Porting the Internet Protocol to the Controller Area Network

Michael Ditze[1], Reinhard Bernhardi[2], Guido Kämper[1], Peter Altenbernd[2]

[1]*University of Paderborn / C-LAB, Fürstenallee 11, 33094 Paderborn*
[2]*Siemens Business Services / C-LAB, Fürstenallee 11, 33094 Paderborn*
{Michael.Ditze, Reinhard.Bernhardi, Guido.Kaemper, Peter.Altenbernd}@c-lab.de

Abstract

Dedicated non-standardized platforms and software environments require to deploy middleware solutions in distributed heterogeneous systems. In order to mask this heterogeneity and guarantee application interoperability, the middleware must provide interplay between communication technologies concealed behind a common service interface. The Internet Protocol (IP) has proven to be a promising higher level information carrier that may serve as such a service. Having been ported to various network technologies, it allows for Internet access of associated devices and serves as a foundation for many middleware approaches. Convergence trends between the IP world and the automotive and factory automation domain require to port IP to the Controller Area Network (CAN) mostly featured in these domains. As the requirements for packet delivery as imposed by the IP protocol distinguish significantly from the transmission mechanisms provided by the CAN protocol, this paper discusses porting challenges and presents a new and efficient approach for transmitting full IP packets via CAN that enables enhanced and innovative high-level applications on CAN nodes

1. Motivation

Various platforms and operating systems, diverse network media and protocols as well as different programming languages make the deployment of middleware approaches that guarantee interoperability inevitable in heterogeneous distributed systems. One of the major design issues to mask the inherent heterogeneity is to provide a standard set of interfaces and services that distributed applications can assume present. One of these services is a homogeneous communication protocol. The Internet Protocol (IP) with its increasing significance in an Internet converging world proves to be a promising candidate that may serve as a higher level information carrier for various networks, e.g. Firewire [5] and Bluetooth [1]. Many available middleware approaches use IP as a standard communication interface, e.g. Corba, Jini or Java.

Serial bus systems as the Controller Area Network (CAN), on the other hand, have mainly neglected the use of IP. Consequently, there are only few approaches that would allow distributed IP- or web-based applications like diagnostic services and software upload to connect to low cost-devices as often used in the industrial automation and car domains. The need for these applications, however, increases.

CAN is one of the most commonly used network media in these domains. The CAN protocol in its current version 2.0B is an international standardized [4] scalable serial bus communication system originally developed for automotive in-vehicle networks. It is increasingly deployed in embedded control environments, e.g. factory automation and medical control. The major benefits of CAN that explain its widely use are its reliability, real-time behavior, multimaster capabilities, broadcast messaging and most significantly, its cost effectiveness. Most of these features, however, contradict the requirements as imposed by IP employed on top of CAN.

This paper describes how IP-based services can be adapted to CAN networks through communication middleware. It identifies the major challenges and introduces a new service component, the Internet Gateway Service (IGS), responsible for transforming full IP packets to CAN messages. To our knowledge, this is the first approach to transmit and re-assemble IP packets on CAN. It allows for enhanced innovative high-level applications, e.g. web-based diagnostic services, to be ported to CAN nodes or remote clients that access internal automotive status information. It also enables remote software installation by adjusting automotive standard communication to the IP world. The approach has been developed within the European ITEA East-EAA-project that aims to standardize the future automotive middleware, by featuring a wide scale of car manufactures and tool suppliers[1].

[1] East-EAA is an ITEA Eureka international project funded by the German Bundesministerium für Forschung und Wissenschaft, http://www.east-eea.net

The rest of this paper is organized as follows: Section 2 gives a short introduction on CAN and IP and identifies the porting challenges. Section 3 summarizes related work. The new approach for transmitting IP packets over CAN is presented in Section 4, followed by a prototype description. Section 6 concludes with a summary and future work.

2. Introduction and Problem Description

CAN uses a message oriented communication protocol that allows for broadcast communication. Messages are identified through a unique 29-bits extended header that serves as a message identifier that simultaneously defines the content and the priority of the message. Whereas the priority of a message is required for bitwise deterministic CSMA/CA arbitration in case of multiple simultaneous sending entities, the receiving devices require the content description for message filtering. Likewise, CAN transmission occurs in broadcast mode to any node in the subnet that then filters messages according to its content description. The deterministic arbitration mechanism makes CAN real-time capable [6]. The CAN 2.0B protocol allows to send a maximum transmission unit (MTU) of 8 data-bytes of payload at a maximum data-rate of 1 Mbit/s.

IP is part of the TCP/IP protocol suite that allows cooperating devices to share resources across the network. It resides above the CAN protocol on the network layer of the ISO/OSI reference model. Unlike the message-oriented CAN, IP is connection-oriented and network entities are identified through a unique 4 byte IP address that allows for one-to-one, multicast or broadcast communication through respective router support. In contrast to CAN that usually communicates low-sized control messages, IP targets at transmitting high data rate bursts commonly found in ftp- or multimedia data transfers. The MTU of IP packets consequently amounts to 1500 bytes where IP packet headers are as large as 12 bytes, thus composing additional information overhead compared to the 29-bits extended CAN header. In its current most widespread version IPv4 does not support priority assignment to messages and Ethernet CSMA/CD arbitration prevents real-time transmission.
Consequently, porting IP packets to the CAN protocol imposes the four following challenges:

i. Adjustment of connection oriented IP transmission to message oriented CAN communication
ii. Real-time enabling priority assignment to IP packets to suit CAN arbitration mechanisms that must not interfere other system real-time traffic.
iii. Fragmentation of large IP data packets to suit the CAN MTU
iv. Decrease of IP header overhead for more efficient resource utilization

3. Related Work

Only few approaches have addressed the complexity of porting IP to CAN, recently.

Cach and Fiedler [3] presented an approach to allow CAN connected devices access the Internet. CAN nodes are dynamically assigned IP addresses using DHCP. The CAN message ID is filled with basic information relevant for IP packet delivery allowing for dedicated point-to-point communication with minor priority assignment. However, as almost all 29 bits in the extended messages header are used for IP restricted information, messages IDs become quite inflexible and may easily interfere other predefined real-time system traffic. Moreover, the approach supports flow control only for messages less than 128 bytes, whereas IP packets may easily amount to 1500 bytes. The service gateway for external communication remains undefined. This paper presents an Internet Service Gateway that leaves some system flexibility for header definition, hence minimizing the risk of disturbing common CAN traffic. Furthermore, full flow control to IP messages is provided.

Beveridge examined the possibility of porting Jini to the CAN [2], identifying the need to transmit IP messages thereon. Consequently, the approach modifies CAN message IDs to suit Jini control and configuration messages, providing a 20 bits priorization field, a 3bits Jini message ID field and a 6 bit sender ID field in the CAN header. Jini message types and destination IDs are transmitted within the CAN data payload. The definition of Jini message types requires a lot of overhead at the expenses of reduced data payload. Furthermore, each node needs to check the data field to determine if a message was destined for him. This requires to first copy the content to the nodes´ data buffer and causes additional overhead. The approach presented in this paper only requires one byte of data payload for message sequencing, thus increasing the transmission efficiency. Furthermore, each node may determine the destination of a message directly from the message header, reducing unnecessary data buffer copying.

4. IP over CAN

This section describes how IP packets can be mapped to CAN messages for transmission via the CAN network. We achieve this by building an Internet gateway service (IGS) implemented on a CAN node in the network. The IGS addresses the four challenges identified in Section 2.

4.1. The Internet Gateway Service (IGS)

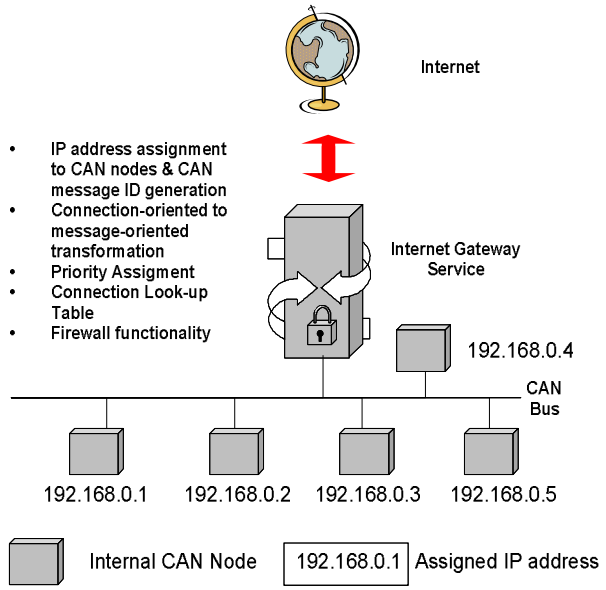

Figure 1: The Internet Application Gateway

We developed the IGS as the central part of the new CAN integrated middleware service. It receives and transmits external IP packets and transforms them into CAN specific messages that are then forwarded to the respective CAN node. Unlike other approaches, the IGS also transforms the IP packet header such as IP packets can be fully re-assembled at the receiving CAN node.
This is particularly useful for IP based applications to be executed on CAN nodes, e.g. remote diagnostic web-based services or software upload.

However, implementing IGS on other CAN nodes in the network also allows these nodes to internally interchange IP packets. This is required for IP-based middleware approaches with automatic device detection and configuration, e.g. Jini. The IGS is also responsible for shielding the internal CAN network from the heterogeneous Internet world. Hence, it may also support firewall functionality to protect the CAN network from malicious applications. The IGS needs to address the challenges as identified in order to provide proper IP packet transformation. Fig.1. shows how the IGS fits into CAN.

4.2. Transforming Connection-oriented to Message-oriented Communication

As identified in Section 2, the IGS needs to transform the connection-oriented IP packet delivery mechanisms to suit the message-oriented approach deployed by the CAN 2.0B specification
In order to realize connection-oriented communication on CAN, we logically assign each CAN node an IP address in a first step. We assume that this can be done either statically at compilation or dynamically by the application gateway using a suitable version of the DHCP protocol. However, as currently available CAN transceivers impose a theoretical limit of 128 nodes on a CAN network [3], we require at most the last 8 bits of the 4 byte IP address to identify a particular CAN node. In order to make this dedicated node receive IP packets destined for him, the message filter of this particular node is set such as it accepts messages with IDs containing its 8 bit long identification. Other nodes that do not have their filters set to this identification will reject this message, resulting in the dedicated point-to-point communication as required by IP.

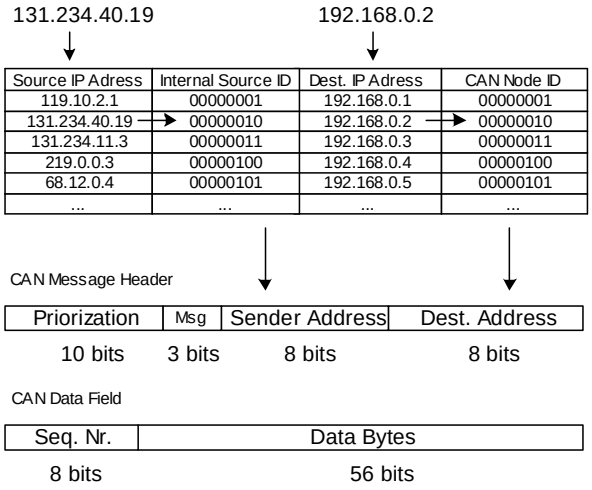

Figure 2:CAN Data Format for IP Transmission

In order to allow the Internet Gateway to route messages to the receiver, the address transformation is also stored in a corresponding connection table in the IGS (see Fig.2). Upon packet reception, the IGS strips the last byte of the destination IP off and identifies the respective CAN node through a table look-up. Similarly, the source IP address is stored in the table and mapped to an internal identification number ranging from 128 to 255. The first 128 integers are reserved for internal CAN communication purposes.

In a next step, the IGS transforms the IP packet header and content into a series of CAN messages to be accepted by the respective CAN node, only. Therefore, the IGS generates a new 29 bit comprising CAN message ID where the 8-bit identification builds the last part of the message ID. This part is preceded by the 8 bits source ID, allowing the receiver to uniquely identify the sender of the message. We reserve an additional 3 bits Message field to transmit middleware specific messages, e.g. for node initialization or DHCP. Furthermore, the IGS allows to assign a priorization code to the generated message by setting the first 10 bits of the CAN message

ID accordingly. In CAN, the priority of a message derives from the first bits of the ID in the message header. Nodes begin to transmit their headers on the bus. The transmission is interrupted when a dominant bit is detected on the bus, indicating that a higher prioritized message ID is being transmitted elsewhere. Likewise, setting the first bits of a message ID allows for message priorization. However, as a considerable amount of message ID headers in the applications of the respective domains have already been pre-defined, the respective bits representing a priority level must be chosen carefully. Priorization allows to add restricted real-time support for internal transmission of IP messages. Hence, the IGS may assign higher priorities to messages required for device control than for multimedia traffic.

As the MTU of the data field may amount to 1500 bytes, each IP packet needs to be fragmented such as it fits the CAN MTU. Hence, a packetizer shapes IP traffic into 7 bytes long CAN messages. As a maximum of 215 CAN messages is required to transmit the full IP packet data, the remaining 8 bits in the CAN data field are reserved for sequencing numbers. Sequencing is required to assemble IP packets at the receiver. This is necessary to encode the full header of the IP packet. However, as a lot of IP header information is neglected and mainly determined for routing purposes, the header may be further compressed in order to reduce the information overhead and thus increase the efficiency. Fig.2 shows the new data format for IP transmission. Given this approach, the receiver may identify messages destined for him according to the message ID and rebuild full IP packets exploiting the sequencing mechanisms and the IP header information transmitted within the data payload.

5. Prototyping & Testbed

Fig.3. shows the COTS component-based evaluation system architecture. Each node is a linux based personal computer that provides an internet connection over Ethernet. In order to access the CAN hardware through the SJA 1000 CAN controller, a character device driver for linux is deployed. The IP-functionality is based on a standard network driver for linux, along with an additional software layer that is planned to manage the complete conversion of IP packets to CAN-Messages and vice versa. All functions are implemented in the C programming language.

The Infineon Microcontroller C167 is used as a interfering transmitter in order to disturb message transport, thus simulating traffic as in common CAN based systems. Experimental results demonstrate that is generally possible to apply web-based applications with average data rate bursts on CAN networks.

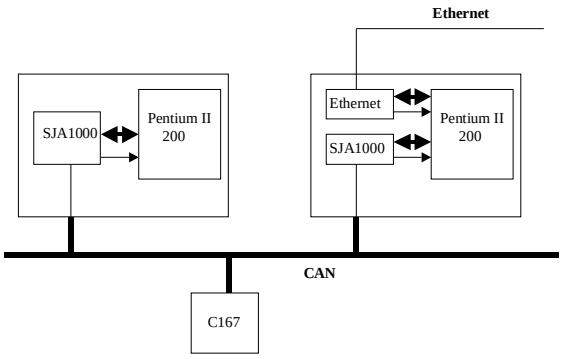

Figure 3:IP over CAN Hardware Architecture

6. Summary & Future Work

This paper proposes the Internet Protocol as a higher level communication service interface for external CAN connectivity. It addresses the challenges of porting the IP to CAN as imposed by the CAN transmission protocol 2.0B and presented the IGS as a new middleware service component that enables new innovative IP-based services to be ported to CAN nodes. Currently, we are implementing this new approach on a restricted prototype network with three clients. In future we will extend this network to host multiple clients in order to obtain real world evaluation results. Furthermore, we will integrate the IGS as a bundle in OSGI, a communication middleware that maps external to internal network communication. Moreover, we will examine the IP protocol to achieve better header compression in order to reduce the transmission overhead.

[1] S. Baatz, M. Frank, R. Gopffarth, D. Kassatkine, P. Martini, M. Schetelig, A. Vilavaara,*:* ”Handoff support for mobility with IP over Bluetooth”, *25th Annual IEEE Conference on Local Computer Networks, 2000*

[2] Beveridge,M.: ”Jini on the Control Area Network (CAN): A case Study in Portability Failure”, 2001.

[3] Cach,P., Fiedler, P.:”IP over CAN” *Internet Draft, Category: Experimental*. March, 2001.

[4] ISO 11898; "Road vehicles - Interchange of digital information - Controller area network (CAN) for high-speed communication

[5] Santamaria, ,R.: ” IP over IEEE-1394: Using the Internet Protocol over a high-performance serial bus”, Dedicated Systems Magazine, Networks, 2000

[6] Tindell,K., Burns,A., Wellings,A. : ” Calculating Controller Area Network (CAN) Message Response Times”, Control Engineering Practice, vol. 3, no. 8, pp. 1163-- 1169, 1995.