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.

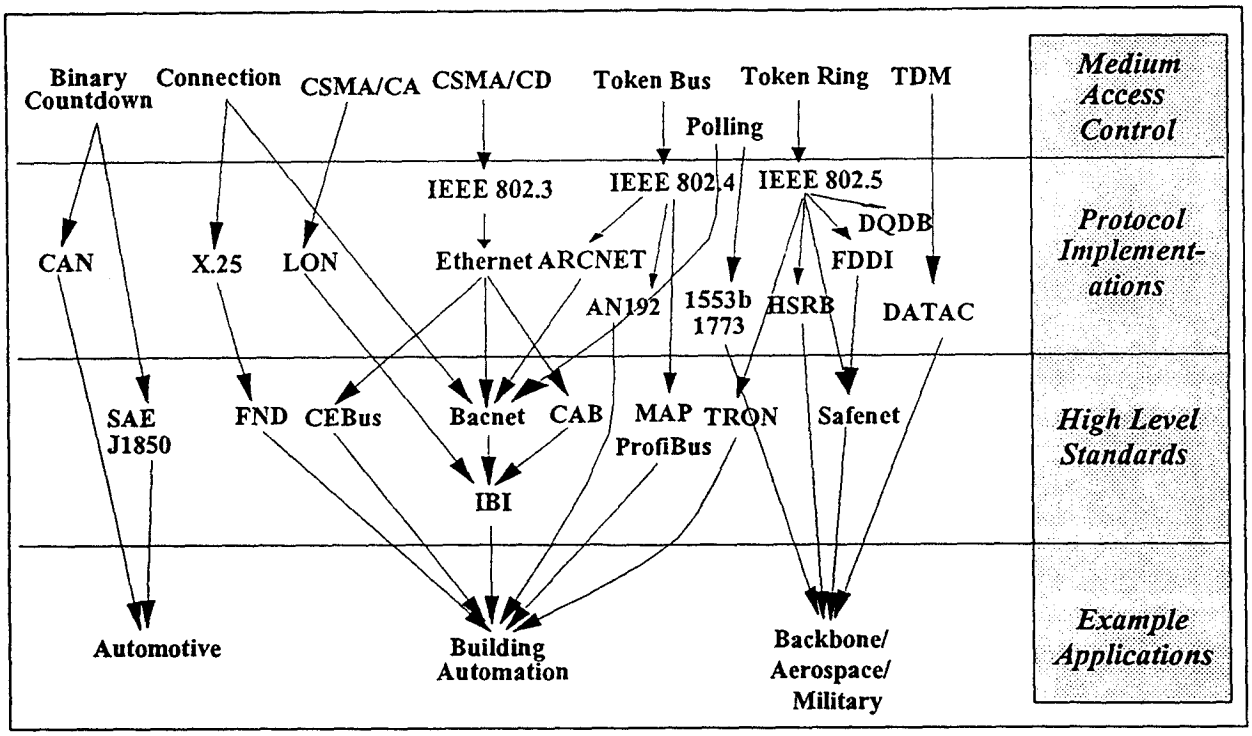

Figure 1: "Standard" Protocol Family Tree

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

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.