FNV2 Vehicle Network Manager (L3/IP) Architecture (FNV2-L3-VNM)

Status	IN PROGRESS
Completion (%)	10 %
Version	0.6
Owner	Unknown User (memborg) Mandya Nayak, Prakash (P.)
Reviewer	Lisak, Darko (D.) Murugesapillai, Nava (N.) Alievsky, Michael (M.)
Review Date	v0.1 - 18 Jan 2017 v0.2 - 07 Mar 2017 v0.3 - 15 Mar 2017 v0.4 - 24 Mar 2017
Approver	Murugesapillai, Nava (N.)
Approve Date	v0.1 - 18 Jan 2017 v0.2 - 08 Mar 2017 v0.3 - 16 Mar 2017 v0.4 - 31 Mar 2017
HLD Jira	ECG-10033 - Authenticate to see issue details

Requirement JIRA Links	
Test Story Link(s)	ECG-10033 - Authenticate to see issue details
Test Plan Link	Test Plan: Vehicle Network Manager

Table of Contents

1. Glossary

Term	Definition

IEEE 1588	The IEEE standard that defines Precision Time Protocol (PTP)
ACL	Access Control List
ARP	Address Resolution Protocol
BPF	Berkeley Packet Filter
DHCP	Dynamic Host Configuration Protocol
GRE	Generic Routing Encapsulation
IP	Internet Protocol
iptables	linux' command-line utility for manipulating IP filters and translation rules
MAC	Media Access Control
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NIC	Network Interface Card
PF	Packet Filter
PMTUD	Path MTU Discovery
QDISC	Queuing discipline
QoS	Quality of Service
SDLC	Smart Data Link Connector
SPS	Secure Provisioning Service
TC	Traffic Class
ToS	Type of Service
TSN	Time-Sensitive Networking
TSO	Transmit Segmentation Offload

Extensive list of Silver terminology can be found here: Terminology.

2. Summary

2.1. Background/Overview

In earlier generations of the Ford Networked Vehicle, the Smart Data Link Connector (SDLC) was the gateway through which other ECUs connected. All communication between the ECUs were via high-speed CAN lines and IP data communication was restricted to a few

internet-connected ECUs like SYNC and TCU (with a 4G/LTE modem). With increasing technological changes in the connected vehicle space, the need for a better vehicle architecture where big data could be leveraged and where the vehicle could be automatically upgraded to get better over time was apparent. A new architecture, Fully Networked Vehicle 2 (FNV2) was developed to meet these new needs and the Enhanced Central Gateway (ECG) designed to support the FNV2 architecture. The ECG replaces a vehicle's Smart Data Link Connector (SDLC) and provides similar gateway functionality and adds capabilities for Ethernet and computing to support FNV2.

The focus for this document is the IP network topology and configuration. As such, it only peripherally references software components on the ECG. The features of the operating system kernel in ECG, including addresses, routing, packet filtering and NAT features are configured in an operating system specific manner by the Vehicle Network Manager, based on decisions and instructions communicated to it by the ECG Connection Manager. Application processes accessing the ethernet network access the socket layer of the kernel to transmit and receive data. The processes may also negotiate with the ECG Connection Manager in order to establish the optimal path for external data.

2.2. Purpose

This document describes the high-level design for Layer 3 Internet Protocol (IP) Architecture (ECG-442).

Internet Protocol (IP) is the Layer 3 protocol of choice for networking the FAST ECUs (eg. ECG and TCU) in the Fully Networked Vehicle (FNV2) architecture. This high-level design document focuses on the functionalities/features provided by the FNV2 Vehicle Network Manager which is the networking IP manager responsible for all IP operations within the FNV2.

2.3. Scope

The L3 FNV2 Vehicle Network Manager (FNV2-L3-VNM) covers the following features and functionality:

1. Defining the in-vehicle Layer 3 IP network topology
2. Configuration of the IP Network - Providing separation between trusted and untrusted networks via IP subnets
3. IP Address assignment - Exploring static vs role-based provisioning models
4. Provisioning of the network for Layer 3 related features
5. IP Networking service assignments
6. IP Routing and address re-writing - Utilizing routing tables, providing packet forwarding and packet filtering for the vehicle network
7. Layer 3 IP Quality of Service
8. Layer 3 security considerations
9. Fault monitoring and handling

3. Requirements

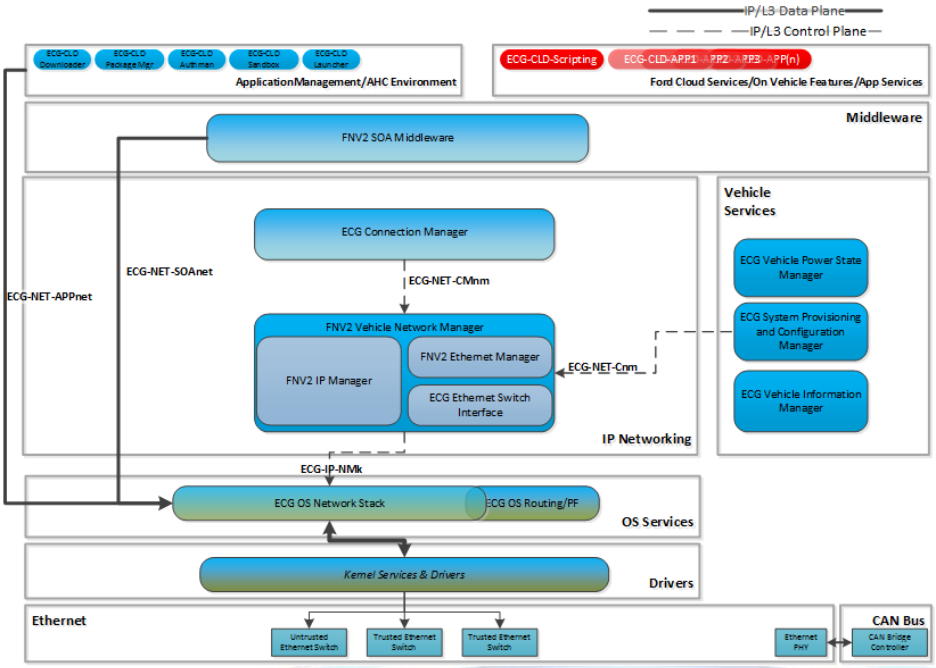
Current requirements identified for FNV2 IP networking architecture are available at the ECG IP Networking Requirements Analysis page. These requirements are listed below and on the ECG Connection Manager (CM) and Networking dashboard.

Requirement ID	Description
ECG-165	Separate IPv4 subnets for trusted and untrusted domains
ECG-166	DHCP Server on ECG for IPv4 address assignment
ECG-168	Support non-DHCP ECUs
ECG-169	DHCP Client Name Submission
ECG-170	DHCP Client Default Gateway Configuration
ECG-173	BIND Name Resolution Service on ECG
ECG-174	IP Packet Forwarding Configuration on ECG
ECG-175	IP Port Blocking Support
ECG-176	Stateful Packet Filtering Translation Service
ECG-177	ECG must support multi-home routing
ECG-966	ECG providing ARP proxying

4. Block Diagram

The following block diagram shows both the data plane as well as the control plane interfaces involved with FNV2 Vehicle Network Manager - IP/L3 Networking.

Figure 4.1 FNV2 Vehicle Network Manager - IP/L3 Networking Block Diagram



5. Interfaces

The interfaces in the FNV2 Vehicle Network Manager (L3/IP) Architecture are as represented in the block diagram and are described in detail in the Data Plane and Control Plane sections.

5.1. Data Plane

1. **ECG-NET-SOAnet** (*FNV2 SOA Middleware to Network Stack*): The interface that is the transport data path for data shipped across from FNV2 SOA Middleware out/into the ECG via the ECG network stack - MQTT over TCP/IP data path.

2. **ECG-NET-APPnet** (*Application Management/AHC Environment and App Services layers to Network Stack*): The interface used by higher-layer applications to access networking functionality (e.g., sockets).

5.2. Control Plane

1. **ECG-IP-NMk** (*FNV2 Vehicle Network Manager* to *Network Stack*): This is basically the BSD sockets API exposed by the kernel. The Vehicle Network Manager may provide some abstraction layer to this API. As well, this provides access to the layer 2 driver calls necessary to configure the various Ethernet components.

2. **ECG-NET-CnM** (*ECG System Provisioning and Configuration Manager* to *FNV2 Vehicle Network Manager*): Allows configuration of the Layer 2 and Layer 3 components.

Eg: Control commands from the ECG System Provisioning and Configuration Manager to FNV2 Vehicle Network Manager could be something like:
- Enter learning mode
- Exit learning mode
- etc.

3. **ECG-NET-CMnm** (*ECG Connection Manager to FNV2 Vehicle Network Manager*): This interface is required for any lower layer networking requests made by the ECG Connection Manager.
   Eg: Control command from the ECG Connection Manager when it determines that client/service tunnel needs to be established with certain nodes, pairing of client-service tunnels, tear-down of tunnel paths etc.

6. High Level Design (HLD)

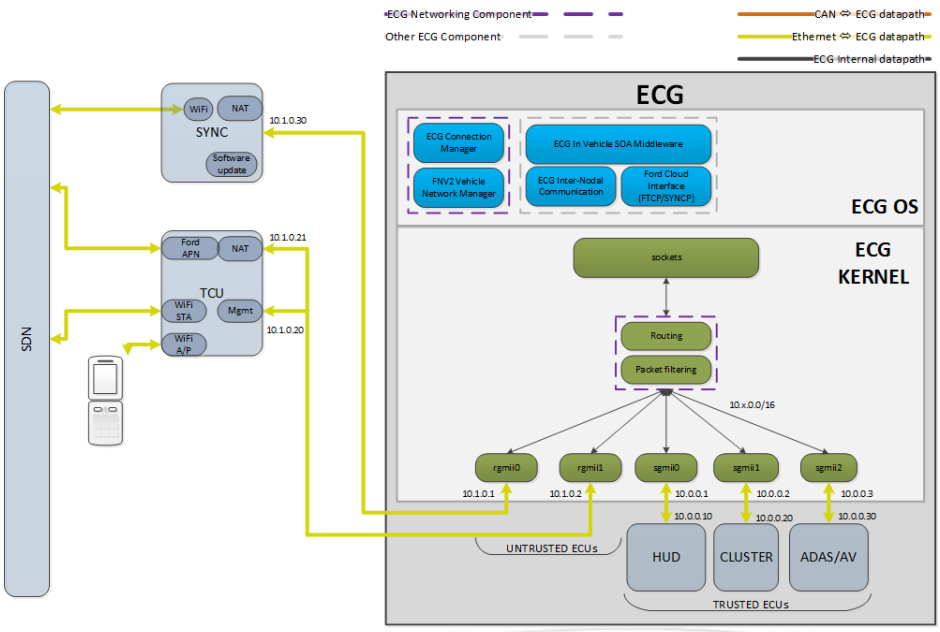
6.1. FNV2 Networking Architecture Overview

Below is the basic topology for IP networking. Assuming at least three ethernet ports on ECG, there will be at least 3 IP subnets, here shown using addresses in 10.x.0.0/16 subnets. The subnets are assumed to be used for one trusted network, one untrusted network, and one port used for DoIP (although DoIP might not be in scope for the initial stage).

ECG, as the central point of communications, has a presence in each network and is assigned an IP address in each subnet.

The ECG is designated as the default gateway in each module, except for the ECG itself. Therefore, when a module needs to communicate with a module on a different subnet, packets will be routed by ECG. This further gives ECG the opportunity to install packet filtering (firewall) rules to restrict the the destinations and packet types which are permitted to pass.

Figure 6.1 FNV2 IP Networking Architecture Overview



Visio diagram here: IP_tunnels-5portVariant_Overview.vsd

6.2. L3 provisioning

Network provisioning is an area that goes beyond L3 scope (ie. L2, security, EoL timing impact, etc). As such, the below recommended options may change as use-cases and trade-offs are better understood.

The network operates in a hybrid mode, wherein each ECU falls into one of three categories.

1. Static well-known address, pre-provisioned
2. Static well-known address, DHCP assigned
3. Pool address, DHCP assigned

For vehicles where *all* ECUs fall in to category 1, it is recommended that certain services in the ECG be disabled, e.g ARP proxying and DHCP service.

6.2.1. Pre-provisioned static address

A pre-provisioned ECU has all required information about its vehicle network loaded to it prior to installation in the vehicle. This includes:

- The ECU's own IP address and IP addresses for a minimal set of basic services on the vehicle network.
- An ARP table. This table needs to list MAC addresses for all ECUs in its local subnet (e.g. Ethernet switch), including at least those it will at some point be sending packets to.

The following table shows an example of how IP addresses and related network identifiers will be mapped, were all ECUs to follow the pre-provisioned method. Port designations are Un for a port on the untrusted switch, Tn for a port on the trusted switch.

Figure 6.2 FNV2 IP Networking Architecture Overview

ECU	Port assignment	MAC address	IP address
ECG	U0	02:00:00:00:00:01	10.1.0.1
ECG	T0	02:00:00:00:00:02	10.2.0.1
TCU	U1	02:00:00:00:00:03	10.1.0.2
Sync	U2	02:00:00:00:00:04	10.1.0.3
Cluster	T1	02:00:00:00:00:05	10.2.0.2
ADAS	T2	02:00:00:00:00:06	10.2.0.3
HUD	T3	02:00:00:00:00:07	10.2.0.4

The example above reflects the expectation that locally administered addresses be used. The second-least significant bit in the most-significant byte of MAC addresses is the "local" bit, indicating that the addresses are locally administered and carry a guarantee of uniquess locally only.

6.2.1.1. Startup sequence

A pre-provisioned ECU will be able to send and receive IP packets the moment its IP stack has initialized. If the entire vehicle network consists entirely of pre-provisioned ECUs, from a layer 3 perspective, the network is fully able to communicate the moment the Ethernet switch has initialized.

6.2.2. DHCP-assigned static address

A DHCP-assigned address is an address assigned to a host running a DHCP client, after exchanging messages with a DHCP server according to the DHCP v4 protocol.

A *static* DHCP-assigned address is a well-known, predictable address which the DHCP server assigns to the client based on some token uniquely identifying the requesting client. The client conveys this token via the DHCP request.

Typically, in everyday computer networks, the token will be the client's globally unique MAC address, because this allows a host running on a given piece of hardware to consistently receive the same address each time it boots up.

However, in the FNV2 network, the MAC address for some ECUs in a given vehicle may only be known at a late stage, e.g. during vehicle manufacture rather than ECU software imaging. For this reason, it does not make sense to anchor DHCP-administered IP addresses with specific instances of hardware. In the vehicle network, they will instead be anchored with specific functional labels communicated by specific ECU software. This functional name is known as the *DHCP client identification code*.

This label will be similar to a host name, and will uniquely identify the function of the ECU, rather any particular iteration of the hardware.

The DHCP client identification code will be built in to the ECU software image, and will be listed within the ECG DHCP server configuration. As such, the ECG DHCP service is able to assign a predictable IP address for a given ECG function, without regard to the MAC address asssigned to it.

ECUs falling in this category would map like the example in the below table. (The ECUs from the previous section are listed but this time assuming they are DHCP-managed). The information listed is entirely maintained in ECG, and not provisioned in other ECUs.

Figure 6.3 FNV2 IP Networking Architecture Overview

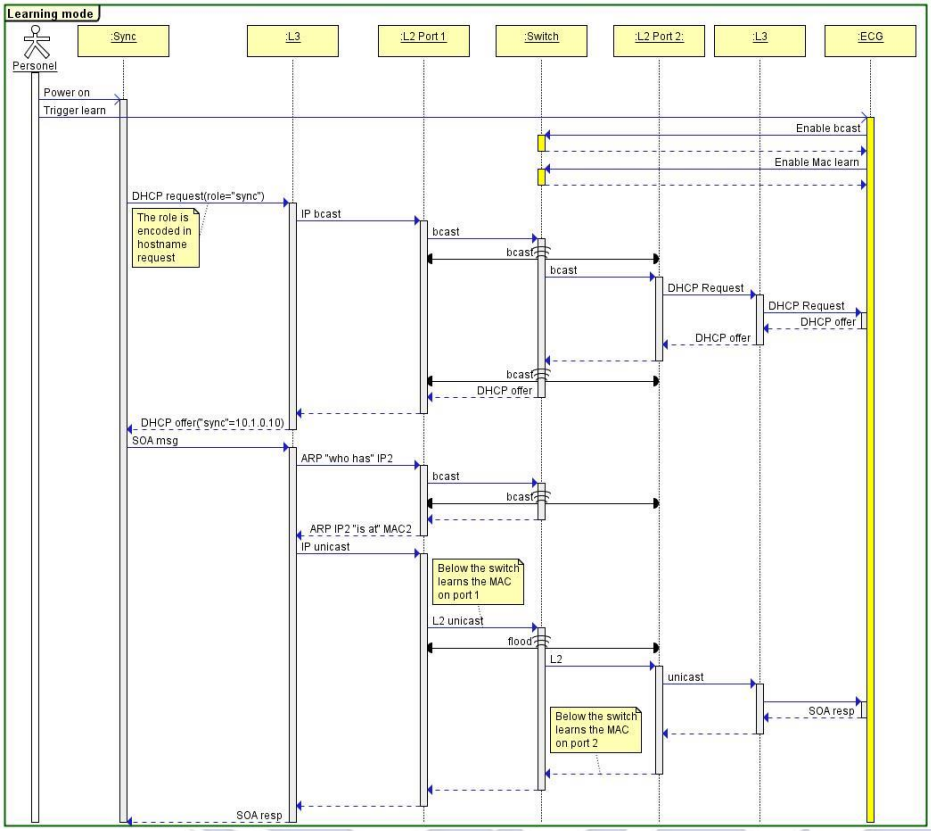
ECU	Port assignment	MAC address	client identifier	IP address
ECG	U0	(supplier-assigned)	ecg	10.1.0.1
ECG	T0		ecg	10.2.0.1
TCU	U1		tcu	10.1.0.2
Sync	U2		sync	10.1.0.3
Cluster	T1		cluster	10.2.0.2
ADAS	T2		adas	10.2.0.3
HUD	T3		hud	10.2.0.4

Please see further below regarding module authentication.

6.2.2.1. Startup sequence

When DHCP-assigned ECUs power up, they proceed through a DHCP cycle, which assigns them their own IP address and IP addresses for other basic services in the local subnet. Beyond the DHCP cycle, regular operation begins. ARP messages are employed during operation in order to dynamically resolve IP addresses into MAC addresses as required. Please further see discussion below regarding ARP. The timing of these details are as depicted in the below figure.

Figure 6.4 Network startup sequence



6.2.3. DHCP-assigned pool address

If a need arises to support an Ethernet ECU which is not pre-configured with a well-known IP address, and does not possess a client identifier, they can be accommodated by assigning IP addresses out of a pool defined by the ECG DHCP server. There is no need to enable this function unless there is a need to accommodate such an ECU. In fact, this option will likely not be used in near-term vehicle releases.

6.2.3.1. Startup sequence

Initialization for DHCP-assigned ECU is fundamentally the same, whether or not their IP address is functionally based or pool-based. Please see previous section for the discussion of this sequence.

6.2.4. Multi-location ECUs

If there is a type of ECU, such as a camera, which installs into multiple positions in the car, there may be a need to form an association between a particular data stream and the physical location of the device generating the data. There are a few different ways to handle this situation.

- **Unique software image**. Each position ECU may be manufactured with unique part numbers and unique software images. The unique software image will be using a particular (hard-coded) client-identifier to identify the position. This directly links the device's IP address to the position.
- **Hardware configurable ECU**. Each position ECU may have something akin to a DIP switch which sets the ECU as serving a particular position. The ECU software reads the switch and subsequently uses the switch setting to select the client identifier being sent during the DHCP cycle. As such, the ECU's IP address is linked to the position via a physical setting on the device.
- **Harness-configured hardware**. This solution is logically identical to the hardware-configured ECU, only instead of a user-settable switch, the electrical harness for the device includes one or more signal pins readable by software. The signal pin(s) are driven or not driven to a voltage in a pattern unique to the position. Reading the pins leads to results similar to reading a DIP switch.

6.2.5. Network authentication

ECUs are authenticated by them having registered with the SOA MQTT broker (SOA traffic secured via TLS).

References: Secure SOA traffic with TLS Session & FNV2 Security Architecture - In-Vehicle SOA security (FNV2-SecM-SOA)

6.2.6. ARP considerations

6.2.6.1. The ARP protocol

The ARP protocol is the standard protocol by which hosts on a dynamic Ethernet network discover mappings between Ethernet MAC addresses and IP addresses. Normally, on a dynamic Ethernet network, each host discovers for itself the MAC address for the IP address that it intends to transmit a packet to. Each host's kernel does so by broadcasting ARP messages on the network and listening for responses. Typically the host using the solicited IP address will broadcast a response indicating its own MAC address as the owner.

The requesting host will store the discovered information in its *ARP cache* for future use, to reduce redundant ARP messaging.

6.2.6.2. ARP proxying

Even though typically the owner of a requested IP address will broadcast the response for it, it is also possible for a third party host to broadcast a response, if it stores the relevant mapping. This unfortunately also opens the door to *ARP poisoning* attacks. By responding to ARP broadcast messages with incorrect information, malicious hosts will be able to cause legitimate hosts to store the incorrect information.

In the FNV2, we will configure the switch such that broadcast messages originating from any non-ECG port will be masked from any non-ECG port. Only broadcast messages originating from the ECG port will be accepted at the remaining ports. With this scheme, ECG will be the source of all ARP information.

6.2.6.3. ARP in static networks

A fully static network, where fundamentally all network identifiers are known in advance, can be made to operate without ARP messaging. In the place of ARP caches, basically hosts operate a static ARP table pre-configured with all known and necessary mappings.

It is recommended that IP and ARP broadcast messages be disabled in vehicles except where required. That is, in FNV2 vehicle networks where all outfitted ECU network identifiers are pre-recorded in static tables, IP and ARP broadcast messaging can and should be disabled.

CORE Netcom may provide static ARP tables to ECG to cover classic Autosar ECUs with pre-defined static network identifiers.

6.2.7. IP broadcast

As described in the previous section, the switch will be configured such that the ECG will be the only permitted broadcast listener, unless the ECG is the sender. This principle will in fact be extended to cover IP broadcasting, such that broadcasts in general mask out the ports such that non-ECG originated broadcast packets are only seen by the ECG. Please see the L2 HLD for further detail on the enforcement of this.

6.3. Configuration files

The following config files are used from the data partition:

ECU	File path	Comments
ECG	/data/config/network/dhcpd_default.conf	Seed file for DHCP-server cfg
ECG	/data/config/network/dhcpd.conf	Constructed from seed file + ARXML input
ECG	/data/tmp/vnm	Fallback for housing vnm tmp data, ONLY in the case of /tmp/vnm not being usable

In the event of file corruption, no backup strategies should be needed.

For #1 and #2 above, DHCP clients trying to connect to ECG will fail. This however should not be a concern as all of FNV2 networking provisioning is done statically (ie. not via DHCP). DHCP is only used for connecting with test devices.

For #3 above, this is dynamic/tmp data only meant for one ECG session. Corruption of this data would affect vnm-process-restarts, but vnm on ECG is configured as a CRITICAL/CATASTROPHIC process which necessitates an ECG reboot when the process crashes. Hence this data corruption should not affect vnm behavior (as the tmp data is not used anymore on an ECG reboot).

6.4. Multi-homing

Multi-homing refers to a network with gateways to multiple other networks. Typically these other networks ultimately also are interconnected such that multiple paths to a particular destination are available. In the context of the general Internet, often the multiple gateways can be seen as multiple lanes to the Internet, although in some cases, a gateway may provide access to an isolated network, a "walled garden".

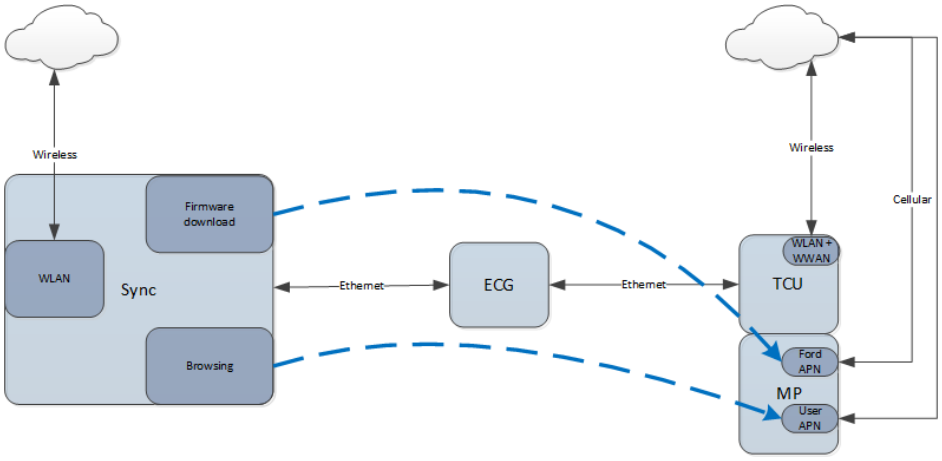
The connected vehicle is a multi-homed network due to the existence of the following potential connectivities:

- One or more wlan radios (e.g. Sync WiFi, TCU WiFi).
- cellular radio, potentially providing multiple network access points via respective APNs
- bluetooth, USB or other link to a wireless handheld device capable of further uplink connectivity.

Each connectivity may have specific characteristics with respect to availability, security, data rate, latency and cost. Some of them may further be able to be combined to yield a connection aggregating data rate and availability. A technology named MPTCP achieves this aggregation.

The following diagram shows simultaneously running applications in Sync utilizing separate cellular APNs.

Figure 6.5 Multi-homing Overview



Per design, both data paths traverse the ECG. Both applications' packets hit ECGs ethernet port and both have Sync as the source IP address and a cloud address in the destination address. So

how will ECG be able to route packets to the correct APN? We do not wish to assume anything about the cloud destination addresses, so the appropriate route cannot be inferred solely from IP addresses in the packet. The trick here is GRE tunnels.

The pair of applications each uses a separate pair of encapsulation tunnels. Browser traffic is classed as user traffic, so traffic from the browser enters the user client tunnel. The firmware download is classed as vehicle traffic, so this traffic enters the vehicle client tunnel. Each respective application ties to the appropriate tunnel by binding to the pseudo networking device representing the tunnel in the host, or its parent process does so on behalf of the application.

Packets are decapsulated on the ECG. Since packets now arrive out of a specific tunnel, the traffic classification is preserved and is used to select the most appropriate cellular APN or other transport. Based on entries added to the routing table when the tunnels were established, the routing table essentially is ensuring the pairing of client tunnel to service tunnel.

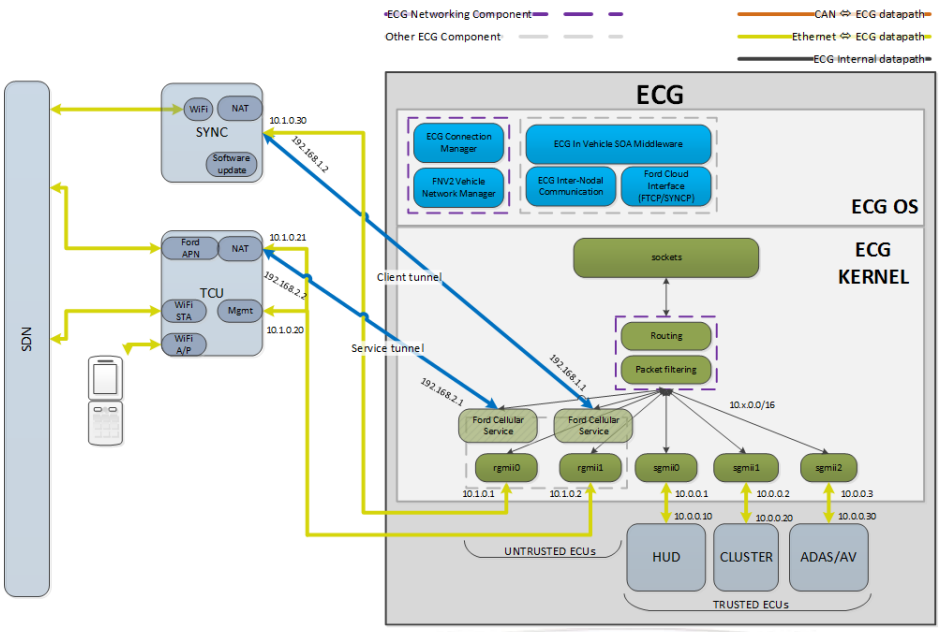
6.5. Cloud communications

The system will include a number of components capable of providing transport to networks beyond the vehicle. Among them, one or more cellular APNs, WiFi radios (in sync and/or TCU), and AppLink (ie. transport via portable devices linked to the vehicle via Bluetooth, USB etc.)

At any given time, zero or more of these transports may actually be available. As well, differing levels of cost, bandwidth or other factors may influence the optimal choice of transport. There may even be opportunities to aggregate the bandwidth from multiple transports, using techniques such as multi-path TCP (MPTCP).

To enable the capability to route an external transaction a/ through a dynamically chosen transport and b/ route such traffic through the ECG, a system of Routing Encapsulation tunnels is used, as described in the previous paragraph. The tunnel configuration changes dynamically to reflect the currently available transports. The below diagram depicts the set of tunnels being used to establish one particular route.

Figure 6.6 ECG IP Networking GRE Client/Service Tunnels



Visio diagram here: [IP_tunnels-5portVariant_Tunnels.vsd]

One tunnel is a client tunnel. This tunnel connects an arbitrary module to the ECG and is used by a module to provide the first route hop over to the ECG. A few flavors of client tunnels may be available to a module. One would be known as the vehicle traffic client tunnel, another the user traffic client tunnel. The distinction between two grades of traffic allows the most appropriate transport to be chosen based on the type of application (the process) requesting the traffic, and the permissions granted to it.

The second tunnel being used in a route is the service tunnel. This is a tunnel which several client tunnels may associate with, such that client tunnel traffic towards the ECG gets routed into the service tunnel, and returning traffic route over the service tunnel and into the client tunnel. When a service tunnel is associated with a client tunnel, it reflects the selection of a particular off-board transport to be used with the type of traffic travelling the client tunnel.

Where desired, a pair of service tunnels may be tagged as eligible for aggregation using MPTCP. The MPTCP feature in the ECG kernel stack would then combine two tunnel endpoint network interfaces and create sub-channels via both.

6.5.1. Detailed view of packet traversing tunnels

The following views depict the details of the outgoing and returning packets traversing the routing tunnels.

Figure 6.7 OTA Software Update Use-Case Sync --> Ford Cloud

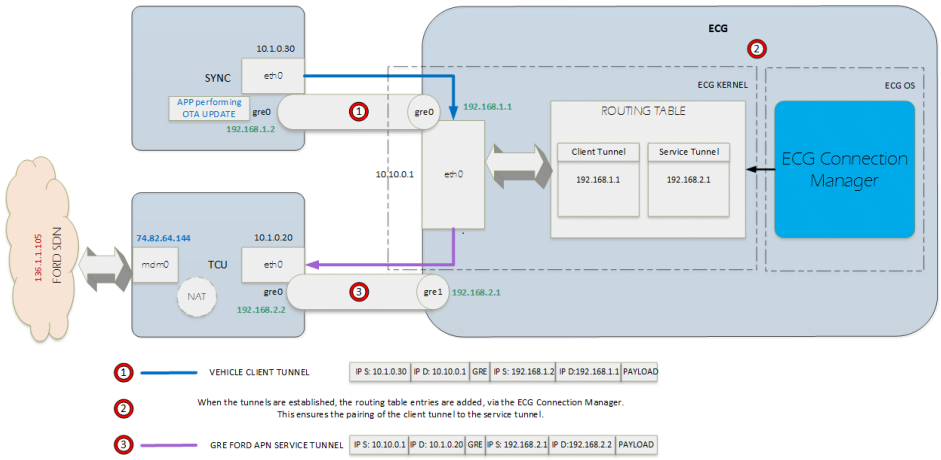
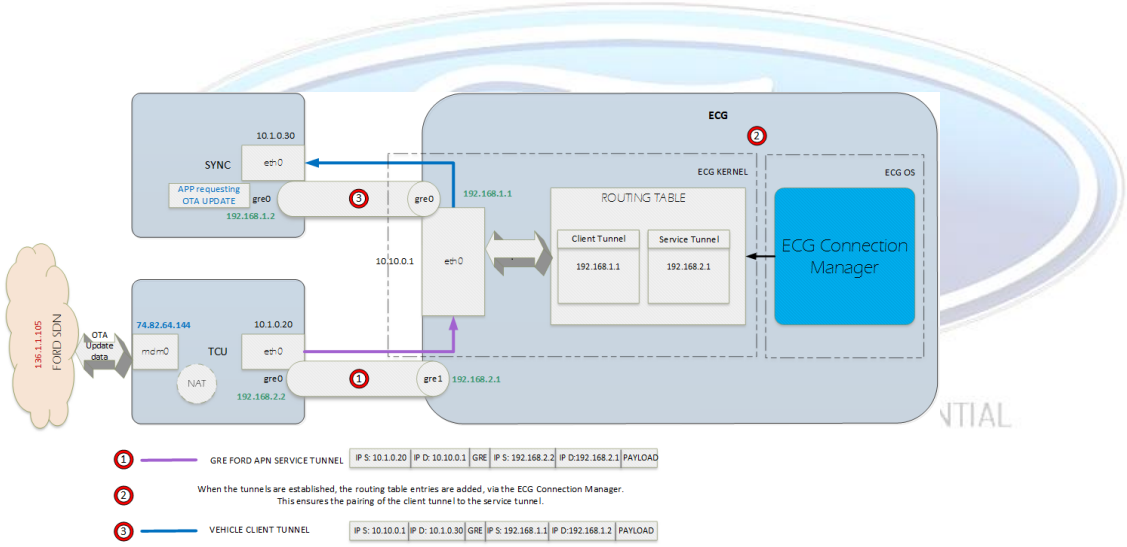


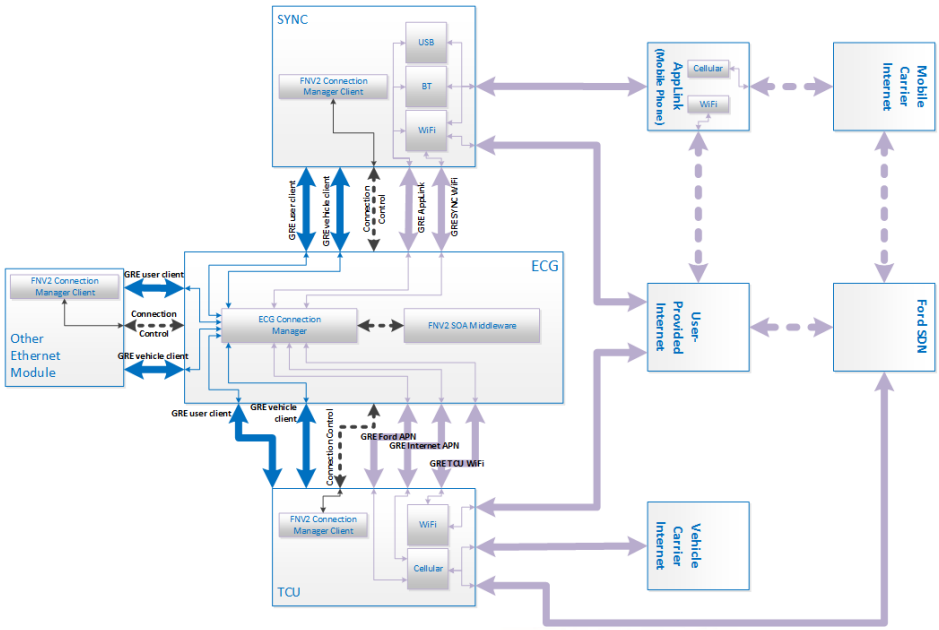
Figure 6.8 OTA Software Update Use-Case Ford Cloud --> Sync



6.5.2. System wide view of routing tunnel use

Below is a view of what the system might look like with several client and service tunnels active. The ECG emerges as a type of patch-panel which oversees pairing of tunnels and the choice of preferred transport.

Figure 6.9 ECG IP Networking GRE Tunnels Overview



6.5.3. Tunnel management: Example

Using the IP addresses depicted in figure 6.4, we will look at the setup of the client and service tunnels for vehicle traffic between Sync and the Ford cellular APN. The following will be the addresses and names taken from the illustration.

Figure 6.10 Tunnel address example

	tunnel i/f name	Local IP	Remote IP	tunnel IP	tunnel IP net mask	key
ECG client end	vclient	10.1.0.1	10.1.0.3	192.168.1.1	192.168.1.0/24	1
Sync client end	vclient	10.1.0.3	10.1.0.1	192.168.1.2	192.168.1.0/24	1
ECG service end	fordapn	10.1.0.1	10.1.0.21	192.168.2.1	192.168.2.0/24	1
TCU service end	fordapn	10.1.0.21	10.1.0.1	192.168.2.2	192.168.2.0/24	1

6.5.3.1. Tunnel management: Linux commands

```
On Sync:

        ip tunnel add vclient mode gre remote 10.1.0.1 local 10.1.0.3 key 1
ttl 255
        ip link set vclient up
```

```
        ip addr add 192.168.1.2 dev vclient

        ip route add 192.168.1.0/24 dev vclient


On ECG:

        ip tunnel add vclient mode gre remote 10.1.0.3 local 10.1.0.1 key 1
ttl 255
        ip link set vclient up


        ip addr add 192.168.1.1 dev vclient


        ip route add 192.168.1.0/24 dev vclient

        ip tunnel add fordapn mode gre remote 10.1.0.21 local 10.1.0.30 key 1
ttl 255

        ip link set fordapn up

        ip addr add 192.168.2.1 dev fordapn

        ip route add 192.168.2.0/24 dev fordapn


On TCU:
        ip tunnel add fordapn mode gre remote 10.1.0.30 local 10.1.0.21 key 1
ttl 255

        ip link set fordapn up
```



```
        ip addr add 192.168.2.2 dev fordapn

        ip route add 192.168.2.0/24 dev fordapn
```

6.5.3.2. Tunnel management: QNX commands

TBD

6.6. Packet Filtering

6.6.1. Overview

Packet filtering is the procedure by which packet headers are inspected by a router or firewall to make a decision on whether to let the packet pass.

Header information is evaluated and compared to rules that have been set up (Allow or Deny). Rules could be set up in such a way that an access list that includes all computers in the local network by name or IP address so communications can flow between them.

Stateful vs stateless packet filtering:

- Stateless: Determines whether to block or allow packets—based on several criteria—without regard to whether a connection has been established. Also called static packet filtering.
- Stateful: Performs packet filtering based on contents of the data part of a packet and the header. The packet filter maintains a record of the state of a connection and allows only packets that result from connections that have already been established. They are aware of communication paths and can implement various IP security functions such as tunnels and encryption. It can tell if the MTU has changed and whether packets have fragmented etc.

6.6.2. Stateful firewall on the ECG

On the ECG, we'll have a stateful firewall between the exposed/untrusted and unexposed/trusted domains. Operationally, traffic that needs to go through a firewall is first matched against a firewall rules list. If the packet type is allowed through the firewall, then the stateful part of the process begins. Since the firewall maintains a state table through its operation, the individual configuration entries are not required as would be with an ACL configuration.

Eg: TCP traffic

TCP keeps track of its connections through the use of source and destination address, port number and IP flags. A connection will begin with a three way handshake (SYN, SYN-ACK, ACK) and typically end with a two way exchange (FIN, ACK). For a stateful firewall, this makes keeping track of the state of a connection rather simple. An initial request for a connection comes in from an inside host (SYN). This will initiate an entry in the firewall's state table. If the destination host returns a packet to set up the connection (SYN, ACK) then the state table reflects this. Finally, the initial host will send the final packet in the connection setup (ACK). This will finalize the state to '*established*'. Once a connection is maintained as an established communication, it is freely able to occur between hosts. With TCP, this state entry in the table is maintained as long as the connection remains established (no FIN, ACK exchange) or until a timeout occurs.

Pros	• Maintains a connection state, so all packets of a session can be tracked/policed • More Secure
Cons	• CPU intensive • Slower • Could block ICMP packets thereby disallowing PMTUD -> causing IP fragmentation • Harder to keep track of stateless connections like UDP and ICMP.

6.6.3. Packet filtering on QNX

In principle, the pseudo-devices involved with packet filtering are as follows:

- pf is involved in filtering network traffic
- bpf is an interface that captures and accesses raw network traffic.

The BPF mtap wants to access packets right off the wire without any alteration and possibly copy them for further use. Callers linking into pfil want to modify and possibly drop packets.

6.6.3.1. Packet Filter (pf)

6.6.3.1.1. Packet Filter (pf) Interface

The pf pseudo-device is implemented using pfil hooks. The pfil interface is purely in the stack and supports packet-filtering hooks.

6.6.3.1.2. Packet Filter (pf) module: firewalls and NAT

A read/write/modify/block interface that gives complete control over which packets are received by or transmitted from the upper layers.
The pfil interface is used by the Packet Filter (pf) to hook into the packet stream for implementing firewalls and NAT. This is a loadable module specific to either the v4 or v6 version of the stack (lsm-pf-v4.so or lsm-pf-v6.so). When loaded (e.g. mount -Tio-pkt /lib/dll/lsm-pf-v6.so), the module creates a pf pseudo-device.

6.6.3.1.3. QNX example

The IP filtering and NAT (Network Address Translation) io-pkt* module (lsm-pf-*.so) is a dynamically loadable TCP/IP stack module. This module provides high-efficiency firewall services and includes such features as:
• rule grouping -- to apply different groups of rules to different packets
• stateful filtering -- an optional configuration to allow packets related to an already authorized connection to bypass the filter rules
• NAT -- for mapping several internal addresses into a public (Internet) address, allowing several

internal systems to share a single Internet IP address.
• proxy services -- to allow ftp, netbios, and H.323 to use NAT
• port redirection -- for redirecting incoming traffic to an internal server or to a pool of servers.
The IP filtering and NAT rules can be added or deleted dynamically to a running system. Logging services are also provided with the suite of utilities to monitor and control this module.

Utilities Reference:

pf	Packet Filter pseudo-device
pf.conf	Configuration file for pf
pfctl	Control the packet filter and network address translation (NAT) device

6.6.3.2. Berkeley Packet Filter (BPF)

The Berkeley Packet Filter (BPF) provides link-layer access to data available on the network through interfaces attached to the system. bpf is implemented as a tap in all the network drivers.

6.6.3.2.1. Berkeley Packet Filter Interface

A socket-level interface that lets you read and write, but not modify or block, packets, and that you access by using a socket interface at the application layer. This is the interface of choice for basic, raw packet interception and transmission and gives applications outside of the stack process domain access to raw data streams.
Eg: The tcpdump and libpcap library operate using the BPF interface to intercept and display packet activity.

6.6.3.2.2. QNX example

To use BPF, open a device node, /dev/bpf, and then issue ioctl() commands to control the operation of the device.

6.6.4. Ingress/Egress Data Path and Packet Policing diagrams

Figure 6.11 Ingress Data Path and Packet Policing

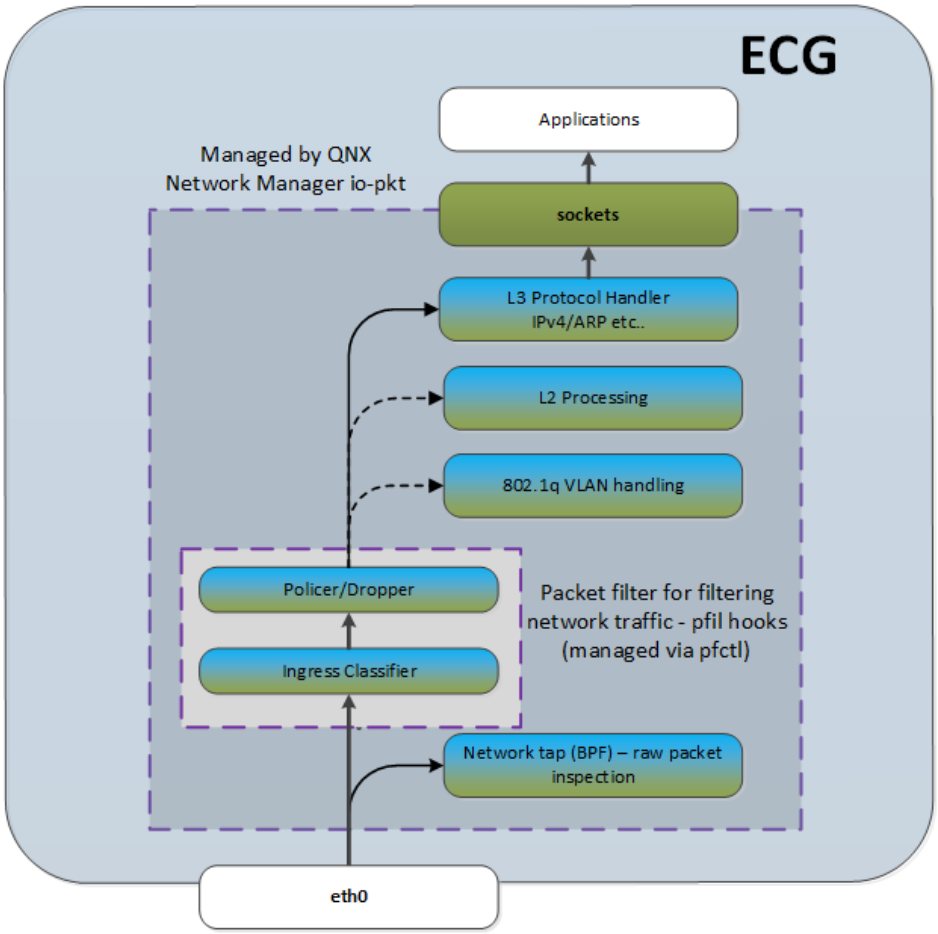
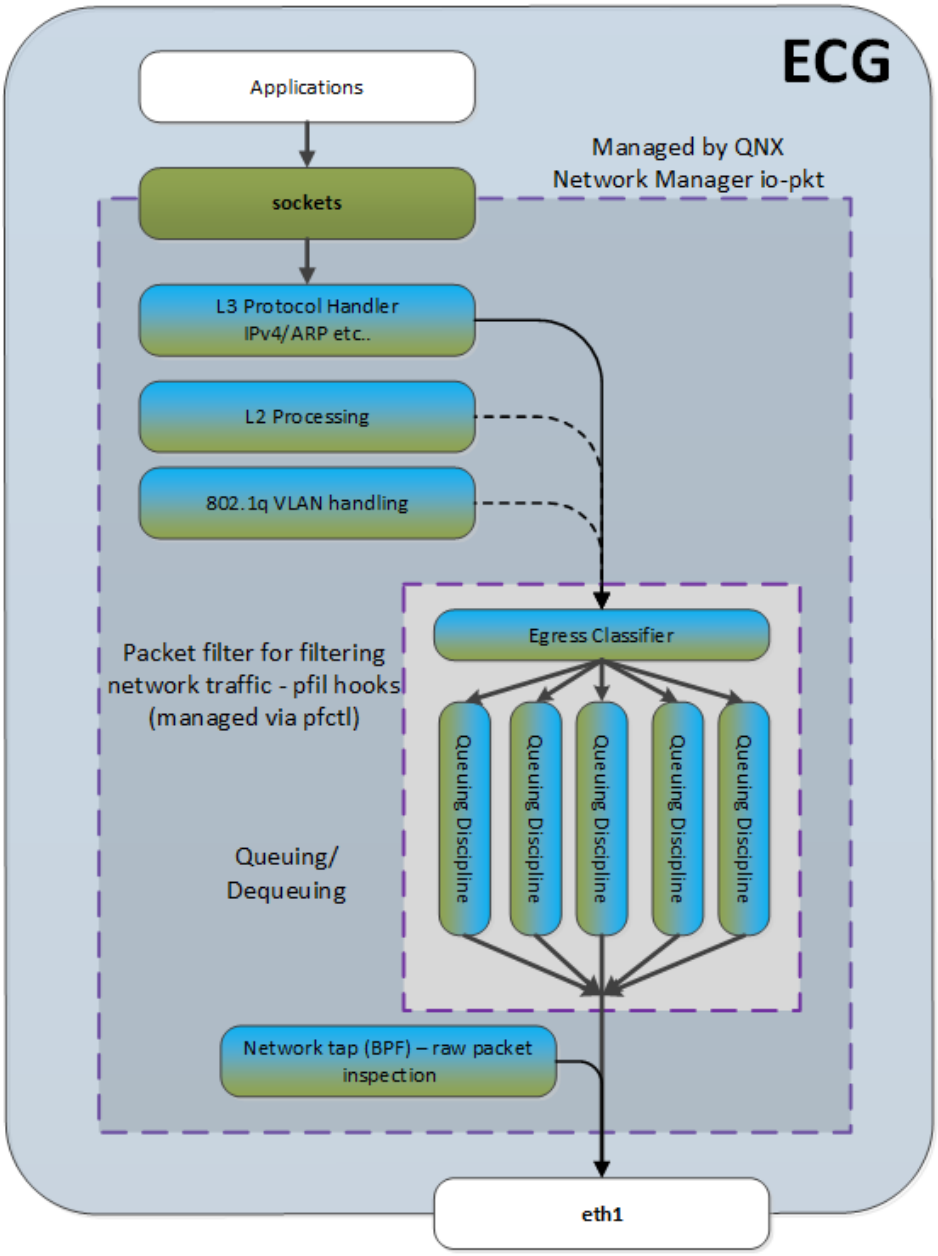


Figure 6.12 Egress Data Path and Packet Policing



6.6.5. Concerns when using packet filtering on the network

There are security and topology issues when tunneling packets. Tunnels can bypass access control lists (ACLs) and firewalls. If you tunnel through a firewall, you basically bypass the firewall for whatever passenger protocol you are tunneling. Therefore it is recommended to include firewall functionality at the tunnel endpoints to enforce any policy on the passenger protocols.

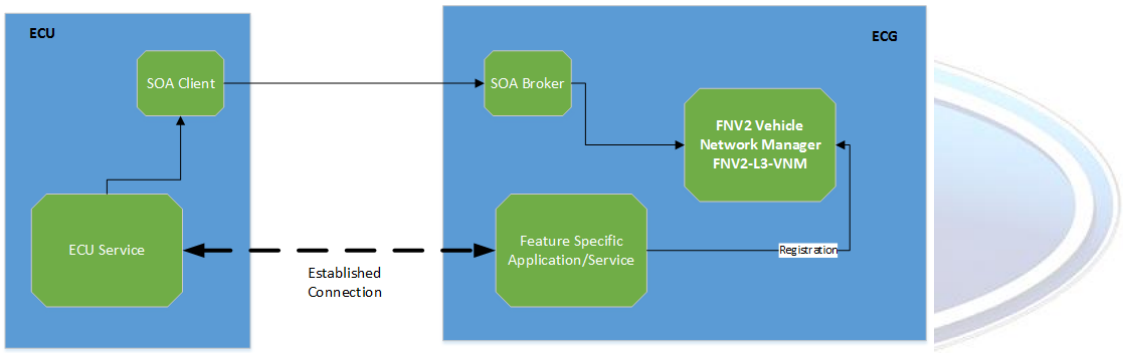
A router can generate and send an ICMP message, but the ICMP message could get blocked by a router or firewall between this router and the sender. This could potentially disallow Path MTU Discovery.

6.7. Feature Specific TCP/UDP Port Request by ECUs

Various vehicle services are distributed and may be required to establish direct TCP/UDP connections between ECG and the respective ECU. Vehicle Network Manager can enhance the system security by:

1. Ensuring that custom communication ports are opened by request only
2. The requests to open port communicated though a secure channel
3. Perform ECU Id verification

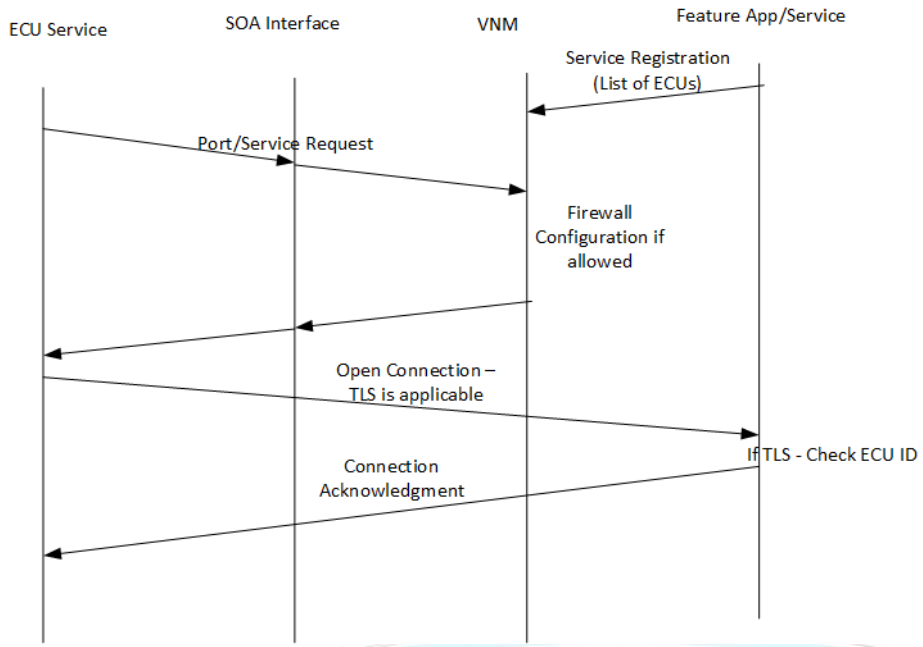
It assumed, that for direct communication between two service components, TLS protocol has to be used to guarantee a correct data source. The requirements can be dropped for some specific services like RTP streaming if it's approved by Ford security team.



Applications/services that require a direct connection with an external ECU have to register with VNM the following information:

1. UDP/TCP access port
2. List of the allowed ECU IDs
3. Application ID

Once VNM receives port open request from the external ECU, it verifies the request against the registration data. After the request is validated, VNM changes the firewall configuration accordingly.



This mechanism could also be effective in setting up direct connections between two ECUs (in addition to direct connections between ECU and ECG as described above). This would require additional support from the underlying Ethernet driver to be able to set up a layer 2 connection between two ports. However, this support does exist and is available although it is not yet integrated into the VNM.

7. Performance

7.1. IP fragmentation and Reassembly

Networks with different hardware usually vary not only in transmission speed, but also in the maximum transmission unit (MTU). The MTU is the largest protocol data unit that the layer can pass onwards. When one network wants to transmit datagrams to a network with a smaller MTU, it may fragment its datagrams. The fragmentation is usually performed at the senders/routers and reassembly at the receivers/hosts.

7.1.1. Performance Concerns

7.1.1.1. CPU and memory overhead

There is a small increase in CPU and memory overhead to fragment an IP datagram. This holds true for the sender as well as for a router in the path between a sender and a receiver.

7.1.1.2. IP fragmentation due to using GRE tunnels

IP fragmentation issues have become more widespread since IP tunnels have become more widely deployed. The reason that tunnels cause more fragmentation is because the tunnel encapsulation adds "overhead" to the size of a packet. For example, the addition of Generic Router Encapsulation (GRE) adds 24 bytes to a packet, and after this increase, the packet might need to be fragmented because it is larger than the outbound MTU.

While designing the ECG IP network, care must be taken to avoid performance costs due to IP fragmentation, as GRE tunnels would be used extensively.

7.1.2. Performance improvements

7.1.2.1. Reducing CPU overhead using Transmit Segmentation Offload (TSO)

Transmit Segmentation Offload (TSO) is a capability provided by some modern NIC cards. This technique could be used to increase outbound throughput of high-bandwidth network connections by reducing CPU overhead. It works by queuing up large buffers and letting the network interface card (NIC) split them into separate packets. Essentially, instead of the stack being responsible for breaking a large IP packet into MTU-sized packets, the driver does it. This greatly offloads the amount of CPU required to transmit large amounts of data.

7.1.2.1.1. QNX Configuration example

```
# ifconfig interface tso6
# ifconfig interface 10.1.0.1
```

7.1.2.2. Considerations for avoiding IP fragmentation

- One important consideration when designing the network should be that the MTU MUST be configured with the same value on each L3 link between 2 routers to reduce packet fragmentation.

- Use Maximum Segment Size (MSS -TCP/L4) and/or Path MTU Discovery (PMTUD), ie. tackle the packet fragmentation problem at peers/hosts instead of at routers. See linked reference for more details: [Avoid IP Fragmentation: What TCP MSS Does and How It Works]

8. Fault Monitoring and Handling

8.1. Packet loss

The below table shows an overview of possible locations of network packet loss.

Figure 8.1 Points of packet loss

Point of packet loss	Cause	Where counted	Recovery

Ethernet driver TX to switch port	Packet rate beyond line rate at switch ingress port	ECU Ethernet driver TX error count	Transport protocol or application layers
Ethernet driver RX delivery to ECU networking stack	Packet arrival rate exceeding processing capability in ECU	ECU ethernet driver RX error count	
Ethernet switch	Packets queuing rate at switch egress port exceeding transmission rate	Switch statistics	
ECG routing	Packet arrival rate exceeding processing capability in ECG	ECG Ethernet driver RX error count	
ECG routing	Packet arrival rate exceeding TX transmission rate at ECG egress port	ECG Ethernet driver TX error count	
Cloud	Congestion in network path outside of vehicle	Not available	

In general, it is the responsibility of a transport level protocol such as TCP to recover from lost or packets or packets arriving out of order. In some cases, applications choose to use a protocol like UDP which doesn't implement delivery guarantees. It may be that packet loss require isn't required for a data stream or it may be that the application handles its own recovery.

A further discussion of packet loss scenarios follows.

8.1.1. Packets being output to the vehicle network from an ECU

There is a certain amount of bandwidth available at the port where the ECU attaches to the Ethernet. Data generated in the ECU is queued for transmission at the Ethernet driver. Should the ECU generate data at a rate exceeding the line rate at the port, the queue may overflow and drop packets. The Ethernet driver counts such packet drops and makes the count available as the TX error count statistic.

8.1.2. Packets delivered to an ECU from the network.

The Ethernet driver queues packets for consumption by the application. Should the application be unable to process incoming packets at fully the rate they arrive, the queue may overflow and drop packets. The Ethernet driver counts such packet drops and makes the count available as the TX error count statistic.

8.1.3. Packets lost while traversing the network

Packets transmitted to the network by an ECU may encounter congestion at a point downstream from the ECU's local Ethernet driver. Within the vehicle Ethernet network, such points would the Ethernet switch or the ECG (if the ECU is transmitting packets across the trusted/untrusted subnet boundary). Statistics regarding packet drops in these scenarios can be retrieved from the switch or from the networking drivers in ECG, respectively.

8.1.4. Cloud communication

Packets being exchanged between an ECU within the vehicle network and a server in the cloud may be lost during routing outside of the vehicle. No counts of these losses are generally available.

8.2. Replacing a faulty ECU

- In a provisioning environment with static provisioning
  - The new part will have the same type hard-coded MAC address, and the IP-address assignment should be straight-forward as the MAC address <-> IP address assignment would be previously known.
  - Vetted part with hard-coded MAC address from factory EoL (which is already known to the ECG), so inherently secure.
- In a provisioning environment with semi-static, role-based provisioning
  - Replacing the faulty ECU would require the 'learning mode' to be toggled, where the new part will advertise its role, position etc. This role would match the role of the ECU that it is replacing and the Role<->IP-address mapping would already have been configured at the ECG.
  - Ethernet broadcast would be needed during the 'learning mode', so might have security implications.

9. Security Features

The Layer 3 security features have been discussed/documented on the ECG Security Architecture wikis: FNV2 Security Architecture (FNV2-SecM)

- Off-board vehicle communications (cloud internet communication) are generally required to use TLS sessions over TCP/IP (transport/network) - should be handled in higher layers: FNV2 Security Architecture - Cloud Internet Communication (FNV2-SecM-Cloud)
- Applications to use TLS for end-to-end encryption/authentication where required for all Internet Protocol (IP) communications.

10. DNS Name Resolution Services

ECG will run the BIND9 name resolution software.The is software behind much of the Internet DNS infrastructure. Besides being a repository for authoritative DNS records, it may also run in

a role where it forwards DNS queries to other DNS servers and forwards the responses as well as caching the answers to queries.

BIND9 will have this latter role in FNV2.

10.1. Dynamic forwarding

Because Bind will be forwarding requests to other servers, management of the server addresses to be targeted is required. Since FNV2 is mobile platform supporting multiple connectivity services which connect and disconnect dynamically, this will not be a static list, but rather a list receiving dynamic updates as the environment changes. The servers whose addresses Bind knows about and forwards to are known in the BIND9 configuration as the *forwarders*.

10.2. Updating DNS information

In FNV2, CM will learn about the DNS server addresses from upstream connectivity providers such as cellular carriers or Wi-Fi access points. As services come on line, CM will use VNM's API on ECG to add their DNS server IP addresses to a global list so as to be able to use them as forwarders. There are two API entry points as documented in the VNM Detailed Design Documents, which permits additions and removals from the list.

In BIND9's operation, when requested to resolve a name, the forwarder list is consumed in order, until a forwarder responds with a non-empty answer set.

BIND9 is configured using a config file, typically located under /etc. The config may involve references to other files which provide some of the sections. In FNV2, the *forwarders* section will be provided via a secondary file located in an updatable filesystem volume. Subsequent to each update of the forwarders config section, Bind will be triggered to re-read its config such that the new list can take effect.

10.3. DNS resolution on ECUs

Each ECU will be configured to point to ECG for name resolution services. This permits applications to resolve URLs via the Bind infrastructure described above.

11. Service differentiation (diffserv)

Utilizing the diffserv architecture can be seen as involving three main parts: marking, mapping, processing.

11.1. Marking

The DS field (formerly TOS field) in a packet originated from an ECU is updated with a value describing the type of forwarding resource the packets should receive. The following is intended for FNV

1. When an application requests connectivity from Connectivity Manager it communicates an *intent*. The intent guides the following:
    1. Which of the currently available networks (if any) are appropriate for the traffic intended by the application.This part is handled by Connectivity Manager.
    2. The forwarding behavior required of downstream routers and of link-layer devices.This part is handled by VNM based on the *Service Level* passed to it when VLANs are requested.
2. Any unique intent leads to the creation of a VLAN among the FNV nodes. One or more applications may send the same intent value. This leads to the coexistence of those applications' packets on the same VLAN. Until it is deallocated, such a VLAN maps to a fixed intent value, and with it, a fixed DS value. Since there is a particular DS value associated with each VLAN, it is beneficial for each outgoing IP packet to be marked with that value as early as possible. If an outbound packet is correctly marked the moment it enters the internal Ethernet, it enables the following benefits:
    1. When an outbound IP packet reaches the router (Sync or TCU) which will forward the packet to an external network, the DS field is consulted to ensure that the intended entry point into the interface's queuing structure is utilized for the packet.
    2. Layer-2 devices and software can be configured to map the DS value to a suitable CoS value which is propagated in 802.1Q headers. This enables access to bandwidth management features found in e.g. hardware switches.

11.2. Mapping

The intent values originated by applications guide CM in interface selection, and is additionally propagated to VNM in the form of the *service level*. From VNM's perspective, this value will guide DS marking in the corresponding VLAN.

There are a few different and somewhat conflicting standards for expressing and mapping the desired packet handling behavior.

1. The IP specification originally reserved the 8-bit "ToS field", internally comprised of 4 ToS bits, 3 precedence bits and one reserved bit. The ToS bits express the requirements of the traffic via bits representing specific concerns: Delay "md", Reliability "mr" Throughput "mt" and Monetary cost "mmc".
2. The later DiffServ specification repurposed the original ToS IP header allocation achieving more of an extensible philosophy.The diffserv specification makes the most sense for new developemnt, with a caveat: Linux uses "PFIFO_FAST" as the default queueing discipline and bases itself on the mapping of raw ToS values.

The below chart indicates how in FNV we can map our application intents via either the ToS specification or the DS specification. The linux QDISC “PFIFO_FAST”, which is the default, maps to priority bands based on the ToS bits.

Intent name	VNM Service Level	DS Interpretation			ToS Interpretation		Prio band
		Tentative DS assignment	DS value	ToS field value	ToS name	ToS field value	
Cheap background	5	BE	0x00	0x00	Normal service	0x00	2
Expired background	4	BE	0x00	0x00	Normal service	0x00	2
Foreground	3	AF12	0x0c	0x30	Minimize delay	0x10	1
FCI	2	AF11	0x0a	0x28	Maximize reliability	0x04	1
Emergency	1	EF	0x2e	0xB8	Minimize delay	0x10	0

11.3. Processing

As mentioned above, Linux uses “PFIFO_FAST” as the default queueing discipline. This means that Linux by default is set up and ready to handle the original ToS drt bits. This type of queueing discipline implements a basic fifo / priority scheme. The highest-priority queue is dequeued first, with lower-priority queues only dequeued when higher ones have no packets waiting. This scheme allows critical traffic flows to pass even when the remaining flows keep the available bandwidth fully occupied. However, the main limitation of this scheme is that no attempt is made to prevent starvation of best-effort flows. It trusts applications accessing higher priorities to utilize them responsibly.

Many other queueing systems in general have been proposed and/or implemented over the past few decades. The following are interesting in the context of this project

1. The DSMARK queuing discipline provides a direct response to the IP DS marking system.
2. The SFQ (Stochastic Fair Queueing) is of interest to prevent starvation and provide fairness between flows.
3. WFQ, an expansion on the SFQ concept to include weighted probabilities rather than an even distribution. An accessible implementation may not be available.
4. Many more, including many schemes to address congestion in particular flows.

11.4. Next tasks

- When QNX VLAN interfaces are created, configure them so that outbound packets are DF marked based on service level, according to the above table. Note this may or may not be directly possible using QNX PF. It this is deemed to not be possible, it may still be possible using pfil. If not possible, approaches per below steps on Sync and TCU can compensate, although this precludes the use of any local layer-2 traffic management features.
- If the above paragraph is not operative, inbound packets at linux VLAN interfaces can be marked. This provides the ability to provide differentiated forwarding towards cellular networks on TCU. We will implement this regardless of the other steps.

- If the initial paragraph is not operative, a method may exist whereby service differentiation can be achieved on Sync by using firewall-type marking on ingress, based on the VLAN interface packets arrive on.

- For Sync, utilize ALTQ to implement differentiated services based either on DS marking or fw marking. Note: This is a lesser priority task, since currently WiFi transport is planned only for "background" type traffic.

12. Future Considerations

This section describes IP networking related topics that have been considered, but are likely not in scope for immediate future. This may change as further requirements and use-cases are identified and solidified.

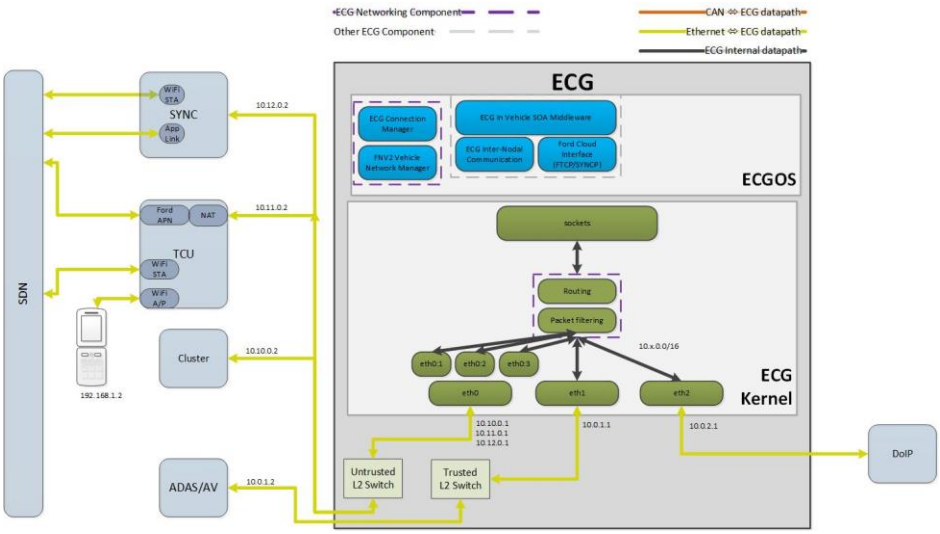
12.1. Additional separation using VLANs

As noted, the above topology shows the minimum of 3 subnets. As a result, modules on the same bus, assuming a basic layer 2 switch setup, are able to discover the MAC addresses of other modules on the same bus, hence permitting the direct passing of packets between modules without ECG's involvement.

It may be desirable to limit this direct interaction between modules on the same bus. By introducing VLANs, further partitioning is possible, confining the direct intra-bus communications to modules on the same VLAN.

In the below diagram, several VLANs were introduced on the untrusted network, such that each module shares a VLAN only with the ECG. A distinct subnet needs to be introduced for each VLAN. The ECG will be using a tagging connection for its port on the untrusted bus, such that it is able to configure three virtual network interfaces under its ethernet driver. Each virtual interface gets configured with an address in one of the subnets of the untrusted bus. Now when a module initiates communication with another module apart from the ECG, the transmission does not take place without the ECG permitting and routing the packet.

Figure 10.1 ECG IP Networking Additional separation using VLAN



The resulting design will resemble one of these diagrams based on the layer 2 switch configuration, and the VLANs defined.

Another consideration impacting the communication between modules is that there may be a SOA broker on the ECG. With an ECG broker many inter-module communications would be brokered by ECG rather than requiring direct messaging between modules.

12.2. IP Quality of Service

As applications are added to the vehicle network, the demand on network bandwidth increases. There may be a need to implement a differentiation of networking service level between different applications or services, by assigning pre-determined qualities of service (QoS) to specific processes or specific data flows. The common case in the vehicle network would be that command and control data flows need to be protected from the impact of network congestion caused by lower priority applications, such as infotainment.

IP QoS was introduced to have a mechanism for assigning a precedence to each IP packet, as well as a mechanism to request specific treatment such as high throughput, high reliability or low latency etc. This was originally introduced via the type of service (ToS) field in the IPv4 header. This ToS field has been redefined to have a six-bit Differentiated Services Code Point (DSCP) field and a two-bit Explicit Congestion Notification (ECN) field.

The figures below depict the typical IP datagram, showing the evolution of the Type of Service (ToS) field.

Figure 10.2 IP datagram depicting Type of Service/DSCP and ECN

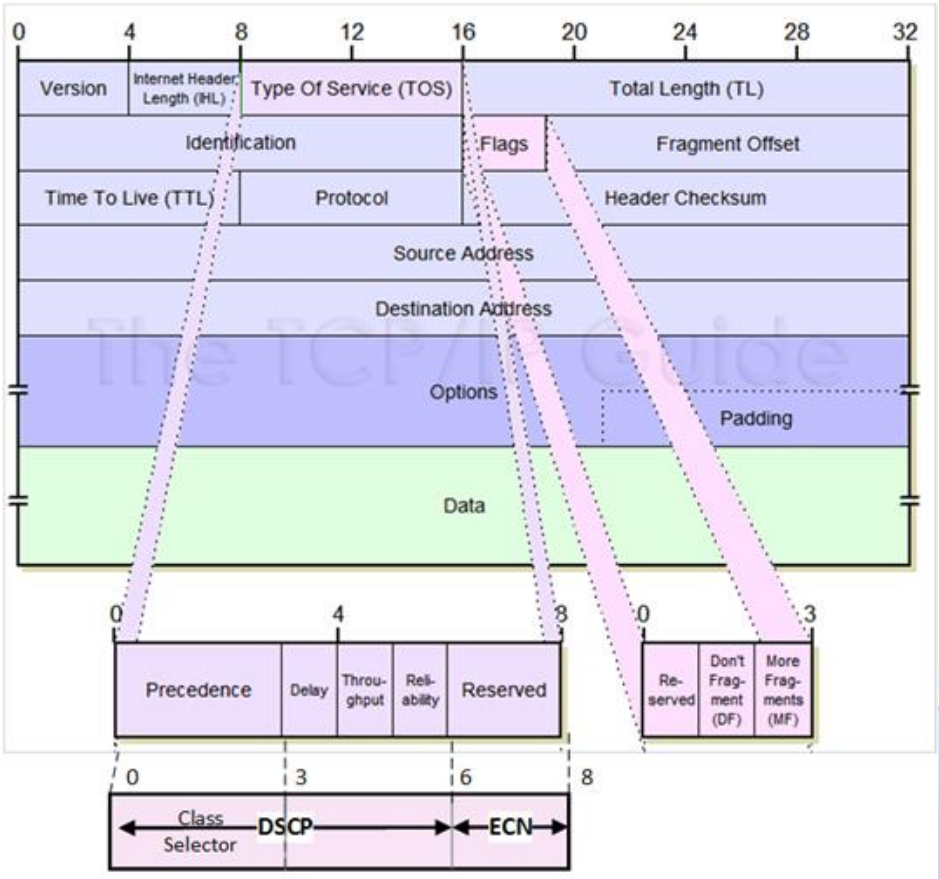
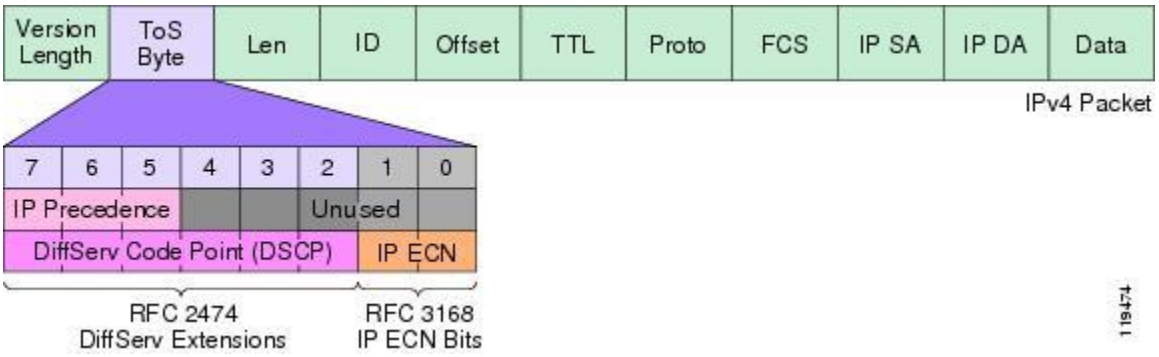


Figure 10.3 The IP ToS Byte (DSCP and IP ECN) with referenced RFCs



A breakdown of the data paths in the vehicle network is as shown in figure 7.1. Two types of service differentiation are being used.

- IP-based ToS. On packet ingress, the switch parses the IP header to find the ToS (Type of Service) field. As described in the following section, the switch implements DSCP by translating the ToS into its internal Traffic Class (TC) representation. The TC lets the switch direct network packets into a set of QoS queues associated with each port for

egress. This approach assumes that the originating host is able to assign appropriate ToS values to packets based on the originating application.

- Host queuing disciplines (Qdisc). The originating host uses queuing disciplines to reduce TX volume to the level appropriate to the port line rate (e.g. 100Mbit for ECUs). Queuing disciplines will be used to prioritize packets based on originating application.

In the below table, and ECU is an ethernet connected vehicle module other than the ECG.

Figure 10.4 Paths for QoS

Path	Limitation	Managed by	Remarks
ECU->ECG	switch ingress	Qdisc	100Mpbs limitation.
Multiple ECUs -> ECG	switch ingress each sender	Qdisc	100Mpbs limitation per sender 1Gbps bandwidth available at ECG port cannot be saturated by the remaining switch ports.
ECG -> ECU	switch egress	IP ToS	100Mbps limitation
ECU -> ECU	switch egress	IP ToS	100Mbps limitation
ECG -> Cloud via RAN	switch egress / RAN throughput	IP ToS	limitation moves to RAN when throughput less than 100Mbps
Cloud -> ECG	switch ingress / RAN throughput	N/A (LTE dedicated bearers?)	Carrier QoS required when RAN throughput less than 100Mbps
Cloud -> ECU	switch ingress / RAN throughput	N/A (LTE?)	"

12.2.1. Ethernet Class of Service (CoS)

Class of service (CoS) is a 3-bit field that is present in an Ethernet frame header when 802.1Q VLAN tagging is present. The field specifies a priority value between 0 and 7, more commonly known as CS0 through CS7 (Class Selector), that can be used by quality of service (QoS) disciplines to differentiate and shape/police network traffic.
CoS operates only on 802.1Q VLAN Ethernet at the data link layer (layer 2) (as discussed here: FNV2 Vehicle Network Manager (L2/Ethernet) Architecture (FNV2-L2-VNM)#L2-VNM)-VLAN(VirtualLAN)andQoS), while other QoS mechanisms (such as DiffServ, also known as DSCP) operate at the IP network layer (layer 3).

12.2.2. Switch IP QoS parsing capabilities

IP Quality of Service can be 1:1 mapped with the Priority Code Point (PCP) values from the Ethernet frame. This way, the priority levels at the frame level (L2) can be 1-to-1 mapped and passed on to the upper layers through IP (L3) and vice-versa.

Figure 10.5 IP DSCP <--> Class Selector/PCP values <--> Equivalent IP precedence values

DSCP	Binary	Hex	Decimal	Meaning/Examples	Equivalent IP precedence value
Critical	101 110	0x2e	46	Expedited Forwarding (EF)	101 - Critical
CS0/PCP0 (Default)	000 000	0x00	0	Best Effort	000 - Routine
CS1/PCP1	001 000	0x08	8	P2P	001 - Priority
CS2/PCP2	010 000	0x10	16	SSH traffic (Network Management)	010 - Immediate
CS3/PCP3	011 000	0x18	24	SCCP,SIP,H.323	011 - Flash
CS4/PCP4	100 000	0x20	32	Interactive/streaming video, TelePresence	100 - Flash override
CS5/PCP5	101 000	0x28	40	Voice	
CS6/PCP6	110 000	0x30	48	EIGRP,OSPF (IP routing)	
CS7/PCP7	111 000	0x38	56		

12.3. Jumbo packets for bandwidth efficiency (Performance improvement)

Nearly all IP over Ethernet implementations use the Ethernet V2 (IEEE 802.3 Ethernet standard) frame format which mandates support for 1500-byte MTU frames. Jumbo frames are network-layer PDUs that have a size much larger than the typical 1500 byte Ethernet MTU size. Jumbo frames can slightly raise the efficiency of Ethernet by reducing the overhead, as efficiency is calculated as the ratio of total payload transmitted to the total transmitted data size (which includes L1, L2, L3, L4 overheads).

For jumbo packets to work, the protocol stack, the drivers, and the network switches must all support jumbo packets.The QNX io-pkt (hardware-independent) stack does support jumbo

packets. If we use jumbo packets with io-pkt, there should be substantial performance gains as more data can be moved per packet header processing overhead.

12.3.1. QNX configuration example

To configure a driver to operate with jumbo packets:

```
# ifconfig interface ip4csum tcp4csum udp4csum
# ifconfig interface mtu 8100
# ifconfig interface 10.1.0.1
```

For maximum performance, hardware packet checksumming (for both transmit and receive) is turned on and arbitrarily chosen a jumbo packet MTU of 8100 bytes.

io-pkt by default allocates 2 KB clusters for packet buffers. This works well for 1500 byte packets, but for example when an 8 KB jumbo packet is received, we end up with 4 linked clusters. We can improve performance by telling io-pkt (when we start it) that we're going to use jumbo packets, like this:

```
# io-pkt-v6-hc -d i82544 -p tcpip pagesize=8192,mclbytes=8192
```

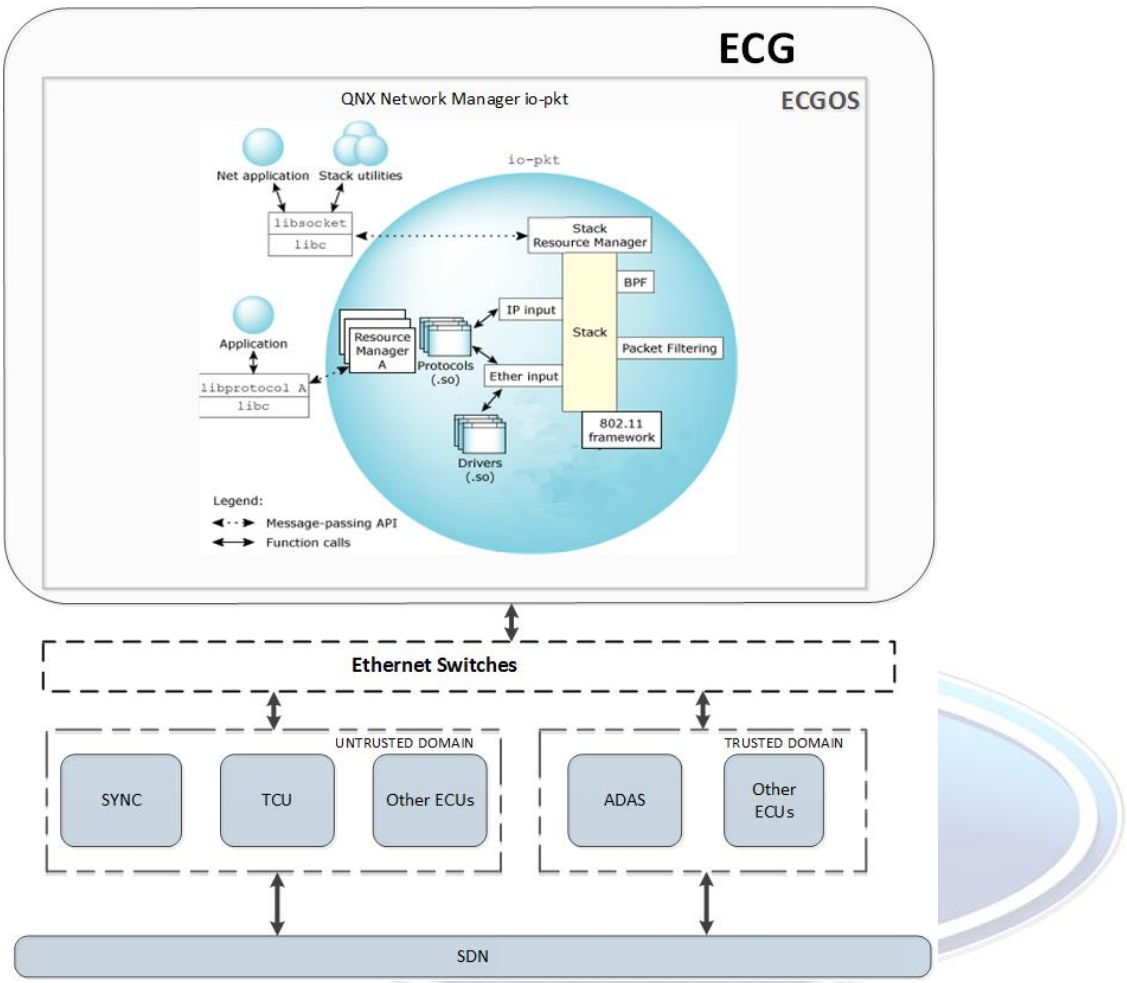
Parameter mclbytes=*size:* The mbuf cluster size. A *cluster* is the largest amount of contiguous memory used by an mbuf. If the MTU is larger than a cluster, multiple clusters are used to hold the packet. The default cluster size is 2 KB (to fit a standard 1500-byte Ethernet packet).

By passing the **pagesize** and **mclbytes** command-line options to the stack, we want to have contiguous 8 KB buffers allocated (which may end up being two adjacent 4 KB pages) for each 8 KB cluster to use for packet buffers. This reduces packet processing overhead, which improves throughput and reduces CPU utilization.

13. Supplemental diagrams

The following diagram depicts the overall network structure.

Figure 11 ECG L2/L3 FNV2 Vehicle Network Manager Overview



14. References

Documents
Hardware System_Architecture Specification_20170213.docx
ECG Hardware & Systems Architecture
ECG Software Architecture Overview
ECG and FNV2 Networking and Data Management Architecture
FNV2 Vehicle Network Manager (L2/Ethernet) Architecture (FNV2-L2-VNM)
(FUTURE) Ford Automotive Ethernet Link Implementation Specification
(FUTURE) Ford Requirements for Ethernet Switches Specification
Mitigating the Threats of Rogue Machines—802.1X or IPsec?
BCM8953X Automotive BroadR-Reach Ethernet Switch

Avoid IP Fragmentation: What TCP MSS Does and How It Works