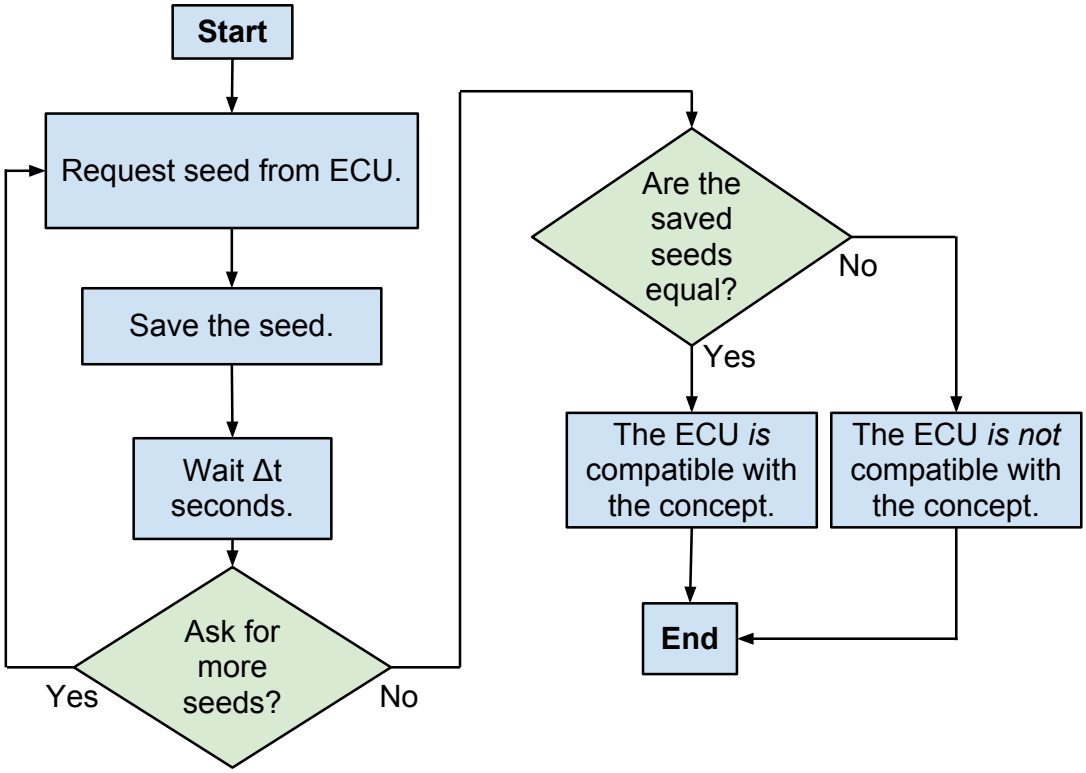

Figure 4.4: Flow chart for the *compatibility* phase of the ARN concept.

The procedure is illustrated in Figure 4.6. This compatibility procedure can be used to tell the advanced diagnostic application if a remote access to the ECU can be performed, or more precise, if time critical diagnostic services can be run or not with high latency present.

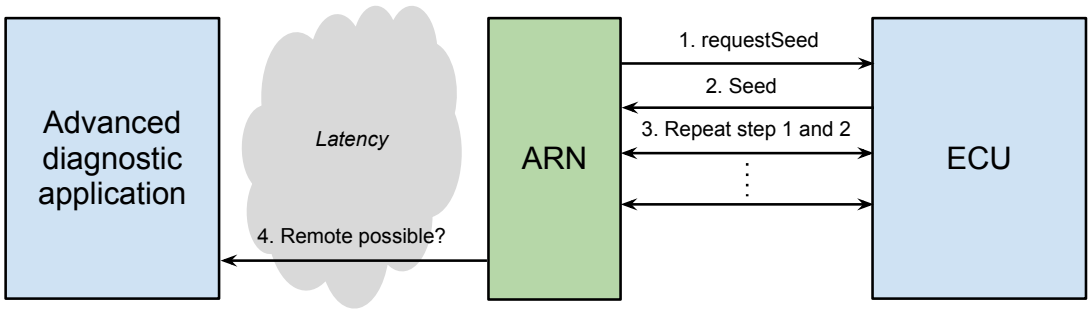

Figure 4.5: Compatibility check for an ECU performed by an ARN Pass-Thru device.

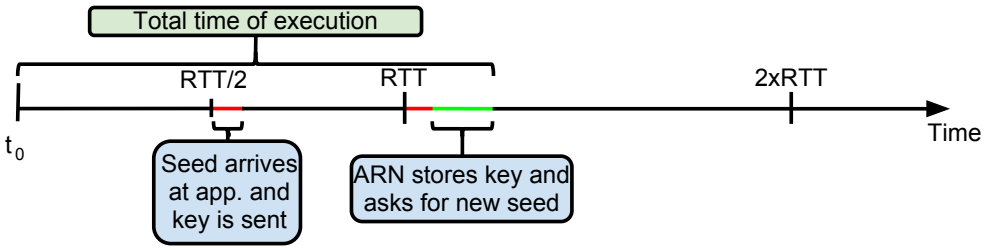

Figure 4.6: The total execution time of the process of receiving a key. At t_0 a seed is requested from the ECU. The total time of execution consists of the transferring time together with the time it takes to process the seed and key.

4.5 Extention of SAE J2534

A prerequisite for the ARN concept to work is that the seeds of the ECU are static, or at least static over a time long enough as described in Section 4.1. The temporary local storage of a seed and its corresponding key does not work if the ECU generates dynamic seeds. A proposal of a future solution to this problem is to extend the SAE J2534 standard. The new standard must allow the manufacturers to send their proprietary data, i.e. their algorithms for generating a key from a seed, in an encrypted and secure way to the vehicle side. This is done by introducing a new function in SAE J2534, the *PassThruSecurityAccessAlgorithm*. The function works in the same manners as the *PassThruSendMsgs* does, but takes one more parameter where the developers can include an algorithm of how to calculate a key from a seed when a *requestSeed* functionality for the *SecurityAccess* diagnostic service is called. This algorithm is temporarily stored at the vehicle side in the ARN Pass-Thru device. The device detects when a seed is sent from the ECU back to the advanced diagnostic application to receive a key and instead of passing it through, the seed is used as a parameter for the received algorithm. When the key is derived it is returned to the ECU immediately. This solution moves the time critical login diagnostic service to be run at the vehicle side and the response of the login attempt is not returned until the whole service is finished. The interaction of the extension with the advanced diagnostic application and the ECU is illustrated in Figure 4.7 and the ARN concept is shown as a flow chart in Figure 4.8.

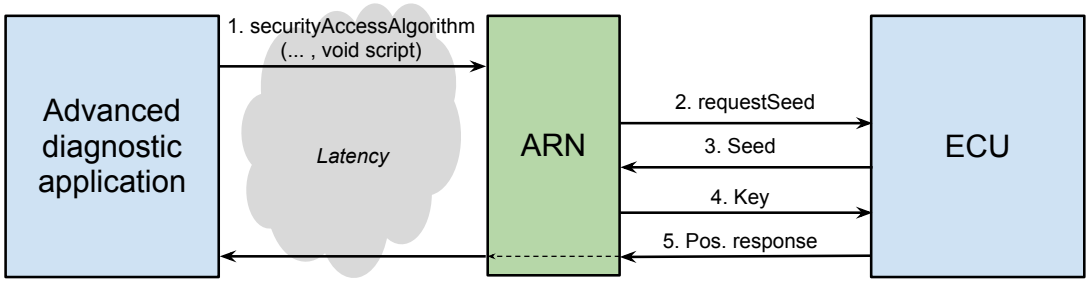

Figure 4.7: Overview of the ARN concept with the extended SAE J2534 standard.

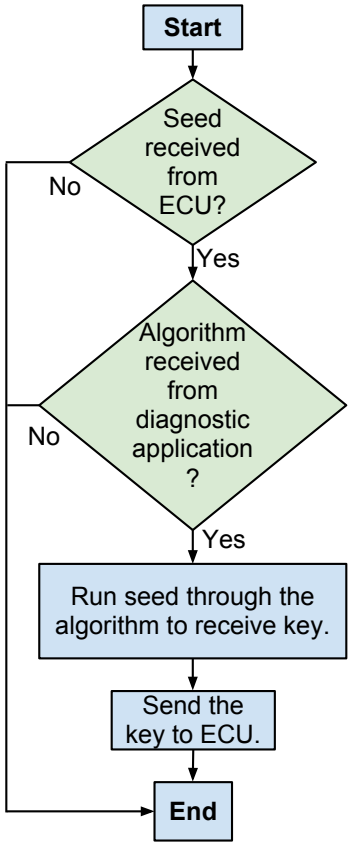

Figure 4.8: Flow chart for the unlocking of the ECU with the extended standard implemented.

Summary of chapter

In this chapter a developed concept for solving the research question and the surrounding problems are presented, namely the ARN concept. It is designed to enable remote

advanced diagnostics for as many kinds of ECUs as possible by use of the standard of today. The ARN Pass-Thru device is developed to move the time critical diagnostic services to run at the vehicle side of the high latency bus. Investigations of different solutions are shown and an explanation of why these do not work is given. To prove that the ARN concept is working it needs to be implemented as a proof of concept. This process is described in Chapter 5.

5

Proof of concept

This chapter describes the implementation of the developed ARN concept. It explains how affecting surrounding environment, such as latency, is implemented and simulated to achieve realistic results. Further, it specifies where and how the ARN concept is implemented, phase by phase.

5.1 Simulation and implementation of the ARN concept

The implementation of the ARN concept is carried out by developing a middle layer between the advanced diagnostic application and an arbitrary Pass-Thru device. The layer is implemented as a DLL file invoked by the advanced diagnostic application and passes the sent parameters along to a proprietary DLL file which includes the SAE J2534 functions of the currently used Pass-Thru device. The layer includes all the SAE J2534 functions specified in the standard as internal methods, but instead of directly execute the function calls at the ECU, they are passed through to the Pass-Thru device using the proprietary DLL file.

The implementation method gives many benefits. It allows a high latency bus to be simulated and for the additional logic in the ARN concept to be inserted between the advanced diagnostic application and the ECU, as if it was included in an ARN Pass-Thru device. This way of implementing the ARN concept allows for the new functionality to be implemented on the vehicle side, on top of the Pass-Thru device. By placing all functionality after the latency seen from the application side, the implementation works as if a Pass-Thru device, with the ARN concept implemented as an extension, is connected.

The high latency bus is simulated by using delays inserted between the advanced diagnostic application and the ARN concept logic. These delays are run each time a function call is made by the advanced diagnostic application, without interfering with either the application nor the Pass-Thru device, and can be seen as just a high latency

connection between the two sides.

5.2 Unlocking phase

To handle the unlocking phase the ARN concept detects seeds and keys passed between the advanced diagnostic application and the Pass-Thru device. It is also able to identify the difference between a negative and positive response message from the ECU. Table 5.1-5.4 shows how GMLAN specifies the structure of these messages. In GMLAN the negative response identifier is specified to be *$7F* and this proceeds every response message which indicates that the requested service is not allowed. The data which follows it specifies why the service is not allowed to run. How this is handled is shown in Algorithm 1 as pseudocode.

#1	#2	#3	#4	#5	#6	#7	#8
$02	$27	$01	—	—	—	—	—

Table 5.1: CAN frame of a *requestSeed* message

#1	#2	#3	#4	#5	#6	#7	#8
$04	$27	$02	$cc	$dd	—	—	—

Table 5.2: CAN frame of a key message, where $cc $dd is the key

#1	#2	#3	#4	#5	#6	#7	#8
$03	$7F	$27	$22	—	—	—	—

Table 5.3: CAN frame of a negative response message (timeout)

#1	#2	#3	#4	#5	#6	#7	#8
$02	$67	$02	—	—	—	—	—

Table 5.4: CAN frame of a positive response message

```
Result: unlocked targetECU
if seed from targetECU then
    save targetSeed from targetECU to seed;
    save targetKey from targetApp to key;
    if negative response message from targetECU then
        write $27 01 to targetECU;
        if targetSeed == seed then
            write key to targetECU;
            if positive response message from targetECU then
                return true;
            else
                return false;
            end
        else
            return false;
        end
    else
        return true;
    end
end
```

Algorithm 1: Pseudocode for unlocking phase. Data size of CAN message is ignored.

5.3 Keep-alive phase

In this phase the ARN concept prevents an unlocked ECU from locking again. To do this it detects a positive response message from the unlocking phase and starts sending *TesterPresent* calls directly from the vehicle side. Table 5.4-5.5 shows how GMLAN specifies the structure of these CAN messages. The implementation of this phase is shown as pseudocode in Algorithm 2. The *TesterPresent* message is sent as a periodic message to the ECU each Δt second, where Δt is specified by the manufacturer of the ECU. This is continued until the connection between the advanced diagnostic application and the ECU is closed.

#1	#2	#3	#4	#5	#6	#7	#8
$01	$3E	—	—	—	—	—	—

Table 5.5: CAN frame of a *TesterPresent* message

```
Data: Δt - time between keep-alive messages
      unlocked - state returned from Algorithm 1
1 if unlocked then
2 |   while keep alive from diagnostic application do
3 |   |   write $3E to targetECU;
4 |   |   wait Δt seconds;
5 |   end
6 end
```

Algorithm 2: Pseudocode for keep-alive phase. Data size of CAN message is ignored.

#1	#2	#3	#4	#5	#6	#7	#8
$04	$67	$01	$aa	$bb	—	—	—

Table 5.6: CAN frame of a seed message, where $aa $bb is the seed

5.4 Compatibility phase

In this phase the ARN concept calls *requestSeed* to receive seeds and store them with a time delay between each request. The seeds received are compared to each other to deduce if the ECU uses static seeds or not. Table 5.1 and 5.6 shows how GMLAN specifies the structure of these messages. The implementation is shown in Algorithm 3 as pseudocode.

5.5 Extension of SAE J2534

The implementation of the extended SAE J2534 standard makes it possible to run advanced diagnostics when high latency is present for all types of seeds. The procedure is shown as pseudocode in Algorithm 4. The algorithm is assumed to have received the seed/key algorithm in prior, when the *requestSeed* functionality of the *SecurityAcces* diagnostic service was run, by using the new SAE J2534 function *PassThruSecurityAccessAlgorithm*.

```
Data: nbSeeds - number of seeds to compare
      Δt - time between seed requests
Result: possibility to run advanced diagnostics with high latency present
1  for i=0 to nbSeeds-1 do
2  |  write $27 01 to targetECU;
3  |  save targetSeed from targetECU to seed[i];
4  |  wait Δt seconds;
5  end
6  for i=0 to nbSeeds-2 do
7  |  if seed[i] != seed[i+1] then
8  |  |  return false;
9  |  end
10 end
11 return true;
```

Algorithm 3: Pseudocode for compatibility phase. Data size of CAN message is ignored.

```
Data: Algorithm - algorithm passed from application
Result: unlocked targetECU
1  if seed from targetECU then
2  |  save targetKey from Algorithm(targetSeed) to key;
3  |  write key to targetECU;
4  |  if positive response from targetECU then
5  |  |  return true;
6  |  else
7  |  |  return false;
8  |  end
9  end
```

Algorithm 4: Pseudocode for login diagnostic service with the extended standard implemented.

Summary of chapter

The chapter describes how the ARN concept developed in Chapter 4 is implemented. The implementation is run in a simulated environment, where the latency is adjustable, and this is also described. The implementation of the ARN concept is divided into three phases: unlocking-, keep-alive- and compatibility phase. Also, a proposal of how to implement the extended SAE J2534 standard which is described in Section 4.5 to solve the cases where the ARN concept is not working is given. The tests designed to evaluate the functionality of the ARN concept is described in Chapter 6. It also describes the test environment used and which results were achieved by them.

6

Evaluation

To prove that the implemented ARN concept in Chapter 5 behaves as intended, a series of test cases are conducted. They are used to show that the described problems occur when an adequate amount of latency is present and that the developed ARN concept solves them as intended. This chapter also describes the design of the tests.

6.1 Test environment

The ARN concept is evaluated in a test environment consisting of a subsystem of the electrical system of a car distributed by a major vehicle manufacturer. This includes a set of 10 ECUs connected by a CAN bus, with a SAE J1962 connector for diagnostic purposes. The protocol used by the environment for the application- and transport layer is GMLAN. The environment runs the ECUs and gives DTCs in the same way as a regular car do by a connected unit which simulates sensor inputs.

The connection between this environment and the PC is handled by a fully SAE J2534 compatible Pass-Thru device, which supports this specific setup, with an USB-connection. It also has possibilities to connect to the PC with the use of Ethernet, WLAN and RS-232.

The advanced diagnostic application used to communicate with the ECUs is developed by Diadrom. It is an application for running diagnostics of vehicles of different brands and includes e.g. the feature that the user is able to send protocol specific commands at byte level.

To analyze the communication between the diagnostic application and the ECU on the CAN bus the software tool CANalyzer developed by Vector is used. The CANalyzer observes all traffic occurring at the CAN bus and may also be used to send and log data [23].

6.2 Tests for evaluation

To evaluate that the implemented ARN concept works as expected, five tests are designed to cover the different problem areas that are pointed out in Chapter 4.

Problem observation

The problem observation is conducted to prove the hypothesis that modern real-time systems in cars will timeout due to high latency when running time critical functions. This is observed by running several iterations of relevant functions and increasing the latency time in the implemented code at the application side of the system. The simulated latency will be increased stepwise from 0 to 6 seconds to find if and where the timeout occurs.

Non-interference

To investigate if the ARN concept interferes with regular use of diagnostic applications, such as basic- and advanced diagnostics without any appreciable latency present, the non-interference test is designed. The test observes that the implemented logic is not utilised when using the implemented DLL layer without introducing high latency into the system by logging the execution.

Unlocking phase

To evaluate that the unlocking phase of the ARN concept works as expected, the time critical diagnostic service *SecurityAccess* is run with a latency greater than the limit acquired in the problem observation introduced. If the ARN concept is able to unlock the ECU even though the latency is high, it proves that this phase works.

Keep-alive phase

The ARN concept prevents an unlocked ECU from locking again by sending *TesterPresent* calls to it from the Pass-Thru device at the vehicle side. To evaluate that the solution works as expected, a test, which does not take any action for 30 seconds after an ECU is unlocked and then investigates if it is still unlocked, is designed. If so, the keep-alive phase works.

Compatibility phase

The compatibility phase, where the ECUs of the vehicle are examined, is supposed to state whether the ARN concept is compatible or not. A good way to do this is to try the ARN concept in different test environments, where different types of seeds are used and the latency can be altered. This is not conducted since the access to test environments is limited. The only test envirionment available uses static seeds. This will be used for evaluation of the compatibility phase.

6.3 Test results

The results from the observation of when the problem appears for the used test environment are presented in Figure 6.1. When no latency is present all unlocking attempts succeeds and when the latency is greater than 4 seconds all attempts fail. A threshold value is detected when the latency exceeds 3 seconds. At this value more than half of the unlocking attempts fails.

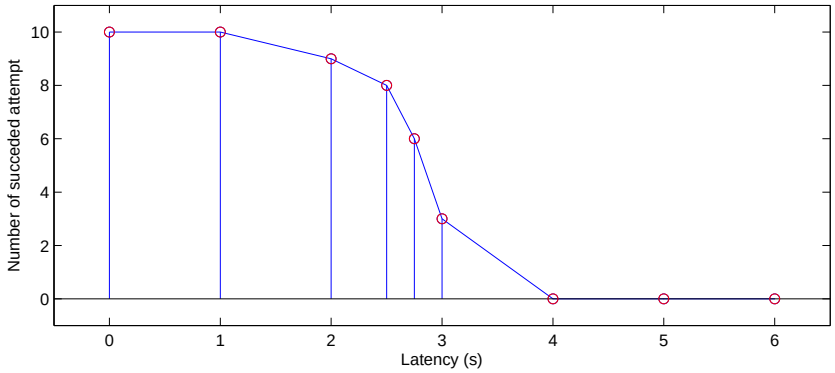

Figure 6.1: Number of succeed unlocking attempts as the latency increases. 10 attempts are conducted at each tested latency value. When the latency approaches 3 seconds, less than half of the attempts succeeds.

The results of the remaining tests are presented in Table 6.1.

Test	Passed
Non-interference	X
Unlocking phase	X
Keep-alive phase	X
Compatibility phase	X

Table 6.1: Presentation of the test results for the ARN concept evaluation.

All conducted tests where successful. Since the advanced diagnostic services require that the ECU is unlocked during the entire diagnostic process, the test results indicates that the ARN concept enables advanced diagnostic services to be run with high latency present. Further interpretations of the test results are discussed in Chapter 7.

Summary of chapter

A description of the used test environment, on which the ARN concept is implemented, is given in this chapter. The problem verification observes that the problem arises when high latency is present. The non-interference test shows that the ARN concept is not run when the latency is not a problem. The test for the unlocking phase is conducted to investigate if the ECU unlocks when the ARN concept is run. The keep-alive phase is tested to see that the ECU remains unlocked and the compatibility phase is investigated to see whether it returns the correct evaluation of if the ECUs are compatible with the ARN concept or not. Further, a presentation of which tests are passed summarizes the chapter. The outcome of the results and how they are interpreted are discussed in Chapter 7.

7

Discussion

Throughout the entire thesis it is of importance that the ARN concept works for an arbitrary advanced diagnostic application and an arbitrary car. The ARN concept is intended to be implemented on its own Pass-Thru device, the ARN Pass-Thru device, and in a proprietary DLL file, used by the application, specific for this device. By doing so, the transfer bus between them can be of an arbitrary type. For example, in the case where an independent repair facility needs assistance from experts, see Chapter 1, the SAE J2534 function calls could be sent over Internet using remote procedure calls. Since the application uses the new DLL file as an ordinary one, the messages sent to the application can be controlled and in that way it does not see any difference from running normal advanced diagnostics.

The implementation of the ARN concept is made on a model of a real-time system from a real car running GMLAN over CAN using a SAE J2534 fully compliant Pass-Thru device. Since the implementation is based on a middle layer which does not introduce any new functions, only new functionality of the SAE J2534 functions, the only thing which has to be done when changing to an arbitrary Pass-Thru device, is to bind the ARN DLL file to the proprietary DLL file of that Pass-Thru device instead.

The test environment used in this thesis is highly tolerant when performing time critical functions. Through observations it is observed that the threshold time for allowing *SecurityAccess* is when the latency exceeds 3 seconds. Probably, it would work to run time critical functions directly over Internet with this environment, because of this tolerant time span. Through interviews it was found out that this is one of the most generous time intervals on the market and there are a huge amount of ECUs with shorter accepting intervals where the ARN concept is needed. The ARN concept is able to handle the latency and unlock ECUs, as long as the seed is not dynamic, with very short allowing time interval, as seen in the test results in Section 6.3. If the seeds of the ECU is dynamic, the ARN concept detects this in the compatibility phase and could report this directly to the diagnostic application. Further, the test results shows that if

an ECU is unlocked, it will stay so by the help of the ARN concept until the connection is closed.

Due to limitations of the accessibility of different test environments, the only implemented protocol is GMLAN over CAN. This does not mean that the ARN concept is not implementable with other protocols, the conversion would only require some slight modifications to comprehend with the other protocols specified in SAE J2534, presented in Section 3.5. This would make not only the ARN concept brand unspecific, but also the implementation of it, since it is able to run at any vehicle which has implemented the SAE J2534.

The related work investigated gave the impression that the automotive industry is becoming more and more interested in the field of telematic solutions. One can argue that the concept for solving problems arising when there are high latency in the system, as it might be in these cases, is obsolete in the sense that there already exist many telematics solutions on the market today which deals with this problem. As stated in Section 3.6.1-3.6.3 both Volvo Trucks and General Motors have their own proprietary solutions and new standards are developed, such as the DoIP. But the ARN concept is unique by not implicating additional production cost for the individual car by adding new hardware. Further, it also deals with the already existing vehicle fleet which has been produced since the SAE J2534 standard was introduced and statutorily in North America and Europe. These vehicles will be in use for many years to come and it is foremost these the ARN concept is targeting. We also think that the standards of today will be phased out in a very slow pace, since introductions of new standards are expensive and will most likely contain faults at their release. If the standards used today, such as the SAE J2534, is updated or extended as proposed in Section 4.5 instead of introducing a brand new standard it may reduce the amount of faults at the introduction of it, since it does not need to introduce that much new functionality. It will also make it easier for the developers, which are already used to the old standard, since they only need to learn to use the new functionality. By this reasoning the ARN concept will not be obsolete in a long time.

A problem observed when the ARN concept is implemented is if the advanced diagnostic application uses internal timers to prevent the application from locking when no response is received. If these timers are too short, the application treats the call to the ARN Pass-Thru device as lost and timeouts. This is easy to solve by adjusting the timers lengths, but since one of the goals of the ARN concept is to work with every advanced diagnostic application without altering it, this is not a good solution. Another solution is to use pending messages. These messages indicates that the ECU is still working but needs more time. The pending message function is implemented in GMLAN and it is likely that there exists a similar function in the other protocols specified in SAE J2534.

When the keep-alive phase is implemented, it is observed that the internal buffer of the Pass-Thru device overflows. When a *TesterPresent* function is called, the Pass-Thru device indicates that the transmission is done and stores this flag in its internal buffer. Since this is executed every Δt second, which is a shorter interval than the latency, the size of the buffer increases. The buffer can not simply be cleared because the risk of

mistakenly remove a message expected by the advanced diagnostic application. If the solution is implemented in an ARN Pass-Thru device developed from the scratch it could handle the keep-alive phase internally and not add the response of the *TesterPresent* sent by itself to the receive buffer. A better solution is to collect messages on the vehicle side and send them as packages to a new buffer on the diagnostic application side. As the implementation is made in this thesis, each reading is sent forth and back in the same pace as the diagnostic application calls the Pass-Thru function. This makes the thread in the application lock as it waits for the answer. If it reads from a local buffer, the reading rate would increase significantly. If the the D-PDU API specified in the ISO 22900 standard is used instead, there are functions implemented for sending messages to the ECU and ignoring the response. This helps the developers to attend to the observed problem but would reduce the number of vehicles that the ARN concept is applicable to.

Since the ARN concept is observing the messages sent through the Pass-Thru device, it cannot be said to be fully compliant to SAE J2534. SAE J2534 states that the Pass-Thru device should not interpret the message content, only pass it forward. The ARN concept is not interpreting the messages to see what functionality is being passed through, the messages are merely monitored to detect the time critical functions to know when the messages has to be saved and resend to enable the procedure requested by the advanced diagnostic application.

The proposed extension of SAE J2534 is not implemented for evaluation because of lack of time. This could be done in a test environment containing ECUs with dynamic seeds. The implementation would only affect the Pass-Thru devices and advanced diagnostic applications and not the already existing vehicles. If the proposal would be legislated, the manufacturers would need to implement the function in their software.

8

Conclusion

The developed and evaluated ARN concept proves that it is possible to, in an advanced diagnostic application, run time critical functions on the real-time system in a car when located elsewhere geographically and high latency is present, if the seeds of the ECUs are static, or are static long enough. The time critical functions includes the login diagnostic service and the providing of a recurrent keep-alive message to keep the ECU unlocked. Since the problems of running these with high latency present are resolved, advanced diagnostics can be performed.

The proposed ARN concept is developed based on the SAE J2534 standard to cover all cars produced in North America after 2004 and all cars produced in Europe since September 2010. The use of this legislation eliminates the difficulty of supporting many different brands and standards. Since the ARN concept provides the advanced diagnostic application with a SAE J2534 DLL file, the application will not experience any difference from a regular DLL file. Further, the application does not need to be changed because the ARN concept manages necessary actions without alerting the application. This implies that all already existing advanced diagnostic applications could work with this concept.

Pros of the ARN concept:

- Can unlock ECUs when high latency is present if the ECU has seeds which are static, or are static long enough.
- Prevents an unlocked ECU from returning to a locked mode because of inactivity.
- Can examine the possibility to run time critical functions on ECUs when high latency is present.
- Prevents an ECU from entering a blocking mode when running the ARN concept because of too many failed login attempts.

- Will only act when the latency is too high for running time critical functions.
- Only uses legislated standards of today.

Cons of the ARN concept:

- Cannot unlock ECUs when high latency is present if the ECU has dynamic seeds.

In the case when the seeds of the ECUs are dynamic, the ARN concept is not able to handle the time critical functions. Instead a proposal is given of how the SAE J2534 standard could be extended to support such functions to be run remotely, i.e. with high latency present.

Bibliography

[1] M. Broy, I. H. Kruger, A. Pretschner, C. Salzmann, (2007) Engineering automotive software, Proceedings of the IEEE 95 (2) 356–373.

[2] F. Ljungberg, H. Fagrell, (2011) Diagnostik; Att utveckla, producera och underhålla flexible mjukvarubaserade kapitalprodukter med hög tillgänglighet, Diadrom AB, Gothenburg.

[3] E. Johansson, S. Romero Skogh, H. Svensson, (2011) Software management within Product Development, Tech. rep., Department of Product and Production Development.

[4] Mobilscan, (2011) Product description for Mobilscan OBD, [Accessed: 20 May 2013].
URL `http://www.mobilscan.dk/2/en/descriptionobd.html`

[5] M. Johanson, P. Dahle, A. Soderberg, (2011) Remote Vehicle Diagnostics over the Internet using the DoIP Protocol, in: ICSNC 2011 : The Sixth International Conference on Systems and Networks Communications, IARIA, Barcelona.

[6] Drew Technologies Inc, (2003) SAE J2534 API reference, [Accessed: 30 January 2013].
URL `http://tunertools.com/prodimages/DrewTech/Manuals/PassThru_API-1.pdf`

[7] SAE International, (2004) Recommended Practice for Pass-Thru Vehicle Programming.

[8] European commission, (2011) Commission Regulation (EU) No 566/2011 of 8 June 2011 amending Regulation (EC) No 715/2007 of the European Parliament and of the Council and Commission Regulation (EC) No 692/2008 as regards access to vehicle repair and maintenance information, [Accessed: 20 May 2013].
URL `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32011R0566:EN:NOT`

[9] International Standardization Orginasation, (2008) ISO/FDIS 22900-2:2008(E).

[10] V. Vaishnavi, W. Kuechler, (2013) Design Research in Information Systems, [Accessed: 20 May 2013].
URL `http://desrist.org/design-research-in-information-systems/`

[11] A. Karimi, J. Olsson, J. Rydell, (2004) A Software Architecture Approach to Remote Vehicle Diagnostics, Tech. rep., Department of Informatics.

[12] C. Elbert, C. Jones, (2009) Embedded Software: Facts, Figures, and Future, Computer 42 42–52.

[13] International Organization for Standardization, (2002) Road vehicles Diagnostics on Controller Area Network (CAN) Part 1: General information.

[14] G. C. Buttazzo, (2011) Hard Real-Time Computing Systems, Springer US.

[15] J. Tyson, (2013) How OSI Works, [Accessed: 19 March 2013].
URL `http://computer.howstuffworks.com/osi.htm`

[16] I. T. UNION, (1994) Data networks and open system communication, open systems interconnection - model and notation, International Telecommunication Union.

[17] Bosch, (2012) What is CAN?, [Accessed: 18 May 2013].
URL `http://www.bosch-semiconductors.de/en/ubk_semiconductors/safe/ip_modules/what_is_can/what_is_can.html`

[18] GMLAN Diagnostic Sub-Group, (2004) GMLAN enhanced diagnostic test mode specification.

[19] OBD news, (2013) OBD2 data connector, [Accessed: 19 March 2013].
URL `http://www.obdnews.com/obd2-data-connection/`

[20] AB Volvo, Remote Diagnostics, [Accessed: 15 March 2013].
URL `http://www.volvotrucks.com/trucks/na/en-us/business_tools/connectedvehicleservices/Pages/default.aspx`

[21] B. Borgna, (2013) Volvo Trucks Announces Remote Diagnostics as Standard on All North American Models, [Accessed: 15 March 2013].
URL `http://www.volvogroup.com`

[22] OnStar, OnStar Vehicle Diagnostics, [Accessed: 16 March 2013].
URL `https://www.onstar.com/web/portal/ovdexplore`

[23] Vector Informatik, (2013) ECU analysis with CANalyzer, [Accessed: 12 May 2013].
URL `http://vector.com/vi_canalyzer_en.html`

A

Appendix - J2534 API Functions

Function	Description
PassThruOpen	Establish a connection with a Pass-Thru device.
PassThruClose	Terminate a connection with a Pass-Thru device.
PassThruConnect	Establish a connection with a protocol channel.
PassThruDisconnect	Terminate a connection with a protocol channel.
PassThruReadMsgs	Read message(s) from a protocol channel.
PassThruWriteMsgs	Write message(s) to a protocol channel.
PassThruStartPeriodicMsg	Start sending a message at a specified time interval on a protocol channel.
PassThruStopPeriodicMsg	Stop a periodic message.
PassThruStartMsgFilter	Start filtering incoming messages on a protocol channel.
PassThruStopMsgFilter	Stops filtering incoming messages on a protocol channel.
PassThruSetProgrammingVoltage	Set a programming voltage on a specific pin.
PassThruReadVersion	Reads the version information for the DLL and API.
PassThruGetLastError	Gets the text description of the last error.
PassThruIoctl	General I/O control functions for reading and writing protocol configuration parameters (e.g. initialization, baud rates, programming voltages, etc.).

Table A.1: J2534 function description.