

The CIP Family of Fieldbus Protocols and its Newest Member – EtherNet/IP

© Institute of Electrical and Electronic Engineers, EFTA 2001 By Viktor Schiffer Engineering Manager, European Technology Development Unit Rockwell Automation

October 2001

*Abstract – DeviceNet™ and ControlNet™ are two well known members of the same family of protocols – the CIP family (**CIP = Control an Information Protocol**). Both protocols have been developed by Rockwell Automation, but are now owned and maintained by the two manufacturers organizations **ODVA (Open DeviceNet Vendors Association)** and **CI (ControlNet International)**. ODVA and CI have recently introduced the newest member of this family – EtherNet/IP ("**IP**" stands for "**Industrial Protocol**"). This paper describes the basics of CIP and gives an overview over the newest member of this family of protocols – **EtherNet/IP**.*

I. Introduction In the past, typical fieldbus protocols (Profibus, Interbus-S, FIP, P-Net, AS-i) have been isolated implementations of certain ideas and functionalities that the inventors thought were best suited to solve a certain problem or do a certain job. This has led to quite effective fieldbuses that do their job quite well, but that are suitable only for certain layers within the automation pyramid or that are limited in their functionality (e.g. strict single master systems running a master/slave protocol). This typically results in "barriers" within the automation architecture that are difficult to penetrate and that require complex bridging devices without being able to fully bridge the gap between the various systems that can be quite different in nature.

In contrast, the CIP family of protocols offers a very well scalable solution that allows a uniform protocol to be employed from the very top to the very bottom of the automation pyramid without burdening the individual devices too much.

DeviceNet was the first member of this protocol family that was made public in 1994. DeviceNet is a CIP implementation on the very popular CAN protocol layer. CAN in its typical form (ISO 11898, [6]) only defines layers 1 and 2 of the OSI 7-layer model while DeviceNet covers the rest. The low cost of implementation and the ease of use of the DeviceNet protocol has led to a large number of manufacturers with more than 300 of them organized in the ODVA.

ControlNet, introduced a few years later (1997), implemented the same basic protocol on a new physical layer that allows much higher speed (5 MBd) and strict determinism and repeatability while extending the range of the bus (several kilometers with repeaters) for more demanding applications. Vendors and users of ControlNet products are organized within CI to promote the use of these products.

ODVA and CI have recently introduced the newest member of the CIP family – EtherNet/IP ("IP" stands for "Industrial Protocol").

In addition to these public implementations, the CIP protocol is also used in a product range of Rockwell Automation, the ControlLogix product family. The universal principles of the CIP protocol also lend themselves to future implementations on new physical layers, e.g. ATM, USB or Firewire.

II. Description of the CIP Protocol CIP is a very versatile protocol that has been designed with the automation industry in mind. However, due to its very open nature, it can be applied to many more areas. The general CIP protocol specification [1] is available for download from the [ODVA web site](#). It is beyond the scope of this paper to fully describe each and every detail of this specification, but the key feature will be looked at. The specification is subdivided into several chapters and appendices which describe the following features:

Object Modeling

Messaging Protocol

Communication Objects

General Object Library

Device Profiles

Electronic Data Sheets

Services

Data Management

There are a few more chapters containing description of further CIP elements, but they are not of significance in the context of this paper.

Let's now have a look at the individual elements of CIP:

A. Object Modeling CIP makes use of abstract object modeling to describe:

The suite of communication services available

The externally visible behavior of a CIP node

A common means by which information within CIP products is accessed and exchanged

Every CIP node is modeled as a collection of objects. An object provides an abstract representation of a particular component within a product. Anything not described in object form is not visible through the CIP protocol. CIP objects are structured into classes, instances, and attributes. A class is a set of objects that all represent the same kind of system component. An object instance is the actual representation of a particular object within a class. Each instance of a class has the same attributes, but it has its own particular set of attribute values. As [Figure 1](#) illustrates, multiple object instances within a particular class can reside within a CIP node. An object instance and/or object class has attributes, provides services and implements a behavior.

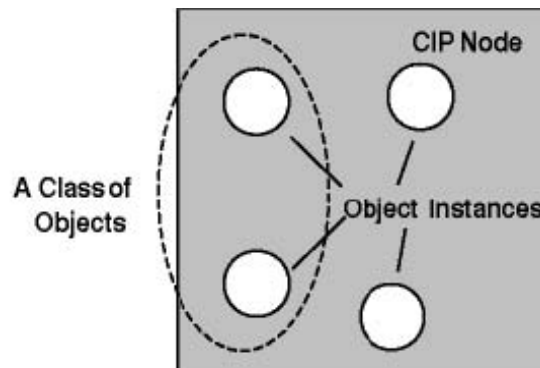


Figure 1: A Class of Objects

The objects and their components are addressed by a uniform addressing scheme consisting of:

Media Access Control Identifier (MAC ID), an integer identification value assigned to each node on a CIP network.

Class Identifier (Class ID), an integer identification value assigned to each Object Class accessible from the network.

Instance Identifier (Instance ID), an integer identification value assigned to an Object Instance that identifies it among all Instances of the same Class.

Attribute Identifier (Attribute ID), an integer identification value assigned to a Class and/or Instance Attribute.

Service Code, an integer identification value which denotes a particular Object Instance and/or Object Class function.

Figure 2: Object Addressing Example

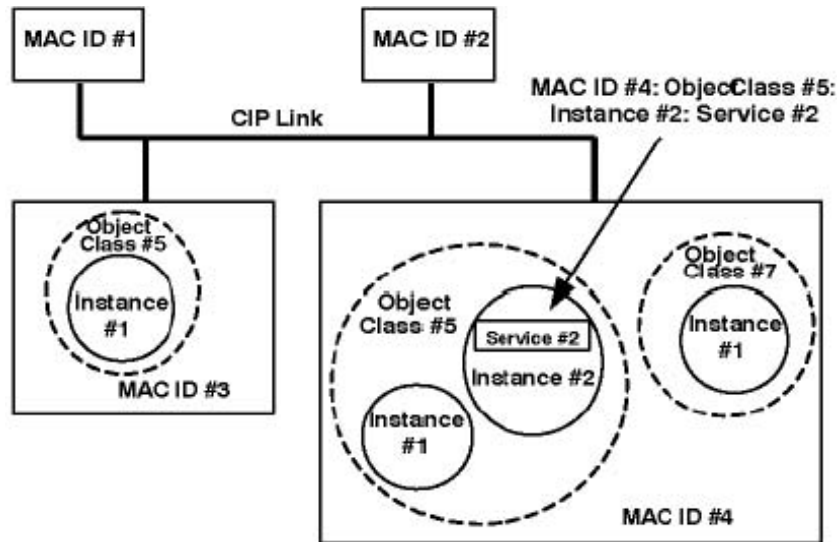
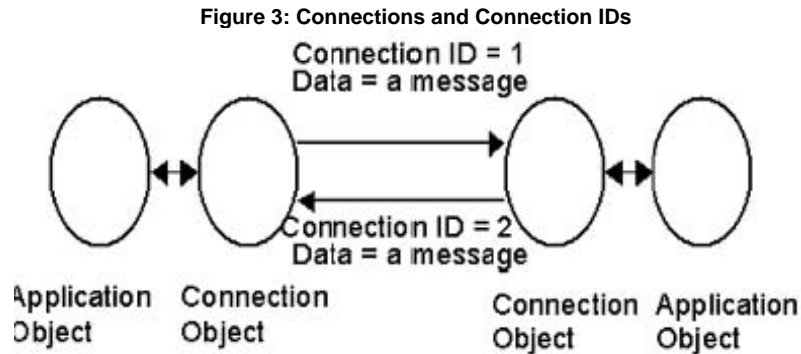


Figure 2 shows an example of this object addressing scheme. More details of the object modeling can be found in chapters 1 and 4 of the CIP specification [1].

B. Messaging Protocol CIP is layered on top of a connection-based network. A CIP connection provides a path between multiple applications. When a connection is established, the transmissions associated with that connection are assigned a Connection ID (CID). If the connection involves a bi-directional exchange, then two Connection ID values are assigned. [See Figure 3.](#)



The definition and format of the connection ID is network dependent. For example, the connection ID for CIP connections over DeviceNet is based on the CAN Identifier Field.

Since most messaging on a CIP network is done through connections, a process has been defined to establish such connections between devices that are not "connected" yet. This is done through the Unconnected Message Manager (UCMM), which is responsible for processing.

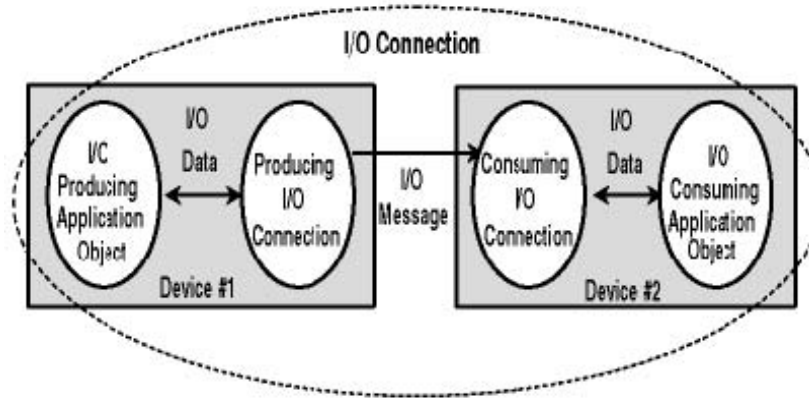
All connections in a CIP network can be divided into I/O connections and explicit messaging connections.

I/O connections provide dedicated, special purpose communication paths between a producing application and one or more consuming applications. Application-specific I/O data moves through these ports and is often referred to as implicit messaging.

Explicit messaging connections provide generic, multi-purpose communication paths between two devices. These connections are often referred to as just "messaging connections." Explicit messages provide the typical request/response-oriented network communication.

More details of the messaging protocol can be found in chapter 2 of the CIP specification [1].

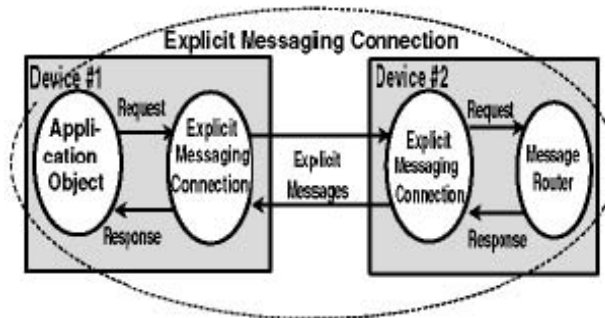
Figure 4 CIP I/O Connection



C. Communication Objects The CIP communication objects manage and provide the runtime exchange of messages. While these objects follow the overall principles and guidelines for CIP objects, the communication objects are unique in a way since they are the focal point for all CIP communication. It therefore makes sense to have a look at them in more detail.

Every instance of a communication object contains a link producer part or a link consumer part or both. I/O connections may be producing only, consuming only or producing and consuming, while explicit messaging connections always are producing and consuming.

Figure 5: CIP Explicit Messaging Connection



[Figures 4](#) and [5](#) show the typical connection arrangement for CIP I/O messaging and CIP explicit messaging.

The attribute values in the connection objects define a set of attributes that describe vital parameters of this connection.

First of all, they state what kind of connection this is. They specify whether this is an I/O connection or an explicit messaging connection, but also the maximum size of the data to be exchanged across this connection and the source and sink of this data.

Further attributes define the state of this connection and what kind of behavior this connection is to show. Of particular importance is how messages are triggered (from the application, through change of state or change of data, through cyclic events, or by network events), and the timing of the connections (time-out associated with this connection and pre-defined action if a time-out occurs). CIP allows multiple connections to coexist in a device, although simple devices, e.g. simple DeviceNet slaves, will only have very few connections alive at any given point in time.

More details of the communication objects can be found in chapter 3 of the CIP specification [1].

D. General Object Library The CIP family of protocols contains a fairly large collection of commonly defined objects (currently 46 object classes). The overall set of object classes can be subdivided into three types:

General use objects

Application specific objects

Network specific objects

Apart from the objects that are network specific, all other objects are common objects that can and will be used in all three protocols.

The following are general use objects:

Identity Object	Parameter Object
Message Router Object	Parameter Group Object
Assembly Object	Acknowledge Handler Object
Connection Object (see C)	Connection Configuration Object
Connection Manager Object	Port Object

A further group of objects is application specific objects:

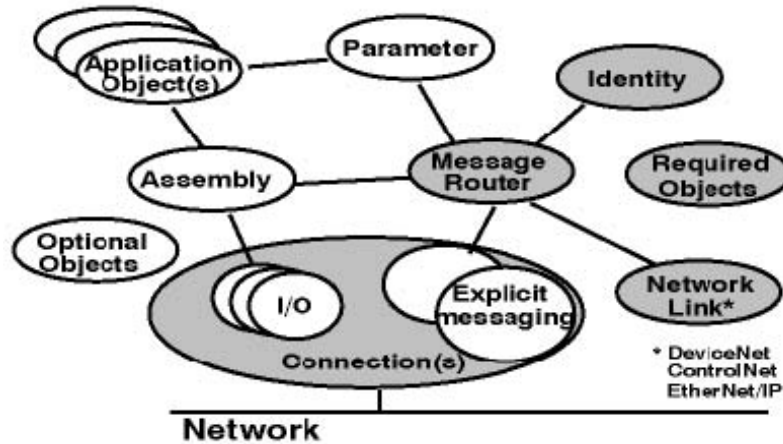
Register Object	Position Controller Supervisor Object
Discrete Input Point	Position Controller Object Block
Register Object	Sequencer Object
Discrete Input Point Object	Command Block Object
Discrete Output Point Object	Motor Data Object
Analog Input Point Object	Control Supervisor Object
Analog Output Point Object	AC/DC Drive Object
Presence Sensing Object	Overload Object
Group Object	Softstart Object
Discrete Input Group Object	Selection Object
Discrete Output Group Object	S-Device Supervisor Object
Discrete Group Object	S-Analog Sensor Object
Analog Input Group Object	S-Analog Actor Object
Analog Output Group Object	S-Single Stage Controller Object
Analog Group Object	S-Gas Calibration Object
Position Sensor Object	Trip Point Object

The last group of objects is network specific objects:

- DeviceNet Object (specific to DeviceNet only)
- ControlNet Object (specific to ControlNet only)
- ControlNet Keeper Object (specific to ControlNet only)
- ControlNet Scheduling Object (specific to ControlNet only)
- TCP/IP Interface Object (specific to EtherNet/IP only)
- Ethernet[®] Link Object (specific to EtherNet/IP only)

The general use objects can be found in many different devices, while the application specific objects are typically only found in devices hosting such applications.

Figure 6: Typical Device Object Model



This looks like a huge number of object types, but typical devices only implement a subset of these objects. [Figure 6](#) shows the object model of a such a typical device.

The objects required in a typical device are:

- At least one Connection Object
- An Identity Object
- One or several network link related object (depends on network)
- A Message Router Object (at least its function)

Further objects are added according to the functionality of the device. This allows very good scalability of devices so that small devices (e.g. a proximity sensor on DeviceNet) are not burdened with unnecessary overhead. A developer typically uses publicly defined object (see above list), but can also create his own objects in the vendor specific areas, e.g. class ID 100 – 199. However, it is strongly encouraged to work in the Special Interest Groups (SIGs) of ODVA and ControlNet International to create common definitions for further objects instead of inventing private ones.

As an example of the many existing objects, the identity object (class code: 1) is described below. Since the vast majority of devices only support one instance of the identity object, there is typically no requirement for any class attributes. Thus only instance attributes are required in most cases. These are:

IDENTITY OBJECT	
Mandatory Attributes	Optional Attributes

Vendor ID Device Type Product Code Revision Status Serial Number Product Name	State Configuration Consistency ValueHeartbeat Interval
---	---

Typically, devices do not change their identity, so all attributes (with the exception of the Heartbeat Interval attribute) are read-only.

More details of the CIP objects in general and the identity object in particular can be found in chapter 5 of the CIP specification [1].

E. Device Profiles With the definitions of communication links and objects, it would very well be possible to design products. However, similar products could easily have quite different structures inside and could also show quite different behavior. To overcome this situation and to make the application of CIP devices much easier, devices of similar functionality have been grouped into device types with associated profiles. Such a CIP profile contains the full description of the object structure and their behavior. The following device types and associated profiles have been fully defined so far (August 2001):

- Generic Device
- AC Drives
- Motor Overload
- Limit Switch
- Inductive Proximity Switch
- Photoelectric Sensor
- General Purpose Discrete I/O
- Resolver
- Communication Adapter
- ControlNet Programmable Logic Controller – Position Controller
- DC Drives
- Contactors
- Motor Starter
- Soft Start
- Human Machine Interface
- Mass Flow Controller

Pneumatic Valves

Vacuum Pressure Gauge

ControlNet Physical Layer

Device developers must use a profile. Any device that does not fall into the scope of one of the specialized profiles must use the generic device profile or a vendor specific profile. What profile is used and which parts of it are implemented must be described in the user documentation of the device.

Every profile consists of a set of objects, some required, some optional, and a behavior associated with that particular type of device. Most profiles also define one or several I/O data formats that define the meaning of the individual bits and bytes of the I/O data. In addition to the publicly defined object set and I/O data assemblies, manufacturers can add objects and assemblies of their own if they have additional data not described in the public profile. Again, it is encouraged to coordinate additional data used by many manufacturers through discussion in the SIGs and eventually create additions to the public profiles for everybody's use and for the benefit of the device users.

More details of the CIP profiles can be found in chapter 6 of the CIP specification [1].

F. Electronic Data Sheets CIP has made provisions for several options to configure devices:

A printed data sheet

Parameter Objects and Parameter Object Stubs

An Electronic Data Sheet (EDS)

A combination of an EDS and Parameter Object Stubs

A Configuration Assembly and any of the above methods

When using configuration information collected on a printed data sheet, configuration tools can only provide prompts for service, class instance, and attribute data, and relay this information to a device. While this procedure can do the job, it is the least desirable solution since it does not determine the context, content, or format of the data.

Parameter objects, on the other hand, provide a full description of all configurable data of a device. This allows a configuration tool to gain access to all parameters and maintain a user-friendly interface since the device itself provides all the necessary information. However, this method burdens a device with full parameter information and this may be too much for a small device, e.g. a simple DeviceNet slave. Therefore, an abbreviated version of the parameter object, called parameter object stub may be used. This still allows access to the parameter data, but it does not describe any meaning of this data. This is where an EDS is very handy. An EDS supplies all the information that a full parameter object contains on top of what the parameter object stub provides. The combination of EDS and parameter object stub thus provides the full functionality and ease of use of the parameter object without burdening the individual devices.

Finally, a configuration assembly allows the bulk upload and download of a full block of parameters.

More details of electronic data sheets used in CIP can be found in chapter 7 of the CIP specification [1].

G. Services Service codes are used to define the action that is to take place when an object of parts of an object are addressed using the addressing scheme described in A. Apart from the simple read and write functions, a set of CIP Common Services (as of August 22, 2001) has been defined. These CIP Common Services are common in nature which means that they are universal in use and a large number of objects typically support these services. Furthermore, there are object specific service codes which may have a different meaning for the same code depending on the object. Finally, there is a possibility to define vendor specific services according to the requirements of the developer. While this gives a lot of flexibility, the disadvantage of vendor specific services is that they may not be understood universally.

More details of the CIP service codes can be found in appendix A of the CIP specification [1].

H. Data Management The data management part of the CIP specification describes addressing models for CIP entities and the data structure of the entities themselves.

The entity addressing is done by so-called segments, a method that allows to be used in a very flexible way so that many different types of addressing methods can be accommodated. In its simplest form, the segment addressing provides the class/instance/attribute addressing required for CIP objects.

The data types in CIP follow the requirements of IEC 61131-3 [5]. There is a set of elementary data types and data types that are derived from these elementary types.

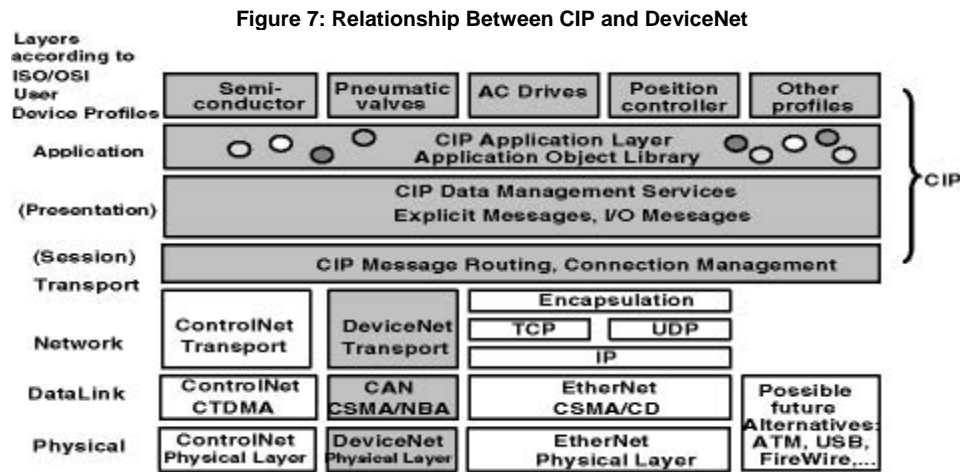
More details of the CIP data management can be found in appendix B of the CIP specification [1].

III. Applications Of The CIP Protocol

Up to now there are three public derivatives of the CIP protocol. These three derivatives are based on quite different transport mechanisms, but they maintain the principles of CIP.

A. DeviceNet DeviceNet was the first public implementation of CIP. As already mentioned in chapter I, DeviceNet is based on CAN and the adaptations of CIP are mainly done to accommodate certain limitations of the CAN protocol.

Figure 7 shows the relationship between CIP, DeviceNet and the ISO/OSI layer model. DeviceNet uses a subset of the CAN protocol (11 bit identifier only, no remote frames), and its physical layer is an extension of the ISO 11898 standard [6]. This extension requires the support of up to 64 nodes per network and circuitry for overvoltage and miswiring protection. The transmission baud rates for DeviceNet have been defined as 125, 250, and 500 kbd resulting in maximum system lengths of 500, 250, and 100 meters.



There are basically two adaptation of CIP (apart from the addition of the DeviceNet object) that have been made to better accommodate it to the CAN protocol:

Limitation to short messages (8 bytes or less) where possible, introduction of fragmentation for longer messages.

Introduction of master/slave communication profile for better scalability.

These two features have been introduced to allow the use of small and thus inexpensive microcontrollers. This is particularly important for small, cost-sensitive devices like photoeyes and proximity sensors. As a result of this specialization, the DeviceNet protocol can reside in very small chips like the Motorola 68HC05X4 micro with approximately 4 kbyte of code memory.

The master/slave communication profile makes pre-definitions for typical communication links between centralized and decentralized devices (masters and slaves). Masters typically collect input information and distribute output information. However, unlike the limited functionality of other commonly-used fieldbus systems [like Interbus-S (strict single master system) or Profibus DP (multiple masters possible, but all traffic limited to pure master/slave communication)], DeviceNet has a very flexible set of communication links, even for slave devices:

Multiple message triggers possible: Network (Strobe, Poll), Change-of-State, Cyclic.

Every device may act as Explicit Message client.

Master and slave device functionality possible in same device.

Peer-to-peer communication allowed between any set of devices that support this functionality.

Sharing of input and output data through multicast messages.

Only a minimum subset of the full possible DeviceNet functionality is required to successfully participate in a DeviceNet system. Thus very simple and very complex devices can easily cooperate on the same link. Most devices will support the master/slave connection set, but the DeviceNet specification allows enough freedom to use additional "pure" CIP messaging without interfering with the master/slave connection set.

DeviceNet makes use of the full, unmodified set of objects and profiles described in the CIP specification. The additional data set required to manage DeviceNet communication is contained in the DeviceNet object (class code 03hex).

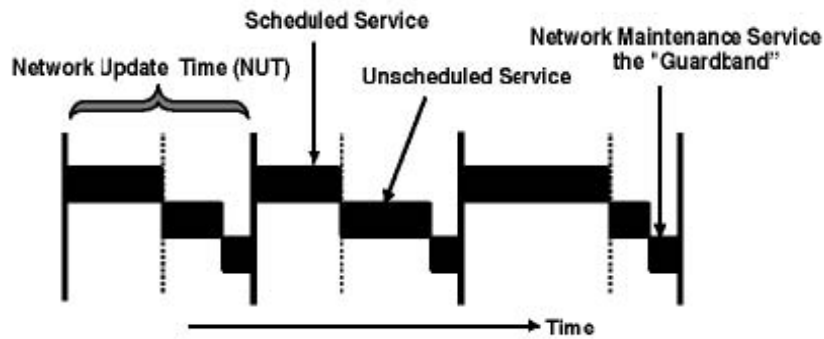
The main benefits of DeviceNet are its low cost (development and manufacturing), its ease of use through user-friendly configuration tools, the high functionality, and the capability of achieving very good speed performance through Change of State data transmissions.

A full description of DeviceNet can be found in the DeviceNet specification [2].

B. ControlNet ControlNet is based on a new high-speed (5 Mbaud) physical layer and bus access mechanism. The physical layer uses either fiber-optic cabling or RG6 coaxial cable. With the coaxial cable, the maximum segments can be up to 1000 meters long. With fiber-optic cabling, the maximum system length depends mainly on the characteristics of the optical fiber. The bus access mechanism allows full determinism and repeatability while still maintaining sufficient flexibility for various I/O message triggers and explicit messaging.

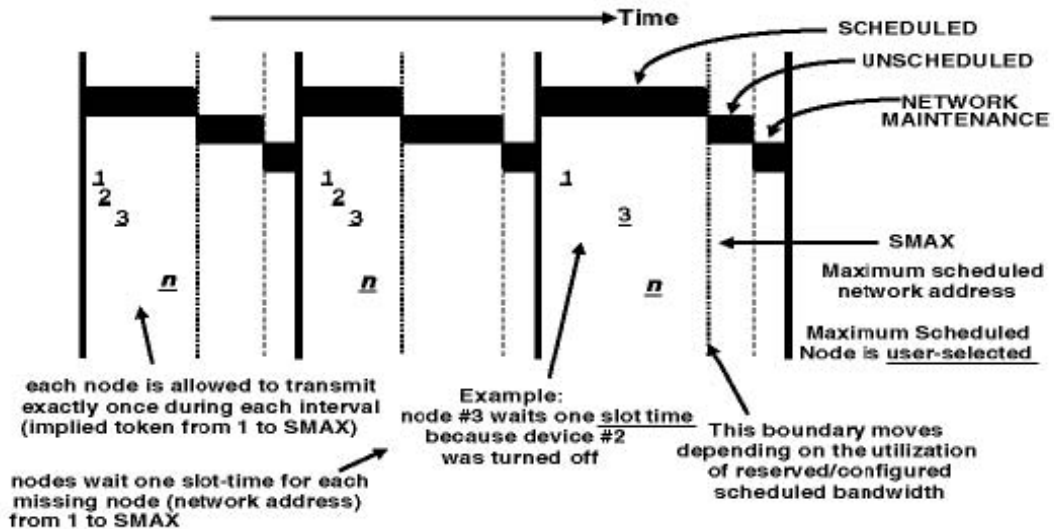
This bus access mechanism is called Concurrent Time Domain Multiple Access (CTDMA) illustrated in [Figure 8](#).

Figure 8: Media Access through CTDMA (Concurrent Time Domain Media Access)



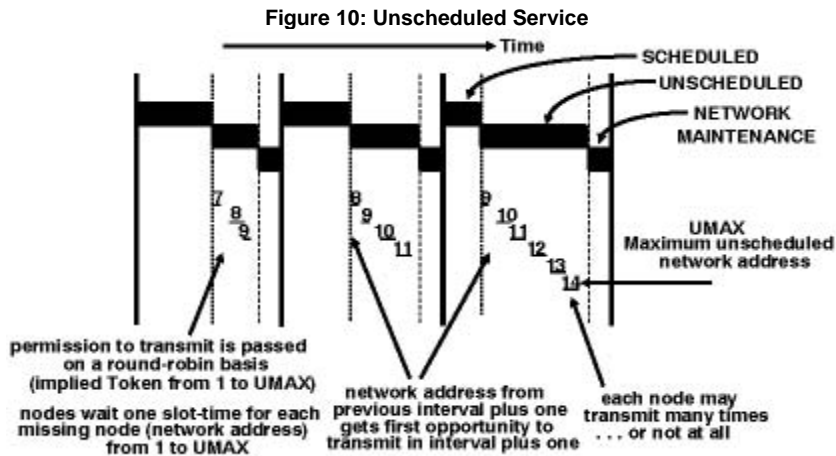
The time axis is divided into equal intervals, called Network Update Time (NUT). Within each NUT, there is a subdivision into a scheduled service time, an unscheduled service time, and a network maintenance service time.

Figure 9: Scheduled Service



[Figure 9](#) shows the function of the scheduled service. Every node up to the SMAX node (maximum node number participating in the scheduled service) has a chance to send a message within the scheduled service. If a particular node has no data to send, it will nevertheless send a short frame to indicate that it is still alive. If a node fails to send its frame, the next higher node number will step in after a very short, predetermined waiting time. This makes sure that a failure of a node will not lead to an interruption of the NUT cycle.

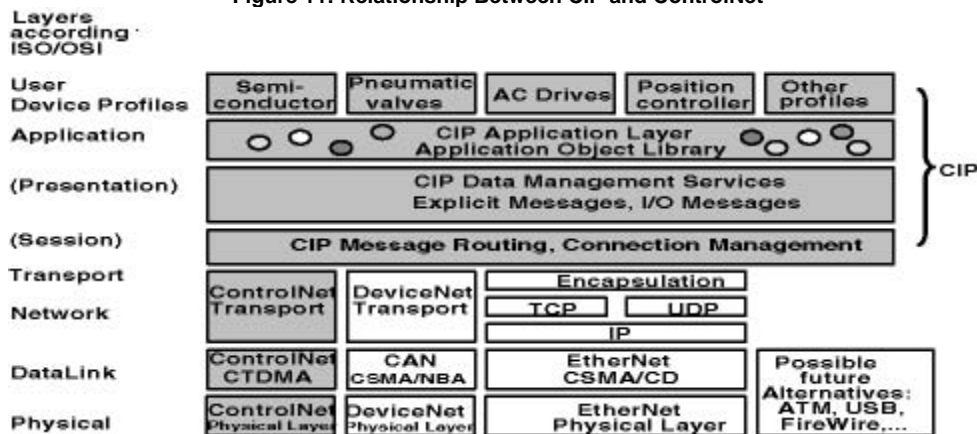
[Figure 10](#) shows the function of the unscheduled service. Since this service is designed for non time-critical messages, only one node is guaranteed to get access to the bus during the unscheduled service time. If there is time left, then the other nodes (with higher node numbers) will also get a chance to send. As with the scheduled service time, if a node fails to send when it is its turn, the next node will step in. In every NUT, the node number that is allowed to send first within the unscheduled service time is increased by one. This method guarantees an equal chance to all nodes.



Those two service intervals combined guarantee determinism and repeatability, while still maintaining sufficient freedom to allow for unscheduled message transmissions, e.g. for parameterization.

Frames sent by a particular node (called M-packets) may contain multiple messages (called L-packets). This feature allows fine-tuned multicasting of small amounts of data to different sets of consumers without too much overhead.

Figure 11: Relationship Between CIP and ControlNet



[Figure 11](#) shows the relationship between CIP, ControlNet and the ISO/OSI layer model. Due to the higher transmission speed and larger maximum data frame size (> 500 bytes) of ControlNet, more powerful processor are required than for DeviceNet. Since these processors typically also have more addressing capabilities, it does not make sense to introduce a somewhat limited communication profile for master/slave applications. Instead, there are four classes of ControlNet devices:

Explicit message servers. These devices can only respond to explicit messages.

I/O message servers. These devices cannot start up I/O connections. Once the I/O connections have been established, these devices can send multiple I/O messages with various message triggers.

These devices are typically called adapters.

Adapters with explicit message clients.

Full functionality devices with I/O client and server and explicit messaging client and server capability.
These devices are typically called scanners.

The full application of the CIP protocol thus allows very efficient transfer of data between devices. In particular fully deterministic broadcast transfer of data using the producer/consumer model (under full control of the associated connection objects) is no problem at all. ControlNet therefore offers more efficient data transfers than other networks running at higher baud rates.

There are three additional objects required for ControlNet:

ControlNet Object (class code F0hex, containing communication parameters for the ControlNet communication link)

ControlNet Keeper Object (class code F1hex, containing information on the overall structure of the ControlNet subnet)

ControlNet Scheduling Object (class code F2hex, containing information on the various scheduling services)

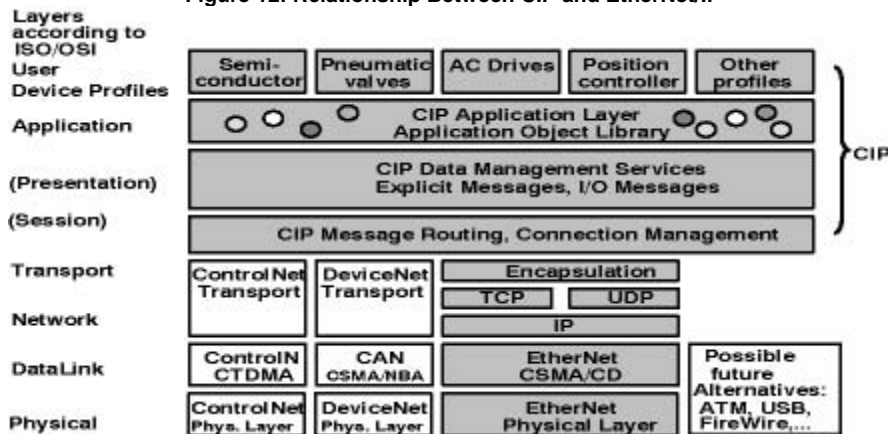
A full description of ControlNet can be found in the ControlNet specification [3].

C. EtherNet/IP EtherNet/IP is the newest member of the CIP family. EtherNet/IP is very similar to ControlNet in how the CIP specification is applied. Due to the length of the Ethernet frames and the typical multi-master structure of Ethernet networks, there are no particular limitations in the EtherNet/IP implementation of CIP. Basically all that is required is a mechanism of how to encode CIP messages in Ethernet frames. [Figure 12](#) shows that there is an encapsulation mechanism (described in chapter 2 of the EtherNet/IP specification [4]) that details how I/O and explicit messages are coded into Ethernet frames. The well-known TCP/IP protocol is used for the encapsulation of explicit messages while UDP/IP is used for the encapsulation of I/O messages. The use of the very popular TCP/IP and UDP/IP protocol stack for encapsulation means that many applications will not require extra "middleware" for this purpose, since these stacks are already in use in many applications anyway.

Since EtherNet/IP is taking the Ethernet protocol to the factory floor, there are some restrictions and some further requirements on the physical layer [7] that is to carry EtherNet/IP in a typical factory automation environment. These additional details are specified in chapter 8 of the EtherNet/IP specification [4].

EtherNet/IP makes the same subdivision into four classes of devices as ControlNet to distinguish their communication capabilities.

Figure 12: Relationship Between CIP and EtherNet/IP



There are two additional objects required for EtherNet/IP:

TCP/IP Object (class code F5hex, containing information on the use of the TCP/IP protocol)

Ethernet Link Object (containing information on the communication parameters for the EtherNet/IP communication link)

A full description of EtherNet/IP can be found in the EtherNet/IP specification [4].

Figure 13: Relationship of CIP to Other Typical Ethernet Protocols

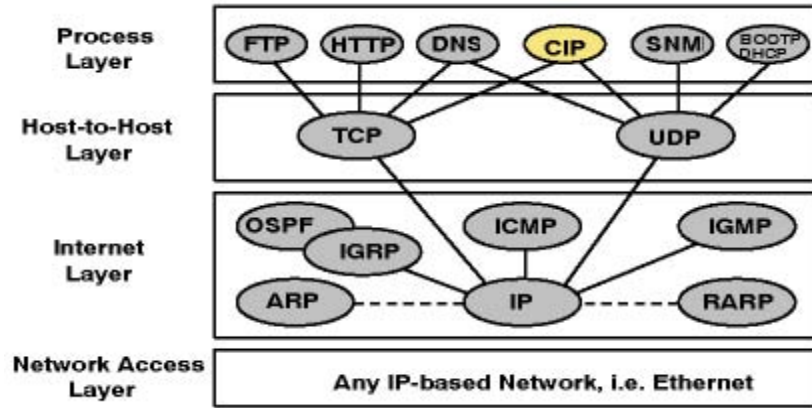


Figure 13 shows the relationship between CIP and other typical Ethernet based protocol stacks. This shows that coexistence with many other services is no problem at all, and CIP blends nicely into the set of already existing functions. This means that anybody already using some or all of these popular Ethernet services can add CIP without too much of a burden. The existing services like http or ftp may remain as before, and CIP will become another service on this layer.

To successfully apply EtherNet/IP to the automation world, the issue of determinism has to be considered. The inherent principle of the Ethernet bus access mechanism – whereby collisions are detected, nodes back off and try again after a while – cannot guarantee determinism. While Ethernet in its present form cannot be made strictly deterministic, there are ways to improve this situation.

First of all, the typical hubs used in an office environment have to be replaced by intelligent switches that will forward only those Ethernet frames that are intended for nodes connected to this switch. With the use of switch technology, collisions are largely avoided except for those cases where two or more messages are sent to the same node at the same time. The situation can be further improved by switching to a higher baud rate without increasing the number of messages. This will result in a lower bandwidth utilization and will thus reduce the chance of collision.

Since today's Ethernet cannot be made fully deterministic, the ODVA/CI recommendation for applications requiring strict determinism is to switch to ControlNet, which can guarantee full determinism and repeatability even when used with repeaters to extend the physical length of the system.

IV. Benefits Of The CIP Protocol Family The benefits of the CIP family can be subdivided into two groups:

Benefits for the manufacturer of devices

Benefits for the user of devices and systems

A. Benefits for the manufacturer of devices Major benefits for manufacturers come from the fact that existing knowledge can be reused from one protocol to another. This results in lower training costs for development, sales and support personnel. Reduced development costs can be achieved, since certain parts (e.g. parameters, profiles) of the embedded firmware can be reused from one network to the other since they are identical. As long as these parts are written in a high-level language, the adaptation is simply a matter of running the right compiler for the new system.

Another very important advantage for manufacturers results from the easy bridging from one system to another. Any bridging devices can be designed very easily, since there is no need to invent a "translation" from one system to another; both systems already speak the same language.

B. Benefits for the users of devices and systems Major benefits for users come from the fact that existing knowledge can be reused from one protocol to another, e.g. device profiles and the behavior of devices is identical from one system to another. This results in lower training costs, fewer technical personnel, and the user does not

have to make very large changes to adapt an application from one field bus type to another. It is thus up to the system integrator to choose the field bus that is best suited to his application without having to sacrifice functionality.

A further, very important benefit comes from the ease of bridging and routing within the CIP family. Bridging and routing between non-compatible fieldbuses is always difficult and cumbersome, since it is almost impossible to translate functionality from one fieldbus to another. This is where the full benefits of CIP can be reaped. Forwarding of data and messages from top to bottom and back again is very easy to implement and uses very little system overhead. This results in fast and efficient service.

Finally the very efficient producer/consumer mechanisms used in all CIP protocols result in very fast and efficient use of the transmission bandwidth with the result that system performance is often much higher than with other fieldbuses running at higher raw baud rate.

V. Conclusion The CIP family of protocols is a very versatile set of field bus protocols that are scalable to many applications and many levels of an automation architecture. Due to the universal applicability of the underlying protocol it is very easy to switch from one system to another. The producer/consumer principle together with the open object architecture used in the CIP family allow very efficient use of the communication bandwidth and makes sure that these modern systems can be used for many years to come.

VI. References [1] CIP Common Specification, Release 1.0, © 2000, 2001 by ControlNet International and Open DeviceNet Vendor association. [2] DeviceNet Specification, Release 2.0, including Errata 4, April 1, 2001, © 1995-2001 by Open DeviceNet Vendor Association. [3] ControlNet Specification, Release 2.0, including Errata 2, December 31, 1999, © 1998, 1999 by ControlNet International. [4] EtherNet/IP Specification, Release 1.0, June 5, 2001, © 2000, 2001 by ControlNet International and Open DeviceNet Vendor association. [5] IEC 61131-3:1993 - Programmable controllers – Part 3: Programming languages. [6] ISO 11898:1993 - Road vehicles – Interchange of digital information -- Controller area network (CAN) for high-speed communication. [7] IEEE 802.3:2000, ISO/IEC 8802-3:2000 – IEEE Standard for Information technology – Local and metropolitan area networks – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications

© Institute of Electrical and Electronic Engineers, EFTA 2001

www.rockwellautomation.com

Corporate Headquarters

Rockwell Automation, 777 East Wisconsin Avenue, Suite 1400, Milwaukee, WI, 53202-5302 USA, Tel: (1) 414.212.5200, Fax: (1) 414.212.5201

Headquarters for Allen-Bradley Products, Rockwell Software Products and Global Manufacturing Solutions

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

Europe/Middle East/Africa: Rockwell Automation SA/NV, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

Asia Pacific: Rockwell Automation, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Headquarters for Dodge and Reliance Electric Products

Americas: Rockwell Automation, 6040 Ponders Court, Greenville, SC 29615-4617 USA, Tel: (1) 864.297.4800, Fax: (1) 864.281.2433

Europe/Middle East/Africa: Rockwell Automation, Brühlstraße 22, D-74834 Elztal-Dallau, Germany, Tel: (49) 6261 9410, Fax: (49) 6261 17741

Asia Pacific: Rockwell Automation, 55 Newton Road, #11-01/02 Revenue House, Singapore 307987, Tel: (65) 6356-9077, Fax: (65) 6356-9011