

EB tresos车载以太网培训



Elektrobit

2020.09.10



什么是车载以太网

- 什么是以太网?

全球使用最广泛的**局域网**技术，是一种有线网络技术

- 什么是车载以太网?

用以太网连接车内电子单元的**新型局域网**技术



笔记本电脑上的以太网线
——非屏蔽双绞线+RJ45接口

BroadR.Reach

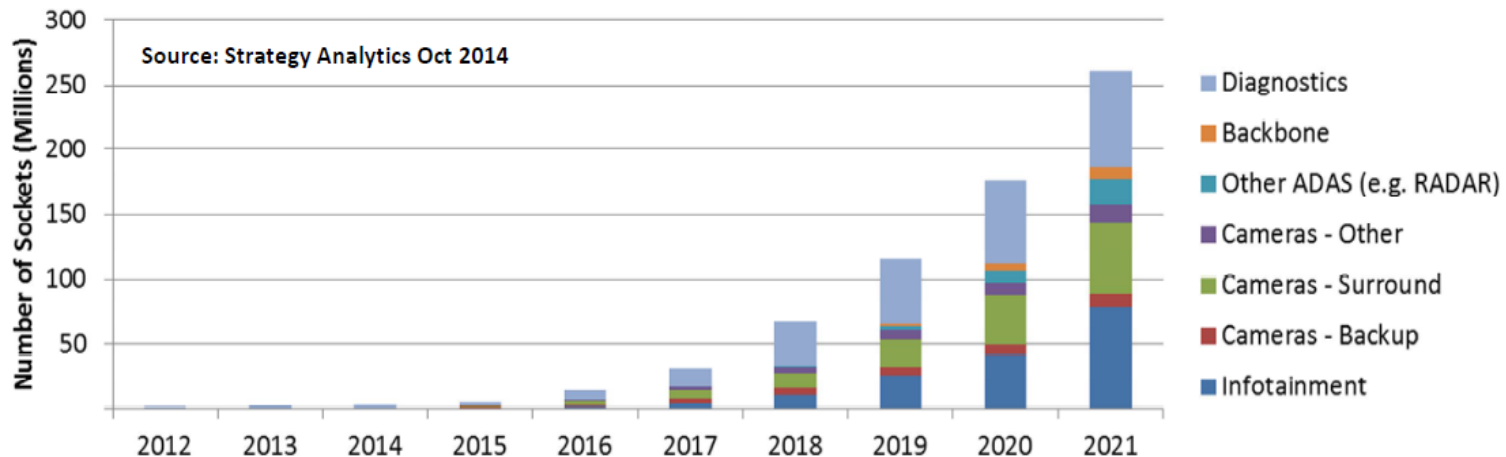
100BASE-T1

- 他们的区别?

可靠性更强、更低电磁辐射、更低功耗、更低延迟以及更高的同步实时性

100Mbps

为什么使用车载以太网



策略分析: 2018: 65,000,000 Ethernet-Sockets, 2021: >250,000,000

推动车用以太网发展

- 高品质车载娱乐(Infotainment)
- 高级驾驶辅助系统(ADAS)
- 车用诊断远程升级(Diagnostics)

以太网与传统CAN总线的区别

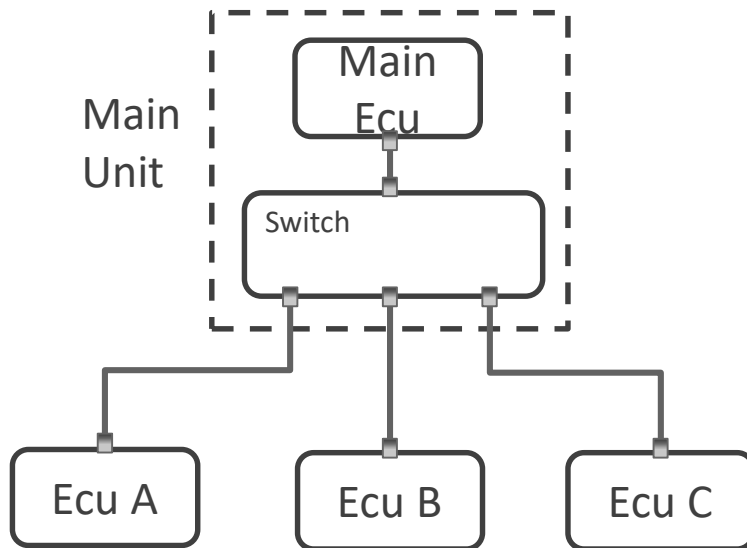


Elektrobit

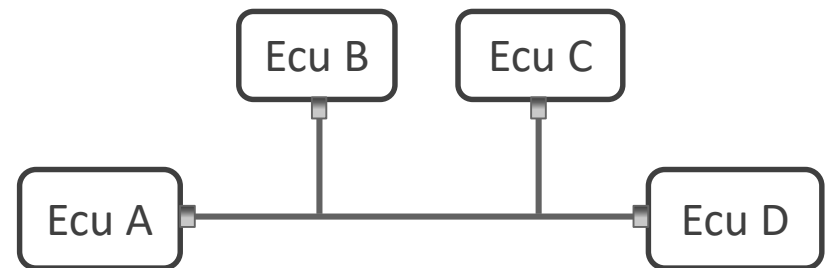


以太网 vs. CAN

车载以太网



CAN

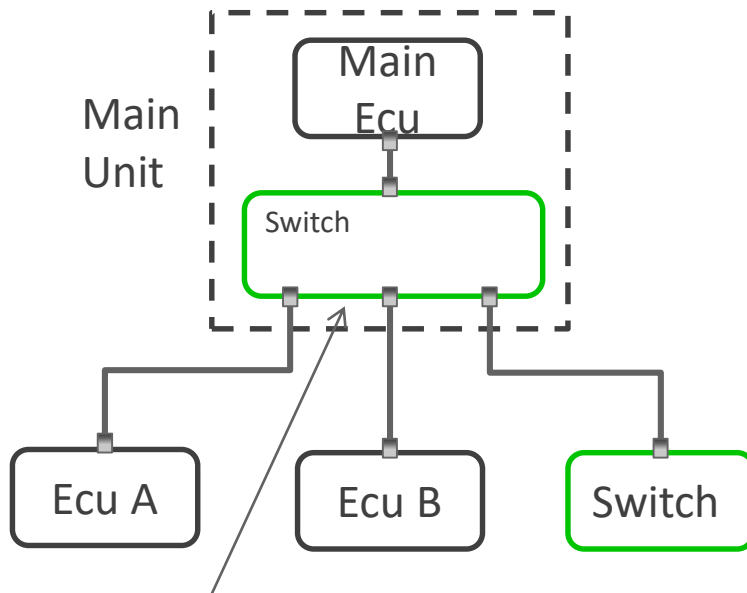


为什么以太网和传统汽车CAN网络如此不同？

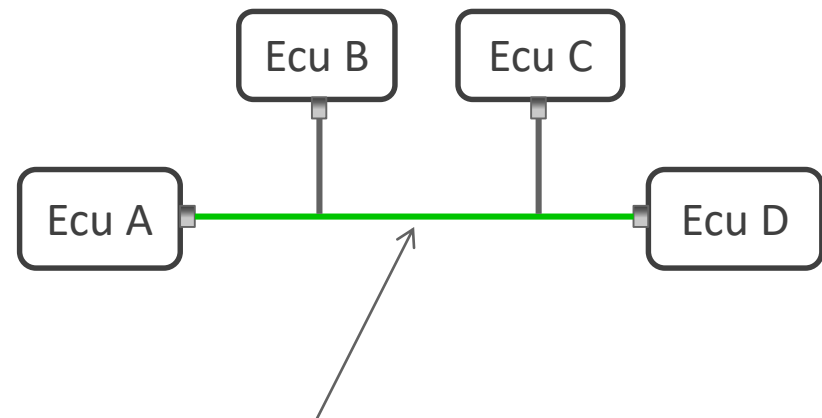
以太网 vs. CAN: 网络类型

车载以太网

CAN



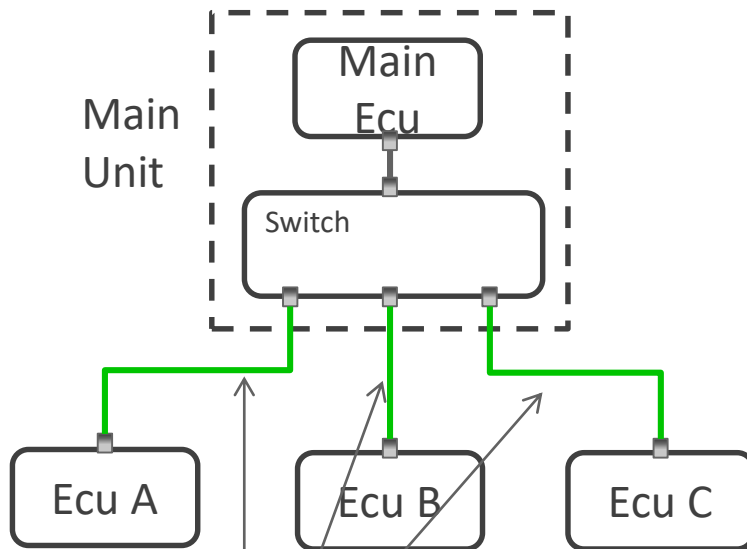
交换机网络
Switched network
(星型、树型拓扑)



总线网络 Bus network
(线性拓扑)

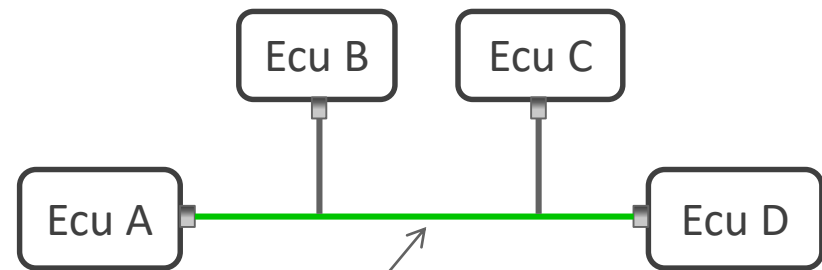
以太网 vs. CAN: 媒介访问控制

车载以太网



Point-to-point 点对点连接
网段中没有冲突产生

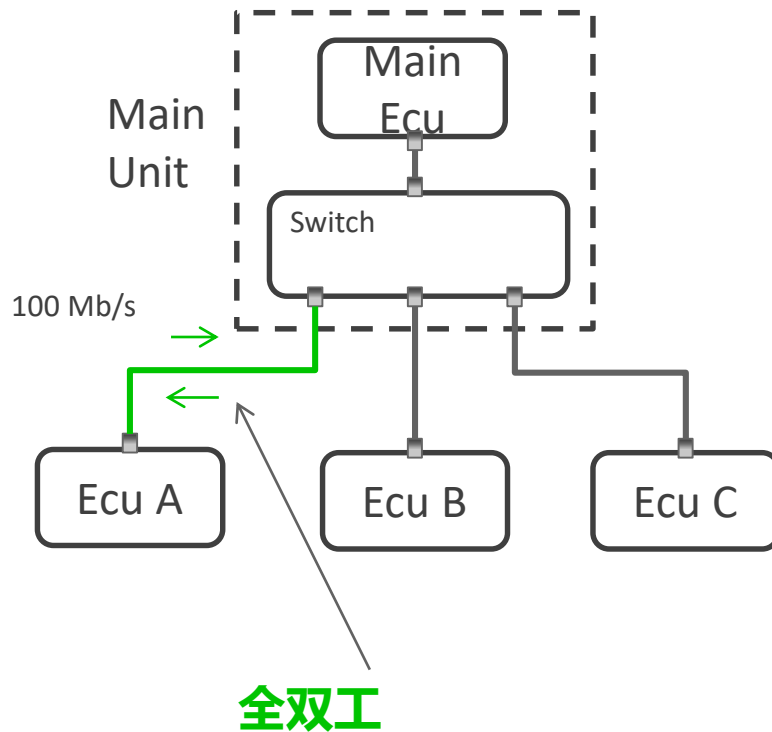
CAN



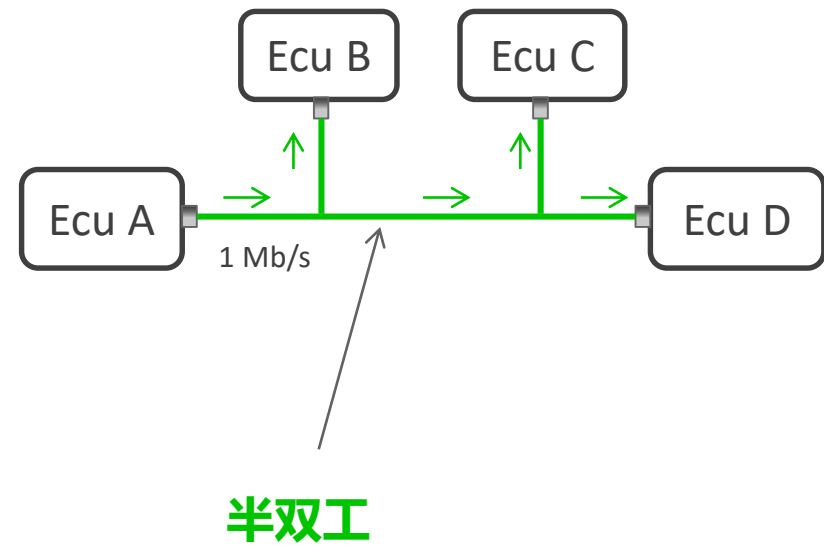
公用媒介的访问冲突需要仲裁机制

以太网 vs. CAN: 数据发送服务

车载以太网

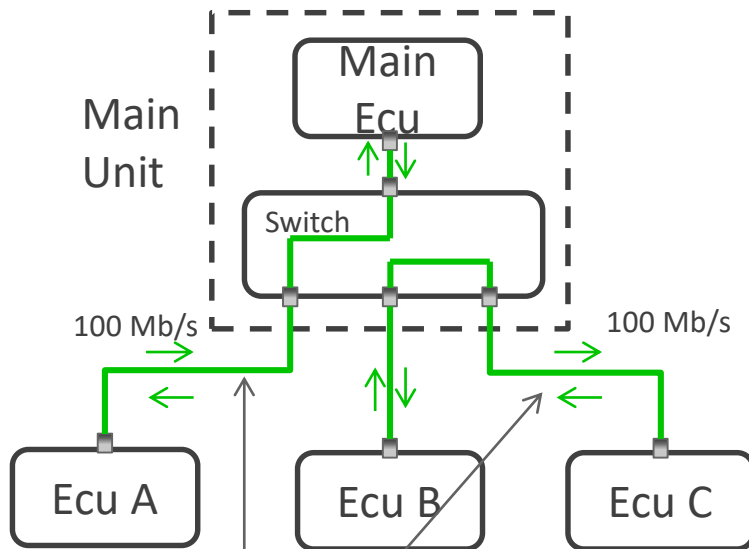


CAN



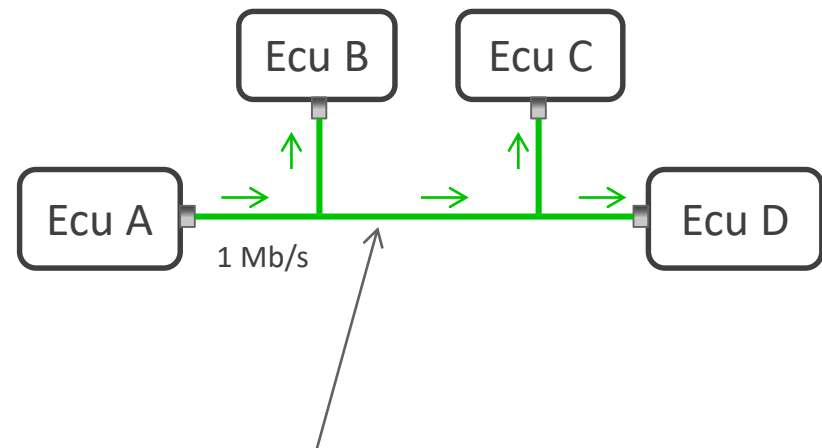
以太网 vs. CAN: 吞吐量

车载以太网



支持数个同时发送的数据流

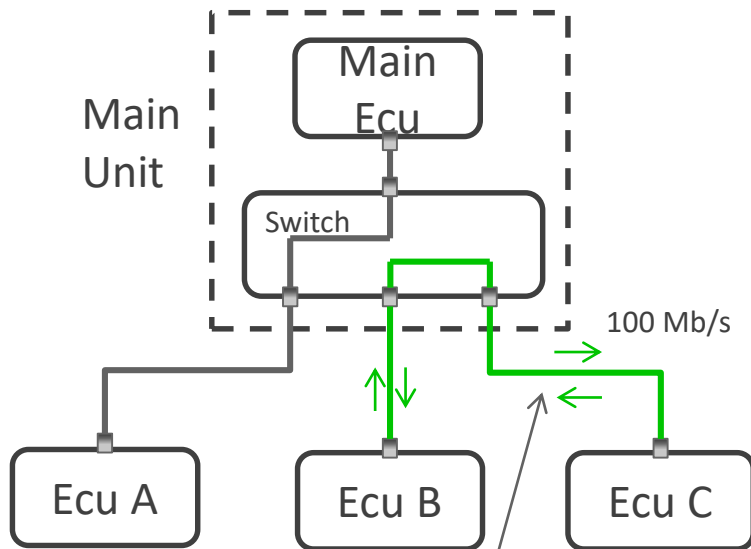
CAN



通讯需要依次进行

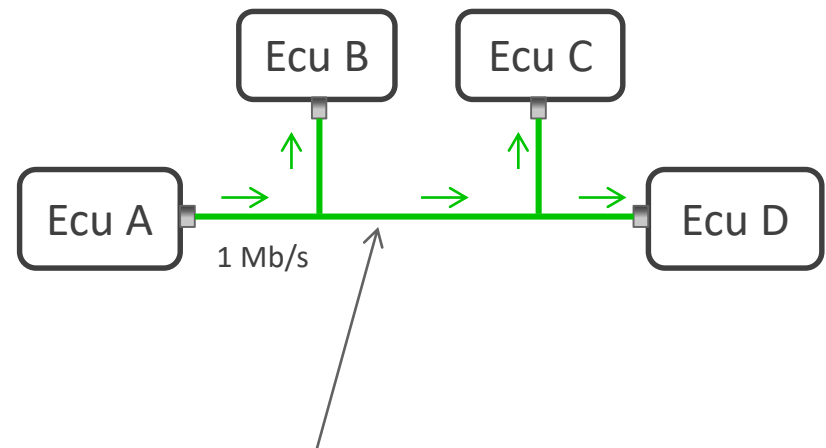
以太网 vs. CAN: 寻址

车载以太网



以太网报文包含发送方和接收方的物理地址

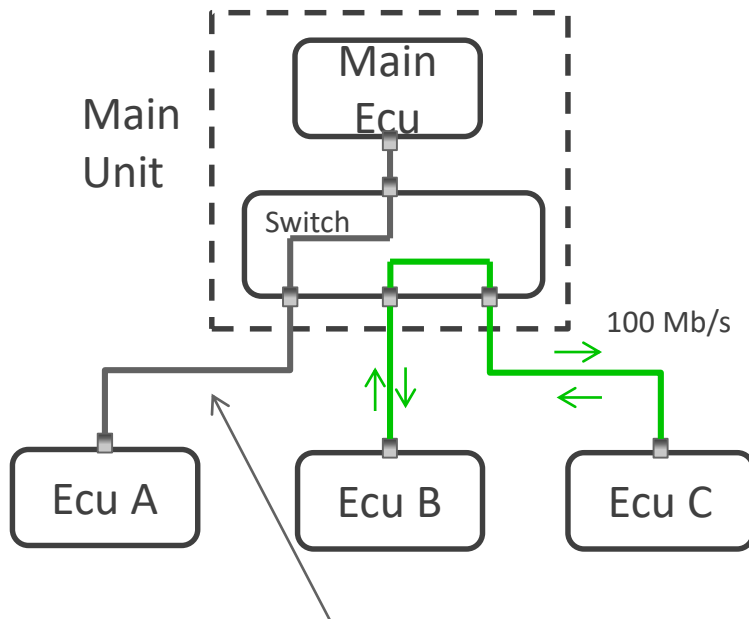
CAN



CAN报文只包含报文ID CAN节点不具备物理地址

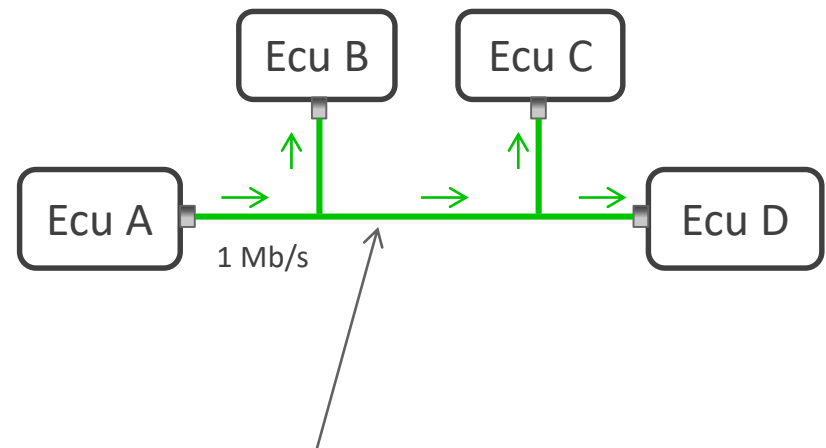
以太网 vs. CAN: 网络监听

车载以太网



此刻该网络段不会收到任何报文

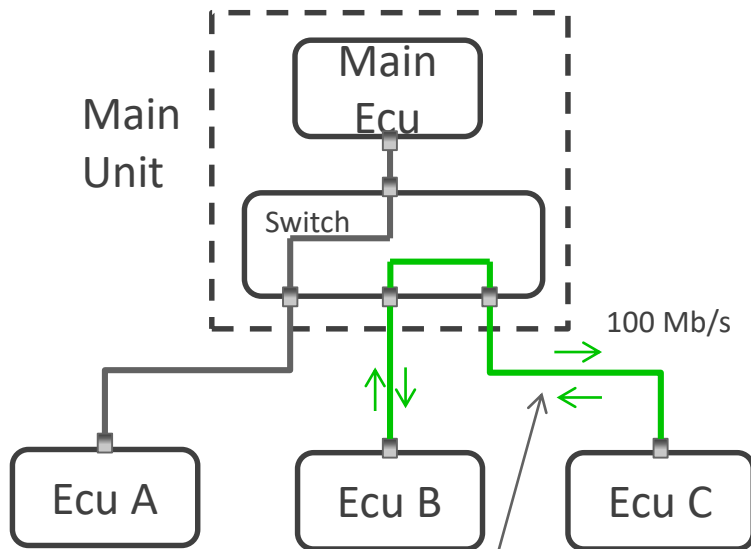
CAN



广播通讯，能被所有节点接收

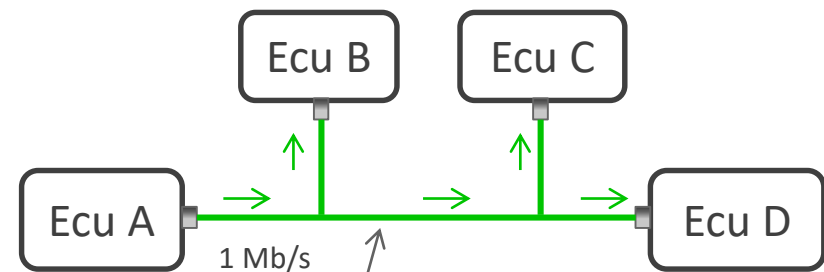
以太网 vs. CAN: 基于服务 (Service-based)

车载以太网



基于服务的通讯, 只有当某节点发送了服务请求, 通讯才会产生

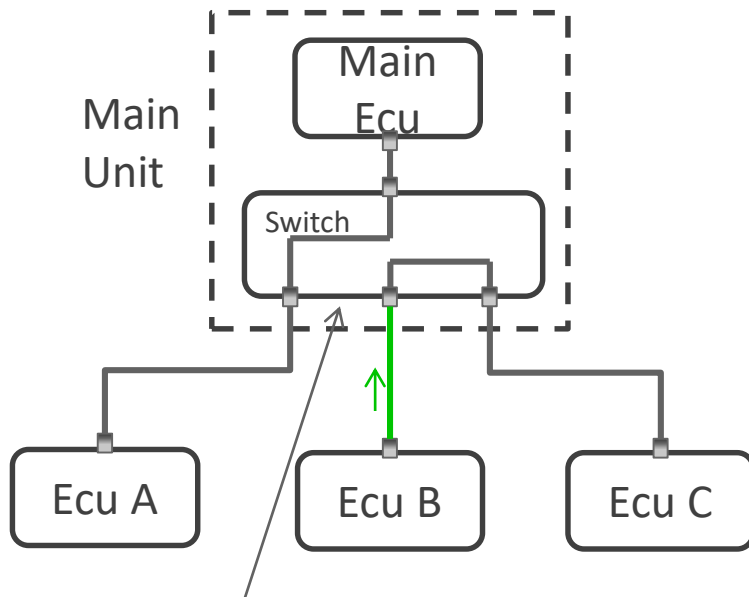
CAN



主动通讯, 报文的发送与服务请求无关

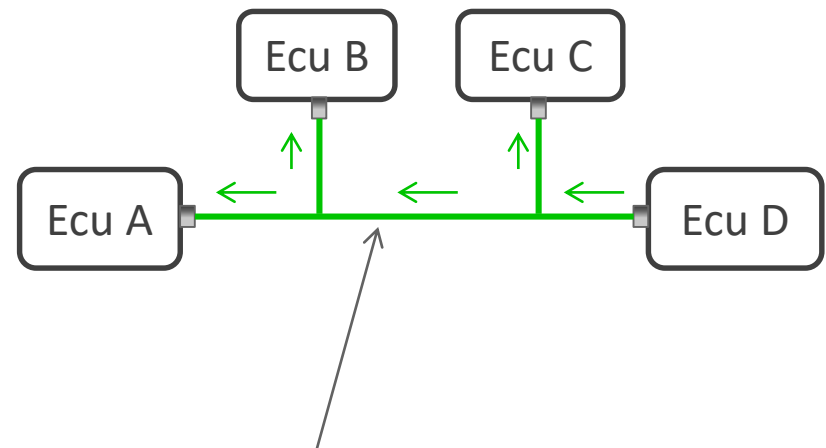
以太网 vs. CAN: 错误帧

车载以太网



包含错误的报文会被交换机或节点直接丢弃

CAN



当节点发现错误会在总线上发送错误帧

以太网标准化组织

- **IEEE 802** Develop LAN/MAN (active since 1980)
 - 定义了数据链路层和物理层
 - <http://standards.ieee.org/about/get/>
 - 802.1 局域网高层标准工作组(TSN, ...)
 - e.g. 802.1AS (TimeSync)
 - 802.3 以太网工作组 (standards for the Ethernet network)
- **IETF** (国际互联网工程任务组)
 - 在RFC (<https://www.ietf.org/rfc.html>) 网站上发布和维护技术标准
 - 例如. RFC#791 (IP协议), RFC#768 (UDP协议), ...

车载以太网标准化组织

- **AUTOSAR**

- 车用以太网模块 (eth stack)



- **AVnu Alliance**

- 推广AVB/TSN技术



- **OPEN Alliance (One-Pair Ether-Net) Special Interest Group**

- OPEN Alliance是非盈利性的汽车行业和技术联盟
- 旨在鼓励大规模使用以太网作为车联网标准
- BroadR - Reach 以太网技术

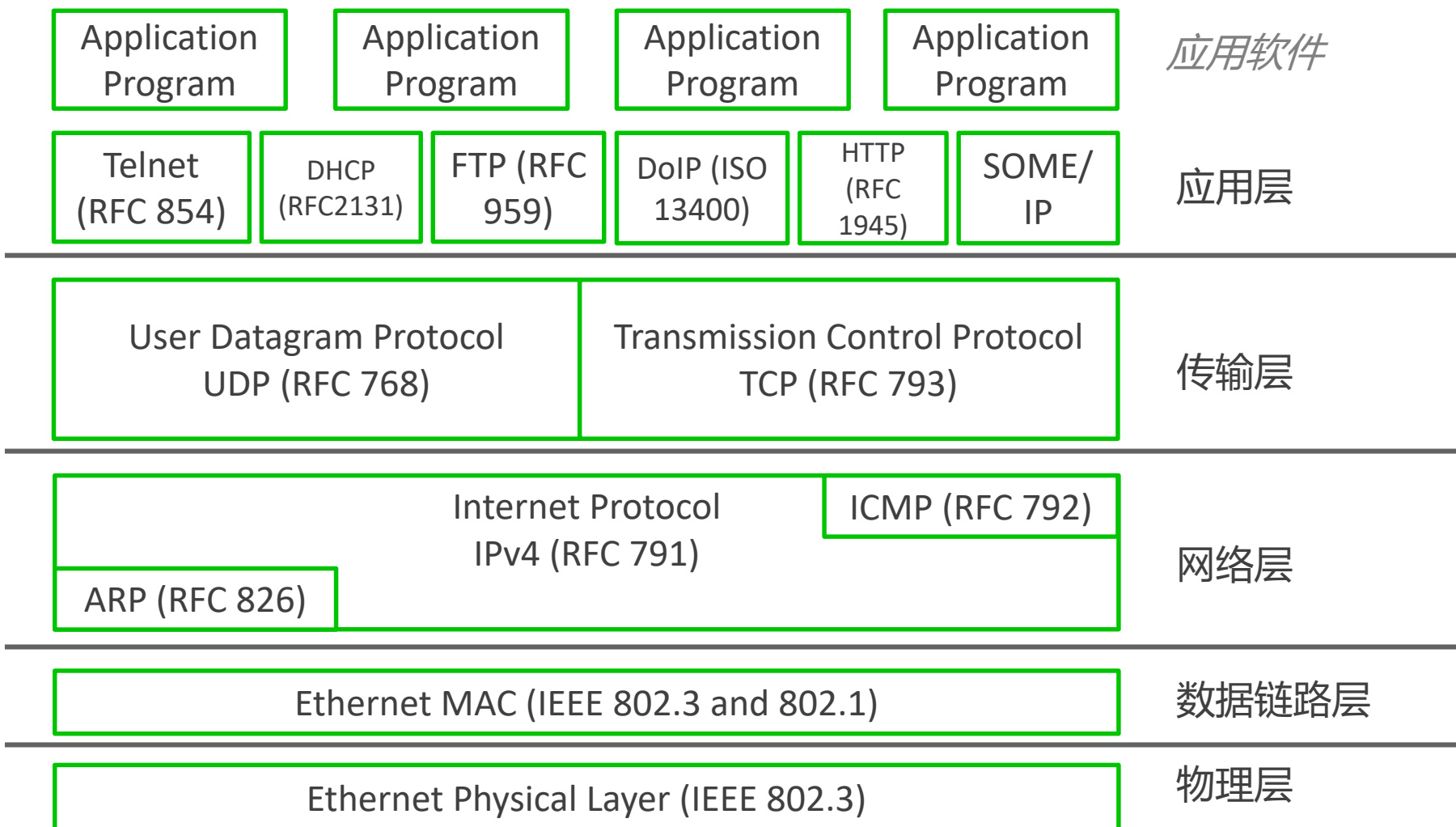


AUTOSAR IP/Ethernet概念

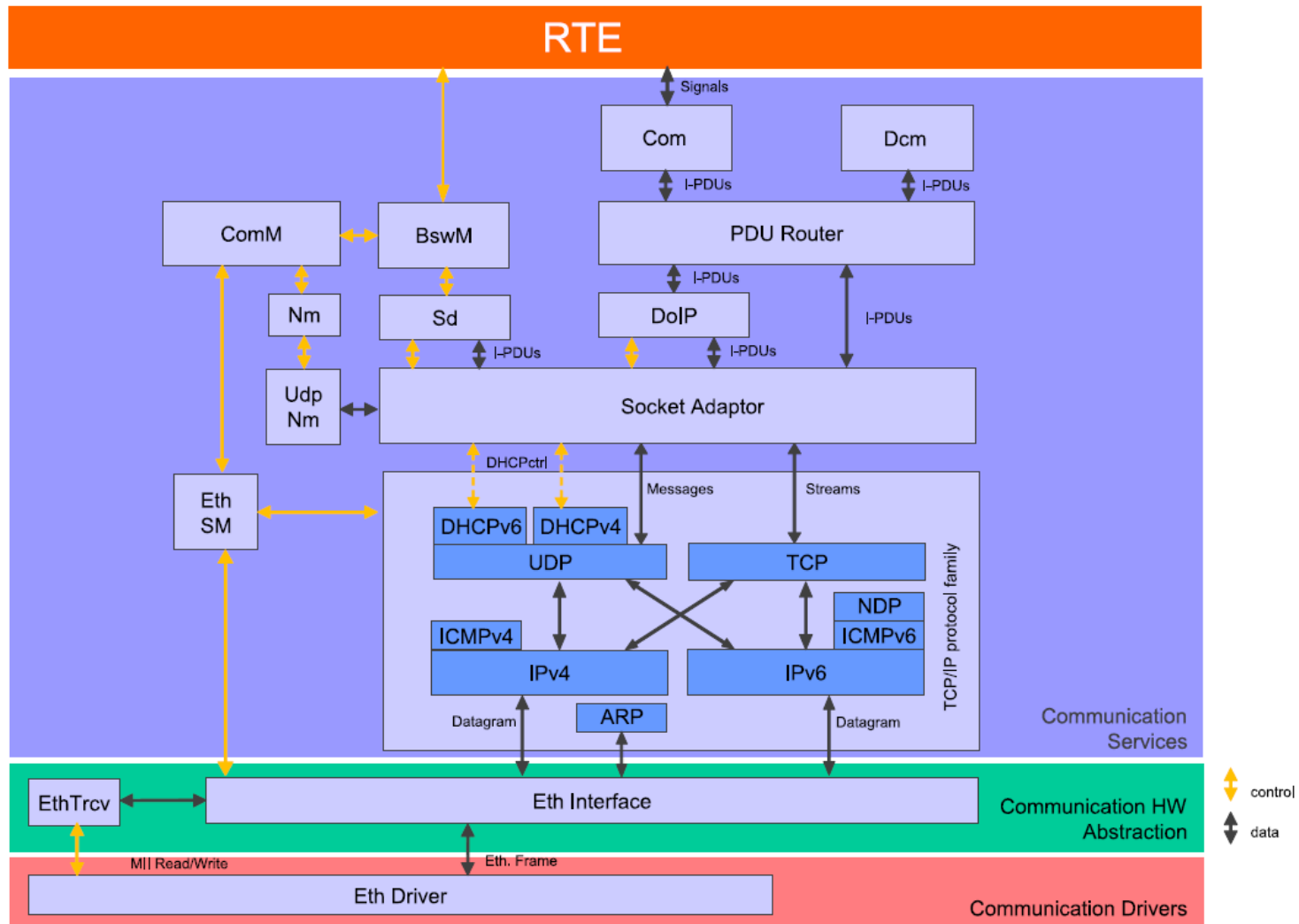
- 以太网协议层级和AUTOSAR以太网模块总览
- 以太网控制器基本硬件原理
- AUTOSAR – Ethernet Driver
- AUTOSAR – Ethernet Transceiver and Switch
- AUTOSAR – Ethernet Interface
- AUTOSAR – TcpIp
- AUTOSAR – SoAd
- AUTOSAR – Ethernet StateManager



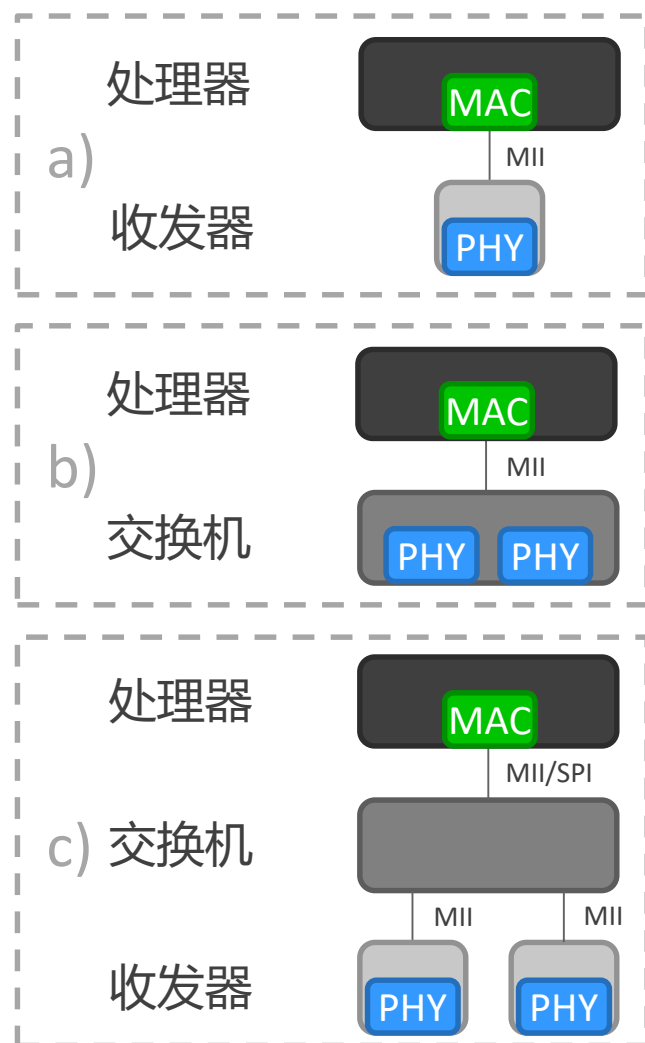
车载以太网协议层级概览



AUTOSAR以太网协议栈总览

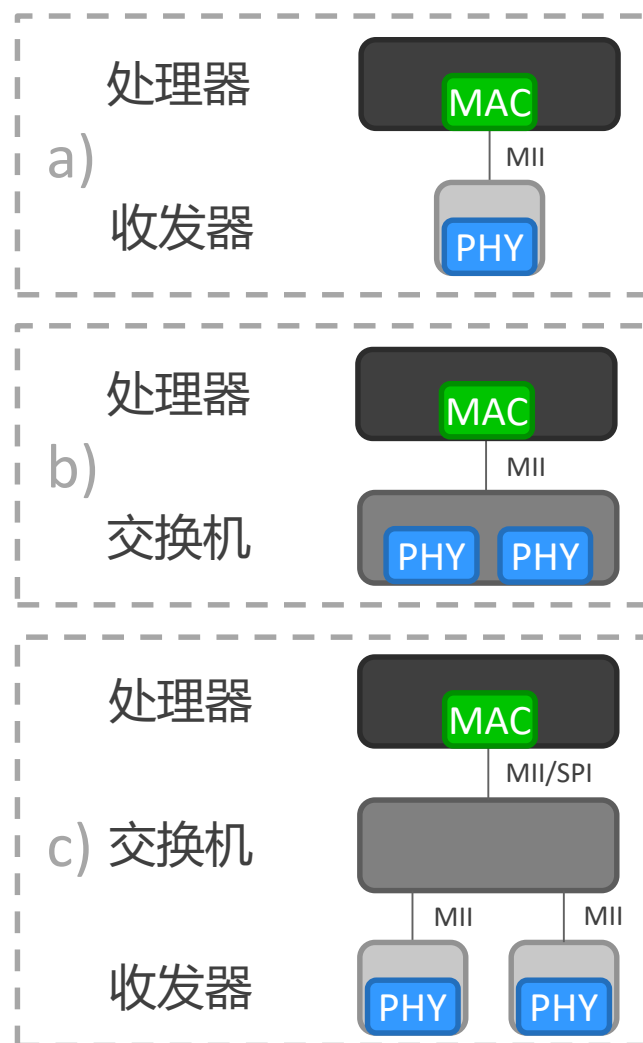


以太网控制器基本硬件原理

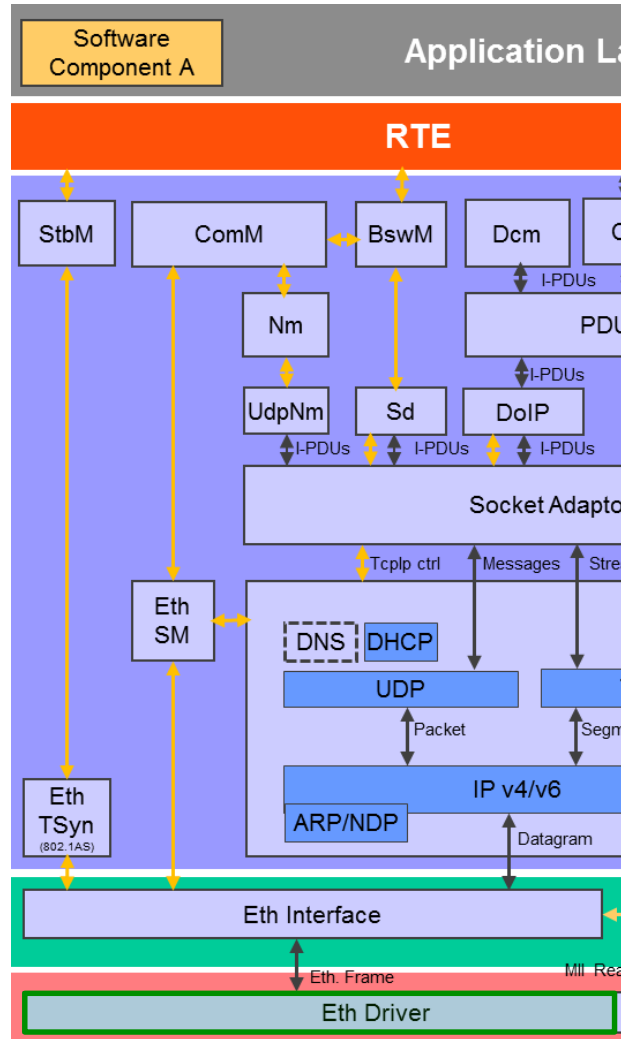


- MAC – “Media Access Control”
 - 芯片上的以太网控制器核心
 - **发送、接收**以太网帧
- MII – “Media-Independent Interface”
 - 一种**MAC和PHY通讯**的**标准接口**
 - 同样接口还有RMII, GMII等
- PHY – “Physical Layer”
 - 以太网收发器
 - 将MAC发送的数据**编码、解码**成相应的物理层特性 (例如. FastEthernet, BroadRReach, WLAN, ..) 传输到如光纤或铜缆线等物理媒介上。

以太网控制器基本硬件连接



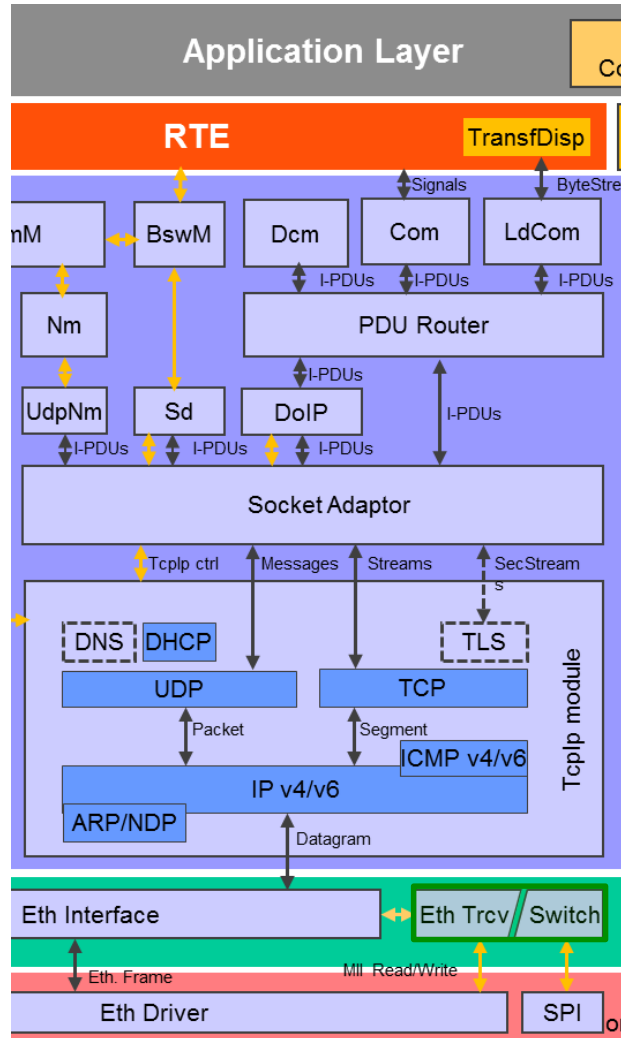
AUTOSAR模块 – Ethernet Driver



- **基于硬件**属于MCAL层
- 抽象出一个以太网的控制器的硬件参数
- 提供给上层**独立于硬件的模块API接口**
 - 初始化接口
 - 配置工作模式接口
 - 数据传输接口
 - 等等

AUTOSAR标准文档: AUTOSAR_SWS_EthernetDriver.pdf

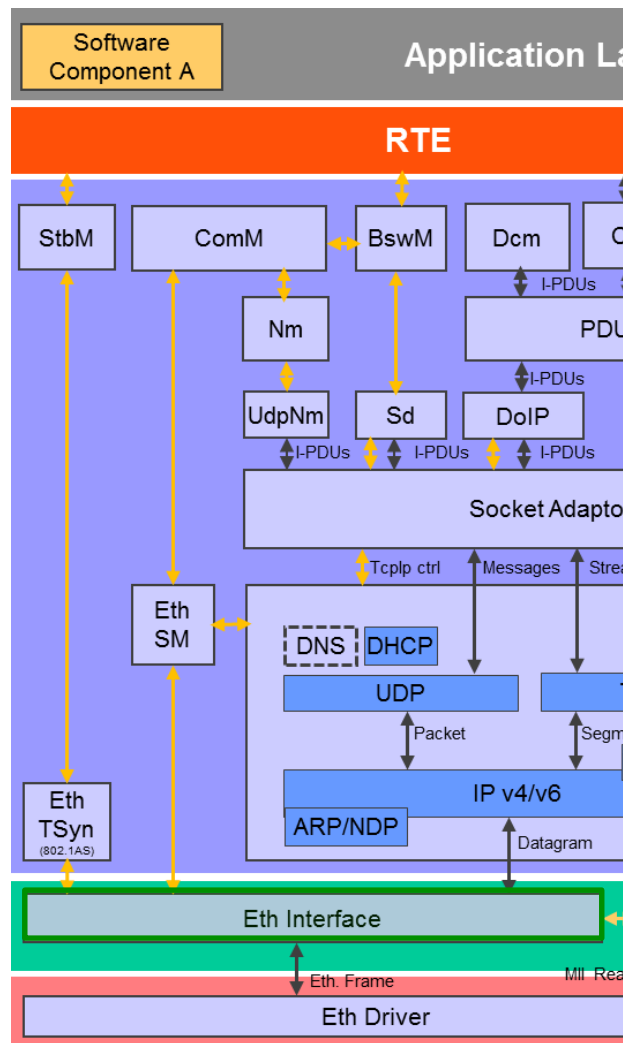
AUTOSAR模块 – Ethernet Transceiver and Switch



- 主要功能
 - 使能连接
 - 配置连接速度, 双工模式, 等
 - 连接状态监测
- 以太网收发器 - EthTrcv
 - 连接单个PHY
- 以太网交换机 - EthSwt
 - 连接多个PHY
- 可使用SPI来配置硬件

AUTOSAR标准文档: [AUTOSAR_SWS_EthernetTransceiverDriver.pdf](#),
[AUTOSAR_SWS_EthernetSwitchDriver.pdf](#)

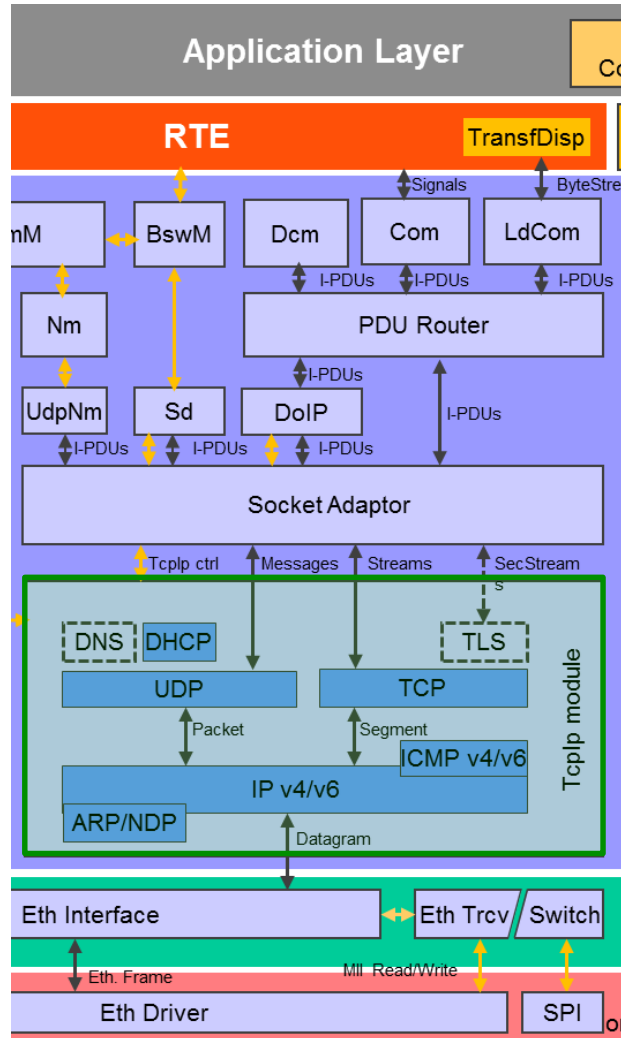
AUTOSAR模块 – Ethernet Interface



- 属于ECU抽象层
- 独立于硬件的以太网的抽象界面
 - 控制以太网控制器，VLAN，以太网收发器和交换机
 - 对于不同芯片厂的以太网控制器的访问接口进行了统一
 - 使上层应用与ECU硬件相剥离

AUTOSAR标准文档: AUTOSAR_SWS_EthernetInterface.pdf

AUTOSAR模块 – Tcplp

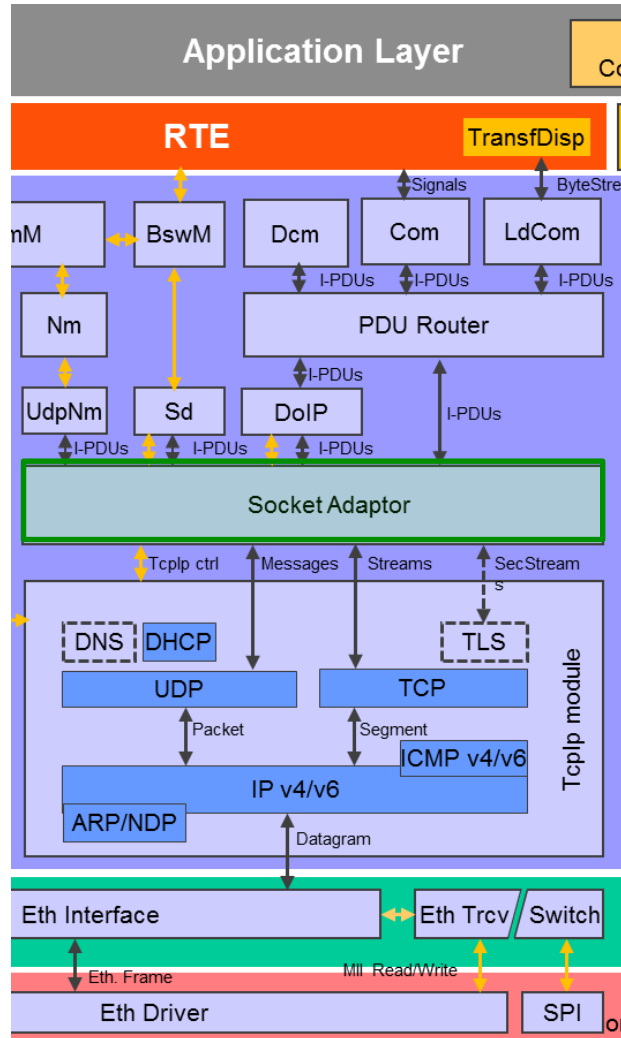


- **Tcplp模块**包含以下协议

- 应用层: **DHCP** (Dynamic Host Configuration Protocol)
- 传输层: **TCP**(面向连接的、可靠传输), **UDP**(无连接, 不可靠性)
- 网络层:
 - **IP**(包含IPv4和IPv6)
 - **ICMP**(Internet Control Message Protocol, 比如.可使用 Ping功能)
 - **ARP**(Address Resolution Protocol) and **NDP**(Neighbor Discovery Protocol)

AUTOSAR标准文档: AUTOSAR_SWS_Tcplp.pdf

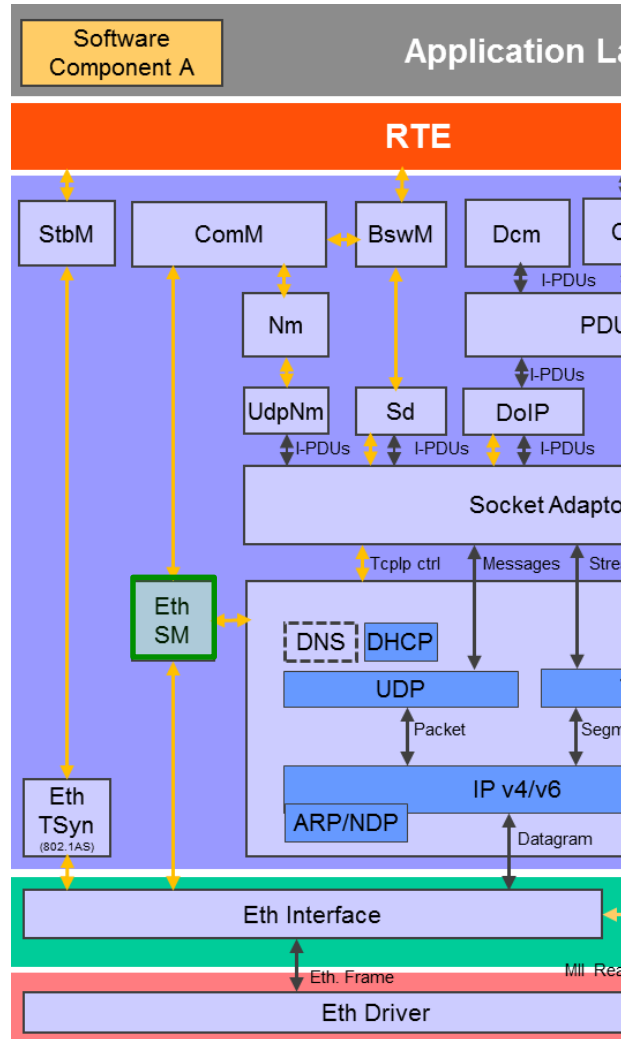
AUTOSAR模块 - SoAd



- SoAd的上层模块基于PDU通讯
- SoAd的下层模块基于socket通讯
- SoAd是一个接口模块：
 - 映射AUTOSAR I-PDUs到TCP或UDP sockets
 - 为Socket配置远程IP地址，远程和本地端口号
 - 多种上层模块：UdpNm，Sd，DoIP

AUTOSAR标准文档: AUTOSAR_SWS_SocketAdaptor.pdf

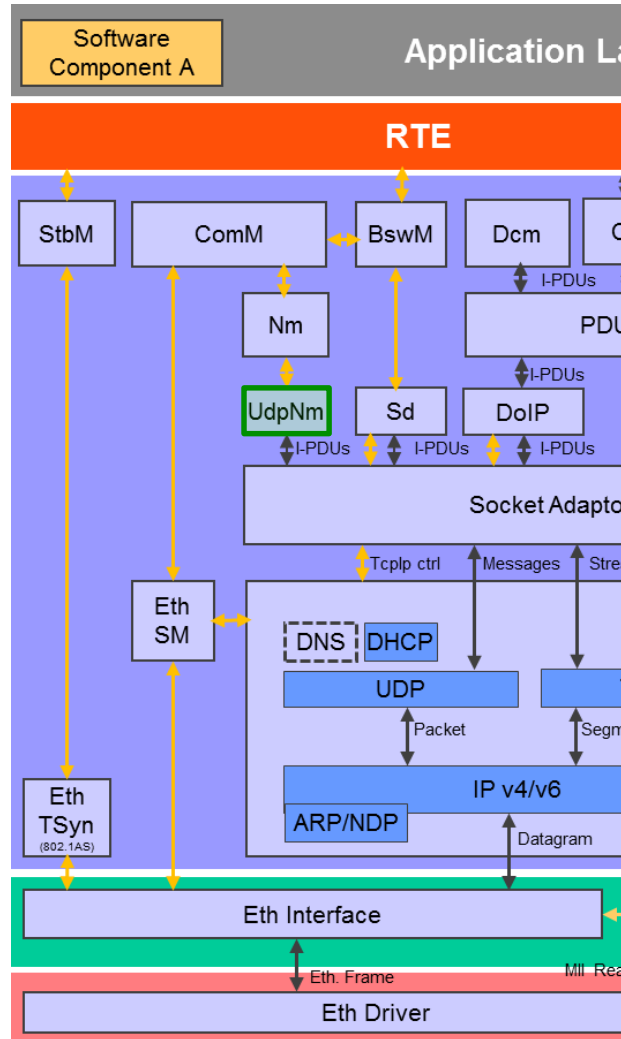
AUTOSAR模块 - Ethernet StateManager



- **打开和关闭**以太网通道
 - 调用Ethif的接口函数打开，关闭以太网发射器，以太网控制器
 - 调用TcpIp的接口函数来切换Tcp/Ip的状态
 - 也可以控制VLAN的状态

AUTOSAR标准文档: AUTOSAR_SWS_EthernetStateManager.pdf

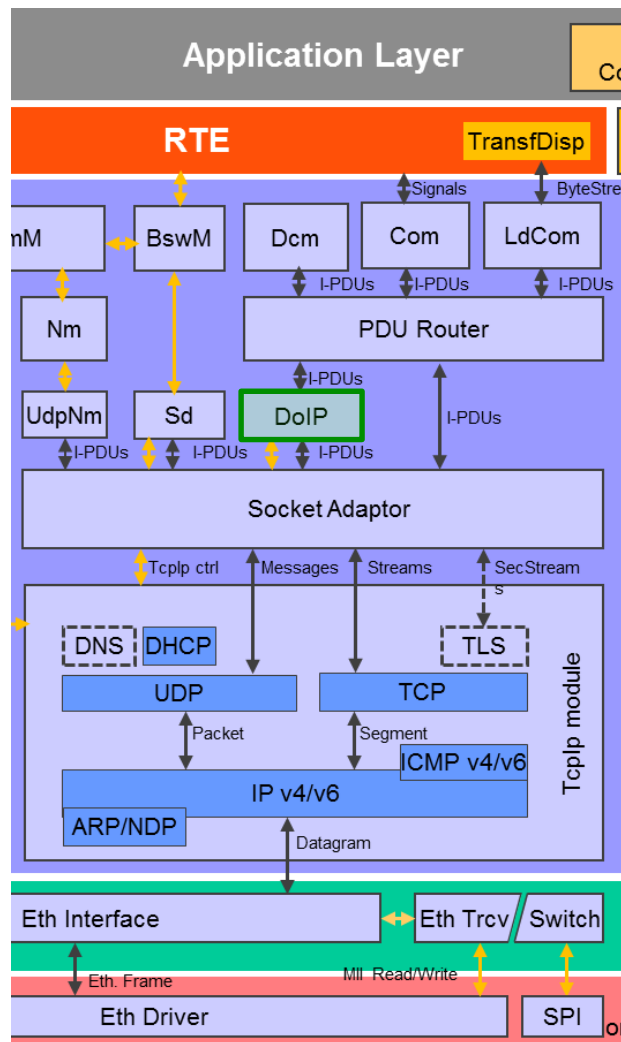
AUTOSAR模块 - UDP NetworkManagement (UdpNm)



- 以太网的网络管理模块
 - 工作模式类似于CAN的网络管理模块
 - 是SoAd模块的上层模块之一
- 通讯使用UDP**广播报文**
- 可防止本节点进入休眠以及防止网络上其他节点进入休眠
- 与其他节点配合进入休眠

AUTOSAR标准文档: AUTOSAR_SWS_UDPNetworkManagement.pdf

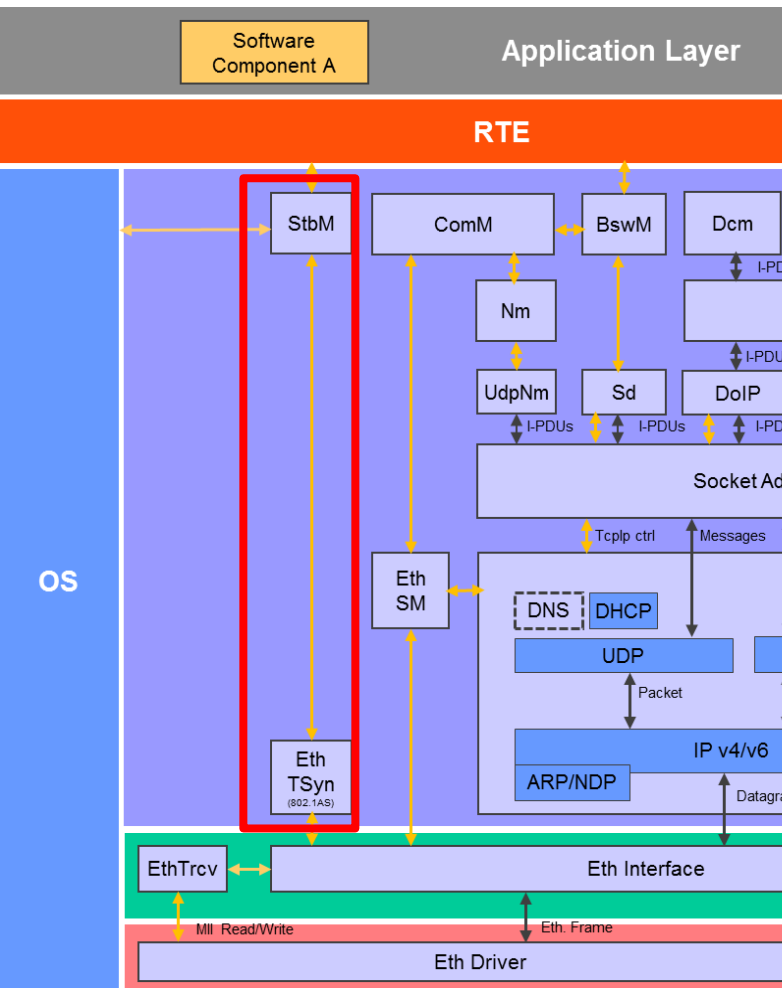
AUTOSAR模块 – DoIP



- 与诊断仪交互的以太网模块
 - 基于ISO 13400标准
 - 建立连接, 使用 AutoIP 标准动态分配地址
 - 路由激活(Routing Activations)
 - 收发诊断报文
- SoAd的上层模块之一
 - 通过**UDP标准**收发车辆ID(Vehicle Identification)
 - 通过**TCP标准**收发路由激活和诊断报文(Diagnostic Messages)
- PduR的下层模块之一
 - 路由激活使能了诊断报文收发通道后, 在PduR和SoAd模块之间传输诊断报文
 - 诊断请求最终会传递给**DCM模块**进行处理

AUTOSAR标准文档: AUTOSAR_SWS_DiagnosticOverIP.pdf

AUTOSAR模块 – StbM、EthTsyn



- 以太网时钟同步
 - IEEE 802.1AS协议在AUTOSAR框架下的实现
 - 基于PTP(Precision Time Protocol)协议
- StbM(Synchronized Time Based Manager)
 - 为用户提供与其他节点进行同步后的标准时间
 - 可处理来自不同总线的同步时间信息，包括CAN, Flexray
- EthTsyn(Time Synchronization over Ethernet)
 - 根据时间同步协议获取和分配以太网上的同步时间信息
 - 与StbM交互工作

工具演示



Elektrobit



测试工具介绍

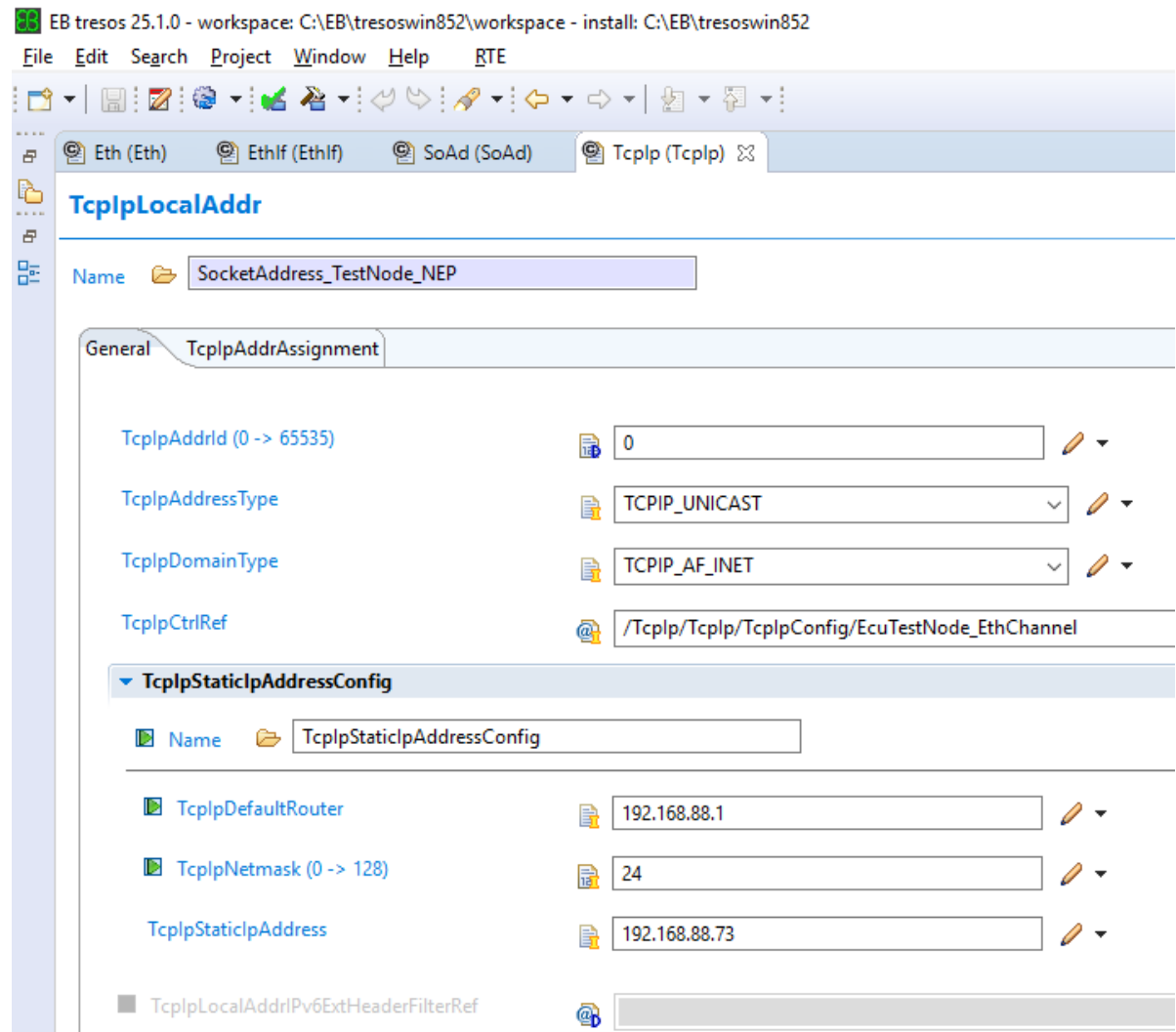
- EB tresos Studio
 - 符合AUTOSAR规范的代码生成工具
 - 用来对以太网模块进行配置
 - 在没有硬件的情况下，支持Windows平台仿真
- Wireshark
 - Windows环境下开源的网络抓包工具
- WinPcap
 - Windows环境下的工业级别数据链路层工具
 - Windows环境测试时，EB tresos使用它传输网络包到数据链路层
 - 作为Wireshark等开源抓包工具的引擎，安装Wireshark时会一并安装

EB tresos配置

- 测试目标：使用EB tresos工具配置一个使用UDP发送的以太网帧，改帧含有一个每1s更新的信号。
 - 在EB tresos的工程中导入我司提供的AUTOSAR标准以太网系统描述文件示例EthSystem.arxml
 - 抽出ECU(Ecu Extract)，导出相关以太网配置项之后可以在各模块中看到demo配置。
 - 本章节中我们挑选并先针对SoAd模块和TcpIp模块配置进行描述

EB tresos Configuration

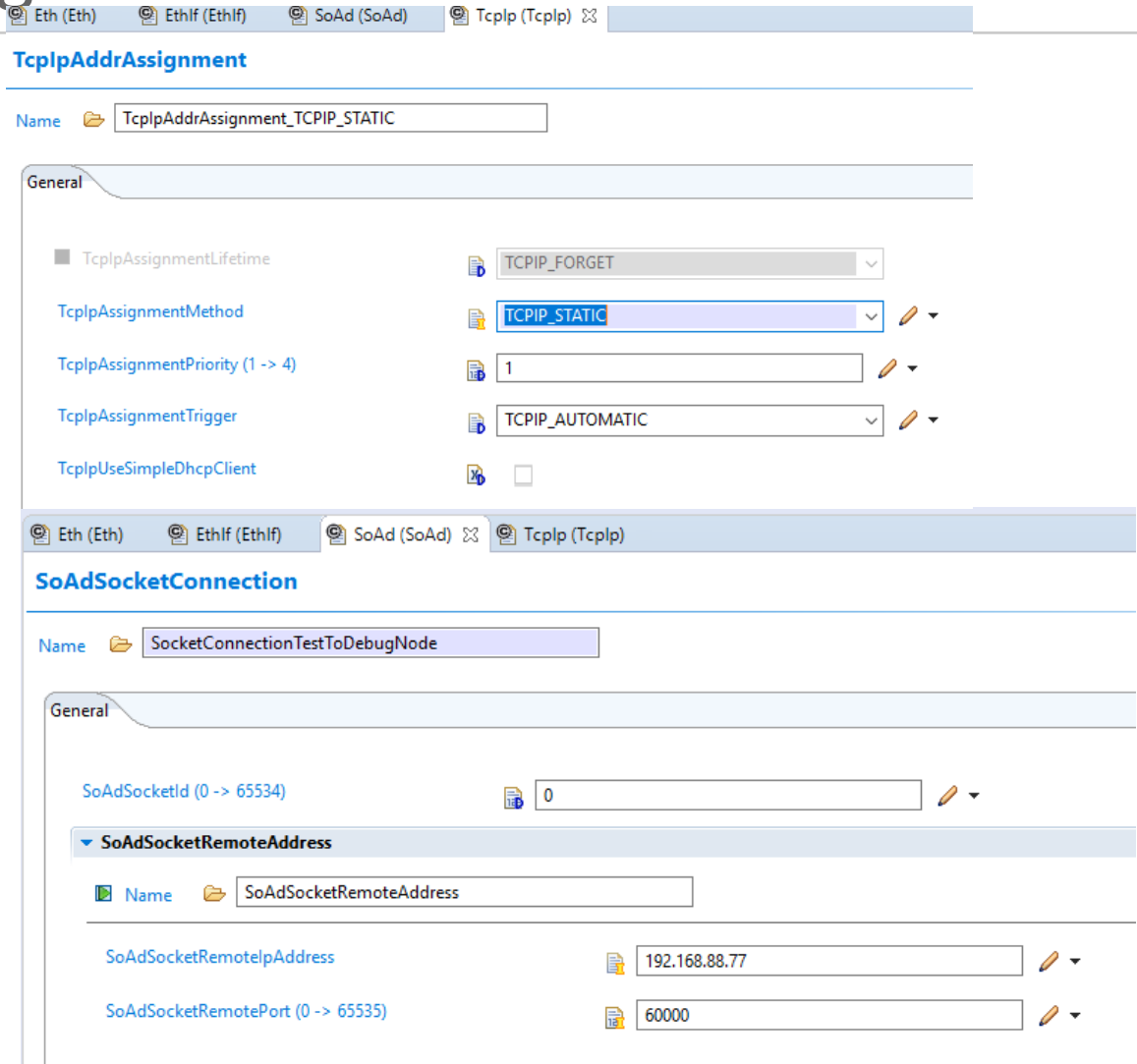
- Tcplp模块配置本机IP地址。



EB tresos Configuration

- TcpIp模块还可以配置本机地址的分配方式为静态地址或DHCP方式，这里选择静态IP

- SoAd模块为Socket配置目标IP地址以及目标端口号
- 并且对PDU的发送路径和接受路径关联Socket连接



The screenshot displays the EB tresos configuration interface for two modules: **TcpIpAddrAssignment** and **SoAdSocketConnection**.

TcpIpAddrAssignment Configuration:

- Name:** TcpIpAddrAssignment_TCPIP_STATIC
- General Tab:**
 - TcpIpAssignmentLifetime:** TCPIP_FORGET
 - TcpIpAssignmentMethod:** TCPIP_STATIC
 - TcpIpAssignmentPriority (1 -> 4):** 1
 - TcpIpAssignmentTrigger:** TCPIP_AUTOMATIC
 - TcpIpUseSimpleDhcpClient:** ☐

SoAdSocketConnection Configuration:

- Name:** SocketConnectionTestToDebugNode
- General Tab:**
 - SoAdSocketId (0 -> 65534):** 0
 - SoAdSocketRemoteAddress:**
 - Name:** SoAdSocketRemoteAddress
 - SoAdSocketRemoteIpAddress:** 192.168.88.77
 - SoAdSocketRemotePort (0 -> 65535):** 60000

运行生成的软件

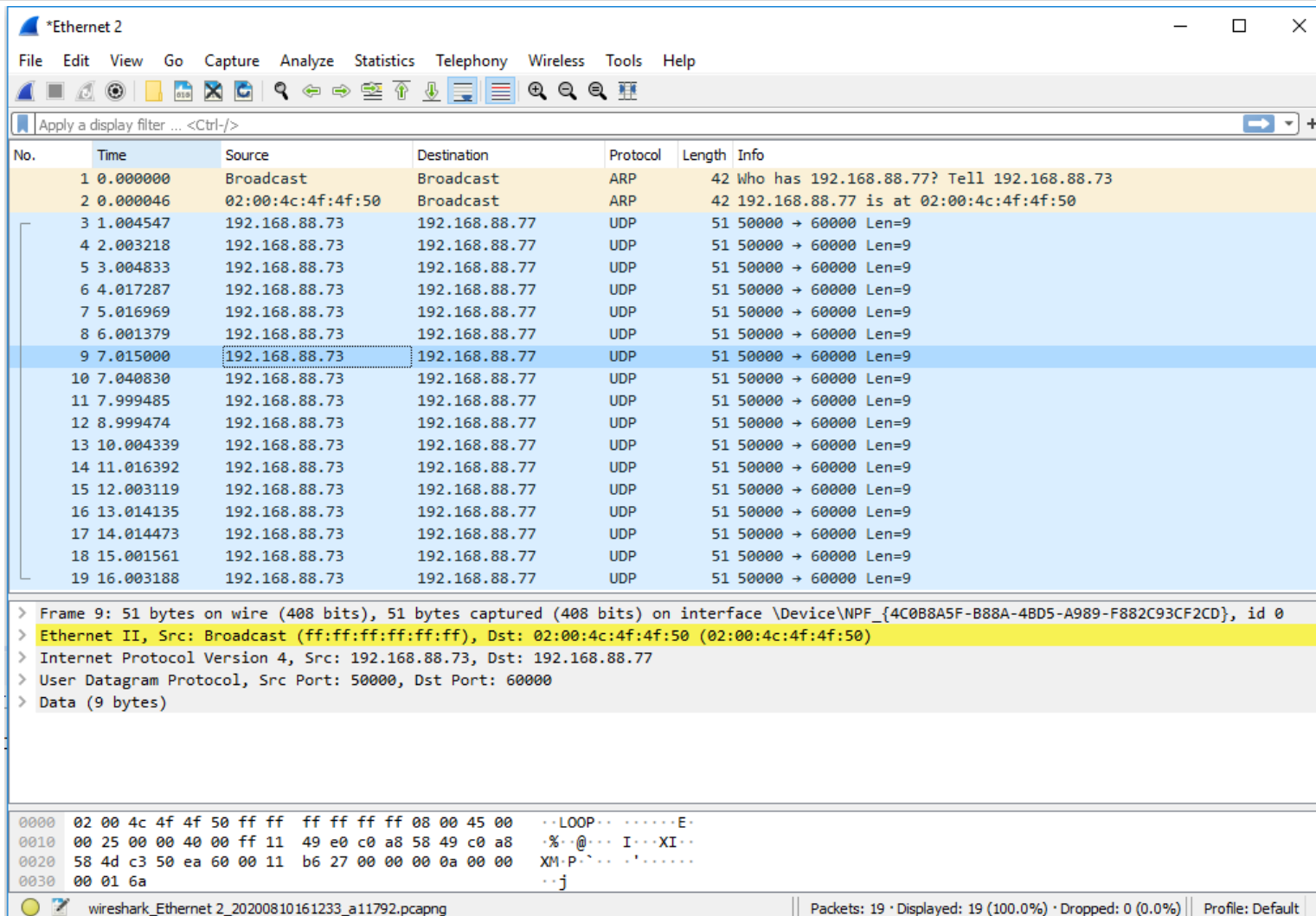
```
Select Command Prompt - WINDOWS_WIN32X86_simple_demo_eth_rte.exe
Microsoft Windows [Version 10.0.17134.1610]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\yiwa271734>cd C:\EB\tresoswin852\workspace\simple_demo_eth_rte_original\output\bin

C:\EB\tresoswin852\workspace\simple_demo_eth_rte_original\output\bin>WINDOWS_WIN32X86_simple_demo_eth_rte.exe
1. rpcap://\Device\NPF_{2F4BA22E-1609-4D2B-9439-2A36B51BE53C} (Network adapter 'Microsoft' on local host)
2. rpcap://\Device\NPF_{4C088A5F-B88A-4BD5-A989-F882C93CF2CD} (Network adapter 'MS NDIS 6.0 LoopBack Driver' on local host)
3. rpcap://\Device\NPF_{B7A59EFB-889B-4368-868C-93EA590227BB} (Network adapter 'Intel(R) Ethernet Connection (3) I218-LM' on local host)
4. rpcap://\Device\NPF_{85AB9C01-82C5-4F57-BF72-512737E92427} (Network adapter 'Microsoft' on local host)
5. rpcap://\Device\NPF_{90B4F49F-F957-499B-B307-178DBF771F56} (Network adapter 'Fortinet' on local host)
6. rpcap://\Device\NPF_{230C78A3-8D10-4962-B2CE-C3DB6E452261} (Network adapter 'Microsoft' on local host)
7. rpcap://\Device\NPF_{23952CCF-3920-4132-919A-9464D48FAAC4} (Network adapter 'Juniper Networks Network Agent Virtual Adapter' on local host)
8. rpcap://\Device\NPF_{9867711C-E333-4DB5-8A9D-F73DD8D4E30D} (Network adapter 'Oracle' on local host)

listening on Network adapter 'MS NDIS 6.0 LoopBack Driver' on local host...
.Dio channel 0: 0
Counter: 99
.....Dio channel 0: 1
Counter: 100
packet transmitted, len=42
.packet received, len=42
.packet received, len=42
.....Dio channel 0: 0
Counter: 101
packet transmitted, len=51
.packet received, len=51
.....Dio channel 0: 1
Counter: 102
packet transmitted, len=51
.packet received, len=51
.....Dio channel 0:
```

网络抓包



Wireshark interface showing a network packet capture on interface \Device\NPF_{4C0B8A5F-B88A-48D5-A989-F882C93CF2CD}, id 0. The packet list shows 19 packets, with packet 9 selected. The packet details pane shows the following information:

- Frame 9: 51 bytes on wire (408 bits), 51 bytes captured (408 bits) on interface \Device\NPF_{4C0B8A5F-B88A-48D5-A989-F882C93CF2CD}, id 0
- Ethernet II, Src: Broadcast (ff:ff:ff:ff:ff:ff), Dst: 02:00:4c:4f:4f:50 (02:00:4c:4f:4f:50)
- Internet Protocol Version 4, Src: 192.168.88.73, Dst: 192.168.88.77
- User Datagram Protocol, Src Port: 50000, Dst Port: 60000
- Data (9 bytes)

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```

0000  02 00 4c 4f 4f 50 ff ff ff ff 08 00 45 00  ..LOOP.. ....E.
0010  00 25 00 00 40 00 ff 11 49 e0 c0 a8 58 49 c0 a8  .%.@... I...XI..
0020  58 4d c3 50 ea 60 00 11 b6 27 00 00 00 0a 00 00  XM.P`... '.....
0030  00 01 6a                                     ...j
  
```

Wireshark Ethernet_2_200810161233_a11792.pcapng | Packets: 19 · Displayed: 19 (100.0%) · Dropped: 0 (0.0%) | Profile: Default

联系我们!



 Elektrobit

saleschina@elektrobit.com

www.elektrobit.cn

