Relaying Controller Area Network Frames over Wireless Internetworks for Automotive Testing Applications

Mathias Johanson
Alkit Communications
Mölndal, Sweden
mathias@alkit.se

Lennart Karlsson
Dept. of Computer Aided Design
Luleå University of Technology
Luleå, Sweden
lennart.karlsson@ltu.se

Tore Risch
Dept. of Information Technology
Uppsala University
Uppsala, Sweden
tore.risch@it.uu.se

Abstract— **In this paper we describe how Controller Area Network (CAN) frames can be relayed over a wireless Internet connection, enabling remote access to the CAN buses of vehicles for applications in automotive telematics. This opens up many new possibilities for diagnostics, monitoring, testing, analysis and verification, which we believe can significantly reduce the time required for the testing and verification phases of automotive development. A CAN-over-IP tunneling protocol is described and the design and implementation of a generic system for remote access to the CAN bus of vehicles is presented. Examples of applications of the technology are given and implications in terms of new possibilities and challenges in automotive engineering are discussed.**

Keywords-CAN; automotive telematics; wireless networks

I. INTRODUCTION

For many years, the Controller Area Network (CAN) [1] protocol has been an important standard for communication between Electronic Control Units (ECU) in vehicles, and for communication with external equipment for testing, analysis, diagnostics, calibration and software updates. Since a modern car can have as many as 50 ECUs, having all units connected to a communication bus reduces the need for wiring in the vehicle. The CAN bus also provides a standardized way to connect external testing equipment to the vehicle for monitoring, diagnosing and analyzing the vehicle. Although new and improved automotive communication technologies have emerged, such as FlexRay [2], the CAN bus will continue to be of great importance in the automotive industry for years to come.

When the electronics of a vehicle is tested using a CAN analysis tool, an instrument is connected to the external CAN bus connector, typically a standardized J1962 connector. All traffic on the bus can then be tapped, and the communication between the ECUs can be analyzed. Special diagnostics messages can also be inserted on the bus, instructing ECUs to report status information. ECU calibration and software updates can also be done using the CAN protocol.

With the advent of wireless communication technologies, such as GPRS, WLAN and 3G networks, an opportunity has emerged for interconnecting the CAN bus in a vehicle with the Internet, enabling remote access to the ECUs from virtually anywhere. This makes it possible to design applications such as remote diagnostics, remote CAN bus analysis, remote ECU calibration and remote ECU software download. In order to accomplish this, a protocol is needed to transport CAN frames over an internetwork, and a gateway functionality is needed to relay the frames between the CAN network and the Internet. This paper describes one such approach.

A. *The Controller Area Network Protocol*

CAN is a broadcast serial bus technology, which means that each connected unit receives all transmitted messages, and that only one unit can transmit at a time. Each unit on the bus is identified by a CAN id (11 or 29 bits long depending on whether extended addressing mode is used), that is contained in the header of every CAN frame transmitted by the ECU. If two or more units begin transmitting on the bus simultaneously, a priority scheme based on dominant bits in the unit identifiers will determine which unit will continue transmission and which will back off for subsequent retransmission. The maximum size of a CAN message is 12 bytes, out of which 0 to 8 are payload data. The maximum data rate is 1 Mbps (CAN 2.0 B specification).

B. *The Internet Protocol*

The Internet Protocol (IP) is the network protocol for packet switching on the Internet. It is based on a best-effort service model of packet delivery with reliability being the responsibility of higher level protocols (typically TCP). Communication endpoints are identified by 32-bit IP addresses. IP packets are of variable length, with typical packet sizes in the order of 1500 bytes.

Generally, an application uses a transport protocol on top of IP, the most common being the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP is connection oriented and reliable, whereas UDP is connectionless and unreliable.

C. *Relaying CAN frames over IP*

Although CAN is a level 2 (data link layer) protocol and IP is a level 3 (network protocol), with respect to the OSI reference model, there is nothing that prevents encapsulation of CAN frames in IP packets for transmission over the Internet. This can be done using a simple tunneling protocol, such as the one proposed in section II below. Using this approach, a gateway node connected both to a CAN bus and an IP network relays CAN frames between an ECU on the

CAN bus and a remote IP host. The situation is illustrated schematically in Figure 1.

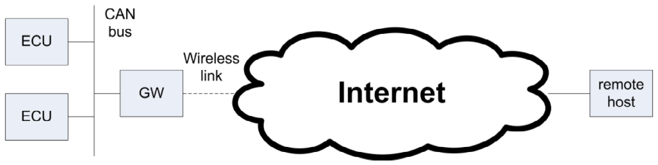

Figure 1. The client application on a remote host (right) communicates over an internetwork with the ECUs connected to the CAN network (left), via a gateway node (GW)

The easiest way to relay CAN frames over IP is by encapsulating them in UDP datagrams for transmission over an IP connection to a tunnel endpoint, where they are decapsulated and forwarded. However, since UDP is unreliable, CAN frames can be lost without notice and this can be difficult to handle by the application. To detect losses, sequence numbers can be used. Given the real-time nature of many CAN applications, a good option would be to use the Real-time Transport Protocol (RTP) over UDP for encapsulation of CAN frames. RTP provides sequence numbers (for loss detection) and timestamps that can be used to calculate packet jitter and delays. However, RTP does not provide reliability, so packet loss must still be handled by the application, although the loss detection is provided by the transport protocol. Another option is to use TCP, which supports reliable data transfer using acknowledgements and retransmissions. This is the approach chosen for the implementation of the prototype system described in this paper.

Since there are no guarantees on end-to-end delays and jitter on IP networks (unless specific QoS techniques are being used), applications that require a fixed transmission delay between sender and receiver are not generally feasible. Despite this, a number of interesting automotive applications can be realized using traditional best-effort Internet connections, as described in section III.

D. *Related work*

In this paper, we propose tunneling of CAN messages over IP, in order to provide Internet access to ECUs connected to a CAN network. Another approach explored by Ditze et al. [3] is to implement IP over CAN, whereby an IP address is mapped to a CAN identifier, and the IP packets are fragmented into a number of consecutive CAN frames. Lindgren et al. [4] suggest a similar approach and present a generic modular lightweight IP stack implementation suitable for embedded platforms. Although interesting, these approaches assume that the ECUs are IP capable devices, and most ECUs in use today are not. Instead we propose an approach where a gateway device encapsulates and relays CAN frames over an IP connection to an application that communicates using a well-defined Application Programming Interface (API) for reading and writing CAN frames. With this approach, standard ECUs can be accessible remotely, without alteration. Moreover, many automotive engineering tools for diagnostics, CAN analysis and software download can be used without modification, since the CAN-over-IP tunnel will be transparent to the applications.

Similar to our approach, Bayilmis et al. [5] describe a system based on what they call a Wireless Internetworking Unit for interconnecting a CAN bus with an IEEE 802.11b WLAN using an encapsulation technique. Contrary to our work, it is a data link level encapsulation instead of our IP level encapsulation, and it is unclear whether the system was ever implemented in practice or only simulated.

II. THE CAN OVER IP TUNNELING PROTOCOL

The protocol we have designed for tunneling CAN frames over IP connections is based on a generic message format that is shown in Figure 2. The messages can be transported over UDP, UDP/RTP or TCP, depending on whether the application can handle packet loss or not. For our prototype implementation, described in section IV, we have used TCP. In case a datagram based transport protocol (e.g. UDP) is used, the message format allows aggregation of several messages into the same datagram.

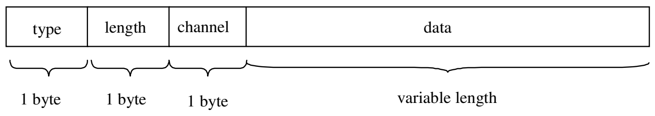

Figure 2. Message format for CAN-over-IP tunneling protocol

A message consists of a fixed length (three bytes) header and a variable length payload data section. The header consists of a one byte type specifier that defines what type of data is in the payload, a one byte length field giving the length of the payload data, and a one byte channel identifier, used to convey which CAN bus the CAN frame in the payload is intended for. (Many vehicles have more than one CAN bus.) Using this general message format, a wide range of distributed CAN applications can be implemented. In addition to carrying the CAN data frames to be tunneled, messages are also defined for installation of filters, defining which CAN frames should be relayed and which should not, for periodic transmission of CAN frames, and for setting the CAN bus's baud rate. A list of currently defined message types is given in Table I below.

Since the length field is only eight bits, the longest payload that can be carried is 255 bytes. This is not a problem for relaying CAN frames, since these have a maximum length of 12 bytes. However, since the tunneling protocol can be useful also for relaying other data, such as software updates to ECUs, an extension mechanism has been designed, enabling the length field to be two bytes long whenever bit seven of the type identifier is set (i.e., message types with id>127).

For relaying of CAN data, the packetization overhead of the protocol is considerable (20%). However, compared to the overhead imposed by TCP/IP packetization (40 bytes per packet) it is not a big issue.

TABLE I. CAN-OVER-IP PROTOCOL MESSAGE TYPES

Name	Value	Description
CAN_ FRAME_11BIT	0	The data portion of the message contains an 11 bit CAN frame to be relayed.
CAN_ FRAME_29BIT	1	The data portion of the message contains a 29 bit CAN frame to be relayed.
CAN_PERIODIC_ FRAME_11BIT	2	The data portion contains a CAN frame to be relayed and then transmitted periodically on the specified CAN bus. The transmission interval is contained in the data segment.
CAN_PERIODIC_ FRAME_29BIT	3	Send periodic 29 bit CAN frame.
CAN_CANCEL_ PERIODIC_FRAME	4	Cancel the transmission of periodic frames to the CAN bus.
CAN_FILTER_PASS	5	Install pass filter for CAN frames on the specified channel.
CAN_FILTER_BLOCK	6	Install block filter for CAN frames on the specified channel.
CAN_FILTER_FLOW_ CONTROL	7	Install flow control filter for CAN frames on the specified channel.
CAN_CANCEL_ FILTER	8	Uninstall a CAN filter.
CAN_STATUS_ REQUEST	9	Request device status.
CAN_STATUS_ REPORT	10	Report device status.
CAN_SET_DATA_ RATE	11	Set the data rate on the CAN bus.
CAN_DISCONNECT	12	Terminate CAN-over-IP session.
CAN_NACK	13	Negative response to CAN query.
CAN_FLUSH	14	Flush the CAN bus's receiver buffer.
CAN_SW_UPDATE	128	Download new software to the WCU.

III. APPLICATIONS

We believe that many automotive CAN application that traditionally require physical access to the vehicle can be realized over a network, using a CAN-over-IP tunneling mechanism such as the one described in this paper. In general, all applications where timing concerns are not too critical can be implemented straightforwardly. This includes both one-way "listen only" applications (e.g. CAN bus monitoring), and two-way applications (e.g. diagnostics). As a matter of fact, in many situations where we originally believed that timing issues would present problems, we have found that the low latency requirements can in fact be relaxed without problems.

Some of the automotive applications we have investigated in more detail are discussed below.

A. *Remote Diagnostic Read-out*

In automotive diagnostics, the ECUs of the vehicle being tested are queried for status and statistics data. Specifically, Diagnostic Trouble Codes (DTC), identifying malfunctions or anomalies in system components are read out, translated into a human-readable form and presented for the engineer for troubleshooting purposes. The DTCs are frequently also collected in a centralized database storage for offline statistical analysis.

In a remote diagnostic read-out scenario, an embedded system with a wireless network interface and one or more CAN interfaces is connected to the J1962 port of the vehicle. When a diagnostic read-out is to be effected, the device connects over the wireless internetwork to a server that starts sending diagnostic CAN queries over the CAN-over-IP tunnel. The CAN frames are decapsulated by the gateway device and inserted on the appropriate CAN bus in the vehicle. The ECU that the query is intended for responds with CAN messages that are relayed back to the server. When all diagnostic queries have been performed, the connection is closed and the DTCs are uploaded to the database for storage and analysis.

For a more in-depth description of the remote automotive diagnostics system, see Johanson and Karlsson [6].

B. *Remote CAN bus monitoring and analysis*

When a specific problem with a vehicle is investigated, a CAN analysis equipment can be connected to the CAN bus of the vehicle and the traffic on the bus can be analyzed. Examples of such tools are CANalyzer and CANgraph from Vector. Filters can be designed to select the interesting frames. The CAN data can be visualized in a number of ways depending on what kind of data it is.

With a CAN-over-IP relay device connected to the car, all traffic can be sent to a remote destination for real-time analysis. At the remote host, a virtual CAN driver can be installed, making it possible to use the standard CAN analysis tools without modification. It is also possible to inject CAN frames onto the remote CAN bus.

C. *ECU software download*

Many problems with modern vehicles are due to software bugs in ECUs. Once a bug has been fixed, and a new version of the ECU software is available, it can be downloaded to the vehicle. With a CAN over IP protocol, this can be done remotely over an Internet connection.

D. *Automated CAN data analysis using stream query processing*

With the decreasing cost of embedded systems and the increasing availability of wireless access networks, we can foresee a future where large fleets of test vehicles are monitored using wirelessly interconnected devices. This will result in large volumes of monitoring data, which will require automated analysis tools that assist the engineers in processing the data. To some extent, such analysis can be performed locally in the embedded systems, but for analysis requiring more demanding computations the analysis will have to be done at a central data processing site. Moreover, cross-correlation analyses involving many simultaneous streams can be performed at a centralized site, or statistical analyses of data collected over longer time periods from many vehicles.

For these applications, we are currently designing a system that uses a Data Stream Management System

(DSMS) [7, 8] for specifying the desired computations over the data streams and the required filtering based on these computations. The filters are expressed as *continuous queries* that filter the streams emitted from the ECU. In our approach, the continuous queries will contain calls to predefined *filter functions* that the engineer can use when searching on-line measurement data. Such filter functions are implemented beforehand and registered for the DSMS. The DSMS provides means for executing these operators in parallel at different locations of the distributed system to obtain acceptable performance. The filters can be adaptively refined, based on feedback from the analysis of the streams.

We believe a DSMS based data analysis system can be a very powerful general data analysis and problem solving tool for automotive engineers in the future.

IV. Prototype system implementation and experiments

In a research and development project striving to improve efficiency in automotive testing, we have designed and implemented a system for transparently relaying CAN frames over a wireless Internet connection from a Wireless Communication Unit (WCU) that is connected to the CAN buses of a vehicle to a remote host running a CAN application. The WCU is based on an embedded system running the Linux operating system. It has two CAN ports, an Ethernet interface, an IEEE 802.11b WLAN interface, and a GPRS/EDGE wireless network interface. The unit is shown in Figure 3.

Figure 3. The Wireless Communication Unit

A software component, known as *cantunnel*, running on the WCU, implements the CAN-over-IP protocol. At the other end, a specialized middleware component implements the remote end of the CAN-over-IP tunnel, allowing a CAN application, such as a diagnostic read-out, or software download application to send and receive CAN frames through a standardized CAN API known as SAE J2534 [9]. Actually, for practical reasons, such as NAT firewall traversal issues, cantunnel does not connect directly to the middleware component, but rather via a proxy server. For applications not supporting the J2534 API, a virtual CAN device driver can be installed, that exposes a CAN device driver API (such as CANlib from Kvaser) to the application, while communicating using the CAN-over-IP protocol with the WCU. The client application side of the system is illustrated in Figure 4.

The prototype has been successfully tested for diagnostic read-out and CAN monitoring applications, as described in section III. When using the WLAN port for communication, two simultaneous CAN channels of 500 kbps and 125 kbps respectively can be monitored remotely without filtering. When used for diagnostics, CAN software filters are typically installed, which restricts the wireless communication to only the desired CAN frames, i.e. the answers to the diagnostic CAN queries. This can be realized over a narrowband GPRS connection.

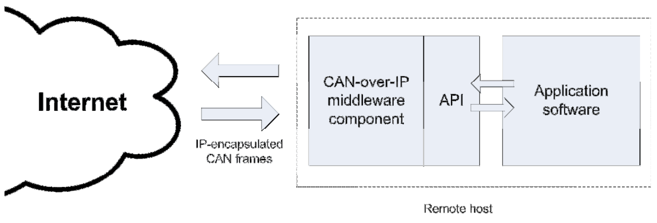

Figure 4. Schematic illustration of the remote host part of the system

A. *Performance optimizations*

Since end-to-end delays over a wireless Internet connection can be substantial (for a GPRS connection up to a few seconds), special consideration must be given to situations where time-critical CAN interactions are required. One such issue is the flow control mechanism for segmented data transfer when using the ISO15765-2 protocol [10] in combination with the J2534 API. This mechanism requires that the first frame of a segmented message (i.e. a message spanning several CAN frames) is responded to by a so called Flow Control frame, containing the block size and separation time for subsequent frames. This response is expected to arrive within a 1 second timeout period, and since the round trip of a congested GPRS connection can sometimes be larger than this, it is necessary to handle the response locally in the WCU. Even if the round-trip delay is less that 1 second, it is nevertheless a performance optimization to handle the flow control locally in the WCU.

Another performance optimization for query/response CAN applications (e.g. diagnostic read-out) is to use a negative acknowledgement (NACK) scheme to signal that there was no reply to a CAN query. Generally, if a diagnostic CAN query is not responded to in less than 100 ms, there will not be a response. (Diagnostic CAN messages have the highest priority on the CAN bus and will hence be swiftly responded to.) However, with a high latency wireless Internet connection between the client application and the CAN bus, a much longer timeout value will have to be used, since the worst case response time for a query is long, due to link congestion. If there are many queries without responses, such timeouts will impact the total read-out time significantly. By sending a NACK after 100 ms, the latency imposed by a query without a response will be close to the average response time instead of the worst-case response time, which turns out to be a huge performance gain in practice.

Yet another performance optimization that we have found to be of significant benefit is related to the "flush" operation, when the application tells the CAN driver that it wants to discard all CAN frames received and buffered by the driver. In the CAN-over-IP situation, this is not so simple since the flush message must be propagated over the network to stop relaying CAN frames from the remote end. While the flush message is en route, many new CAN frames can be transmitted, and these frames must be discarded upon reception. Since the end-to-end delays can be long, this means that a worst-case round-trip delay timer must expire before we can be sure that no frame that should be discarded is still in transit. If the application performs many flush operations, this will impact performance significantly. The solution is to use a flush bit in the message header (we have used bit 6 in the type field of the CAN-over-IP protocol header) to indicate that a flush operation has been effected. When the flush bit has been set, all subsequently received frames are discarded until one arrives that also has the flush bit set. The remote CAN-over-IP endpoint inspects the flush bit, and if it is set, the flush bit of the next outgoing frame will be set. Upon arrival, this frame with the flush bit set, and subsequent frames, will not be discarded. Since the flush operation is very common is many CAN applications, this optimization can have a huge performance impact.

B. Safety and security issues

Enabling access to the CAN buses of vehicles from the Internet has a number of safety and security implications that need to be addressed. First and foremost, the wireless communication units must be protected from malicious access by unauthorized users. Failure to do so could enable a "hacker" to gain control not only of the WCU, but also of the vehicle, with dire safety consequences for a driver or passenger. Secondly, the potentially sensitive data transported over the communication link must be protected from illicit eavesdropping.

In our prototype systems, the WCUs running Linux are protected by software firewalls configured to let only the wanted traffic through. By the use of Network Address Translation and private IP addressing, all network communication must be initiated from the WCU, which makes malicious access much harder. Moreover, access control based on public key encryption is being used to authenticate remote endpoints.

To protect sensitive engineering data communicated over the easily eavesdropped wireless Internet links, strong application level encryption based on the Blowfish algorithm [11] is being used.

V. Conclusion and Future Work

In this paper we have presented a novel CAN-over-IP tunneling protocol and described the design and implementation of a system supporting transparent relaying of CAN frames over TCP/IP. A number of automotive applications have been identified for which remote operation is feasible. Experiments with the prototype implementation for automotive diagnostics and CAN bus monitoring serves as a proof-of-concept and highlights the potential of the technology for shortening automotive development cycles by making the testing and verification stages more efficient. Our experiments show that applications hitherto believed to be unfeasible over a variable-delay internetwork, can indeed be realized by relaxing the time constraints. Moreover, many of the performance problems associated with substantial propagation delays over narrowband wireless Internet connections can be overcome through optimizations of the CAN interactions between the endpoints.

Our current focus is on integrating DSMS functionality into the system, enabling a generic platform for online measurement data processing and analysis. This will make it possible to design massively distributed automated measurement data acquisition, communication, processing and analysis tools for the next generation automotive engineering applications.

Acknowledgment

This work was supported by the Centre for Automotive Technology and Testing (CASTT) at Luleå University of Technology and by VINNOVA through the iStreams project.

References

[1] ISO International Standard 11898 "Road vehicles – Controller Area Network (CAN) – Part 1: Data link layer and physical signalling," 1993.

[2] R. Shaw and B. Jackman, "An introduction to FlexRay as an industrial network," IEEE International Symposium on Industrial Electronics, Cambridge, United Kingdom, July 2008.

[3] M. Ditze, R. Bernhardi-Grisson, G. Kämper and P. Altenbernd, "Porting the Internet Protocol to the Controller Area Network," 2nd International Workshop on Real-time LANs in the Internet Age (RTLIA 2003), Porto, Portugal, July 2003.

[4] P. Lindgren, S. Aittamaa and J. Eriksson, "IP over CAN, transparent vehicular to infrastructure access," 5th IEEE Consumer Communications and Networking Conference, January 2008.

[5] C. Bayılmış, İ. Ertürk and C. Çeken, "Extending CAN segments with IEEE 802.11 WLAN," The 3rd ACS/IEEE International Conference on Computer Systems and Applications, Egypt, January 2005.

[6] M. Johanson and L. Karlsson, "Improving vehicle diagnostics through wireless data collection and statistical analysis," IEEE International Symposium on Wireless Vehicular Communications, Baltimore, MD, USA, October 2007.

[7] E. Zeitler and T. Risch, "Using stream queries to measure communication performance of a parallel computing environment," Proc. First International Workshop on Distributed Event Processing, Systems and Applications (DEPSA), Toronto, Canada, June 29, 2007.

[8] G. Gidofalvi, T.B. Pedersen, T. Risch and E. Zeitler, "Highly scalable trip grouping for large scale collective transportation systems," Proc. 11th International Conference on Extending Database Technology, EDBT 2008 , Nantes, France, March 2008.

[9] SAE International Standard J2534-1, "Recommended Practice for Pass-Thru Vehicle Programming", December 2004.

[10] ISO International Standard 15765-2, "Road vehicles – Diagnostics on Controller Area Networks (CAN) – Part 2: Network layer services," 2004.

[11] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," Proceedings of the 1993 Cambridge Security Workshop, 1994.