

Exploring Use of Ethernet for In-Vehicle Control Applications: AFDX, TTEthernet, EtherCAT, and AVB

Rodney Cummings
National Instruments

Kai Richter
Symtavision GmbH

Rolf Ernst and Jonas Diemer
TU Braunschweig

Arkadeb Ghosal
National Instruments

ABSTRACT

Vehicle communication networks are challenged by increasing demands for bandwidth, safety, and security. New data is coming into the vehicle from personal devices (e.g. mobile phones), infotainment systems, camera-based driver assistance, and wireless communication with other vehicles and infrastructure. Ethernet (IEEE 802.3) provides high levels of bandwidth and security, making it a potential solution to the challenges of vehicle communication networks. However, in order to be used in control applications, Ethernet must provide known timing performance (e.g. bounded latency and jitter), and in some cases redundancy.

This paper explores use of Ethernet for in-vehicle control applications. It begins by providing an overview of four Ethernet-based protocols under consideration for in-vehicle control: Avionics Full Duplex Switched Ethernet (AFDX), Time-Triggered Ethernet (TTEthernet), Ethernet for Control Automation Technology (EtherCAT), and standard Ethernet (IEEE 802.1 including Audio Video Bridging). The paper continues with discussion of key topics related to use of these Ethernet protocols for in-vehicle control. The topics include cost, physical layer, topology, startup time, redundancy, time-triggered scheduling, security, gateways, and higher layer software (e.g. AUTOSAR). Given the real-time nature of the application domain, the paper devotes a section to latency analysis (for communication based on the Ethernet protocols), which computes the worst-case transmission time for control data from source to destination(s).

CITATION: Cummings, R., Richter, K., Ernst, R., Diemer, J. et al., "Exploring Use of Ethernet for In-Vehicle Control Applications: AFDX, TTEthernet, EtherCAT, and AVB," *SAE Int. J. Passeng. Cars - Electron. Electr. Syst.* 5(1):2012, doi: 10.4271/2012-01-0196.

INTRODUCTION

In recent years there has been an explosion in the complexity of the automotive electronic architectures. In over a decade the electrical and software cost is poised to rise from 25% of the net cost to about 40-50% of the net cost of a vehicle [1]. In one decade the amount of software code written per vehicle will rise to 100 million Lines Of Code (LOC) from 1 million LOC. Current automotive electronic control system architectures now include tens of communication busses, scores of electronic control units,

hundreds of functions, and thousands of messages. Figure 1 shows that the average number of ECUs per car is doubling in a decade for all of the automotive markets across the world [2].

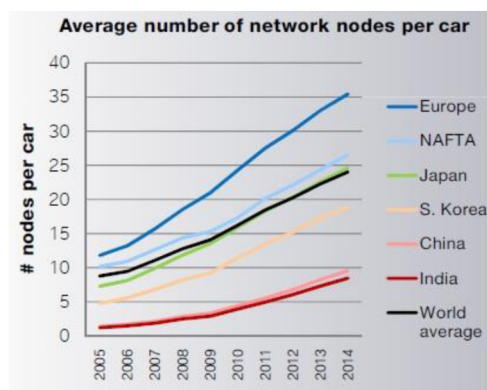


Figure 1. Average Number of ECUs per Car

The rise of electronic systems has been attributed to particularly three domains of vehicle design: control, safety and convenience. The control domain includes all aspects of vehicle driving (e.g., steering, braking, suspension etc), vehicle performance (e.g., fuel efficiency), and vehicle diagnostics. With rapid increase of market and regulation pressures, safety features (both active and passive) have become key differentiators for OEMs. Of course, standardization efforts like ISO 26262 [3] have become a platform for ensuring safety. Convenience applications like entertainment, location based services, and intelligent navigation have also been on the rise. One such trend can be seen in Figure 2.

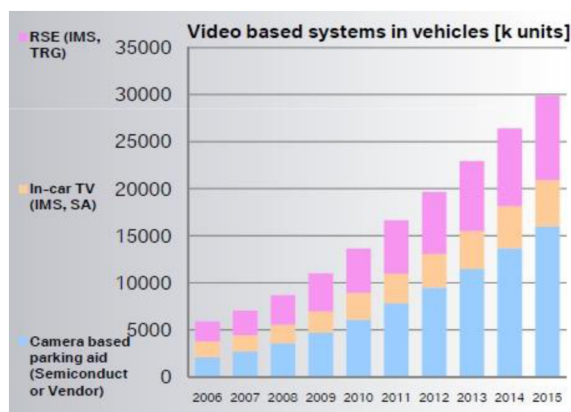


Figure 2. Video Based System in cars

One of the side effects of such an increasing complexity is exponential increase in communication - both intra-vehicle and V2X (vehicle-to-vehicle, vehicle-to-commerce, vehicle-to-infrastructure etc.). V2X communication is out-of-scope of this paper. We will focus on the intra-vehicle communication. The major protocols [4] that have been used for such communications over the last couple of decades are the following:

- Direct wire connections have been used between sensors and control units.

- Local Interconnect Network (LIN) is a low cost single-wire communication system based on enhanced ISO 9142 (standard UART interface) which supports transmission speed up to 20 Kbps (limited by EMC). The protocol supports single master/ multiple slaves, self synchronization of slave clocks, and guaranteed latency times for signal transmission.

- Controller Area Network (CAN) is a twisted-pair differential signal communication system which can support transmission speed up to 1 Mbps (however mostly used from 125 to 500 kbps). The protocol supports CSMA (Carrier Sense Multiple Access) with prioritized arbitration between messages.

- FlexRay is a twisted pair signal communication system with transmission speed up to 10 Mbps (with available versions supporting 2.5 and 5 Mbps). The protocol supports a fully deterministic approach with TDMA (Time Division Multiple Access), continuous clock synchronization with local clock, and active/passive network topologies.

- Media Oriented Systems Transport (MOST) has been used for infotainment and supports transmission speed up to 150 Mbps. The protocol can run over both optical (MOST 25, MOST 150) and electrical (MOST 50) systems. The protocol uses synchronous data for audio video content, and asynchronous packet data for information. Systems connected over MOST usually use a ring topology.

Of course an OEM may use a specific version of the above protocols; e.g., the CAN protocol used by General Motors is referred as GMLAN. With the rise of bandwidth demand, there has been push for introducing Ethernet in automotive systems. Ethernet is a switched full-duplex system that can support networking speeds from 10Mbps to 100Gbps (depending on cost). Ethernet has been extremely popular as a local area network for backend, home/office and industrial usage. There are several industrial Ethernet protocols available, e.g., Ethernet Powerlink, PROFINET, Ethernet/IP, EtherCAT, VARAN, SERCOS etc. Though there have been discussions on using Ethernet for automotive systems, the problems are best summarized by Figure 3 from TTTech [5]. Automotive control systems belong to the world of strict requirements on real-time, dependable and safe behavior. The major market for Ethernet till now has been the open consumer world which does not require strict performance guarantees. A variation of Ethernet called Ethernet AFDX has been used for avionics, but cost structure of avionics control systems (where number of backup replications can be high) is very different from automotive control systems. Though Ethernet has been used for end-of-line flashing of software for automotive product lines, use of Ethernet for intra-vehicle communication is still in research phase.

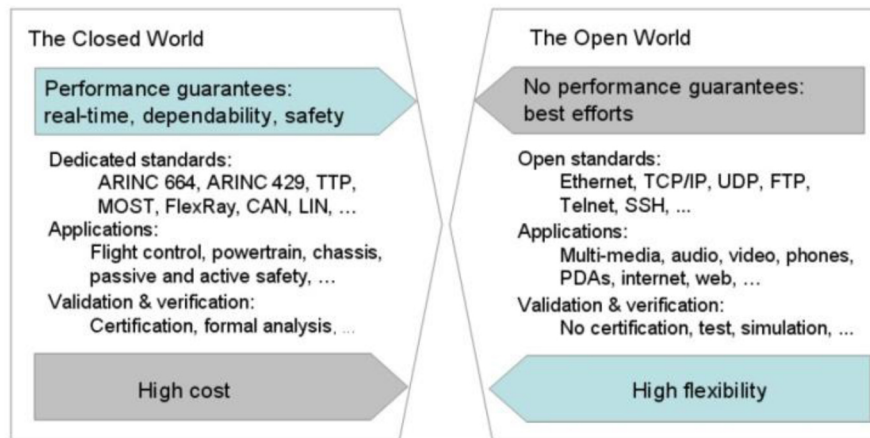


Figure 3. Closed World vs. Open World [5]

In this paper, we will discuss the latest advancements in using Ethernet for real-time control in vehicles. The focus is on four protocols: Avionics Full Duplex Switched Ethernet (AFDX), Time-Triggered Ethernet (TTEthernet), Ethernet for Control Automation Technology (EtherCAT), and standard Ethernet (IEEE 802.1 including Audio Video Bridging).

OVERVIEW OF ETHERNET-BASED PROTOCOLS

This section provides an overview of four Ethernet-based protocols that are designed for real-time embedded applications, such as in-vehicle control.

The protocols are based on the following assumptions:

- **Switched:** Early implementations of Ethernet used a linear bus topology similar to CAN, with a media access method (CSMA/CD) that was unpredictable under high bus loads. To avoid these problems, modern implementations of Ethernet are switched, in that every physical connection is between two ports (point to point). Switches route Ethernet frames received on one port to other ports of the switch.
- **Full duplex:** Although early implementations of Ethernet supported half duplex operation (one direction at a time), modern implementations of Ethernet are full duplex, supporting transmission in both directions simultaneously.
- **Best-effort and time-sensitive traffic on single network:** Best-effort refers to a class of traffic that does not impose constraints on quality-of-services metrics, such as latency, jitter, or loss. Time-sensitive refers to a class of traffic that requires a guaranteed latency. Time-sensitive traffic classes use higher priorities than best-effort. Generally speaking, most Ethernet protocols like TCP run as best-effort by default. For in-vehicle applications, time-sensitive traffic classes are used for control data and audio/video streams. Allowing both traffic classes to mix on the same network provides maximum flexibility and bandwidth usage (e.g. flashing an ECU without adverse effects on control algorithms).

AVIONICS FULL DUPLEX SWITCHED ETHERNET (AFDX)

AFDX is an Ethernet-based protocol designed for avionics (electronics in aircraft). It was developed by Airbus for use in their aircraft, and has since been used in Boeing aircraft as well. The protocol is specified in standard ARINC 664 Part 7 [6]. As shown in Figure 4, the topology of an AFDX network is analogous to an Ethernet network in the home or office environment. Each subsystem (ECU) uses a standard IEEE 802.3 PHY [7], which in turn uses two pairs of point-to-point wires (one for transmit and another for receive). The subsystem also uses a standard IEEE 802.3 MAC, along with additional hardware/software to implement AFDX-specific features. The subsystems connect to a switch, which routes data between them. The use of point-to-point wiring enables higher speeds than past technologies based on a linear bus.

Although the topology is similar to standard Ethernet, the chips within the AFDX switch add features to IEEE 802.1 (the collection of standards for Ethernet switches) [8]. AFDX switches are specially designed for the AFDX protocol in order to meet the timing requirements of avionics applications. If you were to take a standard IEEE 802.1 Ethernet switch from your local electronics store, and integrate it into an AFDX network, the AFDX protocol would not work.

The AFDX protocol is specifically designed for safety-critical control systems, such as fly-by-wire. Although it is not shown in Figure 4 for the sake of simplicity, each AFDX end-system implements full network redundancy, as shown in the simpler example of Figure 5. This means that every frame that is transmitted by an end-system is duplicated on two ports, across two distinct switched networks. On the receiving side, the end system detects both frames and removes duplicates. This results in immediate failover performance for any single point of failure in the network.

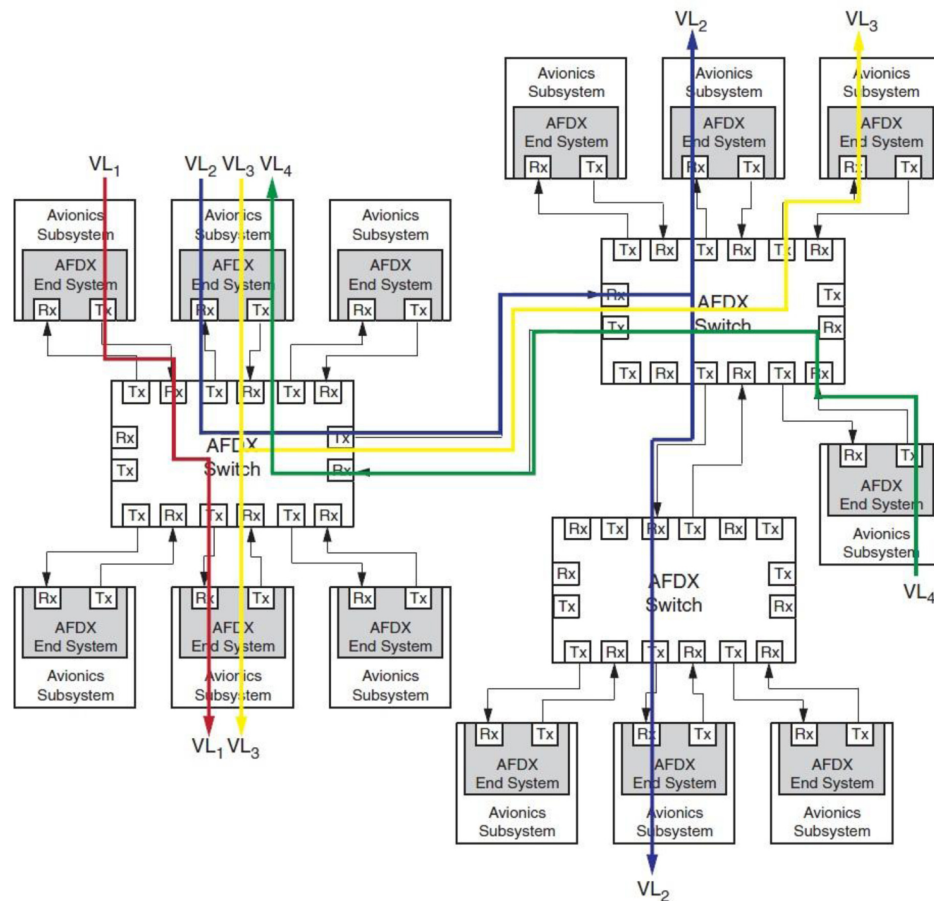


Figure 4. Example AFDX network, including Virtual Links

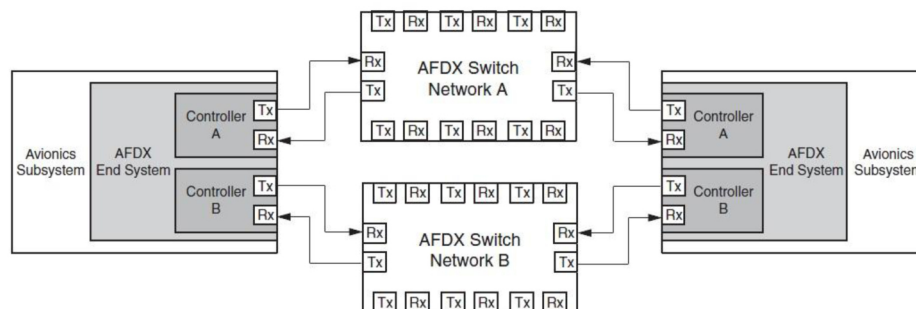


Figure 5. AFDX redundancy

Figure 6 shows the format of an AFDX frame, which is a fully compliant IEEE 802.3 frame. The Ethertype element (element after Source Address) specifies use of the Internet Protocol (IP), and within the IP packet, AFDX uses the User Datagram Protocol (UDP). UDP is used instead of TCP due to its determinism for control data. The IP and UDP structures each provide a Cyclic Redundancy Check (CRC) to enhance the safety characteristics of the frame. The first byte of AFDX Payload is a sequence number that also augments safety.

Within the Destination Address of the frame, the AFDX protocol encodes a Virtual Link Identifier. A Virtual Link (VL) represents a specific set of data that flows from one source to one or more destinations (multicast). Each VL is configured statically with a timing specification. The timing specification includes the fastest rate at which the frame can be transmitted, and the maximum size of each frame. This provides the bandwidth for each VL, which the transmitting end system must guarantee. Each AFDX switch prioritizes frame routing using the VL Identifier, and the switch also

7 Bytes	1 Byte	6 Bytes	6 Bytes	2 Bytes	20 Bytes	8 Bytes	1–1471 Bytes	0–16 Bytes	1 Byte	4 Bytes	12 Bytes
Preamble	Start Frame Delimiter	Destination Address	Source Address	0x800 IPv4	IP Structure	UDP Structure	AFDX Payload	Padding	SN	Frame Check Sequence	Inter-Frame Gap

Figure 6. AFDX frame

polices traffic to ensure that the timing specification is guaranteed (in case the transmitting end system fails to do so). Figure 4 shows an example of four VL configured in an AFDX network.

AFDX does not require time synchronization among the end systems, so the timing of data on each VL is independent (similar to CAN). Although each VL specifies its timing requirement, the guarantee of that requirement (worst-case latency) must be analyzed on a system-wide basis using the entire physical and logical configuration.

TIME-TRIGGERED ETHERNET (TTEETHERNET)

SAE-AS6802 Time-Triggered Ethernet (TTEthernet™) is an Ethernet-based protocol designed for in-vehicle control and industrial automation. TTEthernet was originally developed by the European company TTTech in conjunction with academia. With respect to avionics applications, TTEthernet switches can implement AFDX as well, to support integration into commercial aviation networks. TTEthernet was recently standardized as SAE Aerospace Standard SAE-AS6802 [9]. Figure 7 shows the typical services provided by a TTEthernet product, including AS6802, AFDX (ARINC 666 P7), and Best Effort Ethernet.

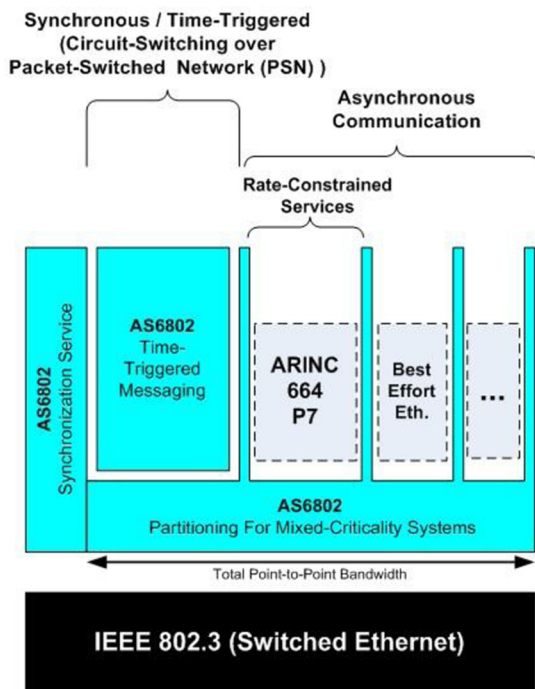


Figure 7. TTEthernet Services

TTEthernet adds a time synchronization service that provides all end systems with a common time base to sub-microsecond accuracy. The time synchronization technique is tightly integrated into redundancy protocols. Multiple time masters are used such that if any one fails, recovery is immediate with no loss of accuracy.

The AFDX Virtual Link concept is integrated into a TTEthernet product by adding an AFDX service that provides analogous Rate Constrained (RC) communication. As with AFDX, each Rate Constrained frame transmits at an independent rate, asynchronous to the global time.

TTEthernet adds the concept of Time-Triggered (TT) frames, which use the time synchronization service to transmit at precise times. Time-Triggered frames are higher priority than Rate Constrained and Best Effort. Each TTEthernet switch is configured with knowledge of the time window in which time-triggered traffic will occur. If an interfering frame (Rate Constrained or Best Effort) is about to egress out of the switch prior to the Time-Triggered window (such that it would delay TT traffic), the switch holds the interfering frame until the TT window is complete. This results in very consistent latency (low jitter) for TT frames, which in turn enables control algorithms (tasks) in multiple end systems to exchange data at well-defined points in time. This sort of task synchronization supports by-wire control and similar applications in a manner that is similar to FlexRay. In addition to the option of prioritizing TT frames, TTEthernet supports another option in which TT frames are delayed by interfering frames, to optimize bandwidth over latency.

Figure 8 shows an example network with two TT frames: one TT frame transmits with a periodicity of 2 milliseconds (ms), and another TT frame transmits with a periodicity of 3 ms [9]. Using offline configuration tools, the TTEthernet network can be configured using an overall 6 ms cycle. The TT frames are not required to be contiguous within the overall 6 ms cycle, which allows for significant flexibility in placement of the 2 ms and 3 ms TT frames. Unlike other time-triggered protocols like FlexRay that organize TT frames in a contiguous segment, TTEthernet's TT frames are organized as individual messages, mixed arbitrarily with RC and BE frames.

As with AFDX, TTEthernet uses standard IEEE 802.3 MAC and PHY, but in order to obtain the precise timing advantages, the switch and end system require hardware that is specific to TTEthernet.

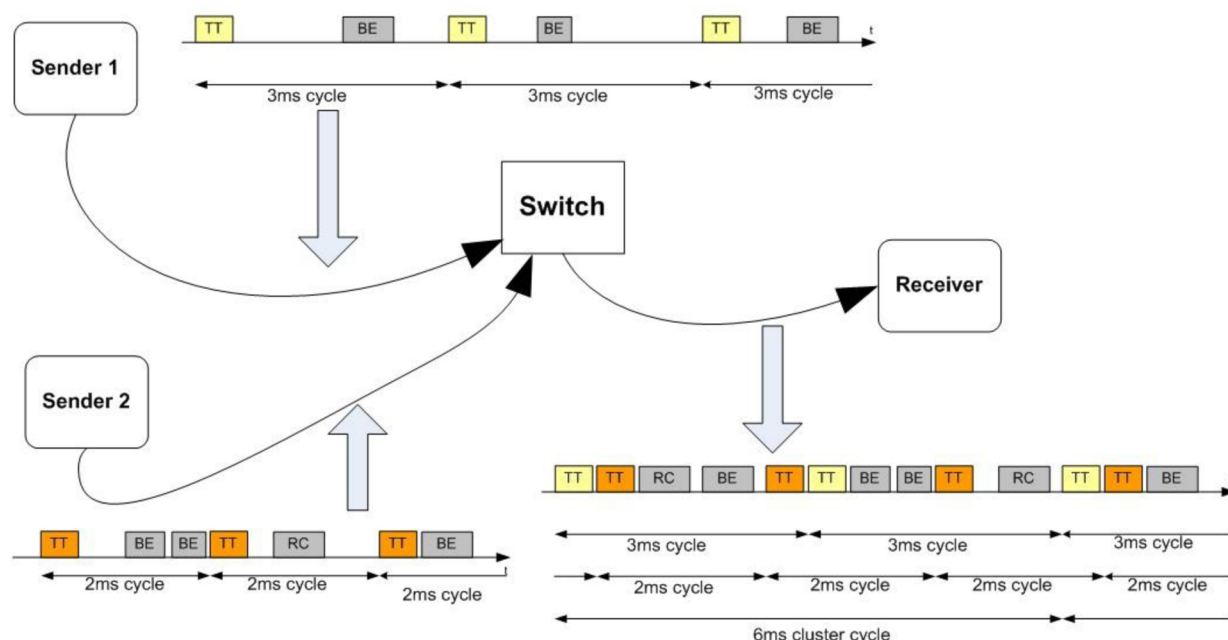


Figure 8. Example of Time-Triggered Traffic

ETHERNET FOR CONTROL AUTOMATION TECHNOLOGY (ETHERCAT)

EtherCAT® [10] is an Ethernet-based industrial automation protocol, designed for high-speed applications like multi-axis motion control. EtherCAT was originally developed by the German industrial automation company Beckhoff. EtherCAT is included in various industrial communication standards such as IEC 61158 [11].

EtherCAT is a master/slave protocol. The master is typically a controller, and slave devices provide I/O. Nevertheless, the master/slave distinction applies primarily at a hardware level, and slave-to-slave communication is straightforward. This sort of peer-to-peer communication fits in-vehicle control applications.

All EtherCAT devices use an IEEE 802.3 PHY for connection to the cable. Speed is currently limited to 100Mbps (not 1Gbps or higher like the other three protocols in this paper).

The EtherCAT master uses a standard IEEE 802.3 MAC (NIC), such that the only EtherCAT-specific element is software. Unlike the other three protocols in this paper, slave devices use an EtherCAT-specific MAC (ASIC or FPGA).

The typical topology for EtherCAT is a daisy-chain (line). Each slave device typically contains two EtherCAT ports. The master connects to port 1 of the first slave. Port 2 of the first slave connects to port 1 of the second slave, and so on. No external switch devices are required. The EtherCAT slave MAC contains specialized switching logic to pass frames

between its ports. More complex topologies are supported as well, including ring, star, and so on.

A special memory management unit in the slave MAC maps bits of frame data to/from the slave's processor memory. This allows one frame to service multiple slaves thereby decreasing the overhead and improving bandwidth utilization.

Figure 9 shows an example of EtherCAT data transfer. EtherCAT uses its own dedicated frame format (Ethertype). Multiple datagrams are contained within each EtherCAT frame. Each datagram contains a distinctly identified set of control data. The bits of the EtherCAT frame pass between the slave's ports with minimal delay (i.e. nanoseconds). The slave inserts and extracts bits of each datagram as it passes by. When the EtherCAT frame reaches the end of the line, it passes back to the master using the full-duplex IEEE 802.3 signals. This is very different than an IEEE 802.1 bridge (switch) which stores each incoming frame completely prior to forwarding it to outgoing ports.

EtherCAT data transfer can be conceptualized using a train analogy. The datagrams are passenger cars, and the bits are individual passengers, who can get on or off as the train passes by. This technique results in the lowest latency and jitter possible using Ethernet technology. Bandwidth utilization is also extremely high, since the master can send data for the next cycle immediately after receiving data from the previous cycle.

For example, up to 1486 datagram bytes can be transferred in a single frame. This maximum-sized frame takes 123.6 microseconds at the IEEE 802.3 physical layer. Given the minimum overhead added by EtherCAT hardware, this frame can travel through all slaves back to the master in