UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

DAIMLER AG,
Petitioner

v.

BLITZSAFE TEXAS,
Patent Owner

_____

U.S. Patent No. 7,489,786

"Audio Device Integration System"

_____

Inter Partes Review No. 2018-____

DECLARATION OF PHILIP KOOPMAN, PH.D IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 7,489,786 UNDER 35 U.S.C. §§ 311-319 AND 37 C.F.R. §§ 42.100 *et seq*.

TABLE OF CONTENTS

Page

APPENDICES

Ex. A	Curriculum Vitae of Dr. Philip Koopman
Ex. B	Schill, J., "An Overview of the CAN Protocol," Embedded Systems Programming, Sept. 1997
Ex. C	Leen, G., et al.,"Digital Networks in the Automotive Vehicle," Automotive Electronics, Jan. 2000
Ex. D	Upender, B. & Koopman, P., "Embedded Communication Protocol Options," Proceedings of Embedded Systems Conference 1993, Santa Clara, pp. 469-480, October 1993
Ex. E	18-549 Distributed Embedded Systems course outline
Ex. F	18-540 Distributed Embedded Systems lecture notes

I, Philip Koopman, do hereby declare and state as follows:

I. BACKGROUND AND QUALIFICATIONS

1. I am a tenured Associate Professor in the Electrical and Computer Engineering Department at Carnegie Mellon University. I have a B.S. (1982), M.Eng. (1982) and Ph.D. (1989) in Computer Engineering. I have been a professor at Carnegie Mellon since 1996. Prior to that time, I spent several years in the military and in industry working as a computer engineer and an embedded system engineer, including significant experience in the area of embedded networks. I am a named inventor on twenty-six patents, and an author or co-author of over 100 non-patent publications in a wide variety of fields within electrical engineering and computer science, including many in the technological area of embedded system networks. I have been working in computer engineering and embedded systems since approximately 1980.

2. I have extensive experience in the field of embedded communication networks, including automotive networks. For example, I have been the instructor of the course "Distributed Embedded Systems," taught to Carnegie Mellon seniors and graduate students almost every year from Fall 1999 to Fall 2015. This course includes several lectures dedicated to embedded network operation and performance, as well as lectures on more generalized embedded networking topics, including real-time scheduling, reliability, and system safety. The course features a

significant emphasis on automotive networks across a number of lectures. The course also features a semester-long distributed embedded system project in which we teach students to build a system that uses embedded network messages to coordinate operation of a distributed embedded system while guaranteeing that they can meet real-time deadlines over that network.

3. I am also the instructor of the course "Dependable Embedded Systems," which covers distributed computing and fault tolerance, including the role of embedded networks in safety-critical system design. I taught this course as part of a multi-year course rotation between Spring 1999 and Fall 2010.

4. I have supervised a number of student independent projects and thesis projects involving embedded networks. As part of this work, my lab has owned and operated increasingly sophisticated hardware Controller Area Network (CAN) testbeds from approximately 1997 to approximately 2015, and applied those testbeds to automotive applications for research projects.

5. Starting in 1999, I have been an external reviewer for at least 175 embedded system design reviews of products for industry clients, many of which have included review of the use of embedded network protocols. I have further been involved in the network protocol selection process and related system architecture selection process for several embedded system companies in which network protocols were considered. I taught seminars on embedded network protocol

selection to attendees of the Embedded Systems Conference in 1993 and 1994.

6. I served as the Guest Editor of a special edition of the magazine IEEE Micro titled "Critical Embedded Automotive Networks" in July-August 2002, which included automotive embedded network content.

7. I am a named author on numerous papers that discuss or are relevant to embedded networks, including:

- Koopman, Driscoll, Hall, "Selection of Cyclic Redundancy Code and Checksum Algorithms to Ensure Critical Data Integrity," *DOT/FAA/TC-14/49*, March 2015.
- Koopman, P. & Szilagyi, C., "Integrity in Embedded Control Networks," *IEEE Security & Privacy*, 2013.
- Szilagyi, C. & Koopman, P., "Low cost multicast authentication via validity voting in time-triggered embedded control networks," *Workshop on Embedded System Security*, October 2010.
- Koopman, P. & Ray, J., "Mitigating the Effects of Internet Timing Faults Across Embedded Network Gateways," *MMB/DFT 2010*, p. 1, March 2010.
- Szilagyi, C. & Koopman, P., "A flexible approach to embedded network authentication," *DSN* 2009, pp. 165-174.
- Ray, J. & Koopman, P., "Queue management mechanisms for embedded gateways," *DSN* 2009, pp. 175-184.
- Maxino, T., & Koopman, P. "The Effectiveness of Checksums for Embedded Control Networks," *IEEE Trans. on Dependable and Secure Computing*, Jan-Mar 2009, pp. 59-72.
- Driscoll, K., Hall, B., Koopman, P., Ray, J., DeWalt, M., *Data Network Evaluation Criteria Handbook*, AR-09/24, FAA, 2009.

- Szilagyi, C. & Koopman, P., "A flexible approach to embedded network multicast authentication," *WESS* 2008.
- Ray, J., & Koopman, P. "Efficient High Hamming Distance CRCs for Embedded Applications," *DSN06*, June 2006.
- Paulitsch, Morris, Hall, Driscoll, Koopman & Latronico, "Coverage and Use of Cyclic Redundancy Codes in Ultra-Dependable Systems," *DSN05*, June 2005.
- Koopman, P. & Chakravarty, T., "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks," *DSN04*, June 2004.
- Morris, J. & Koopman, P., "Critical Message Integrity Over A Shared Network," *FeT03*, July 2003.
- Koopman, P., "Critical Embedded Automotive Networks," *IEEE Micro*, July-August 2002.
- Koopman, P., Tran, E. & Hendrey, G. "Toward Middleware Fault Injection for Automotive Networks," *Fault Tolerant Computing Symposium*, pp. 78-79, June 23-25, 1998.
- Koopman, P., "Tracking down Lost Messages and System Failures" *Embedded Systems Programming*, 9(11), October 1996, pp. 38-52.
- Upender, B. & Koopman, P., "Communication protocols for embedded systems," *Embedded Systems Programming*, 7(11) 46-58, November 1994.
- Upender, B. & Koopman, P., "Embedded Communication Protocol Options," *Proceedings of Embedded Systems Conference* 1993, Santa Clara, pp. 469-480, October 1993; repeated in Proceedings of Embedded Systems Conference East 1994, Boston, April 1994.

8. I have industry experience in embedded network protocol use and selection, specifically including embedded networks in elevators (Otis Elevator,

circa 1991-1995), jet aircraft engines (Pratt & Whitney, circa 1992-1995), and heating/ventilation/cooling systems (Carrier, circa 1995).

9. I have been Principal Investigator or co-Principal Investigator on a number of sponsored research projects over the past two decades that directly involved the use or analysis of embedded network technology, including:

- General Motors Corporation, "Dependable Systems." Full support for 1 to 3 students (varies by year) as one of four thrust area leaders in the CMU/GM research laboratory, including work with both CAN and FlexRay. (2000-2015)
- Bosch, "Intelligent Sensors." Research on a CAN testbed. (2000-2001)
- Federal Aviation Administration (FAA), Evaluation Criteria for Databuses. (2005-2006)
- US Army TARDEC, "Safety Subsystem" task within contract for "Autonomous Platform Demonstrator (APD)." (2010)
- US Army, "Unmanned and Autonomous Systems Test (UAST) Science and Technology (S&T): A methodology for stress-testing autonomy architectures," BAA W9000KK-09-R-0038 topic #3. (2011-2016)
- FAA, "Software and digital systems program – data integrity techniques," DTFACT-11-R-00002. (2011-2013).

10. I have extensive experience in evaluating, selecting and using embedded network protocols in safety-critical systems. For example, I am a co-author of the Federal Aviation Administration's Data Network Evaluation Criteria Handbook, already cited above, which sets forth evaluation criteria for embedded networks to be used in safety-critical flight control applications. The "Distributed

Embedded Systems" course I previously mentioned teaches students how to select an embedded network for a particular embedded system application, covering network options such as CAN and FlexRay in depth, and includes a discussion of other protocols specifically including LIN and J1850.

11. I am a named inventor on the following patents that specifically address embedded control networks:

- U.S. Patent No. 5,535,212; "Implicit Token Media Access Protocol Without Collision Detection"; Koopman & Brajczewski," filed on Jan. 31, 1994, issued on Jul. 9, 1996.
- U.S. Patent No. 5,450,404; "Explicit and Implicit Token Media Access Protocol with Multi-Level Bus Arbitration"; Koopman & Brajczewski, filed on Dec. 21, 1992, issued on Sep. 12, 1995.
- U.S. Patent No. 5,436,901; "Synchronous Time Division Multiplexing Using Jam-Based Frame Synchronization"; Koopman, filed on Feb. 25, 1994, issued on Jul. 25, 1995.

12. I was the General Chair for the Dependable Systems and Networks Conference in 2008 (which is a first-ranked international academic conference on dependability, fault tolerance, and related topics including networked embedded system dependability). I was also Program Chair for the Dependable Computing and Communications Symposium (DCCS) of this same conference in 2012. I am a member of International Federation of Information Processing (IFIP) Working Group 10.4, an invitation-only organization of international researchers on the topic of Dependable Computing and Fault Tolerance that holds periodic workshops. These proceedings routinely address the topic of achieving safe and

reliable operation of distributed embedded networks and systems using such networks.

13. I am a senior member of both the Institute of Electrical and Electronic Engineers and the Association for Computing Machinery. I am a member of the Society of Automotive Engineers (SAE International).

14. Based on the above education and experience, I believe that I have a detailed understanding of the networking technology during the relevant period and, specifically, embedded networking.

15. A copy of my curriculum vitae is attached hereto as Ex. A.

II. ASSIGNMENT AND MATERIALS REVIEWED

16. I submit this declaration in support of the petition for Inter Partes Review of U.S. Patent No. 7,489,786 ("the '786 patent") submitted by Petitioner.

17. I am not an employee of Daimler or of any affiliate or subsidiary thereof.

18. I am being compensated for my time at my customary rate of $595 per hour.

19. My compensation is in no way dependent upon the substance of the opinions I offer below, or upon the outcome of Daimler's petition for *Inter Partes* review (or the outcome of the *Inter Partes* review, if trial is instituted).

20. I have been asked to provide certain opinions relating to the patentability of the ’786 patent. Specifically, I have been asked to provide opinions related to various aspects of the Bosch CAN 2.0 specification, including my opinions regarding (i) whether a publication entitled “BOSCH CAN Specification Version 2.0,” Ex. 1011, was publicly available before December 2002 to those of skill in the art and (ii) Ex. 1011 is an authentic copy of that specification.

21. For the reasons set forth below, it is my opinion that the Bosch 2.0 specification was published in approximately 1991, and was publicly available by at least 1993, but in any event was publicly available no later than October of 2001. It is my further opinion that the copy of the Bosch CAN specification Ex. 1011 is a true and correct copy of the version that would have been available before December, 2002.

III. Controller Area Network (CAN) and The 1991 Bosch CAN 2.0 Specification

22. By the late 1990s, the Controller Area Network (CAN) protocol was well established. A commonly used version of the CAN specification, was the Bosch CAN Specification Version 2.0 from 1991 (Ex.1011), which was initially targeted for use by the automotive industry.

23. As I describe in more detail below, Ex. 1011 came from my personal archives, which based on personal knowledge, I downloaded at least in October of 2001, cited in publicly available conference papers I authored before 2001, and used as suggested readings in courses I taught in the Fall of 2001.

24. I know from personal experience that, by the late 1990s, CAN chips were being made by many companies, and the use of CAN had spread beyond automotive and had become widespread use in industrial applications. This is supported, for example, by Schill, J., "An Overview of the CAN Protocol," Embedded Systems Programming, Sept. 1997. Ex. B. As another example, Leen notes that "[i]t is estimated that there are already over 140 million CAN nodes installed worldwide" by 1999, with "the majority of CAN applications exist[ing] outside of the automotive industry, employed in numerous other applications ranging from farm machinery to photocopiers." Ex. C at 6. I have personal knowledge that Schill was publicly available before the priority date of the '786

patent because, among other reasons, it was required reading for a course lecture I taught in October 2001.

25. Compelling reasons for CAN's popularity were its suitability for real-time applications and its relatively low cost, with CAN chips costing perhaps two dollars ($2) to four dollars ($4). *See, e.g.,* Ex. B at p. 2. But the cost was also lower because its adoption by the automotive industry led to broad market appeal, which in turn led to cost reduction via high-volume production. *See, e.g., id.*

26. Version 2.0 of Bosch's CAN specification (Bosch, CAN Specification Version 2.0, Robert Bosch GmbH, Stuttgart, 1991 at A-6 (the "Bosch CAN Specification, version 2.0")) is attached at Ex. 1011.

27. Ex. 1011 is the specification for the **C**ontroller **A**rea **N**etwork protocol. Version 2.0 of this specification was first published in 1991, and further, the copy attached at Exibit 1011 is a true and correct copy of version 2.0 that was first published in 1991. I have personal knowledge that the Bosch CAN Specification, version 2.0 was publicly available before the filing date of the '786 patent (which I am informed is December, 2002) because, among other reasons, I used it as a reference in papers I wrote prior do that date. Indeed, the 2.0 version of the specification was well-known to those of ordinary skill in the art related to the '786 patent, and was publicly available to anyone interested in the specification at Bosch's web site.

28. Moreover, I used and cited version 2.0 of the Bosch CAN specification attached at Ex. 1011 in Upender, B. & Koopman, P., "Embedded Communication Protocol Options," Proceedings of Embedded Systems Conference 1993, Santa Clara, pp. 469-480, October 1993 ("Upender & Koopman") (*e.g.*, it is reference 14 in Ex. D, which I co-authored) as well as in my teaching. Ex. D was publicly available via being given to attendees and additionally via publicly available purchase from the conference organizers in 1993, and was a well-known resource to anyone interested in automotive computer technology. Additionally, as author I know that I made the contents of this paper (Ex. D) publicly available via the Carnegie Mellon University Web site as of January 18, 1997.

29. Moreover, I know from personal experience that Ex. 1011 was publicly available because I cited it in another of my publications, *see, e.g.,* Koopman, P., "Control Area Network," *18-540 Distributed Embedded Systems on-line lecture notes* ("Koopman") (Ex. F at 1,16), which I know as the author, I made publicly available via Carnegie Mellon University ECE's Department web site on or before October 4, 2000. The Carnegie Mellon University's ECE department's web site was a well-known, publicly available website, and was well-known to those in the field related to the '786 patent, and would have been a well-known, and readily available resource.

30. I also am personally aware that the Bosch 2.0 CAN specification was

publicly available because I taught a course in Fall 2001 at CMU that used this specification. The course was 18-549 Distributed Embedded Systems (a re-numbered but substantially similar course to 18-540 referenced above), and I posted the updated course materials that including a link to the Bosch CAN 2.0 specification no later than October 8, 2001. It is my normal practice to create an archive (zip file) of course materials each semester, and I was able to find and refer to the archive for that semester for the below information.

31. One of the course web pages was a list of references for students to use. That web page with the list was publicly visible to the public Internet (not just Carnegie Mellon University) via the university's web site and, indeed, anyone of ordinary skill in the art related to the '786 patent could be expected to have been aware of that web site. I am personally aware that others of ordinary skill in the art were in fact aware of the course materials for that site, as I was personally aware of other professors' course materials from other Universities. In any event, the Bosch CAN specification was publicly available not only by virtue of my posting it in my course materials, but by virtue of the fact that I posted a link that was publicly available and accessible to anyone of ordinary skill in the art (indeed, I had no problem locating the specification and referencing it for my students). A copy of that web page is attached at Exibit E.

32. I assigned suggested reading for lecture #11 as the Bosch CAN

specification. At that time the specification was publicly available at:

http://www.bosch.de/de_e/productworld/k/products/prod/can/docu/can2spec.pdf

I know this because I have a web page from my Fall 2001 course that has this URL as the external URL to obtain the CAN specification, and it is my normal practice to ensure that those links work properly. Since it is suggested reading, my normal practice would have been to put that link in place on or before the date of the lecture, which was October 8, 2001.

33. Additionally, that same course web page points to a local copy with the file name: bosch91_canspec.pdf. I have an archived copy of that file with a date stamp of: October 5, 2001. That date is correct because it was (and is) my normal practice to keep an accurate date on my computer and use an operating system that I have observed to consistently record correct dates when saving files. It also checks with the lecture timeline as having been set up three days before the October 8, 2001 lecture.

34. It was (and is) also my practice to download a file from the public URL indicated and save it as a local copy for use by students in case of an internet access problem at the last minute before homework assignments are due. Therefore, I am sure that this file, which is Ex. 1011 is a true and accurate copy of the file that was available from Bosch at the above URL on or before October 5, 2001.

35. Moreover, in preparing this declaration I also located a copy of a

substantially identical CAN specification (with some addendum material) at archive.org, publicly available as of June 2001. *See https://web.archive.org/web/20010612044724/http://www.bosch.de:80/de_e/productworld/k/products/prod/can/docu/can2spec.pdf*. I have used archive.org on numerous occasions and have found that it is a reliable source of archived materials. This further corroborates that Ex. 1011 is the same specification I saved on my hard drive.

36. Based on my industry experience and on my personal knowledge, those of ordinary skill in the art at least as early as 1993 were very well-aware of the Bosch 2.0 specification through its prolific use in the industry, and would have found the specification readily available and accessible at least before December, 2001.

37. Based on all of the above, it is my opinion that Ex. 1011 was publicly available in approximately 1991, and at least before December, 2002 and that Ex. 1011 is a true and correct copy of the version of what was available at that time.

IV. Conclusion

38. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Signature: Philip Koopman

Dated: 6 June 2018

EXHIBIT A

Philip Koopman
Associate Professor, Electrical and Computer Engineering
Carnegie Mellon University

1. Biographical Data

1.A. Name

Philip Koopman

1.B. Citizenship

US Citizen by birth

1.C. Education

Degree	Discipline	University	Date
B.S. (Magna cum Laude)	Computer Engineering	Rensselaer Polytechnic Troy, NY	1982
M.Eng.	Computer Engineering	Rensselaer Polytechnic Troy, NY	1982
Ph.D.	Computer Engineering	Carnegie Mellon Pittsburgh, PA	1989

1.D. Positions

1983-1985 **Submarine Officer.** United States Navy, USS Haddock (SSN-621), U.S. Pacific Fleet.
Sonar Officer and Weapons Department Head aboard nuclear-powered fast attack submarine. Responsible for operation and maintenance of on-board sonar and target tracking computer systems; watch standing; supervising 25 men. Awarded Naval Achievement Medal, Naval Expeditionary Medal, and Sea Service Ribbon. Official combat veteran status for participation in the Cold War.

1985-1987 **Engineering Duty Officer.** United States Navy, Trident Command and Control Systems Maintenance Activity (TRICCSMA), Newport, RI.
Deputy Department Head in charge of Systems. Project management and technical consultation for submarine computer systems. Led research and development in embedded system prototyping, system certification, and configuration management.

1984-1990 **Startup Founder.** WISC Technologies Inc., La Honda, CA.
Co-Founder/Chief Engineer. Conducted computer technology research and development as the technical half of a two-person startup company. Developed, prototyped, and patented an embedded CPU design; negotiated a technology license to Harris Semiconductor. (This was concurrent with my time in the Navy, my time as a Ph.D. student, and my time at Harris Semiconductor.)

1989-1990 **Senior Scientist.** Harris Semiconductor, Melbourne, FL.
Chief architect for Real Time Express (RTX) family of embedded control microprocessors, reporting directly to VP of Processors. Technical contributions to CPU design, architectural tradeoffs, simulation, compiler optimization, and product roadmaps.

1991-1995 **Principal Research Engineer.** United Technologies Research Center, East Hartford, CT.
Team leadership, technical contribution, and research planning at corporate R&D center. Contributions in embedded communications, embedded processor applications, system design methodologies, discrete event simulation, and cryptographic security. Application areas included elevators (Otis), large-scale air conditioning (Carrier), automobiles (UT Automotive), jet engines (Pratt & Whitney, Hamilton Standard), Radars/Sonars (Norden), and helicopters (Sikorsky).

1996-1997 **Visiting Senior Research Engineer.** Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.
Team leadership and technical contributions in wearable computers, automated highway systems, and embedded system reliability. **Visiting Associate Professor** at CMU ECE department.

1997-2001 **Assistant Professor.** Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA.
Research and teaching in affordable dependability, embedded computer systems, and computer architecture. Major projects: Ballista, Amaranth, RoSES, Automated Highway Systems. Also Embedded and Reliable Information System thrust leader for the Institute for Complex Engineered Systems.

2001-… **Associate Professor.** Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA. (See Assistant Professor description above for details.)
Granted tenure effective July 1, 2002 at the rank of associate professor. Dependable Embedded System thrust leader for GM/CMU Vehicular Technology Collaborative Research Lab through 2015. Principal Investigator for Stress Test for Autonomy Architectures project and follow-ons at the National Robotics Engineering Center. Courtesy faculty member of Robotics Institute, and Institute for Software Research.

2014-… **Co-Founder.** Edge Case Research, LLC, Pittsburgh PA
Start-up company for robotic and embedded system software robustness testing tools and techniques.

1.E. Technical Consulting Engagements

1996-1999 United Technologies Automotive, Intellectual Property Department. Dearborne, MI 48126. Client: Phil LeMay. (Company now part of Lear Corp.)
Support of patent application and evaluation of intellectual property positions (*e.g.*, technical evaluation of possible patent infringement) for automotive applications. Additionally, technical support and training for their low-cost cryptographic security technology.

July 1999 AT&T Laboratories, Florham Park, NJ. Client: Yennun Huang.
Dependable distributed system architecture, testing, and hardening techniques.

1999-2003 Adtranz/Bombardier, West Mifflin PA. Client: Tom Lemak/Bob DiSilvestro
Technology assessments and design reviews for various aspects of train and peoplemover systems. Taught advanced embedded systems course to 30 ADtranz employees in Spring 2001 & Spring 2002.

1999-… Emerson Electric, St. Louis MO & Pittsburgh, PA. Client: Bill Trosky
Technology assessments and design reviews at various Emerson business units, including ASCO (power control), ASTEC (central switch emergency power; power supplies), Avocent (data center infrastructure), Branson (ultrasonic welding), Brooks (flow meters), Climate Control, Copeland (compressors; manufacturing tests), CPC (HVAC equipment), CSI (adaptive maintenance monitoring), Daniel/DeltaV (critical process flow meters, process monitoring and control), Dent Instruments (power meters; Emerson supplier), EC&P (power supplies; embedded computer boards), EMC (motion controllers), Emerson Network Power China (data center infrastructure), Fisher (flow control), HIROSS (compressors), Intermetro (medical carts), Kato (machinery monitoring),

Krautkramer (ultrasound inspection), Liebert (large UPS systems), Marconi (large power supplies), MicroMotion (Coriolis flow meters), Moobella (ice cream machines; Emerson partner), NetSure (DC power systems); Network Power (telecomm power regulation; safety-critical computers), Power & Water Solutions (power generation), Remote Automation Solutions (SCADA and controls), Ridge Tools (small tool control), Rosemount (chemical process instrumentation devices, networks & security), Tekmar (sample processing automation), Therm-O-Disc (temperature controllers; automotive components), White Rodgers (thermostats), and others. Principal author of Emerson corporate software review risk screening process.

2000 Gravitate, San Francisco, CA. Client: Geoff Hendrey
R&D for wireless telephony location-aware applications.

2001 Verizon, Bedminster, NJ.
Evaluation of wireless base station computers in context of evolution from analog to digital cell phone system network.

2001-2002 ABB Corporate Research, Baden, Switzerland. Client: Hubert Kirrmann
Embedded network protocol reviews for next-generation critical embedded systems.

2001-… Lutron, Allentown PA. Clients: J.P. Steiner, W. Zaharchuk
Evaluation of several embedded network protocols for integrated lighting systems

2002 DirecTV, El Segundo, CA.
Evaluation of piracy technology with respect to smart-card based security systems.

2003 Ingersoll Rand.
Embedded design tutorial.

2003-2007 ThyssenKrupp Elevator, San Diego, CA.
Architectural definition of elevator system product families.

2004 FlexRay Consortium.
Embedded network protocol analysis.

2006 Violin Technologies, New Jersey
Tailored CRC polynomial selection for startup company.

2008-2009 John Deere Co.
Embedded system security

2011 Residential Control Systems
Thermostat software and hardware design review

2011-2013 Google Inc.
Topic area: Embedded system design.

2014-2015 Barr Group, LLC
Design review team member and other embedded system consulting assignments. Assignments to date include: MTD (lawnmowers).

2014-2015 The National Transportation Systems Center (Volpe).

Future automotive software safety standards analysis and recommendations to NHTSA.

2014-… Chief Technologist & Startup Co-Founder, Edge Case Research LLC, Pittsburgh PA. Commercialization of robust embedded software services. Consulting for multiple clients including government, startup companies, and Fortune-500 companies.

2. Teaching and Education

2.A. Courses Taught at CMU

Semester	Number	Course Title	Number of Students	Joint Faculty	Hours/ Week	FCE Score Instructor (out of 5)	FCE Score Course (out of 5)
Fall 97	18-742	***** Advanced Computer Architecture	27	N/A	4 hours; 12 units	3.85	3.77
Spring 98	18-742	Advanced Computer Architecture	26	N/A	4 hours; 12 units	4.65	4.70
Fall 98	18-548/ 15-548	Memory System Architecture	31	N/A	4 hours; 12 units	4.68	4.64
Spring 99	18-849b	* Dependable Embedded Systems	13	N/A	4 hours; 12 units	4.75	4.92
Fall 99	18-540	* Distributed Embedded Systems	33	N/A	5 hours; 12 units	4.59	4.41
Spring 00	18-849	Dependable Embedded Systems	13	N/A	4 hours; 12 units	4.92	4.92
Fall 00	18-549	Distributed Embedded Systems (* new capstone design project)	33	N/A	5 hours; 12 units	4.72	4.72
Spring 01	18-849	Dependable Embedded Systems	12	N/A	4 hours; 12 units	4.91	4.91
Fall 01	18-549	Distributed Embedded Systems	35	N/A	5 hours; 12 units	4.48	4.29
Spring 02	18-749	* Embedded Internet (joint class with ISRI)	22	Priya Narasimhan	4 hours; 12 units	4.45	4.10
Fall 02	18-549	Distributed Embedded Systems	36	Bruce Krogh	5 hours; 12 units	4.82	4.71
Spring 03	18-749	* Dependable Embedded Systems	25	N/A	4 hours; 12 units	4.94	4.89
Fall 04	18-549	Distributed Embedded Systems	26	N/A	5 hours; 12 units	✯4.3	✯3.8
Spring 05	18-240	Introduction to Computer Engineering	68	N/A	4 hours; 12 units	2.6	2.7
Fall 05	18-849	Dependable Embedded Systems	14	N/A	4 hours; 12 units	4.9	4.9
Spring 06	18-549	Distributed Embedded Systems	23	N/A	4 hours; 12 units	4.7	4.6
Fall 06	18-348	* Embedded System Engineering	47	N/A	4 hours; 12 units	4.2	4.0
Spring 07	18-649	Distributed Embedded Systems	37	N/A	3 hours; 12 units	4.6	4.3
Fall 07	18-348	Embedded System Engineering	29	N/A	4 hours; 12 units	4.7	4.4
Spring 08	18-649	Distributed Embedded Systems	35	N/A	3 hours; 12 units	4.4	4.2
Fall 08	18-849	Dependable Embedded Systems	8	N/A	4 hours; 12 units	5.0	4.7

Spring 09	18-649	Distributed Embedded Systems	40	N/A	4 hours; 12 units	4.1	3.7
Fall 09	18-348	Embedded System Engineering	78	N/A	4 hours; 12 units	4.7	4.4
Spring 10	18-649	Distributed Embedded Systems	47	N/A	4 hours; 12 units	4.7	4.6
Fall 10	18-849	Dependable Embedded Systems	5	N/A	4 hours; 12 units	5.0	5.0
Spring 11	18-649	Distributed Embedded Systems	35	N/A	4 hours; 12 units	4.5	4.4
Fall 11	18-649	Distributed Embedded Systems	42	N/A	4 hours; 12 units	4.6	4.6
Spring 12	18-348	Embedded System Engineering	54	N/A	4 hours; 12 units	4.6	4.5
Fall 12	18-649	Distributed Embedded Systems	68	N/A	4 hours; 12 units	4.5	4.6
Spring 13	18-348	Embedded System Engineering	54	N/A	4 hours; 12 units	4.7	4.8
Fall 13	18-649	Distributed Embedded Systems	88	N/A	4 hours; 12 units	4.4	4.3
Spring 14	18-348	Embedded System Engineering	66	N/A	4 hours; 12 units	4.6	4.7
Fall 14	18-649	Distributed Embedded Systems	49	N/A	4 hours; 12 units	4.6	4.6
Spring 15	18-348	Embedded System Engineering	71	N/A	4 hours; 12 units	4.7	4.7
Fall 15	18-649	Distributed Embedded Systems	43	N/A	4 hours; 12 units	4.7	4.7
Spring 16	18-348	Embedded System Engineering	68	N/A	4 hours; 12 units	4.8	4.8
Fall 17	18-642	* Embedded System Software Engineering	24	N/A	4 hours; 12 units	4.6	4.0
Spring 18	18-642	Embedded System Software Engineering	37	N/A	4 hours; 12 units	4.7	4.2
Fall 18	18-642	Embedded System Software Engineering	TBD	N/A	3 hours; 12 units	…	…

* = new or completely re-designed course
Note: Sabbatical Fall 03 – Spring 04; Fall 2016 – Spring 2017; no courses taught.

2.B. Student Projects

2.B.1. Undergraduate & professional MS projects

1. Nelson, Jacob, “Fixing data integrity issues with aviation network standard ARINC-825,” Spring 2016.

2. Parker, Malik, “Exploration in Control Systems,” Fall 2015-Spring 2016.

3. Waters, Kevin & Matt Burnett, “Running Mate Racer,” Spring 2013.

4. Toth, Andrew, “Solar Splash Sensor Suite,” Spring 2013.

5. Flores, Brian, “Error Detection Coding Research,” Fall 2012.

6. Roop, Lincoln, “High reliability power system controller for use in a lunar rover,” Spring 2012.

7. Kelner, Ilya; Nangia, Siddarth, “SURG: SWANDRIVE: Smart Wireless Analysis Network for Driver Information and Vehicle,” Fall 2010-Spring 2011

8. Stroz, Glenn, “Undergraduate acoustic I/O course project development,” Fall 2010.

9. Kelner, Ilya; Nangia, Siddarth; Debner, Joshua, “SURG: Joules, a smart electrical socket,” Fall 2009-Spring 2010.

10. Zaman, Jason; Potter, Jacob, “Computer Controlled Multicolor Laser Projector,” Fall 2009.

11. Amber Imam, “Watchdog timer effectiveness,” Spring 2008.

12. Gregory Collins, “Fluorescent currency reader embedded system,” Spring 2008.

13. K.C. Choi, “Mobot without a camera,” Fall 2007.

14. David Guttendorf, “Remote Software Robustness Testing,” Summer 1998-Fall 1998; Summer 1999-Spring 2000; Fall 2000. “Navigation Testbed” Spring 2001.

15. John Esper, “CAN data recording gateway,” Fall 2000.

16. Aditi Bajoria, “CAN testbed,” Summer 2000; Error detection code effectiveness, Fall 2000.

17. Pratish Halady, Chris Martin & Ernie Pusateri, “Bosch Network Interface,” Fall 1999-Spring 2000.

18. Tridib Chakravarty, “Fire-Wire Protocol Simulation & Train Network Protocols,” Fall 1998-Spring 2000.

19. Chih-Chia, “Scott” Wen, “Software Robustness Testing Web Site,” Spring 1999.

20. Asad Zaidi, “Web Interface to Operating System Robustness Testing,” Summer 1997.

21. Ms. Kanda Runapongsa, “Testing Harness for Operating System Robustness Testing,” Summer 1997. Entered M.S. program at U. Michigan, Fall 1997.

22. John Sung, “Comparative Operating System Robustness Benchmarking,” Spring 1997. (Daniel Siewiorek was secondary advisor). CIT honors project; led to co-authorship on conference paper.

2.B.2. Master Students

1. Malcolm Taylor. Admitted for M.S.+Ph.D. 8/09; MS 5/14.

2. Justin Ray. Admitted for M.S.+Ph.D. 9/04; MS 5/06. (Continued for Ph.D. at Carnegie Mellon.)

3. Ms. Theresa Maxino. Admitted for 9/04; MS 5/06.

4. Ms. Jennifer Ann Morris, “A Fault Tolerance Analysis of Safety-Critical Embedded Systems,” MS 5/04. NSF Fellowship. (Continued for Ph.D. at Carnegie Mellon.)

5. Ms. Yang Wang, admitted August 2001. Transferred to Dr. Chenxi Wang as advisor Dec 2002 to change research topic.

6. Ms. Beth Latronico, “Representing Embedded System Sequence Diagrams As A Formal Language," 5/02. NDSEG Fellowship; Intel MS Fellowship. (Continued for Ph.D. at Carnegie Mellon.)

7. Tridib Chakravarty, “Performance of cyclic redundancy codes for embedded networks,” 12/01. Position: Panasas Inc., Pittsburgh, PA.

8. Christopher Martin, “Functional fault simulation for distributed embedded systems,” 12/01. Position: Bosch Research Center, Pittsburgh, PA.

9. Ms. Meredith Beveridge, “Jini on CAN,” 3/01. Intel IMAP Scholar; NSF Fellowship. Position: SWRI, Houston, TX.

10. Pan, Jiantao, “Robustness Testing and Hardening of CORBA ORB Implementations,” 12/00. (Continued in Ph.D. program at Carnegie Mellon.)

11. Sandeep Tamboli, “Evaluation of Admission Policies for Probabilistic Quality of Service (QoS),” 08/00. (co-advisor with Siewiorek). Position: Marconi, Pittsburgh, PA.

12. Charles Shelton, “Embedded System Robustness Testing,” 03/00. Lucent Minority Research Fellowship; Carnegie Scholar. (Continued for Ph.D. at Carnegie Mellon.)

13. Arjun Cholkar, “Simulation Testbed for Distributed System Quality of Service Experiments,” 5/99. Position: GTE, Dallas TX.

14. Ms. Kimberly Fernsler, “Robustness Evaluation and Improvement for the HDL Simulation Backplane,” 5/99. Carnegie Scholar. Position: IBM, Austin, TX.

15. Geoffrey Hendrey, “Embedded Communication Protocol Performance Evaluation,” 5/99. Position: CTO of Gravitate Inc. (start-up in location-aware mobile/wireless information services).

16. Ms. Eushiuan Tran Tsung, “Improving the Reliability of Safety-Critical Automotive Embedded Systems,” 5/99. Carnegie Scholar. Position: Lincoln Laboratories, Boston MA.

17. John DeVale, “Automated Software Robustness Testing,” 12/98. (Continued for Ph.D. at Carnegie Mellon.)

18. Nathan Kropp, “Automatic Robustness Testing of Off-The-Shelf Software Components,” 5/98. (co-advised with Daniel Siewiorek). Position: Bayer Corp, Ohio.

2.B.3. Ph.D. Students

1. John Filleau, Entered direct Ph.D. program Fall 2013. NSF Fellowship. Passed qualifiers 12/15.

2. Milda Zizyte, Entered direct Ph.D. program Fall 2012. NSF Fellowship. Passed qualifiers 5/15. Proposal 2/17.

3. Casidhe (Felix) Hutchison, Entered direct Ph.D. program Fall 2012. Passed qualifiers 5/14. Graduated with MS 8/2016. Position: National Robotics Engineering Center.

4. Malcolm Taylor, research area: Embedded system task isolation. Entered direct Ph.D. program 8/09. NSF Fellowship; passed qualifiers 5/12. Graduated with MS, May 2014. Position: Johns Hopkins/APL.

5. Aaron Kane, research area: Embedded safety monitoring. Entered direct Ph.D. program 8/09; passed qualifiers 12/10. Proposal 12/11. General motors Foundation Fellowship 2012. Graduation 2/2015. Position: Edge Case Research LLC.

6. Justin Ray, research area: Embedded Gateway Survivability. Entered Ph.D. program 9/06 with M.S.; passed qualifiers 12/06; proposed 2/11; graduation 12/13. Position: National Robotics Engineering Center.

7. Christopher Szilagyi, research area: Embedded Security. Entered direct Ph.D. program 8/06; passed qualifiers 5/09; proposed 5/10; graduation 5/12. Position: Northrup Grumman.

8. Ms. Jennifer Ann Morris Black, research area: Safety Invariant Expression, Monitoring, and Recovery. NSF Fellowship, General Motors research fellowship. Entered Ph.D. program 9/04 with M.S.; passed qualifiers 4/05; proposal 5/07; defense 4/09; graduated. Position: full-time Mom.

9. Ms. Elizabeth Latronico, research area: Design Reliability Validation of Group Membership Services for X-by-Wire Protocols, started 09/00. Carnegie Scholar; Intel IMAP Fellowship; DoD NDSEG Fellowship; American Association of University Women fellowship; Amelia Earhart Fellowship from Zonta International; selected for NSF Fellowship. Admitted direct Ph.D. program 8/00 with B.S.; passed qualifiers 12/01; proposal 5/03; graduated 5/05. Position: Bosch Research Center, Pittsburgh PA.

10. Charles Shelton, research area: System Architecture for Graceful Degradation. Lucent Minority Research Fellowship; Carnegie Scholar. Entered Ph.D. program with M.S. 3/00; passed qualifiers 5/00; proposal 5/02; defense 6/03. Position: Bosch Research Center, Pittsburgh PA.

11. Bill Nace, research area: Robust system degradation. Entered Ph.D. program 6/99; passed qualifiers 5/00; proposal 12/01; defense 5/02. Air Force Officer fellowship (active duty US Air Force officer while student). Position: USAF Academy Faculty; now Carnegie Mellon University.

12. John DeVale, research area: performance for exception handling. IBM Research Fellowship. Entered Ph.D. program 1/99; passed qualifiers 5/99; proposal 3/01; defense 10/01. Position: Intel architecture lab, Austin TX; moved to JHU/APL in 2007.

13. Jiantao Pan, research area: Off-the-shelf component robustness hardening. Entered Ph.D. program 8/97; passed qualifiers 8/98, proposal 12/00; graduated. (Co-advisor: Daniel Siewiorek.)

14. Ms. Grace McNally, "Automated Architecture Synthesis for Embedded Multicomputer Systems," 8/98, (Daniel Siewiorek was primary advisor). Position: Research engineer, Stratus Computer.

15. Ms. Ann Marie Grizzaffi-Maynard, "Error Detection in High Performance Pipelined Processors," 10/89, (jointly advised with Daniel Siewiorek). Position: Director, IBM Center for Advanced Studies, IBM research, Austin, Texas.

3. Publications

3.A. Books

1. Koopman, P., *Better Embedded System Software*, Drumnadrochit Press, 2010, 398 pages, ISBN-13: 978-0-9844490-0-2.
   This book collects the wisdom of approximately 90 design reviews performed on industry embedded systems over more than a decade to present 28 areas in which embedded system designers can improve. This book is being used as a senior/MS text at Carnegie Mellon, NCSU, and University of Utah.

2. Driscoll, K., Hall, B., Koopman, P., Ray, J., DeWalt, M., *Data Network Evaluation Criteria Handbook*, AR-09/24, FAA, 2009.

3. Siewiorek, D., & Koopman, P., *The Architecture of Supercomputers: Titan, a case study*, Academic Press, 1991. 202 pages. ISBN 0-12-643060-8. Foreword by Gordon Bell.
   This a detailed case study (and, to our knowledge, the only such book) of how to design a high-performance vector computer. While such supercomputers are not currently being designed, the very same ideas are being used in current research on vector microprocessors.

4. Koopman (ed.), *SIGForth '90 and SIGForth '91 Conference Proceedings*, ACM Press, 1991. 134 pages. ISBN 0-89791-462-7.

5. Koopman, P., *An Architecture for Combinator Graph Reduction*, Academic Press, 1990. 155 pages. ISBN 0-12-419240-8.
   [This is the book version of my Ph.D. Thesis.]

6. Koopman, P., *Stack Computers*, Ellis Horwood/Halstead Press, 1989. 234 pages. ISBN 0-7458-0418-7.
   After two decades, this remains the definitive work on embedded stack computers and still sells a reprint hard copy every few weeks. Although not well publicized and in an area that is not fashionable of late, these machines have found significant application in embedded control worldwide and NASA spacecraft applications. An electronic copy was included in the October 2001 issue of the French technology magazine *Login*, circulation 13,000.

7. Koopman, P., *Forth Floating Point*, Mountain View Press, 1985. 346 pages. ISBN 0-914699-28-8.

3.B. Book Chapters

1. Wagner & Koopman, "A Philosophy for Developing Trust in Self-Driving Cars," In: G. Meyer & S. Beiker (eds.) Road Vehicle Automation 2, Lecture Notes in Mobility, Springer, 2014, pp. 163-170

2. Koopman, P. & Hoffman, R., "Work-arounds, make-work, and kludges," Collected Essays on Human-Centered Computing, IEEE, 2012

3. Koopman, P., DeVale, K. & DeVale, J., "Interface Robustness Testing: Lessons Learned from the Ballista Project," *Dependability Benchmarking*, IEEE press, 2008.

4. Shelton, C. & Koopman, P. “Using architectural properties to model and measure graceful degradation,” in Romanovsky *et al.* (ed.), *Architecting Dependable Systems,* LNCS 2677, Springer, 2003.

5. Koopman, P., “Toward a Scalable Method for Quantifying Aspects of Fault Tolerance, Software Assurance, and Computer Security,” In: Amman (ed.) *From Needs To Solutions*, IEEE Press, 1999, pp. 103-131.

6. Koopman, P., Lee, P. & Siewiorek, D., “Architectural Considerations for Combinator Graph Reduction,” Lee, P. (ed.) *Topics In Advanced Language Implementation,* MI4+T Press, 1991, pp. 369-95.

3.C. Archival Journal Papers Critically Reviewed Before Publication

1. Koopman, P. and Wagner, M., "Challenges in Autonomous Vehicle Testing and Validation," SAE Int. J. Trans. Safety 4(1):2016, doi:10.4271/2016-01-0128
   Review Process: anonymous peer review of complete manuscript prior to publication.

2. Koopman, P. & Wagner, M., “Autonomous Vehicle Safety: An Interdisciplinary Challenge,” *IEEE Intelligent Transportation Systems Magazine*, Vol. 9 No. 1, Spring 2017, pp. 90-96.
   Review Process: anonymous peer review of complete manuscript prior to publication.

3. Abdallah, A., Feron, E., Hellestrand, G., Koopman, P. & Wolf, M., "Hardware/Software Co-Design of Aerospace and Automotive Systems," *Proc. IEEE*, April 2010, pp. 584-602.
   Review Process: anonymous peer review of complete manuscript prior to publication.

4. Maxino, T., & Koopman, P. "The Effectiveness of Checksums for Embedded Control Networks," *IEEE Trans. on Dependable and Secure Computing*, Jan-Mar 2009, PP. 59-72.
   Review Process: anonymous peer review of complete manuscript prior to publication. (Journal has approximately 10% acceptance rate.)

5. Philip Koopman, Howie Choset, Rajeev Gandhi, Bruce Krogh, Diana Marculescu, Priya Narasimhan, JoAnn M. Paul, Ragunathan Rajkumar, Daniel Siewiorek, Asim Smailagic, Peter Steenkiste, Donald E. Thomas, Chenxi Wang, “Undergraduate Embedded System Education at Carnegie Mellon,” *ACM Transactions on Embedded Computing Systems*, Vol 4., No. 3, Fall 2005.
   Review Process: anonymous peer review of complete manuscript prior to publication. I was lead author of this paper and responsible for significant content & overall editing.

6. Raz, O., R. Buchheit, M. Shaw, P. Koopman, & C. Faloutsos, “Detecting Semantic Anomalies in Truck Weigh-In-Motion Traffic Data Using Data Mining,” *Journal of Computing in Civil Engineering*, vol 18, pg. 291 (2004)
   Review Process: anonymous peer review of complete manuscript prior to publication.

7. Hoover, C., Hansen, J., Koopman, P. & Tamboli, S., “The Amaranth Framework: policy-based quality of service management for high-assurance computing,” *International Journal of Reliability, Quality, and Safety Engineering*, Vol. 8, No. 4, 2001, pp. 1-28.
   Review Process: anonymous peer review of complete manuscript prior to publication.

8. Koopman, P. & DeVale, J., “The Exception Handling Effectiveness of POSIX Operating Systems,” *IEEE Transactions on Software Engineering*, Vol. 26, No. 9, September 2000, pp. 837-848.

Review Process: anonymous peer review of complete manuscript prior to publication.
(This is a more comprehensive, journal version of the 1998 FTCS paper.)

9. Koopman, P. & Bayouth, M., "Orthogonal Capability Building Blocks for Flexible AHS Deployment," *Journal of Intelligent Transportation Systems*, Vol. 4, pp. 1-19, 1998.
Review Process: anonymous peer review of complete manuscript prior to publication.

10. Bayouth, M. & Koopman, P., "Functional Evolution of an Automated Highway System for Incremental Deployment," *Transportation Research Record*, #1651, Paper #981060, pp. 80-88.
Review Process: anonymous peer review of complete manuscript prior to publication.

11. Koopman, P., "Obstacles to Using CAD Tools for Embedded System Design: a case study," *Integrated Computer Aided Engineering*, 5(1) 85-94, 1998.
Review Process: anonymous peer review of complete manuscript prior to publication.

12. Koopman, P., "A Preliminary Exploration of Optimized Stack Code Generation," *Journal of Forth Applications and Research*, vol. 6, no. 3, pp. 241-251, 1994.
Review Process: anonymous peer review of complete manuscript prior to publication.

13. Koopman, P., Lee, P. & Siewiorek, D., "Cache Behavior of Combinator Graph Reduction," *Transactions on Programming Languages and Systems*, vol. 14, no. 2, pp. 265-297, April 1992.
Review Process: anonymous peer review of complete manuscript prior to publication. ("Flagship" journal for programming language community.)

14. Koopman, P. & Siewiorek, D., "ICs for workstations," *IEEE Spectrum*, vol. 29, no. 4, pp. 52-54. April, 1992.
Review Process: anonymous peer review of complete manuscript prior to publication.

3.D. Archival Papers in First-Tier Symposium and Conference Proceedings, Critically Reviewed Before Publication

1. Hutchison et al., "Robustness Testing of Autonomy Software," ICSE-SEIP, 2018.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international software engineering conference.

2. Kane, Koopman, "Monitor Based Oracles for Cyber-Physical System Testing," *DSN 2014: The International Conference on Dependable Systems and Networks*, June 2014.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference.

3. Black, J. & Koopman, P., "System safety as an emergent property in composite systems," *DSN 2009: The International Conference on Dependable Systems and Networks*, June 29-July 2, 2009.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 21% acceptance rate for archival papers.

4. Ray, J. & Koopman, P., "Queue management mechanisms for embedded gateways," *DSN 2009: The International Conference on Dependable Systems and Networks*, June 29-July 2, 2009.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 21% acceptance rate for archival papers.

5. Szilagyi, C. & Koopman, P., “Flexible Multicast Authentication for Time-Triggered Embedded Control Network Applications*DSN 2009: The International Conference on Dependable Systems and Networks*, June 29-July 2, 2009.
   Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 21% acceptance rate for archival papers.

6. Ray, J., & Koopman, P., “CRC Checksums for Embedded Systems,” *DSN 2006: The International Conference on Dependable Systems and Networks*, June 28-July 1, 2006.
   Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 18% acceptance rate for archival papers.

7. Latronico, E. & Koopman, P., "Design Time Reliability Analysis of Distributed Fault Tolerance Algorithms," *DSN 2005: The International Conference on Dependable Systems and Networks*, June 28 - July 1, 2005.
   Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 25% acceptance rate for archival papers.

8. Paulitsch, Morris, Hall, Driscoll, Koopman & Latronico, "Use and Misuse of Cyclic Redundancy Codes in Ultra-Dependable Systems," *DSN 2005: The International Conference on Dependable Systems and Networks*, June 28 - July 1, 2005.
   Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 25% acceptance rate for archival papers.

9. Shelton, C. & Koopman, P., "Improving System Dependability with Functional Alternatives," *DSN 2004: The International Conference on Dependable Systems and Networks*, June 28 - July 1, 2004.
   Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 21% acceptance rate for archival papers.

10. Latronico, E.; Miner, P.; & Koopman, P., "Quantifying the Reliability of Proven SPIDER Group Membership Service Guarantees," *DSN 2004: The International Conference on Dependable Systems and Networks*, June 28 - July 1, 2004.
    Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 21% acceptance rate for archival papers.

11. Morris, J.; Kroening, D; & Koopman, P., "Fault Tolerance Tradeoffs in Moving from Decentralized to Centralized Embedded Systems," *DSN 2004: The International Conference on Dependable Systems and Networks*, June 28 - July 1, 2004.
    Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 21% acceptance rate for archival papers.

12. Koopman, P. & Chakravarty, T., "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks," *DSN 2004: The International Conference on Dependable Systems and Networks*, June 28 - July 1, 2004.
    Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 21% acceptance rate for archival papers.

13. DeVale, J. & Koopman, P., “Robust software – no more excuses,” *Fault Tolerant Computing Symposium (FTCS-31)/Dependable Systems and Networks*, Washington DC, July 2002, pp. 145-154.
    Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 26% acceptance rate for archival papers.
    Awarded the William C. Carter *outstanding paper award*.

14. Koopman, P., "32-bit cyclic redundancy codes for Internet applications," *Fault Tolerant Computing Symposium (FTCS-31)/Dependable Systems and Networks*, Washington DC, July 2002, pp. 459-468.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference; 26% acceptance rate for archival papers.

15. Raz, O., Shaw, M. & Koopman, P., "Semantic anomaly detection in on-line data sources," *International Conference on Software Engineering (ICSE)*, Orlando FL, May 2002.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international software engineering conference; 13% acceptance rate.

16. Pan, J., Koopman, P., Siewiorek, D., Huang, Y., Gruber, R. & Jiang, M., "Robustness testing and hardening of CORBA ORB Implementations," *Fault Tolerant Computing Symposium (FTCS-31)/Dependable Systems and Networks*, July 2001, Gotenberg Sweden, pp. 141-150.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference.

17. Shelton, C. & Koopman, P., "Robustness Testing of the Microsoft Win32 API," *Fault Tolerant Computing Symposium (FTCS-30)/Dependable Systems and Networks*, pp. 261-270, June 2000, New York City.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference.

18. Fernsler, K. & Koopman, P., "Robustness Testing of a Distributed Simulation Backplane," *International Symposium on Software Reliability Engineering (ISSRE)*, pp. 189-198, 1999.
Review Process: anonymous peer review of complete manuscript prior to publication.

19. Devale, J. & Koopman, P., "Comparing the Robustness of POSIX Operating Systems," *Fault Tolerant Computing Symposium (FTCS-29)*, pp. 30-37, June 1999.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference.

20. Kropp, N., Koopman, P. & Siewiorek, D., "Automated Robustness Testing of Off-the-Shelf Software Components," *Fault Tolerant Computing Symposium*, pp. 230-239, June 23-25, 1998.
Review Process: anonymous peer review of complete manuscript prior to publication. This is the top international fault-tolerant computing conference.

21. Koopman, P., Lee, P. & Siewiorek, D., "Cache Performance of Combinator Graph Reduction," *1990 International Conference on Computer Languages*, pp. 39-48, March 12-15, 1990.
Review Process: anonymous peer review of complete manuscript prior to publication. This is a first-tier conference.

22. Koopman, P. & Lee, P., "A Fresh Look at Combinator Graph Reduction," *Proceedings of the 1989 SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pp. 110-119, June 21-23, 1989.
Review Process: program committee review of extended abstract prior to publication; however, this is considered a highly selective, first-tier conference publication.

3.E. Conference papers critically reviewed in their entirety before publication

1. Koopman, P. “Practical Experience Report: Automotive Safety Practices vs. Accepted Principles,” SAFECOMP, Sept. 2018. 8 pages.
   Review Process: anonymous peer review of complete manuscript prior to publication.

2. Koopman, P. & Wagner, M., "Toward a Framework for Highly Automated Vehicle Safety Validation," SAE World Congress, 2018. SAE-2018-01-1071.
   Review Process: anonymous peer review of complete manuscript prior to publication.

3. Vernaza, Guttendorf, Wagner & Koopman, Learning Product Set Models of Fault Triggers in High-Dimensional Software Interfaces, IROS 2015.
   Review Process: anonymous peer review of complete manuscript prior to publication.

4. Koopman, P. & Wagner, M., "Toward a Framework for Highly Automated Vehicle Safety Validation," SAE World Congress, 2018. SAE-2018-01-1071.
   Review Process: anonymous peer review of complete manuscript prior to publication.

5. Kane, Chowdhury, Datta & Koopman, "A Case Study on Runtime Monitoring of an Autonomous Research Vehicle (ARV) System," RV 2015.
   Review Process: anonymous peer review of complete manuscript prior to publication.

6. Koopman, P. & Szilagyi, C., "Integrity in Embedded Control Networks," IEEE Security & Privacy, 2013. May/June 2013, pp. 61-63.
   Review Process: review by two column co-editors of complete manuscript prior to publication.

7. Kane, A. & Koopman, P., "Ride-through for Autonomous Vehicles," 2nd International Workshop on Critical Automotive applications: Robustness and Safety (CARS), Sept 24, 2013.
   Review Process: anonymous peer review of complete manuscript prior to publication.

8. Koopman, P., "Lessons Learned in Teaching a Complex Distributed Embedded System Project Course," CPS-Ed 2013, April 8, 2013.
   Review Process: anonymous peer review of complete manuscript prior to publication.

9. Koopman, P., “Risk Areas In Embedded Software Industry Projects,” Workshop on Embedded System Education (WESE), October 2010.
   Review Process: anonymous peer review of complete manuscript prior to publication. Promoted to keynote talk after acceptance.

10. Szilagyi, C. & Koopman, P., "Low cost multicast authentication via validity voting in time-triggered embedded control networks," Workshop on Embedded System Security (WESS), October 2010.
    Review Process: anonymous peer review of complete manuscript prior to publication.

11. Black, J., & Koopman, P., "Indirect control path analysis and goal coverage strategies for elaborating system safety goals in composite systems," Pacific Rim Dependability Conference (PRDC), Dec. 15-17, 2008.
    Review Process: anonymous peer review of complete manuscript prior to publication.

12. Szilagyi, C. & Koopman, P., "A flexible approach to embedded network multicast authentication," Workshop on Embedded System Security (WESS), Oct. 23, 2008.
    Review Process: anonymous peer review of complete manuscript prior to publication.

13. Morris, J. & Koopman, P., “Representing Design Tradeoffs in Safety-Critical Systems,” ICSE 2005 Workshop on Architecting Dependable Systems, May 17, 2005.
Review Process: anonymous peer review of complete manuscript prior to publication. 54% acceptance rate.

14. Orna Raz, Rebecca Buchheit, Mary Shaw, Philip Koopman, and Christos Faloutsos, “Automated Assistance for Eliciting User Expectations," *International Conference of Software Engineering and Knowledge Engineering*, June 2004, 6 pages.
Review Process: anonymous peer review of complete manuscript prior to publication. 38% acceptance rate.

15. Martin, C. & Koopman, P., "Representing User Workarounds As A Component Of System Dependability," *PRDC 2004: 10th IEEE Pacific Rim International Symposium on Dependable Computing*, March 3-5, 2004, 10 pages.
Review Process: anonymous peer review of complete manuscript prior to publication.

16. Latronico, E. & Koopman, P., "A Period-Based Group Membership Strategy for Nodes of TDMA Networks," *FeT 2003 (5th IFAC International Conference on Fieldbus Systems and their Applications)*, July 2003, 7 pages.
Review Process: anonymous peer review of complete manuscript prior to publication.

17. Morris, J. & Koopman, P., "Critical Message Integrity Over A Shared Network," *FeT 2003 (5th IFAC International Conference on Fieldbus Systems and their Applications)*, July 2003, 6 pages.
Review Process: anonymous peer review of complete manuscript prior to publication.

18. Koopman, P., “Elements of the self-healing system problem space” *Workshop on Architecting Dependable Systems* (affiliated with ICSE 2003), May 2003, 6 pages.
Review Process: anonymous peer review of complete manuscript prior to publication.

19. Raz, O., Koopman, P., & Shaw, M., “Enabling automatic adaptation in systems with under-specified elements,” *First Workshop on Self-healing Systems (WOSS02)*, 18-19 November 2002, Charleston SC. 6 pages.
Review Process: anonymous peer review of complete manuscript prior to publication.

20. Koopman, P., “What’s wrong with fault injection as a dependability benchmark?,” *Workshop on Dependability Benchmarking* (in conjunction with DSN 2002), Washington DC, June 2002, 6 pages.
Review Process: anonymous review of complete manuscript prior to acceptance.

21. Raz, O., Koopman, P. & Shaw, M., “Benchmarking semantic availability of dynamic data feeds,” *Workshop on Dependability Benchmarking* (in conjunction with DSN 2002), Washington DC, June 2002, 2 page position paper.
Review Process: anonymous review of complete manuscript prior to acceptance.

22. Shelton, C., & Koopman, P., "Using Architectural Properties to Model and Measure System-Wide Graceful Degradation,” *Workshop on Architecting Dependable Systems* (affiliated with ICSE 2002), May 25 2002, 5 pages.
Review Process: anonymous review of complete manuscript prior to acceptance.

23. Nace, W. & Koopman, P., “A Graceful Degradation Framework for Distributed Embedded Systems,” *Workshop on Reliability in Embedded Systems* (in conjunction with Symposium on Reliable

Distributed Systems/SRDS-2001), October 2001. 5 pages.
Review Process: anonymous peer review of complete manuscript prior to publication.

24. Shelton, C. & Koopman, P., "Developing a Software Architecture for Graceful Degradation in an Elevator Control System," *Workshop on Reliability in Embedded Systems* (in conjunction with Symposium on Reliable Distributed Systems/SRDS-2001), October 2001. 5 pages.
Review Process: anonymous peer review of complete manuscript prior to publication.

25. Latronico, E., Martin, C. & Koopman, P., " Analyzing Dependability of Embedded Systems from the User Perspective," *Workshop on Reliability in Embedded Systems* (in conjunction with Symposium on Reliable Distributed Systems/SRDS-2001), October 2001. 5 pages.
Review Process: anonymous peer review of complete manuscript prior to publication.

26. Latronico, E. & Koopman, P., "Representing Embedded System Sequence Diagrams As A Formal Language," *UML 2001*, Toronto Ontario, 3-5 Oct. 2001, pp. 302-316.
Review Process: anonymous peer review of complete manuscript prior to publication. 31% acceptance rate.

27. DeVale, J., & Koopman, P., "Performance Evaluation of Exception Handling in I/O Libraries," *Fault Tolerant Computing Symposium (FTCS-31)/Dependable Systems and Networks*, July 2001, Gotenberg Sweden, pp. 519-524.
Review Process: anonymous peer review of complete manuscript prior to publication; accepted as practical experience report.

28. Madeira, H. & Koopman, P. "Dependability benchmarking: making choices in an n-dimensional problem space," *Workshop on Evaluating and Architecting System dependabilitY (EASY)*, concurrent with *Fault Tolerant Computing Symposium (FTCS-31)/Dependable Systems and Networks*, July 2001, Gotenberg Sweden. 5 pages.
Review Process: anonymous peer review of complete manuscript prior to publication.

29. Nace, B. & Koopman, P., "A Product Family Approach to Graceful Degradation", *DIPES 2000 Workshop* (International IFIP WG 10.3 / WG 10.4 / WG 10.5 Workshop on Distributed and Parallel Embedded Systems), in Paderborn, Germany, 18-19 October 2000.
Review Process: anonymous peer review of complete manuscript prior to publication.

30. Cholkar, A. & Koopman, P., "A Widely Deployable Web-based Network Simulation Framework using CORBA IDL-based APIs," *Winter Simulation Conference*, December 5-8 1999, Phoenix, AZ, pp. 1587-1594.
Review Process: anonymous peer review of complete manuscript prior to publication. The Winter Simulation Conference (WSC) is the premier international forum for disseminating recent advances in the field of system simulation.

31. Koopman, P. & Madeira, H., "Dependability Benchmarking & Prediction: A Grand Challenge Technology Problem," *The 1st International Workshop on Real-Time Mission-Critical Systems: Grand Challenge Problems*, November 30, 1999; Phoenix, Arizona USA. 4 pages.
Review Process: program committee review of complete manuscript prior to publication.

32. Hoover, C., Hansen, J., Koopman, P. & Tamboli, S., "The Amaranth Framework: Probabilistic, Utility-Based Quality of Service Management for High-Assurance Computing," *IEEE Fourth International High-Assurance Systems Engineering Symposium (HASE'99)*, IEEE Computer Society

Press, Los Alamitos, CA, Nov. 17-19, 1999, pp. 207-216.
Review Process: anonymous peer review of complete manuscript prior to publication.

33. Devale, J., Koopman, P. & Guttendorf, D., "The Ballista Software Robustness Testing Service," *Testing Computer Software Conference (TCS-99)*, pp 33-42,1999.
Review Process: anonymous peer review of complete manuscript prior to publication.

34. Koopman, P., Sung, J., Dingman, C. & Siewiorek, D., "Comparing Operating Systems using Robustness Benchmarks," *16th IEEE Symposium on Reliable Distributed Systems*, pp. 72-79, October 22-24, 1997.
Review Process: anonymous peer review of complete manuscript prior to publication.

35. Koopman, P., "A Taxonomy of Decomposition Strategies Based on Structures, Behaviors, and Goals," *Design Theory and Methodology Conference*, Boston, pp. 611-618, September 1995, ASME.
Review Process: anonymous peer review of complete manuscript prior to publication.

36. Koopman, P. "Design Constraints on Embedded Real Time Control Systems," *System Design and Network Architecture Conference*, pp. 71-77, May 8-10, 1990.
Review Process: review of complete manuscript by program committee prior to publication.

37. Koopman, P. "Modern Stack Computer Architecture," *System Design and Network Architecture Conference*, pp. 153-164, May 8-10, 1990.
Review Process: review of complete manuscript by program committee prior to publication.

38. Koopman, P. & Siewiorek, D., "The Impact of Rent's Rule on Massive Parallelism," *Frontiers of Massively Parallel Computation*, pp. 59-62, Fairfax VA, 1988.
Review Process: program committee review of complete manuscript prior to publication.

3.F. Invited Papers and Other Symposium, Conference, and Workshop Proceedings

1. Koopman, P., "Challenges in Autonomous Vehicle Validation," SCAV 2017 Keynote, ES Week, April 21, 2017, Pittsburgh PA. 1 page.
Review Process: program committee review of abstract.

2. Koopman, P., "Embedded System Software Quality: Why is it so often terrible? What can we do about it?" ISSRE 2016 invited keynote, October 25, 2016, Ottawa. 1 page.
Review Process: program committee review of abstract.

3. Koopman, P., "Software Quality, Dependability and Safety in Embedded Systems," SAFECOMP 2014 invited talk, September 2014. 1 page.
Review Process: program committee review of abstract.

4. Koopman, P., & Wagner, M., "Transportation CPS Safety Challenges," NSF Workshop on Transportation CyberPhysical Systems, Washington DC, 2014.
Review Process: program committee review of abstract.

5. Koopman, P., "The Grand Challenge of Embedded System Dependability," DSN 2011 panel session, June 2011. 1 page.
Review Process: program committee review of abstract.

6. Koopman, P., “Avoiding the top 43 embedded software risks,” Embedded Systems Conference Silicon Valley, May 2, 2011. 4 pages.
   Review Process: program committee review of abstract.

7. Koopman, P., "Challenges in representing CPS safety," Workshop on developing dependable and secure automotive cyber-physical systems from components, Flint MI, Mar 17-18, 2011.
   Review Process: Position paper reviewed by program committee for NSF workshop.

8. Koopman, P. & Ray, J., "Mitigating the Effects of Internet Timing Faults Across Embedded Network Gateways," MMB/DFT 2010, p. 1, March 2010.
   Review Process: Invited keynote talk with accompanying extended abstract.

9. Koopman, P., “Reliability, Safety, and Security in Everyday Embedded Systems,” *Latin American Dependability Conference (LADC-07)*, Morelia, Mexico, Sept. 26-28, 2007.
   Review Process: Invited keynote talk with accompanying extended abstract.

10. Koopman, P., Morris, J., Maxino, T., “Position Paper: Deeply Embedded Survivability,” *ARO Planning Workshop on Embedded Systems and Network Security*, Raleigh NC, February 22-23, 2007.
    Review Process: program committee review of position paper.

11. Koopman, P., Morris, J. & Narasimhan, P., "Challenges in Deeply Networked System Survivability," *Nato Advanced Research Workshop On Security and Embedded Systems*, August 2005.
    Review Process: Invited paper.

12. Shelton, C., Koopman, P. & Nace, W., “A framework for scalable analysis and design of system-wide graceful degradation in distributed embedded systems,” WORDS 2003.
    Review Process: Invited paper; reviewed by PC chairs.

13. Koopman, P. & Madeira, H., "Workshop on Dependability Benchmarking," *Proc. International Conference on Dependable Systems and Networks (DSN)*, July 2002.
    Review Process: Invited paper in main proceedings.

14. Koopman, P., “Critical Embedded Automotive Networks,” IEEE Micro, July 2002, pp. 14-18.
    Review Process: Editorial introduction and summary of high-level issues written in my capacity as guest editor.

15. Beveridge, M. & Koopman, P., “Jini Meets Embedded Control Networking: a case study in portability failure,” *Seventh IEEE Workshop on Object-Oriented Real-Time Dependable Systems: WORDS 2002*, San Diego, January 2002, 8 pages.
    Review Process: Invited paper; accepted after review of complete manuscript by program committee.

16. Pan, J., Koopman, P. & Siewiorek, D., “A Dimensionality Approach To Testing and Improving Software Robustness,” *Autotestcon 99*, San Antonio, Texas, August 30 – September 2, 1999.
    Review Process: program committee review of extended abstract prior to acceptance. Awarded ***best technical paper*** for the conference.

17. Koopman, P., “Embedded System Design Issues—The Rest of the Story,” *Proceedings of the 1996 International Conference on Computer Design*, pp. 310-317. October 7-9, 1996, Austin, Texas. 90-minute tutorial given at conference based on paper.

Review Process: invitation and program committee review of abstract as a paper accompanying a special tutorial.

18. Upender, B. & Koopman, P. "Structured Functional Modeling in SES/workbench," *1994 SES User Group Conference*, Austin TX, 16 pages, April 1994. [Proceedings published only the presentation slides.]

19. Koopman, P., "A Brief Introduction to Forth," *ACM SIGplan Notices*, vol. 28, no. 3, pp. 357-358, March 1993, (History of Programming Languages HOPL-II preprints issue).
Review Process: invited based on my services as the conference's Forth Language Expert.

20. Upender, B. & Koopman, P., "Embedded Communication Protocol Options," *Proceedings of Embedded Systems Conference 1993*, Santa Clara, pp. 469-480, October 1993; repeated in *Proceedings of Embedded Systems Conference East 1994*, Boston, April 1994. (Paper plus two-hour tutorial.)

21. D'Anniballe, J. & Koopman, P., "Towards Execution Models of Distributed Systems: a case study of elevator design," *ICCD Workshop on Hardware/Software Codesign*, Boston, October 1993. 9 pages.

22. Koopman, P. "Some Ideas for Stack Computer Design," *1991 Rochester Forth Conference*, pg. 58, June 1991. [Presentation slides only.]

23. Koopman, P. "TIGRE: Combinator Graph Reduction on the RTX 2000," *1990 Rochester Forth Conference*, pp. 82-86, June 1990.

24. Koopman, P. "Architectural Opportunities for Future Stack Engines," *1990 Rochester Forth Conference*, pp. 79-81, June 1990.

25. Koopman, P. & VanNorman, R. "Adding a Third Stack to a Forth Engine," *1990 Rochester Forth Conference*, pp. 150-151, June 1990.

26. Koopman, P. & VanNorman, R., "RTX 4000," *1989 Rochester Forth Conference*, pp. 84-86, June 6-10, 1989.

27. Koopman, P. "32 Bit RTX Chip Prototype," *Journal of Forth Application and Research* (Rochester Forth Conference Proceedings), vol. 5, no. 2, pp. 331-335, 1988.

28. Koopman, P., "Writable Instruction Set Stack Oriented Computers: The WISC Concept," *Journal of Forth Application and Research* (Rochester Forth Conference Proceedings), vol. 5, no. 1, pp. 49-71, 1987.
Review Process: one of five invited papers.

29. Haydon, G., & Koopman, P., "MVP Microcoded CPU/16: History," *Journal of Forth Application and Research* (Rochester Forth Conference Proceedings), vol. 4, no. 2 , pp. 273-276, 1986.

30. Koopman, P., & Haydon, G., "MVP Microcoded CPU/16: Architecture," *Journal of Forth Application and Research* (Rochester Forth Conference Proceedings), vol. 4, no. 2, pp. 277-280 1986.

3.G. Sections In Volumes of Collected Papers

1. Koopman, P., "Fractal Landscapes," *Dr. Dobb's Toolbox of Forth Vol. II*, M&T Publishing, pp. 347-356, 1988.

2. Koopman, P., "Bresenham Line Drawing Algorithm," *Dr. Dobb's Toolbox of Forth Vol. II*, M&T Publishing, pp. 357-365, 1988.

3.H. Published Abstracts, Discussions, Reviews

1. Interview quotes on the topic of automotive security. http://finance.yahoo.com/news/worried-car-getting-hacked-fbi-154514990.html, 3/25/2016.

2. Half-hour interview, DMCA exemption for reverse engineering automotive and Internet of Things software, Knowledge@Wharton XM Radio Channel 111, 9/30/2015. https://businessradio.wharton.upenn.edu/bestof/knowledge-@wharton/?h=KXK2e&t=4m51s

3. Interview quotes on the topic of automotive software and security. NY Times 9/26/2015, NetworkWorld 9/27/2015, Huffington Post 9/28/2015, others.

4. Koopman, P., "Flexible multicast authentication for time-triggered embedded control network applications," Report from Dagstuhl Seminar 11441, p. 1.

5. Morris, J. & Koopman, P., "Software Defect Masquerade Faults in Distributed Embedded Systems," FastAbs, *DSN*, June 2003, 2 pages.
   Review Process: program committee review of complete extended abstract prior to publication.

6. Interviews and explanations to US Press about the impact of the UCITA and UCC Article 2 legislative changes on embedded computer systems and computer products. (UCITA is the Uniform Computer Information Transactions Act; UCC is the Uniform Code of Commerce.) Interviews and coverage of talks appeared in: *The Wall Street Journal* (December 11, 2000 page B1), *Infoworld* (December 1, 2000), Bureau of National Affairs publications (December 3, 2000), Pike & Fischer I, Volume 2, No. 44, November 10, 2000, and Pike & Fischer *dot.com IP advisor*, Volume 1, No. 9, November 10, 2000.

7. Koopman, P., Tran, E. & Hendrey, G. "Toward Middleware Fault Injection for Automotive Networks," *Fault Tolerant Computing Symposium*, pp. 78-79, June 23-25, 1998.
   Review Process: program committee review of complete extended abstract prior to publication.

8. Unger, S. (moderator), "Doing the Right Thing," *IEEE Spectrum*, Ethics Roundtable, December 1996, pp. 25-32. [I was an invited participant.]

9. Koopman, P., "Stack Machines," in "Usenet Nuggets," Mark Thorson, (ed.), *Computer Architecture News*, vol. 21, no. 1, pp. 36-37, March 1993.
   Review process: Usenet post selected for publication by column editor.

3.I. Other Writings

1. Koopman, "How to Make Self-Driving Car Road Testing Safe," Embedded.com, 4/2/2018. https://ubm.io/2H3Hr9j (Designlines Automotive Op-Ed.)

2. Koopman, “A rebuttal to “Why every embedded software developer should care about the Toyota verdict,” embedded.com, 12/6/2017. https://goo.gl/DGbnyE (Letter to editor)

3. Koopman & Wagner, “Safe self-driving? It’s not there yet: a lot of testing remains before self-driving cars should be widely deployed.” Op-Ed, *Pittsburgh Post-Gazette*, 9/30/2016.

4. Koopman, Foreword to *Trustworthy Cyber-Physical Systems Engineering*, edited by A. Romanovsky and F. Ishikawa, CRC Press, 2016.

5. Koopman, Driscoll, Hall, "Selection of Cyclic Redundancy Code and Checksum Algorithms to Ensure Critical Data Integrity," Final Report, DOT/FAA/TC-14/49, March 2015.

6. Koopman, “Why you need a software specific test plan,” Blogger of the Month, LogiGear Magazaine, August 2014, Vol. VIII, Issue 3, pp. 21-22.

7. Koopman, P. & Szilagyi, C., "Integrity in Embedded Control Networks," *IEEE Security & Privacy*, 2013, pp. 61-63.

8. Wagner, M., Koopman, P., Bares, J., and Ostrowski, C. (2009) Building safer UGVs with run-time safety invariants. National Defense Industrial Association Systems Engineering Conference. (Presentation-only conference publication.)

9. Kanoun, K., Spainhower, L., Koopman, P., Madeira, H., “Dependability Benchmarking – A Reality or a Dream?,” *Dependability Benchmarking*, IEEE press, 2008. (Introduction to book.)

10. Koopman, P., Foreword. In: Rogério de Lemos (Editor), Cristina Gacek (Editor), Alexander Romanovsky (Editor), *Architecting Dependable Systems III*, Springer, 2005.

11. Koopman, P., “Embedded System Security,” *IEEE Computer*, July 2004, pp. 95-97.

12. Koopman, P. and Hoffman, R., “Work-arounds, make-work, and kludges” *IEEE Intelligent Systems*, November/December 2003, pp. 70-75.

13. Orna Raz, Rebecca Buchheit, Mary Shaw, Philip Koopman, and Christos Faloutsos, *Eliciting User Expectations for Data Behavior via Invariant Templates*, Technical report CMU-CS-03-105, January 2003.

14. Koopman, P. & Madeira, H. (eds.), *Proceedings on the Workshop on Dependability Benchmarking*, section within *International Conference on Dependable Systems and Networks (DSN) Supplement*Washington DC, July 2002.

15. Koopman, P. & Kaner, C., “The problem of embedded software in UCITA and drafts of revised Article 2,” *UCC Bulletin*, 5 pages in February 2001; 5 pages in March 2001; 6 pages in April 2001.
Note: This bulletin is a very influential publication venue for influencing the opinions of the commerce code legal community. Fully half of each of three back-to-back issues is devoted to our explanation of how software regulations are drafted in a way that will compromise the safety and reliability of embedded systems, and they have significantly influenced opinions within the national lawmaking process.

16. Koopman, P. & Chakravarty, T., "Analysis of the Train Communication Network Protocol Error Detection Capabilities," http://www.iec-tcn.org/publications.html, February 25, 2001, 8 pages.
Note: this is an analysis of an international protocol standard published by the standardization group.

17. Tran, E. & Koopman, P., *Mission Failure Probability Calculations for Critical Function Mechanizations in the Automated Highway System*, Technical Report CMU-RI-TR-97-44, Carnegie Mellon University, December 16, 1997. 34 pages.

18. Koopman, P., "Tracking Down Lost Messages and System Failures," *Embedded Systems Programming*, vol. 9, no. 11, pp. 38-52, October 1996.

19. Koopman, P., "On Being the Bearer of Bad News" (engineering ethics), *The Institute*, IEEE, vol. 20, no. 6, pg. 15, June 1996. Reprinted in *Engineering Dimensions*, Professional Engineers of Ontario, January 2000 pp. 25-26.

20. Koopman, P., "Synthesis Meets an Industrial Application (poster session)," *HW/SW Codesign Workshop*, Pittsburgh, PA, April 1996; published on WWW.

21. Koopman, P., *Obstacles to Using CAD Tools for Embedded System Design: an automotive case study*, Technical Report EDRC 05-103-96, Carnegie Mellon University, Engineering Design Research Center, 1996. 16 pages.

22. Koopman, P., *Design-by-Composition Technology Assessment*, Technical Report RR-9600018, United Technologies Research Center, 1996. 5 pages

23. Koopman, P., Upender, B. & Dean, A., *Ten Problems with Using Echelon's LonWorks Technology for Embedded Control Applications*, Technical Report RR-9500490, United Technologies Research Center, 1995. 22 pages

24. Koopman, P. *RFA-B Secret ID Manufacturing Process*, Technical Report, United Technologies Research Center, 1995. [proprietary contents, but much of this is published in U.S. Patent #5,757,923 and US Patent #5,696,828.]

25. Koopman, P. & Upender, B., *Time Division Multiple Access without a Bus Master*, Technical Report 9500470, United Technologies Research Center, 1995. 15 pages

26. Upender, B. & Koopman, P., "Communication Protocols for Embedded Systems," *Embedded Systems Programming*, vol. 7, no. 11, pp. 46-58, November 1994.

27. Koopman, P., *Technology Trends in Embedded Processing*, Technical Report RR-9400488, United Technologies Research Center, 1994. (proprietary)

28. Koopman, P., "Perils of the PC Cache," *Embedded Systems Programming*, vol. 6, no. 5, pp. 26-34, May 1993.

29. Keown, W., Koopman, P. & Collins, A., "Performance of the Harris RTX 2000 Stack Architecture versus the Sun 4 Sparc and the Sun 3 M68020 Architectures," *Computer Architecture News*, June 1992, vol. 20, no. 3, pp. 45-52.

30. Keown, W., Koopman, P. & Collins, A., "Real-Time Performance of the Harris RTX 2000 Stack Architecture versus the Sun 4 SPARC and the Sun 3 M68020 Architectures with a Proposed Real-

Time Performance Benchmark," *Performance Evaluation Review*, May 1992, vol. 19, no. 4, pp. 40-48.

31. Koopman, P., "Testing Toolkit," *Forth Dimensions,* vol. 12, no. 3 , pp. 31-32; 41, September 1990.

32. Koopman, P., "Heavyweight Tasking," *Embedded Systems Programming*, vol. 3, no. 4, pp. 42-52, April 1990.

33. Koopman, P., "Design Tradeoffs in Stack Computers," *Forth Dimensions*, vol. 11, no. 6, pp. 5-9, March 1990.
Review Process: anonymous peer review of complete manuscript prior to publication. First place winner in best-paper contest.

34. Koopman, P., "Embedded control as a path to Forth acceptance," SIGForth 1990, pp. 23-26.

35. Koopman, P., *An Architecture for Combinator Graph Reduction*, PhD Thesis, Electrical and Computer Engineering Department, Carnegie Mellon University, 1989.

36. Lee, P. & Koopman, P., *Compiling for Direct Execution of Combinator Graphs*, Ergo report, Carnegie Mellon University, Pittsburgh, 1989. 15 pages

37. Koopman, P., "Transcendental Functions," *Forth Dimensions*, vol. 9, no. 4, pp. 21-22, September 1987.

38. Koopman, P., "Fractal Landscapes," *Forth Dimensions,* vol. 9, no. 1, pp. 12-16, May 1987.

39. Koopman, P., "Bresenham Line-Drawing Algorithm," *Forth Dimensions*, vol. 8, no. 6, pp. 12-16, March 1987.

40. Koopman, P., "The WISC Concept," *BYTE*, vol. 12, no. 4, pp. 187-194, April 1987.

41. Koopman, P., "Microcoded vs. Hard-Wired Control," *BYTE*, vol. 12, no. 1, pp. 235-242, January 1987.

42. Koopman, P., "Redefining Words," *Forth Dimensions*, vol. 7, no. 4, pp. 36-37, November 1985.

43. Koopman, P., *CADAM Training on an IBM Personal Computer: Low Level Software and Hardware*, M.Eng. Thesis, Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy NY, 1982.

3.J. Discussion or Reviews of Work

1. Levy, N., "Our robot neighbors: Hanging out with the mechanical inhabitants of 'Robotics Row'," Geekwire, February 21, 2018.
Article content based on interview.

2. Yoshida, J., "Experts weigh in on Modbileye's AV Safety Model," EE Times, October 26, 2017. Article based on an interview.

3. Yoshida, J., "5 Unresolved Issues Facing Robo-cars," EE Times, April 13, 2017. Article includes materials from one of my slide presentations and a brief discussion of my ITS paper.

4. Plungis, J., “Self-Driving Cars: Driving Into the Future,” Consumer Reports Magazine, April 2017.
Article based on an interview.

5. Plungis, J., “With Autonomous Cars, How Safe Is Safe Enough?” Consumer Reports Magazine, April 2017.
Article based on an interview.

6. Stackpole, B., “Self-Driving Cars Test Traditional Procedures,” Digital Engineering, 10/1/2016, http://www.digitaleng.news/de/self-driving-cars-test-traditional-procedures/
Article based on an interview.

7. Silver, A., “Why AI Makes It Hard to Prove That Self-Driving Cars Are Save,” IEEE Spectrum, 10/7/2016, http://spectrum.ieee.org/cars-that-think/transportation/self-driving/why-ai-makes-selfdriving-cars-hard-to-prove-safe
Article based on an interview.

8. Yoshida, J., “Deep Learning: Achilles Heel in Robo-Car Tests,” 10/3/2016,
http://www.eetimes.com/document.asp?doc_id=1330561&print=yes
Article based on my response to DoT proposed policy for highly automated vehicle safety.

9. Yoshida, J., “Robo-Car’s Safety Challenges DoT,” 9/21/2016,
http://www.eetimes.com/document.asp?doc_id=1330494&print=yes
Article based on an interview and the contents of my SAE World Congress presentation.

10. Yoshida, J., “Questions About Tesla Autopilot Safety Hit Stone Wall,” 9/16/2016,
http://www.eetimes.com/document.asp?doc_id=1330477&print=yes
Extensively quotes one of my blog posts on safety argument calculations based on test data.

11. ASEE Prism, “How Software Can Kill,” Special issue on Troubled Waters: the rewards and risks for academic engineers who plunge into public controversies, April 2016, page 12.

12. Arbesman, S., “Overcomplicated,” 2016 (augmented with personal correspondence)
References my work on the Toyota UA cases and acknowledgement for contributing to book development.

13. Herdman, P., “When Cars Decide to Kill”, 2015 (augmented with personal correspondence)
Acknowledges my Toyota UA presentation as an important work in the field and says it was an impetus for writing her book.

14. FAA, AC 00-66, “Selection of cyclic redundancy code and checksum algorithms to ensure critical data integrity,” Aug 4, 2015.
Promotes FAA CRC & Checksum report TC14-49 as reference material for the aviation industry.

15. Ganssle, J., “New Embedded Systems Books,” *Embedded Systems Design*, Dec. 2010, pp. 34-36.
Positive review of *Better Embedded System Software* book.

16. MIT Technology Review, “10 Emerging technologies that will change the world,” Feb. 2003.
Mention as leading researcher in the area of mechatronics.
http://www.technologyreview.com/articles/print_version/emerging0203.asp

17. Koerner, B., The Bugs in the Machine, Embedded Code Is The Future – Get Ready To Reboot, *Wired*, August 2002. *http://www.wired.com/wired/archive/10.08/start.html?pg=2* Article is based in part on an extended discussion I had with the author on this topic. I'm quoted; much of the rest is based on our discussion as well.

18. *ABB Technology Report 2000*, ABB Corporation. (Half-page discussion of our joint project applying Ballista to ABB software is featured in the corporate-wide annual report from the ABB Chief Technical Officer.)

19. Plotnick, Neil, "Net Adviser: Sometimes, IT success is academic," *PC Week*, August 30, 1999. In print and on-line at: *http://www.zdnet.com/pcweek/stories/columns/0,4351,2320978,00.html*. (Discusses Ballista as a relevant academic research project.)

20. Dixard, Wilson, "DARPA project probes COTS software testing and hardening," *Military & Aerospace Electronics*, March 1998, pp. 1, 13. (Front page coverage of the Ballista project in the trade press)

21. Bailey, C.; Sotudeh, R.; Ould-Khaoua, M.; "The effects of local variable optimisation in a C-based stack processor environment," *Proceedings The European Forth Conference, EuroForth '94; Winchester, UK; 4-6 Nov.*, pp. 17-22, 1994. (Builds upon stack processor compiler work with extensive comparision of techniques to my previous work.)

3.K. Patents

1. *Method and apparatus for location-sensitive, subsidized cell phone billing*, US Patent #7,668,765, Tanaka, Hendrey & Koopman, February 23, 2010.

2. *Method and system for analyzing advertisements delivered to a mobile unit*, US Patent #6,647,269; Hendrey, Tanaka & Koopman, November 11, 2003.

3. *Method and system for selectively connecting mobile users based on physical proximity*, US Patent #6,542,750; Hendrey, Tanaka & Koopman, April 1, 2003.

4. *Method and system for connecting proximately located mobile users based on compatible attributes*, US Patent #6,542,749; Tanaka, Hendrey, Koopman & King, April 1, 2003.

5. *Method and system for automatically initiating a telecommunications connection based on distance*, US Patent #6,542,748; Hendrey, Koopman, King & Tanaka, April 1, 2003.

6. *Method and system for connecting mobile users based on degree of separation*, US Patent #6,539,232; Hendrey, Tanaka, Koopman & King, March 25, 2003.

7. *Vehicle anti-theft system including vehicle identification numbers programmed into on-board computers*; US Patent #5,991,673; Koopman & Carroll, November 23, 1999.

8. *Wireless automotive door*; US Patent #5,975,622; Koopman, November 2, 1999.

9. *System and method of updating communications in a security system*; US Patent #5,952,937; Koopman, Carroll, Grzybowski, Marshall, September 14, 1999.

10. *Security system with random number remote communication*, U.S. Patent #5,940,002; Finn, Koopman & Carroll, August 17, 1999.

11. *Vehicle security system with combined key fob and keypad anti-driveaway protection*. U.S. Patent #5,783,994; Koopman, Carroll, Grzybowski & Marshall, Jul. 21, 1998.

12. *Method of generating secret identification numbers.* U.S. Patent #5,757,923; Koopman, May 26, 1998.

13. *Random number generating system and process based on chaos*. US Patent #5,696,828; Koopman, Dec. 9, 1997.

14. *Elevator position determination*. US Patent #5,682,024; Koopman & Finn, Oct. 28, 1997.

15. *Elevator level control system using elevator/landing gap as a reflection duct.* US Patent #5,659,159; Koopman, Aug. 19, 1997.

16. *Pseudorandom composition-based cryptographic authentication process*. US Patent #5,649,014; Koopman, Finn & LaBarre, Jul. 15, 1997.

17. *Pseudorandom composition-based cryptographic authentication process.* US Patent #5,619,575; Koopman, Finn & LaBarre, Apr. 8, 1997.

18. *Random clock composition-based cryptographic authentication process and locking system.* US Patent #5,598,476; LaBarre & Koopman, Jan. 28, 1997.

19. *Implicit Token Media Access Protocol Without Collision Detection.* US Patent #5,535,212; Koopman & Brajczewski, Jul. 9, 1996.

20. *Explicit and Implicit Token Media Access Protocol Arbitration*. US Patent #5,450,404; Koopman & Brajczewski, Sep. 12, 1995.

21. *Synchronous Time Division Multiplexing Using Jam-Based Frame Synchronization*. US Patent #5,436,901; Koopman, Jul. 25, 1995.

22. *Cryptographic Encoding Process.* US Patent #5,398,284; Koopman & Finn, Mar. 14, 1995. (Also issued as European Patent EP727118B1.)

23. *Cryptographic Authentication of Transmitted Messages Using Pseudorandom Numbers.* US Patent #5,377,270; Koopman & Finn, Dec. 27, 1994. (Re-issued with 64 claims as US Patent #RE36,752.)

24. *Pseudorandom Number Generation and Cryptographic Authentication.* US Patent #5,363,448; Koopman & Finn, Nov. 8, 1994. (Also issued as European Patent EP00706735B1; Re-issued with 93 claims as US Patent #RE036181.)
    This patent (together with the preceding two patents), was at last count the basis for more than $150 million of business for United Technologies Automotive. The technology it covers is listed as **one of the two key technical contributions of United Technologies Research Center in 1993**.
    Note: My patents account for 4% (14 out of 350) of the total issued to all of United Technologies Corporation ($24B sales, 150,000 employees) from 1994-1998 (sources: U.S. Patent Office, and *Technology Review*, March/April 2000, pg. 87)

25. *Stack-Memory-Based 32-Bit Writable Instruction Set Computer Having a Single Data Bus.* US Patent #5,053,952; Koopman & Haydon, Oct. 1, 1991.
This patent was licensed to Harris Semiconductor as the basis for their 32-bit CPU designs.

26. *Stack-Memory-Based Writable Instruction Set Computer Having a Single Data Bus*. US Patent #4,980,821; Koopman & Haydon, Dec. 25, 1990.

4. Grants and Contracts Awarded to Date

1. (P.I.) US Army, "Robustness Inside-Out Testing: Automated White-Box Stress Testing of UAS Software (RIOT)," BAA, up to approximately $6.5M over 3 years. Awarded. (NREC collaboration: co-PI Mike Wagner). 2015-2018.

2. (P.I.) US Army, "BAA UAST CMU STAA W900KK-11-C-0025-P00019 Anomaly Detection ECP," Anomaly detection algorithms in support of safety invariant creation. $600,000 over 12 months. Awarded (NREC collaboration: co-PI Mike Wagner). 2015-2016.

3. (P.I.) US Army, "BAA UAST CMU STAA W900KK-11-C-0025-P00015 AMAS Safety ECP," Safety evaluation process and architecture for AMAS. $500,000 over 12 months. Awarded (NREC collaboration: co-PI Mike Wagner). 2014-2015.

4. (P.I.) CMU-SYSU Collaborative Innovation Research Center (CIRC), "Measuring effectiveness of watchdog timers for embedded software safety," $148,980 over 12 months. Awarded. 2015.

5. (P.I.) US Army, "BAA UAST CMU STAA W900KK-11-C-0025-P00009 SFM ECP," Semi-Formal Monitoring of Autonomous Vehicles (SFM) and additional AMAS funding. $1,535,083 over 14 months. Awarded (NREC collaboration: co-PI Mike Wagner). 2013-2014.

6. (P.I.) Volkswagen Group of America, "Stress testing of path planning module for an autonomous vehicle," $50,000 over one year. Awarded (NREC collaboration: co-PI Mike Wagner). 2012-2013.

7. (P.I.) US Army, "BAA UAST CMU STAA W900KK-11-C-0025-P00005 AMAS ECP," Stress-testing the Autonomous Mobile Applique System (AMAS). $610,984 over one year. Awarded (NREC collaboration: co-PI Mike Wagner). 2012-2013.

8. (P.I.) US Army, "Unmanned and Autonomous Systems Test (UAST) Science and Technology (S&T): A methodology for stress-testing autonomy architectures," BAA W9000KK-09-R-0038 topic #3, $1,834,165 over 3 years. Awarded. (NREC collaboration: co-PI Mike Wagner). 2011-2014.

9. (P.I.) FAA, "Software and digital systems program – data integrity techniques," DTFACT-11-R-00002, $150K over 18 months, including subcontract to Honeywell, Awarded. 2011-2013.

10. (Co-P.I.) CMU CERT, "Embedded virtualization," seed money for proposal development, 1 student for 3 months, 2011. Awarded.

11. (P.I.) General Motors Corporation, "Dependable Systems," 1/00-12/15, full support for 1 to 3 students (varies by year) as one of four thrust area leaders in the CMU/GM research laboratory. Awarded.

12. (P.I.) John Deere Co., "Embedded System Security," 1/09-12/09, $44K unrestricted grant. Awarded.

13. (Co-P.I.) US Army TARDEC, "Safety Subsystem" task within contract for "Autonomous Platform Demonstrator (APD)". $35K available contract task funds within $14M program. 2010. Awarded.

14. (P.I.) National Robotics Engineering Consortium (NREC), "Autonomous vehicle safety seed funding," $18K. 2008. Awarded.

15. (P.I.) Thyssenkrupp, "Distributed Elevator Architecture," $9000 grant, 1/2007. Awarded.

16. (P.I.) PITA, “Embedded Computing Safety: Architectures and Design Tactics,” 9/1/05-12/31/06, $50,000. Awarded.

17. (P.I.) Bosch, “Safety Critical Architecture Assessment,” 10/1/05-3/1/06, $50,000. Awarded.

18. (co-P.I.) Federal Aviation Administration (FAA), Evaluation Criteria for Databuses, $70,000 subcontract over two years, 2005-2006. Awarded.

19. (P.I.) Honeywell, X-by-wire protocol dependability, $25,000 grant, 2003; $25,000 grant, 2004; $12,000 grant, 2005.

20. (P.I.) DaimlerChrysler, distributed embedded system dependability, $18,000 grant, 2003.

21. (P.I.) PITA, General Theory of Critical Embedded Networks, 11/02-12/03, $45,271.

22. (co-P.I.) NASA/Ames, “High Dependability Computing Program,” 12/01-5/03, $4.66 million/year total spread among consortium members. Awarded; I was one of five thrust leaders. Approximately $100,000/yr.

23. (P.I.) Bombardier Corporation (formerly ADtranz), “Embedded Control Systems,” $17,800 grant, 2002.

24. (P.I.) ADtranz Corporation, “Dependability Assessment of a Train Control Network Protocol”, $38,000 grant. PITA matching funds of $50,000 for 2001.

25. (P.I.) AT&T Corporation, “Robustness testing of CORBA”, $20,000 unrestricted grant for 2001.

26. (P.I.) Bosch Corporation, “Intelligent Sensors,” 4/00-6/01, $100,000.

27. (P.I.) Microsoft Corporation, unrestricted grant for $80,000 based on contributions in robustness testing of Windows. 4/00.

28. (P.I.) ADtranz Corporation, “Embedded System Architecture,” 11/99-11/02, $75,000 grant, matched with $50,000 from PITA (Pennsylvania state funding).

29. (Co-P.I.) National Science Foundation, "CISE Experimental Partnerships: Processing/Communication/ Interface Tradeoffs to Optimize Energy Locality in Mobile Systems," 9/99-9/02, $948,035.

30. (P.I.) Emerson Electric Corporation, grant for work on software robustness. 7/99, $50,000.

31. (P.I.) ABB Corporation, “Application of Ballista to Windows/COM software,” 7/99-7/02, $330,000 over three years.

32. (P.I.) Cisco Corporation, grant for work on software robustness. 5/99, $50,000.

33. (Co-P.I.) Amaranth: A Real-Time System for Adaptive Optimization of End-to-End Quality of Service, DARPA, N66001-97-C-8527, 7/97-6/00, $1,383,994. (primary author of proposal)

34. (Co-P.I.) *Automated Highway System*, CMU Robotics Institute from U.S. Department of Transportation. FY 97-98, $233,891 for reliability.

35. (P.I.) Ballista: A System for Automated Hardening of COTS Components, DARPA, DABT63-96-C-0064, 9/96-9/99, $1,221,345.

36. (Co-P.I.) *System Robustness Through Middleware Software*, United Technologies Research Center, $70,000 1996; $50,000 1997 (co-investigator).

5. Professional Activities

5.A. Seminars, Invited Conference Talks & Tutorials

1. Invited talk, "A Case Study of Unintended Acceleration and Software Safety." Venues: Google Mountain View, 8/28/2014; National Robotics Engineering Center, 9/16/2014; CMU ECE Department Seminar, 9/18/2014; CMU Software Engineering Department (ISR) Seminar, 9/22/2014; Software Engineering Institute, 9/30/2014; Opening Keynote, TSP Symposium, Pittsburgh PA, 11/4/2014; University of California at Berkeley, 11/14/2014; Sun Yat-Sen University SMIE, Zhuhai China, 1/9/2015; Johns Hopkins University APL, 1/16/2015; Georgia Tech.; 2/10/2015; General Motors 2/12/2015; TARDEC 2/12/2015; Keynote 3rd Scandinavian Conference on System and Software Safety, Stockholm, 3/24/15; Renault Safety Group, Paris, 3/26/15; SAE World Congress, Detroit, 4/22/2015; Ford Motor Company Engineering, Dearborn MI, 4/23/2015; NITRD Washington, 6/3/2015; Chalmers University SAFER Consortium, Gothenburg Sweden; 6/15/2015; Volvo AB (Volvo Trucks), Gothenburg Sweden; 6/16/2015, Volvo Cars, Gothenburg Sweden, 6/17/2015; LG Electronics (CMU Executive Education), Pittsburgh, 6/24/2015, 1/22/2016, 6/3/2016; Lutron 7/1/2015; Ajou University Summer Program (CMU Executive Education) 7/23/16, 7/25/2016; Keynote High Integrity Software, Bristol UK, 11/5/2015; Keynote Software Solutions Conference, Washington DC 11/17/2015; IFIP WG 10.4 1/11/2015, University of Alabama invited lecture 2/4/2015, Emerson Software Center of Excellence Tutorial Pittsburgh 2/10/2016; Mine Safety Appliance 4/15/16, Raytheon High Integrity Software group Tucson AZ 5/2/2016; Raytheon Technical Network Symposium Keynote Tucson AZ 5/3/16; Emerson Process Management St. Louis May 5, 2016; Fisher Valve Marshalltown IA 6/1/2016; LG Executive Education Pittsburgh 6/3/2016; Ajou University Summer BootCamp Pittsburgh 7/26/2016; Emerson Electric Software Process Improvement Network Webinar 8/8/2016, LGE Executive Training 6/12/2017, 11/9/2017.

2. Workshop keynote talk, "Autonomous Vehicle Perception Validation," Mathworks Research Summit, Boston, June 2, 2018.

3. Workshop talk, "Toward a Framework for Highly Automated Vehicle Safety Validation," AUVSI XPonential, Denver, May 1, 2018.

4. Invited plenary talk, "Ensuring Safe Road Testing of Self-Driving Cars," 9th China-U.S. Transportation Forum, Beijing China, April 26, 2018. (US autonomous vehicle technical expert, invited by Administrator of NHTSA and US Secretary of Transportation.)

5. Workshop talk, "Ensuring Safe Road Testing of Self-Driving Cars," EWICS Spring 2018 meeting, Munich Germany, April 24, 2018.

6. Keynote talk,, "Ensuring the Safety of On-Road Self-Driving Car Testing," PA AV Summit, Pittsburgh, April 9, 2018. (Invited by PennDOT.)

7. Keynote Talk, "Highly Autonomous Vehicle Validation: It's More Than Just Road Testing!" TechAD conference, Detroit, Nov. 17, 2017.

8. Keynote Talk, "Challenges and Solutions in Autonomous Vehicle Validation," ANSYS workshop on autonomous vehicle simulation, Detroit, Oct. 20, 2017.

9. Keynote Talk, "Challenges and Solutions in Autonomous Vehicle Validation," SRC Meeting on autonomous vehicles, Dallas, Sept. 28, 2017.

10. Keynote Talk, “Challenges and Solutions in Autonomous Vehicle Validation,” Autonomous Vehicle Summit, Detroit, June 24, 2017.

11. Invited panelist, Automated Vehicle Leading Researcher Panel, NHTSA Enhanced Safety of Vehicles Conference, Detroit, June 5, 2017.

12. Visiting evaluation committee, Digital Safety & Security Department, Austrian Institute of Technology, Vienna, April 28-29, 2016.

13. Koopman, P., “Embedded System Software Quality: Why is it so often terrible? What can we do about it?” ISSRE 2016 invited keynote, October 25, 2016, Ottawa.

14. Keynote Talk, “Embedded Software Quality, Safety and Security,” LGE Software Quality Conference, Seoul Korea, 7/19/2016.

15. Tutorial series on Embedded Software Quality (half day), Embedded Safety (half day), and Embedded Security (half day), Emerson Electric, Xi’an, 7/11-7/13/2016.

16. One-day tutorial on Software Quality and Embedded Software Safety, Emerson Electric. Pittsburgh 2/10/2016. Repeated in St. Louis 5/4/2016; Marshalltown IA 5/31/2016. Half-day version in Shenzhen 7/5/2016.

17. Keynote Talk, “Challenges in Autonomous Vehicle Testing and Deployment,” SSIV workshop (DSN), Toulouse France, 6/28/16. Also, invited panel session member on Cyber Security & Safety.

18. Invited Talk, “Validation of Autonomous Vehicles,” SCC (HCSS workshop), Annapolis, 5/9/16; HILT Workshop on Model-Based Development (SIGAda) Pittsburgh 10/7/2016.

19. Webinar, “Introduction to Software Safety and IEC 60730,” Emerson corporation, 4/15/2016.

20. Invited talk, “Validation and stress testing of autonomous systems,” Workshop on Autonomous System Safety, Stockholm, Sweden, March 25, 2015

21. Invited talk, “An exercise in applying junk science criteria to fuzz testing,” IFIP WG 10.4 meeting, Bristol, UK, January 23, 2015.

22. Invited talk, “Safety Critical Hardware and Software Issues,” Attorneys Information Exchange Group, San Antonio TX, October 22, 2014.

23. Invited talk, “Data Integrity Techniques,” 2014 National Systems, Software and Airborne Electronic Hardware Standardization Conference, (FAA Designated Engineering Representative Symposium), Los Angeles, September 25, 2014.

24. Invited panel session, “CPS Education Experiences & CPS Faculty Perspective,” National Research Council/National Academies, Washington DC, October 3, 2014.

25. Keynote Talk, “Software Quality, Dependability and Safety in Embedded Systems,” SAFECOMP 2014, Florence Italy, Sept. 11, 2014.

26. Invited talk, “Software Quality, Dependability and Safety in Embedded Systems,” Google, Mountain View CA, August 28, 2014.

27. Invited talk, “Software robustness testing and run-time monitoring of autonomous vehicles,” Carnegie Mellon SV Campus, Mountain View, CA, August 28, 2014.

28. Invited talk, “Software robustness testing and run-time monitoring of autonomous vehicles,” Ajou-CMU summer program, Pittsburgh PA, July 30, 2014

29. Invited talk, “Security for cloud-connected embedded systems,” Emerson Software Process Improvement Network meeting, Pittsburgh PA, June 11, 2014.

30. Invited talk, “Software Quality, Dependability and Safety in Embedded Systems,” LAAS-CNRS, Toulouse France, June 4, 2014

31. Invited talk, “Software Robustness Testing and Run-Time Monitoring of Autonomous Vehicles,” ONERA lab, Toulouse France, June 3, 2014.

32. Invited talk, “Exception Handling Robustness: lessons learned from the Ballista project,” Sun Yat Sen University, Guangzhou, China, April 10, 2014

33. Invited talk, “Exception Handling Robustness: lessons learned from the Ballista project,” Southeast University, Nanjing, China, April 11, 2014

34. Invited talk, “What Does It Take To Get Good Enough Software? What Everyone Can Learn from Safety Critical Embedded System Design,” PDT Partners, New York City, Jan 31, 2014.

35. Invited talk, “NSF 2014 Transportation CPS Workshop,” Arlington VA, Jan 23-24, 2014.

36. Invited tutorial, “CRCs and ARINC-825,” ARINC-825 committee meeting, Munich Germany, Dec 3, 2013 (teleconference).

37. Invited tutorial, “Data Integrity Techniques,” FAA-hosted two hour tutorial webinar, Oct. 29, 2013

38. Session Chair, Error Control Codes Session (SAFECOMP); Security & Network Session (CARS workshop), SAFECOMP, Sept 2013.

39. Invited talk, “Software Dependability in Embedded Software Industry Projects,” University of Toulouse, France, Sept. 23, 2013.

40. Invited talk, “External Runtime Monitoring for Critical Embedded Systems,” Technical University of Darmstadt, Germany, Sept. 19, 2013.

41. Invited talk, “12.5 Years Teaching Distributed Embedded System Design,” CPS Education Workshop Planning Meeting, Baltimore MD, Oct 24, 2012.

42. “External Runtime Monitoring for Critical Embedded Systems,” IFIP WG 10.4 meeting, Visegrad Hungary, June 30, 2013.

43. Webinar, “Checksum and CRC Data Integrity Techniques for Aviation,” invited presentation to FAA Designated Engineering Representatives, sponsored by FAA, May 9, 2012.

44. Panel session, “The Grand Challenge of Embedded System Dependability,” DSN panel on grand challenges in dependability, Hong Kong, June 29, 2011.

45. Panel session, “Cyberphysical System Safety and ISO 26262,” Workshop on Developing Dependable and Secure Automotive Cyber-Physical Systems from Components, Troy MI, Mar 17-18, 2011.

46. Invited speaker, “Background in Embedded System Security,” Trusted Computing in Embedded Systems Workshop (TCES), Pittsburgh PA, Nov 2010.

47. Keynote speaker, “Risk Areas In Embedded Software Industry Projects,” Workshop on Embedded System Education (WESE), Scottsdale, AZ, Oct 2010.

48. Invited speaker, “Embedded System Security & Survivability,” Darmstadt University, Germany, Sep 2010.

49. Invited speaker, “Embedded (Cyber Physical) Security and Survivability,” Symposium on Automotive/Avionics Systems Engineering, SAASE 2009, San Diego CA, Oct 2009.

50. Invited speaker, “Embedded System Security and Survivability,” Embedded Systems and Communications Security Workshop (ESCS09), Niagara Falls NY, Sept. 2009

51. Invited speaker, “Cyber-Physical System Security and Survivability,” Cyberphysical Systems Summer School, Atlanta GA, June 2009.

52. Seminar, “Created system architectures,” Lutron Corporate training facility, April 2009.

53. Invited keynote talk, “Exception Handling Robustness: lessons learned from the Ballista Project,” 4th Workshop on Exception Handling, Atlanta GA, Nov 2008.

54. Invited talk, "Embedded system survivability,” Georgia Tech, Nov. 2008.

55. Weekly Seminar Series, “Current practice in embedded system safety,” National Robotics Engineering Center (NREC), Pittsburgh PA, September 4, 2007.

56. Weekly Seminar Series, “Toward Improved Embedded System Safety Using Run-Time Invariants,” National Robotics Engineering Center (NREC), Pittsburgh PA, March 28, 2007.

57. Distinguished Speaker Series, “Embedded Security,” University of Central Florida, December 2, 2005.

58. Weekly Department Seminar, “Embedded Security,” Carnegie Mellon ECE Dept., February 10, 2005.

59. Distinguished Speaker Series, “Embedded Security,” Princeton University, November 2, 2005.

60. Invited talk, “Experiences with Component Interference on Shared Hardware Resources,” IFIP WG 10.4 meeting, Moorea, March 7, 2004.

61. Invited talk, distinguished seminar series: "Progress in resilient embedded systems," Xerox Webster Research Center, Nov 4, 2003.

62. Panel session on “Can we trust what we embed,” Workshop on Dependable Embedded Systems (SRDS 2003), Florence Italy, October 2003.

63. Panel session on “Safety Critical Automotive Networks,” Dependable Systems and Networking Conference (DSN 2003), San Francisco, June 2003.

64. Panel session on “Fault Tolerance & Self-Healing Software”, Workshop on Architecting Dependable Systems (ICSE/WADS), Portland Oregon, May 5, 2003.

65. Koopman, P., “A Ballista Retrospective,” invited departmental seminar at Coimbra University, Coimbra Portugal. January 9, 2003.

66. Koopman, P. & Shelton, C., “Scalable Specification of Graceful Degradation in Distributed Embedded Systems,” presentation at IFIP WG 10.4 meeting, Sal, Cabo Verde, January 7, 2003.

67. Koopman, P., “Dependable embedded systems,” National Robotics Engineering Center seminar, August 7, 2002, Pittsburgh PA, (invited presentation).

68. Koopman, P., “In pursuit of dependable software-intensive systems,” DBench meeting, February 4, 2002, LAAS/CNRS, Toulouse France (invited presentation).

69. Koopman, P. & DeVale, J., “Creating robust software interfaces: fast, cheap, good -- now you can get all three,” IFIP WG 10.4 meeting invited talk, January 2002.

70. DeVale, J., Nace, B., DeVale, K. & Koopman, P., three 90-minute tutorials on the topics of embedded systems and software reliability, “SEEK” program educating visiting Korean educators and industrial researchers, Carnegie Mellon University, July 2001.

71. Nace, B. & Koopman, P., “The Internet Meets Embedded Systems,” invited talk, *IBM Conference on Third Wave of Connectivity*, IBM Academy of Technology, April 18, 2001, Yorktown, NY.

72. Koopman, P., “Challenges in Embedded System Research & Education,” North Carolina State University computer systems seminar, February 5, 2001, Raleigh NC (host: Alex Dean/Tom Conte).

73. Koopman, P., “Software Robustness Testing,” University of Texas at Austin computer systems seminar, February 1, 2001, Austin TX (host: Yale Patt).

74. Koopman, P., “Software Robustness Testing,” IBM Austin Research Laboratory, January 31, 2001, Austin Texas (host: Mootaz).

75. Invited panelist, “Embedded Software Licensing?”, *Workshop on Warranty Protection for High-Tech Products and Services*, Federal Trade Commission, October 26, 2000, Washington D.C.

76. Invited panelist, “Embedded Systems Education,” Design Automation Conference (DAC 2000), June 2000.

77. Invited panelist, "Where are the challenges for emerging embedded applications?”, *The Second International Workshop on Compiler and Architecture Support for Embedded Systems (CASES'99)*, Washington D.C., October 1-3, 1999.

78. Koopman, P., “Why Things Break (and what it means to embedded system designers),” Princeton University, July 14, 1999 (host: Wayne Wolf).

79. Koopman, P., “Ballista Software Robustness Testing,” AT&T Shannon Laboratory, Princeton NJ, July 13, 1999 (host: Yennun Huang).

80. Koopman, P., "The Ballista Software Robustness Testing Service," IFIP Working Group 10.4 on Fault Tolerance, Lake Geneva WI, June 21, 1999.

81. Koopman, P., "Ballista: Software Robustness Testing & Hardening," DARPA ITO PI meeting, Wailea HI, March 16, 1999

82. Koopman, P., "Software Robustness Testing," Microsoft, Redmond, WA, March 3, 1999 (host: Yi Min Wang).

83. Koopman, P., "Software Robustness Testing," NSF Workshop on security, fault tolerance and software assurance (invitation only), Williamsburg, VA, November 11-13, 1998.

84. Koopman, P., "Software Robustness Testing," Bell Laboratories, November 5, 1998 (host: Yennun Huang).

85. Siewiorek, D. & Koopman, P, "Improving Reliability of Commercial Off-the-Shelf Software Using Robustness Benchmarks," Keynote Address (invited talk), 1997 Pacific Rim International Symposium on Fault-Tolerant Systems, National Taiwan University, Taipei, Taiwan, December 16, 1997. (Talk delivered by Siewiorek based on Koopman's work.)

86. Koopman, P., "Ballista: COTS Software Robustness Testing", University of Pittsburgh, Pittsburgh, PA, October 13, 1997 (host: Rami Melham).

87. Maxion, R. & Koopman, P., "Off-the-Shelf Software Robustness Evaluation," DARPA ITO PI Meeting, March 1997.

88. Koopman, P., "COTS Software Robustness Evaluation and Hardening," DARPA ITO PI meeting, San Diego, June 1996.

89. Siewiorek, D. & Koopman, P., "Wearable Computer Workshop," Digital Cambridge Research Lab, April 1996.

90. Gupta, R., Koopman, P., & Wolfe, A., "Tutorial: CAD for Digital Embedded Systems," *Design Automation Conference*, San Francisco, June 1995. Full day tutorial session.

91. "Engineering Design Methodologies," Siemens Corporate Engineering, Princeton, NJ, July 13, 1994.

92. Koopman, P., "Hidden Horrors of Cached CPUs," *Proceedings of Embedded Systems Conference 1993*, Santa Clara, pp. 72-80, October 1993; repeated in *Proceedings of Embedded Systems Conference East 1994*, Boston, April 1994. One-hour tutorial.

93. "Embedded Computing at United Technologies," Carnegie Mellon University, March 15, 1993.

94. "Embedded Computing at United Technologies," Princeton University, March 2, 1993.

95. "Simulation of Multiprocessor DSP algorithm performance," Carnegie Mellon University, March 5, 1992.

5.B. Government Committees, Civic Appointments, Board Memberships

1. Industrial Advisory Board, DBench Consortium (European project on dependability benchmarking). 2001-2004.

2. Board of advisors, Gravitate Corp. http://www.grvt8.com/ 2000-2001.

3. Workshop participant and co-author: Lightner, M. (ed.), *National Science Foundation Workshop on CAD Needs for System Design*, Final Report, http://dufay.colorado.edu/~ lightner/system-workshop/system-workshop-final.html, April 3-4 1995.

4. Supported Pittsburgh regional alliance efforts to convince large companies to create a Pittsburgh regional design center. All-day meetings 6/4/98, 6/10/98 plus preparation time.

5.C. Membership and Activities in Honorary Fraternities, Professional Societies

1. Member, IFIP WG 10.4 on Dependability

2. Senior Member, Institute of Electrical and Electronic Engineers (IEEE)

3. Senior Member, Association for Computing Machinery (ACM)

4. Member, SAE International (Society of Automotive Engineers)

5. Member, Eta Kappa Nu

6. Member, Tau Beta Pi

5.D. Editorial Roles on Publications, Major Activities in Professional Meetings

1. Program Chair, DCCS track, Dependable Systems and Networks (DSN) conference, June 2012, Boston. (156 submissions; 17.3% accept rate; 39 PC members with 2-day physical selection meeting.)

2. General Chair, *Dependable Systems and Networks (DSN)* conference, June 2008, Anchorage Alaska. (International conference; approximately 250 attendees. Generated approximately 10% surplus on $190K of revenue. $35K donations from industry.)

3. Associate Editor, Design Automation of Embedded Systems: an international journal (1996-2004)

4. Program committee, *Dependable Systems and Networks Conference (DSN/FTCS)*, (2000, 2002, 2003, 2005-2008, 2010-2016; 2018. Corresponding member (Western Hemisphere publicity mailings) 2005-2010; Workshop selection committee 2006-2007; Industrial Track program committee 2016, 2018.

5. Program Committee, SAFECOMP 2013-2016, 2018.

6. Co-Organizer, Workshop on Autonomous System Safety, 3rd Scandinavian Conference on System and Software Safety, 2015.

7. Student Forum chair, *Dependable Systems and Networks (DSN)* conference, 2005. Student forum committee, 2010.

8. Program Committee, International workshop on Dependable and Secure Machine Learning (DSML), 2018.

9. Program Committee & Steering Committee, Workshop on AI Safety Engineering (Safecomp), 2018.

10. Member SAE Ground Vehicle Reliability Committee, emphasis on software reliability, 2012…2014

11. Program Committee, Workshop on Open Resilient Human-Aware Cyber-Physical Systems (WORCS-2013).

12. Program Committee, EDCC 2015.

13. Program Committee, ICDCS 2015.

14. Program Committee, SRDS, 2014, 2015, 2016.

15. Program Committee, SSIV 2015, 2016.

16. Program Committee, CRE 2016.

17. Program Committee, SMARTCOMP 2014.

18. Program Committee, HotDep, 2012

19. Program Committee, CPS-Ed 2013.

20. Program Committee, DCNET 2013.

21. Program Committee, Workshop on Critical Automotive applications: Robustness and Safety (CARS), 2010, 2013, 2015, 2016.

22. Program Committee, Cyber Resilience Economics Workshop, 2016.

23. Program Committee, Third Workshop on Software-Based Methods for Robust Embedded Systems (SOBRES '15)

24. Program Committee, IEEE International Workshop on Software Certification (WoSoCer), 2015.

25. Program Committee, 5th International Workshop on Exception Handling (WEH), 2012.

26. Program Committee, 2nd International Workshop on Software Health Management (SHM), 2011

27. Program Committee, Languages, Compilers, and Tools for Embedded Systems (LCTES), 2011

28. Program Committee, *Embedded Systems and Communications Security Workshop (ESCS)*, 2009.

29. Program Committee, *IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems (FeT)*, 2007, 2009.

30. Program Committee, *EMSOFT Workshop on Embedded Systems Security* (WESS07), 2007

31. Program Committee, *EMSOFT*, 2005.

32. Program Committee, *4th International Workshop on Real Time Networks*, 2005.

33. Program Committee, *Workshop on Dependable Embedded Systems*, 2005.

34. Program Committee, *Pacific Rim Dependability Conference* (2004).

35. Workshop Co-Chair, *Workshop on Distributed Embedded Systems*, 2003-2004.

36. Program Committee, *Workshop on Software Architectures for Dependable Systems (WADS)*, 2003-2006; 2009.

37. Guest Editor, *IEEE Micro*, themed issue on embedded real-time automotive networks; July/August 2002 issue.

38. Workshop Chair, *Workshop on Dependability Benchmarking*, DSN 2002, Washington DC. June 2002.

39. Founding Chair, IFIP 10.4 WG Special Interest Group on Dependability Benchmarking, (1999-2001). Meeting chair: November 1, 1999, Boca Raton, FL; April 3-4 2000 Menlo Park, CA; November 9-10 2000 Pittsburgh PA; May 10-11 2001 Poughkeepsie NY.

40. Program committee, *Embedded World*, 2013.

41. Program committee, *Workshop on Embedded Software (EMSOFT)*, 2005.

42. Program committee, *Workshop on Evaluating and Architecting System dependabilitY (EASY)*, concurrent with *ASPLOS*, 2002.

43. Program committee, *HW/SW Codesign Workshop*, (1994-1998)

44. Program committee, International Conference on Computer Design, (1997-1998)

45. Program committee, Design Automation and Testing, Europe (1998)

46. Language expert (Forth) for History of Programming Languages Conference II, 1993

47. Program chair, 1991 ACM SIGForth conference.

5.E. Awards, Prizes, Honors

1. 2018 IEEE-SSIT Carl Barus Award for Outstanding Service in the Public Interest.

2. Eta Kappa Nu Excellence in Teaching Award, CMU ECE Department, May 2013.

3. SAE Excellence in Oral Presentation Award for "Challenges in Autonomous Vehicle Testing and Validation," SAE World Congress, April 2016; awarded July 2016.

4. Best Paper Award, DSN 2002. DeVale, J. & Koopman, P., "Robust software – no more excuses," *Fault Tolerant Computing Symposium (FTCS-31)/Dependable Systems and Networks*, Washington DC, July 2002. Awarded the prestigious William C. Carter *outstanding paper award*.

5. George Tallman Ladd Research Award, Carnegie Mellon University, 1999.
   "The G. T. Ladd award is made to a regular faculty member within the Carnegie Institute of Technology in recognition of outstanding research and professional accomplishments and potential. The award is restricted to Assistant Professors in CIT in the year of reappointment or the year after."

6. Best Paper award, Autotestcon 99. Pan, J., Koopman, P. & Siewiorek, D., "A Dimensionality Approach To Testing and Improving Software Robustness," *Autotestcon 99,* San Antonio, Texas, August 30 – September 2, 1999.

7. Listed in *Who's Who in Science and Engineering*, Marquis Who's Who, 1999-2002

8. United Technologies Research Center Technical Achievement Award (top annual technical award within Research Center), 1993.

9. NASA Ph.D. Fellowship, 1987-1989.

10. Navy Achievement Medal, U.S. Navy, for professional junior officer achievement, 1984.

11. Naval Expeditionary Medal, U.S. Navy Submarine Force Pacific Fleet, 1983. This award confers combat veteran status.

5.F. Service on CMU Committees

1. ECE graduate admissions committee, 2017-2019.

2. ECE undergraduate advising committee, 2006-2009; 2012-2016

3. ECE Undergraduate Studies Committee (including ABET accreditation) 1997-1998; 1999-2001; 2004-2005; 2007-2008; 2009-2011

4. ECE Graduate Education Committee 1998/99, 2005-2006

5. ECE Undergraduate curriculum Committee 2005-2006

6. Chair, ECE Facilities Committee 2002-2003. Member, ECE Facilities committee, 2011.

7. ECE Ad hoc committee on graduate study process improvement, 2014.

8. ECE Head Search Committee 1999

9. ECE Ad Hoc faculty hiring Committee 1998/99

10. ECE Graduate Student Office reorganization ad hoc committee 1998.

5.G. Other

1. Numerous interviews and quotes in press articles regarding autonomous vehicle safety, 2017-….

2. Video tutorial series on embedded systems distributed via YouTube: http://ece.cmu.edu/~koopman/lectures/

3. Various blog entries on embedded software: http://betterembsw.blogspot.com/

4. Interview on Smarter Cars Podcast, Mar. 15, 2018, https://itunes.apple.com/us/podcast/smarter-cars/id1223757516?mt=2

5. Interview on Embedded.fm Podcast, Episode #183, Jan. 11, 2017, http://embedded.fm/episodes/183

6. Technical advisor for Carnegie Mellon Racing (Formula SAE Race Car), 2014-2015.

7. External Visiting Committee, Digital Safety and Security Department, Austria Institute of Technology, April 2016.

8. External MS Examiner, Mr. Stephen Grixti, University of Malta (contact: Dr. Ing. Nicholas Sammut), 2014-2015. "A Study of the Dependability of an IMA-SP Testbed Focusing on Hypervisor Robustness."

9. Invited participant at NSF/NRC workshop: 21st Century Cyber-Physical Systems Education: Developing Solutions, , Washington DC, Oct 2-3, 2014

10. Participant in Dagstuhl #13511, Software Engineering for Self-Adaptive Systems: Assurances, Wadern Germany, Dec. 15-19, 2013.

11. Participant in Dagstuhl #1141, Science and Engineering of Cyber-Physical Systems, Wadern Germany, Nov. 1-4, 2011.

12. Invitation-only IFIP WG 10.4 meetings on Dependability: June 1999, January 2000, June 2000, January 2001, semi-annually for majority of meetings held since January 2002. Program co-chair March 2001. General Chair June 2008. Elected Member of IFIP WG 10.4 in July 2002. Regular presentations at work-in-progress sessions and panel discussions.

13. Blog: http:// betterembsw.blogspot.com discusses embedded software design, practical issues, and research. 9,914 visits as of August 22, 2011 (source: clustrmaps.com)

14. Created course syllabus and gave majority of lectures for 4-week embedded systems summer course for 25 engineers from Hyundai Motor Company, summer 2004.

15. Volunteer sysadmin for Pittsburgh Urban Christian School (PUCS) www.pucs.org (2000-2006)

16. Guest Speaker, Women in ECE graduation dinner, 4/24/02.

17. CMU-sponsored Boy Scout presentation, "Careers in computer engineering" as part of their Merit Badge University program, 03/11/00.

18. IEEE student chapter meeting, CMU ECE, *Testing Software Robustness* research presentation, 2/28/00.

11. Pittsburgh Regional High School Science Fair Judge, Duquesne University, 1998, 1999.

12. Invitation-only NSF workshop on security, fault tolerance, and software assurance, Williamsburg, VA, November 11-13, 1998.

13. Guest lecture at Colfax Elementary School, Pittsburgh PA on Galileo's gravity experiment, June 6, 1997.

14. Monthly visits to Fourth Grade classrooms as part of the "Science Improvement Project" in Hartford, CT, area schools, Spring 1992.

15. Additional peer review services for:

- National Research Council
- National Science Foundation
- Proceedings of the IEEE
- IEEE Transactions on Computers
- IEEE Transactions on Dependable and Secure Computing
- IEEE Transactions on Software Engineering
- IEEE Transactions on Parallel & Distributed Systems
- IEEE Transactions on Software Engineering
- IEEE Transactions on Industrial Electronics
- IEEE Transactions on Intelligent Vehicles
- IEEE Transactions on Communications
- IEEE Security & Privacy
- IEEE Communications Letters
- IEEE Computer (Senior Reviewer, 1996)
- IEEE Design & Test
- IEEE Micro
- IEEE Embedded Systems Letters
- ACM Computing Surveys
- IEE Proceedings: Software
- Fault Tolerant Computing Symposium/Dependable Systems and Networking conference (FTCS/DSN)
- Design Automation for Embedded Systems (journal)
- Integrated Computer-Aided Engineering (journal)
- Journal of Theoretical Computer Science
- Journal of Software and Systems
- International Journal on Performability Engineering
- Journal of Circuits, Systems, and Computers
- SAFECOMP and associated workshops
- EMSOFT and associated workshops
- SMARTCOMP
- European Dependable Computing Conference (EDCC)
- Pacific Rim Dependability Conference
- Fieldbus Technology Conference (FeT)
- Evaluating and Architecting System dependabilitY (EASY)
- International Symposium on Computer Architecture (ISCA)
- Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)
- International Symposium on Architecting Critical Systems (ISARCS)
- Design Automation Conference (DAC)
- Hot Topics on Dependability (HotDep)

- Workshop on Architecting Dependable Systems (WADS)
- Embedded Systems and Communications Security workshop (ESCS)
- Languages, Compilers, and Tools for Embedded Systems (LCTES)
- EuroForth conference
- Microelectronic Systems Education Conference
- IEEE Intelligent Transportation Systems Conference
- International Symposium on Wearable Computers
- Workshop on Exception Handling
- Cyber-Physical Systems Education Workshop (CPS-Ed)
- Workshop on Open Resilient Human-Aware Cyber-Physical Systems (WORCS)
- SAE Conference
- American Control Conference
- Innovations in Systems and Software Engineering
- Hawaii Conference on System Sciences

5.H. Significant Expert Witness Engagements

1. Plaintiff testimony on software safety for Toyota Unintended Acceleration Economic Loss Class Action (~$1.6 billion) involving Model Year 2002-2010 vehicles:
   In Re: Toyota Motor Corp., Unintended Acceleration, Marketing, Sales Practices, and Products Liability Litigation, US District Court Central District of California, Case No. 8:10ML2151 JVS (FMOx). Deposition on October 9, 2012; settled before trial.

2. Plaintiff testimony on software safety for multiple Toyota Unintended Acceleration liability cases (death & injury) in multiple state courts & federal courts (>500 cases seeking millions of dollars per case), including the only public trial:
   Bookout et al. v. Toyota Motor Corp. et al., District Court of Oklahoma County, Oklahoma CJ-2008-7969. Trial testimony Oct. 11, 2013, Oklahoma City.

3. Defendant testimony on non-infringement in largest dollar-amount patent suit in the history of Ford Motor Company (approx.. $200M alleged damages):
   Eagle Harbor and Mediustech v. Ford Motor Company, US District Court for Western Dist. Washington, Tacoma, Civil Action 3:11-cv-05503-BHS. Trial Mar. 19, 2015.

4. Plaintiff testimony on software safety for the Ford Unintended Acceleration case and related death and injury case(s) involving Model Year 2004-2010 Ford vehicles:
   Johnson et al v. Ford motor Company, US District Court for Southern Dist. West Virginia, Civil Action 3:13-cv-06529.

6. Expert Services

6.A. Testimony in other cases, previous 4 years (client in italics)

- Spilman, Thomas & Battle, 2017-… (*Johnson* v. Ford), US District Court Southern District West Virginia, Huntington, Civil Action 3:13-cv-06529 (Unintended Acceleration Class Action). Depositions Jul. 28 and Aug. 18, 2017.
- Thomas Murray, 2016-2017 (*Morrissey* v. Subaru), US District Court Middle District Florida Tampa, Civil Action 8:16-cv-00048-JDW-AAS. Deposition Nov. 10-11, 2016.
- Fried Frank, 2015-2016 (Signal v. *Mercedes Benz USA*), US District Court Central District California, Civil Action 2:14-03109-JAK (JEMx). Markman Hearing. Deposition on Jan. 7, 2016. Claims invalidated in Summary Judgment.
- Wilmer Hale, 2011-2015 (Eagle Harbor and Mediustech v. *Ford Motor Company*), US District Court for Western Dist. Washington, Tacoma, Civil Action 3:11-cv-05503-BHS. Depositions on Dec. 5 & Dec. 9 2014. **Trial Mar. 19, 2015.**
- Pepper Hamilton LLP, (Samsung Electronics v. *Rembrandt Wireless Technologies*), PTAB, IPR2014-00518, IPR2014-00519, Patent 8,023,580, Deposition on Jan. 13, 2015.
- Stroock & Stroock & Lavan, 2010-2014 (*Inventio* v. Thysskenkrupp), 2010-2014 US District Court – District of Delaware, 08-CV-0874-RGA. Deposition on Jul. 30, 2013. **Trial Feb. 25-28, 2014.**

6.B. Clients of expert opinions, other than testimony, previous 4 years (client in italics)

- Sweeney & Dietzler, 2017 (Barich v. *Treacy*), automotive safety.
- Poole Shaffery, 2017-… (confidential defendant), automotive safety.
- Thomas Murray, 2017-… (confidental plaintiff), automotive safety.
- Rothwell Figg, 2016-2017 (confidential defendant), embedded networking.
- K&L Gates 2016 (4Moms). Software safety.
- Wilmer Hale, 2016-2017 (confidential defendant), computer storage.
- Quinn Emanuel Urquhart & Sullivan, 2016-… (confidential defendant), automotive middleware.
- Wilmer Hale, 2015-2016 (confidential defendant), medical device software.
- Heim, Payne & Chorush LLP, 2014-2015 (confidential plaintiff), computer networking.
- Kreindler & Kriendler/Nelson & Fraenkel, 2013-2015 (*Sachs* v. Toyota)
- Toyota UA cases : Robinson Calcagnie Robinson Shapiro Davis, 2012-2013 (*Class and Van Alfen* v. Toyota) ; Lieff Cabraser Heimann & Bernstein, 2013 (*St. John* v. Toyota) ; Bailey & Glasser, 2013 (*Alberto* v. Toyota) ; Beasley Allen, 2013 (*Bookout* v. Toyota) ; Heiskell, 2013 (*Vance* v. Toyota); 2017 McCune Wright Arevallo (*Hadi* v. Toyota); numerous other cases with confidential status, including testimony at confidential events.
- Niro, Haller & Niro, …–2016. Confidential patent matters.

6.C. Other recent collaborative relationships (see also technical consulting & research funding):

General Motors
US Army/Office Secretary of Defense
U.S. Dept. of Transportation
TnT IP, LLC
Federal Aviation Administration
US Navy
Clients of Edge Case Research LLC

EXHIBIT B

by JOHN SCHILL

An Overview of the CAN Protocol

The CAN protocol offers a comprehensive standard for network communications. It supports numerous automotive and industrial control applications. This article lays out the major elements of the CAN protocol and describes two application layer definitions.

The Controller Area Network (CAN) protocol, developed by Robert Bosch GmbH, offers a comprehensive solution for managing communication between controllers, sensors, actuators, and human-machine interfaces. The CAN protocol specifies versatile message formats that can be mapped to specific control information categories. Industrial applications such as plant floor automation successfully implement CAN-based networks using open standards developed by Allen-Bradley (DeviceNet) and Honeywell (Smart Distributed System—SDS). CAN chips help link devices within the system, enabling them to work harder, smarter, and faster than before. In this article, I present an overview of the CAN protocol and descriptions of DeviceNet and SDS.

Nance Paternoster

The CAN protocol is an open standard, and over eight companies are currently licensed to design and manufacture CAN chips. The broad assortment of CAN chips and wide-spread usage in automotive and industrial applications are good indicators of CAN's long term viability.

BENEFITS FROM CAN

The CAN protocol provides five primary benefits. First, as a standard communications protocol, CAN simplifies and economizes the task of interfacing subsystems from various vendors onto a common network. Second, the CAN protocol supports over five million message identifiers and provides the flexibility to implement sophisticated messaging schemes. Error detection and response are handled in hardware by the CAN chips themselves, which minimizes error recovery software. Third, the communications burden is shifted from the host CPU to an intelligent peripheral; the host CPU then has more time to run its system tasks. Fourth, as a multiplexed network, CAN reduces wire harness size by eliminating much of the point-to-point wiring.

Last, as a standard protocol, CAN

CAN has a broad market appeal, which motivates semiconductor makers to develop competitively-priced CAN chips.

has a broad market appeal, which motivates semiconductor makers to develop competitively-priced CAN chips. For example, future high-volume applications may expect CAN node costs in the two- to four-dollar range. This includes the CAN protocol chip and its bus driver.

The CAN protocol is implemented with a standalone CAN controller chip interfaced to a host CPU or with a CAN peripheral integrated with the host CPU, as shown in Figure 2. A CAN bus driver or transceiver supplies the current to drive the bus and converts CAN bus signals to CMOS levels for the CAN peripheral.

CAN PROTOCOL OVERVIEW

The CAN protocol is licensed to silicon manufacturers by Robert Bosch GmbH to design and manufacture CAN chips. CAN silicon implementation must satisfy the CAN protocol so that any CAN chip will communicate with another CAN chip, regardless of manufacturer or vintage. The CAN protocol defines the format of messages transmitted on the bus, the timings of transmitted bits, arbitration priority, and error detection. CAN networks may operate at up to 1Mbit/s, allowing a device to request and then receive information back from another device in less than 0.2ms.

Figure 1 shows an industrial network application. In material handling applications, a photoelectric sensor can recognize an overload condition on a conveyer. The sensor will then signal the host controller to re-route packages along another conveyer, which is controlled by a set of actuators.

The message format consists of a start bit, an 11- or 29-bit message identifier, a remote message bit, data length code, data bytes, and error detection code.

The CAN message formats are shown in Figure 3. With an 11-bit message identifier, called a standard format 2032, distinct messages may be defined in the system. Likewise, a 29-bit message identifier, or extended format, permits over 500 million distance message identifiers, providing significant flexibility to partition messages by system function. A CAN node typically supports multiple CAN message identifiers because it transmits and receives a number of system messages containing various parameter data. One may associate a message identifier with a system parameter and not necessarily with a particular system node or module address, as with conveyer speed and motor temperature.

The timings of transmitted bits are specified with respect to a bit sampling procedure. This procedure ensures that all nodes transmit and receive messages by synchronizing themselves with the CAN bus.

The arbitration priority is determined by the priority of the transmitted messages in which the lowest-numbered message identifier has the highest priority. The CAN protocol requires each node to continuously monitor the CAN bus transmissions at all times. If two CAN nodes begin transmitting at the same time, one node will detect that the message identifier transmitted on the CAN bus doesn't match the message identifier it's attempting to transmit. When this occurs, the node with the unmatched message identifier discontinues its message transfer until the bus is once

again free. This process is called *message arbitration*.

The CAN protocol also defines an error detection algorithm that must be executed against all transmitted messages by each CAN bus node. If an error occurs, automatic retransmission takes place. Additionally, if a node finds repetitive errors, it will take itself off the bus and notify its host CPU.

COMMUNICATION MESSAGE OBJECTS

The CAN protocol supports two types of communications message objects: *transmit* and *receive*. Communication message objects correspond to specific "mailboxes" or "communication slots" available within the CAN implementation. Currently available CAN chips implement no less than two message objects (one transmit, one receive) and up to 15 message objects, which may be independently configured as transmit or receive.

A message object consists of several dedicated bytes including control, the message identifier, message configuration, and data. The control bytes manage message operations and contain bits such as message valid, transmit request, message lost status, and interrupt pending status.

Message identifiers usually correspond to a particular control message. In an industrial application, there may be different message identifiers for parameters such as motor speed, device temperature, and device load. The data associated with each message object may be one to eight bytes. The message configuration byte specifies the number of data bytes, transmit or receive function, and possibly whether the message identifier is standard or extended (11 or 29 bits).

Network nodes use receive message objects to receive information from other nodes. The CAN protocol permits a receive message object to request data by sending a *remote message*. A motor may require position information from a gearbox. Remote messages allow the motor to request information from the gearbox without waiting for the gearbox's CPU to initiate data transmission. Enabled transmit message objects will respond to remote messages automatically by sending data (without host CPU intervention).

CAN PROTOCOL CHIP FEATURES

Silicon implementations of the CAN protocol are typically viewed as smart RAM chips. The RAM locations are reserved for communication objects used to manage and for control registers. The host CPU reads and writes to the CAN chip, where read operations usually interrogate receive messages and write operations load data to be transmitted. Read and write operations also manage control registers such as the interrupt pointer, CAN bus transmission rate

configuration, and error status. The CAN chip continues to monitor and transmit messages on the CAN bus, even when the host CPU is addressing the CAN chip.

Interrupts may be enabled when messages are received or transmitted, or when errors are detected. Status bits indicate message and error status.

Some CAN implementations support acceptance mask filtering. This feature allows certain communication objects to receive more than one message by declaring certain message identifier bits to be "don't care," meaning either a 1 or a 0 will suffice for the corresponding bit. For each don't care bit, the number of message identifiers accepted by a receive message object increases by a factor of two. With this feature, a CAN chip may receive more than one message per communication message object.

FIGURE 1
Industrial network application.

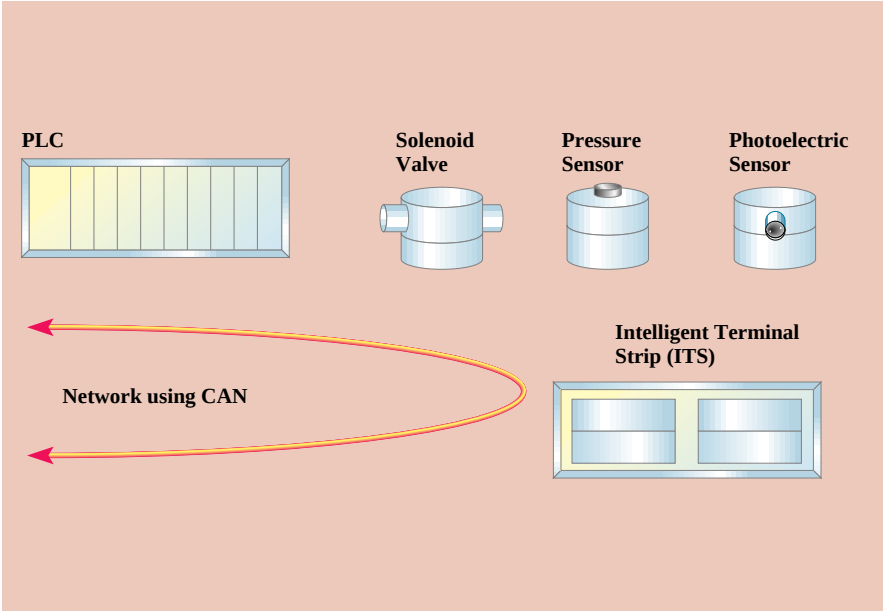
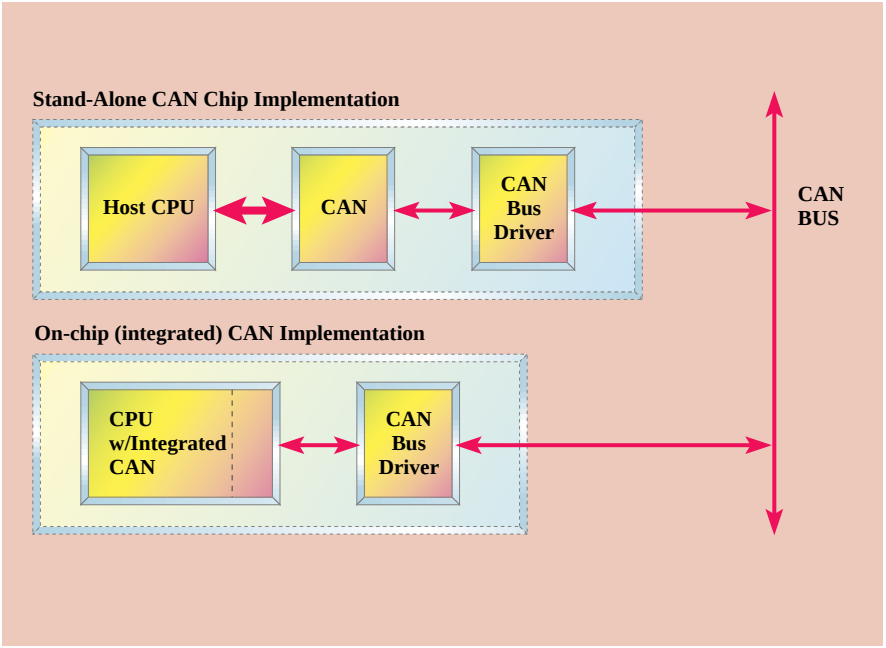


FIGURE 2
CAN bus implementation (integrated/stand-alone).



COMMUNICATIONS INTEGRITY

To increase data integrity, the CAN protocol specifies that each node should continually check bus transmissions for errors, using a cyclical redundancy test. Each CAN chip is capable of calculating a polynomial which is compared with 15 cyclic redundancy check (CRC) bits to detect burst errors and up to five randomly distributed errors within a message. When an error is detected, a stream of bits called an error frame is transmitted to initiate a synchronization of all CAN bus nodes.

STANDARD AND EXTENDED MESSAGE FORMATS

Both standard and extended CAN message formats support four frame types:

- Data: carries data
- Remote: sent when a node requests data from another node
- Error: sent when a node detects a message error
- Overload: sent when a node requires extra delay

The data and remote frame types are initiated by the application, meaning that CAN nodes transmit these frame types to exchange necessary information. The error and overload frames are initiated by the CAN hardware, independent of the application, to recover from an error condition or to delay message transmissions. Most CAN chips are capable of handling the maximum CAN bus speeds of 1Mbit/s, and therefore do not transmit overload frames.

The following describes the standard and extended message formats for the data frames shown in Figure 3.

- SOF: start of frame (dominant bit) marks the beginning of a data/remote frame
- Arbitration: one or two fields that contain the message identifier bits. The standard format consists of one 11-bit field and the extended format has two fields, 11- and 18-bits wide
- RTR: remote transmission request bit is dominant for data frames and is recessive for remote frames. This bit is in the arbitration field
- SRR: substitute remote request bit is used in extended messages and is recessive. This bit is a substitute for the RTR bit in the standard format and is in the arbitration field of the extended format
- IDE: identifier extension bit is dominant for standard format and recessive for extended format. This bit is in the arbitration field of the extended format and in the control field of the standard format
- Control Field: reserved bits r0 and r1 are sent as dominant bits. The 4-bit data length code (DLC) indicates the number of bytes in the data field
- Data Field: the data bytes are located in the data frame (zero to eight bytes). A remote frame contains zero data bytes
- CRC Field: this field is composed of a 15-bit cyclic redundancy code error code and a recessive CRC delimiter bit
- ACK Field: acknowledge is a dominant bit sent by nodes receiving the data/remote frame and is followed by a recessive ACK delimiter bit
- End of Frame: seven recessive bits end the frame
- INT: intermission is the three recessive bits that separate data and remote frames

In addition to these pre-defined bits, "stuff" bits may be added to the message. Stuff bits assist synchronization by adding transitions to the message. A stuff bit is inserted in the bit stream after five consecutive equal-value bits are transmitted; the stuff bit is the opposite polarity of the five consecutive bits. Because CAN chips synchronize on high to low transitions on the CAN bus, stuff bits ensure a sufficient number of synchronization events occur regardless of the message contents. All message fields are stuffed except the CRC delimiter, the ACK field, and the end of frame.

INDUSTRIAL APPLICATIONS: DEVICENET AND SDS

Allen-Bradley's DeviceNet and Honeywell's SDS are open communication standards particularly suited for industrial and factory floor applications. This technology provides solutions to users and device

FIGURE 3
CAN message format.

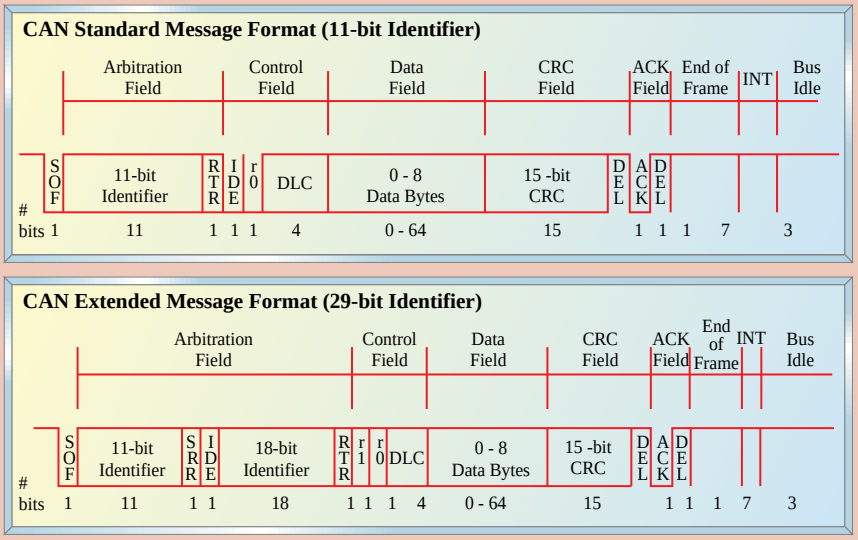
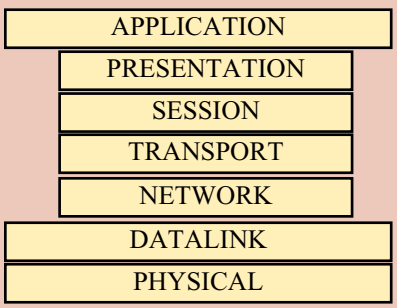


FIGURE 4
Seven-layer OSI communications model.



- The application layer is the message field usage and device hierarchy defined by DeviceNet and SDS.
- The CAN protocol specifies the datalink layer by defining message format and transmission rules.
- The physical layer defines the electrical medium to support the network, such as tranceivers and bus wire.

DeviceNet and SDS application layers are consistent with CAN Specification 1.2; however, CAN Specification 2.0 may also be used, resulting in a slightly reduced message.

manufacturers who interconnect sensors, actuators, PLCs, and controllers for real-time control applications, as in Figure 1. Users gain the flexibility to interconnect devices from multiple suppliers, knowing the network is proven and well-supported. Device manufacturers benefit from reduced development cost and quicker time to market. CAN is the communications technology chosen to support both DeviceNet and SDS because of its robust arbitration and advanced error management capabilities.

SYSTEM-WIDE NETWORK

Referencing the OSI seven-layer communications model, as depicted in Figure 4, DeviceNet and SDS networks utilize the application, datalink, and physical layers. The implementation of these three layers offers optimized dialogue between network devices with minimal overhead. Although the presentation, session, transport, and network layers support unique services to the network, the additional complexity and required device intelligence is not justifiable for the targeted applications.

APPLICATION LAYER: DEVICENET

DeviceNet networks are capable of a four-level hierarchy known as Message Groups 1 to 4. This hierarchy allows the proper device priority on the network so that computers, PLCs, and actuators and sensors may coexist on the network. Peer-to-peer and master/slave configurations are supported. Figure 5 shows the DeviceNet CAN identifier field usage to create the four Message Groups.

Group 1 messages have the highest priority on the network and are available for peer-to-peer communication. The Source Media Access Control (MAC) ID field identifies up to 64 source nodes. A particular transmitting node may implement up to 16 different messages using the 4-bit Group 1 Message ID field. The intent of the Group 1 Message scheme is to provide each transmitting node access to high-priority messages.

Message Group 2 has lower network priority than Message Group 1 and is used for devices that are more control oriented. The MAC ID used for Group 2 messages may indicate either source or destination node IDs, as specified by the end point. Within the range of Group 2 messages, the value of the MAC ID determines bus priority; therefore, particular nodes (transmitting or receiving) are assigned priority. The outcome of the Group 2 message schemes is to rank device priority by MAC ID.

Message Group 2 also serves two other important network features. Some Group 2 messages are reserved for predefined master/slave connection

FIGURE 5
CAN identifiers field usage.

IDENTIFIER BITS											HEX RANGE	IDENTITY USAGE
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MAC ID						000-3FF	Message Group 1
1	0	MAC ID						Group 2 Message ID			400-5FF	Message Group 2
1	1	Group 3 Message ID			Source MAC ID						600-7BF	Message Group 3
1	1	1	1	1	Group 4 Message ID (0-2FH)						7C0-7EF	Message Group 4
1	1	1	1	1	1	1	X	X	X	X	7F0-7FF	Invalid CAN Identifiers
10	9	8	7	6	5	4	3	2	1	0		

CAN supports both DeviceNet and SDS with its robust arbitration and advanced error management capabilities.

FIGURE 6
SDS application protocol data unit (APDU), long form, in a CAN message.

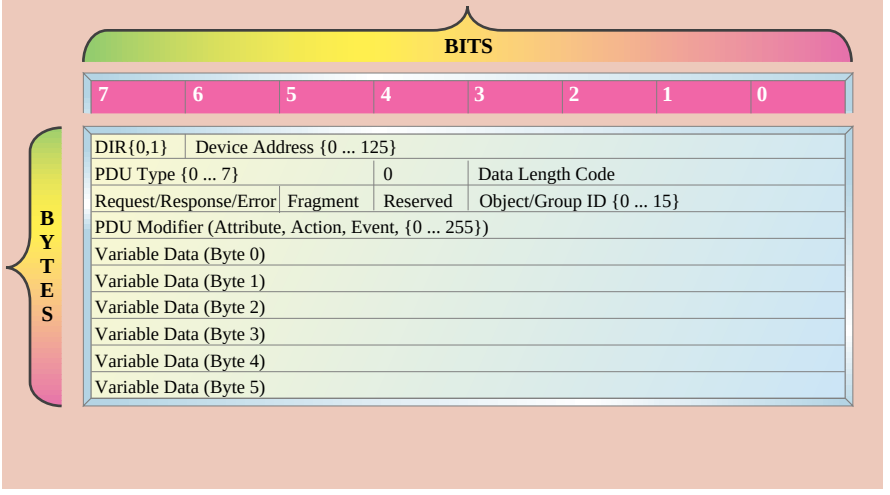
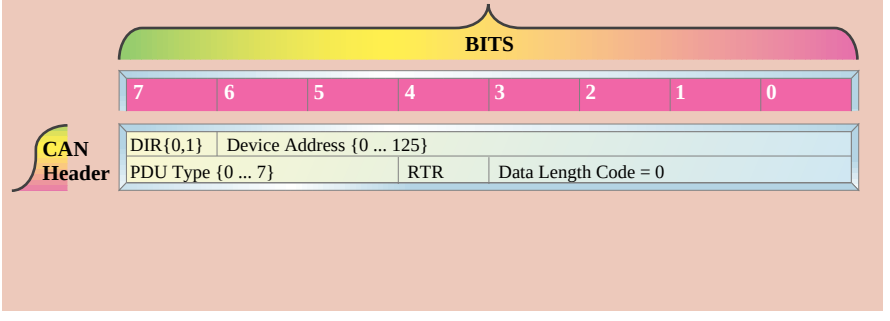


FIGURE 7
SDS APDU, short form, in a CAN message.



management and other messages are reserved to resolve duplicate MAC ID assignments, which are prohibited by the CAN protocol.

The Group 3 message scheme is similar to that of Group 1, except each node is allowed eight messages of lower priority.

Group 4 messages have lower priority than Groups 1 to 3 and are currently reserved for future use.

DeviceNet specifies the application layer information within the 11-bit CAN message identifier. This placement leaves all eight data bytes of the CAN message available for information communication.

APPLICATION LAYER: SDS

The SDS application layer supports four generic classes of services, called *read*, *write*, *action*, and *event*. *Read* and *write* are used for configuration and interrogation of attributes of a device. Attributes of a node are parameters such as device name, date code, device address, and vendor name. Action is a command to a device to perform a specified function. *Event* is a notification by the device of an event that was sensed or generated by the device.

SDS may be operated in either peer-to-peer or master/slave environments. In peer-to-peer mode, devices such as sensors may transmit data of an *event*

as a broadcast message without an explicit acknowledgment from other devices that the message was received. In master/slave mode, the initiating master device receives acknowledgment by the slave device responding to the requested *action*.

The device address supports up to 126 devices to be present on the network. The DIR field indicates whether the device address is a source or destination address. The PDU type field indicates the service class: *read*, *write*, *action*, or *event*.

The APDU long form, shown in Figure 6, utilizes two CAN message data bytes to create additional fields. The first CAN data byte contains a field to specify particular message services such as request, response, or error response. A transport layer type feature is supported by the fragment and object/group ID fields, whereby a fragment = 1 indicates that the message is one of 16 possible objects of a larger total message. The second CAN data byte stores the PDU modifier that contains *attribute*, *action*, and *event* information. The remaining six CAN data bytes are available to transfer data.

SDS also specifies a short message type, shown in Figure 7, which is used to quickly communicate simple digital input or output such as device on/off status.

PHYSICAL LAYER

DeviceNet and SDS have similar physical layer requirements. DeviceNet and SDS implement a five-wire cable with a shielded twisted pair to carry the two CAN bus signals, another shielded twisted pair for power, and a drain wire attached to the shielding. Industry standard tee connectors allow quick installation and disconnect capability. Up to 64 nodes may be attached to a single bus. Transmission speed is dependent on overall length (for example, 500Kbps @ 100m, 250Kbps @ 200m, 125Kbps @ 300m). Future SDS physical layer specifications are planned to support 128 nodes.

Both DeviceNet and SDS adopted practices consistent with ISO/DIS 11898, a working discussion committee establishing physical layer standards for CAN communications.

CAN OUTLOOK

The CAN protocol provides users with a clear and comprehensive standard for network communications. CAN is supporting a growing number of applications in automotive and industrial control applications. With the number of CAN products increasing, CAN will continue to be a versatile and cost-effective networking solution.

John Schill is a technical marketing engineer for automotive operation at Intel, and is responsible for technical support of Intel's CAN product family. He has an extensive background in test development and fault grading of 16-bit microcontrollers. Schill is a graduate of DeVry Institute of Technology.

This article was taken from a manuscript originally written by Craig Szydlowski.

REFERENCES

82527 Serial Communications Controller data sheet, order #272250, Intel Literature, (800) 548-4725.

CAN Specification version 2.0, Robert Bosch GmbH, Postfach 50, D-7000 Stuttgart, 1991.

Crovella, Robert M. *SDS: A CAN Protocol for Plant Floor Control*. Presented at CAN In Automation Conference, Mainz, Germany, September 1994.

DeviceNet Specification, Volume 1, Release 1.1, Allen-Bradley.

EXHIBIT C

Digital networks in the automotive vehicle

Article *in* Computing and Control Engineering · January 2000

DOI: 10.1049/ccej:19990604 · Source: IEEE Xplore

3 authors, including:

Gabriel Leen
University of Limerick
86 PUBLICATIONS **841** CITATIONS

SEE PROFILE

Digital Networks in the Automotive Vehicle

By Gabriel Leen[1], Donal Heffernan[1], and Alan Dunne[2]

Nothing stands still for long in the world of electronics. The automotive industry is party to this phenomenon and is fast becoming a breeding ground for binary electronic life forms. Today's vehicles include a complex symbiosis of intelligent electronic systems and integrated mechanical structures. The rapid growth of the computer industry has spawned a host of modern solutions and opportunities for automotive systems. It could be argued that the electronics revolution of the past two decades is the single biggest driving force behind the evolution of the motor car. Today, electronic components and systems account for over 20% of the cost of a high-end passenger car, and this percentage figure is increasing rapidly.

The customer is demanding more and more sophisticated features at an affordable price. The automotive manufacturers are competing to meet such customer demands within the market price envelope. New regulatory safety and fuel related standards (emissions, fuel efficiency etc.) are presenting further challenges to the manufacturers. Electronic systems now provide the technology to enable the manufacturer to deliver new features and to meet the mandatory regulation requirements in a cost-effective manner. Vehicle electronic systems are now common place and are growing in terms of both quantity and complexity. In this paper we take a glimpse under the hood of the modern vehicle to provide an insight into the world of automotive electronics, with special emphasis on the networking aspects of these electronic systems.

Developments in automotive networks

Up until recently the in-vehicle communication between simple devices such as switches and actuators was achieved using point-to-point wiring; resulting in bulky, expensive, and complicated wiring harnesses which were difficult to manufacture and install. With the expanding number of features within the vehicle the amount of wiring grew to a stage where the volume, reliability and weight became a real problem. Figure 1 shows the growth of vehicle wiring requirements for Volvo passenger cars over nearly eight decades[i]. The problems associated with the vast amount of vehicle wiring can be summarised as follows:

- shrinking layout space
- manufacturing and assembly difficulties
- deterioration of serviceability
- the cost/benefit ratio, does not encourage when adding additional functions at the expense of extra wiring

- increased emphasis on fuel efficiency and performance (acceleration, deceleration) requires reduction in vehicle weight
- sensor/input data being inefficiently distributed by multiple discrete signal channels
- the numerous connectors lead to unreliable operation; each link reducing the mean time between failure

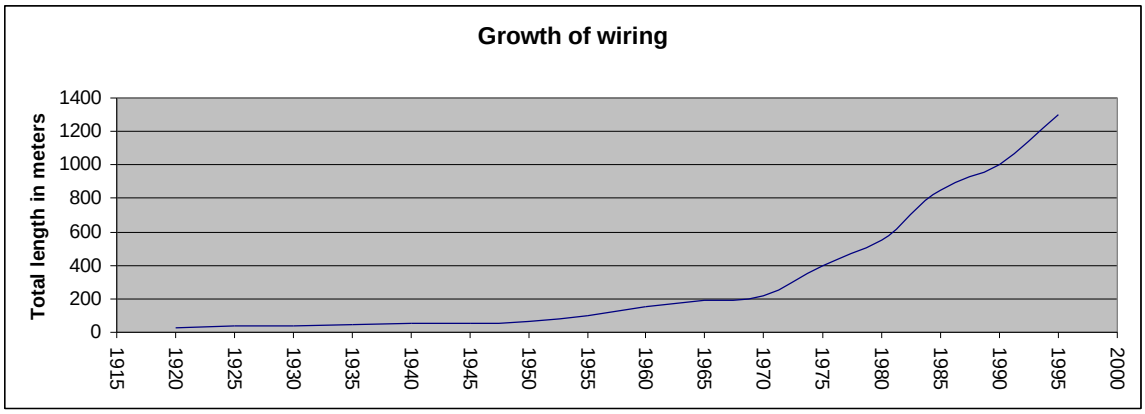


Figure 1 Growth of Automotive Wiring

The need to reduce the vehicle wiring content and to improve the distribution of control and monitoring functions within the vehicle became apparent over the years and solutions for vehicle networking started to emerge during the 1980s. Figure 2 charts this historical progression and shows many of the early and current vehicle networking solutions[ii]**.** Many of the technical concepts for vehicle networking were borrowed from developments in the area of computer data networks but vehicle communication requirements are driven by control strategies rather than by classical data transfer strategies.

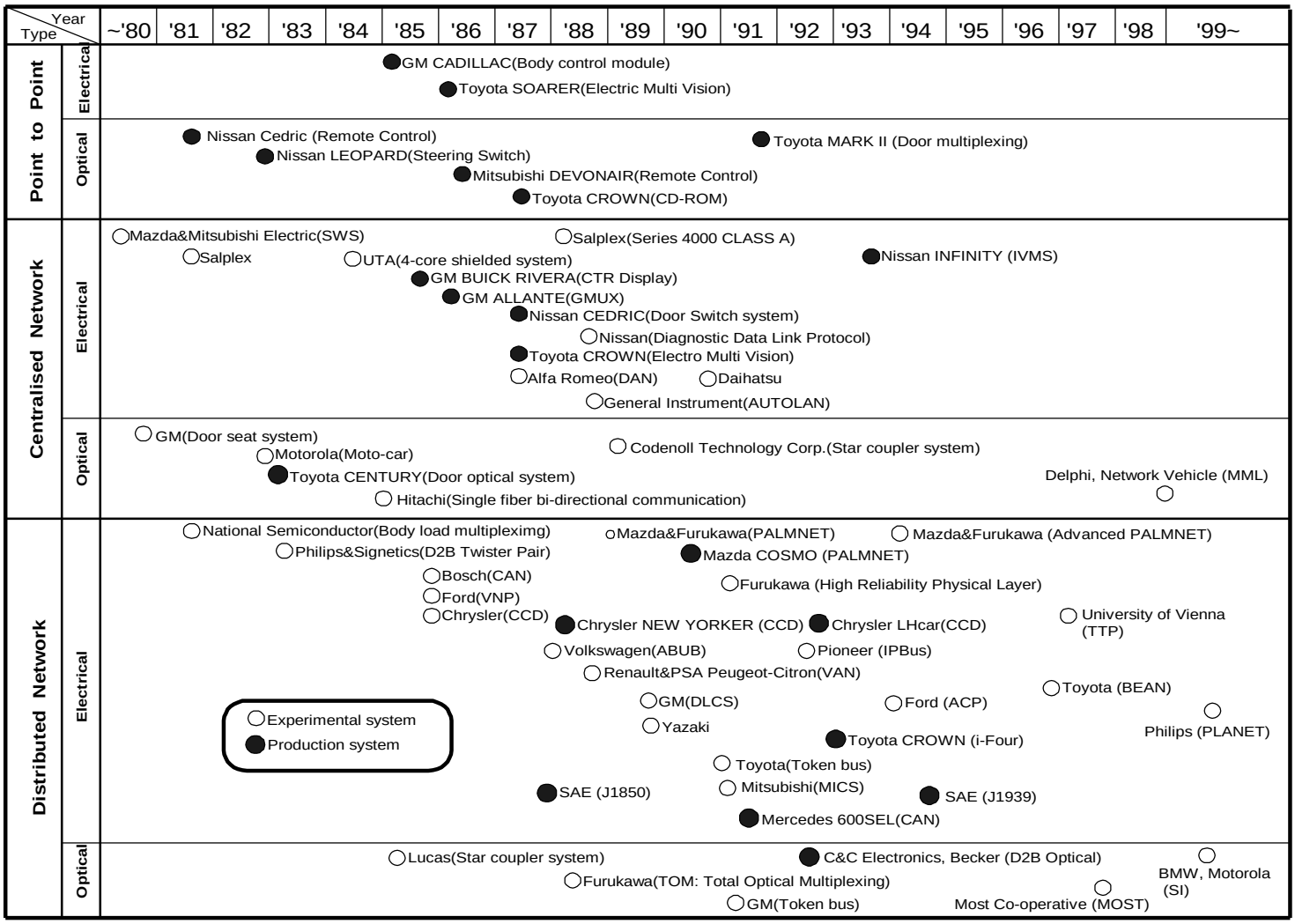


Figure 2 Automotive Network Development

In the early days of vehicle networking development there was little interest in devising common networking standards. Many of the early networks made use of custom circuits and generic UART (Universal Asynchronous Receiver/Transmitter) devices to provide simple serial communication links. This approach was acceptable at the time, as then most manufacturers were vertically integrated and not as highly dependent on external suppliers, as they are today. As confidence grew and the benefits of adopting a standardised networking approach became more apparent, external suppliers were commissioned to develop increasingly more sophisticated modules which finally resulted in a move away from proprietary interfaces in favour of industry-wide standardised protocols. The standardisation of protocols helped facilitate the integration of systems developed by different suppliers, leading to a type of 'open architecture', and added a degree of composability to the system design process. In-vehicle networking was initially introduced in high end luxury class models (as with all leading-edge technology), but the standardisation efforts which followed helped to support the economy of scales necessary for its introduction into the mid-range and standard class

vehicle. Today networked electronic subsystems are a vital element in all classes of vehicle and as can be seen from Figure 3[iii] this electronics presence in vehicles is growing rapidly.

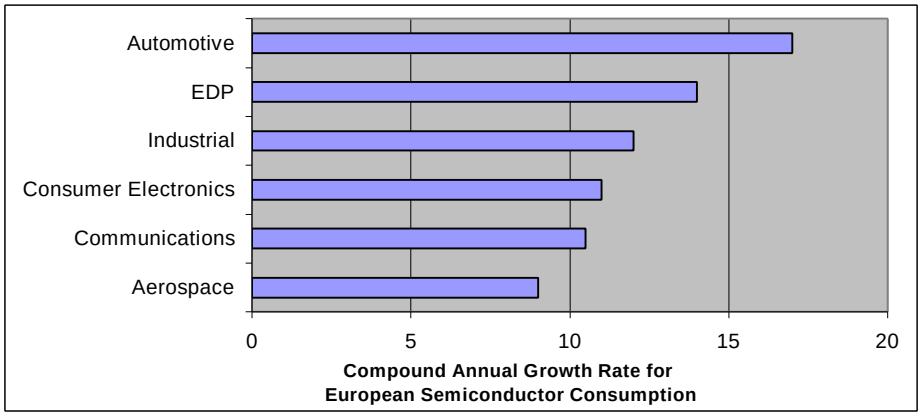


Figure 3 Semiconductor Consumption Compound Annual Growth Rate

In the US the SAE (Society of Automotive Engineers) has published a series of documents describing recommended practices for vehicle networking. The SAE has also formally classified vehicle networks based on their bit transfer rates, Table 1 illustrates these categories. Specific standards exist for Class A, Class B and Class C networks, but the SAE have not yet defined specific standards for Class D networks. However networks which exceed a data rate of 1Mb/s are often referred to as Class D networks.

Network Classification	Speed	Application
Class A	<10 kb/s Low Speed	Convenience features e.g. trunk release, electric mirror adjustment
Class B	10 – 125 kb/s Medium Speed	General information transfer e.g. instruments, power windows
Class C	125 kb/s – 1 Mb/s High Speed	Real time control e.g. power train, vehicle dynamics
Class D	> 1 Mb/s	Multimedia applications e.g. Internet, Digital TV Hard real time critical functions e.g. X-by-Wire applications

Table 1 Classification of automotive networks

A typical motor vehicle represents an extremely hostile environment for electronic equipment, subjecting this equipment to adverse conditions such as: mechanical vibration; temperature swings from –40 ^{o}C to +80 ^{o}C; splashes from oil, petrol and water; ice; strong electromagnetic fields (automotive field strengths can be > 200 V/m, domestic is

around 3 V/m and industrial 10 V/m); electrical spikes/transients of both polarities (> ± 100V); load dumps; jump starts; high humidity; dust; sand storms; and potential mis-wiring of electrical systems (e.g. short circuits to ground or positive, reverse battery, etc.).

Along with the environmental considerations, automotive network solutions require some special design considerations, such as:

- **High integrity** The probability of an undetected error must be negligible for the life span of the vehicle
- **Bounded determinism** A guaranteed upper threshold on message latency time for control problems
- **EMC compliance** Both emitted radiation levels and the tolerated absorption levels must be met
- **Low interconnection count** Each additional connector increases the probability of a potentially life endangering fault
- **Compact connectors** The connector is often the largest component on an automotive electronic module
- **Low cost** Costs are critical. A saving of a few pennies on a component is substantial in high volume production
- **Network composability** Variations across models and after market extras require a network which is easily expand and modified
- **Fault Tolerance** Communication must be restored when faults are removed and redundancy is becoming important

The typical vehicle will have more than one network system

Electronic equipment is distributed throughout a modern vehicle supporting a host of functions as illustrated in Figure 4. The functionality for a given automotive system is often distributed to span more than one network system. For example, the interplay between an engine management system network and the traction control system network needs to be carefully defined where, for instance, a reduction in engine torque is necessary to reduce drive wheel slippage, through fuel and engine timing adjustment. Another example is an intelligent auto-routing function, which might require information from a number of systems, gathering variables such as ABS wheel position data, steering wheel angle, GPS satellite data, and the radio RDS-TMC (Radio Data System - Traffic Message Channel) information.

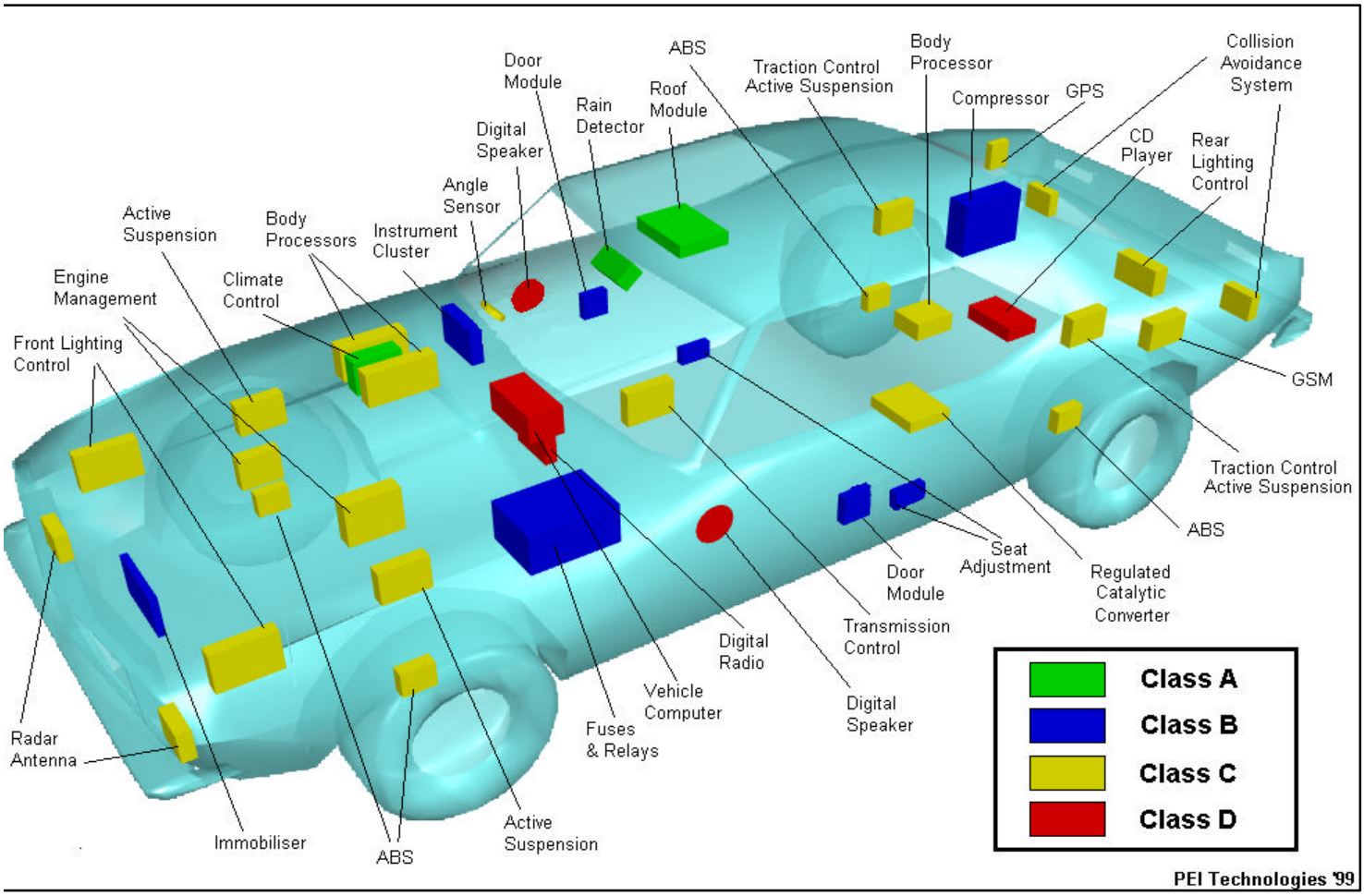


Figure 4 Electronic equipment in a modern motor car

Automotive Network Solutions

Table 2 shows a selection of automotive network solutions. There is a wide range of automotive networks reflecting defined functional and economic niches. High bandwidth networks are used for vehicle multimedia applications, where cost is not excessively critical. Reliable, responsive networks are required for critical real-time control applications such as powertrain control, and vehicle dynamics. In such demanding control applications the functional needs dominate over the economic ones and it is sometimes necessary to implement networks incorporating fail-safe redundancy. Comfort electronic systems such as power windows, adjustable seats and some instrumentation require only modest response times which only just surpass human perception times. Comfort systems are far more cost sensitive and the associated networks and modules are highly tuned to be cost effective. Certain vehicle functions may be implemented on a very minimalist network, supporting simple features, such as: trunk release and central locking, where delays in the order of a second are acceptable. Often single wire networks running at 10 kb/s are suitable here.

Protocol	Affiliation	Application	Media	Bit Encoding	Media Access	Error Detection	Data Field Length	MAX. Bit Rate
ABUS	VW	Control	Single wire	NRZ	Contention	Bit only	16 bit	500 kb/s
APC	Ford	Audio	Twisted pair	NRZ	CSMA/CA	Checksum	64 bit	9.6 kb/s
AUTOLAN	General Inst.	Control	Twisted pair	API	Master/slave	CRC	0 – 64 bit	4 Mb/s
BEAN	Toyota	Control	Single wire	NRZ	CSMA/CD	CRC	8 – 88 bit	10 kb/s
CAN	Bosch	Control	Twisted pair	NRZ & Stuffing	Contention	CRC	0 – 64 bit	1 Mb/s
CCD	Crysler	Sensor Mux.	Twisted pair	NRZ	CSMA/CR	CRC	Un-limited	~7.8 kb/s
CSC	Crysler	Sensor Mux.	Twisted pair	Voltage	Polling / Addressing	__	1 bit	~1kb/s
D2B	Optical Chip Consortium	Audio / Video	Fibre Optic	PWM	Contention	?	?	12 Mb/s
DAN	Alfa Romeo	Dash Board	Twisted pair	NRZ	Master/slave	CRC	8 bit	9.6 kb/s
DSI	Motorola	Sensor Mux.	2 wire	Voltage & Current	Master/slave	CRC	16 bit	= 5 kb/s
IVMS	Nissan	Control	Twisted pair	PWM	Polling	Parity	16 bit	~27.8 kb/s
J1850 PWM	SAE	Control	2 wire	PWM	CSMA/CR	CRC	8 – 64 bit	41.6 kb/s
J1850 VPW	SAE	Control	1 wire	VPW	CSMA/CR	CRC	8 – 64 bit	10.4 kb/s
J1939	SAE	Control	Twisted pair	NRZ & Stuffing	Contention	CRC	0 – 64 bit	1 Mb/s
MML	Delphi	Multimedia	Fibre Optic	NRZ	Master/slave	?	= 2048 bit	110 Mb/s
MOST	Most Co-op	Multimedia	Fibre Optic	?	?	?	?	25 Mb/s
PALMNET	Mazda	Control	Twisted pair	NRZ	Contention	CRC	32 or 64 bit	1 Mb/s
TTP	TTTech	Real Time Control	2 channel	MFM	TDMA	CRC	128 bit	2 Mb/s, 4 Mb/s soon
VAN	Renault & PSA	Control	Twisted pair	Manchester	Contention	CRC	0 – 64 bit	~250 kb/s

Table 2 A selection of automotive networks

Gateways and bridges are devices employed to interconnect multi-network architectures. Gateway devices can connect dissimilar networks whereas bridges are used to connect networks which have common data link layer protocols. Gateways and bridges can filter data passing between functionally independent network segments, allowing only the messages required by a module on another network segment to pass through. For example, if the trip computer requires the fuel tank level data in order to determine the optimum refuelling point, it can receive data, which may have originated on a Class A network node. This data may have travelled via a bridge to a Class B network and then via a gateway to a Class D network. Often diagnostic interfaces are optimally placed on gateway or bridge nodes for strategic traffic monitoring.

Automotive control networks

Conventional automotive control networks are widely used in control systems for applications such as engine management, door control, etc. These networks operate at moderate data rates (SAE Class B or Class C). The automotive system design engineer must consider the functional requirements in terms of information flow, network bandwidth and response times. The following issues will require specific consideration:

- Worst case network traffic load which includes: inter-network traffic; diagnostic data; etc.
- Individual message priority assignment
- Physical and Logical distribution of data sinks and sources
- Network expansion capacity for an entire range of vehicle models
- Error probability and its effects on traffic latency
- System level FMEA (Failure Mode & Effect Analyses) and fault tree analyses results
- Physical length constraints for network segments
- Fault tolerance behaviour and possible redundancy
- Optimum placement of bridges and gateways
- Network management control schemes and associated traffic

Fortunately, the automotive industry has created and are adopting market-wide standards for vehicle control networks; in order to minimise production costs, through mass production. A global consensus on methods and implementations enhances the total product development cycle, ultimately impacting on the final cost to the consumer. CAN and J1850 are currently two of the most successful standards for vehicle control networks.

In Europe the dominant vehicle control network is CAN (Controller Area Network). This protocol was developed by Robert Bosch GmbH in the mid 1980's and was first implemented in a Mercedes Benz S-class car, in 1991. CAN has since been adopted by most major European automotive manufacturers and a growing number of US companies are now using CAN. In the US, in 1994, the SAE Truck and Bus Control and Communications subcommittee selected CAN as the basis for the J1939 standard (a Class C network for truck and bus applications). The ISO standardised CAN as an automotive networking protocol: ISO 11898 and ISO11519-2.

Many of the world's major semiconductor companies now offer CAN implementations. It is estimated that there are already over 140 million CAN nodes installed world-wide. [Although CAN was developed as a vehicle network standard, it is interesting to note that, currently, the majority of CAN applications exist outside of the automotive industry, employed in numerous other applications ranging from farm machinery to photocopiers.] CAN may be implemented as a Class A, Class B or Class C network.

J1850

In the US, the SAE adopted J1850 as the recommended protocol for Class A and Class B networks. This protocol was the result of a co-operative effort among the 'Big Three' car companies: GM, Ford, and Chrysler. The protocol specification is a combination of GM's Class 2 protocol and Ford's SCP (Standard Corporate Protocol). Emissions legislation was highly influential in the standardisation of J1850. The OBD-ll (On Board Diagnostics ll), created by CARB (California Air Resources Board) requires the implementation of diagnostic tools for emission-related systems. It specifies that stored fault codes should be available through a diagnostic port via a standard protocol, namely J1850 and the European standard, ISO 9141.

High bandwidth automotive networks

Formula One competitions are not the only races in the automotive arena. The race to introduce an automotive multimedia network standard is well under way. It is just a matter of time before networked in-car PCs will become options or standard features in motor cars. Such PCs will provide a host of additional functionality for entertainment, navigation and business applications. A number of companies such as Microsoft, Saab, Mecel, Intel, Clarion and others are working on their vision of the street-computer for in-vehicle use, and have demonstrated their work in the Personal Productivity Vehicle. Meanwhile, IBM, Delco, Netscape and Sun Microsystems are developing the Network Vehicle which uses Java as its operating environment. The development of such in-car personal computers will support features such as:

- Voice activated control for many functions
- Internet access from the car
- Text to speech e-mail reading while you “drive and listen”
- Voicemail
- Auto-route planner with real-time updates using the traffic reports from your radio’s RDS or the Web
- Advanced interactive digital audio and video features
- Computer games and in-car-entertainment systems for the back seat passengers

The emerging technology will allow the vehicle to become a true mobile office or a sophisticated gaming arcade, at the press of a button. However, the question of driver distraction and the associated effects on safety has yet to be resolved.

A new class of vehicle network is emerging to connect the forthcoming in-car personal computers and their peripherals. The Optical Chip Consortium has specified a network called D2B (Domestic Digital Bus) which is a fiber optic based solution offering approximately 12 Mb/s of bandwidth. This network is currently used in the new Mercedes S-Class. Oasis Silicon Systems have developed the MOST (Media Orientated Systems Transport) solution, again with a fiber optic physical layer, giving a transfer rate of 25 Mb/s. Delphi Automotive Systems provide a solution in the form of MML (Mobile Media Link). Fiber optic based MML has an impressive transfer rate of 110 Mb/s.

Toyota, GM, Ford, Daimler, Chrysler and Renault founded AMIC (Automotive Multimedia Interface Collaboration) in October 1998[iv]. AMIC represents a global project to standardise the vehicle multimedia architecture for the 21st century. The plug-and-play ‘infotainment’ specification will encompass software interfaces for vehicle systems, connectors and network implementations. Critical to this architecture are gateways and firewalls, preventing PC software (viruses, etc.) or malfunctions from interfering with the vehicle’s other control and data networks. There is a clear incentive for companies to have their technologies incorporated into the vehicle multimedia standards and Microsoft is making a significant effort with the Windows CE based Auto PC, unveiled in 1998. However, the automotive giants are determined to remain in the driving seat, and it is unlikely that they will uncharacteristically tie themselves to a single supplier from the onset. From a network perspective it is probable that an IDB-C (Intelligent Data Bus-CAN) solution will become one of the adopted standards. IDB-C is based on CAN’s physical and data link layers, but will be complemented by a higher speed multimedia bus, almost certainly based on fiber optic media, similar to the MOST or MML solutions. The in-car PC is expected to have a USB connection and a standard IrDA (InfraRed Data Association) port, leaving ample room for peripheral expansion and after market upgrades.

Control by wire networks

Networked electronic modules are replacing the equivalent mechanical systems! Electrical and electronic systems are eliminating power steering pumps, hoses, hydraulic fluid, drive belts, pulleys, and brake servos. Systems like E-Steer (Steer-by-wire) from Delphi Automotive Systems will be seen in high volume European vehicles later this year[v].

X-By-Wire is an EU-funded BRITE-EURAM research project (contract number BRPRCT95-0032), which has resulted in the design of a novel computer architecture, TTA (Time-Triggered Architecture), which is bases on time-triggered technology for fault-tolerant distributed embedded real-time systems. Fundamental to this strategy is the communications protocol TTP[vi] (Time-Triggered Protocol), where channel access control is based on a TDMA (Time Division Multiple Access) scheme, derived from a fault tolerant common time base. Prof. Hermann Kopetz at Vienna University of Technology is the authority on this extremely well designed protocol, which has a bright future, not only in the automobile but also in the aerospace and rail industries. TTP it is extremely reliable and deterministic but somewhat less flexible than most other automotive control networks.

TTA has defined a network architecture to replace the fore-mentioned mechanical systems, Figure 5 illustrates the concept. The replacement of mechanical systems with electronic solutions potentially offers several advantages:

- Less expensive in volume production
- Simplifies manufacturing of left hand drive and right hand drive vehicles
- Simplifies the general assembly of vehicles
- Intelligent self diagnosing systems, offering enhanced reliability and dependability
- Less mass, thus enhancing vehicle performance
- Environmentally compatible, no fluid necessary
- More compact
- Simplifies integration of auxiliary systems like active collision avoidance and cruise control systems

The drive-by-wire concept is perhaps a little daunting at first, however, when one considers that we have been travelling in fly-by-wire controlled aircraft for many years now, drive-by-wire does not seem to be such a big step.

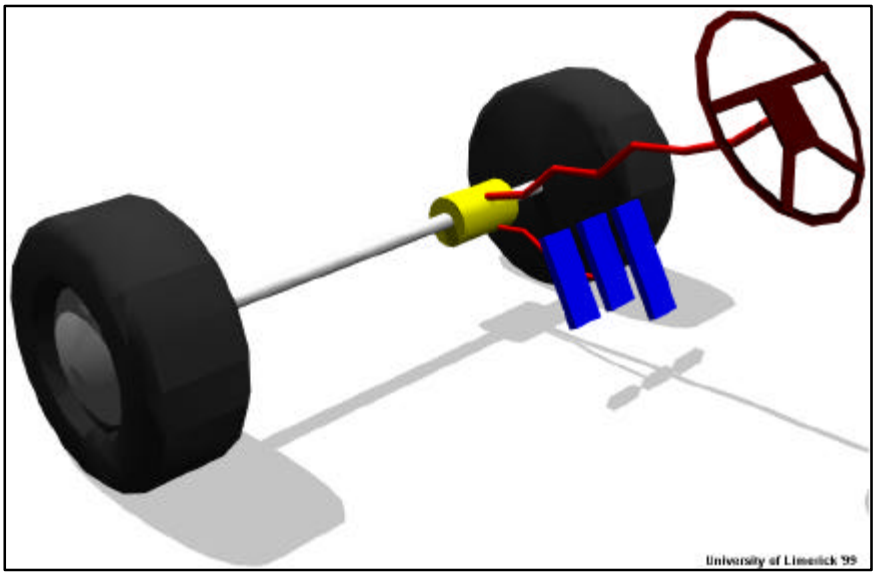


Figure 5 Drive-by-wire

Distributed software development for automotive systems

The development of distributed real-time control software is a challenge for any engineering design team. The combined hardware and software solution needs to be arrived at in a cost-effective manner, which requires a formalised approach to design, prototyping and testing**.** To this end rapid prototyping methods are now being introduced into the development cycle for vehicle electronics. The emerging rapid prototyping based development cycle is initiated with the generation of a requirement analysis specification for the proposed system. A CASE (Computer Aided Software Engineering) tool assists in this analysis and leads into a computer aided design phase. The software design approach is moving towards an object oriented model and the challenges of creating large-scale distributed applications can be approached by using visual modelling tools such as UML (Unified Modelling Language). The output of such tools can be cross-linked to the engineering documentation activity so that both code implementation, textual specifications and system diagrams are combined as different views of the same system. As Figure 6 illustrates the conventional approach of path B is being replaced by the interactive method, path A. Automatic code generation from the specification representations of the system is now possible, often leading to pre-production quality software code. The resulting code must be tested in a hardware environment. Here perhaps the most powerful advances are being made. The prototype software can be executed on a variety of development platforms such as a VME hardware rack, Power PC platforms, PCs, or the actual target hardware. Configurable and scaleable operating systems allow the hardware requirements to be accurately assessed and quantified early in the design cycle. In fact, software development can be completed before the

final hardware is completely designed. HiL (Hardware in the Loop) implementations are possible where the real time simulation is comprised of one or more components which exist as real hardware, while other components are simulated as mathematical models using tools such as Mathlab or Simulink.

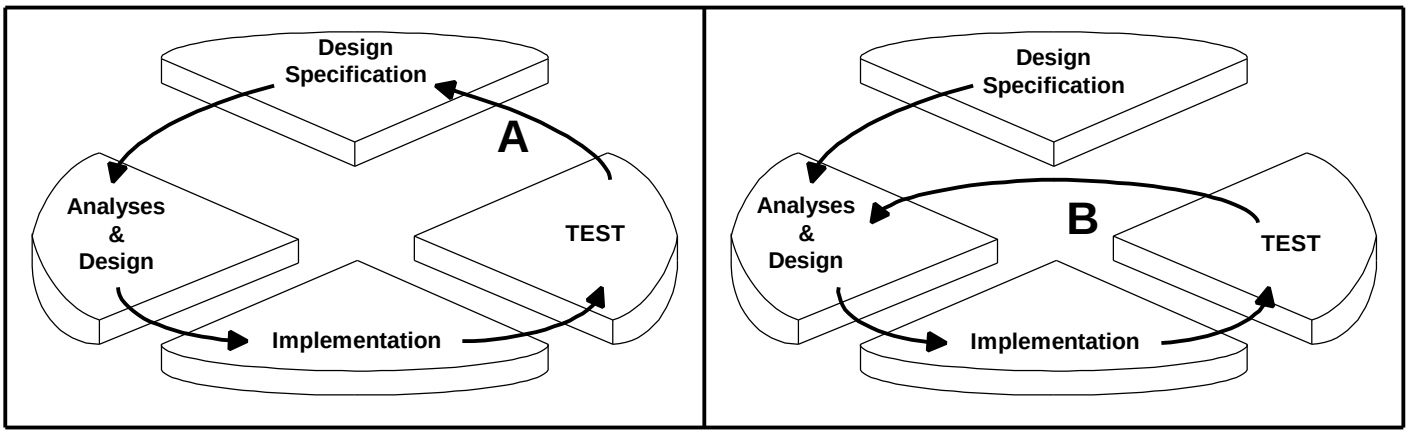


Figure 6 Software development cycle

Software testing of the prototype implementation, using automated test sequences, dramatically cuts down on the otherwise repetitive testing which hitherto may have required actual trials in the car. Based on the evaluation of the implementation performance the loop iterates until ready to proceed directly to early target resident executable code.

Because software has now been assigned such an important role in automotive systems, standardised software layers are being defined for vehicle networks. Such layers offer functionality such as operating systems, communication interface systems and network management systems. The recent developments in the area of OSEK/VDX provide a good example of such an approach.

OSEK/VDX

In May 1993, OSEK was initiated as a joint project by a number of companies within the German automotive industry. OSEK aims to achieve a standard open-ended architecture for distributed control units within vehicles. OSEK is an abbreviation for the German term “Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug”. The English: equivalent is: “Open Systems and the Corresponding Interfaces for Automotive Electronics”. The French car manufacturers PSA and Renault were working on a similar project, the VDX-approach (Vehicle Distributed eXecutive), and they joined OSEK in 1994, giving rise to the OSEK/VDX standard. OSEK/VDX aims to specify a standardised interface to allow the gelling together of hardware modules, network protocols and application software in a co-ordinated and intelligent fashion.

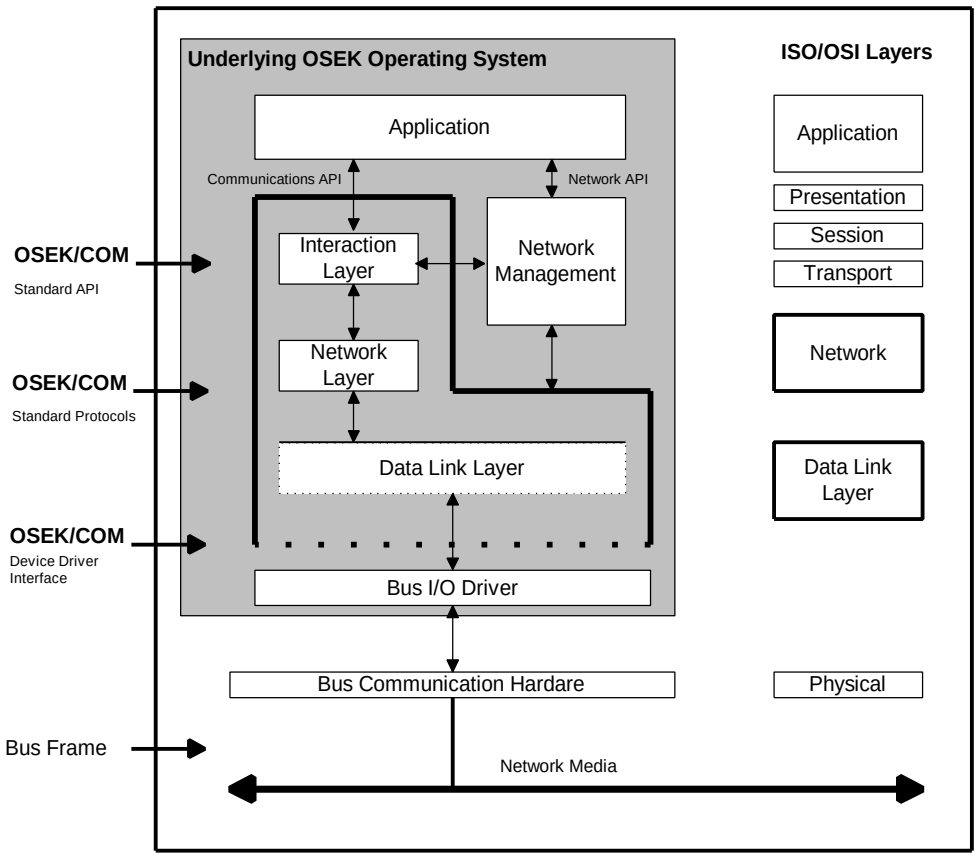


Figure 7 OSEK/VDX structural overview

OSEK/VDX specifies a series of interfaces, which allows for the development of software components, which are portable and reusable. Figure 7 shows a structural overview of OSEK/VDX[vii]. The software application interface specification is abstracted from the hardware and the network involved. This concept is illustrated in Figure 8. Functionality is both configurable and scaleable, enabling optimum tuning of architecture and application. The specification facilitates functional verification and validation. Because the API (Application Programming Interface) is standardised, development on various platforms without the need to learn a new tool set is possible, resulting in a clear saving on development time. Software from different suppliers can now co-habitate within a single microcontroller.

For the first time the potential for real co-operation between the various system suppliers is made possible, where traditionally products were developed independently, and rivalry rather than co-operation was the order of the day. Now those who hold the purse strings are changing the rules somewhat.

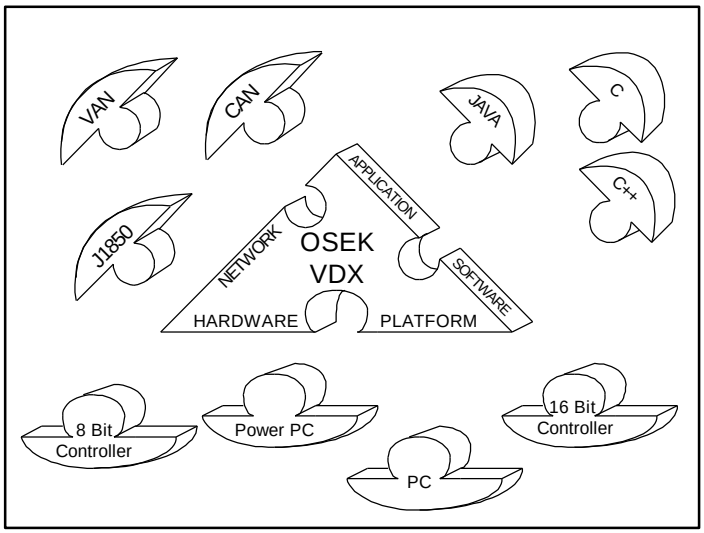


Figure 8 OSEK/VDX: Independence of the network, hardware and applications

OSEK/VDX realises a distributed operating environment within the vehicle; based on the same principles used to implement distributed operating environments within large data networks. OSEK/VDX‘s ultimate goal is to reduce costs by enabling the creation of re-usable embedded application software. The PC industry has shown that once an adopted operating system standard is in place, products become far more commercially competitive.

However, it is worth noting that the flexibility offered by the OSEK/VDX approach has some cost implications in terms of additional ROM, RAM and processor overhead. If OSEK compliant software implementations demand higher specification processors, then cost saving, in terms of software reuse, in the short term may be more than offset by additional hardware costs accrued over millions of units[viii]. Never the less, future silicon prices will decrease, and the benefits of reusable, mature and validated software, in the long term, is probably worth the initial investment.

It is interesting to note that the aerospace industry has expressed considerable interest in the OSEK/VDX developments, viewing it as a potential solution to their similar problem set.

Conclusions

Vehicle networks were invented by necessity to resolve the problems associated with bulk wiring harnesses. However, the coming of automotive networks has opened up new opportunities for the industry. Control networks are now well established in high-end vehicles and are quickly filtering down to all vehicle classes.

Whereas control networks operate behind the scenes, as far as the driver and passengers are concerned, it is the high bandwidth multimedia networks, which will have the biggest impact on customer perceptions; offering new concepts in in-vehicle work and entertainment features.

Control-by-wire solutions will provide new engineering solutions and challenges, where car users will need to trust electronics to guarantee their safety in the absence of the tried and tested mechanical systems. The service and maintenance industry and the average car enthusiast will have to find room in their toolbox's for a PC!

Software development is becoming as important as engine design or chassis design in the automotive design process. Soon we should expect to see the emergence of automotive specific software suppliers and a possible redirection of emphases within existing supplier companies.

Automotive related industries should benefit from the standardisation of vehicle network architectures and software environments. At the end of the day the ultimate winner will be the consumer.

Acknowledgements

The assistance of PEI Technologies, University of Limerick, is greatly appreciated.

Representative vendors and organisations for Automotive Technologies

Networking

Amp
(Connectors)
Harrisburg, PA
1-717-564-0100
fax 1-717-986-7575
www.amp.com

C & C Electronics
(D2B)
Guildford, Surry, UK
+44-1483-540-248
www.candc.co.uk

Oasis SiliconSystems
(MOST)
Austin, TX
1-512-306-8450
fax 1-512-306-8442
www.oasis.com

TTTech Computertechnik GmbH
(TTP/TTA)
Schönbrunnerstrasse 7
A-1040 Vienna
Austria
www.tttech.com

CAN in Automation
(CAN)
Am Weichselgarten 26
D-91058 Erlangen
Germany
49-9131-690 86-0
Fax 49-9131-690 86-79
www.can-cia.de

Operating Systems

Wind River Systems
(Tornado)
Alameda, CA
1-510-784-4100
fax 1-800-545-9463
www.wrs.com

IIIT
(OSEK/VDX)
University of Karlsruhe, Germany.
http://www-iiit.etec.uni-karlsruhe.de/~osek/information.html

TRIALOG
(OSEK)
25, rue du Général Foy
75008 Paris, France
33 1 44 70 61 00
fax 33 1 42 94 80 64
www.trialog.com/osek.html

NRTA
(SSX5 RTOS, OSEK)
Innovation centre
York science park
York, England
11 1904 435129
fax: 11 1904 435130
www.ssx5.com

Automotive Computing

Delphi Automotive Systems
(Network Vehicle, MML)
5725 Delphi Drive
Troy, Michigan
48098-2815 USA
1 248.813.2000
fax 1 248.813.2670
www.delphiauto.com,

Microsoft
(Windows CE)
Redmond, WA
1-425-882-8080
www.microsoft.com/windowsce/autopc

IBM
(Network Vehicle)
US, 1-800-772-2227
http://www.alphaworks.ibm.com/networkvehicle/tdrive/index.htm

Mecel AB
(Auto. systems & PC)
Förrådsgatan 5
Box 73
SE-662 22 Åmål
Sweden
+46 532 621 00
Fax: +46 532 151 39
http://www.mecel.se/html/expanding_areas.htm

Development Tools

ILogix Inc.
(Rhapsody)
3 Riverside Drive
Andover, MA 01810
978 682-2100
fax 978 682-5995
www.ilogix.com

Adept Scientific plc
(Mathcad)
6 Business Centre West,
Avenue One,
Letchworth, Herts
SG6 2HB, UK
01462 480055
fax 01462 480213
www.adeptscience.co.uk

The MathWorks Inc.
(Simulink, Mathlab, Stateflow)
24 Prime Park Way
Natick, MA 01760-1500
508-647-7000
fax 508-647-7101
www.mathworks.com

ETAS GmbH & Co.KG
(LabCar)
Borsigstrasse 10
D-70469 Stuttgart
49 (711)8 96 61-102
fax 49 (711)8 96 61-106
www.etas.de

Others

PEI Technologies
(Design & analyses)
Foundation Building
University of Limerick
Ireland.
353-61-202607
fax 353-61-334925
www.ul.ie/~pei

Surge-Transport
(Transportation Group)
7 Allée des Hetres,
F95570 Bouffémont,
France.
Fax 33-1-39359751
www.surge~europe.com

ERTICO
(Intelligent Transport)
326 avenue louise
1050 Brussels
32 2 4000 700
fax: 32 2 4000 701
www.ertico.com

SAE
(Society of Automotive Engineers)
www.sae.org

ITS America
(Intelligent Transport)
400 Virginia Avenue, SW., Suite 800
Washington, D.C. 20024-2730
202 484 4847
fax: 202 484 3483
www.itsa.org

---

[1] Dept. Electronic and Computer Engineering, University of Limerick, Ireland. *Donal.Heffernan@ul.ie, Gabriel.leen@ul.ie*

[2] PEI Technologies, Foundation Building, University of Limerick, Ireland. *Alan.Dunne@ul.ie*

References:

[i] Melin Kent: 'Volvo S80 Electrical system of the future', www.tech2.volvo.se, December 1998

[ii] Takesue, K.et al.: 'In-vehicle LAN', SAE-Australasia, July/August 1992, p43 - 48

[iii] 'Making it right before you put it on board', Surge Newsletter, 1st Quarter '99, Volume STR 1, Issue 2, p4 – p5

[iv] 'AMIC agrees on multimedia standard', The Hansen report on automotive electronics, VOL.12, No. 3, April '99, p1 – p3

[v] "Delphi Automotive Systems Sets the 'Pace' in Electric Power Steering", Press Release: 1999, March 2nd

[vi] Hermann Kopetz: 'Real-Time Systems', Kluwer Academic Publishers, ISBN 0-7923-9894-7

[vii] OSEK/VDX Communications Specification, version 2.1, revision 1, 17th June '98

[viii] Tindell, Dr. K., 'The challenges for embedded systems in cars', Embedded Systems'99 Nuremberg, 2nd March

EXHIBIT D

Embedded Communication Protocol Options

Bhargav P. Upender
United Technologies Research Center
East Hartford, Conn.

Bhargav Upender is an assistant engineer at United Technologies Research Center. His research interests include protocol development, modeling and simulations, and performance analysis. He holds a BSEE from the University of Connecticut and an MEng in electrical engineering from Cornell University.

Phil Koopman
United Technologies Research Center
East Hartford, Conn.

Phil Koopman is a senior researcher at United Technologies Research Center. He currently designs and evaluates architectures and communications protocols for Otis elevators, Norden radars, UT automotive components, and other embedded applications. He has previously worked for Harris Semiconductors as an embedded CPU architect and the U.S. Navy as a submarine officer. Koopman holds a BSEE and an MEng from Rensselaer Polytechnic Institute and a PhD in computer engineering from Carnegie Mellon University.

ABSTRACT

Developers are realizing that traditional low-speed, point-to-point links are inadequate for their increasingly complex distributed embedded applications. Consequently, they are investigating multiplexed communication network protocols to incorporate advanced system capabilities, increase reliability, and reduce wiring requirements. This paper discusses special considerations for embedded system networks, a family tree of "standard" protocols, media access tradeoffs, and attractive options for off-the-shelf solutions. Based on real-time performance, cost, and hardware availability, ***ARCnet, CAN,*** *and* ***LON*** *are strong contenders for most embedded systems.*

Embedded systems are becoming more and more complex. One of the ways to manage this complexity is to distribute the system functionality across several low cost microprocessors which communicate via a shared medium.

In the past, most physically distributed embedded systems used simple point-to-point links to provide inter-processor communication. With increasing demand for advanced features and the resulting drive for more flexible and cost-effective communications, engineers are starting to use LAN (Local Area Network) technology in embedded systems. Most LANs are based on Ethernet, which is ideal for workstation-like applications having aperiodic, bursty communication traffic. Unfortunately, many embedded systems are unlike workstations in that their communication networks must efficiently support periodic traffic, real-time constraints, prioritized messages, and cost-sensitive applications. In this paper we will discusses these special considerations for real-time embedded networks, explore "standard" protocols, discusses media access tradeoffs, and identify a few attractive off-the-shelf solutions.

SPECIAL CONSIDERATIONS FOR EMBEDDED APPLICATIONS

Based on our examination of several embedded applications, we believe that communication traffic for embedded systems tends to be mostly short, periodic messages. Because cost limits the network bandwidth of many applications, protocol ***efficiency*** (message bits delivered compared to raw network bandwidth) is very important. Efficiency is improved by reducing packet overhead and media access overhead. Packet overhead is all non-data bits added by the protocol to ensure proper routing and reliable transportation (*e.g.*, CRC, address bits, acknowledgments). Media access overhead is the network bandwidth used to arbitrate network access among transmitting nodes (*e.g.*, token passing). Because worst-case behavior is usually important, efficiency should be evaluated both for light traffic as well as heavy traffic. For example, Ethernet is highly efficient for light traffic but gives poor performance if heavily loaded. Token passing protocols have the reverse properties. Therefore, protocol efficiency becomes a strong metric for selecting a protocol.

Due to real-time constraints of many control applications, ***determinacy***, the ability to predict

message latency, becomes very important. Also, ***prioritization*** capability is required in some applications to allow quick channel access to critical messages (*e.g.*, safety critical conditions) and messages in which minimum latency is crucial (*e.g.*, sensitive control loops). Priorities can be either assigned to each node or to individual messages. Additionally, they can be either local or global. In local prioritization, each node is only aware of priorities of its messages, and arranges them in the transmit buffer accordingly. In global prioritization, the protocol allows the message or node with highest priority among all of the network to transmit.

Many applications require robust operation under extreme operating conditions. A protocol is ***robust***, if it can quickly detect and recover from errors (*e.g.*, duplicate or lost tokens). Some applications may require dynamic additions and deletions of nodes from the network. In these situations, the protocol should gracefully initialize and configure itself.

Varied operating environments may dictate use of a flexible ***physical layer*** that can support multiple media and mixed topologies. For example, a system may require expensive fiber in noisy environments, but can tolerate low-cost twisted pair wires in benign environments. Further, a bus topology may be optimum for wires, but a ring or star topology maybe needed for fiber.

Finally, the most important consideration is the ***cost per node***. Most of the protocols discussed in this paper are for high speed, high performance networks that allow expansion of the capabilities of a system (*e.g.*, remote monitoring, diagnostics, and servicing). Therefore, the current costs may not be suitable for low-end embedded systems. However, with the current trend of increasing computing power and protocol support embedded in CPU chips, the costs are becoming more reasonable for all types of applications.

PROTOCOL FAMILY TREE

With the above considerations in mind, we surveyed the market for **standard** protocols for distributed applications. By identifying only standard protocols, we hoped to uncover low cost, off-the-shelf communication components and maintain interoperability with the other products. In particular, we hoped to discover one or two standards that were clear and obvious choices for embedded systems from both a technical and market perspective.

Much to our surprise, our survey resulted in more than sixty "standard" protocols. And, some of these standards specifically permit the use of multiple incompatible physical implementations. So much for simply picking "the" standard protocol for embedded applications!

In order to understand the relationship between these protocols, we developed a family tree (Fig. 1) for the most popular protocols. Most of these protocols can be well characterized as primarily addressing one of three different levels of standards.

•**Medium Access Control (MAC):** this level is part of the Data Link Layer of the Open Systems Interconnect (OSI) seven layer reference model[1]. This low-level sublayer defines the rules for bus sharing and arbitration. Every communication network uses one of these fundamental MAC protocols.
•**Protocol Implementations:** this level consists of hardware/software implementations of a MAC scheme. Market forces have made some of these protocols, the *de facto* standards in their application areas (*e.g.*, Ethernet, ARCnet).
•**High Level Standards:** this level represents protocols that are developed by world-wide standards committees. These standards are trying to provide cohesion and interoperability by addressing the higher, application layers of the OSI model.

MEDIA ACCESS CONTROL MECHANISMS

In order to make sense of this tangle of standards, we will proceed from the low level to high level. MAC protocols determine the basic technical merits of any communication network. Once we understand each MAC scheme, we can then see how higher level standards fit them together.

Connection Oriented Protocols

Before LANs became popular, connection-oriented protocols were heavily used to connect remote terminals to mainframes. Usually, the nodes are connected using point-to-point links (telephone wire, serial line, *etc.*). Communication between two nodes requires physical connection using handshaking signals, or logical connection via intermediate nodes. Connection
based protocols are deterministic between physically connected notes, and have readily available hardware and software. For an embedded system with modest communication requirements, this might be a cost effective protocol. Sometimes, this type of protocol is added to a more complex communication system to provide backward compatibility to older systems (*e.g.*, **BACnet**[2]). This type of protocol is used by the **X.25**[3] public network standard (network services offered by telephone companies) and IBM's System Network Architecture (**SNA**[3]).

Polling

Polling is one of the more popular protocols for embedded systems because of its simplicity and determinacy. In this protocol, a centrally assigned master periodically polls the slave nodes for information. Since polling is done through some type of token (special string of bits) passing, this protocol is also known as the Master/Slave Token Passing or MS/TP. The majority of the protocol software is stored in the master and the communication work of slave nodes is minimal. This protocol is ideal for a centralized data acquisition system where peer-to-peer communication is not required. However, for a more complex embedded system, the single-point-of-failure from the master node is unacceptable. Additionally, the

polling process has high MAC overhead and limited capabilities. These protocols have been standardized by the military (**MIL-STD-1553B**[4] and **MIL-STD-1773**[5]) for aircraft subsystem communications. Some variants of this protocol allow inter-slave communication through the master and multiple masters (e.g., **Profibus**[6]) for redundancy.

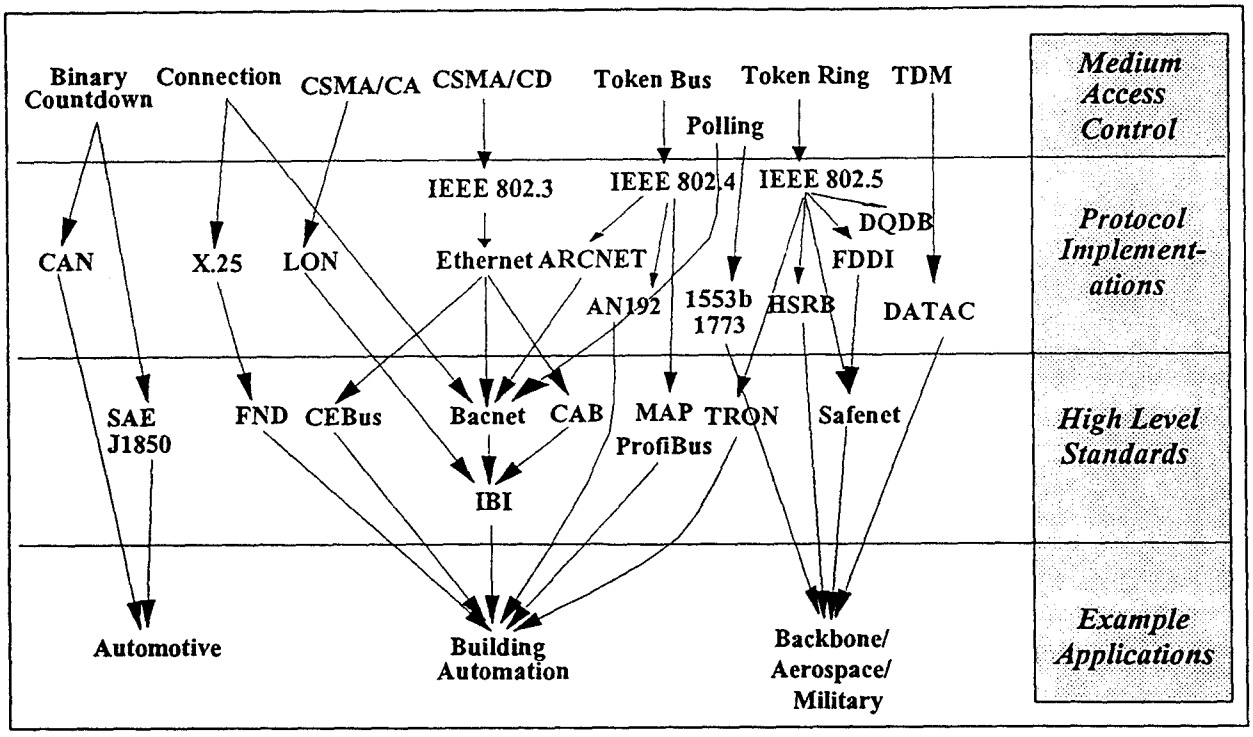


Figure 1: "Standard" Protocol Family Tree

Time Division Multiple Access (TDMA)

TDMA is heavily used in satellite communications[7], but is applicable to local area networks as well. In this protocol, each node transmits during its uniquely owned preallocated time slot. To maintain clock synchronization among all the nodes, a bus master broadcasts a frame sync signal before each round of messages. Like polling, TDMA is a simple protocol with deterministic response time that is well suited for balanced (evenly distributed), fixed length messages. Weaknesses include the bus master constituting a single-point-of-failure and bandwidth wasted by slots reserved for idle nodes. If a slot is not being used in some variations of TDMA, all stations can advance to the next slot early to conserve bandwidth (variable length TDMA). Time based protocols have been popular in military aviation applications. For example, **DATAC**[8], Digital Autonomous Terminal Access Communications, is being used by NASA and Boeing.

Token Ring

In a token ring, the nodes are connected in a ring-like structure using point-to-point links. A single token signal (special string of bits) is passed from one node to another around the ring.

The holder of the token has access to the network. This protocol offers a bounded latency, high throughput during heavy traffic, and global prioritization. Under light traffic, token ring has moderate token passing overhead. Initialization of the token message, detection of dual tokens or token loss adds to the complexity of the protocol. Many users are concerned that a break in the cable or a failed node can disable the network. However, node bypass hardware and dual rings can be used to address this concern[9]. Since the ring topology is unidirectional, it is well suited for fiber optics. Consequently, many LANs and Wide Area Networks (WANs) are moving to this type of protocol. For example, **FDDI** (Fiber Distributed Data Interface) uses dual counter-rotating rings to achieve higher reliability than bus or star topologies[10].

Token Bus

The operation of token bus is very similar to the token ring — a token is passed over the bus from one node to another creating a virtual ring. It works well under heavy traffic with a high degree of predictability. However, global prioritization of messages is inefficient, and latency under light loads is higher than for token ring because sharing a single bus precludes concurrent communication among logically adjacent nodes. Unlike unidirectional token ring, a break in the cable or a failed node does not disable the entire network. A lengthy reconfiguration process, where each node identifies its neighbors, is used to maintain the virtual ring when nodes are added or deleted from the network. Because bus-like topologies are well suited for manufacturing plants, **MAP**[11], Manufacturing Automation Protocol, adopted this protocol. Additionally, **ARCnet**[12], Attached Resource Computer Network, uses this protocol for LAN connectivity and process control. Adaptive Networks' **PLC-192** power line carrier chip uses a hybrid token bus protocol: under light traffic, nodes dynamically join and leave from the logical ring; under heavy traffic, all nodes join the ring to maintain stability[13].

Binary Countdown

In binary countdown, also known as the Bit Dominance Algorithm, all nodes wait for an idle channel before transmitting. Competing nodes resolve contention by broadcasting a signal based on their unique node identification value. During this transmission, a node drops out of the competition if it detects a dominant signal opposite to its own; thus, if a "1" signal is dominant, the highest numbered transmitting node will win the competition and gains ownership of the channel (Figure 2). It is also possible to arbitrate using unique message ID values rather than the node IDs to implement globally prioritized messages. This protocol has good throughput under light loads. A problem is that since all messages are prioritized, it is difficult to achieve fair access for all nodes under heavily loaded conditions. Using this protocol, Bosch developed a complete Controller Area Network (**CAN**[14]) specification for the automotive applications. The Society of Automotive Engineers standard **SAE J-1850**[15] also uses this protocol.

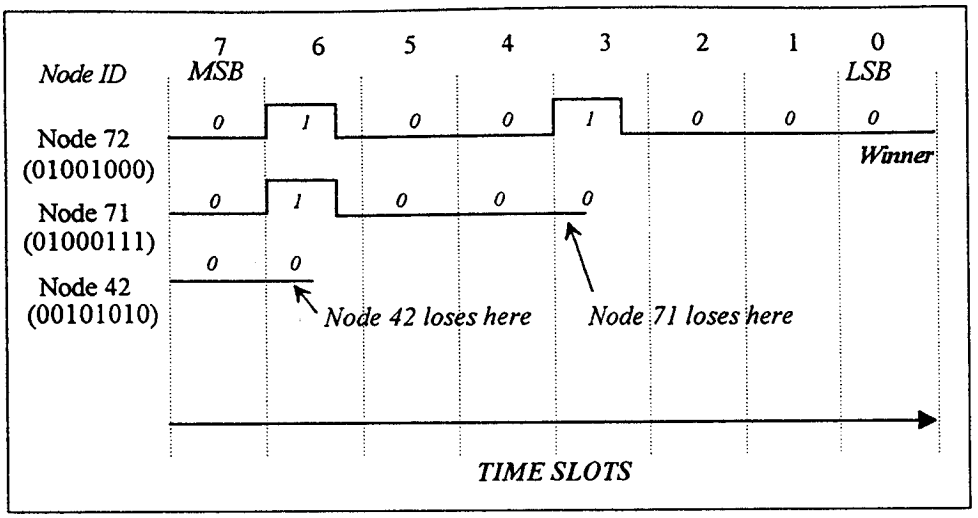


Figure 2: Bit Dominance Procedure

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

CSMA/CD has been widely researched with a large number of published variations[9,16]. In the simplest case, a node waits for the bus to go idle before transmitting. If multiple stations transmit simultaneously (within two bus propagation delays), the messages collide. The nodes must detect this collision, and resolve it by waiting a randomly generated time before retrying. The key advantage to this protocol is that it supports many nodes and allows the processors to enter and leave the network without requiring network initialization and configuration. Thus, for light traffic, MAC overhead is very small. However, under heavy traffic the MAC overhead is unbounded, leading to a protocol with poor determinacy. Furthermore, detecting collisions requires additional analog circuitry which translates to higher costs and difficult implementation in many embedded systems. The popular **Ethernet** protocol is based on CSMA/CD.

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

Many researchers have developed hybrid protocols that combine the light traffic efficiency of CSMA/CD with heavy traffic efficiency of token-based protocols. The resulting protocols are often called CSMA/CA, or collision avoidance algorithms.

As in CSMA/CD, nodes transmit after detecting an idle channel. However, if two or more stations collide, a *jam* signal is sent on the network to notify all nodes of collision, synchronize clocks, and start contention slots. These time slots, typically just over two propagation delays long, are assigned to each of the stations. If the number of slots equals the number of stations, the protocol is called Reservation CSMA or RCSMA. The RCSMA variation works efficiently under all traffic conditions, and global priorities can be assigned through slot allocation[17]. Using rotating slots (slots that change positions after each transmission), fairness can be maintained and latency can be predicted. However, because of

the one-to-one relation of the node to the slot, RCSMA is not practical for a network with a large number of nodes. Echelon's LON[15] (Local Operating Network) avoids this constraint by dynamically varying the number of slots (in some cases, fewer slots than stations) based on expected traffic and handling the case when multiple transmitters attempt to use the same slot.

A HAND-WAVING COMPARISON

In the above discussions we have summarized the major MAC protocols and noted clear differences. Figure 3 shows some of the common traits and the relationships between various MAC protocols.

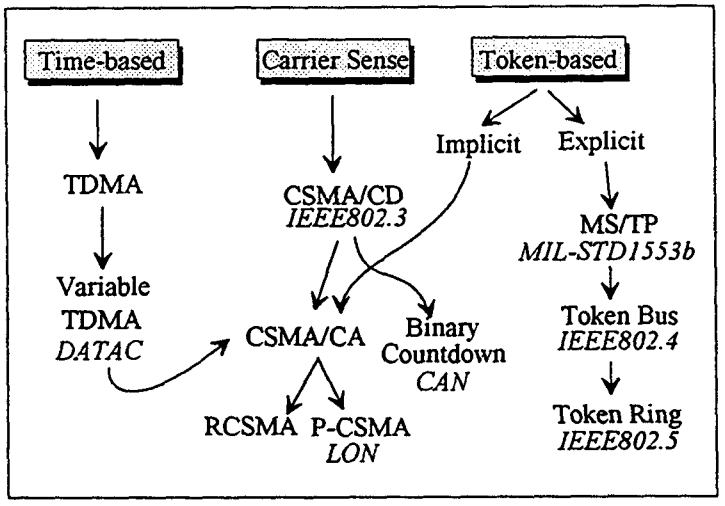


Figure 3: Media Access Relationships

Our opinions on the characteristics of all the media access protocols are compiled in Table 1. The important points to notice are:

Polling, TDMA, and connection-based protocols are simple, but do not provide sufficient flexibility for advanced systems.

Token-based protocols are predictable, but require complex software to maintain robustness.

CSMA/CD is a poor protocol for hard real-time systems with heavy traffic.

The collision avoidance protocols provide the best balance for embedded system requirements.

THE WORLD OF STANDARDS

With the understanding of the MAC protocols and sample implementation standards, we can discuss the high level standards. Most of these standards lack specific hardware to go along with the published specifications. These standards have been developed by many public organizations and corporate alliances at industry, national, and international levels. As one would expect, progress is slow and consensus is minimal. To achieve consensus, some

👍=Good ☞=Ok 👎=Poor	Efficiency Light Traffic	Efficiency Heavy Traffic	Real-Time	Prior-itization	Robust-ness	Physical Layer	Cost/ Node
Connection	-	-	👍	-	👍	☞	☞
Polling	👎	☞	👍	👎	👎	👍	☞
TDMA	👎	👍	👍	👎	👎	👍	☞
Binary Cnt.	👍	👍	☞	👍	👍	☞	👍
Token Bus	☞	👍	👍	☞	☞	👍	👍
Token Ring	👍	👍	👍	👍	☞	☞	👍
CSMA/CD	👍	👎	👎	☞	👍	👍	👍
CSMA/CA	👍	👍	👍	👍	👍	👍	👍

Table 1: Media Access Tradeoffs (a hand-waving approach)

organizations are compromising by endorsing multiple protocols sponsored by members, resulting in standards and metastandards.

While high-level standards may ultimately yield benefits for high-level application interoperability, the compromises involved in permitting multiple physical implementations within a standard umbrella will likely impede standardization and cost reduction of actual communication hardware. For example, in the building automation industry, the Intelligent Building Institute (IBI) standard encompasses LON, CAB (Canadian Automated Building), and BACnet (Building Automation and Control Network). The MAC level of BACnet, in turn, allows the use of ARCnet (Token Bus), Ethernet (CSMA/CD), MS/TP, or a dial-up (connection oriented protocol). So, a product that conforms to the IBI standard could in fact use CSMA/CA, connection-oriented, polling, CSMA/CD, or token bus protocols at the hardware level.

In Japan, a consortium known as TRON[19] (The Real-Time Operating System Nucleus), is attempting to develop open standards for all information processing systems. They have defined the BTRON specification for business computing, CTRON for telecommunication industry, ITRON for industrial applications, and MTRON for Macro networking. In particular, the μBTRON[20], a specification based on the token ring, is aimed at networking real-time microprocessors in home, office, building, and automobile automation. However, TRON's development in the embedded arena is limited.

In Europe, several standards have been developed: Batibus in France, Profibus (MS/TP) and FND (X.25) in Germany. But their effect on the American market remains to be seen.

ATTRACTIVE OPTIONS

There are many options that would be ideal for a particular application, and one should consider all reasonable options within design and business constraints. Nonetheless, we think that based on real-time performance, cost, and hardware availability, the following

three protocols are attractive.

1. **ARCnet**: this token-bus-based protocol has transceiver chips (COM20020), a PC-based development system, and various communication peripherals (routers, repeaters, etc...) for a reasonable cost (from Standard Microsystem Corporation). In fact, SMC's new COM20051 chip integrates the COM20020 with a 8051 microcontroller.
2. **CAN:** this binary countdown based protocol is available from Intel (82527) and a Signetics/Phillips collaboration (8xC592). The 8xC592 combines the CAN protocol with the 8051. Development systems and supporting peripherals are offered by the chip vendors and other third-party consultants. The costs could drop significantly if automotive companies decide to endorse CAN based on studies they are currently performing.
3. **LON:** this CSMA/CA based protocol is a contender for the *de facto* standard in the control industry. LON interfaces are available for a variety of media and provide a large number of predefined network services in silicon. Chips are available from Motorola and Toshiba.

In selecting one of the standards in the family tree, we recommend that you give due consideration to the Media Access mechanism to avoid real-time performance problems later in the product life cycle. The MAC protocol should provide efficient use of available bandwidth, a flexible priority mechanism, bounded delays for messages, and robustness to failures.

Looking at the family tree, it is clear that a strong communication standard for embedded systems is not here yet. To the degree that differing applications have different requirements, a single hardware standard isn't possible. Furthermore, it appears that higher-level standards incorporate multiple protocols in response to political and business considerations rather than technical considerations. So, choices must be made based not only on capability, but also market share, and a prediction of the direction of standards in the future.

REFERENCES

[1]Modiri, Nasser, "The ISO Reference Model Entities," *IEEE Networks Magazine,* vol. 5, no.4, pp. 24-33, July 1991.

[2]ASHRAE 135P, *BACnet: A Data Communication Protocol for Building Automation and Control Networks*, August 1991.

[3]Jordan, Larry, *System Integration for the IBM PS/2 and PC*, Brady, New York, NY, pp. 271-282, 1990.

[4]MIL-STD-1553B, *Military Standard Digital Time Division Command/Response Multiplex Data Bus*, September 1986.

[5]MIL-STD-1773, *Military Standard Fiber Optics Mechanization of an Aircraft Internal Time Division Command/Response Multiplex Data Bus*, October 1989.

[6]Zebermann, C., Kaster, L., and Rake, H., "Profibus - Open On-site Communication," *Proceedings of the 11th Triennial World Congress of the International Federation of Automatic Control*, pp. 143-146, August 1990.

[7]Stallings, W. S., *Data and Computer Communications*, 3rd ed., Macmillan, New York, 1991.

[8]National Semiconductor Corporation, "ARINC 629 Communication Intergrated Circuit - DATAC Data Terminal IC," ver. 2.0, Military/Aerospace Products Group, March 1992.

[9]Tanenbaum, A. S., *Computer Networks*, 2nd ed., Prentice-Hall, Englewood Cliffs, NJ, 1989.

[10]Parker, Karen and Kaufman, Steve, "Fiber-Optic Links Hold Key to Fast LANs of the Future," Electronic Design, vol. 37, no. 25, pp. 46-53, December 1989.

[11]Rizzardi, Victor A., Understanding MAP, Manufacturing Automation Protocol, First Edition, 1988.

[12]Hoswell, Katherine S. and Thomas, George M., *ARCnet Factory LAN premier*, Contemporary Control Systems Inc., Second Printing, Downers Grove, Illinois, 1988.

[13]Gershon, R., Propp, D., and Propp, M., "A Token Passing Network for Powerline Communications," *IEEE Transactions on Consumer Electronics*, vol. 37, iss. 2, pp. 129-134, May 1991.

[14]Bosch, *CAN Specification*, ver. 2.0, Robert Bosch GmbH, Stuttgart, 1991.

[15]SAE J1850, *Class B Data Communication Network Interface*, July 1990.

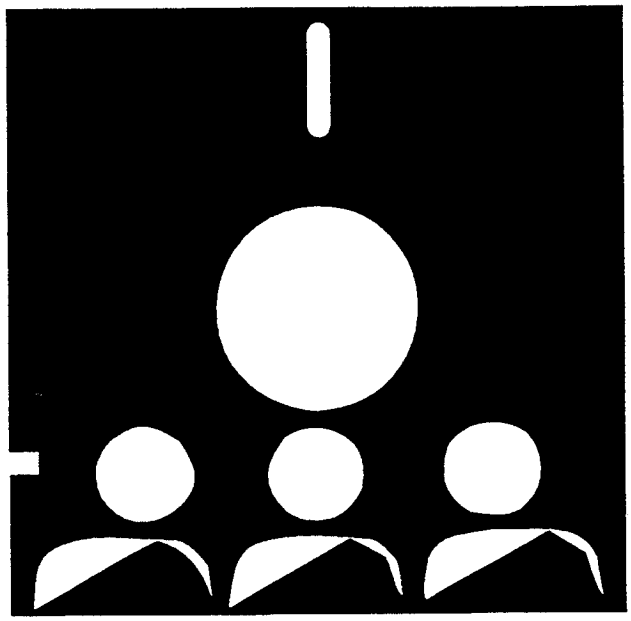
[16]Bertsekas, D., and Gallager, R., *Data Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[17]Chen and Li, "Reservation CSMA/CD: A Multiple Access Protocol for LANs," *IEEE Journal on Selected Areas in Communications*, February 1989.

[18]"Enhanced Media Access Control with Echelon's LonTalk Protocol," *LonWorks Engineering Bulletin*, Echelon Corp., August 1991.

[19] *The TRON Project 1992*, Tron Association, June 1992.

[20]Tanaka, K., Yasuyki, S., Tamai, K., Tsunoda, S., and Kato, H., "Performance Evaluation of the μBTRON Bus," *Proceedings of the Ninth TRON project Symposium*, pp. 40-45, 1992.



Proceedings

of the Fifth Annual

Embedded Systems Conference

Volume 2

Sponsored by Miller Freeman, Inc. and *Embedded Systems Programming*

ISBN # 0-87930-306-9

Miller Freeman, Inc. • 600 Harrison Street • San Francisco, CA 94107 • (415) 905-2200

EXHIBIT E

Topic Coverage

18-549 Distributed Embedded Systems

Carnegie Mellon University
Fall 2001

NOTES:

Files indicated as "(local)" have access restrictions, and can only be accessed from a CMU IP address.

Pointers to (IEEE) and (ACM) require subscriptions to on-line databases and are not free. CMU has a blanket site license; where possible pointers to freely available web material have been provided for non-CMU readers.

Pointers to the *Embedded Systems Conference* require free registration.

Some articles for Embedded Systems Programming are on-line for free. Others can be found on their back issue CD-ROM, which you can order from their web site.

"Kopetz" refers to the book: H. Kopetz, *Real-Time Systems : Design Principles for Distributed Embedded Applications*, Kluwer, 1997.

Overall Topics

The purpose of this course is to develop proficiency in the following skill areas. This includes both technical skills useful in industry as well as deeper foundations of understanding that will serve the student well in keeping up with changing technology over a span of several decades.

- **System Engineering**
  - Requirements Elicitation
  - Requirements Representation
  - Unified Modeling Language
  - Verification & Validation, including Design Reviews
  - Certification/Acceptance/Test
  - Risk Management/Schedule
- **System Architecture**
  - Modeling/Abstraction
  - Design Methodology
  - Business Issues
  - Humans as a system component
- **Embedded Systems**
  - Embedded Design Issues
  - Real-Time Scheduling
  - Time as First-Class Issue

    - Distributed Implementation
    - Performance Estimation
- **Embedded Networks**
    - Protocol Mechanisms
    - Protocol Performance
    - CAN specifics
    - TDMA specifics (TTP)
    - Embedded Internet
- **Critical Systems**
    - Reliability & Fault Tolerance
    - Basic Techniques (FMEA/FTA)
    - Appreciation of SW Safety
    - Appreciation of Network Issues
    - Validation/Certification
    - Humans as a system component
    - Ethics
    - Testing
    - Graceful degradation
- **Case Studies**
    - Elevator
    - Others

Lecture #1: Course Intro. & Foundations

Course Objective Coverage:

- *Heavy:* Setting context for course, Embedded Design Issues
- *Moderate:* Business Issues

Required Reading:

- Koopman, P., "Embedded System Design Issues -- The Rest of the Story", *Proceedings of the 1996 International Conference on Computer Design*, Austin, October 7-9 1996. (local)

Suggested Reading:

- Kopetz Chapter 1
  (Includes discussion of the time constants discussed in the lecture, which are testable material.)
- Selic & Ward, "The challenges of real time software design", *Embedded Systems Programming*, October 1996, pg. 66. (local)

Supplemental Reading about Embedded Systems:

- Embedded Systems Programming magazine
  A good general resource; many on-line articles
- Embedded Systems Conference (also has free proceedings on-line)
  A good industry-oriented conference for upgrading embedded skills. Worth registering to get free access to past conference proceedings

- Computers as Components: Principles of Embedded Computer Systems Design by Wayne Wolf.
  Excellent intro-level embedded systems book.
- Military and high-end embedded system research areas are sponsored by DARPA ITO -- dig around this web site to see the latest in that area.
- Tennenhouse, D., "Proactive Computing", Communications of the ACM, May 2000, pg. 43. (ACM) (video)
  A vision of future computing from a former head of DARPA ITO. "Let's get physical, let's get real, let's get out, let's reinvent computer science."

Supplemental Reading about Control Theory:

- Gaddy, G., "How to write a PID Algorithm", *Embedded Systems Programming*, May 1997, pp. 62-72. (local)
  Someday you're going to have to write a control algorithm. This is a good how-to guide, although it's short on math.
- Crenshaw, J., "The Math of Control Systems", *Embedded Systems Programming*, Parts I, II, III, October-December 1993. (local part 1, part 2, part 3)
  OK, here's the math.
- Morgan, D., "The PID Filter", *Embedded Systems Programming*, July 2000, pp. 126-127. (local)
  A kinder, gentler introduction to math and implementation for PIDs.

- Miller, B., "A case for fuzzy logic", *Embedded Systems Programming*, December 1995, pp. 42-70. (local)
  When the going gets tough, the tough get empirical.

Lecture #2: Intro To Project Design Process & UML

Course Objective Coverage:

- *Heavy:* Design Methodology, Unified Modeling Language
-

Required Reading:

- Douglass, "Designing Real-Time Systems with UML - I," *Embedded Systems Programming*, March 1998 (local)
- Douglass, "Designing Real-Time Systems with UML - II," *Embedded Systems Programming*, April 1998 (local)
- Rational, UML quick reference (local)

Strongly recommended reading:

- Douglass, "Designing Real-Time Systems with UML - III," *Embedded Systems Programming*, May 1998 (local)
- A more detailed UML reference summary (local)

Supplemental Reading:

- Douglass, "The Unified Modeling Language", Embedded Systems Conference, Fall 1999. (local)
- Mellor, S., "The case for using use cases", *Embedded Systems Conference*, Fall 1999. (local)
- Douglass, "UML Statecharts," *Embedded Systems Programming*, January 1999, pp. 22-42 (local)
- Full UML specification (for hard-core junkies!): OMG Original (local copy)

Recitation #1: UML Example & Project #1

Required Reading:

- Douglass, "Designing Real-Time Systems with UML - III," *Embedded Systems Programming*, May 1998 (local)

Lecture #3: Elevators

Course Objective Coverage:

- *Heavy:* Case Studies: elevators

Required Reading:

- Taub, "Elevator Technology: inspiring many everyday leaps of faith," *NY Times*, December 3, 1998, page D-12.
  (local .pdf caution -- 9.4 MB)

Supplemental Reading:

- Is 2000 feet per minute enough? (local)
  Discussion of advanced elevatoring performance techniques.
- Connecting mathematics with work and life: Scheduling Elevators (local)
  Why zoned elevators work.
- Clarkson, M., A. Sobel, T. Lehmkuhl, S. Taylor & B. Williams, "The ups and downs of formal methods" (draft), Miami University, 1999. (local)
  Has some thoughts about elevator modeling.
- Clarkson, M. & A. Sobel, "Formal methods application: an empirical tale of software development" (draft), Miami University, 1999. (local)
  Has thoughts of elevator modeling and discussion of possible dispatching methods -- could be useful for the course project.

- B-25 crash into the Empire State Building and elevator accident. (local)
- August 2000 elevator accident at World Trade Center. (CNN | local)

Lecture #4: Requirements

Course Objective Coverage:

- *Heavy:* Requirements representation, as-practiced design methods
- *Moderate:* Requirements elicitation, Business issues, embedded design issues

Required Reading:

- Wakeham, M., Requirements, the most overlooked and undervalued area of software development, March 2000 (local)
  Summary article, including chart suggesting that requirements errors are a big source of problems.
- IEEE Std. 830-1998. *IEEE Recommended Practice for Software Requirements Specification*. (IEEE)

Suggested Standards Reading:

- IEEE Std. 1223-1998. *IEEE Guide for Developing System Requirements Specifications*. (IEEE)

Suggested Reading:

- Kopetz Chapter 4
- The King's Toaster. (local)
  A little levity, obviously written by a hardware engineer...
- Tech Solutions, Software Development Standard Requirement Document, 2000. (local)
  A discussion of requirements at a high level.
- Bahill, A.T. & F. Dean, Discovering System Requirements, 1997. (local)
  Substantial discussion of the requirements process. Note that requirements methods taught in this course are a small subset of the general requirements process.

Supplemental Reading:

- Robertson, J. & S. Robertson, Volare requirements specification template. (local)
  A template for a requirements document.
- CMU software engineering design studio project report: Ballistic SRS
  SRS created according to the team software process (TSP) by a student group that serves as an example from a real development project. Their assignment was to re-engineer some of my research software.
- Brackett, J., Software Requirements: SEI Curriculum Module SEI-CM-19-1.2, January 1990.
  Sketch of topics to be taught in software requirements, with an annotated bibliography.
- *Are Your Lights On? : How to Figure Out What the Problem Really Is* by Donald C. Gause, Gerald M. Weinberg, Dorset House; ISBN: 0932633161, 1991
  This is about problem solving approaches and requirements elicitation.
- *Exploring Requirements : Quality Before Design* by Donald C. Gause, Gerald M. Weinberg, Dorset House; ISBN: 0932633137, 1999
  A discussion with examples about how to get requirements right in the first place.

Lecture #5: Distributed Solutions

Course Objective Coverage:

- *Heavy:* Distributed Implementation
- *Moderate:* Design Methodology

Required Reading:

- Kvaser AB, "What is a distributed embedded control system", (local)
  A brief look at centralized vs. embedded tradeoffs.
- Ganssle, "Keep it small", *Embedded Systems Programming*, September 1998. (local)
  Argues that breaking a system into small pieces lowers total design complexity.

Suggested Reading:

- Kopetz Chapter 2
- Kassakian, J., H. Wolf, J. Miller & C. Hurton, "Automotive electrical systems circa 2005", IEEE Spectrum, August 1996, pp. 22-27. (IEEE) (local)
  Discusses potential evolution of automobiles to 42V DC instead of 12V DC; this will be used as a case study in the lecture.
- Selic, B., "Distributed software design: challenges and solutions," *Embedded Systems Conference*, Fall 2000. (local)
  Good summary of the types of issues one sees in distributed systems, most of which are covered in various lectures of this course, such as fault management, communication latency, and distributed agreement.

Lectures #6 & 7:

TBD

Lecture #8: Review for Test #1

TBD

Test #1

IMPORTANT NOTE!
Due to the schedule disruption in class and general lack of preparation time caused by the instructor's family emergency, the coverage of Test #1 will be limited to being able to go through the UML-based design process used for the project and covered in Lecture #6 up to and including requirements and traceability. The below information is so that you can understand the important points you should have gotten from the various lectures (and that may still be relevant to answering questions in later tests, but not directly tested).

All the information in the lectures, recitations, project assignments, and homeworks is testable in general. However, the emphasis is on ability to use the knowledge, not rote memorization of trivia. So, you should be able to recall the meaning of key equations, the definitions of key terms, the general use of UML diagram notation, and so on, which shouldn't be a problem if you've been doing the assignments. Beyond that, the emphasis is on skills listed below. If you're missing a small piece ask us during the test and we'll figure out a reasonable way to get you unstuck, perhaps at a cost in points on the question if we have to give you a major hint (but we'll tell you if it is going to cost before giving you the info -- usually small hints are free). Be careful, because if you study *only* the narrow interpretation of the things below you will come up short. The below are likely to be the basis for creating a testable question, but getting from the question to the equation/whatever will usually require understanding the less tangible information in the lectures as well to provide "common sense" information for problem solving (*i.e.*, plug-and-chug equation solving usually won't do it alone!).

- Lecture #1: Course Intro & Foundations:
  - Know the definitions for the various control loop parameters and important rules of thumb; be able to use them in computations.
  - Know typical real-time embedded characteristics and be able to relate them to an application domain.
  - Be able to model a system in terms of dividing components into the categories: computer, user interface, I/O, operator, and plant.
- Lecture #2: Intro To Project Design Process & UML
  - Given a summary of UML notation for reference, be able to represent portions of the design of an arbitrary system in terms of the UML diagrams, textual notation, and other aspects in a manner similar to Projects #1 and #2.
  - Given an example system that is familiar but not previously discussed in the course, create a design for that system potentially including the steps of: UML class diagram, key use cases, scenarios, sequence diagrams, and software requirements specifications.
- Lecture #3: Elevators
  - Given an elevator example, be able to use that in a design question without requiring detailed explanations -- *i.e.*, you should have a reasonable amount of elevator domain knowledge and be familiar with the project materials with respect to elevators to access information in them reasonably quickly.
- Lecture #4: Requirements
  - Given sequence diagrams, create detailed requirements according to the course requirements template.
- Lecture #5:
  - Given a typical embedded system situation, be able to make reasonable engineering tradeoffs involving strengths/weaknesses of a distributed approach.
- Lecture #6:
  - Be able to take an arbitrary system other than an elevator through the same process as in Project #2. Have a general appreciation for what happens after that in the project (but you are not expected to be able to do those steps yourself on a test until you've done them in the project in later weeks of the course).

Lecture #9: Time

Course Objective Coverage:

- *Heavy:* Time as First-Class Issue
- *Moderate:* Distributed Implementation

Required Reading:

- Hinerman, "Pardon Me, Do You Have the Time?", *Embedded Systems Programming*, August 2000. (local)
- Kopetz Section 3.2
  I was unable to find a reasonable non-book publication with the material we need to cover. A copy of Kopetz is on reserve via the library if you did not buy one.
  The really important pages to read are pp. 52-55, which you can find here: (local of 4 pages).

Suggested Reading:

- Kopetz Chapter 3
  We're covering the majority of this chapter in the lecture; if it is not crystal clear from the lecture then the book should help.

Supplemental Reading:

- A brief discussion of time standards.
- A minor horror story with follow-up.

Lecture #10: Embedded Communications

Course Objective Coverage:

- *Heavy:* Embedded Network overview, Protocol Mechanisms

Required Reading:

- Upender & Koopman, Communication Protocols for Embedded Systems, Embedded Systems Programming, November 1994.
  This is a summary of much of the lecture material.

Suggested Reading:

- Kopetz Chapter 7
- Canosa, "Networking Protocols for Consumer Internet Appliances", Questra Corp, *Fall 1999 Embedded Systems Conference*. (local)
  Survey with more of a slant to consumer electronics rather than real time control.

Supplemental Reading: (most of this is material beyond topics we'll cover in class)

- Canosa, "Fundamentals of Firewire", Questra Corp., *Spring 1999 Embedded Systems Conference*. (local)
  A look at a protocol used for consumer video and high-bandwidth applications.

- Canoso, USB basics, *Embedded Systems Programming*, June-July 1997 (local: part 1; part 2)
  Universal Serial Bus -- mostly for I/O to desktop PCs; used with consumer product embedded systems.
- Flynn, Understanding and using the I2C Bus, *Embedded Systems Programming*, November 1997. (local)
  A look at a low-end embedded network, mostly for use over very short distances (e.g., a single circuit board).
- Digital Output techniques (local)
  Shows circuit techniques for driving loads, several of which are applicable to embedded network drivers as well.
- Echelon, Introduction to the LonWorks System, 1999. (local)
  This is a high-level look at the Echelon LonTalk protocol used in many embedded systems.
- Train control network, TCN Tutorial

Lecture #11: CAN Protocol

Course Objective Coverage:

- *Heavy:* CAN specifics
- *Moderate:* Protocol Mechanisms

Required Reading:

- Schill, Overview of the CAN Protocol, *Embedded Systems Programming*, September 1997. (local)

Suggested Reading:

- Bosch, CAN Specification, Version 2, 1991. (local)
  This is the complete CAN specification. (note: "Robert Bosch GmbH" as an author is a German Corporation, not a person).

Supplemental Reading:

- Zeltwanger, "CANopen for embedded networking", *Embedded Systems Conference*, Fall 1999. (local)
  CANopen is an upper-level protocol stack built on top of CAN.

Lecture #12: Protocol building blocks

Course Objective Coverage:

- *Heavy:* Protocol Mechanisms

Required Reading:

- Review Upender & Koopman, Communication Protocols for Embedded Systems, Embedded Systems Programming, November 1994.

Lecture #13: Protocol performance analysis

Course Objective Coverage:

- *Heavy:* Performance Estimation, Protocol Performance,
- *Moderate:* Real-Time Scheduling, Protocol Mechanisms

Required Reading:

- Dean & Upender, "Embedded Communication Network Pitfalls", *Embedded Systems Programming*, September 1997. (local)
  A summary of issues such as messagel clumping.

Suggested Reading:

- Koopman, Time Division Multiple Access Without a Bus Master (JTDMA), 1995 technical report.
  (a robust, master-less communication protocol)

Supplemental Reading:

- Tindell, "Calculating Control Area Network (CAN) message response times", 1994. (local)
  Believe it or not, can opponents say it is impossible to predict maximum message response time in CAN; this article show that it is in fact possible to analytically bound the worst case.
- Echelon, *Enhanced media access control with LonTalk protocol*, 1995. (local)
  BEWARE: the performance graphs are for uniformly distributed traffic; LonTalk pretty much breaks when it gets synchronized messages.
- Echelon, *Determinism in industrial computer control network applications*, 1995. (local)
  Talks in general terms about workarounds possible when using a collision-based protocol.

Lecture #14: TTP protocol

Course Objective Coverage:

- *Heavy:* TDMA specifics (TTP)
- *Moderate:* Real-Time Scheduling, Time as First-Class Issue, Distributed Implementation

Required Reading:

- Bannatyne, R., Time Triggered Protocol: TTP/C, *Embedded Systems Programming*, March 1999. (local)

Suggested Reading:

- Kopetz Chapter 8
  (Note: Kopetz invented TTP, so he is the authoritative source for TTP info)
- Kopetz, H., Holzmann, M. & Elmeureich, W., "A Low-Cost Time-Triggered Sensor Network: TTP/A", TU Vienna, July 1999. (local)

Supplemental Reading:

- Hedenetz, B. & R. Belschner, Brake-by-wire without Mechanical Backup by Using a TTP-Communication Network, SAE #981109, 1998. (local)
- Clarke, P., "Car safety-bus pick may fuel race to silicon," *EE Times*, October 2, 2000. (local)
  Audi has selected TTP for use on the 2004 or 2005 model year... (but see next article)
- Murray, C. & Clarke, P., "Auto giants launch rival safety bus standard," *EE Times*, October 13, 2000. (local)
  ... but FlexRay has announced everyone picked them instead.

TTP Vs. FlexRay http://www.eetimes.com/story/OEG20010927S0080

Lecture #15: Distributed scheduling & I/O

Course Objective Coverage:

- *Heavy:* Real-Time Scheduling
- *Moderate:* Distributed Implementation

Required Reading:

- Tindell, K., "Deadline Monotonic Analysis", *Embedded Systems Programming*, June 2000. (local)

Suggested Reading:

- Kopetz Chapters 10, 11

Supplemental Reading:

- Dibble, P., "Real Time Implementation Techniques," *Embedded Systems Programming*, August 1995. (local)

- Livani, M. & Kaiser, J., "EDF consensus on CAN bus access for dynamic real-time applications", *IPPS/SPDP'98 Workshops Held in conjunction with the 12th International Parallel Processing Symposium and 9th Symposium on Parallel and Distributed Processing*, 1998. (local)
- Ready, J. & D. Barnett, "Tradeoffs drive embedded OS choice in communications designs", *Electronic Design*, May 31, 1999, pp. 38-48. (local)
- Lemieux, The OSEK/VDX standard, *Embedded Systems Programming*, March 2000 (local)
  This is a contender for an automotive operating system standard.
- Bruyer, D., "Sizing throughput requirements on real time systems", *Embedded Systems Programming*, Sept. 1999 (local).
  Another look at the timing problems caused by interrupt service routines and caches.

- Toeppe S. & Ranville, S., "Commercial RTOSes for Automotive Applications", *Embedded Systems Programming*, July 2000, pp. 108-123. (local)
  A look at RTOS & interrupt service overhead issues from an automotive perspective.
- Dean, A., Shen, J. P. "Techniques for Software Thread Integration in Real-Time Embedded Systems," *Real-Time Systems Symposium*, Madrid, Spain, December 2-4, 1998. (local)
  Discusses using a tool to automatically insert software I/O routines as a "guest" thread with other software in superscalar code to get polled I/O "for free".

Test #2

All the information in the lectures, recitations, project assignments, and homeworks is testable in general. However, the emphasis is on ability to use the knowledge, not rote memorization of trivia. So, you should be able to recall the meaning of key equations, the definitions of key terms, the general use of UML diagram notation, and so on, which shouldn't be a problem if you've been doing the assignments. Beyond that, the emphasis is on skills listed below. If you're missing a small piece ask us during the test and we'll figure out a reasonable way to get you unstuck, perhaps at a cost in points on the question if we have to give you a major hint (but we'll tell you if it is going to cost before giving you the info -- usually small hints are free). Be careful, because if you study *only* the narrow interpretation of the things below you will come up short. The below are likely to be the basis for creating a testable question, but getting from the question to the equation/whatever will usually require understanding the less tangible information in the lectures as well to provide "common sense" information for problem solving (*i.e.*, plug-and-chug equation solving usually won't do it alone!).

- Lecture #9: Distributed Time
  - Given a networked message scenario, be able to work out relative message timing in terms of micro-tick and macro-tick values.
  - Understand and be able to apply concepts of offset, precision, accuracy, and correction schemes, including numerical calculations.
  - Understand and apply concepts of network propagation delay (*e.g.*, applications of $2t_{pd}$)
  - Be able to reason through problems involving distributed events and network message ordering
- Lecture #10 & #12: Protocol operation & building blocks
  - Know how the following protocols work, including underlying protocol mechanisms: M/S polling, TDMA, CSMA, CSMA/CD, Token Bus, CSMA/CA, Reservation CSMA, Binary Countdown, jam-based/consensus signal-based synchronization
  - Know the tradeoffs in terms of priority, determinacy in obtaining the bus, hardware tradeoffs (interface, timers), fairness
  - Be able to combine mechanisms to create a custom protocol and understand specific tradeoffs made in the process
  - Be able to use one protocol to emulate another
- Lecture #13: Protocol performance analysis
  - Be able to analyze the performance of a workload including periodic and sporadic messages for an arbitrary protocol
  - Be able to compute efficiency, and latency for an arbitrary protocol and identify performance problems
- Lecture #11: CAN Protocol
  - Understand how bit stuffing works

    - Given a DeviceNet scheme, create an appropriate message dictionary and conduct a performance analysis; be able to create or work with a custom message ID structure different than DeviceNet
    - Understand how message filters work and would be accounted for when creating a message dictionary
- Lecture #14: TTP protocol
    - Understand operation of TTP/A and TTP/C protocols
    - Be able to create a message event description list for an arbitrary workload description
- Lecture #15: Distributed scheduling & I/O
    - Be able to create a conflict-free static schedule and compute latency for an arbitrary workload (given computing latencies at each node) for a time-triggered system.
    - Be able to create a set of message priorities and determine schedulability using rate or deadline monotonic scheduling for a time-triggered system.
    - Understand and be able to work with the various sources of delay and jitter on the path taken from a sensor on one node to an actuator on a different node

---

Lecture #16: Dependability for Embedded Systems

Course Objective Coverage:

- *Heavy:* Reliability & Fault Tolerance

Required Reading:

- Nelson, V., "Fault-tolerant computing: fundamental concepts", *IEEE Computer*, July 1990. (IEEE) (local)

Standards:

- IEEE Std. 1413-1998, *IEEE Standard Methodology for Reliability Prediction and Assessment for Electronic Systems and Equipment*. (IEEE)
  A rather skimpy treatment at the high level, but it is at least a standard.

Suggested Reading:

- Kopetz Chapter 6
- Blanchard et al., *Systems Engineering and Analysis*, 1990, chapter 13 on reliability. (local)
- Siewiorek, D., "Fault tolerance in commercial computers", *IEEE Computer*, July 1990. (IEEE) (local)
- Punches, K., "Design for Reliability: a checklist", *EDN*, November 21, 1996. (local)
  Gives a good intro-level overview to electronic reliability calculations and methods.
- Meyer, B., "Every little bit counts: toware more reliable software", *IEEE Computer*, November 1999. (IEEE) (local)
  A good synopsis of current techniques for reliable software.

Supplemental Reading:

- Nasa's design for reliability course: http://osat.grc.nasa.gov/dfr/dfr.htm

- Lakshmin, F., "Minimizing failures in electronic systems by design", *EDN*, August 3, 2000. (local)
  Hardware failure mode discussion with some nice pictures.
- Bertino, E.; Martino, L., "Reliability modeling: an overview for system designers", *IEEE Computer*, April 1991. (local)

Draft below this point -- subject to change

Back to course home page.

EXHIBIT F

9
Control Area Network (CAN)

18-540 Distributed Embedded Systems

Philip Koopman

October 4, 2000

Required Reading: **Schill, Overview of the CAN Protocol**

Most of the pictures in the lecture are from:

- **CAN specification (Bosch)**
- **Overview to Can; Infineon**
- **DeviceNet materials -- http://www.odva.org/**

Carnegie Mellon

Assignments

- **By next class read about Protocol building Blocks:**
  - Review protocol survey paper from this week
    (I haven't found papers that directly address this topic)

- **Dates to remember:**
  - Project Part #3: due in one week on Wednesday 10/11
  - HW #5 due at 4 PM on Friday 10/13

Where Are We Now?

- **Where we've been:**
  - Protocol Overview

- **Where we're going today:**
  - CAN -- an important embedded protocol
  - Primarily automotive, but used in many places

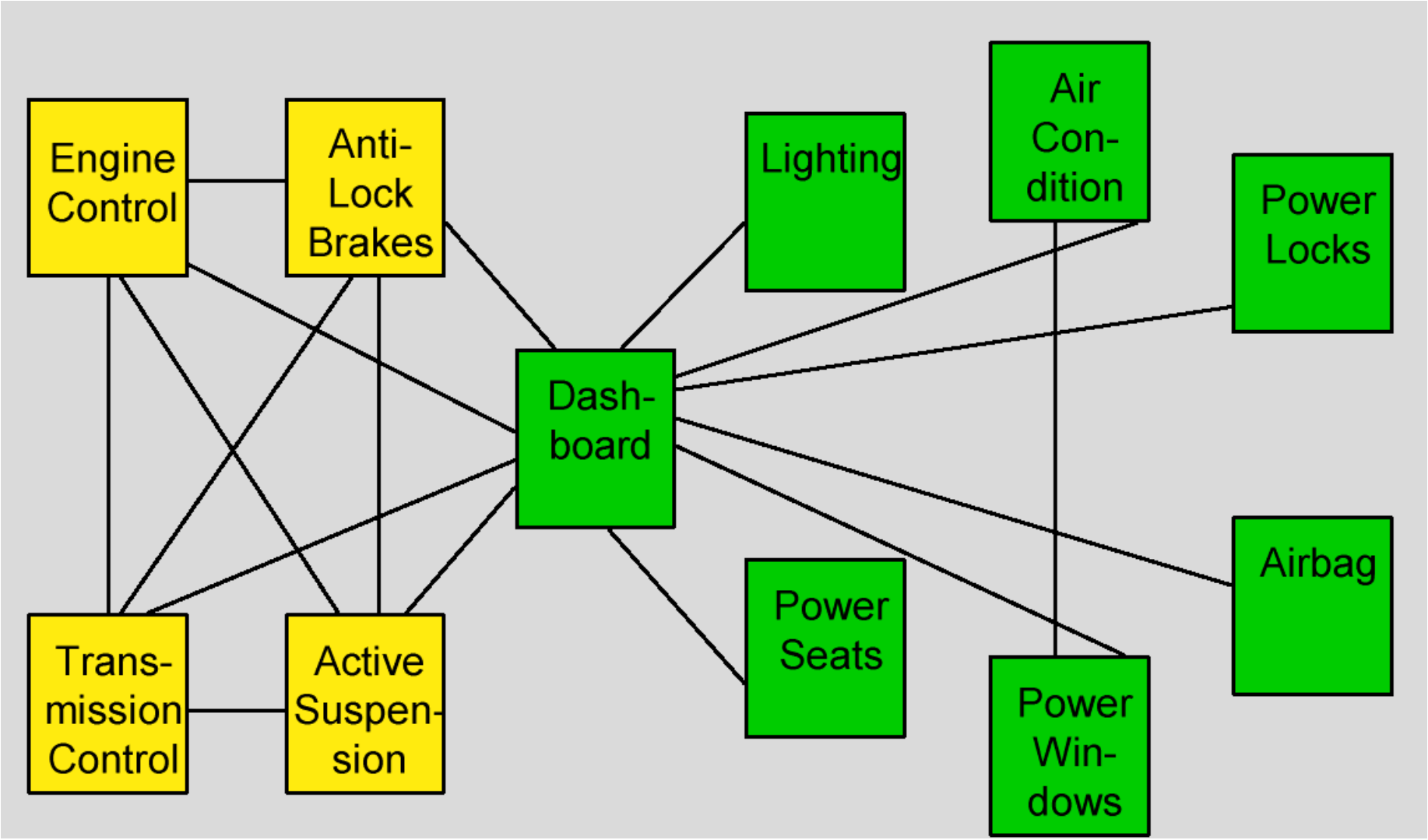
- **Where we're going next:**
  - A building-block approach to protocols:
    - Custom protocols
    - Protocol performance analysis

Preview

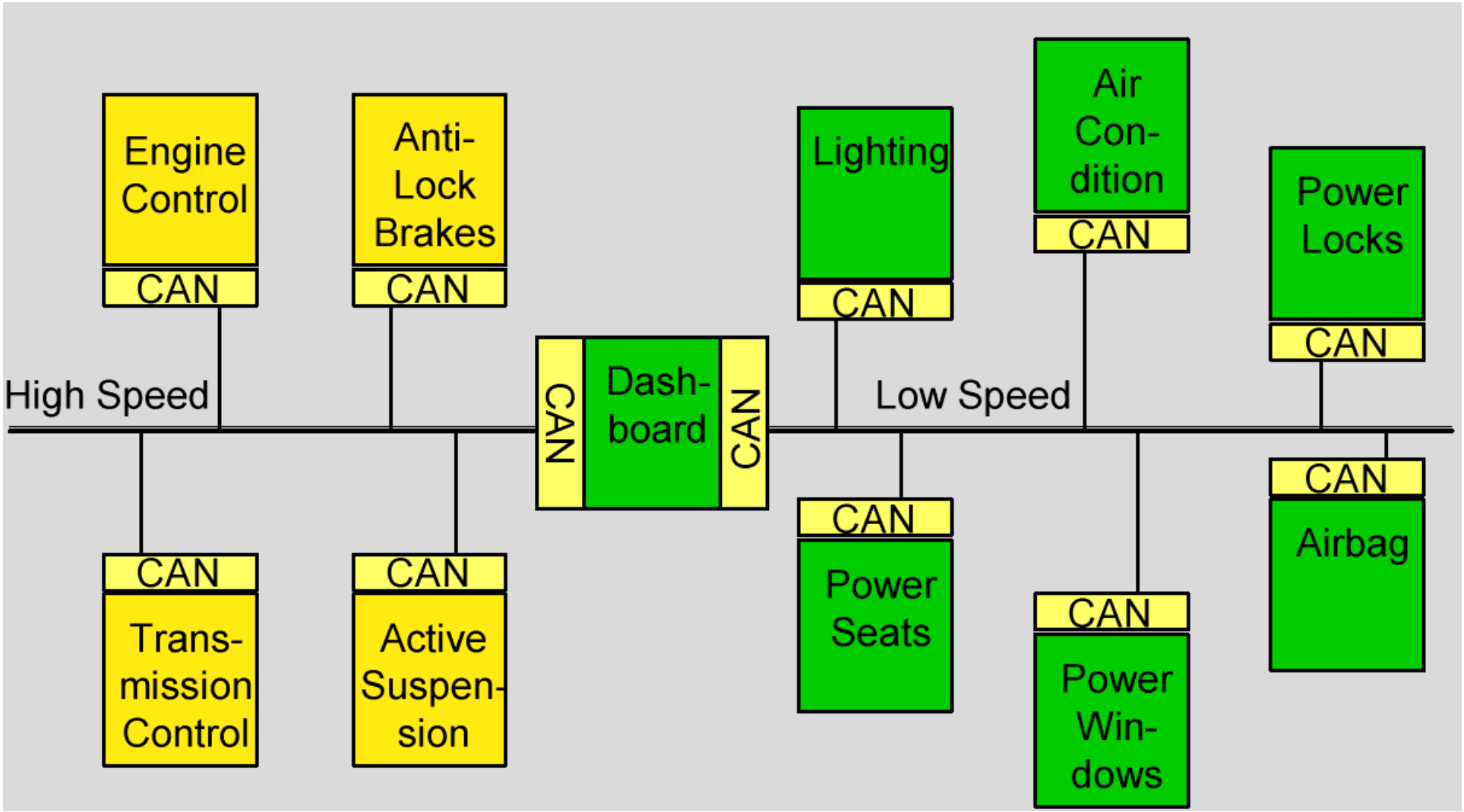
- **CAN – important automotive protocol**
  - Physical layer
  - Protocol layer
  - Message filtering layer (with add-on protocols)

- **Keep an eye out for:**
  - Message prioritization
  - How “small” nodes can be kept from overloading with received messages
  - Tradeoffs

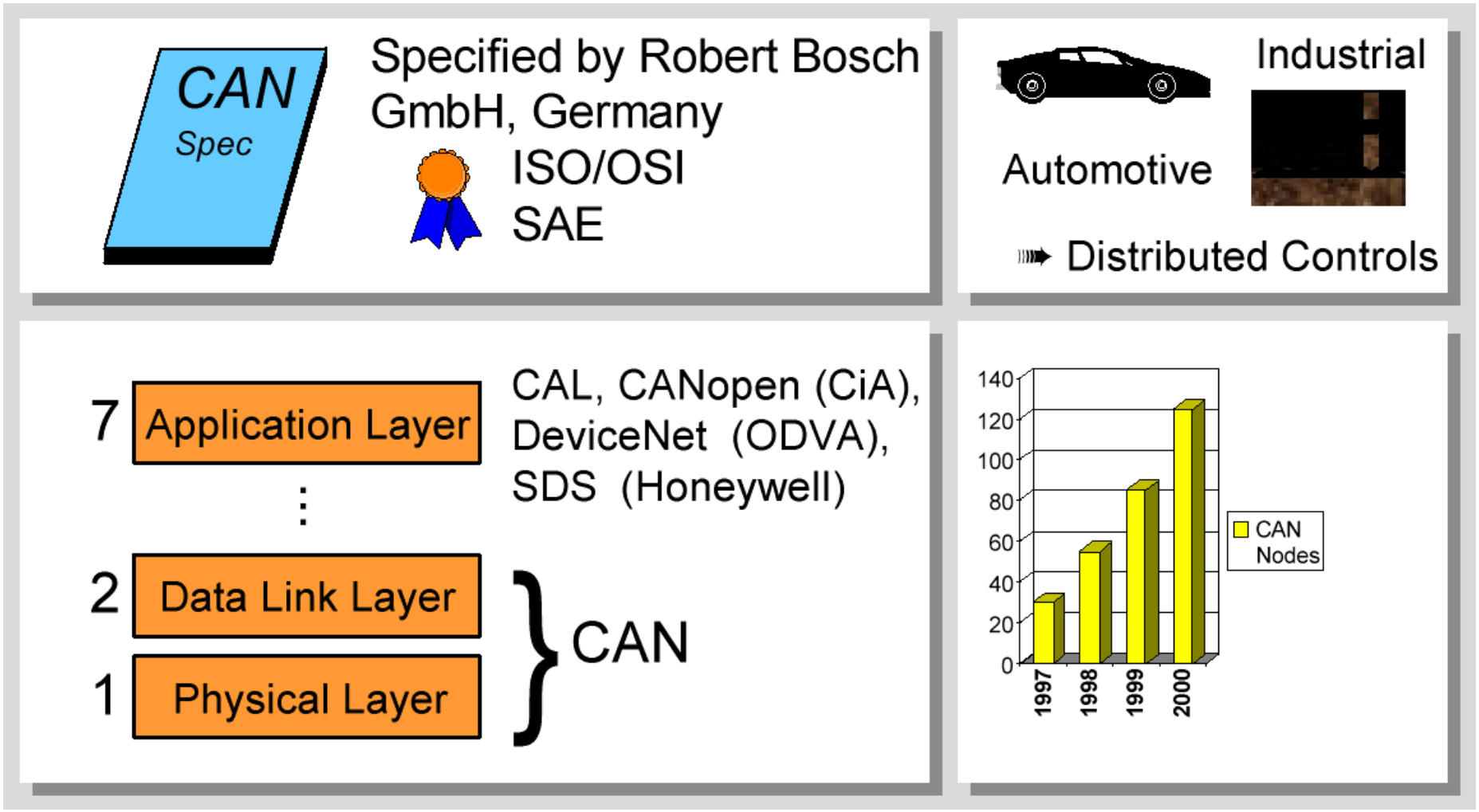
Before CAN


Engine Control
Anti-Lock Brakes
Lighting
Air Con-dition
Power Locks
Dash-board
Airbag
Power Seats
Power Win-dows
Trans-mission Control
Active Suspen-sion

With CAN

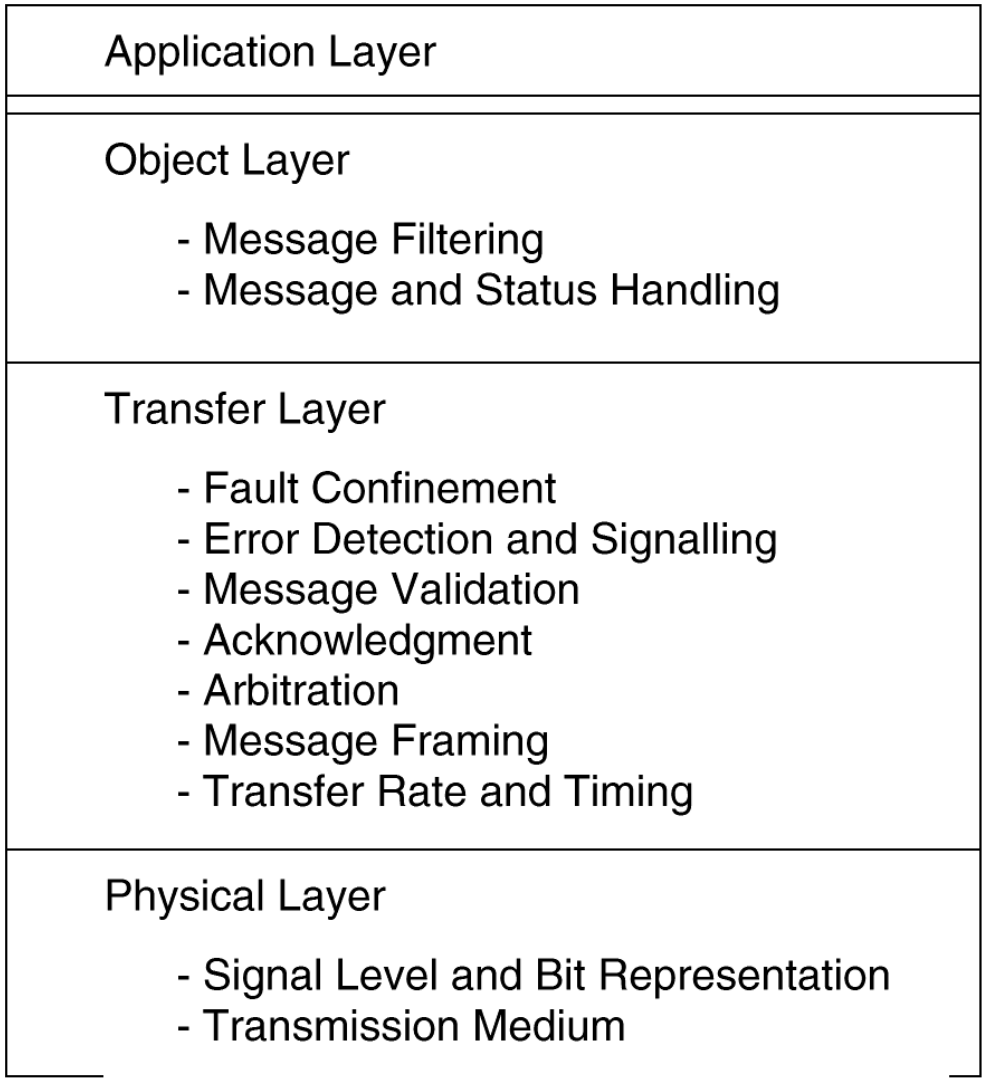

Engine Control
CAN
Anti-Lock Brakes
CAN
Lighting
CAN
Air Con-dition
CAN
Power Locks
CAN
High Speed
CAN
Dash-board
CAN
Low Speed
CAN
Trans-mission Control
CAN
Active Suspen-sion
CAN
Power Seats
CAN
Power Win-dows
CAN
Airbag

Generic CAN Propaganda Slide



CAN & the Protocol Layers

- **CAN only standardizes the lower layers**
- **Other high-level protocols are used for application layer**
  - User defined
  - Other standards

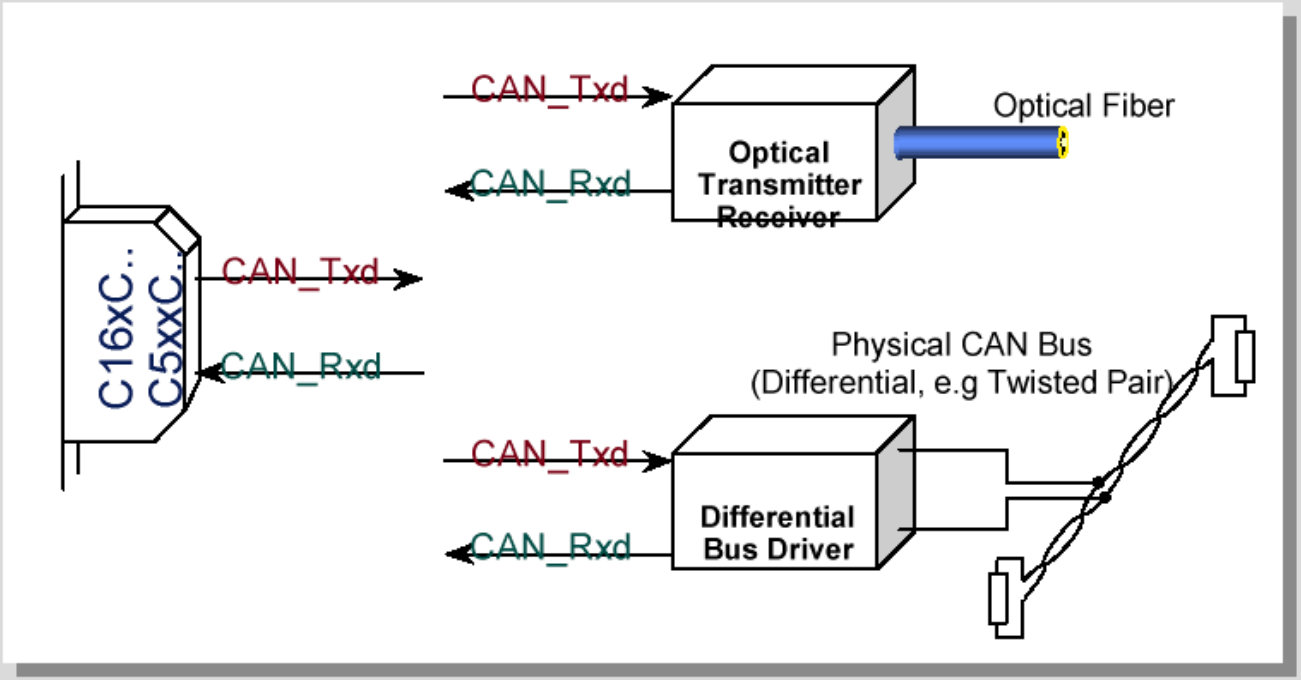


Physical Layer Possibilities

- **MUST support bit dominance (discussed later)**
- **Specifically rules out transformer coupling for high-noise applications**
  - But, cars are high-noise, right????
  - Differential drive and optical fibers help in most cases, but not all

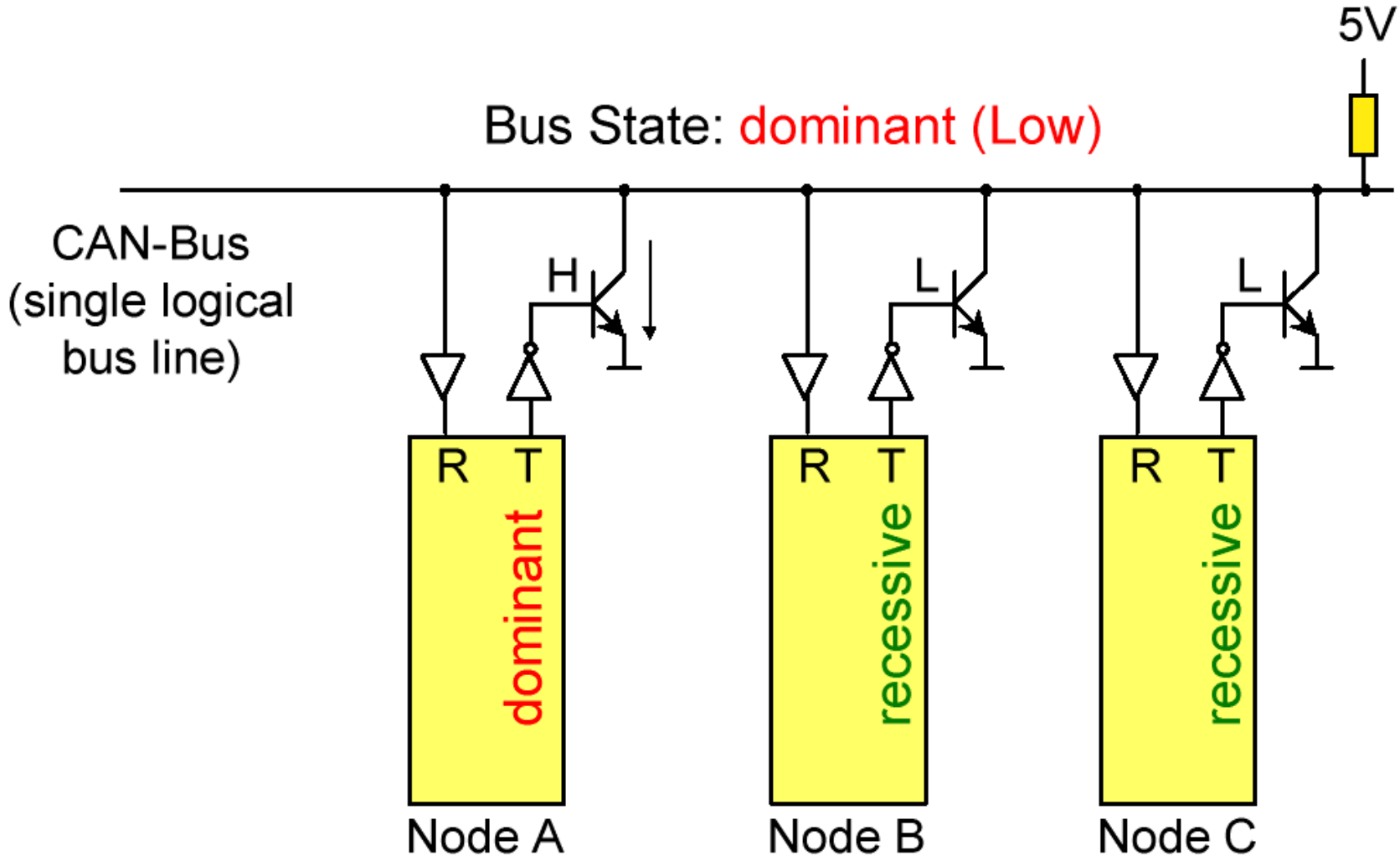
❑ **Usual ISO Physical Layer :-**

- **Bus wires twisted pair, 120R Termination at each end**
- **2 wires driven with differential signal (CAN_H, CAN_L)**

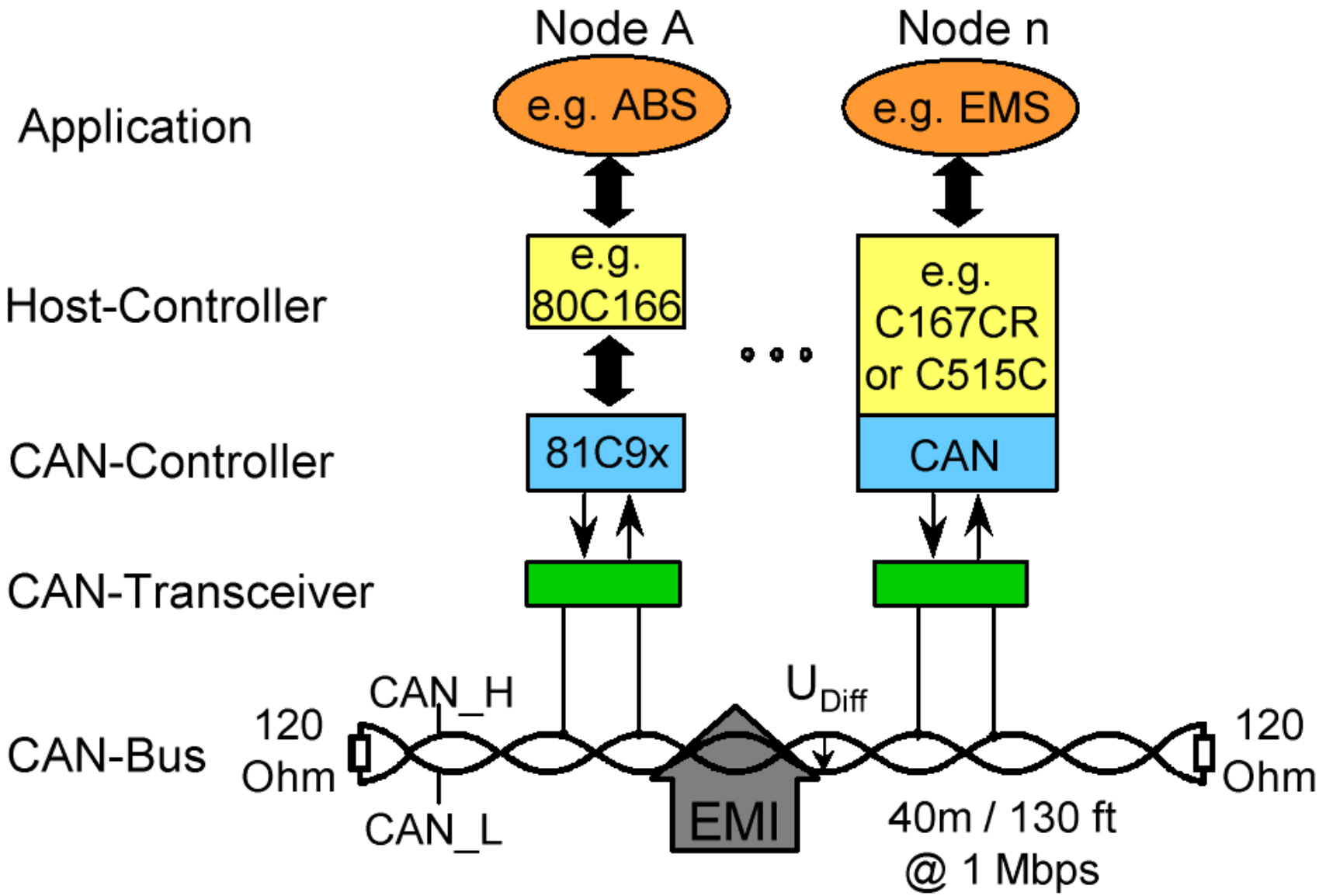


Bit Dominance

- **Wired "Or" design**
  - (Called "open collector logic" before TTL/tristate was invented…)



Generic CAN Network Implementation



Basic Bit Encoding - NRZ

- **NRZ = Non-Return-To_Zero**
  - Fewer transitions (on average) = less EMI, but requires less oscillator drift

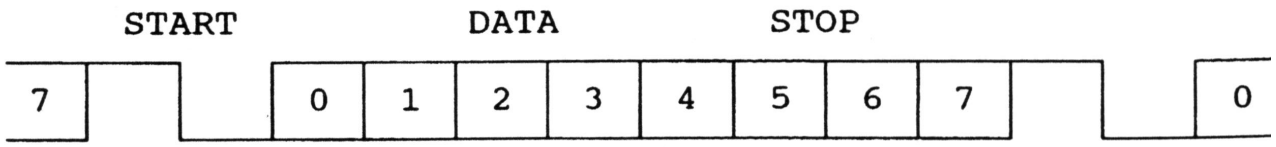


FIGURE 26.21 A 10-bit NRZ waveform (LSB first).

  - Bit stuffing relaxes oscillator drift requirements

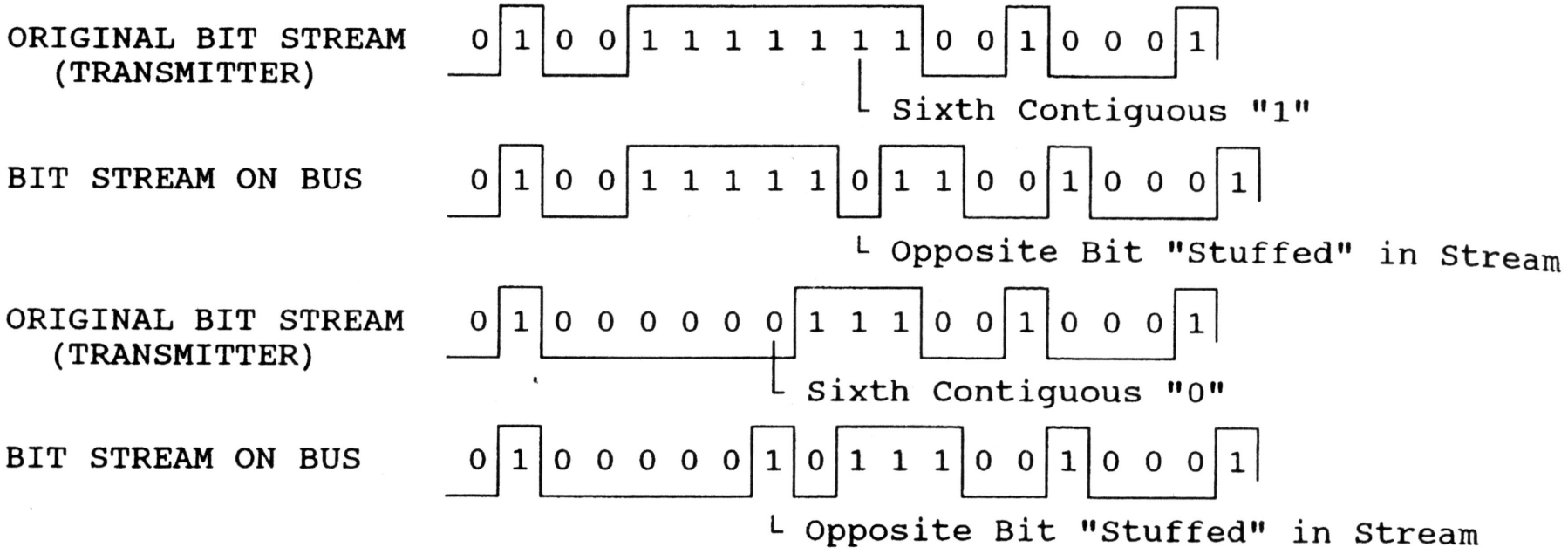
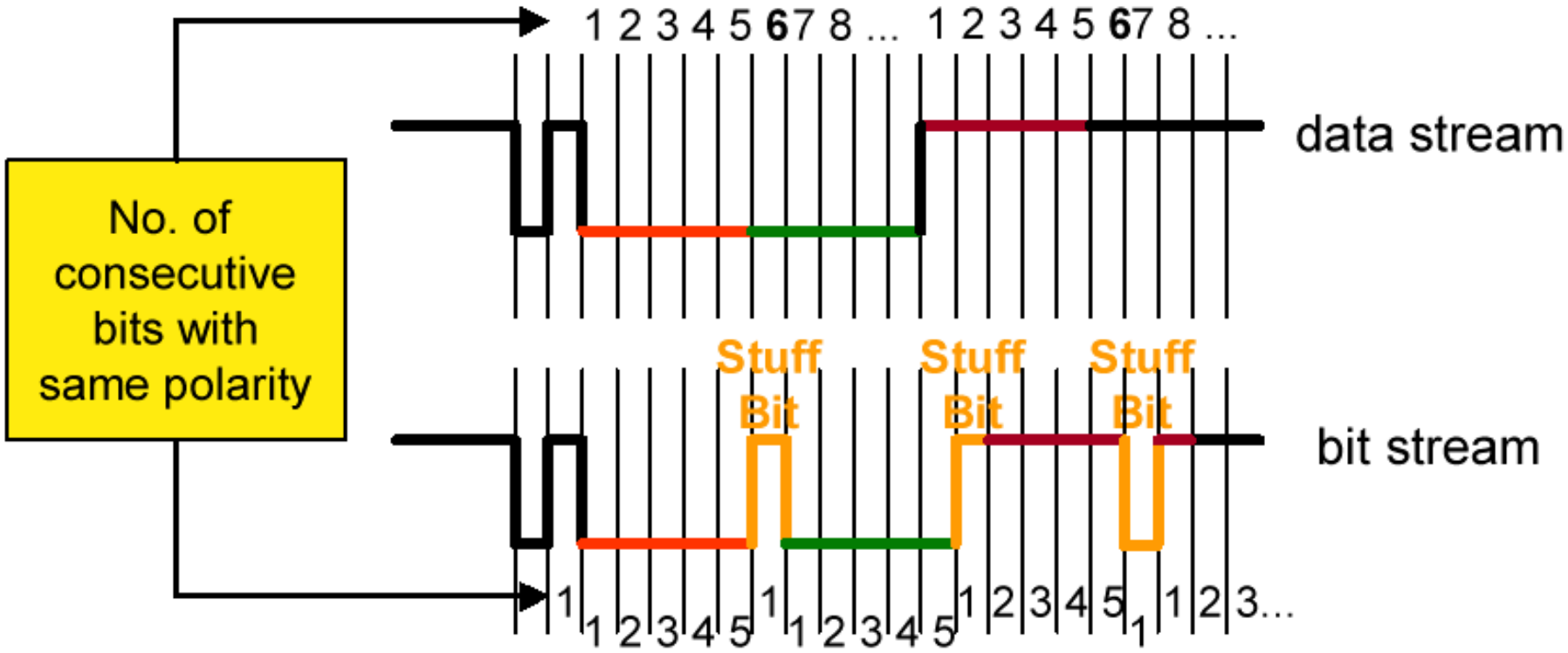


FIGURE 26.23 NRZ bit-stuffing.

Another Look at Bit Stuffing

- **Five identical bits in a row triggers an inverted Stuff Bit**
  - Bit de-stuffer must take it back out on the receiving end…
  - ***[This picture is slightly wrong -- it is 5 bits in source stream, not counting stuff bits…]***



Generic Message Format

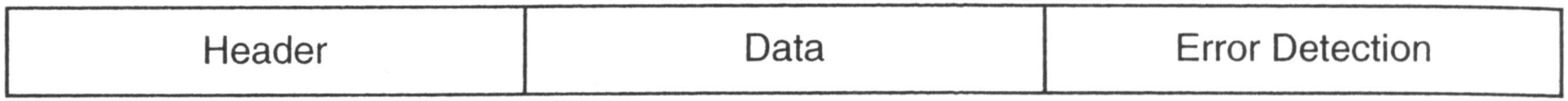
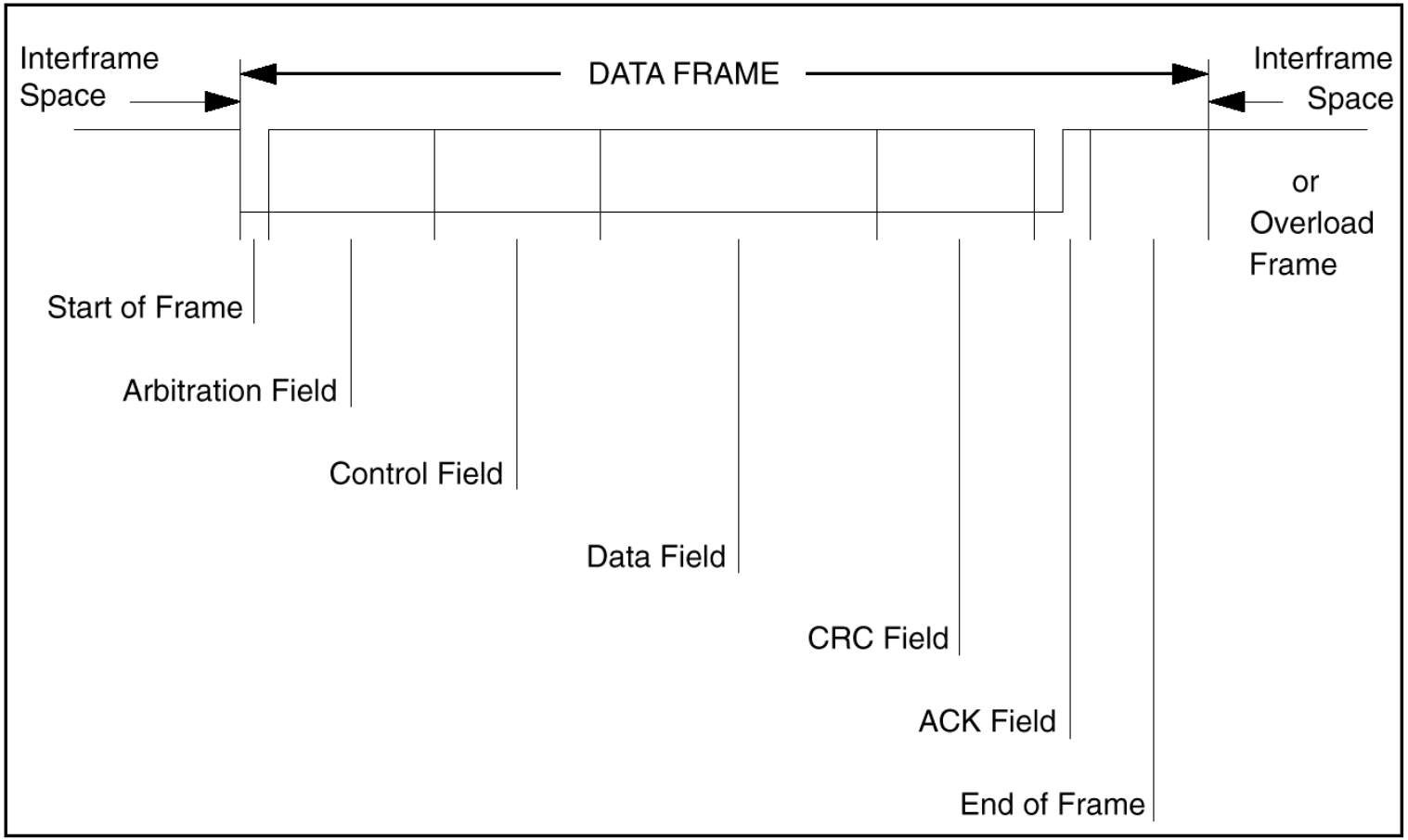


FIGURE 26.1 Three parts of a vehicle network frame or message.

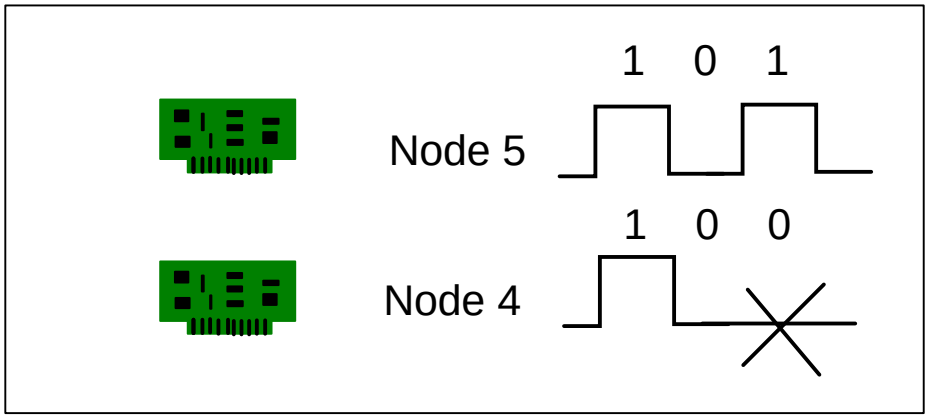
- **Header**
  - Routing information (source, destination)
  - Global priority information (which message gets on bus first?)
- **Data**
  - Application- or high-level-standard defined data fields
  - Often only 1-8 bytes
- **Error detection**
  - Detects corrupted data (e.g., using a CRC)
  - Embedded networks can have *very* high bit error rates

CAN Message Format

- **What's inside the message?**
  - "Arbitration Field" = "Message ID"



Binary Countdown (Bit Dominance)



- **Operation**
  - Each node is assigned a unique identification number
  - All nodes wishing to transmit compete for the channel by transmitting a binary signal based on their identification value
  - A node drops out the competition if it detects a dominant state while transmitting a passive state
  - Thus, the node with the highest identification value wins

- **Examples**
  - CAN, SAE J1850

More Detailed Arbitration Example

Two logic states possible on the bus:
"1" = recessive
"0" = dominant

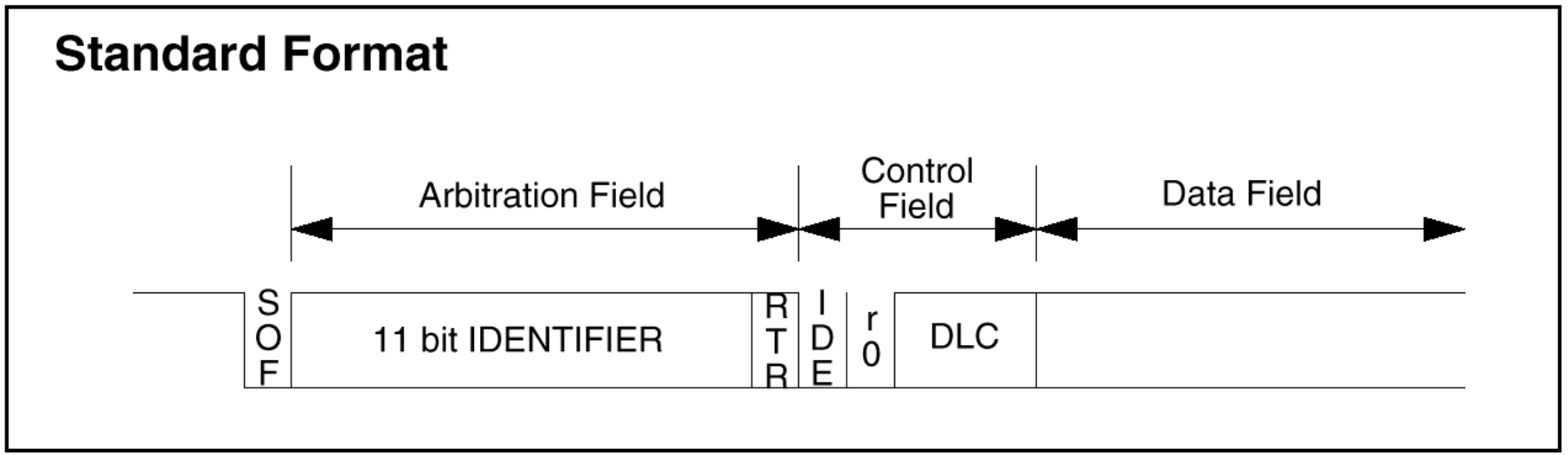
"1" = recessive
"0" = dominant

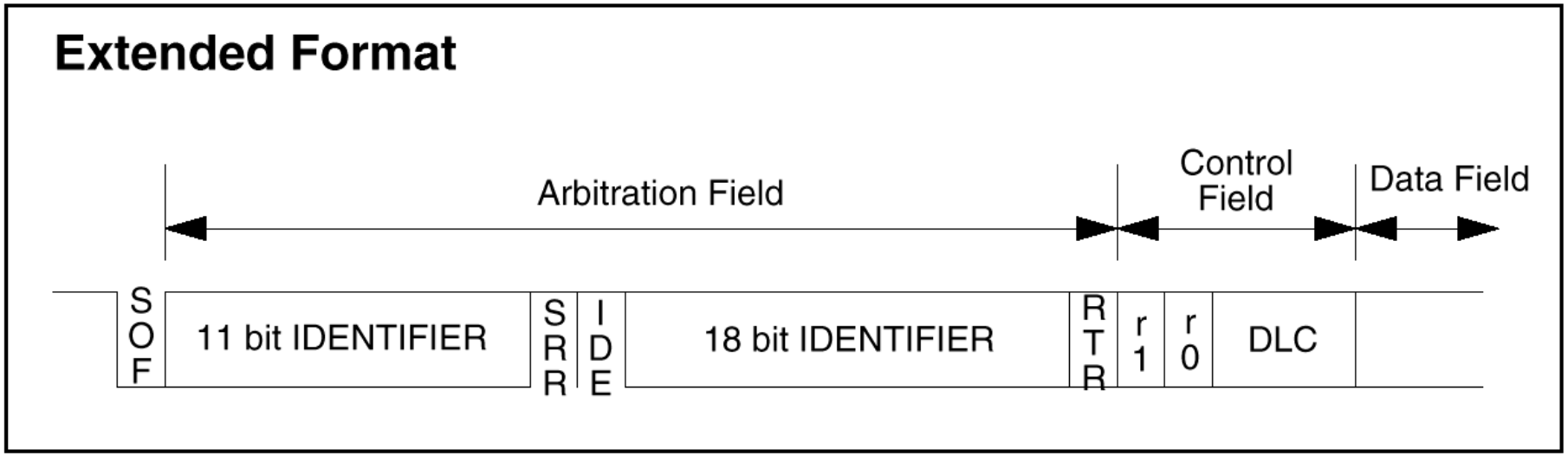
A	B	C	BUS
D	D	D	D
D	D	R	D
D	R	D	D
D	R	R	D
R	D	D	D
R	D	R	D
R	R	D	D
R	R	R	R

As soon as one node nodes transmits a dominant bit (zero):
Bus is in the dominant state.

Only if all nodes transmit recessive bits (ones):
Bus is in the recessive state.

Two Sizes of CAN Arbitration Fields

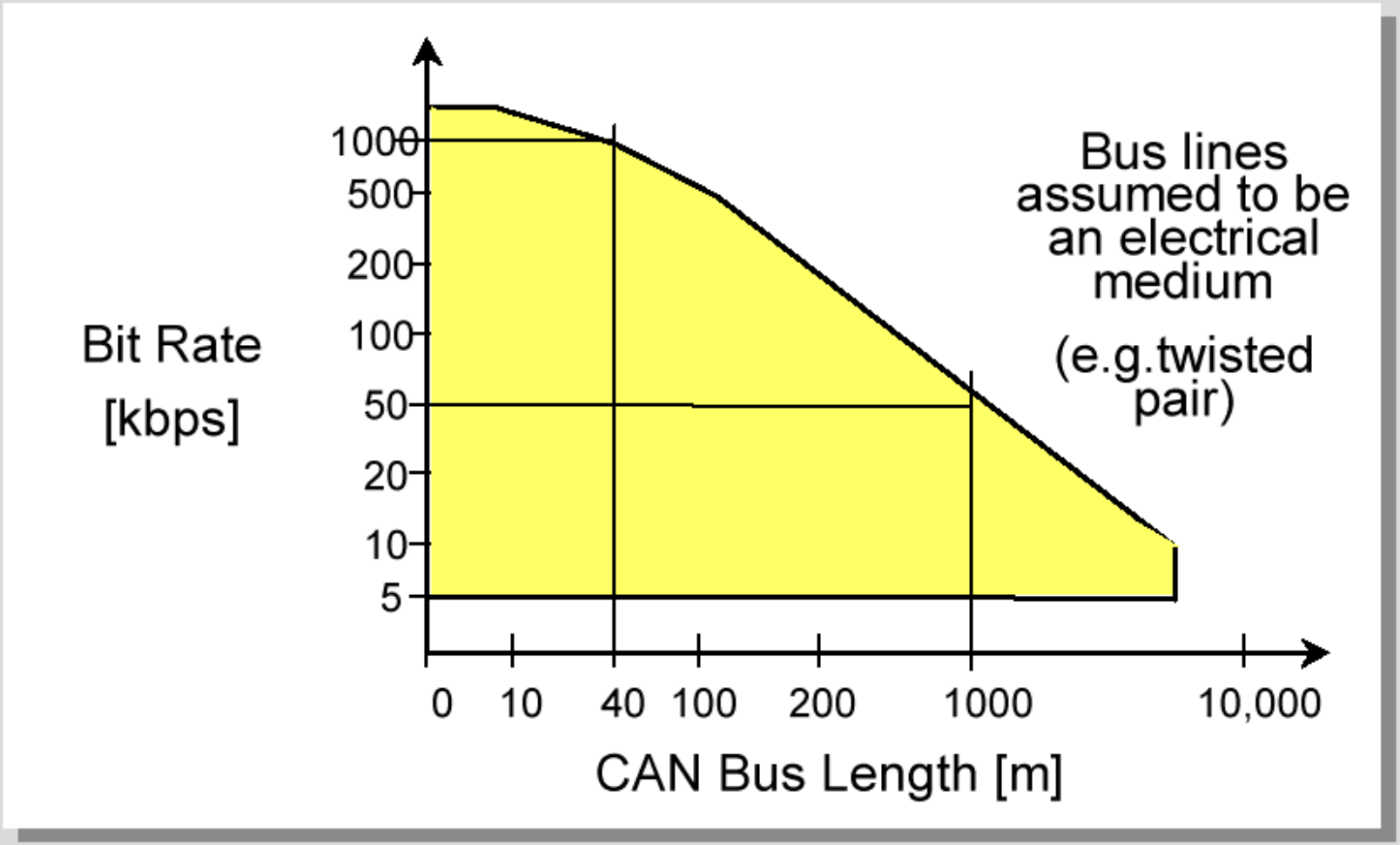




Arbitration Limits Network Size

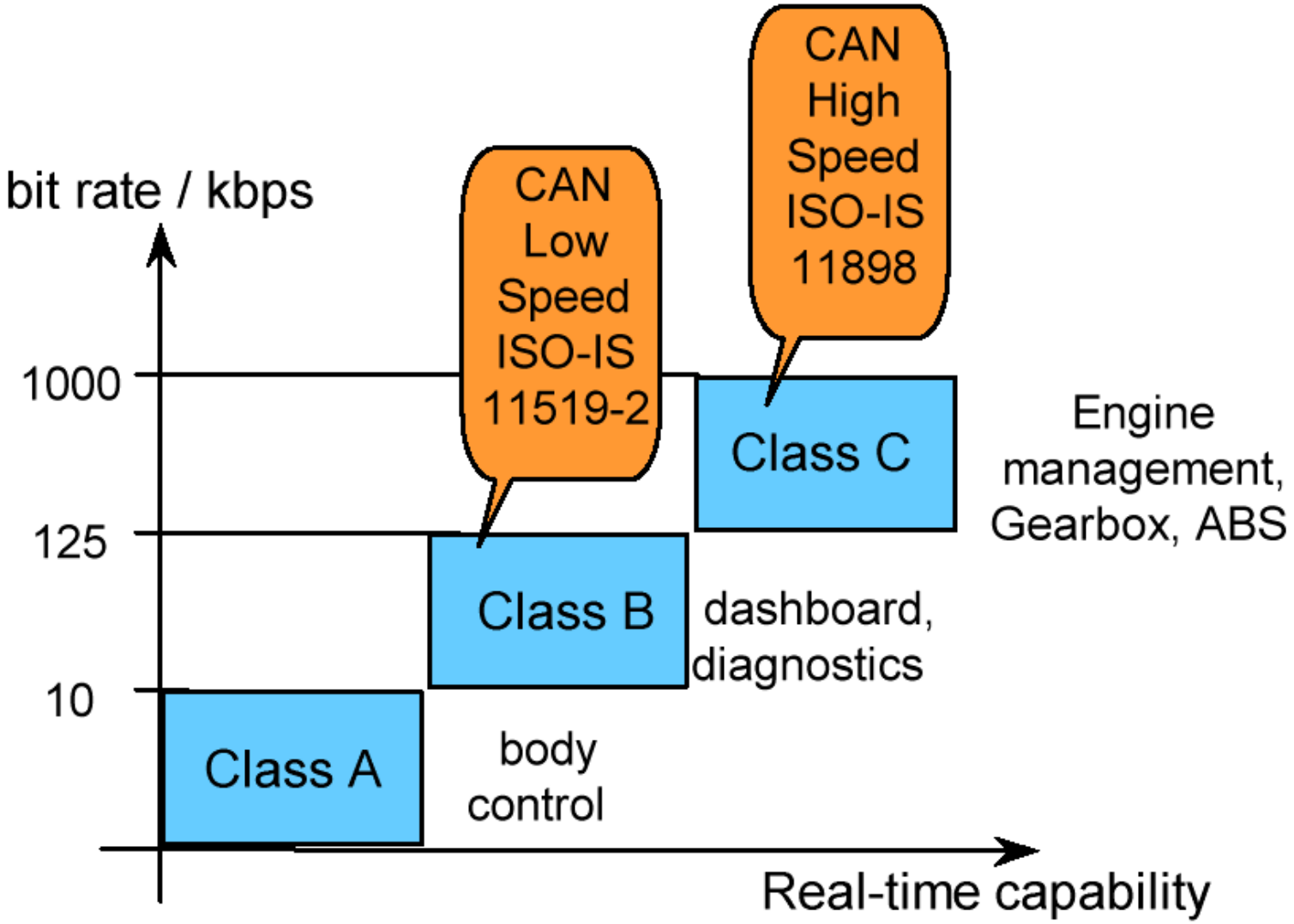
- Need $2*t_{pd}$ per bit maximum speed

❑ Up to 1Mbit / sec @40m bus length (130 feet)



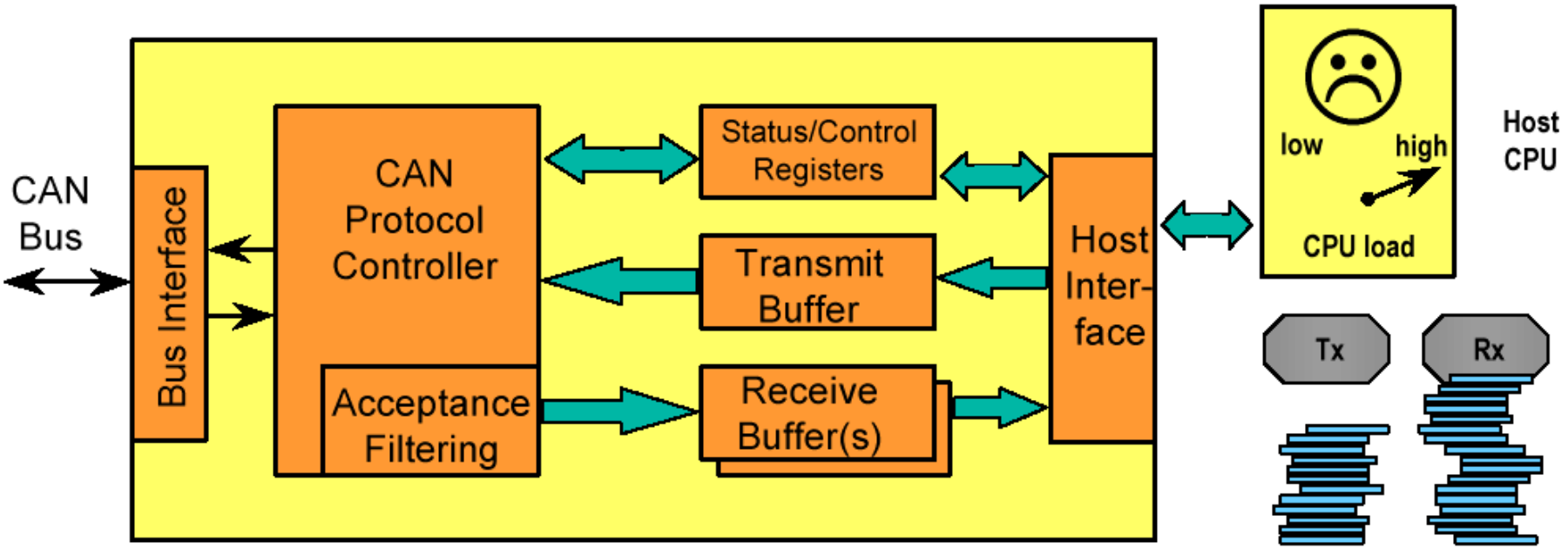
SAE Message Classes

- **Fast tends to correlate with critical control**
  - But, this is not always true; just often true



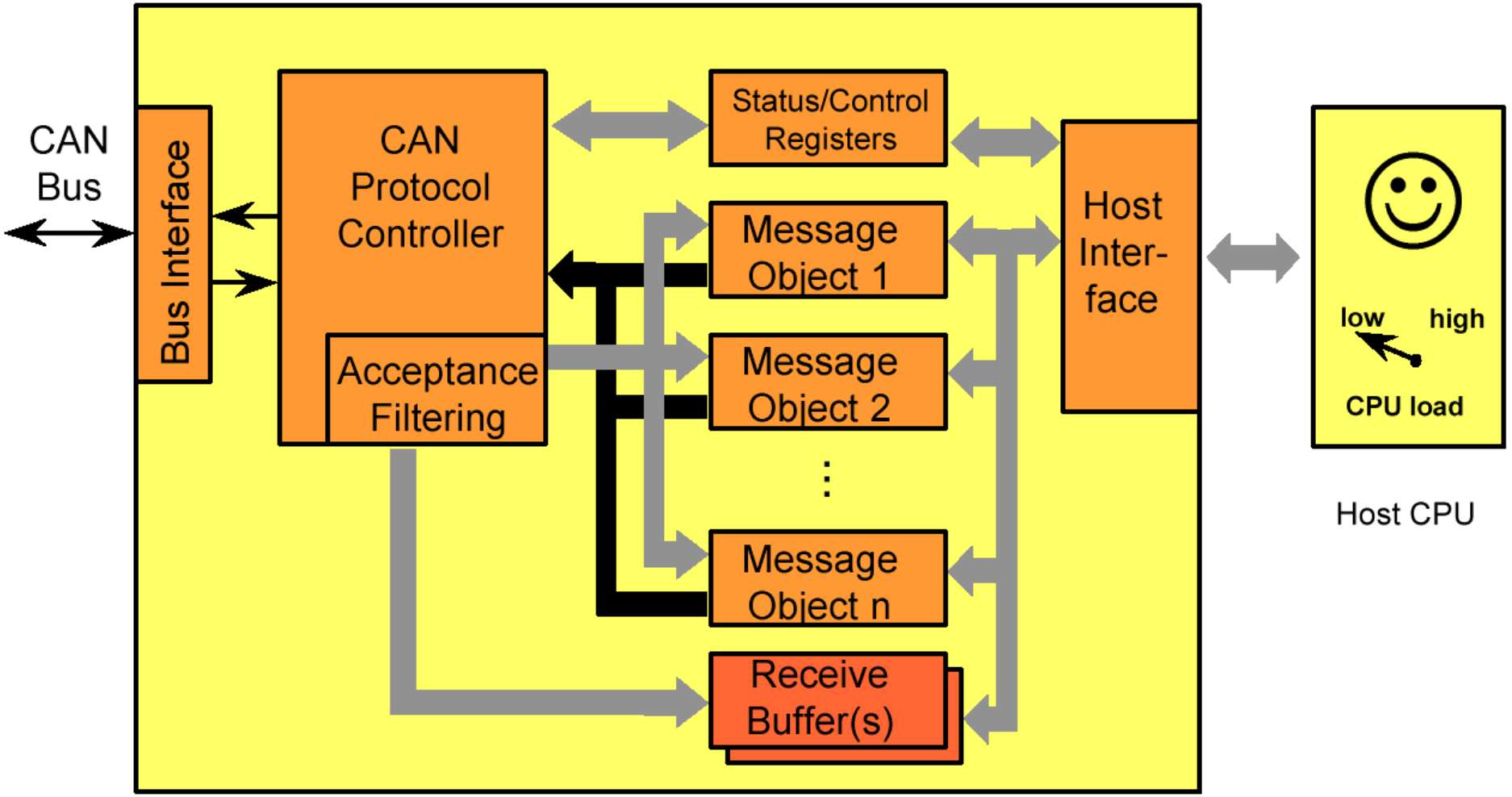
Basic CAN Controller (Don't Use This One)

- **"Cheap" node**
  - Could get over-run with messages even if it didn't need them



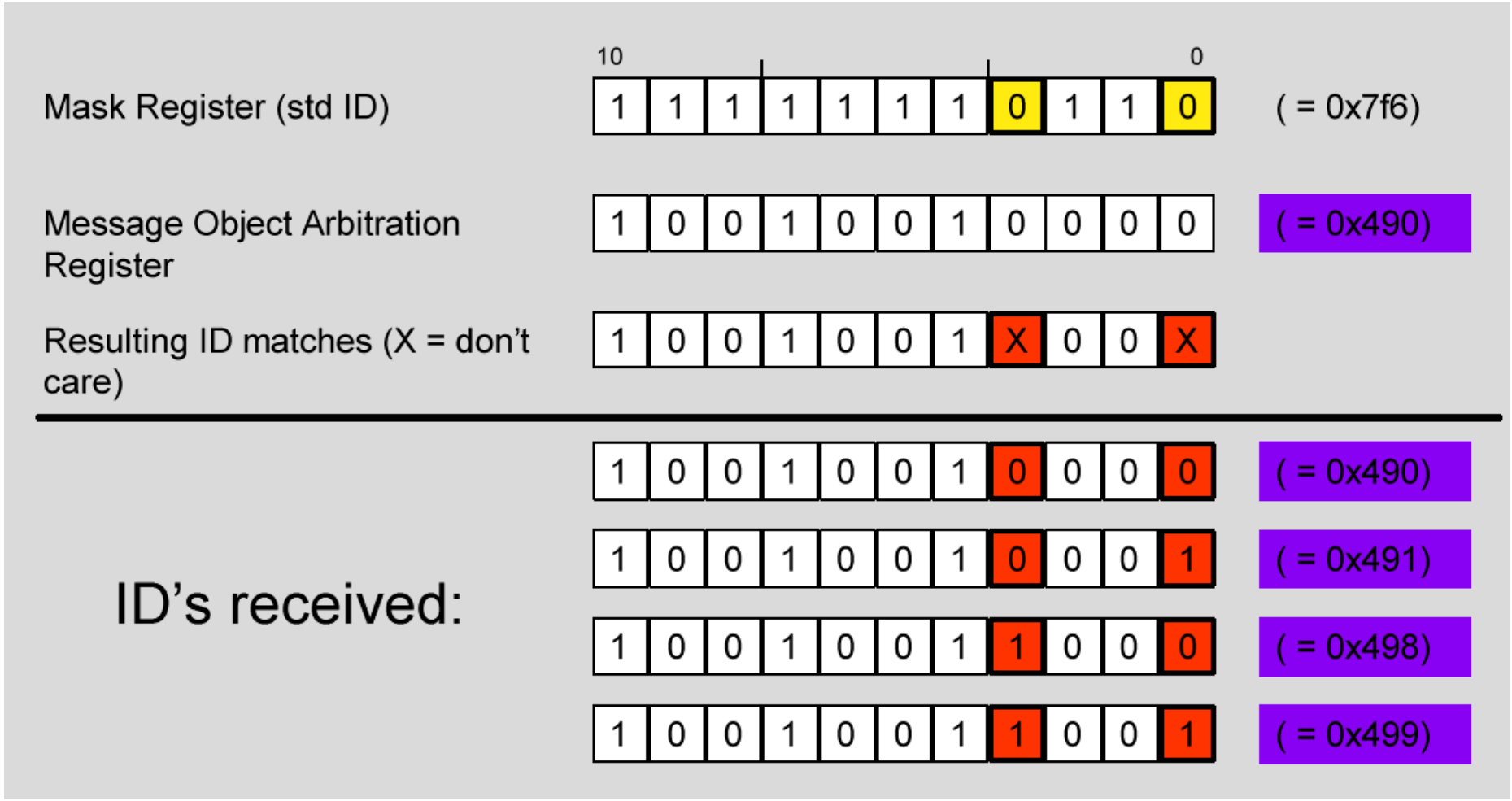
Full CAN Controller

- **Hardware message filters sort & filter messages without interrupting CPU**



Mask Registers

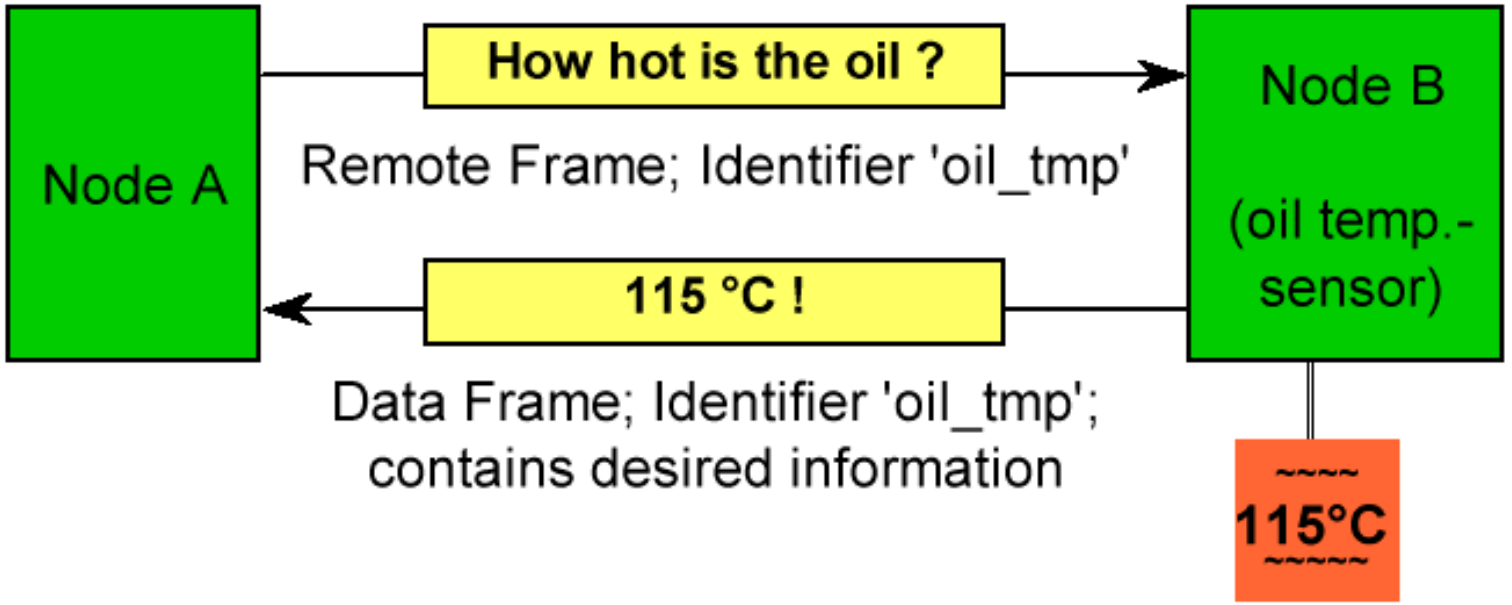
- **Used to set up message filters**
  - Mask register selects bits to examine
  - Object Arbitration register selects bits that must match to be accepted



Various Special Messages

- **Various error messages**

- **Remote Frames –atomic request for data / provide data**
  - (Not used in most in cars)



DeviceNet

- **One of several higher-level protocols**
  - Based on top of CAN
  - Used for industrial control (valves, motor starters, display panels, …)
    - Caterpillar is a member of ODVA as well (Open DeviceNet Vendors Assn.), but for factory automation.

- **Basic ideas:**
  - CAN is used in high volumes = cheaper network chips than competitors
  - Use structured approach to message formats to standardize operation

- **Does *NOT* standardize specific message contents**
  - But it does specify a hierarchy of message ID formats

DeviceNet Message ID Scheme

Message Identifier Bits

10	9	8	7	6	5	4	3	2	1	0	Hex Range	Identity Usage
0	Message ID				Source Node #						000 - 3ff	Group 1
1	0	Source Node #						Msg ID			400 - 5ff	Group 2
1	1	Msg ID (0..6)			Source Node #						600 - 7bf	Group 3
1	1	1	1	1	Message ID (0..2f)						7c0 - 7ef	Group 4
10	1	1	1	1	1	1	X	X	X	X	7f0 - 7ff	Invalid

DeviceNet Group Strategy

- **Group 1**
  - Prioritized by Message ID / Node number
  - High priority messages with fairness to nodes
- **Group 2**
  - Prioritized by Node number / Message ID
  - Gives nodes priority
- **Group 3**
  - Essentially same as Group 1, but allows Group 2 to have higher priority
- **Group 4**
  - Global housekeeping messages / must be unique in system (no node number)

Other Approaches Are Possible

- **And, you can invent your own too…**

- **Variations include:**
  - Automatic assignment of node numbers (include hot-swap)
  - Automatic assignment of message numbers (include hot-swap)
  - Mixes of node-based vs. message-ID based headers

CAN Workloads – Spreadsheets

- **"SAE Standard Workload" (53 messages)** V/C = Vehicle Controller

Signal Number	*Signal Description*	*Size /bits*	*J /ms*	*T /ms*	*Periodic /Sporadic*	*D /ms*	*From*	*To*
1	Traction Battery Voltage	8	0.6	100.0	P	100.0	Battery	V/C
2	Traction Battery Current	8	0.7	100.0	P	100.0	Battery	V/C
3	Traction Battery Temp, Average	8	1.0	1000.0	P	1000.0	Battery	V/C
4	Auxiliary Battery Voltage	8	0.8	100.0	P	100.0	Battery	V/C
5	Traction Battery Temp, Max.	8	1.1	1000.0	P	1000.0	Battery	V/C
6	Auxiliary Battery Current	8	0.9	100.0	P	100.0	Battery	V/C
7	Accelerator Position	8	0.1	5.0	P	5.0	Driver	V/C
8	Brake Pressure, Master Cylinder	8	0.1	5.0	P	5.0	Brakes	V/C
9	Brake Pressure, Line	8	0.2	5.0	P	5.0	Brakes	V/C
10	Transaxle Lubrication Pressure	8	0.2	100.0	P	100.0	Trans	V/C
11	Transaction Clutch Line Pressure	8	0.1	5.0	P	5.0	Trans	V/C
12	Vehicle Speed	8	0.4	100.0	P	100.0	Brakes	V/C
13	Traction Battery Ground Fault	1	1.2	1000.0	P	1000.0	Battery	V/C
14	Hi&Lo Contactor Open/Close	4	0.1	50.0	S	5.0	Battery	V/C
15	Key Switch Run	1	0.2	50.0	S	20.0	Driver	V/C
16	Key Switch Start	1	0.3	50.0	S	20.0	Driver	V/C
17	Accelerator Switch	2	0.4	50.0	S	20.0	Driver	V/C
18	Brake Switch	1	0.3	20.0	S	20.0	Brakes	V/C
19	Emergency Brake	1	0.5	50.0	S	20.0	Driver	V/C
20	Shift Lever (PRNDL)	3	0.6	50.0	S	20.0	Driver	V/C
21	Motor/Trans Over Temperature	2	0.3	1000.0	P	1000.0	Trans	V/C
22	Speed Control	3	0.7	50.0	S	20.0	Driver	V/C
23	12V Power Ack Vehicle Control	1	0.2	50.0	S	20.0	Battery	V/C
24	12V Power Ack Inverter	1	0.3	50.0	S	20.0	Battery	V/C
25	12V Power Ack I/M Contr.	1	0.4	50.0	S	20.0	Battery	V/C
26	Brake Mode (Parallel/Split)	1	0.8	50.0	S	20.0	Driver	V/C

Why Use An Embedded Network

- **Potential Advantages (for CAN?)**
  - Reduces wires and increases reliability
  - Lowers weight, size, and installation costs
  - Logical choice for physically distributed systems
  - Allows sharing of system resources
  - Increases system capability and flexibility
  - Self-configuration, self-installation, and advanced diagnostics
  - Foundation for system integration and automation
  - Integrated, modular product line leads to interoperability

- **Potential Network Drawbacks (for CAN?)**
  - May initially increase product cost
  - Requires knowledge and new skills in networking
  - Requires special tools for fault detection

CAN Tradeoffs

- **Advantages**
  - High throughput under light loads
  - Local and global prioritization possible
  - Arbitration is part of the message - low overhead

- **Disadvantages**
  - Propagation delay limits bus length (2 t_{pd} bit length)
  - Unfair access - node with a high priority can "hog" the network
    - Can be reduced in severity with Message + Node # prioritization
  - Poor latency for low priority nodes
    - Starvation is possible

- **Optimized for:**
  - Moderately large number of message types
  - Arbitration overhead is constant
  - Global prioritization (*but* limited mechanisms for fairness)

Review

- **Controller Area Network**
  - Binary-countdown arbitration
  - Standard used in automotive & industrial control

- **CAN Tradeoffs**
  - Good at global priority (but difficult to be “fair”)
  - Efficient use of bandwidth
  - Requires bit-dominance in physical layer
  - Message filters are required to keep small nodes from being overloaded
    - (But, these are easy to implement)

- **Next lecture: Protocol building blocks (custom protocols)**