Implementation of ETSI ITS-G5 based inter-vehicle communication embedded system

Adrian Abunei[1], Ciprian-Romeo Comșa[1,2], Ion Bogdan[1]
aabunei, ccomsa, bogdani @ etti.tuiasi.ro
[1] Telecommunications Dept., Technical University "Gheorghe Asachi" of Iasi, Romania
[2] Continental Automotive Romania SRL, Iasi, Romania

Abstract — **Inter-vehicle communication (IVC) is a key component enabling the autonomous driving vehicles. Standardization of IVC is advanced but yet improvements are still introduced. Congestion control is one of the sensitive topics. The IVC European standard ITS-G5 addresses the congestion issue enabled by a geo-networking protocol that exploits at networking level the geographic information. Implementation of the ITS-G5 standard onto low cost, open source, embedded systems for testing purposes is a challenge, and this is addressed in this paper.**

Keywords — **VANET, ITS-G5, Inter-vehicular communication**

I. Introduction

As a result of continuous developments in automotive industry toward autonomous driving, cars become more and more interconnected. Future cars will rely on high quality inter-vehicle communication (IVC) that is incorporated with safety applications. Due to their unique characteristics, IVC cannot be achieved using existing technologies, but they require specific research in academic and industrial fields. Due to rapid changing topologies and low latency connectivity together with relatively high power needs, IVC requires new protocols and mechanisms to ensure the interoperability of all vehicles in the so-called Intelligent Transportation System (ITS) or Vehicular Ad Hoc Networks (VANETs). Specific frequency bands were allocated for the development of safety-related ITS applications. For example, the Federal Communication Commission (FCC) assigned 75 MHz of licensed spectrum at the 5.9 GHz band (from 5.850 to 5.925 GHz) in the USA, divided in 7 channels, with one CCH (control channel) for safety messages. In Europe, the European Telecommunication Standard Institute (ETSI) adopted a spectrum allocation of 50 MHz (5.875-5.925 GHz) for ITS applications, divided in 5 channels, with one CCH. In addition to a 5.8 GHz band, Japan has recently allocated 10 MHz at the UHF - 700 MHz band (from 755 to 765 MHZ) for safety ITS applications.

All the three standardization initiatives, reflected by protocol stack denoted as ARIB in Japan, WAVE in USA and ITS-G5 in EU, aim to facilitate increased traffic safety and ITS reliability and to provide platform for the IVC applications [1]. For all three standards, the inter-vehicle wireless communications are based at low level on the IEEE 802.11p protocol, a vehicle-specific version of WLAN. To cope with the rapid changes of the wireless propagation channel, a set of changes were performed in 802.11p compared to IEEE802.11a, e.g., the time domain parameters were doubled, while the frequency domain parameters were halved. Use of a specific media access controller called (MAC) "Out of Context BSS" that does not include WLAN specific procedures, such as scanning for channels, association, etc., is another change. This way a low latency access environment is created. The full IVC protocol stack comprises the following layers: Access (radio and MAC, build up onto 802.11p), Networking & Transportation, Facilities, Applications, Security and Management [2]. ITS-G5 has a set of particularities.

As the way of accessing the channel and destination is regarding, WAVE uses an access scheme with EDCA (Enhanced Distributed Channel Access) for each type of control channel or service. ITS-G5 uses an alternative model that takes into account the state of radio node having access to specific parameters for the environment.

In WAVE, the communication nodes synchronize every 50 ms per channels CCH and SCH, respectively, as presented in IEEE1609.4 standard. The probability of packet loss at the beginning of each sequence channel is quite high. The alternating usage of channels leads to halve the time for the critical messages broadcasting. The advantage of this approach is the usage of a single radio transceiver with lower hardware costs. ITS-G5 eliminates the possibility of packet loss when switching channels by requiring a dedicated transceiver for the control channel, handling safety critical messages, and a separate transceiver for the service channel, handling non-critical messages.

The WLAN-based vehicular communication systems suffer from large delays and high packet losses in scenarios with a dense distribution of nodes and high data load, which can result in system instability and degraded safety application performance. To ensure a stable system and fair share of resources among vehicles, a decentralized congestion control (DCC) function limits the data that is generated by a network node depending on the measured channel load. In a vehicular ad hoc network with multi-hop communication, DCC need to control the data packets that are locally generated by the node as well as forwarded data packets [3]. This approach is possible by incorporating the location information, which is performed throughout the Geo-networking protocol implemented within ITS-G5 at the Networking level[4, 5]. Further details regarding the ITS-G5 protocol stack are given into the next section of this paper. In the followings, the description of an open-source, low cost implementation onto an embedded device, with limited processing and memory resources is presented.

II. ITS-G5 IVC STANDARDIZATION

Osi Layers	**WAVE**	**ITS-G5**		
Upper Layers	SAE BSM	CAM	DENM	***Facilities***
Transport	IEEE 1609.3	BTP		***Networking & Transport***
Network		GeoNet		
Data link	LLC			***Access***
	IEEE 1609.4	DCC		
	IEEE 802.11p			
Physical	IEEE 802.11p			

Figure 1. The US WAVE protocol stack versus EU ITS-G5 approach

ETSI ITS-G5 European standard brings important changes in the Network and the Transport layers; also, it introduces a new layer between the Transport layer and the Applications one: Facilities and a new scheme in minimizing the probability of the radio channel congestion. The 802.11 protocol suite uses the Enhanced Distributed Channel Access (EDCA) with Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and support for Quality of Service that offers fast direct inter-vehicular transmissions of the control messages. This scheme is efficient for low and medium network loads. For high loads the quality of service in wireless networks degrades and many packets are lost. In order to maintain the stability of a communication system its load should be maintain under a specified threshold and the control messages should be transmitted as fast and frequent as possible.

The ETSI ITS-G5 standard uses the DCC protocol for the network load control by dynamically changing the channel access rules:

- transmitting power parametrization;
- modifying the minimum time interval transmission of the periodic messages;
- changing the transfer rate and the radio sensitivity.

By changing the radio sensitivity, a dynamically modified threshold is set for the received signal level in order for the channel to be considered free or busy.

Besides DCC architecture design, new messages are defined for the Facilities level: CAM (Cooperative Awareness Message) – periodic messages and DENM (Decentralized Environmental Message) – event messages, as well as for the infrastructure (Local Dynamic Map, Signal Phase and Timing).

The CAM and DENM messages can be retransmitted, so they bring a supplementary channel load. There are techniques for an efficient radio channel loading and supplementary protocols for an efficient packet retransmission.

The Geo-networking protocol allows for packets' routing based on the geographic position of nodes [3]. Contention Based Forwarding (CBF) is a multi-hop packet forwarding algorithm in ad-hoc networks. According to CBF a node forwards a packet to all of its neighbor nodes and these ones compete to select the only one of them that forwards the packet farther. The CBF algorithm uses the geolocation information to select for retransmission the node that is at the greatest distance from the source node. The selection is based on chronometer type scheme that sets the minimum waiting time for the selected node [3].

The BTP protocol offers a point-to-point connectionless transport service in IVC networks. Its main objective is to multiplex/demultipex the Facilities level messages in order to be transmitted/received through the Geo-networking protocol [6].

III. HARDWARE AND SOFTWARE ARCHITECTURE

Embedded systems are Single Board Computers with specific and limited functional capabilities. A single board hosts all key components such as the microprocessor, memory, I/O units. The software is usually developed by the manufacturer, and it's build on a small non-volatile memory module. Recently, smaller embedded devices with higher computing power and larger memories such as smartphone, PDAs, micro-development boards appeared on the market. Hardware evolution determined a shift from embedded devices with build-in firmware to operating systems (OS). There are several commercial or open-source operating systems for embedded systems. Among the latter, Linux OS evolved constantly due to its popularity and non-dependence on hardware architecture, with high reliability and performance for various standards and applications.

Basically, Linux OS is a kernel, but Embedded Linux represents a complete distribution for a target device. As mentioned previously, embedded systems have limited resources in terms of memory storage. Generally, OS for Embedded Devices use small and efficient binary code to run on these devices.

The chosen development platform, Mikrotik RB433UAH runs on RouterOS, a specific and small specialized networking OS implementing the TCP/IP protocol stack. It supports other applications too, but these must abide the limitations of the HW platform. Among the usual interfaces of a router, the platform system contains other interfaces that allow for connecting a GPS unit or a LTE radio modem on USB interfaces or for connecting vehicle's CAN/OBD interface using the RS232 serial port.

In order to prove the adaptability of the chosen software solution a different model of Single Board Computer have been also used: PC Engines ALIX 2D2, which is equipped with an x86 processor - AMD Geode LX800 500 MHz, and 256 MB DDR memory. The Board is equipped with an RS232 serial port, two miniPCI interfaces, two 100 Mb/s Ethernet ports, one CF card socket, two USB 2.0 interfaces. The technical specifications are similar to the previous development card, and allows connectivity with the same external devices. Platforms were equipped with two different miniPCI wireless WiFi cards, the first operates within the 5.9 GHz frequency band and the second within the 760 MHz band [7, 8].

However, to install specific and dedicated IVC applications, another OS must be compiled and installed. In our previous work [7] we used Chaos_Calmer version of OpenWRT, an open-source Linux based Embedded OS that is compatible with both types of platforms. Additionally, Embedded OS packages for deploying 802.11p standard were cross-compiled on a performance host system - UBUNTU 14. The modifications for the system to operate according to the standard 802.11p were based on existing 802.11a software packages and drivers. These modifications are: radio card operates on the new frequency band 5.9 GHz with 10 MHz bandwidth, and the implementation of the new medium access mode OCB - Out of Context Basic Service Set [9]. An appropriate network and transport protocol was introduced WSMP – Wave Short Message Protocol, according to IEEE1609.3 standard [10]. The changes are based on the 3.18 Linux kernel and Atheros wireless driver package. Wireless devices based on this driver, especially the ath5k or ath9K driver, can be modified by MAC software (softMAC)[11]. All these changes have been achieved with the resources and existing libraries in the package distribution OpenWRT.

IV. VANETZA IMPLEMENTATION

Vanetza [12] represents an open-source implementation of the ETSI ITS-G5 for Linux OS. To implement new communication protocols of ETSI, corresponding to the Transport&Network, Facilities, Security&Management stack layers, its software packages require modifications for Linux in Embedded Devices. There exists commercial equipment at experimental level, however they contain proprietary software modules and software updates are difficult to perform. So far (to the best of our knowledge) the following Open Source software packages have been developed: GeoNet Stack, OpenC2X and Vanetza. All three packages were compiled on Linux systems with enough hardware and software resources such as UBUNTU.

The GeoNet Stack [13] is an pack of applications written in JAVA, that requires Java Virtual Machine to be preinstalled, so it uses significant hardware and software resources, and cannot be implemented on an Embedded Linux OS, corresponding to the limited resources chosen platform. The OpenC2X [14] software package has been designed and written in C++, using a Linux distribution with sufficient hardware and software resources. It requires several more programs and libraries to be preinstalled, which are not usually found on Linux for Embedded Devices. This software package has not implemented all ETSI ITS-G5 protocols (without Security and GeoNetworking), but has implemented the interface with the vehicle's OBD-CAN.

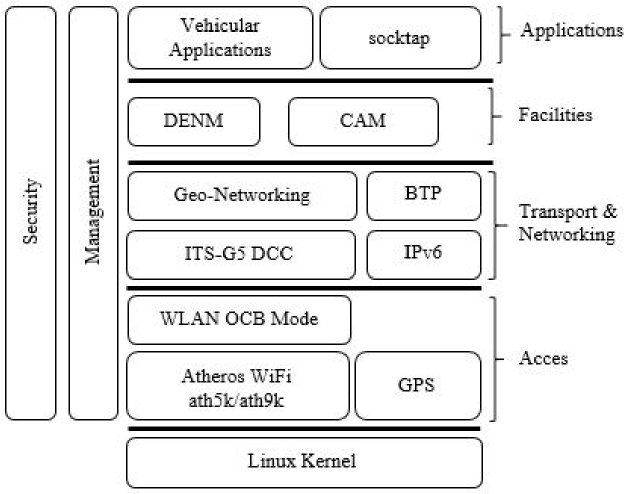


Figure 2. Vanetza architecture for ETSI ITS-G5 stack

Thereby, critical messages can be generated automatically by vehicle control system.

The Vanetza [12] package is also designed and built in C++ on a Ubuntu 16 OS and it has the following pre-requisites for compiling: Boost 1.58 (the math library), GeographicLib 1.37 (geographic coordinate conversion library), Crypto ++ 5.61 (class libraries for cryptographic schemes), OpenSSL (library for security applications). Because the latest version of the Linux kernel 4.4 brings important changes at the networking layer, Vanetza package was compiled under Ubuntu (with the latest version of the Linux kernel) and then it was necessary to cross-compiling the packages for the new distribution for LEDE OS (Linux Embedded Development Environment) [15]. This OS is derived from the OpenWRT and satisfies the following amendments allowing installation of new libraries of Vanetza. For full cross-compiling of Vanetza it was necessary to prepare in advance the new libraries to be integrated in LEDE. For validation and testing, Vanetza uses an external Framework module, GoogleTest. This module needs to be preinstalled before processing Vanetza. Libraries and test modules generated by Vanetza, correspond to the appropriate layers of ETSI ITS-G5: Transport&Networking, Facilities, Management&Security. These can be used in various vehicular communication applications.

The structure of the program modules is as follows: ***access*** – MAC layer interface, ***asio*** – radio modem interface, ***asn1*** - conversion of Abstract Syntax Notation One for CAM and DENM messages, ***dcc -*** DCC protocol implementation, ***gnss*** - GPS interface program, ***geonet*** - Geonetworking protocol program, ***btp*** – implementation of BTP, ***net*** – networking IPv6 protocol, ***facilities*** – Facilities layer implementation, ***security*** - management and security module, ***units*** - units' description (speed, acceleration, time).

A specific application developed (within the Tools directory) for testing Vanetza is ***socktap*** [12]. Another simple application developed for testing purposes is ***fake-feed***, used for periodic transmission of messages to desired IP address and UDP port. The Socktap application demonstrates the basic usage of Vanetza by transmitting raw packets on hardware interface, e.g. wireless. These are used for the validation of the implementation.

V. VALIDATION RESULTS

For cross-compiling LEDE we used the following procedure [16]. In the host operating system, Ubuntu was necessary to preinstall some packages for LEDE distribution kit. Next, we configure the kernel and Atheros wireless drivers to work according to the 802.11p standard: on physical layer, radio operates on 5.9 GHz with 10 MHz bandwidth, introduction of OCB mode on MAC Layer [7]. Then we prepared the necessary additional packages for cross-compiling Vanetza: *geographiclib*, *crypto ++,* which is not present on LEDE distribution. We checked the correct compilation of each package. We used two general configurations for cross-compile: one for Mikrotik board and the other for PC Engines board, as they have different processor class. We configure the rest of the necessary packages as exist on LEDE distribution kit: *boost* library, USB and GPS packages, and CAN package for RS232 connection to vehicle electronic bus. The list of generated libraries with their dimensions are presented in Table 1.

Table 1

Module Name	***Generated Library***	***dimension***
asn1	libvanetza_asn1.so	509.0 KB
dcc	libvanetza_dcc.so	47.9 KB
gnss	libvanetza_gnss.so	266.0 KB
Geonet	libvanetza_geonet.so	835 KB
btp	libvanetza_btp.so	266 KB
net	libvanetza_net.so	79.5 KB
facilities	libvanetza_facilities.so	22.1 KB
Security	libvanetza_security.so	968 KB
GeographicLib	libGeographic.so	1.5 MB
criptopp library	libcryptopp.so	6.3 MB
Gtest-main-library	libgtest_main.so	8.1 KB
Gtest-library	libgest.so	333.3 KB

Due to the dimensions of the generated binary code of ITS-G5 applications, the Embedded Systems must have enough memory resurses.

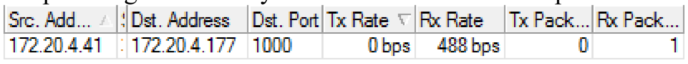
For validation and to prove the correct installation and running on LEDE, we use generated test applications of Vanetza. There are 68 test modules, all passed on both LEDE platforms. One test example is presented:

```
root@LEDE:/usr/lib# ./GTest_Time
Running main() from gtest_main.cc
[==========] Running 2 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 2 tests from Time
[ RUN      ] Time.clock_cast_from_quantity
[       OK ] Time.clock_cast_from_quantity (0 ms)
[ RUN      ] Time.clock_cast_from_chrono
[       OK ] Time.clock_cast_from_chrono (0 ms)
[----------] 2 tests from Time (0 ms total)
[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (0 ms total)
[  PASSED  ] 2 tests.
```

We use the test program fake_feed to generate periodic IPv4 UDP packets to an specific ip address destination and port:

root@LEDE:/usr# ./proxy_fake_feed 172.20.4.5 1000

Response generated by a Mikrotik router is presented :



Src. Add...	Dst. Address	Dst. Port	Tx Rate	Rx Rate	Tx Pack...	Rx Pack...
172.20.4.41	172.20.4.177	1000	0 bps	488 bps	0	1

Also we use Vanetza test application ***socktap*** to generate raw packets on wlan1, with GPS information :

```
root@LEDE:/# iw reg set RO
cfg80211: Regulatory domain changed to country: RO
cfg80211: DFS Master region: ETSIcfg80211: (5850000 KHz - 5925000KHz @ 20000 KHz), (N/A, 2000 mBm), (N/A)
root@LEDE:/# iw wlan0 set type ocb
root@LEDE:/# ip link set wlan1 up
goto…local->ocbs++ ieee80211 recalc idle: active:1;
local->ocbs:1 netif carrier ok!
br-lan: port 3(wlan1) entered forwarding state
```

```
root@LEDE:/# iw wlan1 ocb join 5890 10MHZ
root@LEDE:/# /tmp/./socktap -I wlan1
Will use ethernet device:wlan1 and live GPS data
runtime clock is now at 2017-Feb-27 07:23:37.091984
packet sent with 50 bytes length
Latitude:47.1654,      Longitude:27.5877,      Heading:215.471,
Speed:0.0319222, Time:1.48818e+09
runtime clock is now at 2017-Feb-27 07:23:39.095298
packet sent with 61 bytes length
```

The overall conclusion is that Vanetza runs correctly on LEDE for Embedded devices.

VI. Conclusions

Through this work we demonstrate the possibility of implementing the ETSI ITS-G5 IVC standard on low cost embedded device using open-source Linux distribution for embedded devices.

VII. References

[1] D. Eckhoff, N. Sofra, and R. German, "A performance study of cooperative awareness in ETSI ITS G5 and IEEE WAVE," in *Wireless On-demand Network Systems and Services (WONS), 2013 10th Annual Conference on*, 2013, pp. 196-200.

[2] ETSI, "EN 302 663 (V1. 2.1)(11-2012):" Intelligent Transport Systems (ITS)," *Access layer specification for Intelligent Transport Systems operating in the,* vol. 5.

[3] A. Festag, S. Kühlmorgen, and N. Maslekar, "Decentralized Congestion Control for Multi-hop Vehicular Communication," *23rd ITS World Congress, Melbourne, Australia, 10–14 October 2016,* 2016.

[4] V. Sandonis, I. Soto, M. Calderon, and M. Urueña, "Vehicle to Internet communications using the ETSI ITS GeoNetworking protocol," *Transactions on Emerging Telecommunications Technologies,* 2014.

[5] ETSI, "TS 102 636–3 V1. 1.1 (2010–03): Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 3: Network architecture," *European Telecommunications Standards Institute,* 2010.

[6] ETSI, "TS 102 636-5-1 V1. 1.1 Intelligent Transport Systems (ITS)," *Vehicular communications,* 2011.

[7] A. Abunei, C.-R. Comşa, and I. Bogdan, "Implementation of a Cost-effective V2X hardware and software platform," in *Communications (COMM), 2016 International Conference on*, 2016, pp. 367-370.

[8] J. Heinovski, F. Klingler, F. Dressler, and C. Sommer, "Performance comparison of IEEE 802.11 p and ARIB STD-T109," in *Vehicular Networking Conference (VNC), 2016 IEEE*, 2016, pp. 1-8.

[9] R. Lisový, M. Sojka, and Z. Hanzálek, "IEEE 802.11 p Linux Kernel Implementation," 2014.

[10] H.-J. Jeong and S. Kwon, "How to begin V2X development on Linux," *Automotive Linux Summit 2015 JAPAN,* 2015.

[11] M. Vipin and S. Srikanth, "Analysis of open source drivers for IEEE 802.11 WLANs," in *Wireless Communication and Sensor Computing, 2010. ICWCSC 2010. International Conference on*, 2010, pp. 1-5.

[12] R. Riebl. (2017). *Vanetza, Open-source implementation of the ETSI ITS-G5 protocol stack.* Available: https://github.com/riebl/vanetza

[13] A. Voronov. (2017). *ETSI ITS G5 GeoNetworking stack, in Java: CAM-DENM / ASN.1 PER / BTP / GeoNetworking* Available: https://github.com/alexvoronov/geonetworking

[14] S. Laux, G. S. Pannu, S. Schneider, J. Tiemann, F. Klingler, C. Sommer, *et al.*, "Demo: OpenC2X–An Open Source Experimental and Prototyping Platform Supporting ETSI ITS-G5."

[15] LEDE. (Online). *The LEDE Project ("Linux Embedded Development Environment").* Available: https://lede-project.org/

[16] OpenWRT. (Online). *OpenWrt build system – Installation.* Available: https://wiki.openwrt.org/doc/howto/buildroot.exigence