ECHELON®

2.0

Introduction to the LonWorks® Platform

Revision 2

078-0183-01B

Echelon Corporation
www.echelon.com

Contents

Chapter 1.

Introduction

This chapter presents an introduction to the LONWORKS platform and LONWORKS networks, and provides an overview of how to implement a control network.

Overview

Across a broad range of products and systems—from factory automation systems to building controls to embedded machine controls to consumer electronics—there is a trend away from centralized control systems. Manufacturers are building products based on open standard control network architectures that feature intelligent distributed control using a standard communication protocol and readily available off-the-shelf low-cost firmware and transceivers for communication. These open solutions ensure reliability, flexibility, lower cost, and faster development, and provide enhanced energy monitoring and control. This trend is made possible with the emergence of *control networks*—the low-cost alternative to centralized control and proprietary communication systems.

In a centralized control system, remote sensors provided feedback to a microcontroller, programmable logic controller, or other proprietary controller that in turn sends control impulses to relays and other actuators. Each centralized control system has its own unique input/output and processing requirements. Large, complex control systems may be partitioned into two or more centralized systems whose controllers must communicate continuously. These controllers and their attached systems act as islands of automation, with artificially limited communication between the islands. Whether partitioned or not, these control system are expensive to develop, costly to install, and difficult to expand.

In a control network, intelligent control devices communicate using a common protocol. Each device in the control network contains embedded intelligence that implements the common protocol and performs control functions. In addition, each device includes a communication *transceiver* that couples the device with the communications medium.

Devices in a control network may each perform a simple task, or may be more complex devices that perform a multitude of tasks. Devices may be simple sensors and actuators such as proximity sensors, switches, motion detectors, or relays. Devices may also be complex supervisory control and data acquisition systems that monitor other devices on the network and provide supervisory control of the entire system. Although individual devices may execute simple tasks, the system may perform a complex control application, such as running a manufacturing line or automating a building.

Control networks require a different type of networking platform than data processing or office automation applications. Control networks are distinguished by small messages that are frequently transmitted and that require high reliability with low overhead.

For example, a process control system may have a number of pressure and temperature sensors that provide pressure and temperature data to heater controllers. Each heater controller uses the input to set the power output to the heating elements. This system does not move megabytes of data, but does require reliable delivery of the temperature and pressure updates to ensure correct operation.

Many manufacturers understand the benefits of control networks and have attempted to solve these problems by creating their own control network platform. Manufacturers who develop proprietary control networks have a similar problem to

manufacturers who develop communicating centralized controls systems—they find that most of their engineering effort is spent implementing and testing communications systems, rather than developing control features and applications themselves. Ultimately, the high cost of this design approach has limited the market for control systems.

With thousands of application developers and millions of devices installed worldwide, the LONWORKS platform is the leading open solution for building and home automation, industrial, transportation, and public utility control networks. The LONWORKS platform is accelerating the trend away from proprietary control schemes and centralized systems by providing interoperability, robust technology, faster development, and economies of scale. Distributing the processing throughout the network using an open control networking protocol and providing easy access to every device lowers the overall installation and life cycle costs, increases reliability by minimizing single points of failure, and providing the flexibility to adapt the system to a wide variety of applications. For example, in the building control industry, LONWORKS networks are used to provide a common infrastructure for all building systems. This allows the building automation system designer to eliminate excessive vertical integration, which is the often the reason for vertical isolation.

The LONWORKS platform is based on the following concepts:

1. Control systems have many common requirements regardless of application.
2. A networked control system is significantly more powerful, flexible, and scalable than a non-networked control system.
3. Networked control systems can leverage the control system foundation to easily evolve to address new applications, markets, and opportunities.
4. Businesses can save and make more money with control networks over the long term than they can with non-networked control systems.

The Market Value of Networked Control Systems

Many controls applications are following the same evolutionary path followed by the computer industry. The path is from proprietary closed architectures to open standard architectures, and from purpose built implementations with little room for growth to standard platforms with the flexibility to evolve to serve new applications not envisioned by the original designers. This path has enabled the computer industry to grow to exceed the expectations of the original computer designers, and the same will happen for controls.

An example of the evolution from purpose built to general purpose in the computer industry is the evolution of word processing. The first gold standard for word processing was Wang Computer. They offered a purpose-built word processor which quickly replaced the electric typewriters used before the advent of word processing. The solution was superior to electric typewriters, but ultimately purpose-built word processors were replaced by general-purpose networked personal computers that provided not only a superior solution for word processing, but also served as a platform for many applications not envisioned by Wang. Users of the new general purpose solution benefited from word processing, spreadsheet analysis, and more—decoupling the hardware from the application and setting the working model for all networks to come.

An example of the evolution from purpose built to general purpose in the controls industry is the evolution of lighting controls. A typical lighting control system for lighting a city is typically optimized for longevity, reliability, and cost. A typical purpose-built lighting control system relies on dedicated communications, dedicated user controls, and a dedicated monitoring system (if the system supports monitoring). Non-lighting functions such as energy management, 911 emergency response, or dark sky management have seldom, if ever, been incorporated into the function of a streetlighting system due to the cost and complexity of integrating the new capabilities.

The alternative to a purpose-built lighting control system is a networked control system for streetlighting. The network that provides access to the lights is the city's existing communications infrastructure—for example the city's wide-area network or metro Wi-Fi service. The segment controllers manage lights and act as an interface to the ctiy's wide-area network; however, they are also general purpose connectivity products. Any networked product or sensor sharing the common technology of the streetlighting system can be added to the segment controller. This means that the streetlight is only the first end-point on the network, others can be added later—preserving a city's investment in the streetlighting system. For example, after the system is installed, pedestrian sensors could be added to extend the functionality of the system and provide actionable data to the system—lights could brighten when a pedestrian is detected at a crosswalk. Finally, within the lights themselves, embedded communications distributes the intelligence to the end-point. Doing so allows the lights to act independently of, or in cooperation with external information and events. In the case of our lighting system, were the city's WAN to fail, there would be no impact on the streetlighting. Individual lights would still respond to lighting, time, or environmental conditions at the local level. Were pedestrian sensors in place, lights would still balance their output. Also, if the city enabled their parks and recreation reservation software to send time and location data to the streetlighting system, lights could brighten at the beginning and end of night games—providing additional safety and security.

The networked control system provides other benefits as well. City maintenance and operations personnel can view real-time and historical health data, enabling them to provide improved customer service, reduce downtime, eliminate excess spare part inventory, and reduce maintenance costs.

All of these benefits are achieved because the intelligence is at the end-point of the networked control system, the system is responsive to applications and external information, and the networked control system is built on a platform that can easily evolve to support new applications and services.

This document is an introduction to the basics of the LONWORKS platform. It begins with an overview of networks and protocols, provides an overview of the components of the LONWORKS platform, and highlights the technical aspects of the ISO/IEC standard Control Network Protocol (CNP). The remaining sections in this chapter provide an overview of LONWORKS networks and provide an overview of how to implement a control network. Many of the technical details discussed in this document are handled automatically by the protocol, the network operating system, or network tools. The automatic handling of the lower level details of device communication is, in fact, one of the great strengths of the LONWORKS platform.

LONWORKS Networks

The underlying concept of the LONWORKS platform is that the information in a sensing, monitoring, or control application is fundamentally the same across markets and industries. For example, a garage door and a passenger ferry door send essentially the same information—closed or open. A second concept underlying the platform is the knowledge that networks, regardless of their function, increase in power as nodes are added. Metcalf's Law works in data networks and in control networks.

In many ways, a LONWORKS network resembles a traditional data network. Data networks consist of computers attached to various communications media, connected by routers, which communicate with one another using a common protocol such as TCP/IP. Data networks are optimized for moving large amounts of data, and the design of data network protocols assumes that occasional delays in data delivery and response are acceptable. Even though data networks are based on open protocols, most manufacturers do not choose to develop their own data networking components such as transceivers, routers, and network operating systems—it is typically more cost effective to buy these components from a reliable supplier.

As shown in Figure , control networks contain similar components to data networks, but the control network components are optimized for the cost, performance, size, and response requirements of control. Control networks allow networked systems to extend into a class of applications where data networking technology is not appropriate. Manufacturers of control systems and devices are able to shorten their development and engineering time by designing LONWORKS components into their products. The result is cost-effective development and consistency that allows devices from multiple manufacturers to be able to communicate.

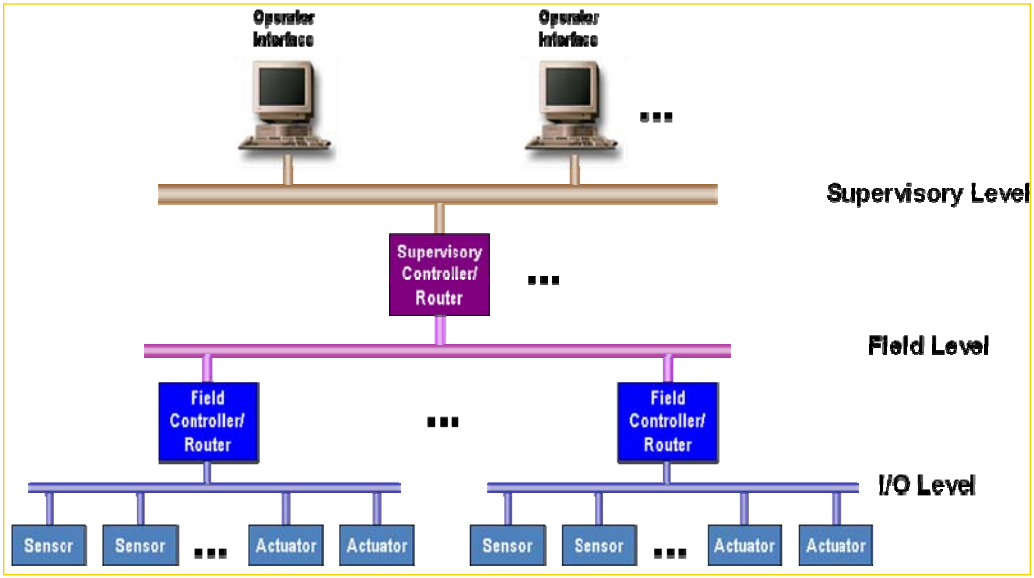


Figure 1 LONWORKS Control Network

LONWORKS networks range in sophistication from small networks embedded in machines to large networks with thousands of devices controlling fusion lasers, paper manufacturing machines, and building automation systems. LONWORKS networks

are used in buildings, trains, airplanes, factories, and hundreds of other processes. Manufacturers are using open, off-the-shelf chips, operating systems, and parts to build products that feature improved reliability, flexibility, system cost, and performance.

Echelon manufactures a wide variety of LONWORKS products to help developers, system integrators, and end-users implement LONWORKS networks. These products provide a complete LONWORKS solution including development tools, network management software, power line and twisted pair transceivers and control modules, network interfaces, routers, controllers, technical support, and training.

Implementing a Control Network

Understanding the power of the LONWORKS platform and leveraging that power by applying it to all control functions is the key to providing the most cost-effective system control solution. Leveraging is achieved by eliminating the walls between control systems and creating a general purpose networked control system that can evolve with changing market requirements. A networked control system leverages a common physical and logical infrastructure to provide holistic system control that can evolve to meet new opportunities and customer needs.

In this case, the entire system is controlled by a single control infrastructure. A standard wiring scheme allows devices to easily access and share communication media. Standard network management services make the network easy to set up, monitor, and control while ensuring compatibility between devices and tools from different manufacturers. Different network control systems may have different needs, however, and different users may have training in different networking tools. The network management standard enables multiple users to use different tools on the same network at the same time. Finally, an application-level standard for the exchange of information between devices exists so devices can easily communicate.

The following sections provide a checklist for designing system-wide open control systems. Following is a summary checklist:

- Cost-effective network wiring
- Compatible devices
- Effective system design
- Standard network management
- Standard network tools

Cost-Effective Network Wiring

The base for a networked control system is cost-effective wiring. Many control systems are created with costly point-to-point wiring or wiring that requires costly connectors, difficult to install network topologies, or costly hubs and switches. The most cost-effective wiring for commercial and industrial networks is a simple pair of twisted-pair wires that can be wired in any topology, is polarity-insensitive, and only requires a single terminator. The most cost-effective wiring for home, utility, outdoor lighting, and transportation networks is to use existing power lines—enabling a networked control system to be installed with no new wires.

Compatible Devices

It is crucial that the devices installed on the common infrastructure share information without effort. So the next checklist item is designing and selecting products adhering to a common communication guideline. This is best achieved within the LONWORKS platform by choosing products that have been certified by LONMARK® International. See the LONMARK Web site at www.lonmark.org for more information.

Effective System Design

Just as a control system implemented on a monolithic processor must be designed taking into account the processing capability of the processor, a networked control system must be designed taking into account the processing capability of the network. Network processing power is typically measured in bandwidth expressed in packets per second, and the intelligent network wiring selected for the first checklist item determines the available bandwidth. Effective system design ensures that each device in a control network uses its appropriate share of bandwidth, and partitions large networks into multiple subnetworks to increase the total available bandwidth.

Standard Network Management

Standard network management provides the necessary network services and published interfaces for the infrastructure. These services allow multiple tools and applications from multiple vendors to coexist on the network. More importantly, it allows the various tools to share the network data. Two solutions are available for LONWORKS networks—the LNS® Network Operating System for commercial, industrial, and transportation systems, and Interoperable Self-Installation (ISI) with optional LonBridge™ Server for home systems.

In the new open marketplace many manufacturers do not want to create entire control systems. These manufacturers simply wish to produce best in class devices. A standard network management solution like LNS or ISI allows the manufacturers of these devices to concentrate on their device and not be concerned about creating an entire control system. This reality, combined with the market presence of LNS and ISI, has caused a proliferation of manufacturers to produce best in class LONWORKS products for use in open systems. These products are the network tools and open devices described below.

Standard Network Tools

Network tools include network integration tools as well as HMI application development tools, data loggers, and other applications with a system-wide view. Network tools can be developed or purchased—in either case they must be compatible with the network management solution selected in the previous checklist item. It is possible to choose any tool for any given project, and it is also possible to combine tools on the same project, as long as they are based on the same network management solution. Tools can be chosen based on functionality and usability rather than who made the physical devices. For commercial and industrial network integration tools, be sure to select tools that support the network operating system

plug-in standard, that fully support LONMARK functional profiles and configuration properties, and that make it easy to reuse parts of your network design. These features are provided by the LonMaker® Integration Tool, which is the most popular network integration tool for open commercial and industrial LONWORKS networks.

Standard Device Configuration

To reduce system cost, a device must not only support standard communication, it must also support a standard interface for configuration. Again, the LONMARK logo on the device is the first place to look. The LONMARK guidelines provide for the physical layer requirements of devices as well as the common functional profiles, data types, configuration capabilities, and installation methodologies.

For simple devices, it may be adequate for product manufacturers to simply document the configuration interface for their device. However, it's often better, and required for complex devices, if they encapsulate the knowledge into a plug-in that can be run inside one of the network integration tools. The plug-in must be based on the selected network management solution. This allows tools from other manufacturers to install and configure the device quickly and easily.

Standard IP Support

The Internet Protocol (IP) suite is the standard on which the Internet is built. A control network must often provide for encapsulation of the control system messages or packets into IP packets. Messages can then be passed around the world without translation into foreign protocols. The cost of transmission is minimal and the ability to leverage existing infrastructure is practically limitless. IP support for a LONWORKS network can be provided either by an LNS Server, *i*.LON SmartServer, or LonBridge Server that is attached to both the LONWORKS network and the IP network.

Getting More Information

This book is organized into three chapters. Chapter 1 introduced the LONWORKS platform and its benefits. Chapter 2 describes the components that make up the LONWORKS platform and describes how these components can accelerate development of LONWORKS devices and systems. Chapter 3 describes the ISO/IEC 14908-1 Control Network Protocol (CNP) that is the foundation of the LONWORKS platform . Chapter 3 also gives a detailed look at layers 1 through 7 of the protocol. Developers will mostly deal with layers 6 and 7, but will also be interested in the transceiver information described in layer 1. System designers and integrators will be interested in the same layers as developers, and will also be interested in understanding the options provided by layer 4.

The addendix provides details on layer 1 that will only be of interest to transceiver designers.

For more information on the LONWORKS platform, see the documentation available at www.echelon.com/docs.

Chapter 2.

Platform Components

This chapter describes the components that make up the LONWORKS platform, and describes how these components work together to reduce development, production, installation, and system costs.

Building a Platform

Echelon Corporation invented the LONWORKS platform and the ISO/IEC 14908-1 Control Network Protocol. Thousands of control manufacturers currently manufacture LONWORKS devices. A partial listing of these devices is available at www.lonmark.org/products.

Echelon began the development of the LONWORKS platform and the Control Network Protocol in 1988. This initial vision continues to drive the company forward; first, by creating a standard, cost effective method to allow inexpensive control devices to communicate with each other effortlessly. Then, by using the standard communication capabilities to allow devices from multiple vendors to easily interoperate on the same network. Echelon understood that simply developing a protocol specification would not achieve the goal of multi-vendor systems. It was necessary to build a cost-effective, standard method through which the protocol could be used and supply all the necessary development tools and networking products.

The overriding goal of the LONWORKS platform is to make it easy and cost effective to build open control systems. Echelon developed the LONWORKS platform believing there were three fundamental issues that had to be addressed to create interoperable products in the control market. First, a protocol optimized for control networks, but generic in its ability to work with different types of controls had to be developed. Second, the cost to incorporate and deploy this protocol in devices had to be competitive. Third, the protocol had to be introduced in such a way that implementation would not vary by vendor as this would destroy interoperability.

In order to address all of these issues effectively, Echelon Corporation set out to create a complete platform for designing, creating, and installing intelligent control devices. The first step was achieved through the creation of the ISO/IEC 14908-1 Control Network Protocol, which is described in Chapter 3, *The Control Network Protocol*. Addressing the cost and deployment issues meant finding an economical way to provide implementations of the protocol to customers along with development tools. The goal of the LONWORKS platform is to provide a well-integrated, optimally designed, and economical platform for creating smart devices and networks. This chapter describes the following components that make up the LONWORKS platform:

- Smart Transceivers
- Development Tools
- Routers
- Network Interfaces
- Internet Servers
- Network Management
- Network Tools

Smart Transceivers

In order to achieve economical and standardized deployment, Echelon designed the *Neuron® core*. The Neuron name was chosen to point out the similarities between proper network control implementation and the human brain. There is no central point of control in the brain. Millions of neurons are networked together, each

providing information to others through numerous paths. Each neuron is typically dedicated to a particular function, but loss of any one does not necessarily affect the overall performance of the network.

The Neuron core is available as a standalone component called the *Neuron Chip*. To further reduce device costs, Echelon also provides Neuron cores combined with communication transceivers, which are called *Smart Transceivers*.

To the developer and the integrator, the beauty of the Neuron core and Smart Transceivers lies in their completeness. The built-in communication protocol and processors removes the need for any development or programming in these areas. The Smart Transceivers eliminate the need to develop or integrate a communications transceiver. The Neuron core provides layers 2 through 6 of the ISO/OSI reference model of a communication protocol, and the Smart Transceiver adds layer 1. The device manufacturer only has to supply the application layer programming and the network integrator provides the configuration for a given network installation. This standardizes implementation and makes development and configuration simple and fast.

Most LONWORKS devices take advantage of the functions of the Neuron core and use it as the control processor. The Neuron core is a semiconductor component specifically designed for providing intelligence and networking capabilities to low-cost control devices. The Neuron core includes up to four processors that provide both communication and application processing capabilities. Two processors execute the layer 2 through 6 implementation of the ISO/IEC 14908-1 protocol and the third executes layer 7 and the application code. LONWORKS 2.0 Neuron cores add a fourth processor for interrupt processing. The device manufacturer provides application code to run on the Neuron core and I/O devices to be connected to the Neuron core. Echelon Corporation designed the original Neuron core and the LONWORKS 2.0 Neuron core. Neuron cores are also designed and manufactured by Echelon's manufacturing partners.

The Neuron core is a system-on-a-chip with multiple processors, memory, and communication and I/O subsystems. At the time of manufacture, each Neuron core is given a permanent unique-in-all-the-world 48-bit code, called the *Neuron ID*. A large family of Neuron Chips is available with differing speeds, memory type and capacity, and interfaces. Approximately 30 million Neuron cores have been shipped as of early 2009.

A complete operating system including an implementation of the ISO/IEC 14908-1 protocol, called the *Neuron firmware*, is available for the Neuron core.

Development Tools

Echelon provides a broad range of tools for developing LONWORKS devices and applications. Following is a summary of the development tools for LONWORKS devices:

- *Mini FX Evaluation Kit*—tools and evaluation boards for evaluating the LONWORKS platform. The Mini kit can be used to develop simple LONWORKS applications for a Neuron Chip or Smart Transceiver, but it does not include a debugger, project manager, or network integration tool required by many devices.

- *NodeBuilder® FX Development Tool*—tools and evaluation boards for developing simple or complex LONWORKS applications for a Neuron Chip or Smart Transceiver. Includes a debugger, project manager, and network integration tool.
- *ShortStack® Developer's Kit*—tools and firmware for developing LONWORKS applications that run on processors that do not include a Neuron core. The ShortStack kit includes firmware that is loaded onto a Smart Transceiver that makes the Smart Transceiver a communications co-processor for the host processor.
- *FTXL Developer's Kit*—tools, firmware, and FPGA design files for developing LONWORKS applications that run on a Nios II embedded 32-bit RISC processor configured on an Altera Cyclone II/III family FPGA device.

Developers using any of these tools typically also require network integration and diagnostic tools. A network integration tool is included with the NodeBuilder FX Development Tool, but the other LONWORKS development tools do not include a network integration tool. None of the development tools include a network diagnostic tool. Network integration and diagnostic tools are described in *Network Tools* later in this chapter.

Routers

Transparent support for multiple media is a unique capability of the LONWORKS platform, allowing developers and network integrators to choose those media and communication methods best suited for their needs. Multiple media support is made possible by *routers*. Routers can also be used to control network traffic and partition sections of the network from traffic in another section, increasing the total throughput and capacity of the network. Network tools automatically configure routers based on network topology, making the installation of routers easy for installers and transparent to the devices.

Routers allow a single peer-to-peer network to span many types of transport media and support tens of thousands of devices. A router has two sides, each with a transceiver appropriate to the two channels to which the router is connected. Routers are completely transparent to the logical operation of the network, but they do not necessarily transmit all packets; when configured by a network integration tool, intelligent routers know enough about the system configuration to block packets that have no addressees on the far side. Using another type of router called an IP-852 router, LONWORKS routers can span great distances over wide-area networks such as the Internet.

Echelon offers routers that connect different types of twisted pair channels, as well as IP-852 routers for routing between twisted pair channels and an IP network such as the Internet, an intranet, or a virtual private network (VPN). A complete list of the routers available from Echelon is available at www.echelon.com/products. Following is a list of the most commonly used routers:

- *i.LON SmartServer*—routes between an IP-852 channel and a twisted pair channel, and also provides controller functionality. The SmartServer is available with and without IP-852 routing.
- *i.LON 600*—routes between an IP-852 channel and a twisted pair channel.

- *MPR-50 Multi-Port Router*—routes among up to 5 channels: four TP/FT-10 free topology twisted pair and one TP/XF-1250 twisted pair.
- *LonPoint Router*—routes between two twisted pair channels. Six different models are available for different types of twisted pair channels.

Network Interfaces

A *network interface* is a card or module that is used to connect a host computer—typically a computer running Microsoft Windows—to a LONWORKS network. The network interface itself does not run an application—instead it provides either layer 2 or layers 2 – 5 of the ISO/IEC 14908-1 Control Network Protocol (CNP), plus a transceiver implementing layer 1, and firmware to exchange layer 2 or layer 5 CNP packets with the attached computer. A complete list of the network interfaces available from Echelon is available at www.echelon.com/products. Following is a list of the most commonly used network interfaces:

- *U10/U20 USB Network Interface*—a small USB device that plugs into a USB port on a Windows computer and attaches to a free topology twisted pair (for the U10) or power line (for the U20) channel.
- *i.LON SmartServer*—a controller that can also be used as a network interface with an IP connection to a host Windows computer.
- *i.LON 600*—an IP-852 router that can also be used as a network interface with an IP connection to a host Windows computer.

Smart Servers

A *smart server* is a programmable device that combines a controller with a Web server for local or remote access, LONWORKS network interface, CNP network manager, legacy-device interface, and optional IP-852 router. Echelon's low-cost smart server is called the *i.LON SmartServer*. The SmartServer connects LONWORKS, Modbus, and M-Bus devices to corporate IP networks or the Internet. It features a built-in Web server that allows Web access to all the data managed and controlled by the SmartServer, as well as built-in applications for alarming, scheduling, data logging, and data translation. It also includes a Web binder application for bridging multiple LONWORKS domains, as well as bridging from Modbus and M-Bus devices to LONWORKS domains. The SmartServer provides a SOAP/XML Web services interface for use by custom Web pages and for integration with enterprise applications.

Network Management

LonWorks networks can be categorized by the method used to perform network installation. The two categories of networks are *managed networks* and *self-installed* networks. A managed network is a network where a shared *network management server* is used to perform network installation. The network management server may be part of a network operating system as described later in this section, or may be part of an Internet server such as the SmartServer. A user typically uses a tool to interact with the server and define how the devices in the network should be

configured and how they should communicate. Such a tool is called a *network management tool* and is described in the next section. Although a network management tool and a server are used to initially establish network communication, they need not be present all the time for the network to function. The network management tool and server are only required whenever changes are made to the network's configuration.

In a managed network, the network management tool and server allocate various network resources, such as device and data point addresses. The network management server is also aware of the network topology, and can configure devices for optimum performance within the constraints of the topology.

The alternative to a managed network is a self-installed network. There is no central tool or server that manages the entire network configuration in a self-installed network. Instead, each device contains code that replaces parts of the network management server's functionality, resulting in a network that no longer requires a special tool or server to establish network communication or to change the configuration of the network.

Network installation includes the following steps:

- Assigning logical addresses to all devices and groups of devices.
- Binding the network variables to create logical connections between devices.
- Configuring the various control network protocol parameters in each device for the desired features and performance, including channel bit rate, acknowledgement, authentication, and priority service.

Network installation may be quite complex, but the complexity is hidden by the network management solutions that are part of the LONWORKS platform. For managed applications, functional network design can be as simple as dragging the devices' application functional blocks onto a drawing and connecting inputs and outputs to determine how functional blocks communicate with each other. The network management tool automatically assigns logical addresses, binds network variables based on the connections drawn by the integrator, and configures network control parameters. For self-installed applications, functional network design is as simple as plugging in a pair of devices and pressing a button on each device to establish communication.

The network installation process for managed networks can be either an ad hoc process or a pre-engineered process. The network installation process for self-installed networks is typically an ad hoc process. In the *ad hoc* method, the devices are first connected to the network and powered-up, and the configuration data is either self-installed or downloaded over the network as each device is defined in a network integration tool. In the *engineered* method, the information is collected into a database by the network integration tool and is downloaded to the devices at installation time. For a managed network using either method, the network integration tool automatically maintains a database that accurately reflects the configuration of each device in the system.

Networks can start out as self-installed networks using ISI and, as size or complexity grows beyond the ISI limits, can be upgraded into a managed network. A self-installed network may also be transitioned to a managed network to take

advantage of the additional flexibility and capability provided by a network management tool and server.

Network Operating System

For managed networks, a network operating system (NOS) can be used to provide a common, network-wide set of services supporting monitoring, supervisory control, installation, and configuration. The NOS also provides programming extensions for easy use of network management and maintenance tools. A LONWORKS NOS additionally provides data access services for HMI and SCADA applications as well as remote access via LONWORKS or IP networks.

A properly designed NOS allows for synchronization services between multiple tools and applications used by a single or multiple users. In order for a NOS to support complete interoperability, it must support standard plug-ins by multiple manufacturers for easy device configuration.

A properly designed NOS is not required for the normal operation of a system. The NOS provides installation and maintenance services when a network is initially commissioned or later changed, but once a network is installed, the NOS is not required to support communication between devices. This is a significant benefit of the peer-to-peer architecture of the LONWORKS platform. Managed networks that are not peer-to-peer utilize a system-to-system (gateway-to-gateway) approach to integration and do not utilize a NOS. These system-to-system solutions are not capable of separating their functionality from their hardware design. Instead, such solutions rely on multiple, custom interfaces to achieve a partial separation of application from design.

To provide interoperability between network tools and applications, the LONWORKS platform includes a single NOS for managed networks called the *LNS Network Operating System*. LNS provides a standard platform for supporting interoperable applications on LONWORKS networks. LNS is an infrastructure that provides the foundation for interoperable LONWORKS network tools and applications, which are products used in designing, configuring, installing, operating, and maintaining LONWORKS networks. LNS supports clients and servers based on Microsoft Windows.

LNS uses a client/server architecture so that multiple applications can be active on a network at the same time, allowing multiple users to install devices, operate a system, diagnose problems, and make repairs simultaneously. LNS is scalable, changeable, and upgradeable.

The LNS plug-in standard encourages LONWORKS device manufacturers to provide more value to users through software components linked to their unique products. Rather than trying to develop custom programming for each project in the field, network integrators use plug-ins that configure the devices used in the project. These device plug-ins often contain built-in troubleshooting tools, user dialogs to aid or confirm configuration choice, as well as custom user interfaces to monitor or graph data held in the device. In effect, manufacturers can write smart software once to simplify the use of their products in thousands of LONWORKS networks.

Using LNS, a manufacturer's device plug-in software runs without modification in any Windows computer, and can be seamlessly integrated with the installation tools on the computer. LNS plug-ins simplify the management of the network by masking

the underlying communication mechanisms between the software component and the device. Thus, many existing devices can become fully interoperable by simply writing a plug-in. A standard interface is set for manufacturers to customize the front end, while LNS makes it possible for multi-vendor software components to work together.

Interoperable Self-Installation (ISI)

Each device in a self-installed network is responsible for its own configuration and does not rely on a network management server to coordinate its configuration. Because each device is responsible for its own configuration, a common standard is required to ensure that devices configure themselves in a compatible way. The standard protocol for performing self-installation with the LONWORKS platform is called the *LONWORKS Interoperable Self-Installation (ISI) Protocol*. The ISI protocol can be used for networks of up to 200 devices and enables LONWORKS devices to discover and communicate with each other. Larger or more complex networks must either be installed as managed networks, or must be partitioned into multiple smaller subsystems, where each subsystem has no more than 200 devices and meets the ISI topology and connection constraints. Devices that conform to the LONWORKS ISI protocol are called *ISI devices*.

The LonBridge Server can be used to create an interface between an IP network and a network of ISI devices. The LonBridge Server is available for Windows and Linux, and can be freely ported to other platforms.

Network Tools

Network tools are software applications built on top of the network operating system for network design, installation, configuration, monitoring, supervisory control, diagnostics, and maintenance. Many tools combine these functions, but the most common combinations are the following:

- *Network Integration Tools*. Provide the essential functions required to design, configure, commission, and maintain a network.
- *Network Diagnostic Tools*. Special-purpose tools to observe, analyze, and diagnose network traffic and monitor network loading.
- *HMI Development Tools*. Tools for creating human-machine interface (HMI) applications. HMI applications are used for operator interfaces to operational systems.
- *I/O Servers*. General-purpose drivers that provide access to LONWORKS networks for HMI applications not originally designed for LONWORKS networks.

Network tools based on the LNS Network Operating System are interoperable, meaning they can operate at the same time on the same network and maintain a consistent view of the devices in the network and their configuration. Echelon's offerings for network tools include the LonMaker Integration® Tool and the LonScanner™ Protocol Analyzer, which are described in the following sections.

LonMaker Integration Tool

The LonMaker Integration Tool is a software package for designing, documenting, installing, and maintaining multi-vendor, open, interoperable LONWORKS networks. Based on the LNS Network Operating System, the LonMaker tool combines a powerful client-server architecture with an easy-to-use Visio user interface. The result is a tool that is sophisticated enough to design, commission, and maintain a distributed control network, yet provide the ease-of-use required by network design, installation, and maintenance staff.

The LonMaker tool conforms to the LNS plug-in standard. This standard allows LONWORKS device manufacturers to provide customized applications for their products, and have these customized applications automatically started when the LonMaker user selects the associated device. This makes it easy for system engineers and technicians to define, commission, maintain, and test the associated devices.

For engineered systems, network design is usually done off-site, without the LonMaker tool attached to the network. Network design may, however, take place on-site, with the tool connected to a commissioned network. This feature is especially desirable for smaller networks or where adds, moves, and changes are a regular occurrence.

Users are provided with a familiar, CAD-like environment for designing a control system. Visio's smart shape drawing feature provides an intuitive, simple means for creating devices. The LonMaker tool includes a number of smart shapes for LONWORKS networks, and users can create new custom shapes. Custom shapes may be as simple as a single device or functional block, or as complex as a complete subsystem with predefined devices, functional blocks, and connections between them. Using custom subsystem shapes, additional subsystems can be created by simply dragging the shape to a new page of the drawing, a time-saving feature when designing complex systems. Any subsystem can be changed to a supernode by adding network variables to the subsystem shape. Supernodes reduce engineering time by exposing a simplified interface to a set of devices.

Network installation time is minimized by the ability of the installer to commission multiple devices at the same time. Devices can be identified by service pin, bar code scanning Neuron IDs, winking, or manually entering the IDs. Auto discovery can be used for systems containing embedded networks to automatically find and commission the devices in the system. Testing and device configuration is simplified by an integrated application for browsing network variables and configuration properties. A management window is provided to test, enable/disable, or override individual functional blocks within a device or to test, wink, or set online and offline states for devices.

The LonMaker tool can both import and export AutoCAD files and generate as-built documentation. An integrated report generator and bill-of-materials generator can also be used to generate detailed reports of the network configuration.

The LonMaker tool is a single expandable tool covering the entire life cycle of the network to simplify the tasks of installers.

LonScanner Protocol Analyzer

The LonScanner Protocol Analyzer is a software package that provides network diagnostic tools to observe, analyze, and diagnose the behavior of installed LONWORKS networks.

The protocol analyzer can be used to collect, timestamp, and save all CNP packets on a LONWORKS channel. Packets are saved in log files that can be later viewed and analyzed; packets may also be viewed in real-time as they are collected by the protocol analyzer.

A sophisticated transaction analysis system examines each packet as it arrives and associates related packets to aid the user in understanding and interpreting traffic patterns in their network.

Logs can be displayed in summary form with one packet per line for quick analysis, or in expanded form with one packet per window for more detailed analysis. Using data imported from an LNS database, the protocol analyzer decodes and displays packet date using the device and network variable names assigned during installation. It also provides text descriptions of each message and a description of the CNP message service used to transmit it. Eliminating the need for the user to manually interpret the ones and zeros of CNP reduces the time and effort needed to diagnose network problems.

The user can specify capture filters to limit the packets collected. Filters can be used to limit the captured packets to packets between selected devices or network variables, or to packets using selected CNP services.

A traffic statistics tool provides access to detailed statistics related to network behavior. The statistics include total packet counts, error packet counts, and network loading. The statistics display provides the user with an easy-to-read summary of network activity.

Chapter 3.

The Control Network Protocol

The ISO/IEC 14908-1 Control Network Protocol (CNP) is the foundation of the LONWORKS platform and provides a reliable, cost-effective, and robust communications standard for control applications. This chapter describes the protocol and the services implemented by each layer of the protocol. Developers will mostly deal with layers 6 and 7, but will also be interested in the transceiver information described in layer 1. System designers and integrators will be interested in the same layers as developers, and will also be interested in understanding the options provided by layer 4.

The appendix provides details on layer 1 of the protocol that will only be of interest to transceiver designers.

ISO/IEC 14908-1 Control Network Protocol

The foundation of the LONWORKS platform is the ISO/IEC 14908-1 Control Network Protocol (CNP), just as the RFC-791/793 TCP/IP protocol is the foundation of most data networks and the Internet. There are many compatible implementations of the ISO/IEC 14908-1 protocol. Echelon's implementation of CNP is called the *LonTalk® protocol*. Throughout this document, the ISO/IEC 14908-1 Control Network Protocol is referred to simply as *CNP*. CNP also refers to the ISO/IEC physical layer standards defined by ISO/IEC 14908-2 and 14908-3, and the IP tunneling standard defined by ISO/IEC 14908-4.

CNP is designed to support the needs of control applications spanning a range of industries and requirements. To meet its broad objectives, the protocol is a complete seven-layer communications protocol, with each layer optimized to the needs of control applications. The seven layers follow the reference model for open systems interconnection (OSI) developed by the International Standard Organization (ISO). By addressing all seven layers defined by the OSI reference model, CNP provides a robust communications solution that meets the needs of a broad range of applications today, and will continue to meet the needs of evolving control applications in the future.

The following list summarizes the major features of CNP:

- ***Efficient delivery of small messages***. A typical control message may consist of 1 to 8 bytes of data, though larger and smaller messages are supported. A CNP device can transmit a message with as few as 9 bytes of protocol overhead. Messages may be delivered to a single device or to any group of devices.
- ***Reliable delivery of messages***. Even though an individual message may consist of a few bytes, the reliable delivery of every message may be critical to the application. CNP includes reliable message delivery services that retry a message transmission when a communication failure occurs and informs the sending application if an unrecoverable failure occurs. Resynchronization is immediate if a previously unreachable destination becomes reachable within the retry interval.
- ***Duplicate message detection***. Some types of control messages must not be delivered multiple times. For example, if a monitoring application that is counting events were to receive duplicate messages, the event count would become incorrect. CNP prevents duplicate messages from being passed to the receiving application.
- ***Multiple communications media***. Many control systems require multiple communications media for lowest total system cost. Twisted-pair cable provides the best performance and is a good solution where it is practical to install the cable. Link power communications lowers device costs by providing power over the same media as communications. Communications over existing power lines provides the lowest installation cost. CNP is media independent so that all these media, and more, are supported. In addition, CNP also supports routers so that devices on different channels can interoperate. A hierarchical addressing scheme is used to support low-cost and easily maintained routers.
- ***Low device cost***. A control device may be as simple as a sensor for a single point such as a limit switch or a temperature sensor. With the capability to put control in every point, it is important that the protocol not be too demanding in terms of memory and computing resources at each device. CNP is optimized to minimize

the code size of protocol firmware and to minimize RAM requirements for buffer storage. For example, the complete implementation of CNP running on a Neuron core requires less than 10 Kbytes of code and less than 1 Kbyte of RAM.

- ***Low installation and maintenance cost***. Low device cost does not lead to low system cost unless devices are easily installed in networks, networks are easily modified, and repairing networks after a failure is simple. CNP includes complete support for installation and maintenance so that simple devices in simple networks can manage their own network installation, low-cost installation tools can be used to install and maintain more complex networks, and low-cost diagnostic tools can be used to diagnose failures in any type of network.
- ***Efficient use of channel bandwidth***. To keep system costs down, many devices must be able to efficiently share a single communications channel. CNP uses innovative media access technologies to provide the most efficient use of the communications channels, even under conditions of high loading.
- ***Interoperable systems***. Multiple systems may need to interoperate to provide additional benefits to the end users. For example, a fire alarm system may interoperate with an elevator control system to keep elevators away from burning floors and may interoperate with an emergency exit lighting system to illuminate exit signs. In addition to supporting interoperability between devices, CNP also allows devices from different vendors to be installed using a common set of installation strategies and tools.
- ***Separation of systems***. Multiple systems that should not interoperate may share a common communication medium, especially in the case of open media such as power line. CNP allows these systems to operate independently without mutual interference by isolating devices that communicate with each other into domains. A *domain* is a logical collection of devices that communicate with each other. Devices in different domains cannot communicate directly with each other—they must communicate through a common gateway.
- ***Prevent tampering***. Because control systems do more than just move data, it should not be possible for an unauthorized user to inject commands into the network. CNP includes an authentication protocol that prevents tampering by unauthorized users.

CNP Layers

To provide a low-cost, reliable, and robust communications standard, CNP is layered as recommended by the International Standards Organization Open Systems Interconnect (ISO OSI) reference model. The OSI layers ensure that the required services are provided without unexpected interactions between the services.

CNP provides the following services for each of the seven layers of the OSI reference model:

1. The *physical layer* defines the transmission of raw bits over a communication channel. A *channel* is a physical transport medium for packets. The physical layer ensures that a 1 bit transmitted by a source device is received as a 1 bit by all destination devices. CNP is media independent, so multiple physical layer protocols are supported depending on the communication medium.
2. The *link layer* defines media access methods and data encoding to ensure efficient use of a single communications channel. The raw bits of the physical layer are combined into data frames. The link layer defines when a source device

can transmit a data frame, and defines how destination devices receive the data frames and detect transmission errors. A priority mechanism is also defined to ensure priority delivery of urgent messages.

3. The *network layer* defines how message packets are routed from a source device to one or more destination devices. This layer defines naming and addressing of devices to ensure the correct delivery of packets. This layer also defines how messages are routed between the source and destination devices when these devices are on different communication channels.

4. The *transport layer* ensures reliable delivery of message packets. Messages can be exchanged using an acknowledged service, where the sending device waits for an acknowledgement from the receiver and resends the message if the acknowledgement is not received. The transport layer also defines how duplicate messages are detected and rejected if a message is resent due to a lost acknowledgement.

5. The *session layer* adds control to the data exchanged by the lower layers. It supports remote actions so that a client may make a request to a remote server and receive a response to this request. It also defines an authentication protocol that enables receivers of a message to determine if the sender is authorized to send the message.

6. The *presentation layer* adds structure to the data exchanged by the lower layers by defining the encoding of message data. Messages may be encoded as network variables, application messages, or foreign frames. Interoperable encoding of network variables is provided with *standard network variable types* (*SNVTs*). Presentation layer services are provided by the Neuron firmware for applications hosted on a Neuron Chip or Smart Transceiver; these services are provided by a host processor and a LONWORKS network interface for applications running on other hosts.

7. The *application layer* defines standard network services that use data exchanged by the lower layers. Standard network services are provided for network configuration, network diagnostics, file transfer, application configuration, application specification, alarming, data logging, and scheduling. These services ensure that devices created by different developers or manufacturers can interoperate with each other, and can be installed and configured using standard network tools.

Table 1 summarizes the OSI reference model layers and the CNP services provided at each layer.

Table 1 CNP Layers

OSI Layer		***Purpose***	***Services Provided***
1	Physical	Electrical Interconnect	Media-Specific Interfaces and Modulation Schemes (twisted pair, power line, radio frequency, coaxial cable, infrared, fiber optic)
2	Link	Media Access and Framing	Framing; Data Encoding; CRC Error Checking; Predictive CSMA; Collision Avoidance; Priority & Collision Detection

OSI Layer		Purpose	Services Provided
3	Network	Message Delivery	Unicast & Multicast Addressing; Routers
4	Transport	End-to-End Reliability	Acknowledged & Unacknowledged Message Delivery; Common Ordering; Duplicate Detection
5	Session	Control	Request-Response; Authentication
6	Presentation	Data Interpretation	Network Variables; Application Messages; Foreign Frame Transmission
7	Application	Application Compatibility	Network Configuration; Network Diagnostics; File Transfer; Application Configuration; Application Specification; Alarming; Data Logging; Scheduling

CNP Data Transmission

CNP ensures that application data sent by a source device is correctly received by the applications on one or more destination devices. CNP layers described in the previous section define transformations on the sending device's application data that occur between the application and the raw bits that are transmitted on the communication channel; the CNP layers define a similar set of transformations on the destination devices that change the raw bits back to the original data. The transformations performed by layers 2 through 6 consist of headers that are added to the application data. Table 2 summarizes the data carried by each layer.

Table 2 Data at each CNP Layer

Layer		Data
1	Physical	Raw Bits
2	Link	Data Frame
3	Network	Datagram
4	Transport	Transport Packet
5	Session	Session Packet
6	Presentation	Presentation Packet
7	Application	Message

Figure illustrates the contents of a typical CNP data frame as carried by the physical layer.

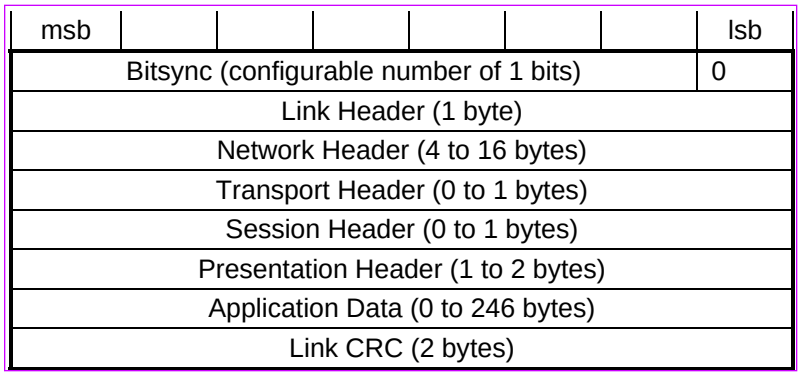


Figure 2 CNP Data Frame

The format of the headers for each layer is described in the sections within this chapter. Each section includes diagrams similar to the preceding diagram that describes the bit and byte format of the fields within the data frame. Bytes are transmitted from top to bottom, with multi-byte fields transmitted least significant byte first. Bits are transmitted from right to left within a byte, which is least significant bit first.

CNP Limits

CNP supports up to 32,385 devices per domain, and a practically unlimited number of domains (the domain identifier may be 0, 1, 3, or 6 bytes long). Each CNP device has a unique 6-byte identifier and can also be configured to recognize logical addresses using *subnet*, *node*, and *group* identifiers as described in the layer-3 documentation in Chapter 3. Devices typically communicate with data values called *network variables* that are described in the layer-6 documentation in Chapter 3. To support flexible connection topologies for network variables, CNP defines *network variable aliases* as describe in the layer-6 documentation in Chapter 3. Any CNP device can monitor and update any number of network variables in a domain, but network variable reads and writes are simplified by using network variable connections as described in the layer-6 documentation in Chapter 3—these connections require bindable network variables and network variable aliases. The following limits apply per CNP domain. These concepts are described in more detail in Chapter 3. The network variable and network variable aliases may be further limited by the CNP implementation on a device.

- A maximum of 32,385 devices per domain.
- A maximum of 255 subnets per domain.
- A maximum of 256 groups per domain.
- A maximum of 127 devices per subnet.
- A maximum of 64 devices per group for acknowledged services
- A maximum of 32,385 devices per group for unacknowledged and repeated services.
- A maximum of 4096 bindable network variables per device.
- A maximum of 8192 network variables aliases per device.

Layer 1—Physical Layer

The physical layer defines the transmission of raw bits over a communication channel. A *channel* is a physical transport medium for packets. The physical layer ensures that a 1 bit sent by a source device is received as a 1 bit by all destination devices.

CNP is media independent, so multiple physical layer protocols are supported depending on the communication medium. A LONWORKS device can be connected to a variety of communications transceivers that manage the electrical interconnection to the communications medium. CNP communications transceivers are available for communication over twisted pair, link power, power line, radio frequency (RF), fiber optic, and infrared media. The specifications for each LONWORKS transceiver provide the distance, bit rates, and topologies supported.

A LONWORKS network is composed of one or more channels. The physical form of a channel depends on the medium. For example, a twisted pair channel is a twisted pair cable; an RF channel is a specific radio frequency carrier; a power line channel is a specific band carried on a contiguous section of power wiring.

Multiple channels are connected by *routers*. Routers are communication devices that connect two channels and route packets between them. Routers can be installed to use one of four routing algorithms: *configured router*, *learning router*, *bridge*, or *repeater*. Configured routers and learning routers are a class of router known as an *intelligent router*.

A set of channels connected by bridges or repeaters is a *segment*. A device sees every packet from every other device on its segment. Intelligent routers can be used to isolate traffic within a segment to increase total system capacity and improve reliability.

The bit rate of a channel is dependent upon the medium and transceiver design. Multiple transceivers with different bit rates may be designed for a medium to allow trade-offs of distance, throughput, and device power consumption and cost. Typical bit rates are 78.1 kbps for TP/FT-10 (ISO/IEC 14908-2) and Ethernet bit rates for IP-852 (ISO/IEC 14908-4).

Channel Types

Table 3 lists the common LONWORKS communication channel types. All of these media are bidirectional, supporting transmission and reception of data by every device. The ID column in the table lists unique identifiers for each channel type that can be used by a network tool to verify compatibility between the device and its channel. The Description column in the table lists the specification that defines the characteristics of the channel. The Standard column in the table identifies the channel types that can be used to create devices that can be certified by LONMARK International. Additional standard channel types are listed at www.lonmark.org/mid.

Table 3 Common LONWORKS Channel Types

Name	ID	Media	Bit Rate	Definition	Standard
IP-852	154	IP-852 Tunneling	N/A	ISO/IEC 14908-4	Yes
PL-20A	15	CENELEC A-band Power Line	2613bps	ISO/IEC 14908-3	Yes
PL-20C	16	CENELEC C-band Power Line w/access protocol	156.3k/3987bps	ISO/IEC 14908-3	Yes
PL-20N	17	CENELEC C-band Power Line w/o access protocol	156.3k/3987bps	ISO/IEC 14908-3	Yes
TP/FT-10	4	Free Topology Twisted Pair	78.13kbps	ISO/IEC 14908-2	Yes
TP/XF-1250	3	Transformer-Isolated Twisted Pair	1.25Mbps	LONMARK Interoperability Guidelines	Yes

See the documents listed in the Definition column in Table 3 for more information on any of the channel types.

TP/FT-10 Free Topology Twisted Pair

A conventional control system using bus topology wiring (such as RS-485) consists of a network of sensors and control outputs that are interconnected using a shielded twisted wire pair. In accordance with RS-485 guidelines, all of the devices must be wired in a bus topology to limit electrical reflections and ensure reliable communications. There is a high cost associated with installing and maintaining the cable plant that links together the many elements of an RS-485-based control system. Bus topology wiring is more time consuming and expensive to install because the installer is unable to branch or star the wiring where convenient: all devices must be connected directly to the main bus.

The best solution for reducing installation and maintenance costs and simplifying system modifications is a free topology communication system. The TP/FT-10 channel type offers just such a solution, and provides an elegant and inexpensive method of interconnecting the different elements of a distributed control system.

The TP/FT-10 channel type uses transceivers that conform to the ISO/IEC 14908-2 *Free-Topology Twisted Pair Channel Specification* on free-topology twisted pair media and supports a 78,125 bps bit rate. These transceivers encode data using differential Manchester encoding as described earlier in this chapter in *Single-Ended Mode*, which is polarity insensitive. A TP/FT-10 channel consists of up to 64 devices on a single network segment; or 128 devices along with a link power source, which supplies DC power to the devices on the channel. The total network length and number of devices may be extended by use of CNP routers, and/or one TP/FT-10 physical-layer repeater.

A free topology architecture allows the user to wire the control devices with virtually no topology restrictions. Figure 3 illustrates a typical network using a TP/FT-10 channel. In this example, power is supplied by a local +5VDC power supply located

at each device. Power may optionally be carried over the same pair of wires as the data.

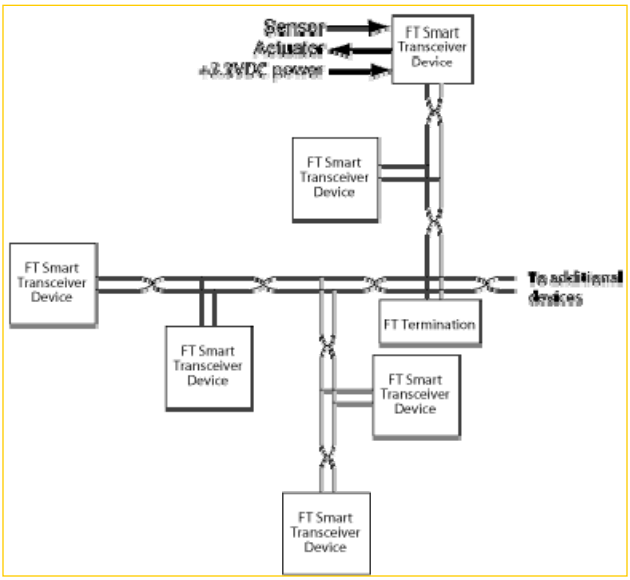


Figure 3 Example TP/FT-10 Channel Wiring

Unlike bus wiring designs, a TP/FT-10 channel uses a wiring scheme that supports any combination of star, loop, and bus wiring. Figure 4 illustrates the channel topologies supported by the TP/FT-10 channel.

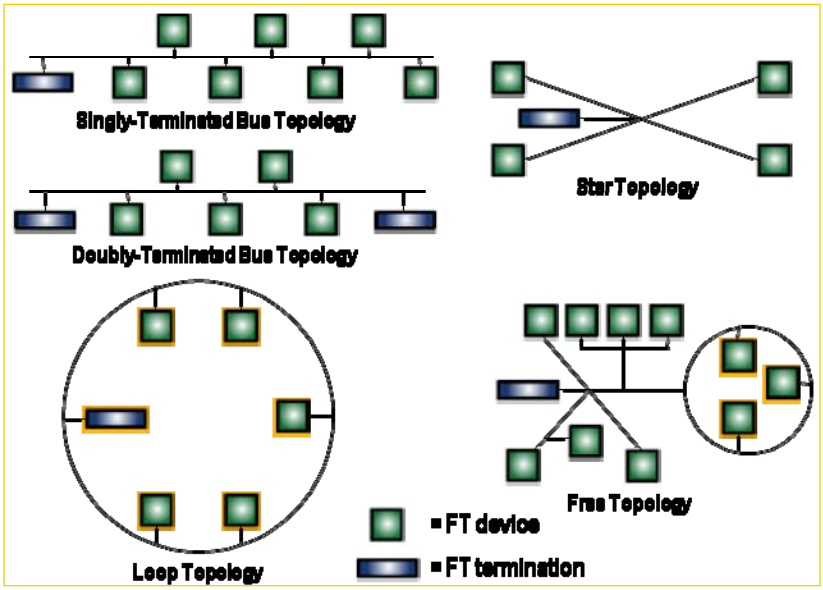


Figure 4 TP/FT-10 Supported Topologies

The wiring flexibility provided by the TP/FT-10 channel type has many advantages:

1. The installer is free to select the method of wiring and termination location that best suits the installation, reducing the need for advanced planning and allowing last minute changes at the installation site.

2. If installers have been trained to use one style of wiring for all installations, free topology technology can be introduced without requiring retraining.

3. Retrofit installations with existing wiring plants can be accommodated with minimal, if any, rewiring. This capability ensures that TP/FT-10 technology can be adapted to both old and new projects.

4. Free topology permits TP/FT-10 systems to be expanded in the future by simply tapping into the existing wiring where it is most convenient to do so. This reduces the time and expense of system expansion, and from the user's perspective, keeps down the life cycle cost of the free topology network.

Link Power System

A TP/FT-10 channel without link power requires separate data and power wiring. Installing separate data and power wiring implies that a technician's time will be spent troubleshooting the wiring harness to isolate and repair cable faults. Moreover, each time a sensor is added or an actuator is moved, both data and power wiring must be changed accordingly, often resulting in network down time until the new connections can be established.

The TP/FT-10 *link power system* provides the best solution for reducing installation and maintenance costs and simplifying system modifications by combining power and data on a common twisted wire pair. The link power system sends power and data on a common twisted wire pair, and allows the user to wire the control devices with virtually no topology restrictions. Support for link power can be added to any TP/FT-10 channel and can be designed into any TP/FT-10 device. A TP/FT-10 channel with link power can support a combination of locally powered and link powered devices. Link powered devices can be created using a Neuron Chip and the Echelon LPT-11 Link Power Twisted-Pair Transceiver. The LPT-11 transceiver integrates a link powered switching power supply that provides regulated +5VDC for the device to power the Neuron Chip, application electronics, sensors, actuators, and displays. Each transceiver can provide +5VDC at up to 100mA.

Power for a TP/FT-10 channel is supplied by a central 48VDC power supply, and flows through an LPI-10 Power Supply Interface onto the twisted pair wire (see Figure 5). The LPI-10 module isolates the power supply from wiring faults on the twisted pair, couples power to the system wiring, and terminates the twisted pair network.

There are two versions of the LPI-10 interface: a simple, low-cost, inductor-based design used with custom power supplies, and an electronic LPI-10 interface used with off-the-shelf 48VDC power supplies.

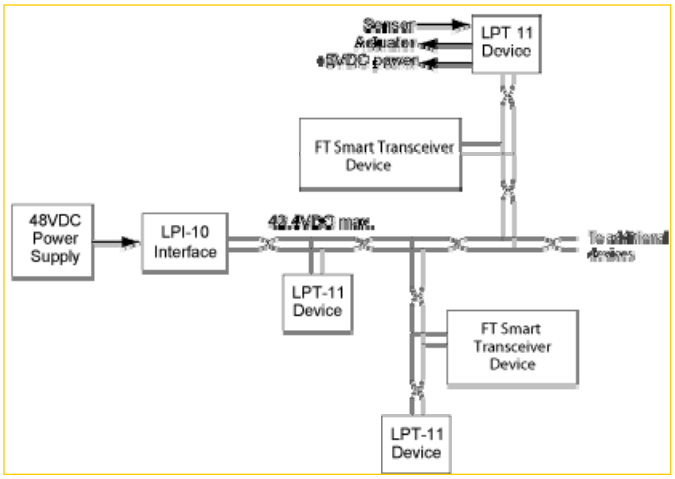


Figure 5 Example TP/FT-10 with Link Power System Wiring

Each link-powered device located along the twisted wire pair includes an integral switching power supply. This supply regulates the voltage on the twisted pair down to +5VDC at currents up to 100mA for use by the device. If a high current or high voltage device must be controlled, then the +5VDC power can be used to trigger an isolating high current triac, relay, or contactor.

The integral power supply does away with the need for a local AC-to-DC power supply, charging circuit, battery, and the related installation and labor expenses. The savings in money and time that results from eliminating the local power supply can be up to 20% of the total system cost; the larger the system, the greater the savings. Moreover, if standby batteries are used, then additional savings will be realized throughout the life of the system, because only one set of batteries will require service.

The link power system uses a single point of ground, at the LPI-10 module, and all of the link power transceivers electrically float relative to the local ground. Differential transmission minimizes the effects of common mode noise on signal transmission. If grounded sensors or actuators are used, then either the communication port or the I/O lines of the device must be electrically isolated.

TP/FT-10 Transceivers

Figure 6 FT 5000 Smart Transceiver

Several TP/FT-10 transceivers are available that device manufacturers can use to create TP/FT-10 devices. For locally-powered devices, the Echelon FT 5000 Smart Transceiver provides the most cost-effective solution because it is the lowest cost solution that combines a TP/FT-10 transceiver with a Neuron core that implements CNP and also executes the device application.

The Echelon LPT-11 transceiver can be used with Neuron Chips from Echelon, Cypress, or Toshiba to create link-powered TP/FT-10 devices. The LPT-11 includes an integral 100mA power supply for powering the link-powered device.

TP/FT-10 Cables

Echelon has qualified a variety of cables for use with TP/FT-10 channels. Based on the cost, performance, and availability of these different cable types, system designers can choose the most appropriate cable for their application. Echelon has qualified the following generic cable types:

- A generic 16AWG (1.3mm diameter) cable (similar to Belden 8471 or 85102)
- NEMA Level 4 cable (this cable is not equivalent to TIA Category IV cable)
- TIA Category 5 cable
- JY(St)Y for specific applications in the European market

The electrical specifications for these cables can be found in the *FT 5000 Smart Transceiver Data Book*. A list of cable vendors can be found in the *Junction Box and Wiring Guidelines for Twisted Pair LONWORKS Networks* engineering bulletin. These cables have been qualified by Echelon in a generic form, and are available from vendors in a number of variations, including shielded, unshielded, plenum, and non-plenum jacketing.

TP/FT-10 Specifications

Table 4 lists the transmission specifications for free topology TP/FT-10 channels. The *maximum total wire length* is the total length of wire with a TP/FT-10 segment. The specifications can be increased by using doubly-terminated bus topology as shown in Table 5.

- Up to 64 locally-powered TP/FT-10 devices or 128 link-powered devices are allowed per network segment. If both locally-powered and link-power devices are used on a TP/FT-10 channel, a maximum of 128 network loads can be attached to the channel, where a locally-powered device counts as two loads and a link-powered device counts as one load.
- For free topology TP/FT-10 channels, the distance from each transceiver to all other transceivers and to the termination (including the LPI-10 termination, if used) must not exceed the maximum device-to-device distance shown in Table 4. If multiple paths exist, such as a loop topology, then the longest path should be used for calculations.
- The average temperature of the wire must not exceed +55°C, although individual segments of wire may be as hot as +85°C.
- As a general rule, the TP/FT-10 channel communication cables should be separated from high voltage power cables. Local electrical codes must be followed with regard to cable placement.
- A doubly-terminated bus may have stubs of up to 3 meters from the bus to each device.
- The sum of the application current of all the link-powered devices in a segment must not exceed 3.2A at +5V.

Table 4 TP/FT-10 Free Topology Transmission Specifications

Cable	*Maximum Device-to-device Distance*	*Maximum Total Wire Length without Link Power (meters)*	*Maximum Total Wire Length with Link Power(meters)*
Belden 85102	500	500	500
Belden 8471	400	500	400
Level IV, 22AWG	400	500	400
JY(St) Y 2x2x0.8	250	450	320
TIA Category 5	250	450	400

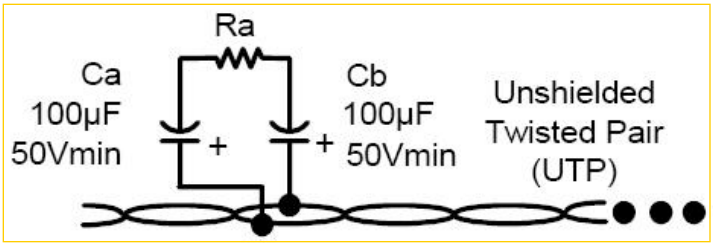
Table 5 TP/FT-10 Doubly-Terminated Bus Topology Transmission Specifications

Cable	*Maximum Bus Length without Link Power (meters)*	*Maximum Bus Length with Link Power (meters)*
Belden 85102	2700	2200
Belden 8471	2700	2200
Level IV, 22AWG	1400	1150
JY(St) Y 2x2x0.8	900	750
TIA Category 5	900	725

TP/FT-10 Termination

TP/FT-10 network segments require termination for proper data transmission performance. The type of terminator varies depending on whether shielded or unshielded cable is used. Free topology and bus topology networks also differ in their termination requirements.

In a free topology segment, only one termination is required and may be placed anywhere on the free topology segment. There are two choices for the termination: an RC network (Figure 7), with Ra = 52.3Ω ±1%, 1/8W, or an LPI-10 Link Power Interface, with the LPI-10 jumper at the "1 CPLR" setting.



Notes:

- Ca and Cb are typically aluminum-electrolytic type for improved longevity in the presense of ESD—observe polarity.
- Ca and Cb are required for connection to link power networks.

Figure 7 TP/FT-10 Termination RC Circuit

In a doubly terminated bus topology, two terminations are required, one at each end of the bus. There are two choices for each termination: an RC network (Figure 7), with Ra = 105Ω ±1%, 1/8W, or an LPI-10 Link Power Interface, with the LPI-10 jumper at "2 CPLR" setting. Only one LPI-10 interface is supported per segment. The LPI-10 contains one of the two required terminators. The other terminator must be an RC-type as shown in Figure 7.

When using shielded twisted pair, the twisted pair must be terminated according to the guidelines listed in the previous paragraphs. In addition, the cable shield must be grounded, as shown in Figure 8. The cable shield should be grounded using a capacitor to tie the shield to earth ground, and a large-value resistor to bleed off any static charge on the shield.

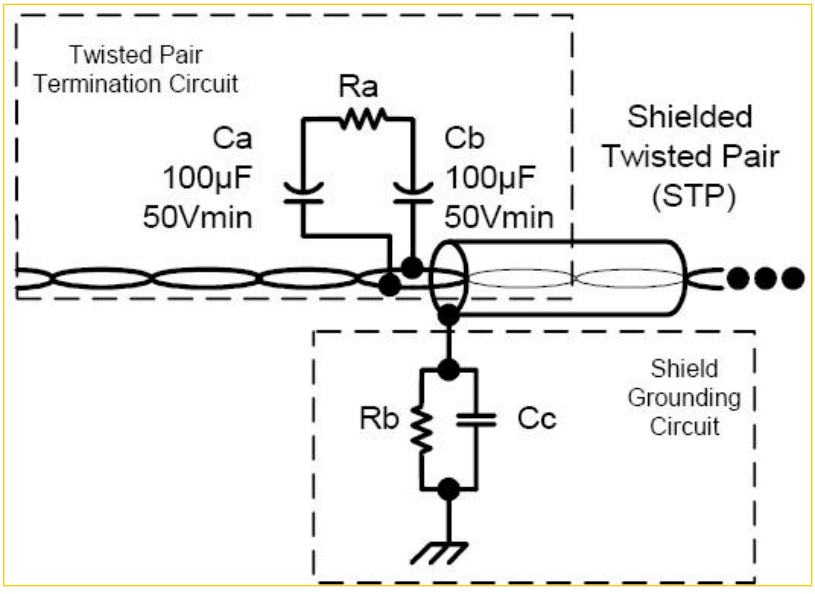


Figure 8 TP/FT-10 Termination and Grounding Circuit for Shielded Cable

The shield is tied to earth ground through a capacitor, instead of a direct connection, in order to avoid DC and 50/60 Hz ground paths from being formed through the shield. Typical values for Rb and Cc are as follows:

Cc = 0.1µF, 10%, Metalized Polyester, ≥ 100V

```
Rb = 470kΩ, 1/4W, ±5%
```

The cable shield should be grounded at least once per segment, and preferably at each device.

PL-20 Power Line

The easiest channel to install in a utility, home, or transportation system is one that communicates on the existing power line wiring in the power grid, home, or vehicle. Power line devices can simply be plugged into a power outlet with no further wiring required. The PL-20 channel type uses transceivers that conform to the ISO/IEC 14908-3 *Control Network Power Line (PL) Channel Specification*. They also comply with worldwide power line signaling regulations, including FCC, Industry Canada, Japan MPT, and European CENELEC EN50065-1 regulations. Worldwide compliance means that PL-20 transceivers can be used in applications worldwide. The European CENELEC communications protocol is automatically managed by PL-20 transceivers, eliminating the need for users to develop the complex timing and access algorithms mandated under CENELEC EN50065-1 (the PL-20C channel type). Additionally, PL-20 transceivers can operate in either the CENELEC utility (A-Band—the PL-20A channel type) or general signaling (C-Band—the PL-20C channel type with CENELEC access control or PL-20N without) bands, eliminating the need to stock multiple parts for different applications. These transceivers use narrow-band signaling over a 115kHz–132kHz frequency band on power mains media and support a 5.4kbps raw bit rate in C-Band, and a 75kHz–86kHz frequency band on power mains media and support a 3.6kbps raw bit rate in A-Band.

PL-20 transceivers offer the most robust, lowest cost means of communicating over AC or DC power lines using low, medium, or high voltage. Echelon offers a PL-20 transceiver that is a complete system-on-a-chip—featuring a highly reliable narrow-band power line transceiver, a Neuron core for running applications and managing network communications, a choice of on-board or external memory, and an extremely small form factor–all at a price that is compelling for even the most cost-sensitive consumer product applications.

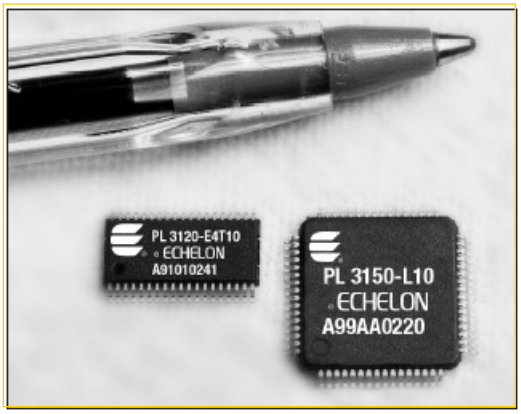


Figure 9 PL 3120 and 3150 Smart Transceivers

PL-20 transceivers offer unmatched communication reliability. They use a sophisticated digital signal processing core employing patented noise cancellation and distortion correction algorithms. These features enable the transceivers to correct for a wide variety of impediments to power line signaling, including impulsive

noise, continuous tone noise, and phase distortion. The result—Echelon's PL-20 transceivers can successfully receive signals that other solutions cannot even detect. A unique dual carrier frequency feature automatically selects an alternate secondary communication frequency should the primary frequency be blocked by noise. The transceivers use a highly efficient, low-overhead forward error correction (FEC) algorithm in addition to a cyclical redundancy check (CRC) to overcome packet errors. PL-20 transceivers have a dynamic range of > 80dB so that on a quiet line the transceivers can receive signals that have been reduced by a factor of 100,000.

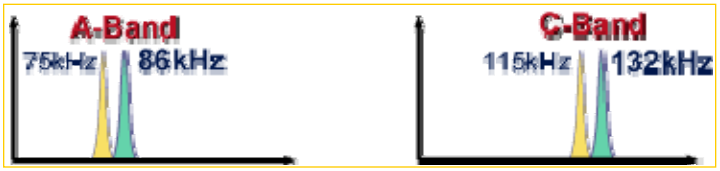


Figure 10 PL-20 Carrier Frequencies

The Echelon PL-20 transceivers are a field-tested, proven technology. The underlying core technology used in the Echelon PL-20 transceivers was developed and optimized through years of field-testing in applications worldwide. Millions of Echelon narrow band transceivers have been deployed in a wide range of consumer, utility, building, industrial, and transportation applications.

PL-20 Coupling Circuits

There are two methods of coupling a PL-20 transceiver to the power line: *line-to-neutral* coupling and *line-to-earth* coupling. These two methods, combined with two sets of carrier frequencies, define three multiple power line channel types referred to as PL-20C(L-N), PL-20N(L-N), PL-20C(L-E), PL-20N(L-E) and PL-20A(L-N). The PL-20C(L-N), PL-20N(L-N), and PL-20A(L-N) channel types use line-to-neutral coupling, and the PL-20C(L-E) and PL-20N(L-E) channel types use line-to-earth coupling. Most installations will consist of devices that use only one channel type in order to achieve maximum communication reliability. However, PL-20C(L-N) and PL-20C(L-E) devices may be mixed within an installation with the possibility of significant loss (2 – 20dB) of communication margin between L-N and L-E devices depending on physical location and power line environment, as can PL-20N(L-N) and PL-20N(L-E). PL-20C and PL-20N devices cannot be mixed within an installation.

The PL-20C(L-N), PL-20N(L-N), and PL-20A(L-N) channel types specify coupling circuits that transmit and receive the power-line communications signals between the line and neutral mains conductors. Line-to-neutral coupling is typically used except when not permitted by local regulations.

The PL-20C(L-E) and PL-20N(L-E) channel types specify coupling circuits that transmit and receive the power-line communications signals between the line and earth mains conductors.

PL-20 Specifications

Table 6 lists the transmission specifications for the PL-20 power line channels.

Table 6 PL-20 Power Line Specifications

Parameter	*PL-20(L-N)*	*PL-20(L-E)*	*PL-20A(L-N)*
Coupling Technique	Line-to-neutral	Line-to-earth	Line-to-neutral
Bit Rate	5kbps		3600bps
Modulation	BPSK		
Frequency Band	125kHz–140kHz		70kHz–95kHz
Output Level	≤ 116dBµV per EN50065-1 for Class 116 EN50065-1 compliance ≥115dBµV per EN50065-1 otherwise		
Output Impedance	|Z|≤6Ω 129kHz–134kHz	|Z|≤8Ω, 120VAC |Z|≤15Ω, 240VAC 129kHz–134kHz	|Z|≤1.1Ω 70kHz–95kHz
Input Impedance [1]	≥100Ω 125kHz–140kHz		≥500Ω 70kHz–95kHz

[1] Input impedance of transceiver and associated coupling circuit only. It does not include effect, if any, of system power supply.

Layer 2—Link Layer

The link layer defines media access methods and data encoding to ensure efficient use of a single communications channel. The raw bits of the physical layer are broken up into *data frames*. The link layer defines when a source device can transmit a data frame, and defines how destination devices receive the data frames and detect transmission errors. A priority mechanism is also defined to ensure delivery of important messages.

When a device has a message to send, the link layer protocol defines the timing of when the message is transmitted. Prior to transmitting a message, each device waits for the communications channel to become idle. Because multiple devices may be waiting simultaneously for the channel to become idle, each device waits for a random interval prior to transmitting. If another device starts transmitting during the waiting interval, the process is repeated. A unique CNP feature is that the number of randomizing slots increases as network load increases. This feature ensures reliable sustained network performance, even for networks that are heavily loaded.

Media Access

CNP uses a unique media access control (MAC) algorithm that enables an overloaded channel to operate close to its maximum capacity. Other MAC algorithms tend to degrade near maximum capacity due to excess collisions consuming the available bandwidth.

Many different MAC algorithms exist in networks today. The three most common are token ring, token bus, and carrier sense multiple access (CSMA). The MAC algorithm used by CNP belongs to the CSMA family.

Token Ring

The token ring MAC algorithm ensures orderly access to a communications channel by passing a token in an orderly fashion between each device on the channel. To ensure that only one device has the token at a time, the devices are linked in a daisy chain ring. This algorithm is not well suited to control applications for the following reasons:

- It is not suitable for open media such as power line, RF, or infrared because all stations receive the token simultaneously and cannot be linked in a ring.
- It increases the cost of each device in a control network because every device requires additional hardware to recover the token when it is lost and to ensure the continuity of the ring.
- In an existing network, it is disruptive to add new devices or replace existing ones. The ring must be broken to add or change a device, temporarily bringing down the entire network.
- Message latency increases as devices are added to a channel.

Token Bus

A token bus architecture solves the problem of sequential passing of a MAC token by including addressing information in the token. However, at low data rates, the process of circulating the token can result in considerable token latency. Because a device cannot transmit without first possessing the token, this latency adversely affects response time, which is a key parameter for control networks.

Additionally, token bus systems must reconfigure themselves each time a new device either becomes active or drops out. This overhead to reconfigure is a problem for all token bus networks. Because reconfiguration brings the network down for its duration, battery-powered devices whose normal operation is to wake up, send some messages, and power down, would cause the token bus system to suffer frequent reconfigurations. Battery-powered devices are required for applications needing RF or IR communication, for security applications, and for fire/life safety applications, to name a few.

Carrier Sense Multiple Access (CSMA)

The CSMA family of MAC algorithms is better suited for control networks than token ring or token bus. The CSMA algorithms require a device to establish that the medium is idle before it begins to transmit. However, each algorithm behaves differently once the idle state is detected. This results in very different network performance results under conditions of heavy data traffic.

Some CSMA algorithms use discrete intervals of time called *slots*, or *randomizing slots*, to access the medium. By limiting access to the medium by a given device to specific time slots, slotted media access greatly reduces the probability of two packets colliding. The most common form of slotted media access is *p-persistent* CSMA. In p-persistent CSMA, when a device has a message to send, it does so in a given randomizing slot with probability p. This algorithm is used by many popular CSMA protocols.

The link layer protocol for Ethernet as defined by IEEE 802.3 uses a degenerate form of p-persistent CSMA called *1-persistent CSMA*. This version of CSMA depends on collision detection which is not practical on many communications media used by control networks.

Control networks may consist of thousands of devices and multiple media on a single network. Because of the characteristics of different communications media and the potential need to cover large distances, LONWORKS networks must be able to support high-speed and low-speed channels, which may occasionally carry traffic approaching the channel's capacity. CSMA/CD (for example IEEE 802.3) behaves poorly during periods of overload, so it is generally not used for control applications. P-persistent CSMA works very well for small values of p (that is, many randomizing slots) at the expense of additional delay during relatively idle periods.

The CSMA family of MAC algorithms does not require a ring topology, synchronization or reconfiguration, and does permit new devices to be added or existing devices to drop out and rejoin the network transparently. Additionally, it supports many devices and is inexpensive to implement in hardware. The CNP MAC algorithm has the advantages of p-persistent CSMA without the disadvantages of additional delay during low traffic, significantly reduced throughput under conditions of high traffic, or the requirement for expensive collision detect hardware.

Predictive P-Persistent CSMA

CNP uses a variant of the p-persistent CSMA MAC algorithm called *predictive p-persistent* CSMA. CNP retains the benefits of CSMA but overcomes its shortcomings for control applications. Existing media access control algorithms such as IEEE 802.2, 802.3, 802.4, and 802.5 do not meet all the CNP requirements for multiple communication media, sustained performance during heavy loads, and support for large networks.

As in p-persistent CSMA, all LONWORKS devices randomize their access to the medium. This avoids the otherwise inevitable collision that results when two or more devices are waiting for the network to go idle so that they can send a packet. If they wait for the same duration after backoff and before retry, repeated collisions will result. Randomizing the access delay reduces collisions. In CNP, devices randomize different levels of delay called *beta 2 slots*, shown in **Figure 11**.

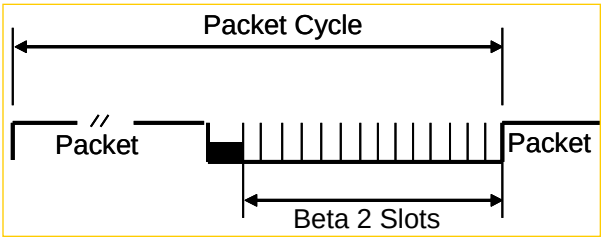


Figure 11 Beta 2 Slots

In p-persistent CSMA when a device has a message to send, it does so in a given randomizing slot with probability p. However, CNP carries the added improvement that p is dynamically adjusted based upon network load. When the network is idle, all devices randomize over 16 beta 2 slots. Thus the average delay in an idle network

is eight slot widths. When the estimated network load increases, devices may randomize over a larger number of beta 2 slots. The number of slots increases by a factor of *n*, where the range of *n* is from 1 to 63. Therefore, the maximum number of beta 2 slots is 63 * 16, or 1008. In CNP, *n* is called the estimated channel *backlog*. It represents the number of devices with a packet to send during the next packet cycle.

This method of estimating the backlog and dynamically adjusting the media access allows CNP to have a small number of beta 2 slots during periods of light load, while having the benefit of many beta 2 slots during periods of heavy load. Thus, media access delays are minimized during periods of light load, and collisions are minimized during periods of heavy load.

The ability to adjust the number of beta 2 slots depends on the ability to estimate the channel backlog. In CNP, a transmitting device includes information in the packet on the number of acknowledgements expected as a result of sending that packet. All the devices that receive the packet increment the estimated channel backlog by that amount. Likewise, the estimated channel backlog is decremented by 1 at the end of each packet. The estimated channel backlog is never decremented below 1. Because CNP packets are frequently acknowledged, even when the destination is a group of devices, 50% or more of the channel backlog is predictable at any time.

In summary, the CNP MAC algorithm specifically overcomes the shortcomings of existing MAC algorithms using a unique collision avoidance mechanism. To avoid collisions, all LONWORKS devices randomize their access to the communication medium using time slots. CNP dynamically adjusts the number of time slots by predicting the channel backlog. By actively managing the collision rate, CNP provides a superior MAC algorithm that is not dependent on collision detection hardware.

Priority

A key requirement of control networks is timely response to priority messages. CNP includes an optional priority mechanism to improve the response time of critical packets. When a device tries to access the communication medium, priority messages are given earlier access than non-priority messages. The protocol permits the network management tool to allocate a fixed number of beta 2 slots per channel as priority time slots dedicated to priority devices. The number of priority slots can be from 0 to 127. The network management tool that assigns priority slots to individual devices can ensure that one and only one device is assigned to a particular priority slot on the channel. Each priority time slot on a channel adds a minimum of two bit times to the transmission of every message. The amount of overhead will vary based upon the bit rate, oscillator accuracy, and transceiver requirements. For example, using a TP/XF-1250 1.25Mbps twisted pair transceiver with all devices on the channel having an oscillator accuracy of 0.02% (200ppm) or better, each priority slot is 30 bit-times wide. Because there is no contention for the media during the priority portion of a packet cycle, devices configured with priority have better response time than non-priority devices.

Figure 12 illustrates priority and non-priority beta slots. The value *m* is the number of priority slots. It is controlled by the network management tool used to install the devices on a channel. Its value may be zero to 127. The value *n* is the number of non-priority slots. It is calculated by each device on a channel and may be any value from 16 to 1008 - *m*.

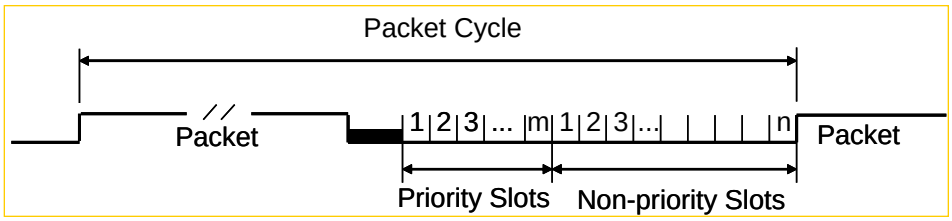


Figure 12 Beta 2 Slots With Priority

The priority slot assigned to a device applies to all priority packets sent from that device. One, all, or some of the packets sent from a device may be marked as using priority service. The priority designation within a device is made on a per network variable or per message tag basis, and may be set at compile time. In the case of network variables, the priority designation can optionally be changed during or after installation.

Lower priority numbers indicate higher levels of priority: a priority packet from a device assigned priority slot 2 will be transmitted before a priority packet from a device assigned priority slot 4. Setting a device's priority to 0 indicates that none of its packets will be transmitted in a priority slot, regardless of the message service assignment made at device compilation or installation time. Slot 1 is reserved for a network management tool, to ensure that no application can render a channel incapable of interruption by a network management tool. Slots 2 through 127 (depending on the medium, and the number of slots allocated on the channel) are then available for prioritized packets from designated devices.

When a priority packet is generated within a device, it travels out of the device on a priority queue, ahead of any pending non-priority packets buffered for transmission. Similarly, when a priority packet reaches a router, it goes to the head of the router queue (behind any other queued priority packets) and is forwarded to the far channel using the router's priority slot if one has been configured.

The effectiveness of priority depends on all devices preparing to send a message being synchronized on the end of the previous packet. When the sending devices are synchronized, then the priority time slots remain uncontested. Priority assignment has two limitations: (i) a maximum of 126 devices per channel may be assigned priority; and (ii) reserving time slots pre-allocates bandwidth. The first limitation is not a factor for channels that support fewer than 126 devices. For a TP/XF-78 twisted-pair transceiver using interoperable communications parameters, each priority slot is 5.6 bit times wide. Using a TP/XF-1250 twisted-pair transceiver with interoperable communications parameters the priority slots are 30 bit times wide. In effect, the length of each non-priority packet is increased by the time required for priority slots, thus using up channel bandwidth.

Frame Format

When a device has a message to send, it uses the MAC algorithm to decide when to send it. When the device sends the message, the bits within the message are encoded as a *frame*. The frame has the following components: the preamble, link layer header, network layer datagram, CRC, and end-of-packet indicator. A CNP frame has the following format (bit-sync is a variable number of bits, but must be at least 6 bits long):

msb							lsb
Bit-sync (configurable number of 1 bits)							0
Pri	Path	Delta Backlog					
Network Layer Datagram (6 to 246 bytes)							
CCITT CRC-16 (2 bytes)							

Figure 13 illustrates the physical encoding of a CNP frame on the communications channel.

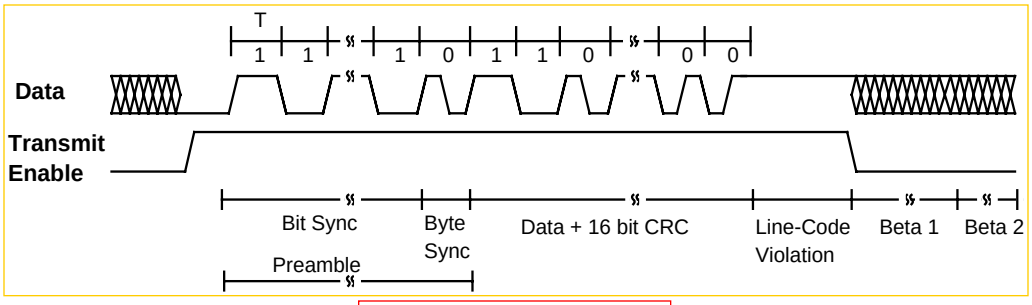


Figure 13 Packet Timing

For differential and single-ended modes, the transmitter transmits a preamble at the beginning of a packet to allow the other devices to synchronize their receiver clocks to the transmitter's clock. The use of a preamble allows every device on a channel to have an asynchronous clock. The preamble allows devices with independent clocks to synchronize their bit and byte sampling times.

The preamble consists of a *bit-sync* field and a *byte-sync* field. The bit-sync field is a series of Differential Manchester 1's; its duration is set to the same value for every device on a channel and is at least six bits long. The byte-sync field is a single bit Differential Manchester 0 that marks the end of the preamble, and immediately precedes the beginning of the first byte of the packet.

The frame is terminated with an idle period on the channel. An idle period is indicated by a lack of transitions on the channel for longer than 2 bit times. For differential and single-ended modes, the transmitter indicates the end-of-packet by forcing a Differential Manchester line-code violation; that is, it holds the data output transitionless long enough for the receiver to recognize an invalid code that signals the end of transmission. The data output can be either high or low for the duration of the line-code violation, depending on the state of the data output after transmitting the last bit. The line-code violation begins after the end of the last CRC bit and lasts for at least 2 bit times. The last bit does not have a trailing clock edge, so the data output actually remains transitionless for at least 2.3 bit times.

For special purpose mode, the preamble and end-of-packet indicator are managed by the transceiver. The preamble and end-of-packet indicator may be similar to those used by differential and single-ended modes, but is not required to be the same.

The **Pri** field specifies the priority of the frame. A value of 0 is used for non-priority frames, and a value of 1 is used for priority frames.

The **Path** field specifies the channel to use for special purpose mode transceivers. A special purpose mode transceiver can be interfaced to two physical channels or to two physical bit encodings on the same channel. This allows error recovery by switching

to an alternate channel if message transmission on the primary channel fails. A value of 0 is used for the primary channel and a value of 1 is used for the alternate channel. The transport layer specifies the alternate channel for the last two retries of an acknowledged message.

The **Delta Backlog** field specifies a channel backlog increment. Receiving devices add this value to their backlog estimate to account for transmission of this frame.

The **CRC** field is a 16-bit CRC computed using the CCITT CRC-16 standard polynomial, which is $X^{16} + X^{12} + X^5 + 1$. The polynomial is computed over the entire packet and link layer header.

Layer-2 Performance

CNP Predictive p-Persistent CSMA Performance

The CNP predictive algorithm works best when a majority of the CNP packets on a channel are acknowledged. The number of acknowledgements that a given packet generates is encoded into each acknowledged packet. Each device on the channel receives all the acknowledged packets on the channel and adds the number of acknowledgements to the channel backlog. If none of the packets is acknowledged, the predictive part of the algorithm does not dynamically expand the number of randomizing slots with an increase in load. Using exclusively unacknowledged services causes CNP to behave like a non-predictive p-persistent CSMA where $p = 0.0625$. However, even this degenerate case is still significantly better than IEEE 802.3 (which is a p-persistent CSMA protocol where $p = 1$) under conditions of heavy network traffic.

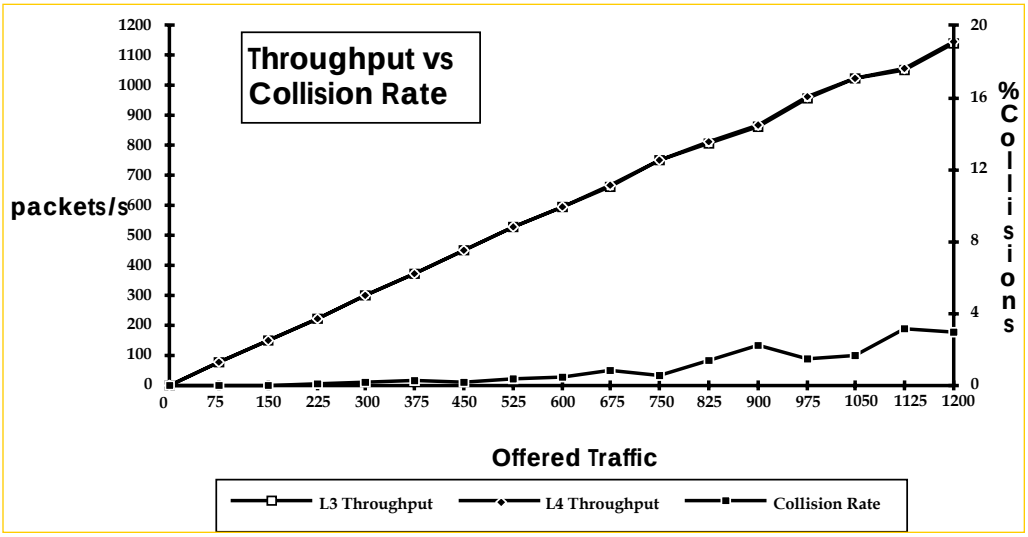


Figure 14 Effective Network Throughput vs. Offered Traffic Using Unacknowledged Services

The results of two benchmarks used to illustrate this point are shown in Figure 14 and Figure 15. The graphs shown in these figures illustrate results achieved from a test bed of 34 devices. Thirty-two of the devices acted as traffic generators. The other two acted as test devices, repeatedly sending messages to each other at

different traffic levels. Both graphs show the amount of actual traffic that got through on the network versus the offered traffic, or the number of packets attempted for transmission.

Figure 14 shows this data using exclusively *unacknowledged* packets. In this mode, the network behavior is similar to a 0.0625-persistent CSMA algorithm. The graph shows how the network throughput is nearly proportional with offered traffic and up to 1200 packets/sec.

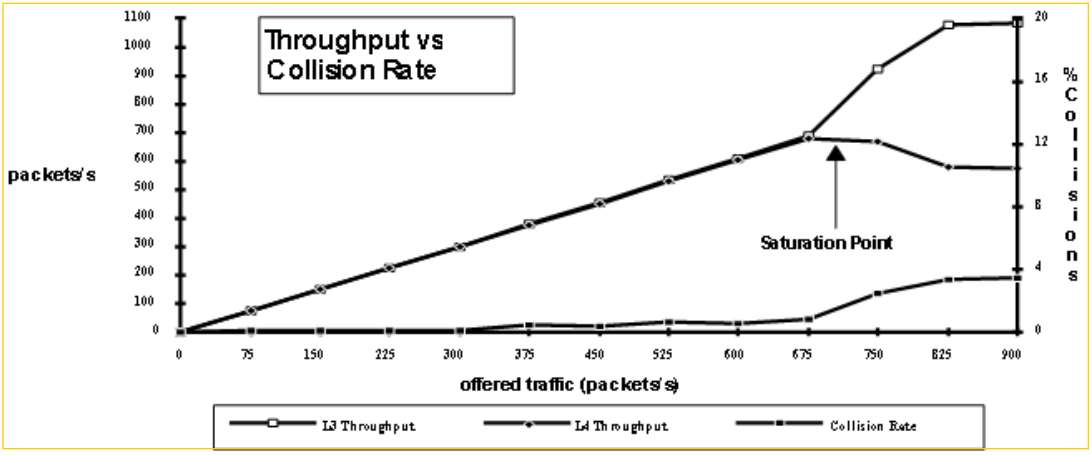


Figure 15 Effective Network Throughput vs. Offered Traffic Using Acknowledged Services

To demonstrate the effectiveness of the predictive algorithm, the experiment was re-run with only one change. In this case, messages used *acknowledged* services. Figure 15 shows the data derived from this experiment. The graph shows that network throughput degrades slightly with an increase in offered traffic past the saturation point.

Figure 14 and Figure 15 illustrate how the collision rate grows slowly, even past the channel saturation point. This active management of the collision rate is what makes the CNP predictive p-persistent CSMA algorithm superior to other CSMA algorithms.

Media Access Delay and Offered Traffic

When a device on a network tries to send a message, it must first wait for the medium to be idle. The time delay between when a device queues the packet to send and the time that it is actually sent on the network plays a role in the response time of that message. This delay is known as the *media access delay*. As the offered traffic (the total number of packets per second offered for transmission by all the devices on a channel) on a given channel of a network increases, the media access delay increases. In networks loaded to near-capacity, when many devices are trying to send messages, this delay can be significant. In designing a network to meet specified response times, the worst-case offered traffic must be considered and designed for.

Following is an example of assessing the impact of media access delay: For an example 78kbps channel type, the preamble is 23.5 bit times, the average packet size is 112 bits (14 bytes), and the average number of bit times between each packet is 48. The amount of time a typical message spends on the media is 183.5 (= 23.5 + 112 + 48) bit times on the channel. The maximum rate at which packets can be sent on

this channel is therefore 425 packets (=78125/183.5) per second, and each packet cycle requires 2.3ms.

For this example, if a device on this network requires a maximum response time requirement of 50 ms, we must set a bound on the offered traffic to limit the media access delays. At 78 kbps with 183.5 bit packet cycles it takes this network 50 ms to send 25 packets. If the packets on the channel may be sent at any time, including all at once, then, in order to always meet the requirement of 50 ms without using any special priority features, the channel should never have an offered traffic greater than 21 packets per second. As this example illustrates, the response time requirements of devices on a network must be used to design the network and ensure that the offered traffic does not cause unacceptable media access delays

Layer 2 Channel Throughput

Layer 2 channel throughput depends on the bit rate, oscillator frequencies and accuracy, transceiver characteristics, average packet size, and the use of acknowledgments, priority, and authentication on the channel. An average packet is in the range of 10 to 16 bytes long, depending on the length of the domain identifier, the addressing mode, and size of the data field. The maximum packet size is 249 bytes including data, addressing, and protocol overhead.

At low bit rates or with long packets, channel throughput is bounded by the packet transmission time and average media access delay. At higher bit rates with short packets, the packet processing power of the Neuron core limits channel performance.

Table 7 and Table 8 estimate the approximate network throughput as a function of bit rate and packet size. The “peak” traffic numbers can be supported for short bursts.

Table 7 CNP Channel Throughput With 12-byte Packets

Bit Rate (kbps)	*Peak Number of Packets/Sec*	*Sustained Number of Packets/Sec*
4.883	25	20
9.766	45	35
19.531	110	85
39.063	225	180
78.125	400	320

Table 8 CNP Channel Throughput With 64-byte Packets

Bit Rate (kpbs)	*Peak Number of Packets/Sec*	*Sustained Number of Packets/Sec*
4.883	7	5
9.766	13	10
19.531	25	20
39.063	50	40
78.125	100	80

Layer 3—Network Layer

The network layer defines how message packets are routed from a source device to one or more destination devices. This layer defines naming and addressing of devices to ensure the correct delivery of packets. Messages can be addressed to a single device, to any group of devices, or to all devices. Group addressing reduces network traffic by supporting the delivery of a single message packet to multiple devices.

This layer also defines how messages are routed between the source and destination devices when these devices are on different communication channels. Addresses are formed using a hierarchical structure which supports the use of routers that filter messages based on their destination address. By supporting routers at the network layer, CNP supports the installation of very large systems with thousands of devices. Routers use the network layer to confine traffic to segments within a large network, thereby increasing total capacity of the network.

Naming and Addressing

A *name* is an identifier that uniquely identifies a single object within an object class. A name is assigned when an object is created and does not change over its lifetime. To ensure that every LONWORKS device can be uniquely distinguished from every other LONWORKS device, every Neuron core includes a unique 48-bit ID called the *Neuron ID*. This ID will always be unique for every Neuron core and does not change over the lifetime of the Neuron core.

An address is an identifier that uniquely identifies an object or group of objects within an object class. Unlike a name, an address may be assigned and changed any time after an object is created.

CNP addresses uniquely identify the source device and destination device (or devices) of a CNP packet. These addresses are also used by intelligent routers to selectively pass packets between two channels.

A Neuron ID may be used as an address. However, the Neuron ID is not used as the sole form of addressing in CNP because such addressing only supports one-to-one transactions (that is, no groups), complicates device replacement, and would require excessively large routing tables and time-consuming processing to optimize network traffic. This addressing mode is used primarily during installation and

configuration, because it allows communications with a device before the device has been assigned an address.

To simplify routing, CNP defines a hierarchical form of addressing using three address components: the *domain*, *subnet*, and *node* addresses. This form of addressing can be used to address an individual device or collections of devices sharing common subnet or domain address components. To further facilitate the addressing of multiple dispersed devices, CNP defines another class of addresses using domain and *group* addresses.

The use of a dynamically assigned address instead of a fixed name simplifies replacement of devices in a functioning network. The replacement device is assigned the same address as the device it replaces. Thus all references to this device from elsewhere on the network do not need to be modified, as would be the case if Neuron ID addressing were to be used.

The various address forms are described in the following sections, along with discussions on routers and address generation.

The Domain Address Component

A *domain* is a logical collection of devices on one or more channels. Communications can only take place among devices configured in a common domain; therefore, a domain forms a virtual network. Multiple domains can occupy the same channels, so domains may be used to prevent interference between devices in different networks.

For example, two adjacent buildings using devices with power line transceivers on the same frequency may be on the same channel. To prevent interference between the applications carried out by the devices, the devices in each building would be configured to belong to different domains.

The Neuron core may be configured so that it belongs to one or two domains. A device that is a member of two domains may be used as a gateway between the two domains. CNP does not support communications between domains, but an application program may be implemented to forward packets between two domains.

A domain is identified by a domain ID. The domain ID may be configured as 0, 1, 3, or 6 bytes. Six byte domain IDs can be used to ensure that the domain ID is unique: for example, using the 48-bit Neuron ID of one of the Neuron cores in the domain as the domain ID ensures that no other network can have the same domain ID, because all Neuron IDs are unique. However, six byte domain IDs add six bytes of overhead to every packet. The overhead may be reduced by using a shorter domain ID. In a system where there is no possibility of interference between multiple networks, the domain ID may be configured as 1 byte (zero byte domain IDs are supported by CNP, but reserved for administrative purposes). Domain IDs may be configured as 1 or 3 bytes in systems where a single administrator controls assignment of domain IDs to prevent duplicate IDs.

The domain ID can also be used for application-level purposes. For example, a domain ID could be used by service personnel as a system identifier.

The Subnet Address Component

A *subnet* is a logical collection of up to 127 devices within a domain. Up to 255 subnets can be defined within a single domain. All devices in a subnet must be on the same segment (subnets cannot cross intelligent routers). Figure 16 illustrates six devices installed in two subnets. The six devices are physically attached to the same communications channel.

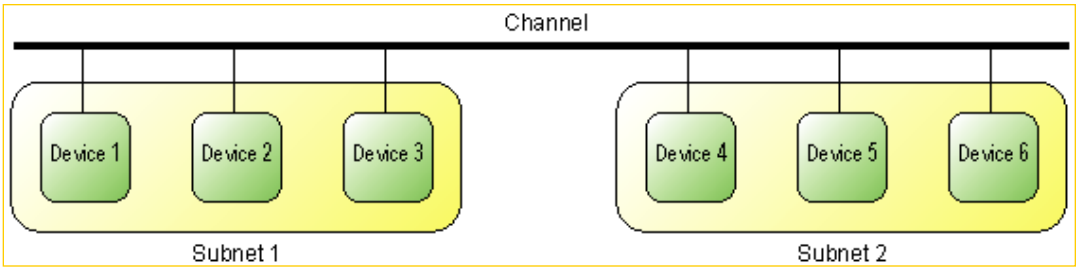


Figure 16 Multiple Subnets on a Single Channel

Figure 17 illustrates seven devices installed in two subnets. The seven devices are physically attached to two communications channels. The two channels are on a common segment because they are connected by a repeater.

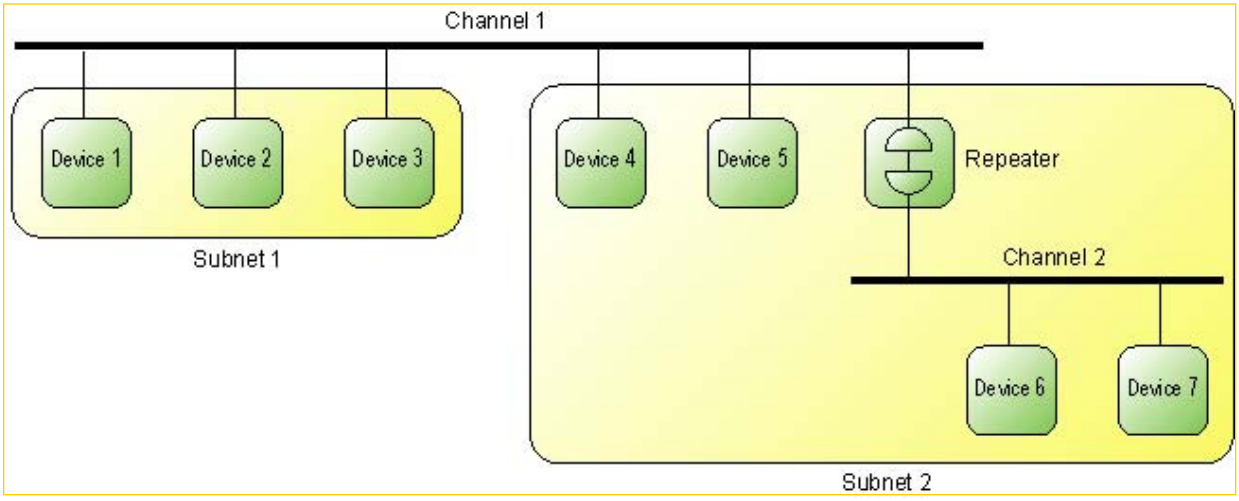


Figure 17 Two Subnets on a Common Segment

Figure 18 illustrates a network with a router. An additional subnet is required because subnets cannot cross intelligent routers.

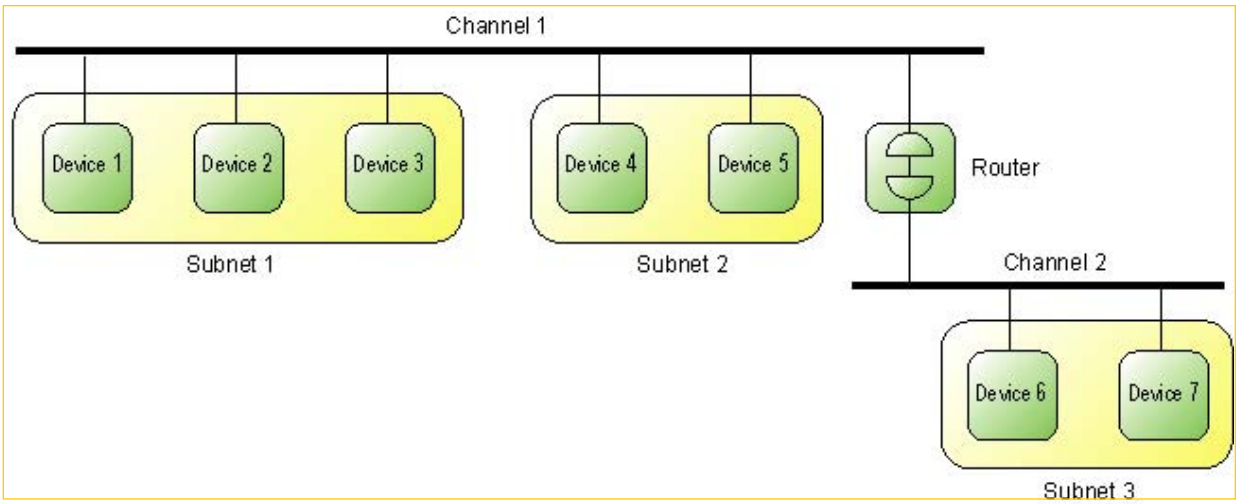


Figure 18 Three Subnets on Two Segments

If a device is configured to belong to two domains, it must be assigned to a subnet within each of the domains. Device 4 in Figure 19 is a member of two domains, and is assigned to subnet 1 in both domains. The subnet numbers do not have to match in the two domains.

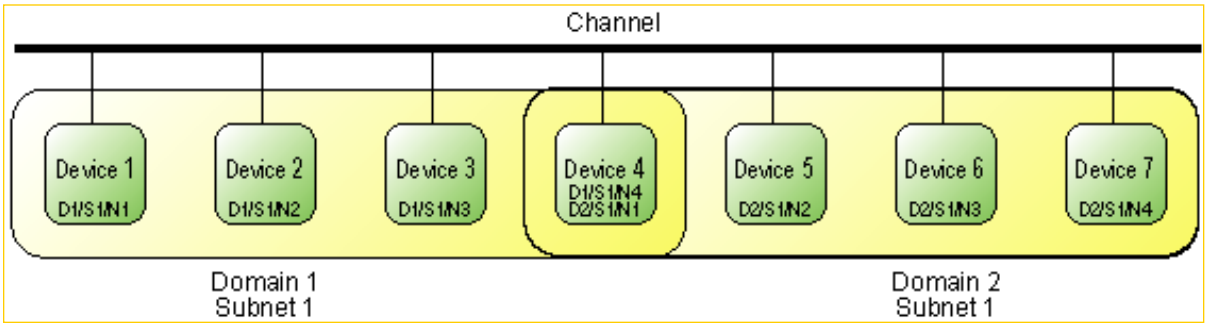


Figure 19 Device Configured in Two Domains

All devices within a domain are typically configured in the same subnet except in the following cases:

- They are located on different segments with intervening intelligent routers. Because subnets cannot cross intelligent routers, the devices must be on different subnets.
- Configuring the devices in the same subnet would exceed the maximum number of devices allowed in a subnet. Subnets are limited to 127 devices. Multiple subnets may be configured on a segment to increase the capacity of the segment above 127 devices (up to the limit of the channel used by the subnet). For example, a segment with two subnets may have up to 254 devices; three subnets may have up to 381 devices; and so on.

The Node Address Component

Every device within a subnet is assigned a unique node ID within that subnet. The node ID is 7 bits, so there may be up to 127 devices per subnet. A maximum of 32,385 devices (255 subnets x 127 devices per subnet) may be in a single domain.

Each configured device can be uniquely identified using its combination of domain ID, subnet ID, and node ID.

Groups

A *group* is a logical collection of devices within a domain. Unlike a subnet, however, devices are grouped together without regard for their physical location in the domain. The Neuron firmware implementation of CNP allows a device to be configured to be a member of up to 15 groups, other implementations have different limits.

Groups are an efficient way to use network bandwidth for messages addressed to multiple devices (that is, one-to-many connections). Figure 20 illustrates a network with two groups. Group 1 consists of devices 1, 2, 3, 10, 11, and 12. Group 2 consists of devices 3, 4, 5, and 7. Device 3 is in both groups.

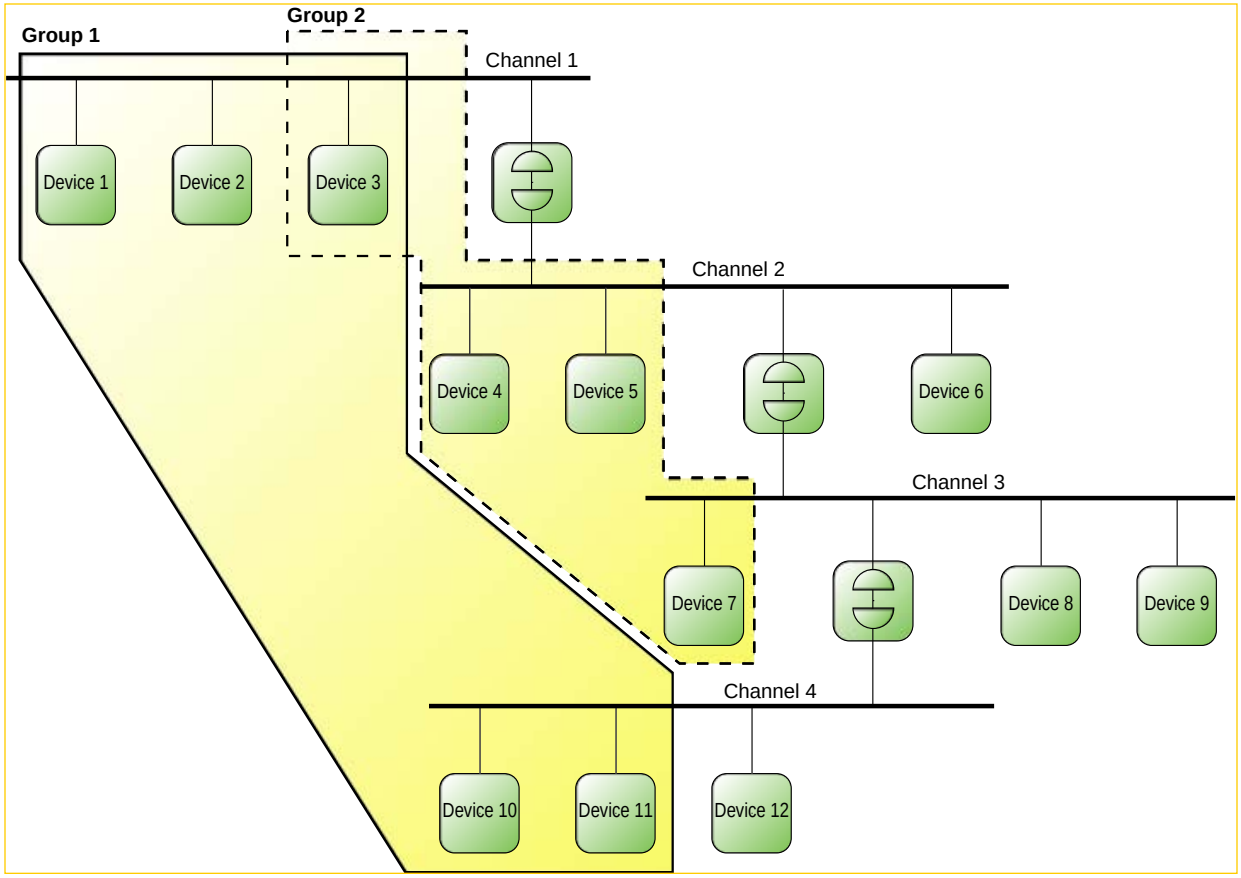


Figure 20 Groups Spanning Channels and Routers

Groups are identified by a one-byte group number, so a single domain may contain up to 256 groups. Layer 6 adds another addressing layer on top of the layer-3 addresses for data points. As a result, it is possible to reuse group IDs when group IDs are used in combination with layer-6 addresses.

Neuron ID

In addition to the subnet/node ID address, a device may always be addressed by a unique ID call the *Neuron ID*. The Neuron ID is 48-bits long, and is assigned when

each CNP device is manufactured. This ID is unique worldwide for every CNP device.

The Neuron ID is used by a network tool in the initial configuration of devices at installation time to assign each device in a network to one or two domains, and to assign subnets and node ID numbers. Neuron ID addressing is not used for application messages except in self-installed systems by a user-interface controller.

Addressing Formats

CNP devices are addressed using one of five addressing formats. The particular addressing format used determines the number of bytes required for the source and destination address. Table 9 defines the formats and number of bytes required for each. The total address size is computed by adding the appropriate number of bytes indicated in the table to the size of the domain ID, which can range from 0 to 6 bytes depending on the configured size of the domain ID.

Table 9 CNP Address Formats

Address Mode	*Address Format*	*Destination*	*Address Size (bytes)*
Domain-wide Broadcast	Domain (Subnet = 0)	All devices in the domain	3
Subnet-wide Broadcast	Domain, Subnet	All devices in the subnet	3
Multicast	Domain, Group	All devices in the group	3
Unicast	Domain, Subnet, Node	Specific device within a subnet	4
Neuron ID	Domain, Neuron ID	Specific device	9

Address Table

Each CNP device contains an *address table* that contains addressing information that can be managed by a network management tool in a managed network, or by the CNP implementation on the device in a self-installed network. A CNP application can specify an address for an outgoing message by referencing an entry in the address table. This type of address is called an *implicit address*. Alternatively, a CNP application can specify the full layer-3 address for an outgoing message. This type of address is called an *explicit address*.

The advantage of implicit addressing is that it eliminates the requirement for a CNP application to manage destination addresses. Layer-3 addresses are typically allocated and managed by the network management tool in a managed network or the ISI engine in a self-installed network. The CNP application can send a message using a fixed index into the address table rather than attempting to determine the full layer-3 address for each message.

The advantage of explicit addressing is that it allows a CNP application to send messages to more destination addresses than can be stored in the address table. An address table may contain up to 65,535 entries, but different CNP implementations may have lower limits. The Neuron firmware is limited to a maximum of 15 entries in the address table.

The address table also specifies the groups that the CNP device belongs to. When an incoming group-addressed message is received, the CNP implementation searches the address table for a matching group ID. As a result, the number of entries in the address table for a CNP device is the upper limit on the number of groups that the device may be a member of. The limit is reduced by the number of non-group implicit addresses stored in the address table.

Routers

Routers are infrastructure devices that connect two channels and route packets between them. Routers can be installed to use one of four routing algorithms:

- **Repeater**. A repeater is the simplest form of router, simply forwarding all valid packets between the two channels. Using a repeater, a subnet can exist across multiple channel segments.
- **Bridge**. A bridge forwards all packets which match its domains between the two channels. Using a bridge, a subnet can exist across multiple channel segments.
- **Learning Router**. A learning router monitors the network traffic and learns the network topology at the domain/subnet level. The learning router then uses its knowledge to selectively route packets between channels. Learning routers cannot learn group topology, so all packets using group addressing are forwarded.
- **Configured Router**. Like a learning router, a configured router selectively routes packets between channels by consulting internal routing tables. Unlike a learning router, the contents of the internal routing tables are defined by a network management tool. A network management tool can optimize network traffic by defining routing tables for both subnet and group address routing. Configured routers do not execute the learning algorithm as a learning router does. Instead, the network management tool pre-configures the router's forwarding tables at the time of installation, based on the tool's knowledge of the network topology.

Configured routers and learning routers are a class of routers known as *intelligent* routers.

Choosing Between Learning and Configured Routers

Initially, each learning router sets its internal routing tables to indicate that all subnets could lie on either side of the router. Figure 21 illustrates a network with two learning routers. If device 5 generates a message addressed to device 2, channel 2 will carry the message to learning routers 1 and 2. Examining the source subnet field of the message, learning router 1 notes in its internal routing tables that subnet 2 lies below it. The router then compares the source and destination subnet IDs. Because they are different, the message is passed on.

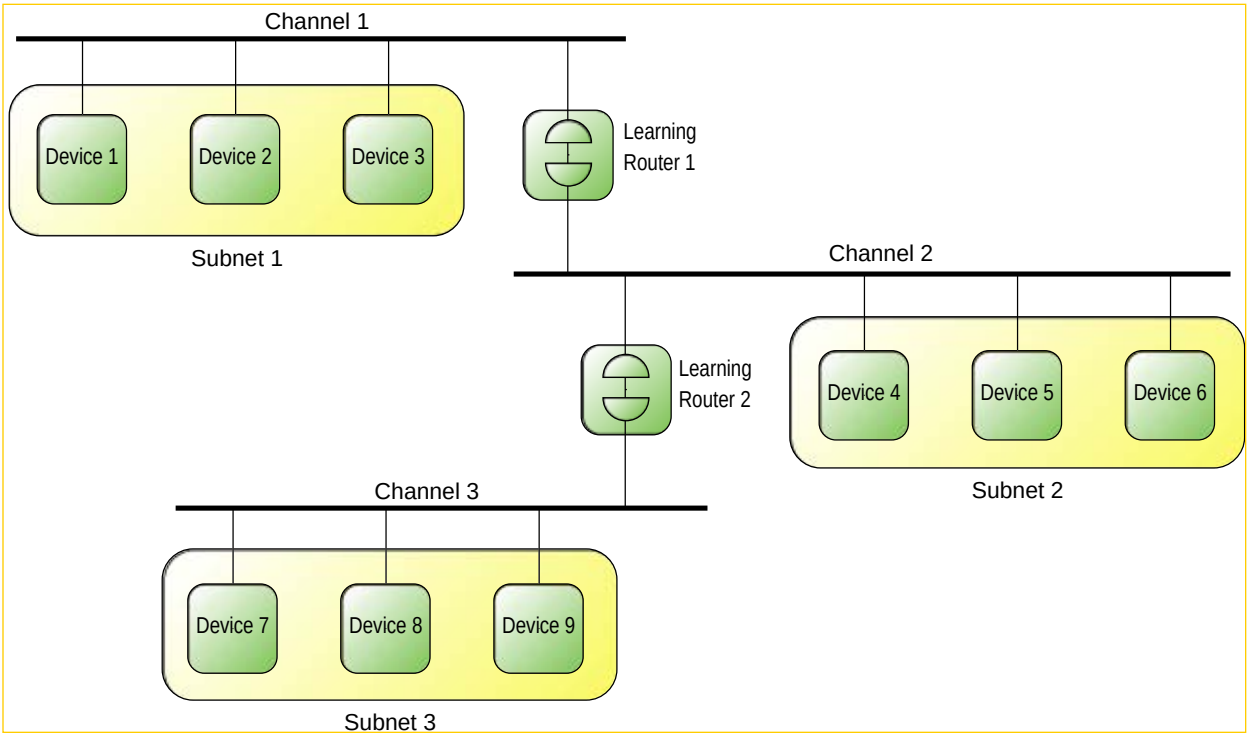


Figure 21 Learning Routers

Meanwhile, learning router 2 has also passed on the message, making an appropriate notation in its internal routing tables regarding the location of subnet 2.

If device 2 generates an acknowledgement, the acknowledgement is picked up by learning router 1, which now notes the location of subnet 1. Learning router 1 examines its internal routing tables, and, upon discovering that subnet 2 lies below, passes the message on. When the message appears on subnet 2, it is noted by both device 5 (the destination device), and learning router 2, who does not pass it on but merely notes that subnet 1, like subnet 2, lies somewhere above. Learning router 2 will not learn of the existence or location of subnet 3 until a message is originated from there.

Configured routers should always be used when possible for the following reasons:

- The initial flood of traffic that occurs while a learning router is learning the network topology may cause congestion problems.
- The network topology may have inadvertent "loops," which are common in power line and RF networks, that can cause a learning router to develop an inaccurate network image.
- Learning routers do not learn about groups but configured routers can be configured to selectively forward group addressed packets.

Subnets cannot cross intelligent routers. While bridges and repeaters allow subnets to span multiple channels, the two sides of an intelligent router must belong to separate subnets. The fact that intelligent routers are selective about the packets they forward to each channel can be used to increase the total capacity of a system in terms of devices and connections. In general, it is always a good idea to segment traffic among "communities of interest" if possible.

Physical Layer Repeaters

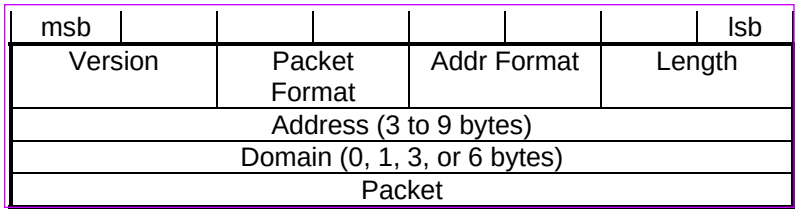
Some channel types support physical layer repeaters (PLRs). A physical layer repeater receives and forwards packets between two or more ports and can be used to overcome a channel's device count or cable length limitations in some cases.

While a physical layer repeater does not forward pure noise (it is not just a bi-directional amplifier), PLRs contain no intelligence, and will therefore forward any packet, even one that might have been corrupted as result of a collision.

A router should be preferred to a physical layer repeater, but PLRs offer a low-cost alternative to routers in some cases.

Datagram Format

The network layer builds a *datagram* by adding a network header to a transport layer *packet*. A CNP datagram has the following format:



The **Version** field defines the CNP version number, and is always 0.

The **Packet** Format field defines the format of the packet enclosed within the datagram, and contains one of the following values:

Packet Format Field	Packet Format
0	Transport Packet
1	Session Packet
2	Authenticated Packet
3	Presentation Packet

The packet formats are defined in chapters 5, 6, and 7.

The **Addr Format** field defines the format of the address contained within the datagram and contains one of the following values:

Addr Format Field	Address Format
0	Subnet Broadcast
1	Group
2	Subnet/Node or Group Acknowledgement
3	Neuron ID

The **Address** field contains the network address for the message. The format of the address is defined by the **Addr Format** field. A Subnet Broadcast address has the following format:

msb							lsb
Source Subnet							
1	Source Node						
Destination Subnet							

A Group address has the following format:

msb							lsb
Source Subnet							
1	Source Node						
Destination Group							

A Subnet/Node address has the following format:

msb							lsb
Source Subnet							
1	Source Node						
Destination Subnet							
1	Destination Node						

A Group Acknowledgement address has the following format:

msb							lsb
Source Subnet							
0	Acknowledged Member						
Destination Subnet							
1	Destination Node						
Acknowledged Group							

A Neuron ID address has the following format:

msb							lsb
Source Subnet							
1	Source Node						
Destination Subnet							
Neuron ID (6 bytes)							

The **Length** field defines the length of the domain field, and contains one of the following values:

Length Field	*Domain Length (bytes)*
0	0
1	1
2	3
3	6

Layer 4—Transport Layer

The transport layer ensures reliable delivery of message packets. Messages can be exchanged using an acknowledged service, where the sending device waits for an acknowledgement from the receiver or receivers and resends the message if the acknowledgement is not received. The application is informed if an acknowledgement is not received after a configurable number of retries. The

transport layer also defines how duplicate messages are detected and rejected if a message is resent due to a lost acknowledgement. Messages that do not require the reliability of acknowledged service can use unacknowledged or repeated services to send the message once, or a configurable number of times, without waiting for an acknowledgement.

Message Services

CNP offers four basic types of message service: acknowledged, request/response, repeated, and unacknowledged. The acknowledged, repeated, and unacknowledged services are managed by the transport layer. The request/response service is managed by the session layer. These services provide a tradeoff between reliability and efficiency.

- The most reliable service is *acknowledged*, or end-to-end acknowledged service, where a message is sent to a device or group of devices and individual acknowledgements are expected from each receiver. If an acknowledgement is not received from all destinations, the sender times out and retries the transaction. The number of retries and the time-out are both selectable (see *CNP Timers*, later in this document). Transaction IDs are used to keep track of messages and acknowledgements so that the application does not receive duplicate messages.
- An equally reliable service is *repeated*, also called the *unacknowledged repeated service*, where a message is sent to a device or group of devices multiple times, and no response is expected. This service is typically used when broadcasting to more than a few devices, because the number of packets required to repeat the transmission will be less than the number of packets required for all the acknowledgements. For example, if a device sends an acknowledged message with four retries to five devices, there will be a minimum of six packets—the original message plus five acknowledgements. If the same message was sent with the repeated service with a repeat count of four, only four packets would be generated, yet the probability of delivery would be identical. A good rule-of-thumb is to use repeated service instead of acknowledged service whenever the size of the group is larger than the retry count. The exception to this rule is if the sending application needs notification of delivery failure, in which case acknowledged service should be used. This is typically only required if the sending application can take alternative action in case of delivery failure.
- The least reliable is *unacknowledged*, where a message is sent once to a device or group of devices and no response is expected. This is typically used when the receiving application is not sensitive to the loss of a message. This is typically the case with data that is sent using a periodic heartbeat. For example, if a temperature sensor used for space comfort control in a room reports the temperature once a minute, missing an update is not serious, because the temperature will be updated on the next heartbeat.

With one exception, the three transport layer message services—unacknowledged, acknowledged, and repeated—can be used with any layer-3 address mode: broadcast, unicast, multicast, or Neuron ID. The exception is that broadcast acknowledged transactions complete once a single acknowledgment is received—any remaining acknowledgements are ignored.

CNP Timers

CNP implementations use the following timers to manage the various layer 4 services:

- *Transaction timer*
- *Repeat timer*
- *Group receive timer*
- *Non-group receive timer*
- *Free-buffer wait timer*

These timers are collectively known as the *layer-4 timers*, and are typically automatically configured by a network management tool. For example, the LonMaker tool automatically calculate and configure these timers.

The following sections detail, for each of the four message services, how packets flow through a CNP implementation and where the timers come into play.

Unacknowledged Service

When this service is used, the only timer that is involved is the free buffer wait timer. This timer determines the maximum length of time the device will wait for a free buffer when sending a message. This timer can be deactivated (the device will wait forever) by setting the timer value to zero. If it is set to another number, *n*, then the device will wait between 2i and 2i + 1 seconds. For example, if the configured number *n* is set to 2, then the device will wait for a free buffer for between 4 and 5 seconds. If a buffer is not obtained before the timer expires, the device assumes a fatal error and resets.

Acknowledged Service

Acknowledged service also uses the free buffer wait timer, but additional timers are necessary. The sending device uses a *transaction timer* to determine when a retry should be attempted. The receiving devices use a *receive transaction timer* to detect duplicate messages.

Sending Acknowledged Messages

The *transaction timer* is used by a sending device to determine how long the device waits for an acknowledgement before retrying. The value of the transaction timer used by the device is taken from the transmitting device's address table entry for the destination address of the packet being sent. The transaction timer is individually configurable by destination address in the address table. If the device does not receive an acknowledgement before the transaction timer expires, it will retry, sending the same packet again (along with an indication of which devices did acknowledge, in the case of group addressing). This retry process will continue until the *retry count* has been exhausted or until all acknowledgements have been received. The retry count is configurable from 0 to 15 by address table entry.

The appropriate length for the transaction timer is dependent on the number of routers and types of channels that a message must pass through between its source

and furthest destination (in terms of transit time including channel and router delays). The transaction timer should be just long enough so that a packet can reach the furthest destination and the acknowledgement from this destination can be received before the transaction timer expires. If the transaction timer is too short, excess retries will be generated; if too long, the time for a transaction to complete will increase on average.

By adjusting the transaction timer, the time delay before messages are retried (due to lack of acknowledgement) can be optimized, therefore improving response time. To keep the number of retries to a minimum, this timer must be set high enough to accommodate the round trip delay of sending a message and getting an acknowledgement back. On a network running at 78 kbps or 1.25 Mbps, where the source and destination(s) are all on the same channel, the transaction timer can be set to a low value such as 64ms or 96ms. In the case where the packet has multiple destinations, the path to the furthest destination must be used to calculate the timer value. A device will not try to initiate retransmission until its transaction timer expires.

The LonMaker tool automatically calculates the transaction time defaults based upon topology, bit rate, and the Neuron system clock frequency for Neuron hosted devices.

Receiving Acknowledged Messages

The *receive transaction timer* is used by destination devices to detect duplicate messages. Each device maintains a *receive transaction table* that stores the source address and a *transaction ID* of all messages received within the interval defined by the receive transaction timer. When a packet arrives at its final destinations, the receiving devices look at the packet's source address and transaction ID. If there are no matching entries in the receive transaction table, a new receive transaction entry is created. If the receive transaction table is full and can accept no more entries, the incoming message is lost. If the lost message was an acknowledged message, no acknowledgement is returned. A *receive timer* is started for each entry in the table. The duration of the receive timer is based upon the layer-3 address mode that the transmitter used. If the transmitter used group addressing and an address table entry for that group exists, the *group receive timer* value is taken from that entry in the address table. If any other addressing mode is used, the device uses its *non-group receive timer* value instead.

When the receive timer expires, the entry in the receive transaction table is deleted, and any new transmission having the same transaction ID from the same source address will be treated as a new transaction. Therefore, this timer must be greater than the greatest product of retry count and transaction timer that can be received from the transmitter. A good rule of thumb for setting this timer is:

```
receiveTimer = (retryCount + 2) * transactionTimer
```

If the receive transaction timer is too long, then it is likely that the device will run out of memory for receive transaction entries. If it is too short, then the device may mistake legitimate retries for new transactions, causing duplicate messages to mistakenly be passed on to the application for processing. A good rule of thumb is to keep the retry count low (typically 4) and design networks with as few router hops from end-to-end as possible; this will keep the transaction timer short. As an example, for a received message that originates from a device with a retry count of 4 and a transaction timer of 200ms, use the rule-of-thumb above to arrive at ((4 + 2) *

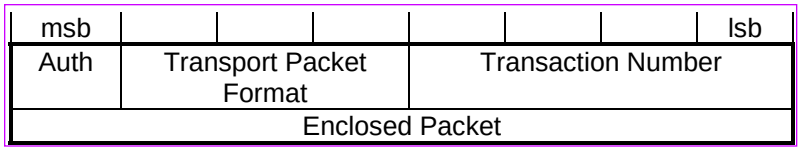
200), or 1200ms. A shorter value could result in retries being interpreted as new transactions; a longer value could result in a device's running out of receive transaction buffers and losing incoming messages.

Repeated Service

This service follows essentially the same message flow as acknowledged service, with a few exceptions. In the address table of the transmitter there is a separate timer known as the *repeat timer*. This timer specifies how frequently the message is repeated when using repeated service. This time can be shorter than the transaction timer, because no acknowledgement is expected (no time for the acknowledgement need be allotted) when these messages are sent. Transaction IDs and duplicate detection are in effect for these transactions. The initial packet is repeated as many times as is specified by the repeat count, from 1 to 15 times. This timer should be long enough for the receiving device or devices to overcome any short term buffer shortages.

Transport Packet Format

The transport layer builds a transport packet by adding a transport header to an enclosed packet. The enclosed packet may be null, in which case the transport packet consists solely of the transport header. A transport packet has the following format:



The **Auth** field is set to 1 for an authenticated message, and is set to 0 for messages that are not authenticated.

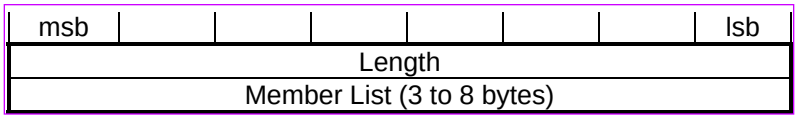
The **Transport Packet Format** field defines the format of the packet enclosed within the transport packet, and contains one of the following values:

Transport Packet Format Field	***Transport Packet Format***
0	Acknowledged Message
1	Repeated Message
2	Acknowledgement
4	Reminder Preamble
5	Reminder Message

Acknowledged (transport packet format 0) and *repeated* (transport packet format 1) enclosed packets consist of presentation packets as described in *Layer 6—Presentation Layer*. The acknowledged transport packet is used for the first transmission of an acknowledged message (the reminder preamble and reminder message are used for subsequent transmissions). The repeated transaction packet is used for all the transmissions of a repeated message.

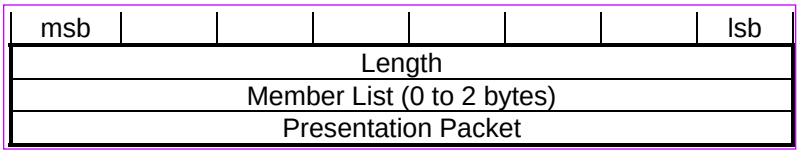
An *acknowledgement* (transport packet format 2) has no enclosed packet and consists solely of the transaction header.

A *reminder preamble* (transport packet format 4) enclosed packet flags the first message of a message pair that is used for selective soliciting of acknowledgments for multicast transactions. This message pair is used by a device that has transmitted a multicast message, and the highest numbered group member that has not responded is member number 16 or higher. The second message of the pair is a retransmission of the original acknowledged message. The reminder preamble has the following format:



The **Length** field specifies the number of bytes in the member list, and will be a value between 3 and 8. The **Member List** field is a bit map of the members in the group. Each bit represents a member of the group where the LSB of the first byte represents member 0, the next bit is member 1, etc. A value of 0 indicates that the member's acknowledgment has not been received by the sender, a value of 1 indicates that the acknowledgment has been received.

A *reminder message* (transport packet format 5) enclosed packet combines a reminder preamble with the original presentation packet and has the following format:



The **Length** and **Member List** fields have the same format as for the reminder preamble. A **Length** value of 0 means all members should acknowledge.

Layer 5—Session Layer

The session layer adds control to the data exchanged by the lower layers. It supports remote actions so that a client may make a request to a remote server and receive a

response to this request. For application messages, the request is passed to the receiving application which generates the response.

The session layer also defines an authentication protocol that enables receivers of a message to determine if the sender is authorized to send the message. This can be used to prevent unauthorized access to devices and their applications.

Request/Response

The *request/response* service is used when a message is sent to a device or group of devices and individual responses are required from each receiver. The incoming message is processed by the application on the receiving device before a response is generated. The same retry and time-out options are available as with acknowledged service. Responses may include data, making this service particularly suitable for implementing remote procedure calls or client/server applications.

With one exception, the request/response service can be used with any layer-3 address mode: broadcast, unicast, multicast, or Neuron ID. The one exception is that when performing a broadcast request/response the application will receive only the first response; all others will be discarded by the network layer.

The message flow for request/response service is identical to acknowledged service, except that the application sends a response in lieu of an acknowledgement. Network management tools must take into account the extra processing time required by the application to generate the acknowledgement when calculating the transaction and receive transaction timers.

Authentication

When using authenticated messages, the receivers of an authenticated message determine if the sender is authorized to send that message. This can prevent unauthorized access to devices and their applications. This can be used to prevent unauthorized access to devices and their applications. For example, by using authentication, an electronic lock device can verify that an "open" request comes from the owner, not from someone attempting to break into the system

Authentication is implemented by distributing 48-bit keys, one per domain, to the devices at or prior to installation time. For an authenticated message to be accepted by the receiver, both sender and receiver must possess the same key. This key is distinct from the device's Neuron ID.

As shown in Figure 22, when an authenticated message is sent, the receiver challenges the sender to authenticate itself, using a different random number as a challenge every time. The sender then authenticates by transforming the challenge, using the authentication key along with the data in the original message. The receiver compares the reply to the challenge with its own transformation on the challenge. If the transformations match, the transaction goes forward. This is called an *authenticated* transaction. The transformation used is designed so that it is extremely difficult to deduce the key, even if the challenge, reply, and authentication algorithm are all known. The use of authentication is configurable individually for each network variable connection. In addition, network management transactions may be optionally authenticated.

Figure 22 Authentication Sequence

The authentication protocol is always enabled on every CNP device, and every domain on every device always has an authentication key. The default authentication key is typically FF FF FF FF FF FF. For example, the LNS Network Operating System uses this value for devices that do not use authentication, but does not allow this value to be used as the authentication key for a device that uses authentication.

Authentication Control

For a given message, it is up to the sender of the message to initiate an authenticated transaction when required. The sender does this by setting the authentication bit in the message. When a receiver receives a message with the authentication bit set, it must respond with an authentication challenge, even if it does not require the message to be authenticated. It is up to the receiver to determine whether or not the message must be authenticated. This means that a sender may initiate an authenticated transaction on any message, whether required or not. However, a sender should not initiate an authenticated transaction unless it is required by the receiver, since authenticated transactions consume double the bandwidth of non-authenticated transactions. In a group connection, there may be a mixture of receivers that require authentication and receivers that do not require authentication. In this case, any update sent to the group must be sent as an authenticated transaction, even though all receivers do not require it.

A receiver may choose to honor a request initiated by a failed authentication transaction, and will typically do so for any messages that do not require authentication. For example, if a receiver does not require network management messages to be authenticated but receives a Read Memory request with the authentication bit set in the message, the receiver sends an authentication challenge, but ignores the authentication response and sends the results of the Read Memory request as if the authentication bit was not set.

Session Packet Format

The session layer builds a session packet by adding a session header to an enclosed packet. The enclosed packet may be null, in which case the session packet consists solely of the session header. A session packet has the following format:

msb							lsb
Auth	Session Packet Format			Transaction Number			
Enclosed Packet							

The **Auth** field is set to 1 for an authenticated message, and is set to 0 for messages that are not authenticated.

The **Session Packet Format** field defines the format of the packet enclosed within the session packet, and contains one of the following values:

Session Packet Format Field	*Session Packet Format*
0	Request
2	Response
4	Reminder Preamble
5	Reminder Message

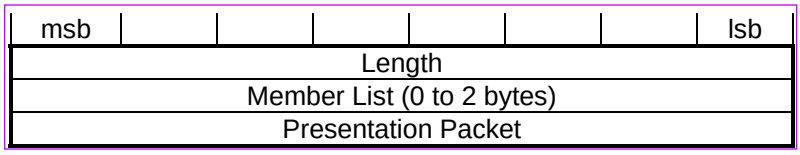
Request (session packet format 0) and *response* (session packet format 2) enclosed packets consist of presentation packets as described in *Layer 4—Transport Layer*. The request session packet is used for the first transmission of a request message (the reminder preamble and reminder message are used for subsequent transmissions). The response session packet is used for response messages.

A *reminder preamble* (session packet format 4) session packet flags the first message of a message pair that is used for selective soliciting of responses for multicast requests. This message pair is used by a device that has transmitted a multicast request, and the highest numbered group member that has not responded is member number 16 or higher. The second message of the pair is a retransmission of the original request. The reminder preamble has the following format:

msb							lsb
Length							
Member List (3 to 8 bytes)							

The **Length** field specifies the number of bytes in the member list, and will be a value between 3 and 8. The **Member List** field is a bit map of the members in the group. Each bit represents a member of the group where the LSB of the first byte represents member 0, the next bit is member 1, etc. A value of 0 indicates that the member's response has not been received by the sender—a value of 1 indicates that the response has been received.

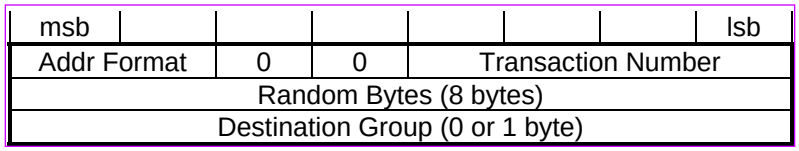
A *reminder message* (session packet format 5) session packet combines a reminder preamble with the original presentation packet and has the following format:



The **Length** and **Member** List fields have the same format as for the reminder preamble. A **Length** value of 0 means all members should respond.

Authenticated Packet Format

Two special packets are generated for authenticated transport packets and session packets. The first is the Challenge Packet which is sent by a receiver whenever a transport packet or session packet is received with the authentication bit set. A Challenge Packet has the following format:



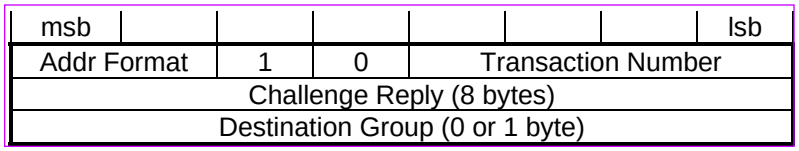
The **Addr Format** field is the same as the **Addr Format** field in the network header, described in *Addressing Formats*.

The **Transaction Number** field contains the same transaction number as the challenged Transport Packet or Session Packet.

The **Random Bytes** field contains a challenge consisting of 8 random bytes.

The **Destination Group** field is present only if the **Addr Format** field specifies a group address (address format 1). This field contains the same value as the **Destination Group** field in the network header.

When the transmitting node receives the challenge, it responds with a Reply Packet. The Reply Packet has the following format:



The **Addr Format**, **Transaction Number**, and **Destination Group** fields are the same as for the Challenge Packet.

The **Challenge Reply** field contains an 8-byte value that is computed using the original presentation packet, the transmitter's authentication key, and the 8 random bytes in the challenge.

Layer 6—Presentation Layer

The presentation layer adds structure to the data exchanged by the lower layers by defining the encoding of message data. Presentation layer services are provided by the Neuron firmware for applications hosted on a Neuron core; these services are provided by the host or split between a host and a LONWORKS network interface for applications running on other host processors.

Messages

Data is exchanged between applications at layer 6 encoded as *messages*. Each message consists of a 1-byte message code followed by 0 to 227 bytes of data, with the exception of network variables that consist of 1 to 31 bytes of data. The message code identifies the type of data contained within the message. Table 10 lists the message types supported by CNP, and the message codes used for each type.

Table 10 Message Codes

Message Type	*Message Code (hex)*	*Message Code (decimal)*
User Application Message	00 – 2F	0 – 47
Standard Application Message	30 – 3E	48 – 62
Foreign Frame Message	40 – 4E	64 – 78
Network Diagnostic Message	50 – 5F	80 – 95
Network Management Message	60 – 7F	96 – 127
Network Variable Message	80 – FF	128 – 255

A message may be encoded as a *network variable*, *application message*, or *foreign frame*. Applications typically exchange data using network variables. Network variables are a class of message packets with an identifier that identifies the data as a data value that may be shared by multiple devices on a network. Interoperable encoding of network variables is provided with *standard network variable types* (*SNVTs*). A SNVT specifies a standard data encoding that determines how applications will interpret the data contained in a network variable. SNVTs promote interoperability by ensuring that applications use a common interpretation of data exchanged through network variables. For example, two devices exchanging a temperature network variable can use a SNVT to ensure that both devices interpret the temperature data using the same units, instead of one device using Celsius and another using Fahrenheit.

Application messages are used by application programs requiring a different data interpretation model than network variables. Application messages are message packets with a 6-bit message code that identifies the packet to the receiving application. The applications exchanging application messages must agree on the interpretation of the message codes. Standard message codes are used for standard services as listed in Table 11.

Table 11 Standard Application Message Codes

Message Type	*Message Code (hex)*	*Message Code (decimal)*
Data Log Access Request Message	3C	60
Interoperable Self-Installation Message	3D	61
File Transfer Message	3E	62

Foreign frames are exchanged as a simple array of bytes that can be interpreted by the application in any way (for example, as a frame of a foreign protocol).

The message types are described in more detail in the following sections.

Message codes are also used for responses to request/response messages, and are encoded as shown in Table 12.

Table 12 Message Codes for Responses

Message Type	*Message Code (hex)*	*Message Code (decimal)*
Application Response	00 – 3E	0 – 62
Responder Offline Response	3F	63
Foreign Frame Message	40 – 4E	64 – 78
Foreign Responder Offline Response	4F	79
Network Diagnostic Success	30 – 3F	48 – 63
Network Diagnostic Failure	10 –1F	16 – 31
Network Management Success	21 – 3F	33 – 63
Network Management Failure	00 – 1F	0 – 31
Network Variable Poll Response	80 – FF	128 – 255

Network Variables

Network variables are CNP data values that may be shared among multiple devices. Network variables may represent a single value or a structure or union of multiple values containing 1 to 31 bytes. A device may have multiple network variables, and each network variable may be shared with one or more network variables on any device or group of devices within a network.

Up to 31 bytes may be embedded in a network variable structure and propagated as a single network variable. If more than 31 bytes of data are needed in a single message, application messages can be used as described in the next section. Network

variables may also be organized as arrays of network variables, where each element of the array may be individually connected to network variables on other devices.

Every network variable has a *direction*, *type*, and *length*. The network variable direction can be either input or output, depending on whether the network variable is used to receive or send data. The network variable type determines the encoding and units of the data.

Network variables of identical type and length but opposite directions can be *connected* to allow the devices to share information. Connections can be established between output and input network variables on different devices, or between output and input network variables on the same device. The latter type of connection is called a *turnaround connection*. Connections may be thought of as "virtual wires," replacing the physical wires of traditional hard-wired systems.

An example of two devices with a connection between them is an application on a lighting device that has a switch-type input network variable, and a second application on a dimmer-switch device that has a switch-type output network variable. A network tool such as the LonMaker tool can be used to connect these two devices, allowing the switch to control the lighting device, as shown in Figure 23. This type of connection from a single output to a single input is called a *unicast connection*.

Figure 23 Unicast Network Variable Connection

The direction indicated by the triangle in Figure 23 indicates the direction of the network variable. A single network variable may be connected to multiple network variables of the same type but opposite direction. The example in Figure 24 shows the same switch being used to control three lights:

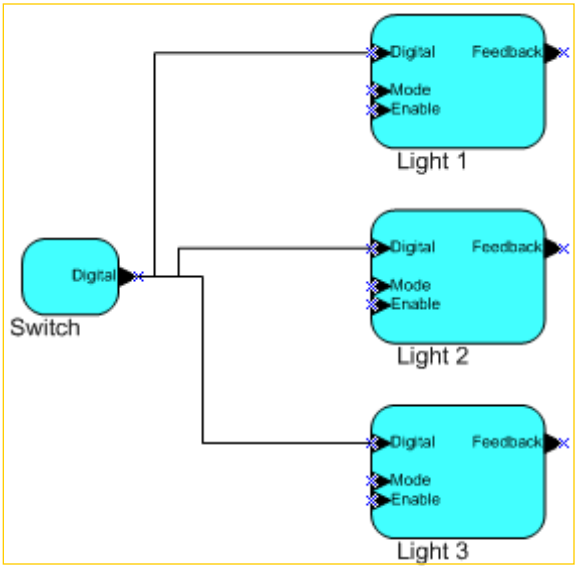


Figure 24 Multicast Network Variable Connection

The application program in a device does not need to know anything about where its input network variable values come from or where its output network variable values go. When the application program has a changed value for an output network variable, it simply passes the new value to the CNP implementation. Rather than expose the internal operation and configuration of every device to every other device on a network, network variables allow devices to only expose data that is required to interoperate with other devices. For example, the dimmer-switch device in Figure 24 could be replaced with an occupancy sensor, without making any changes to the lighting devices.

A connection is created through a process called *binding*. Binding typically takes place during network design and installation. Binding may be done by a network management tool such as the LonMaker too, or may be done by the device application itself through a process called *self-installation*. When binding network variables, the CNP implementation is configured to know the logical address of the other device or group of devices in the network expecting that network variable's values. It assembles and sends the appropriate packets to these devices. Similarly, when the CNP implementation receives an updated value for an input network variable required by its application program, it passes the data to the application program. The binding process thus creates the logical connections between an output network variable in one device and an input network variable in another device or group of devices.

Network variables provide a data oriented application protocol. Application data items such as temperatures, pressures, states, text strings, and other data items are exchanged between devices in standard engineering and other predefined units. Commands are encapsulated within the application programs of the receiver devices rather than being sent over the network. In this way, the same engineering value can be sent to multiple devices which each have a different application program, and potentially different commands, for that data item.

The network variable concept greatly simplifies the programming of complex distributed applications. Network variables provide a very flexible view of distributed data to be operated on by the devices in a system. The programmer need

not deal with message buffers, network addressing, request/response/retry processing, and other low-level details that must typically be managed for other protocols.

Network Variable Selection

Connections consist of addressing information at CNP layers 3 and 6. The layer-3 address delivers a network variable update to the correct set of devices; the layer-6 address selects the correct network variable on those devices.

To support network variable selection at layer 6, each network variable on a device has a unique *network variable index* for that device. Network variable indexes are assigned sequentially for each network variable on the device, starting with index 0. For example, a device with a switch output and a switch feedback input may use index 0 for the output and index 1 for the input.

The layer-6 address for a network variable is called the *network variable selector*. A network variable selector is an identifier that is used to associate a network variable update message with a network variable within the receiving application. Every network variable update message includes a network variable selector, and every CNP device maintains one or two tables that are used to associate a network variable selector with a network variable index. The first table implemented by all CNP devices is the *network variable configuration table*. The second table that may optionally be implemented by a CNP device is the *network variable alias table*.

When sending a network variable update, the layer-6 implementation uses these tables to translate the network variable index on the sending device to a network variable selector. When receiving a network variable update, the layer-6 implementation uses these tables to translate the network variable selector to a network variable index on the receiving device.

Because network variable updates are first filtered by the network-layer address, network variable selectors may be reused within a domain. Network variable selectors must be unique for all network variable update and poll request messages received by a device, but may be reused within two connections that have no common devices.

The network variable selector is a 14-bit identifier with a value between 0 and 3FFF hex. Selector values 3000 to 3FFF hex are reserved for unbound network variables, with the selector value equal to 3FFF hex minus the network variable index. This convention allows unbound network variables to be polled, as long as the polling device has the network variable index of the network variables to be polled. Selector values 0 to 2FFF hex are available for bound network variables. This provides a total of 12,288 different network variable selectors for bound network variables, some of which may be reused as described in the previous paragraph.

Network Variable Aliases

Each network variable on a device may have multiple layer-6 addresses, that is, each network variable may have multiple selectors. Each selector for a network variable creates a virtual network variable. The primary selector defines the *primary network variable*. The additional selectors, if any, create *alias network variables*, simply called *aliases*. A primary network variable and all of its aliases share a

common network variable value. When the CNP implementation sends a network variable update, it sends the updated value to the primary network variable and all of its aliases. When the CNP implementation receives a network variable update, it compares the selector contained in the update to the selectors defined for all primary network variables and all aliases, until a match is found or until all primary network variables and aliases are searched, whichever comes first.

Network Variable Configuration and Alias Tables

The network variable configuration and alias tables map network variable selectors to network variable indexes and vice versa. Each entry in the two tables contains a network variable selector, an address table index, layer-4 message service information, a priority flag, and an authentication flag for the network variable. The network variable configuration table has an entry for each network variable implemented by the device. The network variable configuration table entries are indexed by the network variable index. The network variable alias table has an entry for each alias, and associates each alias with a primary network variable if the alias is active. The alias for a network variable shares the network variable value with the associated primary network variable.

Network Variable Limits

CNP supports up to 4096 primary network variables and 8192 network variable aliases per device, but the maximum number may be further limited by specific CNP implementations. The typical limit for a Neuron hosted device is 254 primary network variables and 127 aliases. The maximum number of network variables and aliases per device may also be limited by the network management tool used to bind the device; different tools will have different capacities depending on their memory and database size.

The number of network variables that can be implemented on a device does not limit the number of network variables that can be accessed by the device. A single input network variable can receive data of the same type from an unlimited number of devices—potentially from every device within the domain. The receiving application can differentiate network variables from different device by examining the source address of the network variable. For example, an alarm monitoring system may have 1000 alarm output network variables bound to a single alarm input network variable on a monitoring device. This configuration only requires a single input on the monitoring device, yet the application on the monitoring device can still determine the source of any alarm by reading the source address field of the message.

A single output network variable can be used to send data of the same type to an unlimited number of devices—potentially to every device within the domain.

Network Variable Types

Network variables provide a mechanism for exchanging data values between devices, but the network variable packets do not specify the encoding of the data. For example, two devices exchanging network variable updates representing current temperature could have different encoding for the data. If the sending device uses Celsius values with a fixed point representation providing 0.1 degree resolution, and the receiving device expects Fahrenheit integer values, the two devices would be incompatible. One solution to this problem is to add a third device in between the

two incompatible devices to perform the temperature conversion. This approach works, but unnecessarily complicates system integration.

Network variable types are used to ensure compatible interpretation of data. A standard set of network variable types defines standard units, ranges, and resolution for most common units of measure, and also defines many standard structures for common aggregate data. These standard types are called *standard network variable types*, or *SNVTs* (pronounced *snivets*). The list of SNVTs includes over 100 types and covers a wide range of applications. The complete list is available at types.lonmark.org. New SNVT definitions are added when new types are required for applications created by multiple vendors. If an application requires a network variable type that is not a SNVT, device-manufacturers can define custom network variable types. These are called *user network variable types* (*UNVTs* pronounced *you-nivits*).

A network variable type can either be a *scalar type* or an *aggregate type*. Table 13 describes the scalar data types, and Table 14 describes the aggregate data types.

Table 13 Scalar Data Types

Data Type	*Description*
Bitfield	A signed or unsigned bitfield, 1-8 bits wide. Only available for fields within a structure or union.
Double Float	An ANSI/IEEE 754 standard 64-bit double-precision floating point value with 1 sign bit, 11 exponent bits, and 52 mantissa bits, for a total of 64 bits. The maximum range is approximately -1E308 to +1E308 units.
Enumerated	A signed 8-bit enumerated value.
Float	An ANSI/IEEE 754 standard 32-bit floating point value with 1 sign bit, 8 exponent bits, and 23 mantissa bits, for a total of 32 bits. The maximum range is approximately -1E38 to +1E38 units.
Reference	A reference to a network variable type. Uses the type definition of the referenced network variable type. If you are creating a structure or union, an individual field can reference a network variable type. If the referenced network variable type changes in some way, the referencing type or field will automatically change as well.
Signed Character	An 8-bit signed character value.
Signed Long	A 16-bit signed integer value. Maximum unscaled range of -32,768 – 32,767.
Signed Quad	A 32-bit signed integer value.
Signed Short	An 8-bit signed integer value. Maximum unscaled range of -128 – 127.
Unsigned Character	An 8-bit unsigned character.
Unsigned Long	A 16-bit unsigned integer value. Maximum unscaled range of 0 – 65,535.

Data Type	Description
Unsigned Quad	A 32-bit unsigned value.
Unsigned Short	An 8-bit unsigned integer value. Maximum unscaled range of 0 – 255.

Table 14 Aggregate Data Types

Data Type	Description
Structure	A structure containing multiple fields. Each field may be any of the scalar types described in Table 13.
Union	A union containing multiple fields. Each field may be a structure or any of the scalar types described in Table 13.

Scalar data types and fields specify *scaling factors* that can be used to modify the range of the type. The scaling factors are defined by three values called *A*, *B*, and *C*. These values are used to calculate a scaled value as follows:

$$ScaledValue = A * 10^{B} * (UnscaledValue + C)$$

For example, the **SNVT_lev_percent** type is defined to represent a one-byte percentage value. The scaling factors are defined as A=5, B=-2, and C=0, resulting in the following scaling formula:

$$ScaledValue = 5 * 10^{-3} * (UnscaledValue + 0)$$

Using this formula, an unscaled value of 200 results in a scaled value of 100. A value of 1 results in 0.5, providing an 0.5 percent resolution.

Scalar data types and fields define a *units* string that describes the data contained within the network variable or field. This string may be specified in multiple languages to allow localization of displayed values. For example, the English-language unit string for the **SNVT_lev_percent** network variable type is "% of full scale."

Units of measures are typically specified in Systeme Internationale (SI) units. For example, temperature is always represented in Celcius. A standard mechanism is provided to scale measurement values. This allows measurement values to be displayed in alternative systems. For example, temperature can be displayed in Fahrenheit using an appropriate format.

Scalar data types and fields define minimum and maximum values for the network variable type. These values restrict the values that may be assigned to the network variable or field. Scalar data types may also define an *invalid value*. An invalid value indicates that the value of the network variable is unknown. For example, a temperature sensor network variable output that reports an invalid value indicates that the current temperature is not available.

Network Variable Self-identification and Self-documentation

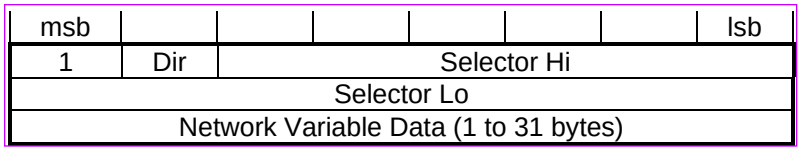
SNVTs are self identifying, meaning that a network tool can extract the type information for all network variables on a device that are declared as SNVTs.

Network variables may also have a self documentation string that can be used to provide more information to a network tool. For example, a device that contains two temperature sensors, one for input temperature and one for output temperature, has two network variable outputs. These outputs are both declared using `SNVT_temp`, automatically identifying the values as encoded Celsius values with a resolution of 0.1°C. Each of the network variables has a self documentation string defined by the device application. The string for the first device is: "Input Temperature Sensor", and the string for the second is "Output Temperature Sensor."

Presentation Packet

The presentation packet defines the physical encoding of the presentation layer data. The first byte of the presentation packet is the message code as described in Table 10 and Table 12. For network variables, the message code specifies the upper 6 bits of the network variable selector.

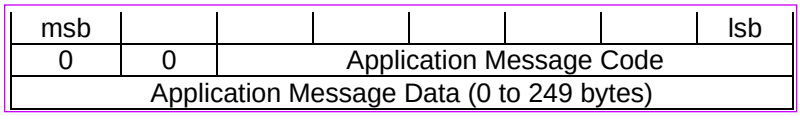
Network variable updates and polls are encoded as CNP messages containing the network variable selector and the new network variable value. The following figure illustrates the format of network variable messages. The **Selector Hi** field contains the top 6 bits of the selector, and the **Selector Lo** field contains the lower 8 bits of the selector. The **Dir** field is 0 for input network variable updates and 1 for output network variables. While only input network variables can receive a new value through layer-6 addressing, both input and output network variables can be read remotely using layer-6 addressing through a network variable poll request message.



msb							lsb
1	Dir	Selector Hi					
Selector Lo							
Network Variable Data (1 to 31 bytes)							

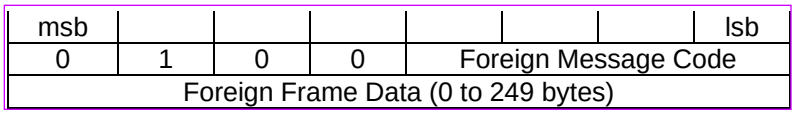
Multi-byte network variables are sent with the most significant byte first. Arrays are sent with the lowest numbered element first. Structures are sent with the first field at the beginning.

The presentation packet for an application message has the following format:



msb							lsb
0	0	Application Message Code					
Application Message Data (0 to 249 bytes)							

The presentation packet for a foreign frame message has the following format:



msb							lsb
0	1	0	0	Foreign Message Code			
Foreign Frame Data (0 to 249 bytes)							

Explicit Network Variable Polls

In certain applications, it is desirable for an application to explicitly create a message that has the same structure as a network variable request message. By doing this, the polling application can bypass the network variable configuration and address tables and explicitly control the destination address of the network variable request. For example, a device application can poll network variables on more than 15 different destination devices (single device or groups) by using explicit addressing to overcome the limit of 15 address table entries on a Neuron hosted device. It can use a single input network variable to receive an unlimited number of responses to polls of any given data type.

Figure 25 illustrates the message structure of a network variable request message used to poll a network variable. The first byte is the message code that includes the **Selector Hi** field containing the most significant six bits of the network variable selector. The most significant bit of the message code is set, indicating a network variable message, and the second most significant bit is set, indicating that the poll is addressed to an output network variable. The first data byte of the request message contains the **Selector Lo** field with the least significant eight bits of the network variable selector; there is no additional data in the request. After the polling device sends the explicit network variable request message, the destination devices will send their network variable values as network variable response messages. A response message contains the same code as the request message, except that the second most significant bit is clear, indicating that the response is addressed to an input network variable. The first data byte of the response contains the least significant eight bits of the network variable selector, and this is followed by the requested data value of the network variable. If the poll is received by a destination device that has no matching network variable, or the destination device is offline, then the response contains the selector, but no data is present.

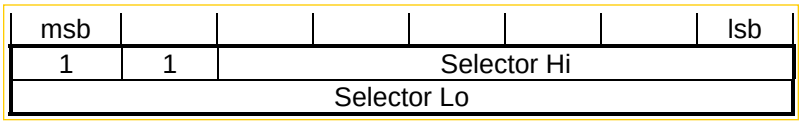


Figure 25 Network Variable Request Message Structure

A response message is processed by the network processor on the polling device, before the response is passed to the application processor. A Neuron C application program cannot receive the response using explicit messaging syntax. If the polling device has an input network variable with the same selector and the same size as the polled network variable, then this network variable will be updated by the response to the poll.

Instead of using an explicit network variable request message, an alternative method of reading the value of a network variable with an explicitly addressed message is with the *Network Variable Fetch* network management message. The Network Variable Fetch message addresses the network variable by its network variable index, while the network variable poll message addresses a network variable (or one of its aliases) by its selector.

When a network variable poll is addressed using group (multicast) addressing with acknowledged service, all members of the group acknowledge the poll request

message. Those members of the group that have output network variables with a matching selector will respond with a message containing the value of the variable. These responses will generate NV update events on the polling device. Those members of the group that have an input network variable with a matching selector, or no network variable with a matching selector, or are offline, will generate a response containing no data. The generation of these responses requires the participation of the application processor in the polled device, and occurs at the end of the currently executing critical section for a Neuron C application. This should be taken into account when designing the application code for a device whose network variables may be polled, and when configuring the transaction timer for the poll message. If all the responses are successfully received by the polling node, an NV update succeeds event is generated. If one or more responses are not received after the configured number of retries, an NV update fails event is generated.

Layer 7—Application Layer

The application layer defines a rich set of standard network services that use data exchanged by the lower layers. These include network configuration and diagnostic services as well as standard application-layer services. The application layer services ensure that devices created by different developers or manufacturers can interoperate with each other, and can be installed and configured using standard network tools. The network configuration and network diagnostic services are defined by the CNP standard. The following list summarizes the standard application-layer services. The remaining standard application-layer services are defined by guidelines and functional profiles published by LONMARK International, a global membership organization created to promote and advance the business of efficient and effective integration of open, multi-vendor control systems utilizing ISO/IEC 14908-1 and related standards. A complete set of guidelines and functional profiles is available at www.lonmark.org.

- ***Network configuration***—provides a standard set of commands for configuring the network attributes of a device. Network attributes include the network address and binding information for the device's network variables.
- ***Network diagnostics***—provides a standard set of commands that can be used by network tools to diagnose network problems.
- ***File transfer***—supports transfer of blocks of data among devices and network tools. LONWORKS FTP transfers large amounts of data (with a theoretical maximum of 2 GB) in smaller packets, taking care of resending individual packets where necessary, and handling out-of-order arrivals as needed. The largest practical amount of data that can be transferred in a single CNP packet is 228 bytes, but LONWORKS FTP transfers data using a stream of 32-byte packets. The size of each packet in the stream is fixed at 32 bytes for interoperability, low device cost, and optimal channel utilization.
- ***Application configuration***—provides a standard interface to configure the behavior of a device. The interface is based on configurable data values called *configuration properties*.

- ***Application specification***—provides a standard set of interfaces for a device to document the tasks that it performs. Each task is exposed as a *functional block* that is defined as a set of network variables and configuration properties contained within a functional block.
- ***Application diagnostics***—provides a standard interface to test functional blocks and devices.
- ***Application management***—provides a standard interface to enable, disable, and override functional blocks on a device.
- ***Alarming***—provides a standard interface for a device to report alarm conditions.
- ***Data Logging***—provides a standard interface for a device to collect data into data logs that can be transferred to a remote server using a standard interface.
- ***Scheduling***—provides a standard interface for scheduling events based on time of day, day of week, and date.
- ***Time and date management***—provides a interface for synchronizing the time-of-day and date for devices within a network.

Application Configuration

A *configuration property (CP)* is a data item that, like a network variable, is part of the device interface for a device. Configuration properties characterize the behavior of a device in the system. Network tools manage this attribute and keep a copy of its value in a database to support maintenance operations. If a device fails and needs to be replaced, the configuration property data stored in the database is downloaded into the replacement device to restore the behavior of the replaced device in the system.

Configuration properties facilitate interoperable installation and configuration tools by providing a well-defined and standardized interface for configuration data. Each configuration property type is defined in a resource file that specifies the data encoding, scaling, units, default value, range, and behavior for configuration properties based on the type. A rich variety of *standard configuration property types (SCPTs* pronounced *skip-its)* are defined. SCPTs provide standard type definitions for commonly used configuration properties such as dead-bands, hysteresis thresholds, and message heartbeat rates. You can also create your own *user configuration property types (UCPTs* pronounced *you-keep-its)* that are defined in resource files that you create with the NodeBuilder Resource Editor.

Application Specification

CNP application provides a standard set of interfaces for a device to document the tasks that it performs. Each task is exposed as a *functional block* that is defined as a set of network variables and configuration properties contained within a functional block. The device interface, also called the *XIF*, may be documented by the device itself, or by a separate file called the *device interface file*, or *XIF file*. The device interface is uniquely identified by an identifier called the *program ID*.

Functional Blocks and Functional Profiles

A CNP device application is divided into one or more *functional blocks*. A functional block is a portion of a device's application that performs a task by receiving configuration and operational data inputs, processing the data, and sending operational data outputs. A functional block may receive inputs from the network, hardware attached to the device, or from other functional blocks on a device. A functional block may send outputs to the network, to hardware attached to the device, or to other functional blocks on the device.

The device application implements a functional block for each function on the device to which other devices should communicate, or that requires configuration for particular application behavior. Each functional block is defined by a *functional profile*. A functional profile is a template for functional block, and a functional block is an implementation of a functional profile.

The network inputs and outputs of a functional block, if any, are provided by network variables and configuration properties. The network variables provide the operational data inputs and outputs for the functional block. The configuration properties configure the behavior of the functional block.

For example, a keypad could implement a functional block based on the **SFPTisiKeypad** ISI Keypad profile, combining a **SNVT_switch_2** typed network variable that represents the current keypad state with configuration properties that specify the state change for each key on the keypad, and hold the name for the keypad. This logical unit—the functional block—can be disabled, enabled, tested, and managed by a network integrator.

Each functional profile defines mandatory and optional network variables and mandatory and optional configuration properties. A functional block must implement all the mandatory network variables and configuration properties defined by the functional profile, and may implement any of the optional network variables and configuration properties defined by the functional profile.

Functional profiles are defined in *resource files*. You can use standard functional profiles or you can define your own functional profiles in your own resource files using the NodeBuilder Resource Editor. A functional profile defined in a resource file is also called a *functional profile template* (FPT).

Each CNP device may include *self-identification* (SI) data and *self-documentation* (SD) data that identifies its device interface to network tools that are used to install the device. Including self-identification and self-documentation data with a device makes it easier to install, as it allows easy, plug-and-play style, integration in multi-vendor networks. While self-identification and self-documentation simplify installation, these methods do not expose any of the algorithms used within the application.

Program ID

The program ID is a 64-bit (16-hex-digit) identifier that uniquely identifies the application contained within a device. A program ID is typically presented as eight pairs of hexadecimal encoded digits, separated by colons. When formatted as a standard program ID, the 16 hex digits are organized as 6 fields that identify the

manufacturer, classification, usage, channel type, and model number of the device. Every standard program ID uses the following format:

FM:MM:MM:CC:CC:UU:TT:NN

The fields are described in the following table.

Table 15 Program ID Fields

Segment	Field	Description
F	Format	A 4-bit program ID format identifier. Set to 8 for LONMARK certified interoperable devices, or to 9 for non-certified application devices. Values less than 8 are used by network interfaces that do not include an application.
M:MM:MM	Manufacturer	A 20-bit identifier for the device manufacturer. A manufacturer ID listing is available at www.lonmark.org/spid. Manufacturers that do not have an ID can either join LonMark International to get a permanent manufacturer ID, or can get a temporary manufacturer ID at no charge from www.lonmark.org/mid. Manufacturers can get more information on joining LonMark International at www.lonmark.org.
CC:CC	Device Class	A 16-bit identifier for the primary function of the device. The primary function of the device is determined by the primary functional profile implemented by the device. The device class is equal to the functional profile number for the functional profile of the primary functional block of the device if the profile is a standard profile and the standard profile is not one of the standard profiles between 0 and 6. For all other devices, the identifier is selected from one of the standard device classes at www.lonmark.org/spid.
UU	Usage	An 8-bit identifier for the intended usage of the device. The most significant bit is set if the device interface is changeable (due to network variables with changeable types being implemented by the interface. The second most significant bit is set if the remainder of the usage bits are specified by the functional profile of the primary functional block of the device. If the second most significant bit is not set, the usage field is set to one of the standard usage values at www.lonmark.org/spid.
TT	Channel Type	An 8-bit identifier for the channel type supported by the device's LONWORKS transceiver. The channel type values are listed at www.lonmark.org/spid.
MM	Model Number	An 8-bit identifier that is used to make the program ID unique for device interfaces that share the same manufacturer, device class, usage, and channel type. This model number does not have to be same as the actual device model number—it just has to be unique from the model number of other device applications that share the same manufacturer, device class, usage, and channel type.

Appendix A.

Layer 1 and 2 Advanced Topics

This appendix describes details of layers 1 and 2 of the ISO/IEC 14908-1 Control Network Protocol (CNP) which are generally of interest to transceiver designers. If you are developing with or integrating one of the standard transceivers described in Chapter 3, the options described in this appendix have already been addressed by the standard transceiver design.

Layer 1 Neuron Communications Interface

The Neuron core supports CNP media independence by using a flexible 5-pin communications port. This port can be configured in one of three modes to support a variety of physical layer transceivers. They are *single-ended*, *differential*, and *special-purpose* modes. For special purpose mode, the Neuron core passes complete data frames to the transceiver which is responsible for data encoding. For single-ended and differential modes, the Neuron core manages all data encoding and uses the transceiver only for the physical layer interface. Table 16 summarizes the pin assignments for each mode.

Table 16 Network Communication Port Pins

Pin	***Single-Ended Mode***	***Differential Mode***	***Special-Purpose Mode***
CP0	Data input	+ Data input	Rx input
CP1	Data output	- Data input	Tx output
CP2	Transmit Enable output	+ Data output	Bit clock output
CP3	~Sleep (~Power-down) output	- Data output	~Sleep output or Wake-up input
CP4	~Collision Detect input	~Collision Detect input	Frame clock output

Single-ended and differential modes use Differential Manchester encoding which is a widely used and reliable format for transmitting data over various media. In this encoding method, there is a minimum of one transition per bit time at the beginning of the bit time. If there is a second transition within the bit time, it occurs in the middle of the bit. By convention, a single transition per bit time is a "1" and two evenly spaced transitions per bit time is a "0." Figure 26 shows the encoding of a sample bit stream.

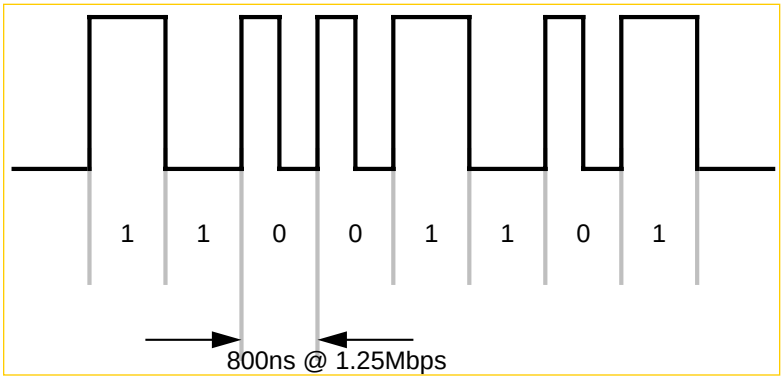


Figure 26 Differential Manchester Encoding

Differential Manchester encoding has the benefits of zero DC offset, polarity insensitivity, and simple bit synchronization between the transmitter and one or

more receivers. Polarity insensitivity simplifies installation of the communications channel because reversal of polarity in the communication link will not affect data reception.

Single-Ended Mode

The single-ended mode is most commonly used with external active transceivers interfacing to media such as free topology twisted pair, radio frequency (RF) carrier, infrared, fiber optic, and coaxial cable.

Figure 27 shows the communications port configuration for the single-ended mode of operation. Data communication occurs via the single-ended (with respect to V_{SS}) input and output buffers on pins CP0 and CP1. Single-ended mode contains an active low sleep output (CP3) which can be used by the transceiver to power down active circuitry when the Neuron core goes to sleep.

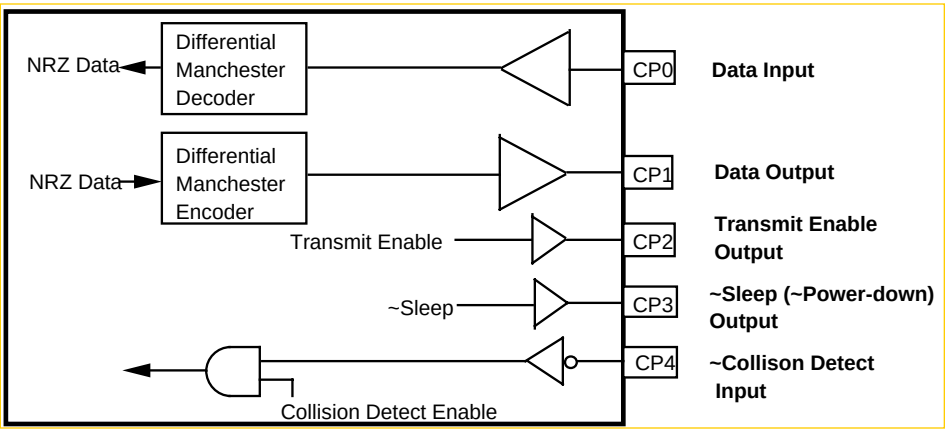


Figure 27 Network Communication Port Configuration in Single-ended Mode

In single-ended mode, the communications port encodes transmitted data and decodes received data using differential Manchester coding (also known as *bi-phase space coding*). This scheme provides a transition at the beginning of every bit period for the purpose of synchronizing the receiver clock, referred to as the *clock transition*. The value of a bit is indicated by the presence or absence of a second transition (the *data transition*) halfway between clock transitions. A mid-cell transition indicates a zero. Lack of a mid-cell transition indicates a one.

Figure 28 shows a typical packet where **T** is the bit period, equal to 1/(bit rate). Clock transitions occur at the beginning of a bit period, and, therefore, the last valid bit in the packet does not have a trailing clock edge.

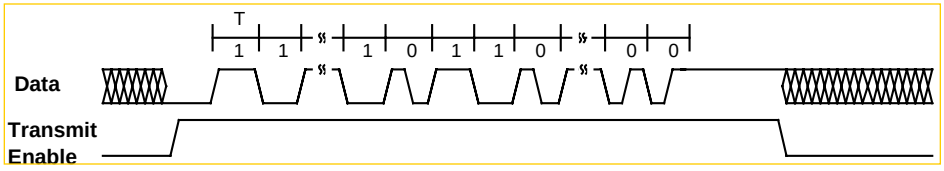


Figure 28 Single-ended Mode Data Format

Before beginning to transmit a data frame, the transmitting Neuron core initializes the output data pin(s) to start low. In single-ended mode, it then asserts Transmit Enable (CP2); this ensures that the first transition in the packet is from low to high. This first transition occurs within 1 bit time of asserting Transmit Enable, and

marks the beginning of the packet. Transmit Enable (CP2) and Data Output (CP1) are actively driven at all times in single-ended mode.

Receiving Neuron cores use two windows for each bit period, **T**. The first window is set at **T**/2. If a transition is detected in the first window, a zero is being received. The second window is at **T**. If no transition occurred in the first window, but a transition occurs in the second window, a one is being received. The second transition also sets up the next two windows (**T**/2 and **T**). If no transition occurs in the first two windows, a Manchester code violation is detected and the packet is assumed to have ended. Table 17 shows the width of this window as a function of the ratio of the Neuron input clock (MHz) and the channel bit rate (Mbps) selected. If a transition falls outside of either window, it is not detected and the packet will contain errors. The receiving Neuron core detects these errors and reports a bad packet. Timing instability of the transitions, known as *jitter*, may be caused by changes in the communications medium, or instability in the transmitting or receiving devices' input clocks. The jitter tolerance windows are expressed as fractions of the bit period, **T**, in Table 17. The MAC clock is 10MHz for a 5000 Series chip running at a 10MHz or greater system clock, is 5Mhz for a 5000 Series chip running at a 5MHz system clock, and is one half the input clock for a 3100 Series chip.

Table 17 Receiver Jitter Tolerance and Line-code Violation Windows

MAC Clock/ Channel Bit Rate	***Next Data Edge***			***Next Clock Edge***			***Line-Code Violation to Receive***
	Min	*Nominal*	*Max*	*Min*	*Nominal*	*Max*	*Min*
4:1	0.375T	0.500T	0.622T	0.875T	1.000T	1.122T	1.62T
8:1	0.313T	0.500T	0.685T	0.813T	1.000T	1.185T	1.46T
16:1	0.345T	0.500T	0.717T	0.845T	1.000T	1.155T	1.46T
32:1	0.330T	0.500T	0.702T	0.830T	1.000T	1.170T	1.46T
64:1	0.323T	0.500T	0.695T	0.823T	1.000T	1.177T	1.46T
128:1	0.318T	0.500T	0.690T	0.818T	1.000T	1.182T	1.46T
256:1	0.315T	0.500T	0.687T	0.815T	1.000T	1.185T	1.46T
512:1	0.315T	0.500T	0.687T	0.815T	1.000T	1.185T	1.46T

For the receiver to reliably terminate reception of a packet, the received line-code violation period must have no transitions until the Neuron core detects the end of the packet. The receiving Neuron core terminates a packet if no clock transitions are detected after the last bit. Table 17 shows the minimum duration from the last clock edge to where the Neuron core is guaranteed to recognize the line-code violation. Data transitions are allowed in this period (and must fall within the data window).

Differential Mode

In differential mode, the Neuron core's built-in transceiver is able to differentially drive and sense a twisted-pair transmission line with external passive components.

Differential mode is similar in most respects to single-ended mode; the key difference is that the driver/receiver circuitry is configured for differential line transmission. Data output pins CP2 and CP3 are driven to opposite states during transmission and put in a high-impedance (undriven) state when not transmitting. The differential receiver circuitry on pins CP0 and CP1 has selectable hysteresis with eight selectable voltage levels followed by a selectable low-pass filter with four selectable values of transient pulse (noise) suppression. The selectable hysteresis and filter permit optimizing receiver performance to line conditions. See Table 18 and Table 19 for specific values.

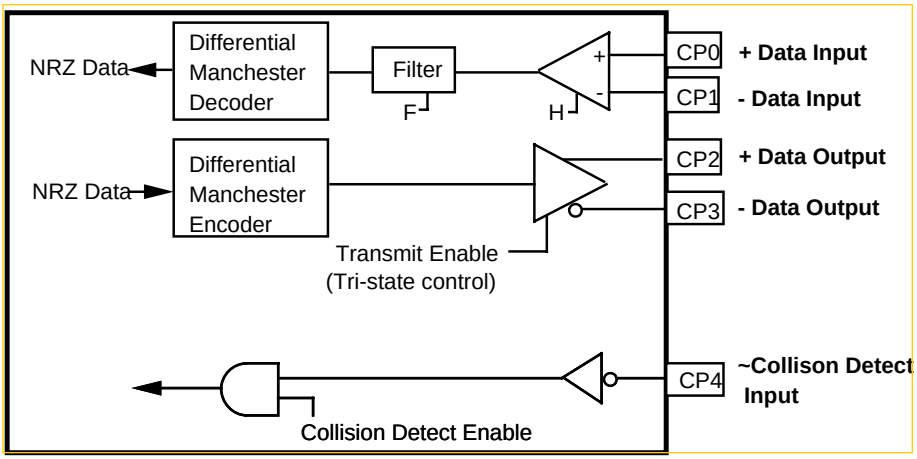


Figure 29 Network Communication Port Configuration in Differential Mode

Figure 29 shows a typical packet waveform in differential mode. The packet format is identical to that in single-ended mode described earlier. The starting level for the data output is the inverse of the last received level to ensure that the first transition occurs on the network as quickly as possible. The coding, jitter tolerance, and minimum line-code violation length to receive apply identically to single-ended mode.

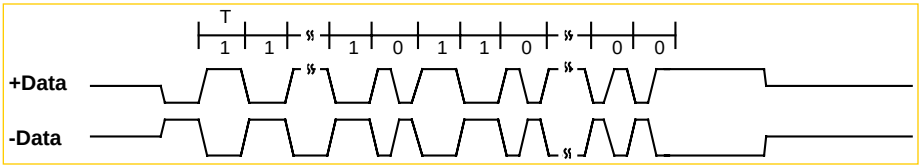


Figure 30 Differential Mode Data Format

Table 18 Hysteresis Values Expressed as Differential Peak-to-peak Voltages in Terms of V_{DD}

Hysteresis (H)	*Min*	*Typ*	*Max*	*Unit*
0	$0.019V_{DD}$	$0.027V_{DD}$	$0.035V_{DD}$	V
1	$0.040V_{DD}$	$0.054V_{DD}$	$0.068V_{DD}$	V
2	$0.061V_{DD}$	$0.081V_{DD}$	$0.101V_{DD}$	V
3	$0.081V_{DD}$	$0.108V_{DD}$	$0.135V_{DD}$	V
4	$0.101V_{DD}$	$0.135V_{DD}$	$0.169V_{DD}$	V
5	$0.121V_{DD}$	$0.162V_{DD}$	$0.203V_{DD}$	V
6	$0.142V_{DD}$	$0.189V_{DD}$	$0.236V_{DD}$	V
7	$0.162V_{DD}$	$0.216V_{DD}$	$0.270V_{DD}$	V

Table 19 Post-hysteresis Filter Values Expressed as Transient Pulse Suppression Times

Filter (F)	*Min*	*Typ*	*Max*	*Unit*
0	2	6	9	ns
1	90	270	580	ns
2	200	535	960	ns
3	410	1070	1920	ns

Special-Purpose Mode

In special situations such as power line transceivers, it is desirable for the Neuron core to provide the packet data in an unencoded format and without a preamble. In this case, an intelligent transmitter accepts the unencoded data and does its own formatting and preamble insertion. The intelligent receiver detects and strips off the preamble and formatting and returns the decoded data to the Neuron core.

Such an intelligent transceiver contains its own input and output data buffers, intelligent control functions, and provides handshaking signals to properly pass the data back and forth between the Neuron core and the transceiver.

In addition, there are many features that can be defined by and incorporated into a special purpose transceiver, for example:

- Ability to configure various parameters of the transceiver from the Neuron core;
- Ability to report on various parameters of the transceiver to the Neuron core;
- Multiple channel operation;
- Multiple bit rate operation;
- Use of forward error correction; and

- Media specific modulation techniques requiring special message headers and framing.

While the special purpose mode offers custom features, most transceivers use the single-ended mode for most types of media, from coaxial cable to RF to fiber optic, because it simplifies transceiver implementation and offers Differential Manchester encoding, which takes care of clock recovery.

Layer 2 Advanced Topics

This section describes the following layer-2 topics that are typically only of interest to transceiver designers:

- Interpacket gap
- Collision detection
- Collision resolution
- Oscillator accuracy
- Preample length

Interpacket Gap

Channel bandwidth is consumed by a series of packet cycles interspersed with idle time. A packet cycle consists of the packet itself followed by the interpacket gap. Packet cycles vary in length due to the randomizing nature of the media access algorithm as well as varying protocol overhead and application data size. This average packet cycle duration determines the rate at which the media access algorithm decreases its assumed backlog of offered traffic during network idle conditions.

The interpacket gap is a variable amount of time following each packet. The interpacket gap ends, and idle time begins, when the MAC sublayers in all devices and routers on the channel reach the idle state (have no packet to send or receive).

The interpacket gap includes the *beta 1 time* and *beta 2 slots*. The beta 1 time is the fixed component in the idle period after a packet has been sent. The beta 2 slots are used for media access control. Figure 31 shows the contents of the interpacket gap.

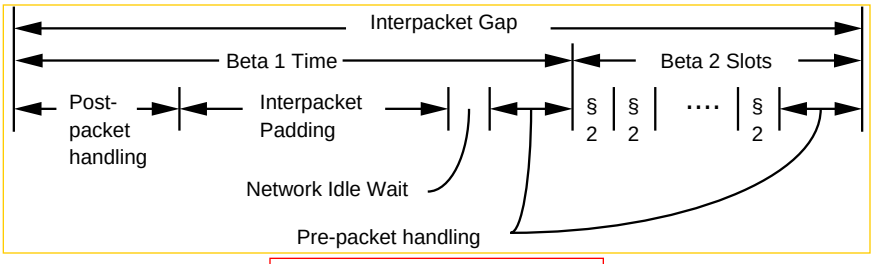


Figure 31 Interpacket Gap

The beta 1 time is a function of the following:

- ***Oscillators:*** Oscillator frequencies and accuracies on the various network devices.

- ***Post-packet handling***. Following the successful transmission or reception of a packet, if the protocol processor is a Neuron core, the Neuron MAC processor performs error checks and handshakes with the Neuron network processor. The number of CPU minor cycles (200ns with a 10MHz input clock) varies as a function of the mode and direction as follows (these times are for Neuron firmware versions 3 and 4 running on Neuron 3120 and 3150 Chips):

Table 20 Neuron Core Post-Packet Processing Time

Direction	*Differential and Single-Ended Mode*	*Special Purpose Mode*
Transmitter	313 cycles	307 cycles
Receiver	295 cycles	285 cycles

- ***Interpacket gap padding***. This padding allows the MAC sublayer in each device to meet a variety of objectives:
  - Synchronize to slower devices on the channel
  - Compensate for *receive end delay*. This is the difference between the transmitting device's view of a packet and a receiving device's view of a packet. This could be due, for example, to buffering in the transceiver. This value does not include variable delays such as propagation delay between the transmitter and receiver or special purpose mode framing delays (these are accounted for in the beta 2 computation). For example, if a special purpose mode transceiver uses loss of carrier for one byte time as the end of packet indicator, the receiver will be one byte time out of sync with the transmitter.
  - Meet the *media indeterminate time*. This is the amount of time following completion of packet transmission where the channel continues to appear to be busy. During this time there could be false transitions on the channel such as transitions caused by ringing, or the transmitter may be unable to detect transitions (due to a transmit-to-receive turnaround time). In either case, reliable network idle detection could not be performed. Therefore, the network idle detection stage is held off until the indeterminate time has passed.

    In differential and single-ended mode, the indeterminate time is measured from the time the code violation is completely output until the transceiver is reliably able to detect valid transitions. An additional 3 bit times is then added to this time. The added 3 bit times is required because the Neuron core receive logic has a 3 bit time memory of network activity.

    The indeterminate time can be determined analytically or empirically. For example, for differential mode, the signal can be examined after a packet and the last point at which it crosses the hysteresis threshold observed. This should be done under worst case conditions. Alternatively, an analytical approach may be used.
  - Meet the *minimum interpacket gap*. This gap provides a means for imposing a minimum on the interpacket gap independent of the other timing factors. This minimum applies to the case where a packet is transmitted in the first beta 2 slot. This can be used to reduce the instantaneous packet arrival rate

or satisfy regulatory requirements such as the CENELEC mains power line carrier access protocol.

In differential mode, the gap is measured from the end of the code violation to the first transmitted transition.

In single-ended mode, the gap is measured from the end of the code violation to the rising TX ENABLE.

- ***Network idle wait***. This period constitutes a check to ensure that the network is indeed idle. The MAC processor ensures there is no network activity for the duration of one beta 2 slot. If there is activity, the MAC processor waits for the channel to go idle.
- ***Pre-packet handling***. Prior to transmitting a new packet, the MAC sublayer selects a beta 2 slot into which to transmit. It must also perform the various I/O operations to begin the transmission. This consumes 270 MAC-processor cycles for differential and single-ended mode and 317 cycles for special purpose mode for Neuron firmware versions 3 and 4 running on Neuron 3120 and 3150 Chips.
- ***Beta 2 slot countdown***. Prior to transmitting, the MAC processor counts down N beta 2 slots, where N is the count selected in the previous step. The size of the beta 2 slot does not vary among devices on the same channel except as a function of oscillator accuracy.

Beta 2 is the interval that is used by both the network idle wait and for media access. The network idle wait constrains beta 2 only in that it must be at least one bit time in width. The beta 2 time is used for both priority and non-priority slots.

For priority beta 2 slots to work, a device choosing to transmit in priority slot N must be recognized as transmitting before any other device attempts to transmit in priority slot N+1. Thus the slots must be wide enough to account for the following:

- Discrepancies among the devices in terms of their views of the end of the previous packet.
- Variances in slot width due to oscillator accuracy.
- Propagation of the transmitted signal to all the devices.

The beta 2 time is a function of the following:

- ***Oscillators.*** Oscillator frequencies and accuracies on the various network devices.
- ***Post-packet handling start***. Different devices will be out of synchronization at the packet end. For Neuron cores using differential or single-ended mode, the variance can be up to one MAC processor minor cycle (300ns with a 10MHz MAC processor clock). In special purpose mode, this variation will be up to one interface frame.
- ***Interpacket variations***. Different devices with different clocks and different interpacket pad values will not have the exact same duration of interpacket overhead.
- ***Priority slot count***. The number of priority slots on the channel.

- ***Neuron core transmit start***. From the time the device protocol software requests start of transmit until transmission actually starts takes up to one bit time in differential and single-ended mode. In special purpose mode, this can take up to one interface frame time.
- ***Receive-to-transmit turnaround***. These are the same components which were discussed for the preamble.
- ***Missed preamble***. This is similar to the missed preamble component discussed for the preamble. It is somewhat different, however, in that it is a measure of the missed preamble which would convey network activity. In differential and single-ended mode, network activity is inferred from any transition and is latched for 3 bit times whereas bit sync qualification cannot begin until valid bits begin appearing. Special purpose mode transceivers make similar distinctions.
- ***Propagation delay***. This is the amount of time for a signal to traverse the worst case distance between two devices.
- ***Receiver delay***. This is amount of time it takes the receiver to recognize channel activity once the signal has arrived. This could take the form of signal propagation delays through the receiver. If a carrier is used, it could be due to carrier detection time.

Special purpose mode transceivers have the ability to reject transmission requests based on the channel activity status. Time to propagate a busy/possible packet condition to the device does not need to be counted.

Figure 32 is an example of the placement of the above parameters. The diagram is not to scale.

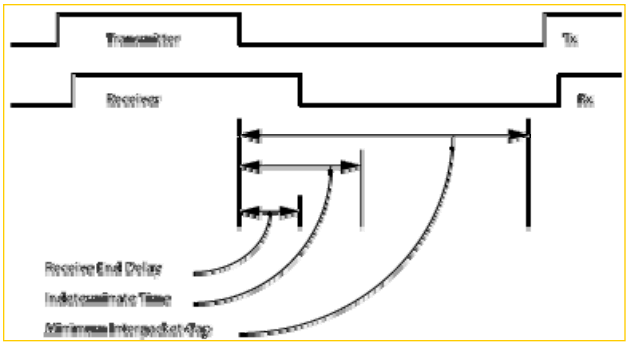


Figure 32 Interpacket Gap Components

Collision Detection

A collision is defined as the event when two or more devices access the communication medium at almost the same time, resulting in mutual interference among their electrical signals on the transmission medium. Packets involved in a collision cannot be successfully received unless collision resolution is used as described in the next section, which results in a delay in response time.

The media access algorithm used by CNP minimizes the number of potential collisions by randomizing access to the medium. Randomizing slots reduce the probability of collisions, however this does not completely eliminate them.

CNP supports optional collision detection. Collision detection hardware at a transmitting device informs the transmitting device shortly after a collision occurs about the need to retry. If hardware is implemented to detect a collision, the firmware cancels transmission of a packet during the preamble upon detecting a collision. This allows the device to immediately retransmit any packet that has been damaged by a collision.

If a device does not use collision detection, then the only way it can determine that a message has not been received is to request an acknowledgement as described in *Acknowledged Service*. When acknowledged service is used, the retry timer is set to allow sufficient time for a message to be sent and acknowledged (typically 48 to 96ms at 1.25Mbps when there are no routers in the transmission path). If the retry timer times out, the device attempts to re-access the channel.

An alternative to acknowledged service described in *Repeated Service* is the *repeated service*. With repeated service, each message is automatically repeated a fixed number of times. This ensures that the message will get though even if one or more collisions occur, as long as the number of collisions is less than the number of retries. This improves response time because the sending device does not have to wait for the acknowledgement before retrying the message.

The collision avoidance built into the predictive p-persistent CSMA MAC sublayer, as described in *Predictive P-Persistent CSMA* earlier in this chapter, is so good that the performance benefit provided by collision detection is minimal.

In differential or single-ended mode, collisions may be detected in the preamble (configurable), near the end of the packet (always), and after the end of the packet (configurable). If collisions are checked for in the preamble, such collisions result in termination of the packet. Otherwise, the colliding packet is completely transmitted. Collision detection is not enabled until 25% of the way into the bit sync portion of the preamble (because false collisions are often a problem at a packet's front end).

Checking for collisions during the preamble is only recommended if it is certain that when one transmitter detects a collision during the preamble, so will all others. If this were not the case, a collider could be left transmitting after termination by the other colliders leaving the lone transmitter with, most likely, a bad packet (due to a corrupted preamble) but with no collision indication.

Checking for collisions after the packet end is not recommended if the code violation might result in a false collision indication.

In special purpose mode, collisions can be detected at any point in the packet. Packets are always terminated immediately upon report of a collision from the transceiver to the protocol processor.

Collision Resolution

Special purpose mode transceivers can implement collision resolution. This is possible because a special purpose transceiver may be able to detect collisions early in the preamble and terminate transmission. The last device left transmitting completes the frame transmission, and all devices backing off can turn around and receive the transmitted frame. To give colliding devices enough time to turn around, the preamble length must be extended as described in the next section.

Oscillator Accuracy

Oscillator accuracy affects many parameters including bit times, beta 1 times, and beta 2 times. The protocol processor on each device is controlled by an independent oscillator with some degree of accuracy. The variance from the nominal frequency is assumed to be symmetric and is typically specified in parts per million.

Wherever asynchronous timing is factored into computations, the oscillator accuracy is considered. This includes processor cycle times, bit times, and interface frame times (as used by special purpose mode). Depending on the circumstances, times are adjusted to their worst case or best case time.

In differential and single-ended modes, the protocol processor receiver synchronizes to the incoming bit stream so oscillator accuracy is not an issue for bit counting conditions such as bit sync threshold. Extreme oscillator inaccuracy will cause loss of synchronization.

Preamble Length

The length of the preamble is affected by the following parameters:

- ***Receive-to-transmit turnaround***. This is the amount of time it takes the transmitting transceiver to switch from receive mode to transmit mode. During this period, nothing is being transmitted on the communications channel, even though the transmitting device is actively transmitting. The device starts timing the preamble when the packet starts from its vantage point.
- ***Missed preamble***. This is the amount of the front end of the preamble lost by the receiving transceiver due to perturbation of the signal as a result of start-up anomalies. For example, on twisted pair, there may be an initial DC offset which results in the first few bits failing to pass the hysteresis threshold.
- ***Packet qualification***. This is the amount of preamble which must be seen by the receiving transceiver's receive logic to decide that a valid packet is present. In differential and single-ended mode, the packet qualification time is controlled by the configurable bit sync threshold (a bit count from 4 to 7 bits). Increasing this count decreases the likelihood that some form of noise will be misconstrued as a packet, but also reduces the maximum channel bandwidth.
- ***Packet response time***. The Neuron firmware detects qualified incoming packets by polling. The worst case interval between detection of the incoming packet and start of the packet input operation is used to compute the amount of preamble needed following packet qualification. Not complying with this means that there is a chance that an overflow condition can occur in the receiver, resulting in a lost or improperly received packet.

  Three different packet response times may be used depending on whether or not collision detection or collision resolution is used. Table 21 summarizes the three response times. These times may be increased for special purpose mode as described in the next section.

Table 21 Neuron Core Packet Response Time

Response Time Type	*CPU Cycles*	*Time*
Standard	227	136.2µsecs * 20/MAC clock
Collision Detection	353	211.8µsecs * 20/MAC clock
Collision Resolution	430	258µsecs * 20/MAC clock

If collision detection and collision resolution are not used, the standard packet response time is used. The standard packet response time is based on the Neuron core's time-to-respond when in the idle state or in various states of preparing for transmission. Packets arriving during the interpacket gap may be lost, however, the probability of a packet arriving during the interpacket gap is typically much lower than the probability of a collision.

A longer packet response time is used with collision detection to avoid losing packets that are transmitted during the interpacket gap. This may occur when devices get out of synchronization with the network due to collisions or noise.

The longest packet response time is used with collision resolution to provide sufficient time for the devices which have terminated transmission to be able to receive the "winning" packet.

Because the Neuron core response time is directly proportional to the Neuron core's MAC clock, the response time for every device on a channel must be based on the slowest device on the channel.

- ***Collision detection***. Special consideration must be made when collision detection is being used. In particular, if collisions are ignored during some beginning portion of the preamble, it is possible that a collision occurring only during that portion would go unnoticed. Therefore, the preamble must be long enough such that the amount of preamble following the collision-ignore portion satisfies the previous three constraints.

  For example, if two devices are transmitting simultaneously and a third device fails to detect this transmission; the third device may transmit near the end of the collision. This can occur if two transmissions cancel themselves so that a third device does not detect either transmission. If the only portion of the third device's transmission which collides is during the collision-ignore period, it will not detect the collision. If the remaining portion of the preamble were not sufficiently long, the packet could be lost.

  The total preamble length is thus determined by adding the contributions of receive-to-transmit turnaround, missed preamble, packet qualification and Neuron core response time. To that sum, the collision detect consideration is factored in.

Example

Here is an example of the components of a differential or single-ended mode preamble. Table 22 lists communications parameters for a sample 78kbps transceiver (this is not the standard TP/XF-78 transceiver type described earlier, it is instead a lower performance transceiver that illustrates how transceiver performance affects preamble length).

Table 22 Example Communications Parameters

Parameter	*Value*
Channel minimum input clock	5 Mhz (1.2 µsec cycle time)
Channel bit rate	78kbps (12.8 µsec bit time)
Oscillator accuracy	2000 PPM
Receive-to-transmit turnaround time	7.5 bits
Missed preamble	2 bits
Packet qualification	6 bits
Neuron core packet response time[1]	274 µsec (227 * 1.2 * 1.002^2) or 22 bits
Collision detection	No
[1] The packet response time is adjusted by the square of the oscillator accuracy to account for the case where the receiver is running at the slowest rate and the sender at the fastest rate.	

This yields the preamble shown in Figure 33.

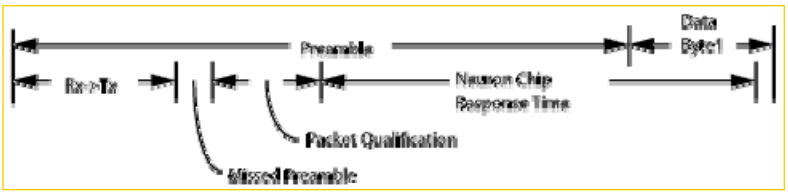


Figure 33 Example Preamble

The preamble is 23.5 bits long (from the transmitting device's perspective). The packet response time was offset by seven bits of the first data byte. Seven was used rather than eight to provide a bit of margin.

www.echelon.com