Rethinking the NTCIP Design and Protocols

Analyzing the Issues

24-Apr-1995[1]

prepared by

Abstract

This working paper discusses the issues involved in changing the current draft NTCIP standard from an X.25-based protocol stack to an Internet-based protocol stack. It contains a methodology which could be used to change NTCIP's base protocols. This paper also includes background information on the current NTCIP draft standard and the Internet network architecture to build a common vocabulary for discussion and to present a review of those topics.

[1]Comments on this document should be addressed to: Joel Snyder, jms@opus1.com, +1 602 324 0494, via FAX to +1 602 324 0495, or via post to Opus One, 1404 East Lind Road, Tucson, Arizona 85719.

Introduction

The Monza implementation team has been tasked with discussing relevant considerations to the use of the Internet Protocol (IP) suite of communication protocols instead of the X.25 suite. In order to best explain the issues involved, it's first necessary to define the role both IP and X.25 present to a networked system, and then examine the differences that the IP suite introduces to such a system.

This document was prepared as a result of user response solicited by FHWA. The Monza team has no doubt that the current NTCIP draft represents an implementable and complete protocol description. However, the use of a different technology base inside of the NTCIP protocol may present additional advantages beyond those offered in the current system.

Modern networked systems are discussed with regard to how their various parts represent the OSI Reference Model of networking, also referred to as the Seven Layer Model. This model of distributed communication includes a layered representation of networking, breaking down "the network" into seven discrete functional areas. Each functional area (layer) presents an idealized interface to an upper layer, and makes use of functionality provided by the lower layers.

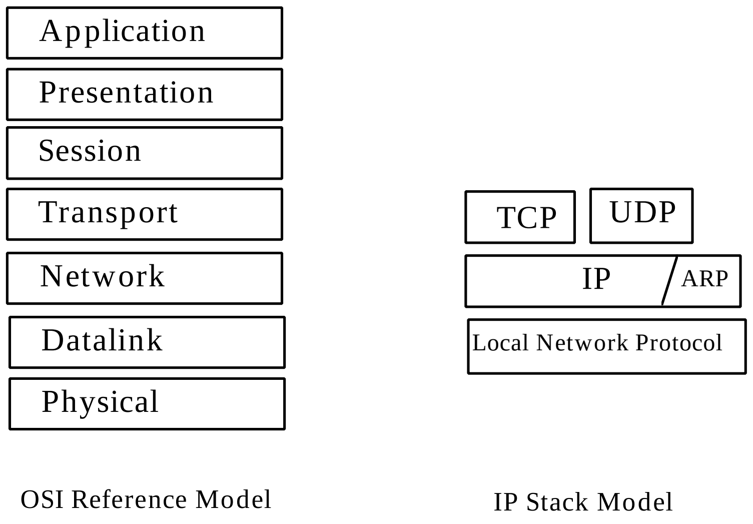


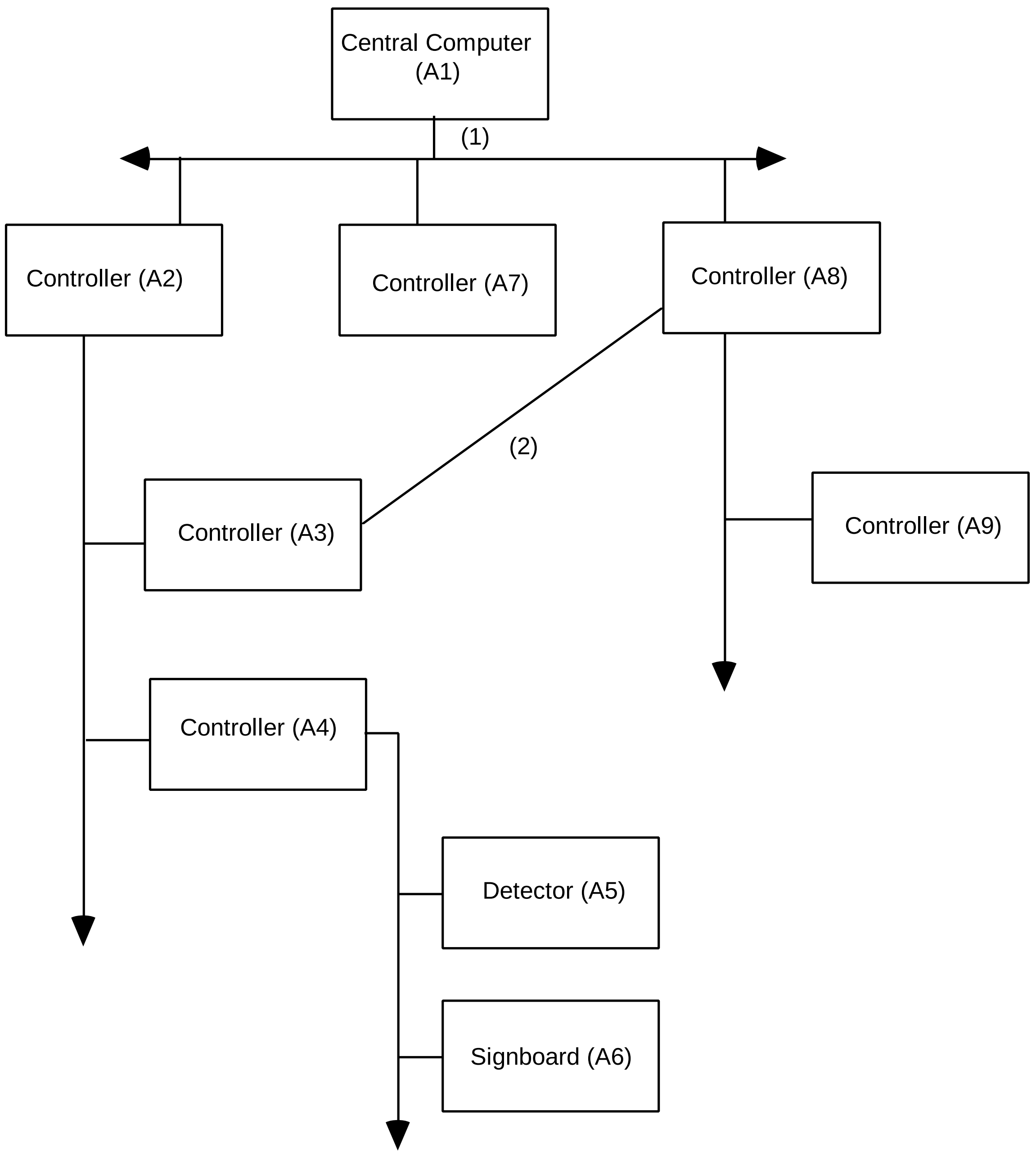
Figure 1: OSI and Internet models

Traffic Control Overview

Before delving into the discussion of how X.25 or IP can be used to solve the problems of the traffic control environment, a review of this environment is given here. This is done in a fashion that is independent of protocol considerations. It includes the topology, link configurations, communication requirements, etc. To the extent that they affect the communications platform, aspects of the traffic control applications are also given here.

Topology

Figure 2 below depicts the general traffic-control topology from which two real topologies are derived. In both cases, the general topological structure is one of a hierarchical tree with a central computer at the root. Each branch (communications link) contains one or more nodes or station types (e.g., central computers, masters, controllers) from which additional branches may emanate. These nodes serve different purposes from a traffic-control perspective but may not be very different, if at all, from a communications perspective.



Notes for Figure 2: (1) Arterial masters may be present at this level. (2) Such links are usually not present (3) Direct non-hierarchical communications between applications, such as between A5 and A9, usually are not present.

Figure 2: NTCIP for Traffic Control Environment

In a centralized topology, the central computer communicates directly with all of the controller units under its control. However, not all controllers need be on the same communications link. For this topology, there is a permanent connection between the central computer and the controllers. Tight control is exercised by the central computer in this topology using a once-per-second control algorithm.

In a distributed topology, an arterial master sits between the central computer and the controller units. These masters are capable of exercising the necessary control over the controller units. For this topology, a permanent connection is **not** necessary between the central computer and the master; when communication is needed, a dial-up connection can be established. A cycle-by-cycle control algorithm is typically used in this topology (it may also be used by some centralized systems).

In either of the above topologies, the communications link may operate at speeds as low as 1200 bps. Because of the real-time aspects of some of the communications (see below), the link operates in a master/slave polling configuration so that control can be exercised as to which station is allowed to put traffic onto the link. One station acts as master while all other stations act as slaves. It is expected that future traffic-control links may operate in a peer-to-peer mode where all stations on a link can originate traffic without being selected by a master first.

In either topology, additional devices, such as detectors and count stations, sit below a controller and exchange data with it.

Temp. Note – Other terms needing definition: field controller, system (input from Ken Vaughn and Workshop).

Application Characterizations

Two types of applications are defined in the traffic-control environment.

Class A applications require that the underlying networking infrastructure provides reliable communications in terms of bit errors and loss. Typical applications might include bulk data transfer.

Class B applications do **not** require reliability from the underlying networking infrastructure other than guarding against bit errors. Loss is **not** of concern since the nature of the information is such that it will be updated on a short-term basis. Typical applications include collection of status information.

Traffic from Class B applications must be given priority by a node for transmission onto a link over any traffic from a Class A application.

Traffic Patterns

Within a given tree as depicted in Figure 2, there may be only one path from a given node to any other node. However, additional paths resulting from non-hierarchical branches are not precluded.

Class B traffic flows only between two nodes on the same communications link (branch). Class A traffic is not restricted in this way – it may flow between nodes on the same link or on different links.

The tree depicted in Figure 2 will exist in many instances – that is, there will be many, many such traffic-control trees within a jurisdiction, county, state, or the nation. It is envisioned that there is **no** direct communication between applications residing in nodes of different trees. Any such communication for traffic control purposes would take place between the applications residing in the central computers which, in turn, would relay the information (possibly in a different format, such as a breakdown of a traffic control pattern into individual controller/unit commands).

On the other hand, Figure 1 above does **not** preclude communications between central computers and other types of traffic systems. However, this was not envisioned to be in the scope of the NTCIP project. Such communications would allow for a central computer to exchange information (other than for traffic control) with a Traffic Management Center or Advanced Traffic Management Systems.

Figure 3 show all of the types of traffic patterns discussed above.

Notes: (1) Communication across traffic control domains, such as between A3 and B25, usually is not present.

Figure 3: Traffic Patterns in and out of NTCIP

Protocol Concepts

This section presents protocol concepts independent of any specific protocol. These concepts serve as an introduction to the following sections of this report as well as background material for the existing NTCIP work.

Basic Requirements

The fundamental requirement of the protocols chosen for the traffic control environment is that they be open in the sense that they are widely accepted (not proprietary) as a result of industry-wide development.

To serve a wide range of current and future needs, an overall architecture should underlie the selected protocols. The Reference Model for Open Systems Interconnection provides such an architecture, as described in the Introduction of this report.

Types of Systems

For the purposes of communications and networking only, it is not important whether a system is a central computer, a field controller, or whatever. For this discussion, a traffic control system can be classified in two ways:

- whether it contains a Class A application, a Class B application, both, or neither (the last case is possible since Class A information, as discussed above, may be exchanged between systems attached to different links, thereby giving rise to the possibility of a system just forwarding such traffic between links); and
- whether it acts as a master or a slave for controlling traffic allowed onto the link.

For the first classification, three logical system types are defined:

1. an End System A is one that has a Class A application;
2. an End System B is one that has a Class B application; and
3. an Intermediate System is one that forwards application information without processing it (which, by definition of Class A and B applications, is just for End System A's).

A real, physical station can be made up of any combination of the above three logical system types. In reality, however, some types of traffic control stations will only consist of certain types of logical system types. Given the nature of the different logical system types, it can be expected that the protocol requirements and functionality for different real stations can be different from one another.

The second classification is straightforward – one station is selected to serve in the roll of master of the communications link while all others serve as slaves. The master station contains a polling schedule by which it selects the slave station with which it needs to communicate. This schedule may depend on the actual configuration and types of stations attached to the link. The designation of a station as a master or a slave does not, by itself, imply any other differences between stations from a communications perspective (although one would expect certain station types to never be masters).

Addressing and Routing

Given a need to communicate between stations, it is necessary to identify each such station in a unique and unambiguous way. This is accomplished by assigning an address to each station with which communications may be necessary. An address is used to determine the location of an object (as opposed to a name, which is location-independent). The location associated with a name may change from time to time.

The scope of an address determines the boundaries in which the address is guaranteed to be unique and unambiguous. Registration authorities (which need not be humans) are used to ensure the necessary scope.

Addresses are independent of particular protocols. As such, the character set used for an address (e.g., 0-9) provides an abstract, protocol-independent basis to describe a station's location. However, addresses need to be carried in protocols to clearly identify the destination of information. One requirement of a protocol is to ensure that it can accommodate both the length and character set used to construct addresses.

Given an address, a path needs to be found to the station identified by its address. The construction of one or more such paths constitutes the routing function. Paths may be statically configured in a system or may be constructed or discovered dynamically.

Reliability

The many types of information handled by different applications result in different requirements for moving the information across a distributed, networked environment. Some applications, identified as Class A applications above, require completely reliable transfer of information. On the other hand, Class B applications do not require complete reliability.

There are several dimensions of reliability:

> bit-error detection: bit errors (a “1” being changed to a “0” or vice versa) may arise as information moves across a link (or even as it moves across a bus within a system); reliability in this dimension requires mechanisms to guard against undetected inversion of bits;
>
> misordering detection: the information transferred between applications may be sent in discrete “packages” called frames, packets, messages or whatever; in some scenarios where multiple frames can be in transit between the sender and the recipient at the same time, it may be possible for frames to arrive in an order different than in which they were sent – such cases require different paths to be traversed by the frames; reliability in this dimension requires mechanisms to guard against inadvertent re-ordering of frames;
>
> loss detection: reliability in this dimension requires mechanisms to guard against loss of a frame and may depend on what assumptions are made about the possibility of misordered frames.

Protocols need to be used such that they satisfy the reliability requirements associated with the application's information. At the very least, reliability requires detection of the particular error type. It may also be desirable to correct the error

Connection-mode and Connectionless-mode Operation

Another dimension in which a protocol may be classified is its need to establish some amount of state information between the communicating entities. For example, the ability to detect loss or misordered frames depends on state information existing (i.e., “what comes next?”) so that such errors can be detected.

The existence of state information is associated with connection-mode (CO) operation. In this mode, an association (i.e., the connection) is required to be established before any information transfer can begin. The establishment of the connection serves to initialize the state information; the establishment and subsequent release of the connection serve to delimit the transfer of information. A typical example of CO operation is a telephone call.

The absence of state information is associated with connectionless-mode (CL) operation. In this mode, there is no connection to be established or released. Without state information, there can be no correlation between consecutive frames (sometimes called datagrams) transferred between two stations. A typical example of CL operation is the mail system, where two letters may arrive in an order different than in which they were sent.

Note that CL operation does not, by itself, imply that any two consecutive frames will ever be misordered. If there is only one path between two stations, then misordering is not possible even without the establishment of a connection and any associated state information. However, basic CL operation does not make such assumptions and, therefore, mechanisms to guard against misordering must be used if this dimension of reliability is needed by the application.

Fragmentation and Reassembly

There may be times when it is not possible to “say everything you need to say at one time.” In such cases, it is necessary to break the message into several smaller pieces. This is also true in a communications scenario where a station, for example, may be limited as to the amount of information it can transfer on the link at one time. Such limits guarantee that no single station can dominate usage of the link with a long message.

Fragmentation (also known as segmentation) provides mechanisms by which a long message can be broken into smaller pieces while also providing the capability to distinguish among different messages. These mechanisms need to take into account whether the resulting smaller pieces can be misordered. Reassembly provides the reverse process of fragmentation.

Internet Architecture

The Internet is a global interconnected mesh of public and private networks all utilizing the Internet Protocol (IP) suite. The original protocol was developed by the United States Department of Defense (US DoD) under its Advanced Research Projects Agency (ARPA). Its roots as a network developed by practical research engineers, Internet standards are peer-review papers called Request-For-Comments papers or RFCs. These RFCs are available on the Internet (ftp://ds.internic.net/rfc) and can be provided as part of the NTCIP Monza project by Opus One on request. The RFC which specifies the Internet Protocol is RFC-791 and is referenced heavily in this document.

Readers who feel comfortable with their knowledge of the Internet architecture and its protocols may feel free to skip this section, as it is entirely tutorial in nature.

Internet Protocol

The network provides datagram-switching services from IP communicating devices. These datagrams are called "IP packets" and the hardware entities creating them are known as "hosts". Each network host has at least one physical device capable of transmitting IP packets. This device is called an "IP interface". Hosts with multiple such interfaces which forward packets between those interfaces are called "gateways" or "routers". Routers are those specialized IP hosts which forward packets on dedicated circuits (often leased lines or point-to-point links) to connect disjoint networks. This organized packet-forwarding function is known as "routing". We referred to "routers" in NTCIP as Intermediate Systems (ISs).

The Internet Protocol is entirely designed to provide packet switched services. It does not provide error detection, error correction, error recovery, reliable service, flow control, sequencing, or any other higher-layer functionality for the data inside of an IP packet. In fact, an IP packet header only includes a checksum for the header itself, and not for any associated data.

Notes: Time To Live was originally specified as a time, but is now a hop count, decremented by one by each router an IP datagram passes through.

Figure 4: Internet Protocol Packet Header example

Since IP is merely layer three (the network layer) of the model, it is up to layer four (the transport layer) to provide any of of those data checking functions.

Physical Layer

Medium Independence

IP was designed to be independent of the restrictions of a particular physical medium. Each physical medium has its own unique set of restrictions. Addressing of stations is a medium-specific concept. IP includes its own addressing (discussed further under the Addressing subsection of the Network Layer).

Address Mapping Mechanism

In order for IP to be able to utilize whatever addressing scheme is in use on the physical medium it is communicating on, IP must have some intermediate protocol to provide address conversion. This MAC-layer protocol is called the Address Resolution Protocol (ARP). ARP takes a medium-specific address (such as Ethernet's 48-bit station address) and maintains and updates a dynamic table mapping those addresses to the IP addresses in use at that station. It can also be updated statically for point-to-point links, serial links, and links not supporting dynamic ARP. This table is called an ARP Table. This functionality is equivalent to the NTCIP X.25 PLP mapping of global addresses to multi-drop line addresses.

Figure 5: ARP Table Example

Datagram Sizing Restrictions

IP's ability to use any datalink medium requires that it be able to effectively determine that medium's maximum datagram size. In IP terms, that Maximum Transmission Unit (MTU) specifies how many bytes of IP header and data can be provided to the lower layer. An Ethernet network, for example, allows MTUs of up to 1514 bytes. FDDI networks allow MTUs of up to 4096 bytes. An IP system running on an FDDI would be allowed to send 4096 contiguous bytes, but one running on Ethernet could only send 1514. For this reason IP also has the ability to fragment large packets into smaller ones, and to reassemble the original packets.

Fragmentation and Reassembly

When the IP host or intermediate router detects that the amount of data it needs to send exceeds the MTU of a particular medium on which it has an interface, it cannot send the data unchanged. If the IP-header flag bit labeled "DF" (Don't Fragment) has not been set, IP will create multiple smaller packets from the original packet. It does so by creating new, smaller IP packets, each having a complete IP header which almost mirrors the original header.   For practical purposes, the new packets are themselves independent IP packets, although the fragmenting host and the recipient host note the "MF" (More Fragments) flag in the header, indicating that these packets should be placed on a packet fragment reassembly queue.

Figure 6: IP Header Flags field bits

The IP system which fragments the packet will place a unique number in the Identification field of the fragment headers, and will place the fragment offset 0-n in the Fragment-Offset field of the headers. Additonally, all but the last fragment shall have the MF flag bit set.

The IP system designated as the end recipient of the packets shall take note of the MF flag being set, shall reassemble the packets in order of fragment offset (for each unique original packet Identification) until it receives a fragment without an MF flag, indicating that it is the last fragment. Providing all fragments have then been received (fragments 0-last) before a fragment reassembly timeout occurs, IP shall then deliver the new and reassembled IP packet to its upper layers.

Figure 7: Fragmentation Example

Broadcast and Multicast Operation

IP can use the ability to address multiple hosts either at a datalink level or at a network level. Broadcasting is the sending of one datagram with everyone on the local network receiving it. Multicasting is the ability to send to multiple hosts with one unicast message. Datalink operations which use broadcasting include ARP requests, where a host will broadcast a request asking "who has the following IP address?" and the host who has that address replies with a unicast message. Multicasting includes the ability to specify a group of hosts to receive a particular message. Both multicasting and broadcasting are a function of a local network protocol which provides it. For example, Ethernet provides broadcasts, but point-to-point serial lines do not.

Summary

IP was designed to be flexible and adapt to any networking medium. ARP maps MAC-layer addresses dynamically. MTU-discovery protocols or static interface constants provide efficient packet sizing. Fragmentation and reassembly allow large packets to traverse smaller-MTU networks transparently to applications.

Datalink Layer

This layer is refered to in IP terminology as the Local Network Protocol. It can be any protocol which encapsulates IP, such as Ethernet, Token-Ring, FDDI, etc. The important item to note is that a standard encapsulation mechanism is required, in order to ensure interoperability of IP on that datalink protocol.

Encapsulation

The encapsulation of an IP datagram (packet) within a local network protocol is usually a function of placing the IP datagram within a local network frame or packet and indicating that the higher layer protocol is IP. For example, in Ethernet, one would make the entire IP datagram fill the "data" portion of the Ethernet frame, and set the Ethernet protocol field to indicate protocol 0800 (hex) for IP.

Figure 8: IP packet encapsulated in various LAN frames

Network Layer

Addressing

IP is based on a simple addressing scheme employing unsigned thirty-two bit addresses. These are often represented as four eight-bit quantities (octets) separated by dots. This form of representing an IP address is known as a "dotted quad". An example of an IP address would be 192.195.240.4. In binary representation it would be `11000000.11000011.11110000.00000100`.

IP addresses are assigned by a global addressing agency in order to provide that all addresses are unique. Organizations seeking IP addresses are given a contiguous range of IP addresses which begin on a boundary of a power of two (e.g. the first address in the provided sequence has a contiguous number of 0-bits on the right-hand side of the 32-bit sequence). This range is called a "network". Modern assigned networks can be any range from 2 hosts through 65,536 hosts (1 bit network through 16 bit network). Past assignments were limited to what used to be "Class" networks.

Figure 9: IP Addressing Example

Class A networks were those whose high bit was 0. These (1.0.0.0-127.0.0.0) were assigned to large agencies who were then allowed to use the 24 bit range of the latter three octets. Class B networks were those whose high bits were 10. These (128.X.0.0 - 191.X.0.0) were assigned to medium-size agencies who were then allowed to use the 16 bit range of the latter two octets. Class C networks were those whose high bits were 110. These (192.X.Y.0-223.X.Y.0) were assigned to smaller agencies who were then allowed to use the 8 bit range of the last octet. Thus with knowledge of the first few bits, IP software could identify the sizing of the local network. Shortage of global addressing space, as well as sizing of router tables in the Internet backbone, has removed this Class distinction.

The new class-less Internet uses what are called Classless Inter-Domain Routing (CIDR, pronounced "cider") style networks. These are specified as a prefix network, and an integer indicating assigned number of left-hand bits. For example, 192.195.240.0/24 refers to an assigned address range (network) of hosts whose leftmost 24 bits are all the same as the leftmost 24 bits of 192.195.240.0. While this may seem redundant or repetitive, note example two, where the network 204.17.32.0/22 means all addresses whose leftmost 22 bits are the same as the leftmost 22 bits of 204.17.32.0, or in this case binary 11001100.00010001.00100000.00000000. Clearly this includes those networks which start with decimal 204.17.33 through 204.17.35. These two examples illustrate that with classless networking, any contiguous addresses starting on a power of two boundary can be assigned, significantly saving global address space.

Routing

IP software is responsible for routing of network-layer datagrams. All IP software implementations (IP Kernels) currently utilize a next-hop routing algorithm. In simple terms, when examining a datagram not destined for the local host, the IP kernel compares the network-layer destination address against a table of potential next-hop gateways (the Routing Table). If a match is found, indicating that a particular gateway supports a known method of reaching the packet destination, then the packet is forwarded to that gateway. Necessarily, all gateways in the routing table must be adjacent on some IP interface. If this were not so, then a gateway would be more than one hop away, and routing to it would require using an intermediate gateway. Packets forwarded on the local network are encapsulated in the appropriate local network protocol, and sent on that medium to the designated (intermediate) next-hop destination.

Routing tables are 2-tuples, mapping from destination network (range of hosts addresses) to the gateway to use. Modern implementations include an optional further specification of which interface to send the datagram on in case of ambiguity, as well as relevant route flags.

Figure 10: IP routing example

Flat Virtual Address Space

With IP's responsibilities to routing, and its attention to the number of bits representing a network, the upper layers need not ever concern themselves with the physical topology of the network. IP presents those upper layers with a virtual address space of 32 bits. For example, it is not necessary for an application-level protocol (say a mail program) to figure out if its address and the destination address are on the same "network" or group of hosts, or whether IP packets traveling to that destination route through one or more intermediate gateways. Those details are obscured by IP as part of its layer function to take care of all routing, addressing, and host-grouping issues. Of course, this same function is provided by any network layer protocol.

Error Reporting

IP itself includes a variety of features designed to minimize potential error conditions. Nevertheless, the nature of networking requires that a mechanism be in place to provide information about errors, route failures, packet delivery failure, etc. To provide this information, the host discovering the failure sends a datagram to the host originating the original packet. However, to prevent this datagram from potentially encountering an error itself, and thus creating an infinite loop of error-condition-notification packets, the packet is specially marked as an Internet Control Message Protocol (ICMP) packet.

ICMP

ICMP is an IP packet with the IP protocol-type field set to indicate an ICMP packet. ICMP packets are used to report error conditions on packet-delivery or forwarding of **IP only.** ICMP is not used to report errors about ICMP packets (to preclude the aforementioned infinite loop) or about ARP, or any other protocol.

ICMP provides multiple classes of errors indicating in general host unreachability, network unreachability, packet time-to-live exceeded in transit, or remote host service unavailable (for higher layer protocols using client-server architecture).

Figure 11: ICMP error codes and types

Transport Layer

The IP protocol suite includes a variety of upper layer protocols which operate reliably and efficiently with IP as a lower layer. These include TCP, UDP, RDP, and several others.

Figure 12: Applications using reliable transport layers over IP network layer

TCP

Transmission Control Protocol is the most popular of the transport layer protocols used with the IP suite. TCP provides an end-to-end reliable byte stream. It implements sliding-window flow-control, multiplexing of virtual circuits ("connections") over virtual circuit "ports", data sequencing, error detection, error correction, retransmission, and out-of-band data delivery. All data is sequenced, and acknowledged, with acknowledgement numbers representing successful data transfer on each of the two half-duplex pipes that form the full-duplex TCP connections.

TCP is a complex protocol, as far as the IP suite goes, providing all of the functionality that any higher layer application might ever need. As a result of this inherent complexity, the TCP header is a large and cumbersome data structure. Fortunately, once a connection has been set up ("established") most of the header fields are ignored, until the connection is torn down.

Figure 13: Example TCP Header

TCP Connection Multiplexing

TCP multiplexes and demultiplexes connections by assigning each local part of a connection a port number identifier. Port numbers are 16-bit unsigned quantities. Some ports are assigned to be used for particular services, and these are called Well-Known-Ports. Services utilizing these ports (e.g. mail, remote login, etc.) are called Well-Known-Services or WKS.

TCP Connection Establishment

A host and port combination is known as a Socket. Two sockets connected together are known as a Socket-Pair or "Connection". A communicating entity ("process") may have multiple such connections, and may even have multiple connections with the same local port number (and different remote port or host numbers). In order for a connection to be established, TCP requires that both sides indicate their TCP sequence numbers, and that each acknowledge the other's sequence space. To prevent a potential mutual-open-attempt with identical port numbers, TCP uses a 3-way handshake. They do so by sending packets with the SYN flag bit set, indicate a sequence-number synchronization. Response packets have the ACK flag bit set, indicating the Acknowledgement Number field has valid data.

Figure 14: TCP Connection Establishment

TCP Flow Control

Flow control is provided by the TCP header "Window" field. The window field indicates how many bytes past the last acknowledged data this half-duplex pipe is willing to accept. Data with sequence numbers past the window size are silently dropped by the recipient. The Window can grow or decrease in size as the buffer empties or fills, and its sequence number grows as the Acknowledgement number grows. Hence it is a variable dynamically-sized sliding window flow-control scheme.

Data Reliability

TCP includes a checksum for the TCP data and the TCP header. Thus, TCP is the lowest layer at which data delivered to the upper layers is guaranteed to be correct. TCP positively acknowledges data received correctly, and retransmits data which is not acknowledged in a timely manner. With positive acknowledgements, a retransmission queue, and a 16-bit ones-complement checksum algorithm, TCP delivers reliable data. (Note that there are more comprehrensive checksum and CRC algorithms. This one was selected for use by TCP and IP because it is computationaly cheap to calculate.)

Connection Teardown

TCP connections are torn down when either sides indicates a desire to close the connection (or when one side explicitly requests a reset of the connection). The FIN or RST flag bit signifies this condition. RSTs are immediate resets and incur no painless teardown. FINs are for controlled circuit close. A 3-way handshake similar to the SYN handshake is used for FINs.

Figure 15: TCP Connection Teardown

Summary

Because of its ease of use, a platform-independent programming interface ("Berkeley Sockets"), and the reliable byte-stream functionality, TCP remains the most popular transport layer protocol. It utilizes large headers, sliding windows, and positive acknowledgements to provide sequenced reliable data transfers.

UDP

The User Datagram Protocol provides datagram-level services to IP network users. As a transport protocol, it provides multiplexing, error detection, but no error correction, no flow control, and no virtual circuit management. Because of this, UDP is considered a lightweight protocol, and its smaller header reflects this.

Figure 16: Example UDP Header

UDP Error Detection

UDP packets include a checksum for determining whether the data arrived successfully. UDP includes no facilities for reception acknowledgement, packet retransmission, or sequencing.

UDP Connection Multiplexing

Like TCP, UDP multiplexes and demultiplexes connections by assigning each local datagram sender or recipient a port number identifier. Port numbers are 16-bit unsigned quantities. Some ports are assigned to be used for particular services, and these are called Well-Known-Ports. Services utilizing these ports (e.g. Network Time Protocol, Network File System, etc.) are called Well-Known-Services or WKS.

UDP Summary

UDP provides a connectionless datagram-oriented multiplexed delivery service. No sequencing, retransmission, flow control, or acknowledgements are provided by the protocol. It is a lightweight protocol both in terms of header size, programming ease, and network overhead.

RDP

Reliable Data Protocol was suggested in July 1984 as a protocol providing reliable end-to-end data yet implementing only a subset of the features that have crept into TCP. RDP was primarily designed for batch loading of remote machines. Hence it was intended to be simple to implement but not ideal for interactive-style applications. Unfortunately, it never became popularly implemented, and its use lies somewhere between "deprecated" and "obsolete."

Figure 17: Example RDP Header

Optional Sequencing

Data sequencing using RDP is optional. Upon connection formation, sequenced delivery may be specified, or unsequenced delivery may occur. The application should determine which is necessary based on its concept of the capabilities of the remote host.

Feature Lack Summary

RDP does not implement window flow control, segment size negotiation, urgent (out-of-band) data delivery, or any of the other options which are available in TCP. This makes RDP a poor choice for interactive sessions requiring out-of-band data transfer (e.g. ^C or ^S). RDP has no standard programming interface, unlike the Berkeley Sockets used with TCP and UDP. RDP employs a variable header size.

Reasons to and not to revisit the architecture

After working with the NTCIP draft standard, the Monza team believes that it is basically stable and represents a viable description of networking protocols which could be used in traffic control systems. There is no significant technical fault with the approach taken in the NTCIP draft standard.

However, there are reasons why it is desirable to consider revisiting parts of the NTCIP draft standard, including the following. Please note that these issues are entirely orthoganol to the question of replacing the NTCIP network layer protocols: they should be considered and discussed regardless of any decision on the packet layer protocol.

- The NTCIP standard uses a data link layer based on HDLC and a reduced X.25 packet layer. The changes adopted in the draft standard document are significant enough that off-the-shelf debugging equipment will not decode the NTCIP protocols and off-the-shelf X.25 source codes would require significant modification to support the NTCIP requirements.[2] To speed the development and debugging process, it would be advantageous if engineers could use off-the-shelf equipment to monitor NTCIP links. This could be accomplished by revisiting the data link layer of NTCIP so that a more standard approach is taken.

- A connection-oriented X.25 network requires maintenance of hop-by-hop connection tables. This increases the complexity of the network to a considerable degree. A significant amount of the design effort on the Monza prototype went into analyzing and supporting the needs of connection maintenance for Class A communications.

- The initial requirement that all stations support both class A and class B communications in an operating-system-less environment has been loosened since the initial design was presented. By far the greatest Monza design went into building a pseudo-operating system and run-time library to properly handle communications traffic in the presence of an absolute minimum in the way of interrupts and operating system facilities.

[2]This is true for the data link layer and, to a lesser extent, for the packet layer.

If the design of NTCIP is changed so that controller-type end systems (the lowest level of a traffic system, both in network topology and in system capabilities) are only required to support Class B communications and that any system which supports both Class A and Class B communications can be assumed to have a simple operating system (say, at least as powerful as Microsoft's MS-DOS or even Unix), the overall complexity of the Monza prototype would drop by 30% or more.

- The message types being defined in the NTCIP application layer have changed scope so that they closely resemble a transaction-oriented GET/SET protocol (SNMP, the Simple Network Management Protocol, is being used as a model). The overhead of a complete connection establishment for Class A communications to support such simple transactions, which will typically require only two data packets, is considerable. An architecture which more efficiently supports simple transactions should be considered for NTCIP.

- Dynamic address assignment or acquisition can simplify field maintenance and initial configuration of networks. Because there is no provision in the NTCIP drafts for dynamic address assignment (*a la* IP's ARP, RARP, and BOOTP protocols), adding this feature set should be strongly considered.

The prior five reasons suggest that the NTCIP architecture in the current draft protocol document could be revisited with a view to reducing cost and time-to-market for equipment manufacturers who wish to support the NTCIP protocols.

In addition, there are reasons to consider replacing parts (or even all) of the NTCIP protocols with ones based more closely on the Internet architecture, including:

- The enormous popularity of the Internet has made knowledge of the Internet protocol suite a common commodity. While any competent communications engineer will understand the design of an X.25 network, these are outnumbered by tens of thousands of undergraduates who are being exposed to the Internet as part of basic business, computer science, and engineering degrees. This easy availability of knowledge about TCP/IP would make it easier for traffic equipment manufacturers to find the new knowledge they need to incorporate NTCIP into new equipment.

- Although freely available software is typically not suitable for inclusion into real-time control systems, the enormous number of IP kernel implementations available for easy inspection by traffic engineering manufacturers will make the job of building data communications kernels into traffic equipment simpler and less expensive.

- While the scope of the NTCIP protocols does not include external connections to analysis systems or traffic management systems, the need for these connections is obvious. Because of the popularity of IP-based networking, it may be a nearly universal requirement for future traffic control systems to include Internet-architecture network protocols to support these additional connections. If this is true, than having a single protocol stack will be simpler, less expensive, and more robust than having two stacks (Internet TCP/IP and NTCIP) in a traffic control or management system.

There are also reasons for **not** replacing the X.25 component of the NTCIP protocols with IP protocols:

- The header overhead of IP's datagrams is fixed at 20 octets (or more, if options are specified). Once a logical channel has been established, X.25 per-packet overhead is only 3 octets. Although IP headers are easily compressed, there is no standard for compressing IP headers alone–TCP and IP are both required. If the environment in which IP is used is strictly limited in bandwidth, then IP has a high overhead which may limit its usefullness. Alternatively, non-standard IP header compression would compress the header somewhat but would cause interoperability problems with non-NTCIP IP implementations.

- Testing in the TCP/IP world usually focuses on interoperability bake-offs rather than strict conformance testing. This approach is looser than that traditionally used in X.25 protocol implementation testing, with its thousand-plus pages of conformance tests. Although no test methodology is perfect and will uncover all problems in an implementation, the conformance testing methodology of X.25 offers a greater confidence of long-term interoperability than the one-on-one testing traditionally used with Internet protocols.

- The idea of a PICS (Protocol Implementation Conformance Statement) does not really exist in the Internet world. As part of years of experience, the collective wisdom of TCP/IP implementors was collected in the Host Requirements RFCs (currently RFC1122 and RFC1123), which serves many of the purposes of a PICS. However, there are still implementation choices available which could cause interoperability failure even while the "letter of the law" was followed.[3] Therefore, the function of the PICS, while not obvious in the Internet world, has especially important value in systems where interoperability is key.

Summary

While the NTCIP architecture is technically sound, experience in the design and implementation of the Monza prototype and changes in the system requirements offer an opportunity for improvement of the draft standard before final publication.

Compatibility and cost issues associated with the use of the IP protocol as a base along with roughly equivalent technical capabilities make it a candidate for inclusion in the NTCIP standard as part of the improvement process.

Balancing these three sets of issues to build the best NTCIP architecture will require agreement on the environments for class A and class B communications, including application requirements and hardware capabilities.

[3]It should be noted that these failures are rare and usually only occur in very complex systems stretching the limits of Internet networking technology. Traditional Ethernet-based IP networks normally interoperate at the IP, UDP, and TCP levels.

Replacing Class A, Class B, or Both Types of Communications

When revisiting the NTCIP architecture, one could consider changing the model for Class A or Class B communications, or for both.

Class B communications currently only require the simplest of protocol elements: the basic NTCIP HDLC (data link layer) protocol plus a single byte of protocol identification data. Class B communications are not routable; they only allow a master station to communicate with a directly connected slave station.

If Class A communications were changed to use IP as a network layer protocol (plus some other transport mechanism) and Class B communications were changed to use IP as the transport over the current NTCIP HDLC, then a new capability would be created: stations more than one hop away from each other could communicate using Class B communications messages.[4]

In the current architecture, the only way that this capability is supported is via a relatively complex application-layer relay. The relay must not only manage address translation, but must also maintain state information about pseudo-connections through it. More importantly, the presence of such a relay severely restricts the possible range of communications between Class B systems and any other systems. In particular, only a single communications connection can be maintained at any one time **and** the Class B station could not asynchronously offer status information during the lifetime of a connection. By making the end systems which support Class B communications IP-compatible, the need for this application-layer relay is completely eliminated.

In the current system requirements definition, there is no expressed need for "routed" Class B communications. In the absence of such a requirement, it is prudent to consider the additional stress which an IP kernel would place on end systems which have limited memory, communications bandwidth, and CPU power.

Relative to the existing Class B communications, a new Class B which includes both HDLC and standard IP (but not TCP) requires:

Communications bandwidth	20 bytes addt'l per packet
CPU cost	thousands of instructions per packet

[4]Note that this was never expressed as a requirement for NTCIP.

Memory cost	8K to 64K for IP protocol and associated buffers

Summary

If Class A communications are changed to be IP-compatible, then Class B communications could be changed as well or left alone.

	Advantages	**Disadvantages**
Changing Class B	End-to-End communications possible Elimination of application relay (possible advantage)	Very high additional cost in memory, CPU, and bandwidth Any existing work invalidated
Leave Class B as-is	Simpler and smaller implementation Reduce time-to-market	Dual stacks in intermediate systems (IP and Class B)

Compatibility of A and B communications

If Class A communications are changed to use IP while Class B communications are unchanged, there are environments where both Class A and Class B communications pass over the same physical layer.[5]

Because Class B communications are only defined over the NTCIP draft data link layer (HDLC UI frames), this issue is reduced to that environment.

Although it might be possible to construct a second data link layer for the Class A communications over multidrop lines, this is strongly discouraged. Having two data link layers would complicate both the implementation and debugging, would take additional memory and CPU resources, and would provide no substantial benefits.

To operate both IP and Class B communications over the same data link layer requires a protocol demultiplexing layer similar to the one in the current NTCIP draft. Because IP is designed to operate in parallel with other network layer protocols, IP provides a mechanism to identify different protocols using the high nibble (4 bits) of the first octet of the packet header.

These values are allocated in the "Assigned Numbers" RFC (currently RFC 1700, dated October, 1994):

Decimal	**Keyword**	**Version**
0		Reserved
1-3		Unassigned
4	IP	Internet Protocol
5	ST	ST datagram mode
6	SIP	Simple Internet Protocol
7	TP/IX	TP/IX: The Next Internet
8	PIP	The P Internet Protocol
9	TUBA	TUBA
10-14		Unassigned
15		Reserved

It would be possible to select a value for the protocol demultiplexing layer such that the protocol demultiplexing layer overlays with the first byte of the IP header. This would cause problems if the value were chosen incorrectly and

[5]In the existing NTCIP protocol draft, both class A and class B communications pass over the same physical and data link layer using a protocol demultiplexing octet to distinguish them.

interoperability at some future date with a future protocol were required. To do this, an unassigned value could be selected in coordination with the IANA (Internet Assigned Number Authority) and be used as the protocol demultiplexing indicator. The bottom nibble of the first octet would be ignored.

This will allow an IP layer to run unchanged directly over the data link layer without significant overhead and will allow the same IP traffic to be easily routed to a larger internetwork. It is unlikely that the future compatibility problems will be offset by the value of saving a single octet for IP traffic (note that Class B traffic itself would not save the octet since something is still required).

As an alternative, a "single-protocol" mode could be defined for links which will run only Class B traffic to omit the demultiplexing byte. This would be a configuration-time parameter which applies to all stations on a particular multi-drop circuit. Again, it is unlikely that the savings of a single octet would be worth the long-term effects of having a configuration parameter which is easily mis-configured and would cause two systems to fail to interoperate. (Interoperability failure would occur if the parameter were misconfigured initially or if class A traffic was later put on a link but an existing system was not reconfigured.)

In any case, if the existing Class B and an IP-based Class A communications must operate over the same non-NTCIP data link layer,[6] it would be important to revisit the design of the protocol demultiplexing layer so that Class B traffic is easily distinguished from Class A traffic by existing standardized IP systems. In the current definition of the protocol demultiplexing layer, Class B traffic could not be carried transparently over an existing network. If it is anticipated that Class B traffic will be carried over non-NTCIP data link layers (such as SLIP or Ethernet), then an appropriate and compatible demultiplexing layer would have to be created.

To operate IP over the NTCIP data link layer will also require the definition of a convergence function to map between IP service requirements and NTCIP data link layer (the "Local Network Protocol" in IP terms) services. This is discussed in the following section ("Swap out NTCIP A for IP-based A").

[6]For example, a SLIP or PPP circuit.

Summary

Changes in the protocol demultiplexing strategy of Class A and Class B communications could save a single octet for each packet, but only at considerable risk of future compatibility or configuration complexity.

If Class B traffic will be carried over non-NTCIP data link layers, than a different protocol demultilexing layer will need to be defined for each of the other data link layers.

A Local Network Protocol must be defined to allow IP-based Class A communications to operate over the NTCIP data link layer.

Swap out NTCIP A for IP-based A

This section discusses the changes which would have to be made to the current NTCIP draft at the network layer only to support IP in place of the current protocol. These changes fall into four categories:

1) How will the NTCIP protocols operate over the physical layer? How will the physical-address to IP mapping work?

2) What options from RFC 791 will be supported? What options from the Host Requirements RFCs will be supported?

3) How will the ICMP function be supported?

4) How will routing be handled?

For convenience sake, we'll refer to "IP-A" as a hypothetical replacement for the network layer using IP.

Local Network Protocol Issues

The local network protocol is responsible for managing the interface between the IP layer and the physical layer. Characteristics of the local network protocol include a MTU (maximum transmission unit), multicast and/or broadcast attributes, and communications characteristics such as latency and thruput.

Operation over the NTCIP data link layer will assume polling such that the multi-drop line appears to all stations as a half-duplex point-to-point link. (Note that round-robin polling would be one way to accomplish this, but that other polling strategies could also be appropriate.) No additional protocol elements to control master-slave polling will be required. In the version of the NTCIP protocol which we have available at Opus One, round robin polling is not explicitly described. This should be added to the NTCIP protocol, whether or not IP-A is included.

IP Topology

To operate over the NTCIP data link layer, IP-A must treat the layer as either a point-to-point interface (in the case of a slave system) or as multiple point-to-point interfaces (in the case of a master system). This is the simplest model. A variant would allow the master to simulate multicast or broadcast communications. However, there is no expressed need for this in the NTCIP requirements.

Support of multiple point-to-point interfaces in the master would typically require that the master emulate a multi-homed host, maintaining a separate IP address for each end of each interface. This, combined with the necessary subnet procedures, would make configuration of a heavily multi-drop service quite confusing. To avoid this, a mechanism should be designed which fits within the framework of the IP addressing and routing techniques to provide a simple configuration for multi-drop service.

In environments where multi-drop links are in use, this could be created through different network masks on each interface (including a 32-bit mask at the furthest leaves on the multi-drop side). This is a simple and easily configurable system which fits in the IP addressing scheme with a minimum of confusion.

Other possible mechanisms for dealing with this configuration issue are possible. It would be valuable for the implementers of NTCIP to receive some guidance in this area to encourage similar configuration paradigms across vendors.

MTU

The current suggested Data Link layer I field maximum is 515 octets. If one octet is lost to protocol identification, this gives an MTU of 514. This offers no incompatibility with existing IP implementations. To reduce fragmentation, some guidance about suggested data link layer I field sizes greater than 514 (say, for example, 577, which would be a better match for other WAN links) should be provided in the NTCIP protocol specifications. Raising the minimum data link layer I field size to 577 might also be appropriate.

Note that the original MTU size of at least 514 octets was chosen based on the projected size of some NTCIP PDUs. If the majority of NTCIP PDUs fit into a smaller packet, as has been suggested will be the case, there may be no reason to have an MTU size that large, and it could be dropped to reduce memory buffer requirements in end systems. Note, however, that small MTUs increase the overhead percentage which packet headers consume when packets are fragmented.

MAC-to-IP address resolution

A mechanism for mapping MAC layer addresses to IP addresses will be needed when operating IP-A over a multi-drop physical layer. In operation over Ethernet, IP normally uses ARP, the Address Resolution Protocol, to discover MAC addresses given IP addresses. Some environments also use RARP, Reverse ARP, to discover IP addresses given MAC addresses.

Neither ARP nor RARP could run un-changed over the NTCIP data link layer. However, the function of ARP and RARP could be provided by a slightly modified version of these protocols.

NTCIP implementors may wish to use one or both of these protocol functions to assist in configuration of traffic control systems or in re-configuration of systems after equipment failure. For environments which are largely static, however, configuration-time assignment of MAC-to-IP mappings is not unreasonable.

IP-A operating over traditional broadcast media, such as Ethernet, should use standard ARP protocol. IP-A running over serial line protocols such as SLIP could use either static configuration or dynamic discovery of IP-to-MAC address mappings.

It is probably a good idea for NTCIP to support the ARP and RARP functions, but not to require it. In the interest of speeding time to delivery of IP-A specifications, we suggest that a placeholder in the NTCIP protocols be left for the function of the ARP and RARP protocols over serial interfaces (such as SLIP or the NTCIP data link layer), but that these not be included in the first version.

DHCP, the Dynamic Host Configuration Protocol, is a newer Internet protocol which builds on the base of RARP (and BOOTP, the Boot Protocol). It would be prudent for NTCIP to investigate including DHCP instead of RARP if a placeholder is left for the RARP-function of dynamic learned configuration.

For the purposes of configuration simplicity in MAC-to-IP mapping, the 1, 2, or 3-octet data link level addresses used in the NTCIP data link layer should be padded internally to right-justified 4-octet addresses. For example, a 2-octet address of `0xCAFE` should be considered internally as a 4-octet address of `0x0000CAFE`.

Non-NTCIP Serial Lines

For simple point-to-point links carrying only IP-A traffic (see, for example, communications between A1 and A8 in the figure below), implementors should be given the choice of a non-NTCIP data link layer such as SLIP or PPP. For reasons of simplicity and because they offer technical equivalence, SLIP should be indicated to NTCIP implementors as the preferred serial line protocol.

RFC-791 Options

A number of fields and options are defined in RFC-791 which may either be inapplicable to NTCIP applications or undefined in the NTCIP environment.

The notes below apply mainly to the issues of changing only class A communications. If class A and class B communications were changed, then the issue of fragementation (i.e., DF bit would always be set in class B communications) should also be addressed.

Type of Service

Most gateways ignore the Type of Service octet in the IP header. NTCIP applications should not assume that this octet can be used to indicate type of service.

IP Options

Few of the options are implemented by common IP kernels and/or gateways. NTCIP applications should not assume that IP options will be useful. For pure NTCIP-to-NTCIP communications, it would not be unreasonable to specify that the IP option field can be ignored in IP-A. However, it would be poor practice to require that no options ever be present.

Addressing

The new CIDR addressing ("classless" addressing) is more flexible and particularly suited to the hierarchical topology of traffic control networks. It would be a good idea for NTCIP IP-A to require support for classless addresses (sometimes called "supernets" and "subnets"). See also RFC-950.

Time To Live

Some IP kernels have hard-wired artificially low TTL values. It would be reasonable for the IP-A specifications to require that the TTL value be set without recompiling the kernel.

IP Header Checksum

Some IP kernels have omitted the IP header checksum. This would not be acceptable for IP-A.

Host Requirements RFC

RFC 1122 is the current Host Requirements RFC which discusses IP and ICMP. It would be reasonable for IP-A to follow the recommendations in RFC 1122 except where explicitly stated above. The following additional notes may be helpful:

IGMP is probably not useful for IP-A. Neither is IP multicasting.

EMTU_R ("Effective MTU to receive") defines the size of the reassembly buffer. This is required to be at least 576 octets by RFC 1122. A larger value may be required for IP-A to efficiently support the application layer protocol. This should be studied and explicitly stated.

ICMP Issues

Per the Host Requirements RFCs, ICMP should be required of all NTCIP stations which implement IP-A.

[Ehud, Dan: are there any exceptions or changes to ICMP which we want to relax? Are there any semantics differences? I don't think so, and in the absence of comment from you, would simply end this section here.]

Routing Issues

IP routing is currently in a state of flux and confusion. In the absence of a clear requirement for a dynamic routing[7] protocol or router discovery protocol, we recommend that static routing using default gateways and static routes be specified in NTCIP.

However, it should be noted in NTCIP that any routing, static or otherwise, must include full support for classless addresses (*i.e.*, both supernets and subnets of existing class-full addresses).

If fault tolerant operation using dynamic routing is required, this recommendation should be revisited.

[7]That's roo-ting, by the way.

Application requirements which drive transport

The expressed direction for application level messages in NTCIP is an SNMP-like system. SNMP uses a simple request/response protocol with operations like "get" and "set." Many, if not most, NTCIP transactions will be completed in two application-level messages: a request of some sort and a response to that request.

In the discussions which follow, we concentrate on the issues involved in changing class A communications to support this application requirement, while leaving class B communications the same. Thus, the choice of class A or class B communications in a particular transaction will depend on the hardware capabilities and protocols configured into the network; many application-level transactions could execute equally well over class A **or** class B communications services.

The overhead associated with bringing up a TCP connection for a two-message exchange is quite large, on the order of 300% overhead in packet count. In addition, TCP is a relatively complex and powerful protocol which takes a substantial amount of software to properly implement.

Alternatively, UDP offers no delivery guarantee and little opportunity to conveniently associate longer multi-message transactions. Although the overhead of UDP is low, applications would have to operate on a timeout-implies-negative-acknowledgement basis.

NTCIP's application layer requires a protocol half-way between TCP and UDP: one which offers connection reliability and association greater than UDP's capabilities without all of the overhead of TCP.[8]

In the Internet model, this could be accomplished in a number of different ways:

1) Create a new protocol at the same level as TCP and UDP which operates directly over IP. This would provide the greatest efficiency by optimizing traffic directly over IP. However, it would also cause the greatest long-term compatibility problems with commercial IP implementations, many of which are not capable of adding in a third protocol over IP. This problem is particularly acute in the MS-DOS/Windows and Unix networking worlds.

[8]Remember that we are only considering class A communications in this case; class B is assumed to be unchanged.

2) Create a protocol which runs at a layer higher than TCP and UDP, presumably over UDP itself, which provides the common services needed by NTCIP application messages. This is less efficient than option (1), but offers the greatest opportunity to interoperate with existing commercial implementations of the Internet protocol stack.

3) Modify the application layer messages to include whatever services that particular message requires of either raw IP or UDP. This alternative supports the popular and oft-quoted "end-to-end argument." However, it also requires application message designers to also become transport protocol designers. This alternative also violates good principles of network layering by placing transport-layer functions into the application layer.

Deciding between these varied options requires a careful analysis of the requirements of the application layer messages. Options (1) and (2) require a small protocol to be defined for operation over either IP or UDP. Option (3) requires a set of common definitions and operations to be defined for inclusion in application-layer messages.

Conclusions

The architecture of the NTCIP draft protocol deserves reconsideration after our experience in designing an implementation of the protocols. Issues such as coexistence of class A and class B communications, operating system support, and routing suggest that there continues to be room for improvements which will shorten design and implementation time and reduce costs without sacrificing interoperability.

A version of the NTCIP standard which includes Class A communications based on the Internet protocol suite, a new transport protocol over UDP, and supports the other recommendations in this document would improve the architecture of NTCIP systems by making them off-the-shelf interoperable with existing Internet-architecture products and systems.

Changing the current NTCIP draft protocol to support IP would be a relatively simple matter. The topics listed in this paper are of a relatively narrow scope which could be acted upon in a short period of time. The transport layer protocol issue will take the longest to resolve since it will involve design of a transport protocol to replace any necessary functionality lost by changing from

X.25 to IP. Nevertheless, a simple transport protocol which provides the needed functionality will not be either complex or lengthy to design.[9]

From the point of view of the Monza team, we feel that the total time to complete a prototype will be slightly longer if this approach is taken, solely because the existing design will have to be revisited. However, the simplifications introduced into the architecture will partially compensate for added time by reducing implementation time.

The total time to functioning non-Monza implementations may be shorter due to increased availability of engineers familiar with TCP/IP.

[9]This protocol would also not be an existing standard, which raises questions of long-term interoperability along the same lines as the question of long-term interoperability with X.25.