实验 2　基于开源 AUTOSAR 的汽车故障诊断实验

实验目的：

掌握 AUTOSAR 诊断管理包含的模块与工作流程，深入理解 DoIP 协议的时序与格式，实现在 AUTOSAR 上发送服务请求报文，以及在 Linux 上使用 Wireshark 抓包进行报文分析。

实验内容：

1. 掌握 AUTOSAR 的诊断管理模块与诊断流程
2. 解析 DoIP 协议
3. 发送故障状态请求报文
4. Linux 上使用 Wireshark 进行报文分析

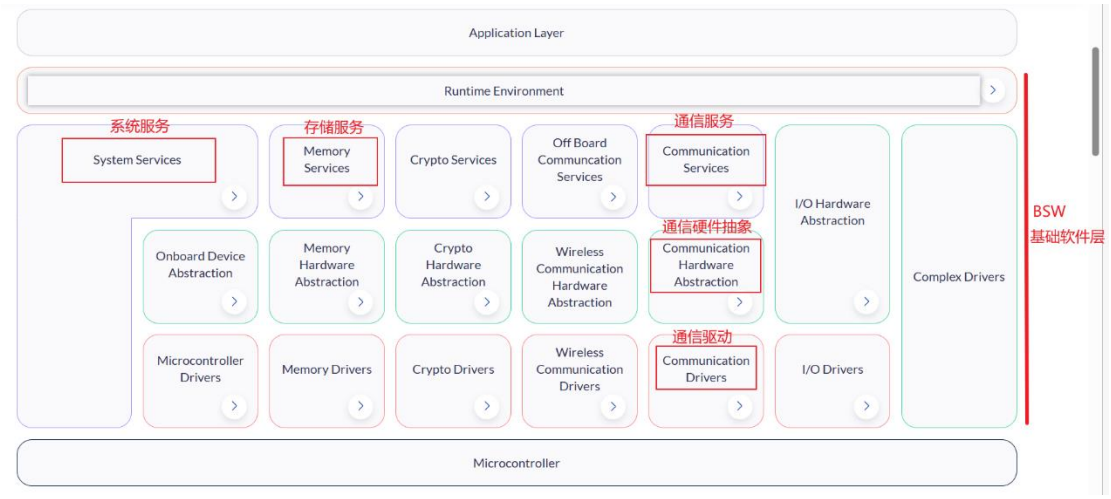
预备知识：

1. Linux、Git 基本命令

实验步骤：

1. AUTOSAR 诊断管理模块

查阅官方文档 https://www.AUTOSAR.org/standards/classic-platform，AUTOSAR 的结构如图所示

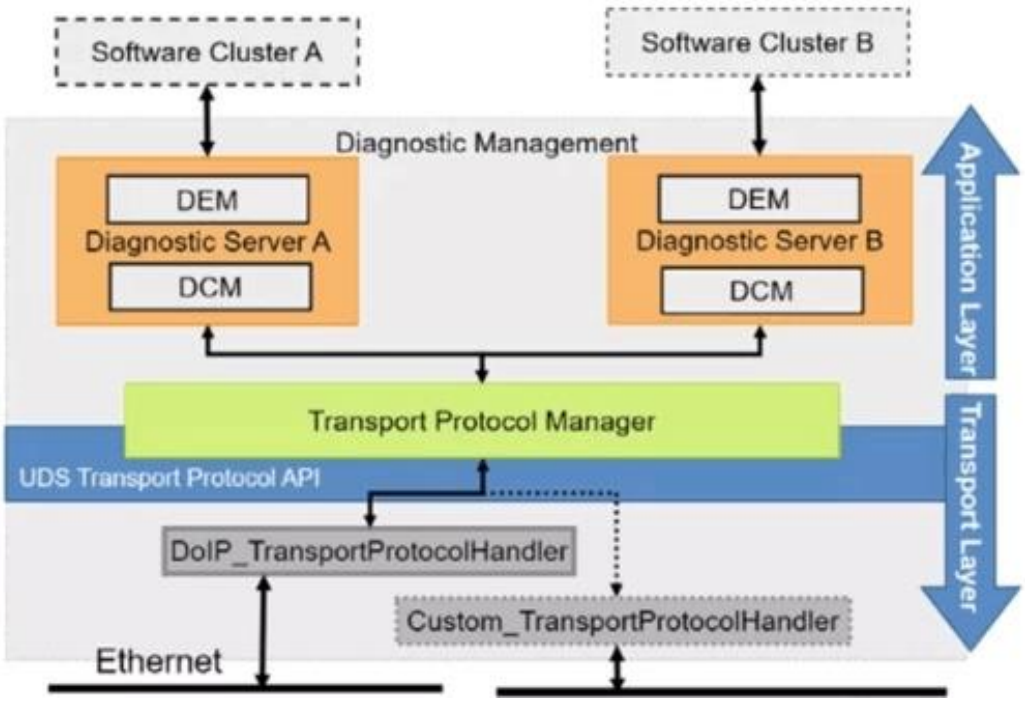


汽车诊断是指通过使用专用的诊断工具和设备来检测和分析汽车的故障和问题。技术人员使用诊断工具，与车辆的电子控制单元（ECU）进行通信，并读取存储在 ECU 中的故障码和实时数据，确定车辆中存在的故障或问题。

而在汽车内部，为了完成上述流程，使用了一系列模块来完成存储数据、数据通信与完成服务的操作，具体分为以下模块：

- DCM：Diagnostic Communication Manager，诊断通信管理。主要负责与外部诊断工具（例如诊断扫描仪）进行**通信**，以便读取和清除故障码，并**执行一些诊断任务**。
- DEM：Diagnostic Event Manager，诊断事件管理。用来**记录和存储诊断事件**的 DTC（诊断故障码），下面连接 NVRAM Manager，就是将这些诊断事件记录到 EEP 或者 Flash 中。
- FIM：Function Inhibition Manager，功能禁止管理。当一些 error 出现的时候，**禁止**一些**功能**。例如当检测到控制器过电流的时候，关闭 MOS 或 IGBT，防止炸管

基于上述组件，汽车内部诊断的大致流程如下：



① 故障检测：ECU 通过监测车辆各个系统和传感器的状态来检测潜在的故障。一旦检测到故障，DEM 会收到故障信息，生成故障码，然后存储在 NVRAM 内存中。

② 诊断请求：当车主或技术人员需要进行诊断时，可以通过诊断工具发送诊断请求。诊断诊断请求可以包括读取故障码、读取实时数据等。

③ 诊断通信：主要由 DCM 管理诊断数据流，与其他 ECU 进行通信。通常使用标准的诊断协议，如 UDS（Unified Diagnostic Services）或 ISO-TP（ISO Transport Protocol）。

④ 诊断服务执行：DCM 执行相应的诊断服务，例如诊断请求读取故障码、实时数据或执行其他诊断操作。

⑤ 诊断响应：DCM 将诊断结果通过诊断通信协议返回给诊断工具，诊断结果包括故障码、实时数据、状态信息等。

⑥ 诊断结果处理与故障修复：技术人员可以根据诊断结果进行故障分析，并采取相应的修复措施。

⑦ 清除故障码：在故障修复后，诊断工具可以发送清除故障码的请求，DEM 清除相关的故障码，以确认问题已经解决。

DCM、DEM 与 FIM 模块具体介绍

1.1 诊断通信管理模块 DCM

The Dcm module ensures diagnostic data flow and manages the diagnostic states, especially diagnostic sessions and security states. Furthermore, the Dcm module checks if the diagnostic service request is supported and if the service may be executed in the current session according to the diagnostic states.The Dcm module provides the OSI-Layers 5 to 7 of Table 1: Diagnostic protocols and OSI-Layer.

OSI-Layer	Protocols				
7	UDS-Protocol - ISO14229-1 [1]				Legislated OBD - ISO15031-5 [2]
6	-	-	-	-	-
5	ISO15765-3	-	-	-	ISO 15765-4
4	ISO15765-2	-	-	-	-
3	ISO15765-2	-	-	-	ISO 15765-4
2	CAN-Protocol	LIN-Protocol	FlexRay	MOST	ISO 15765-4
1	CAN-Protocol	LIN-Protocol	FlexRay	MOST	ISO 15765-4

Table 1.1: Diagnostic protocols and OSI-Layers

根据官方文档 https://www.AUTOSAR.org/fileadmin/standards/R2211/CP/AUTOSAR_ SWS _DiagnosticCommunicationManager.pdf 定义，DCM 模块**确保诊断数据流**并**管理诊断状态**，特别是诊断会话和安全状态。此外，DCM 模块根据诊断状态检查是否支持**诊断服务请求**，以及服务是否可以在当前会话中执行。而定义中提到的服务主要包括两个：UDS（Unified Diagnostic Services）和 OBD（On-Board Diagnostics）车载诊断服务，这是两种应用层协议，以 UDS 为例，其提供的服务如下：

大类	SID（0x）	诊断服务名	服务Service
诊断和通信管理功能单元	10	诊断会话控制	Diagnostic Session Control
	11	电控单元复位	ECU Reset
	27	安全访问	Security Access
	28	通讯控制	Communication Control
	3E	待机握手	Tester Present
	83	访问时间参数	Access Timing Parameter
	84	安全数据传输	Secured Data Transmission
	85	诊断故障码设置控制	Control DTC Setting
	86	事件响应	Response On Event
	87	链路控制	Link Control

数据传输功能单元	22	通过ID读数据	Read Data By Identifier
	23	通过地址读取内存	Read Memory By Address
	24	通过ID读比例数据	Read Scaling Data By Identifier
	2A	通过周期ID读取数据	Read Data By Periodic Identifier
	2C	动态定义标识符	Dynamically Define Data
	2E	通过ID写数据	Write Data By Identifier
	3D	通过地址写内存	Write Memory By Address
存储数据传输功能单元	14	清除诊断信息	Clear Diagnostic Information
	19	读取故障码信息	Read DTC Information
输入输出控制功能单元	2F	通过标识符控制输入输出	Input Output Control By Identifier
例行程序功能单元	31	例行程序控制	Routine Control
上传下载功能单元	34	请求下载	Request Download
	35	请求上传	Request Upload
	36	数据传输	Transfer Data
	37	请求退出传输	Request Transfer Exit
	38	请求文件传输	Request File Transfer

总的来说，DCM 完成数据通信与 UDS 和 OBD 诊断服务的实现。

DCM 模块的内部子结构

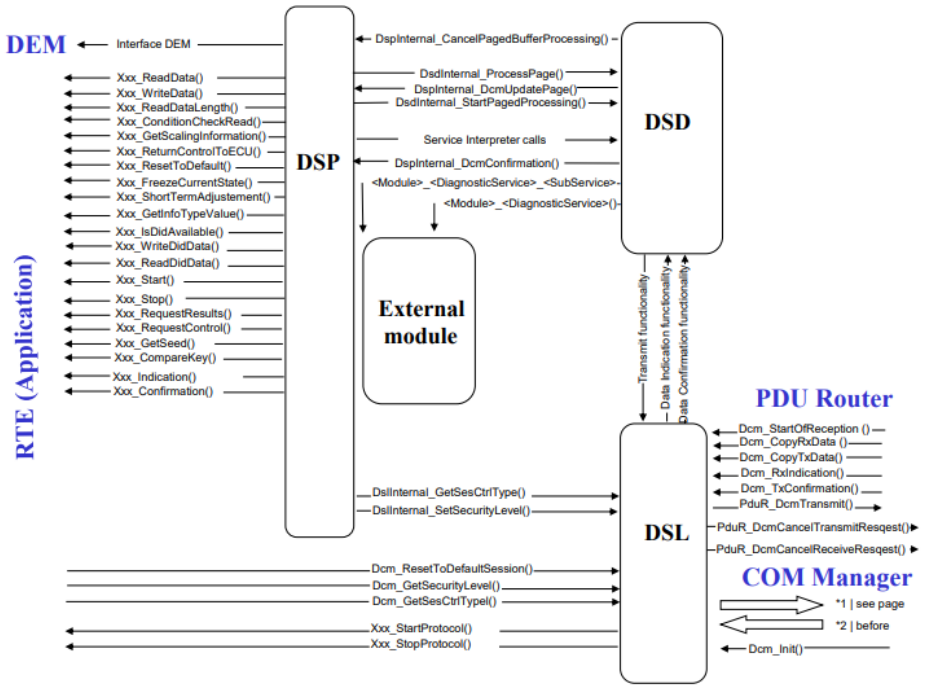


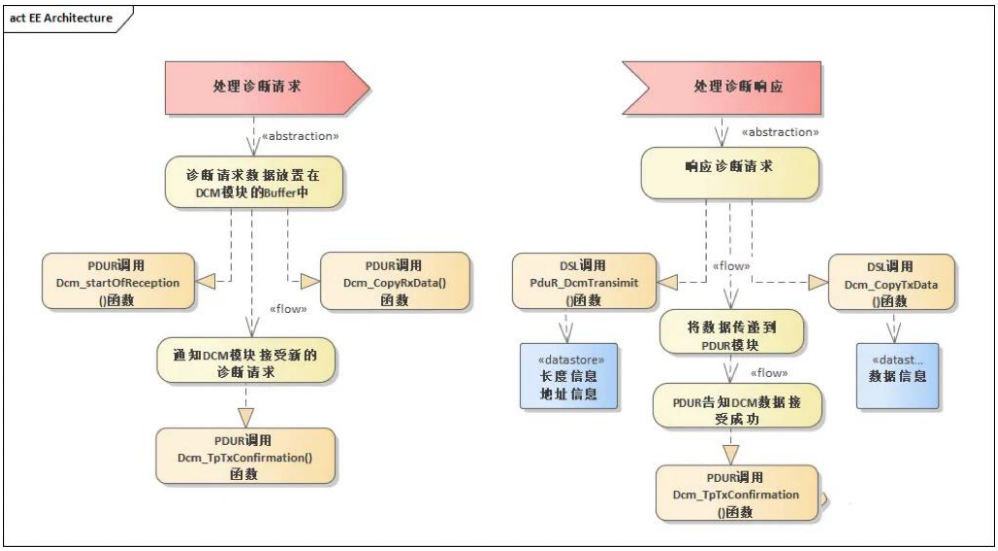
Figure 7.1: Possible interaction between the submodules in the DCM

DCM 模块可以分为四个子层，分别是 DSD（Diagnostic Session Dispatcher）、DSL（Diagnostic Service Layer）、DSP（Diagnostic Service Processor）和 DCL（Diagnostic Communication Layer）。在这个上下文中，DCM、DSD、DSL 和 DSP 之间的关系可以描述如下：

1) DSL ：诊断服务层。

该层处于 DCM 模块的最底层，用于处理诊断数据请求和响应的数据流；监控和确保诊断请求和响应的时序。它接收来自 DSD 层的诊断请求，并根据请求类型将其路由到相应的 DSP 子层服务。同时，DSL 也负责将来自 DSP 子层的诊断响应传输回 DSD 层。

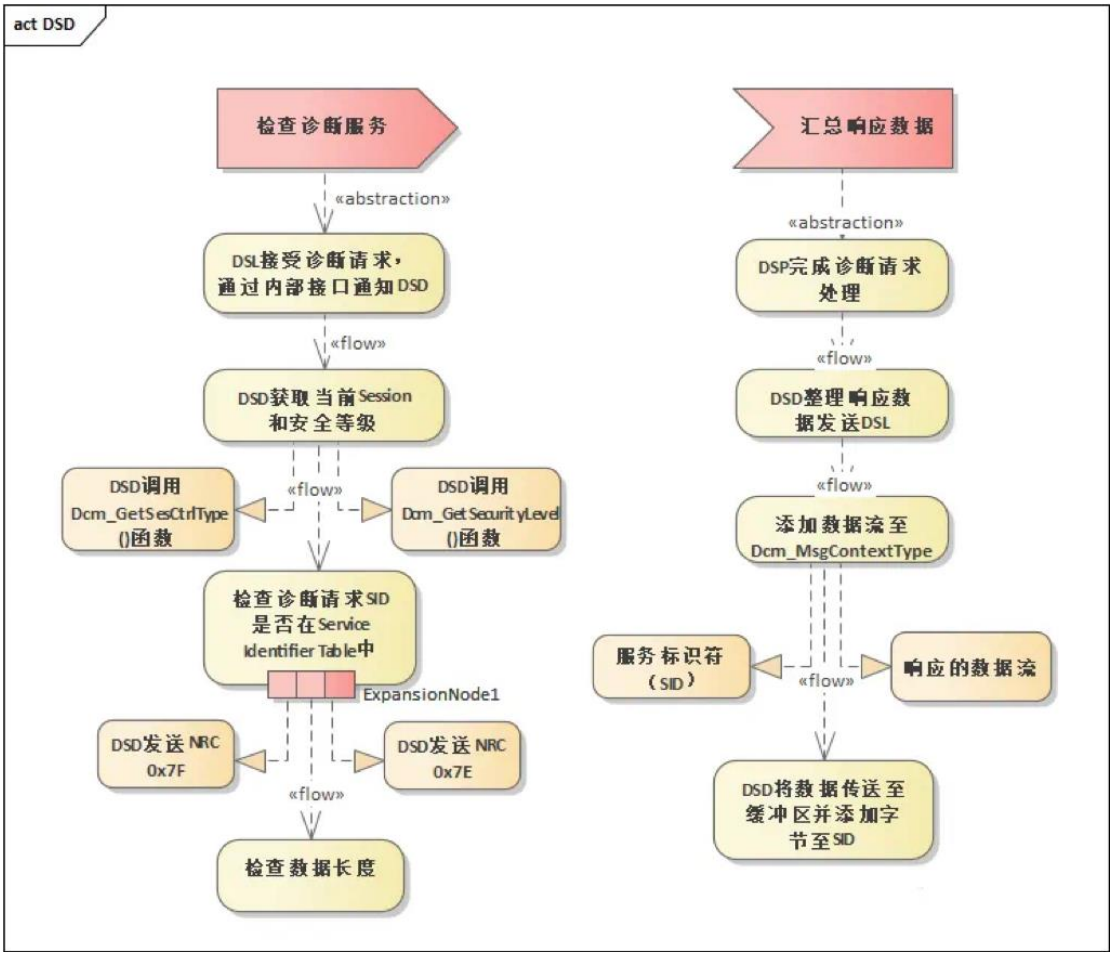
整个处理诊断请求及响应的过程如下：



2) DSD：诊断会话调度器。

处于中间层，这个子层主要负责管理诊断会话，如处理诊断会话切换、请求取消、会话超时等功能。此外，它还负责将来自 DCL 层的诊断请求转发到相应的 DSL 层服务。

当接收到新的诊断请求后转发到诊断服务器，完成诊断请求处理后转发诊断响应。



3) DSP：诊断服务处理器。

处于最上层，具体实施诊断服务处理，当接受到 DSD 请求处理诊断服务并转发诊断请求后，将完成实际的诊断服务功能响应及处理。它包含了处理不同诊断服务（如读取故障码、控制执行、数据参数 ID 请求等）所需的功能。每个具体的诊断服务都可以看作是一个独立的 DSP 子层。

1.2 诊断事件管理 DEM

DEM 模块可以**接收**来自各种传感器和控制器的**诊断信息**，然后根据故障严重程度进行分类和记录，并提供诊断状态和故障码等信息。

此外，DEM 还提供了一些 API（应用程序接口），用于**访问和修改诊断数据**。例如，可以使用 API 来清除已诊断的故障码或设置故障码的优先级。DEM 还提供了诊断通信协议和诊断存储库，以便与其他系统进行通信和记录诊断数据。

DCM 和 DEM 之间的通信链路主要包括以下组件：

1) DCM 提供的 API：DCM 提供了一些 API，用于从 DEM 中读取和更新诊断数据，例如读取故障码和清除故障码等。
2) DEM 提供的 API：DEM 也提供了一些 API，用于向 DCM 提供诊断信息，例如故障码、诊断状态和诊断数据等。
3) DCM-DEM 通信协议：DCM 和 DEM 之间的通信需要使用一些标准化的通信协议，例如 UDS（Unified Diagnostic Services）协议和 ISO 14229 标准。
4) 诊断存储库：DCM 和 DEM 需要共享一些诊断数据，例如故障码和诊断状态等，这些数据通常存储在诊断存储库中，DCM 和 DEM 可以通过这个存储库来交换数据。

1.3 功能抑制管理 FIM

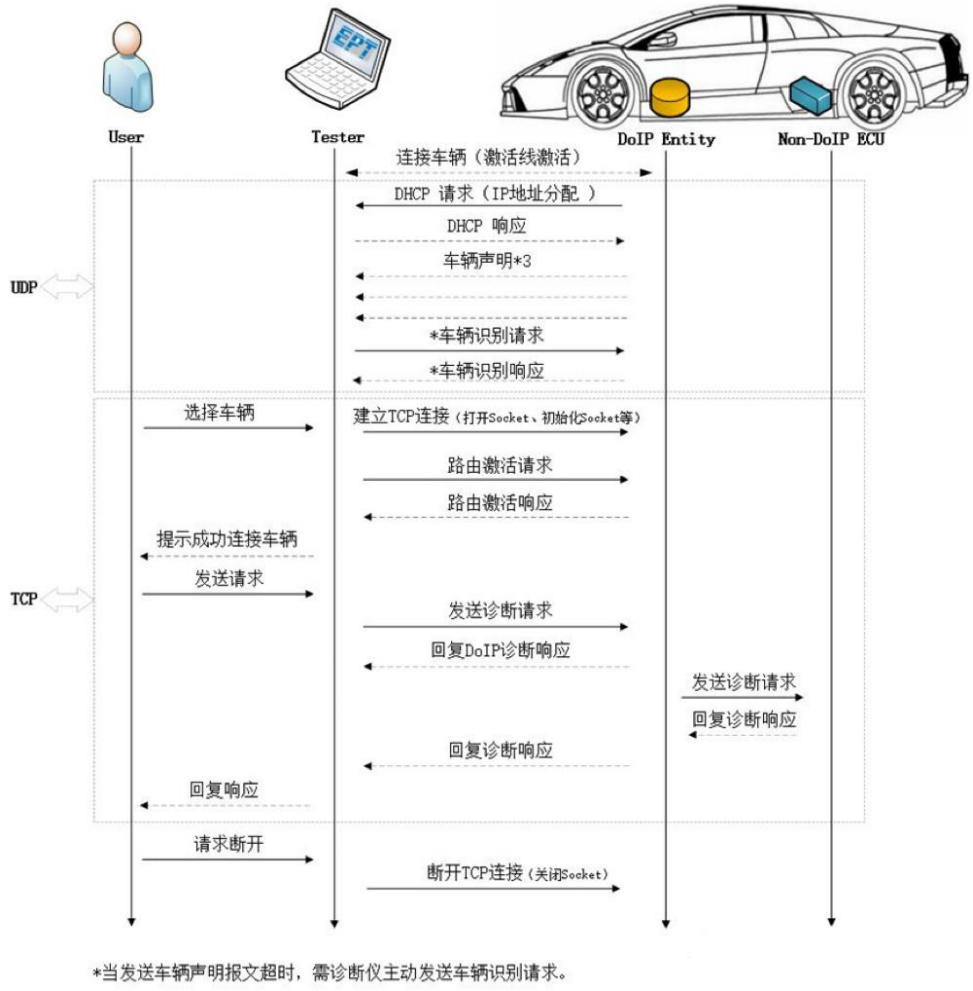
FIM 实际是一种软件组件，用于实现对应功能的抑制管理。功能抑制是指在车辆故障或安全问题出现时，对某些汽车功能进行限制或禁用的操作，以保证车辆和乘客的安全。FIM 组件通过 AUTOSAR 的标准化软件接口（FIM API）与其他软件组件（例如 ECU、Sensor 和 Actuator）进行通信，以检测和响应车辆故障或安全问题，并执行相应的功能抑制措施。

2. DoIP 协议

传统的是基于 CAN 总线的，技术人员用诊断仪器通过 CAN 连接线连接到汽车上导出数据。当前可以通过以太网，使用 wifi 或 5G 网络连接汽车进行数据的传递，而实现数据传输的协议就是 DoIP（Diagnostic communication over Internet Protocol,DoIP）协议。

DoIP 完成了所有和网络相关的任务，包括 DHCP 分配 IP 的过程，网络会话管理、网络路由等。DoIP 模块解析完 DoIP 报文后，**剥离出来 UDS 报文**再给 DCM 处理。通过 DoIP，诊断数据可以通过 IP 网络进行高速传输和远程访问，提供了更灵活和便捷的诊断方式。

DoIP 的时序如下：



3. Wireshark 的搭建

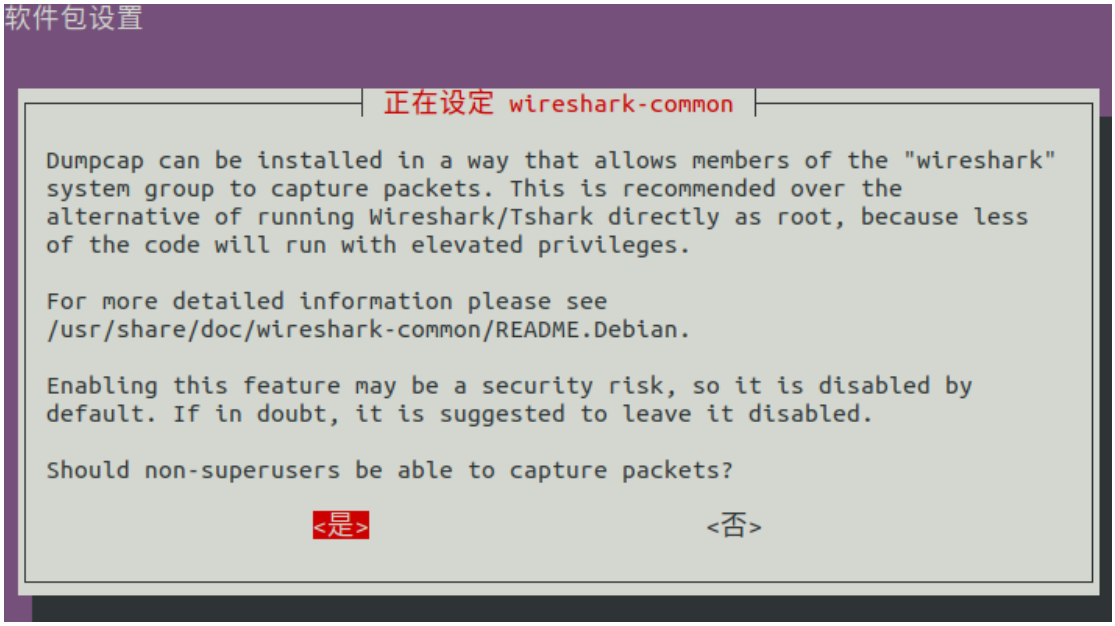
1) 安装 Wireshark

使用如下命令安装 Wireshark

```
1. sudo add-apt-repository universe
2. sudo apt install wireshark
```

2) 设置配置

在安装的过程中会弹出以下窗口，选择“是”



或在之后通过命令

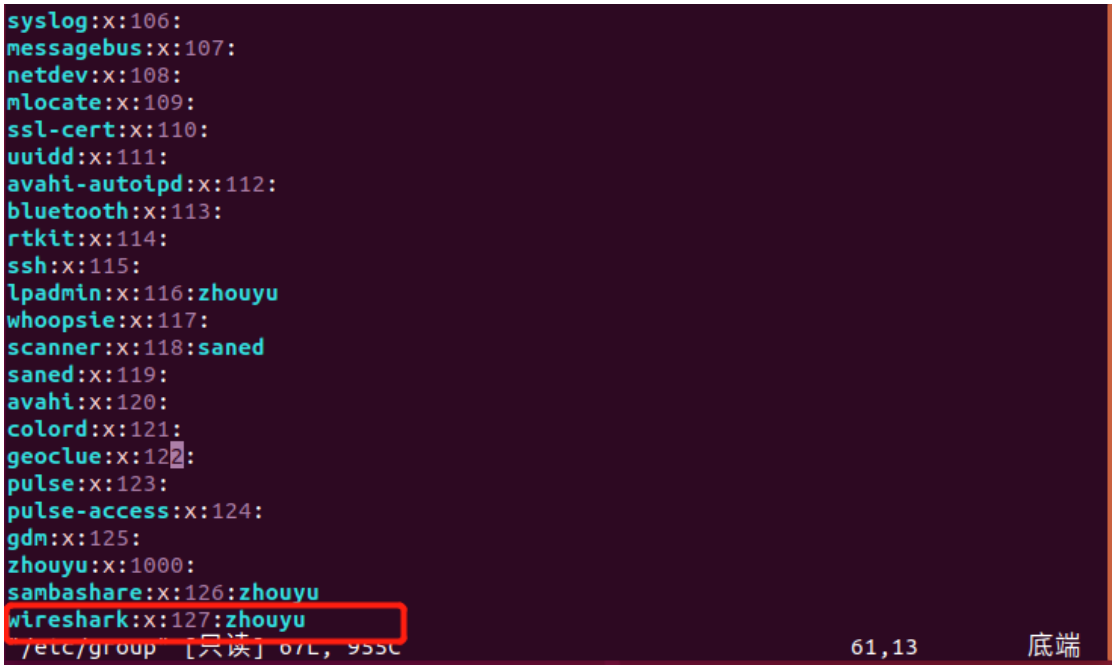
```
sudo dpkg-reconfigure wireshark-common
```

3) 修改 group 文件

使用命令

```
sudo vim /etc/group
```

打开 group 文件，在 wireshark 后加上自己的用户名。修改完成后保存并重启虚拟机。



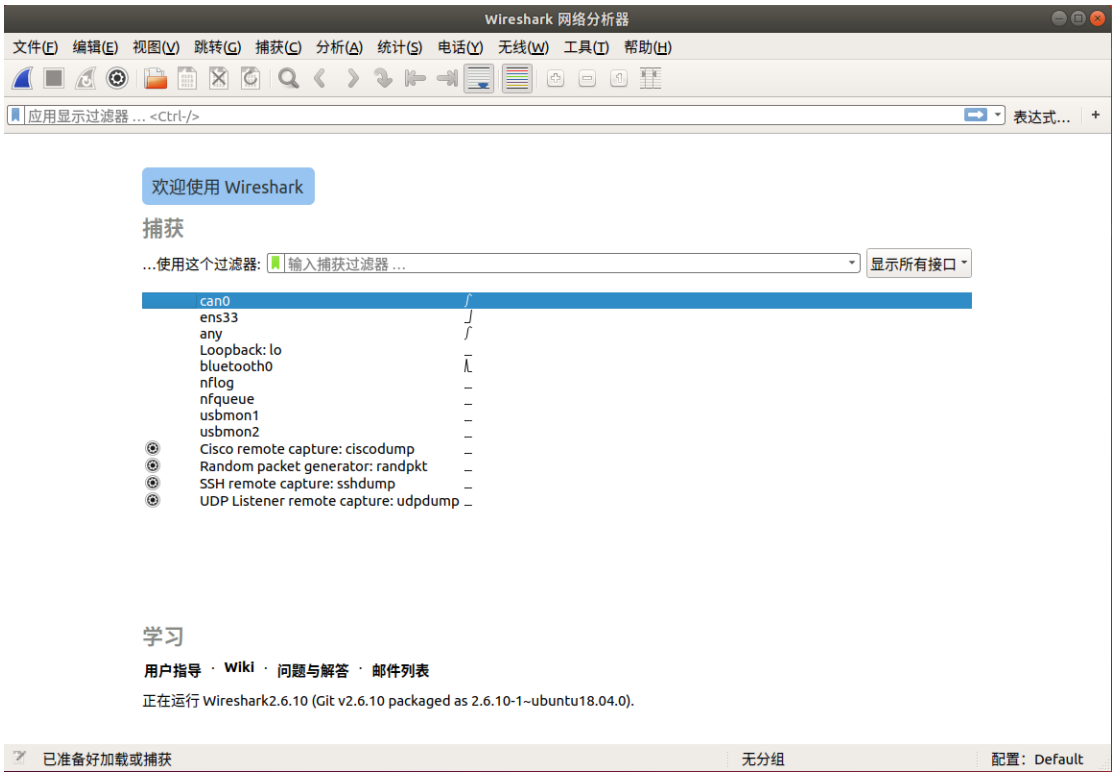
4) 启动 Wireshark

使用命令启动 Wireshark

```
Wireshark
```

启动后可以看到 wireshark 中有许多可监听端口，我们选择 any 端口或 qemu 虚拟出的

can0 端口启动开始抓包。



4. 发送路由激活和服务请求报文

通过以下命令，即可配置 Qemu 环境并发送 DoIP 报文

```
cd as/
./autorun.sh

cd socket_tool/
make
sudo ./send_client.sh
```

发送后的结果如下：(记得加 sudo 权限)

```
zhouyu@zhouyu-virtual-machine:~/as/socket_tool$ ./send_client.sh
DoIP:port=13400, id=0
DoIP:send packet to 172.18.0.200 13400 tcp
send begin
02 fd 00 05 00 00 00 07 be ef da 00 00 00 00
send end, send_num = 15
exit!!!

DoIP:port=13400, id=1
DoIP:send packet to 172.18.0.200 13400 tcp
send begin
02 fd 80 01 00 00 00 06 be ef fe ed 11 01
send end, send_num = 14
exit!!!
```

提示：若连接不成功的，输入 sudo apt install net-tools 指令修复网络接口

5. 报文结构分析

DHCP 过程的代码如下，配置地址为 172.18.0.200，目的端口号为 13400，代码存放在 com/as.application/common/config/SoAd_Cfg.c 中

```
{   /* for DCM */
```

```
2.      .SocketId = 0,
3.      .SocketLocalIpAddress = "172.18.0.200",
4.      .SocketLocalPort = 13400,
5.      .SocketProtocol = SOAD_SOCKET_PROT_TCP,
6.      .AUTOSARConnectorType = SOAD_AUTOSAR_CONNECTOR_DOIP,
7.  },
```

DoIP 数据是应用层协议，其发送是基于 TCP 协议，其报文格式如下图

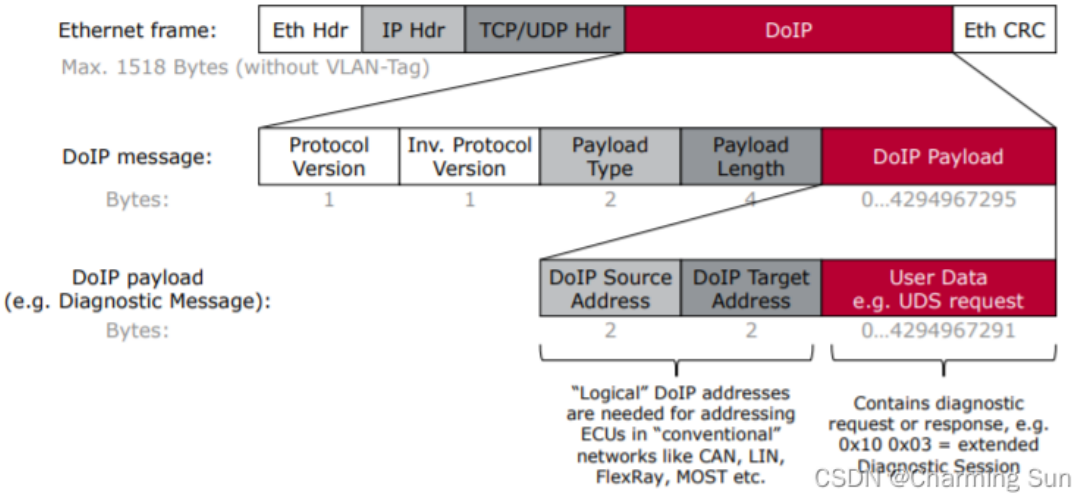
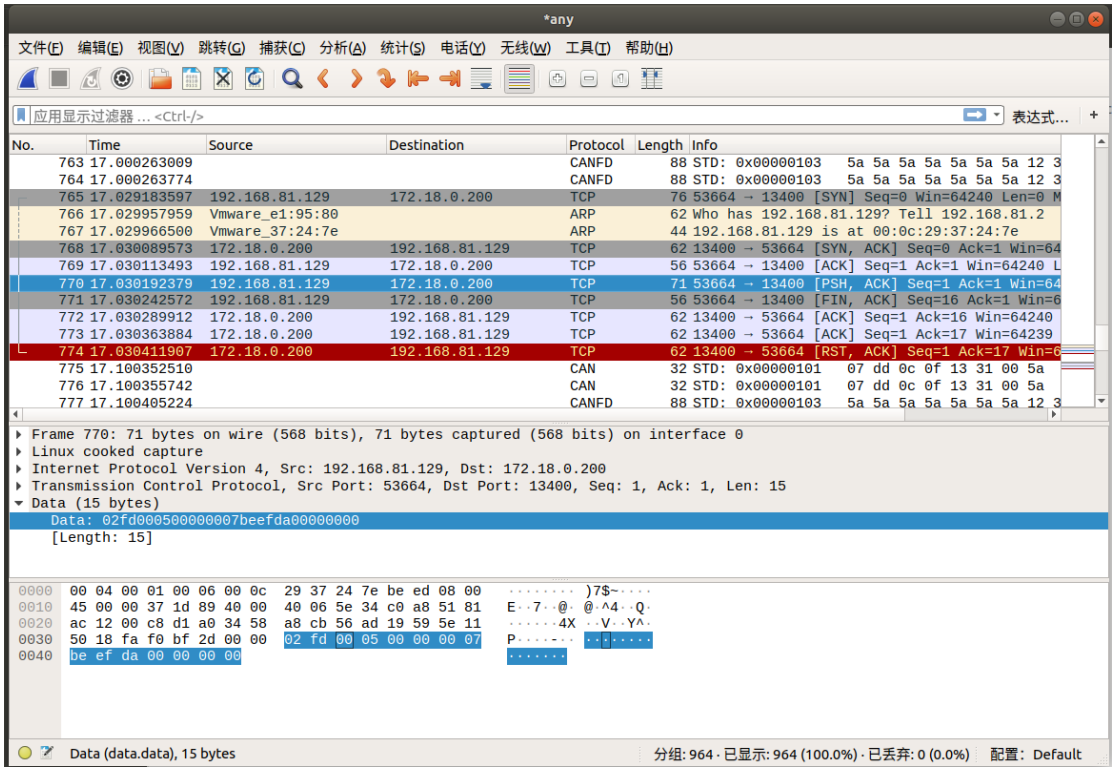


Table 11 — Generic DoIP header structure

Item	Pos.	Len.	Description	Values
Generic DoIP header synchronization pattern				
Protocol version	0	1	Identifies the protocol version of DoIP packets.	0x00: reserved 0x01: DoIP ISO/DIS 13400-2:2010 0x02: DoIP ISO 13400-2:2012 0x03…0xFE: reserved by this part of ISO 13400 0xFF: default value for vehicle identification request messages
Inverse protocol version	1	1	Contains the bit-wise inverse value of the protocol version, which is used in conjunction with the DoIP protocol version as a protocol verification pattern to ensure that a correctly formatted DoIP message is received.	Equals the <Protocol_Version> XOR 0xFF (e.g. 0xFE for protocol version 0x01).
Generic DoIP header payload type and payload length				
Payload type (GH_PT)	2	2	Contains information about how to interpret the data following the generic DoIP header (e.g. gateway command, diagnostic message, etc.)	See Table 12 for a complete list of currently specified payload type values.
Payload length (GH_PL)	4	4	Contains the length of the DoIP message payload in bytes (i.e. excluding the generic DoIP header bytes). Some payload types do not require any additional parameters (payload length is 0), some require a fixed DoIP message length while others allow for dynamic length DoIP messages.	0…4 294 967 295 bytes (= <d>)
Payload type specific message content	8	…	The payload type specific message content starts here. NOTE This implies that, for example, byte position 0 of the payload type-specific part of the message (see 7.1.1) means byte position 8 in the context of the overall DoIP message.	

- Protocol Version：版本号（1B）
- Inverse Protocol Version：Protocol Version 的取反值（1B）
- Payload Type：Data Type,，指出报文的类型，包括路由激活与诊断请求报文等（2B）
- Payload Length：数据长度（4B）
- Payload Type Specific Message Content：根据不同的 payload type 而有特定含义

5.1 路 由 激 活

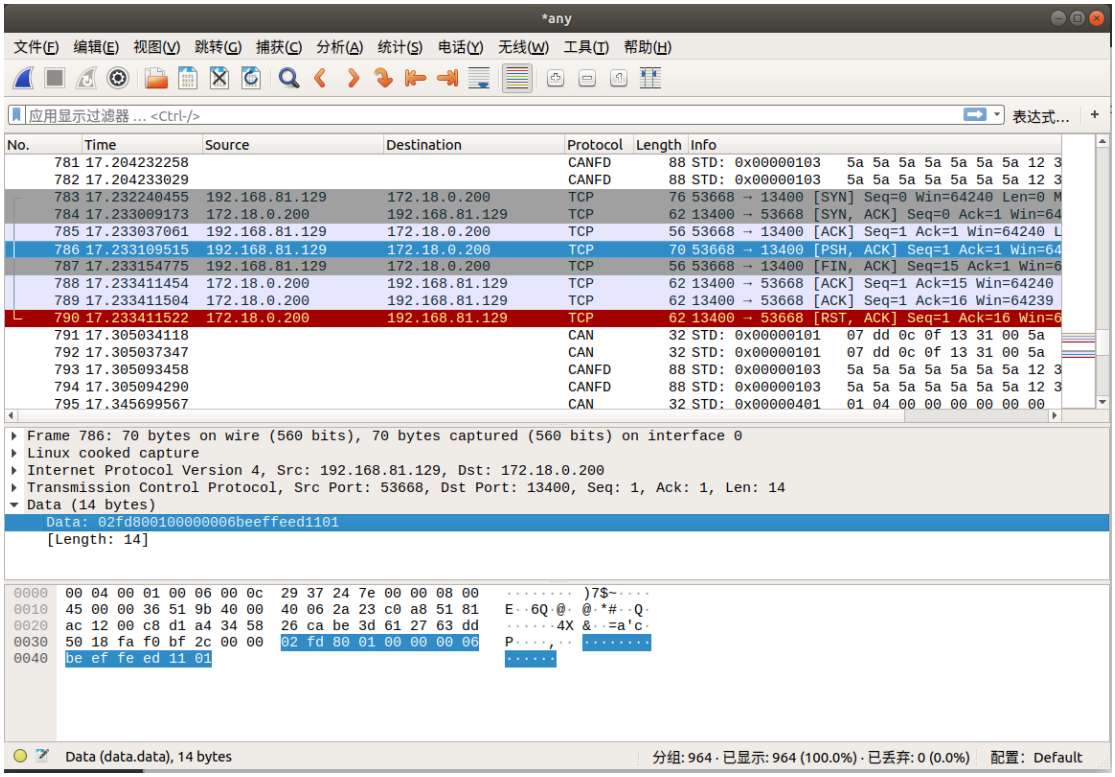


我们发送的报文为

02 fd 00 05 00 00 00 07 be ef da 00 00 00 00

可以看到这条 DoIP 报文基于 TCP，目的端口是 13400，从 192.168.81.129 发给 172.18.0.200，功能是 0x0005 路由激活，源地址是 0xbeef ，激活类型是 0xda。

5.2 诊断请求报文



我们发送的报文为
02 fd 80 01 00 00 00 06 be ef fe ed 11 01
可以看到这条报文是DoIP报文的基础上又封装了UDS报文。DoIP报文的功能类型是0x8001诊断请求，源地址是 0xbeef，目的地址是 0xfeed。

实验报告

完成实验步骤，查看源码，用自己的话给出 UDS 故障诊断协议的概念、时序定义，分析上述故障诊断各模块代码文件的内部结构，捕获并给出诊断请求报文的目标地址与端口号，附上最终运行结果截图。

报告以“**Automobile_lab2_学号_姓名**”格式命名，每人 1 份，提交至研究生信息系统，截至时间为 2024 年 6 月 20 日。