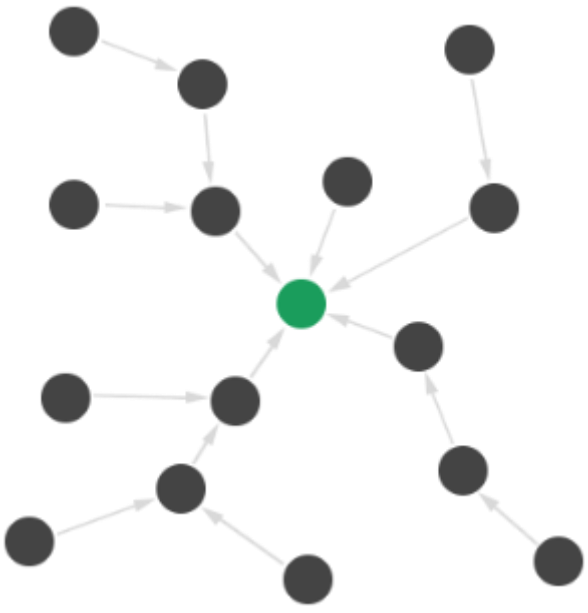
bacnet experts

PolarSoft® Inc.
914 South Aiken Ave
Pittsburgh PA 15232-2212
412-683-2018 voice
info@polarsoft.com
www.polarsoft.com

Discovery and Routing in BACnet

David Fisher

27-Aug-2015



TUTORIAL

Contents

Discovery and Routing in BACnet

27-Aug-2015

David Fisher

Introduction

One of the fundamental problems in BACnet communication is the procedure that devices use to find each other. Before device A can talk to device B, the A device must somehow know the *address* where device B is configured on the BACnet internetwork. In a simple network consisting of only one *network segment* using only one LAN type the address of B might be as simple as a single byte (or *octet* to use the BACnet terminology). Because this address is specific to the type of LAN technology and *media access control (MAC)* discipline, it is commonly called the *MAC address*. In a more complex internetwork consisting of multiple segments interconnected by BACnet routers, each network segment is numbered using a 16 bit unsigned number, each of which must be unique within the entire internetwork. When A and B are on separate network segments, their addresses must consist of the *network number* as well as the MAC address as a pair (network number, MAC address). Since each device is required to be configured with a unique *device instance number* that is unique within that BACnet internetwork, it is common to associate the device instance with a specific (network,MAC) pair. This association is called a *device binding.*

The methods used for device discovery and binding, as well as the rules for routing messages between devices and across multiple BACnet routers are the subject of this paper.

MAC Addresses

Each LAN technology has its own conventions for how individual devices on a segment are identified. There are several key characteristics of these so-called MAC addresses:

- Length of the MAC address (in octets)
- Range of allowed addresses
- Convention for broadcast addresses
- Composition of the MAC address

BACnet messages that are sent directly from one device to another are called *unicast* messages, meaning that they have one destination. However there are some BACnet functions that require sending messages to all devices. These are called *broadcast messages*.

Table 1 shows the most common BACnet LAN types:

LAN Type	*Length*	*Range*	*Broadcast*
BACnet/IP	6	complex	subnet broadcast:UDP
Ethernet 8802-3	6	full	FF:FF:FF:FF:FF:FF
MS/TP	1	0-254	FF
ARCNET	1	1-255	00

Table 1 - Common BACnet LAN Types MAC address Conventions

For BACnet/IP the 6 octet MAC address is composed of a 4 octet *IP address* and a 2 octet *UDP port number* in big-endian form. As a rule, all of the BACnet/IP devices within a given IP infrastructure will be configured to use a single UDP port number, usually 0xBAC0 (47808). In this context, the UDP port number can be thought of as its own LAN segment.

Strictly speaking, IP does not allow broadcast messages except to devices on the same *IP subnet*. To facilitate broadcast communications that reach across multiple subnets, BACnet defines special functionality called *BACnet Broadcast Management Devices (BBMDs)*.

For BACnet over Ethernet 8802-3, MAC address are 6 octets in length. The special address FF:FF:FF:FF:FF:FF is reserved for use as a broadcast MAC address.

For MS/TP LANs, each segment can support a maximum of 255 physical devices. The MAC addresses 0 through 254 are used for this purpose. MAC address 255 (0xFF) is used as a broadcast address. MS/TP further divides that range, allocating 0 through 127 for *master devices*, while *slave devices* may use any address from 0 to 254. Slave devices may not initiate communication; they can only reply to directly addressed (unicast) requests.

For ARCNET, each segment can support up to 255 physical devices whose MAC addresses are 1 to 255. MAC 00 is reserved for broadcasts.

Entities Within the BACnet Device

Although it is not required by BACnet, it is useful to think about the internal organization of responsibilities within the device as shown in Figure 1.

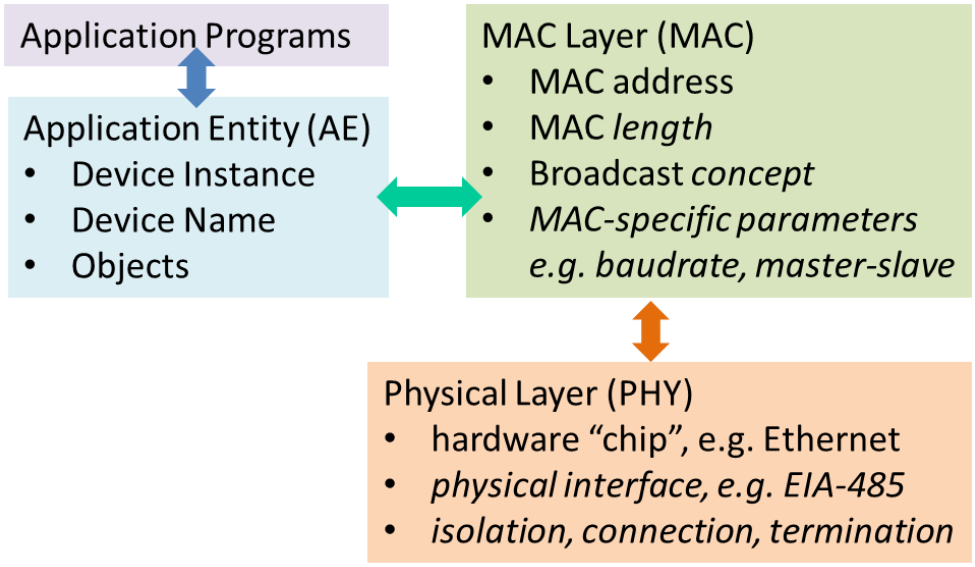


Figure 1 - Conceptual Entities in a BACnet Device

The Physical Layer (PHY) is responsible for the physical electrical connections to a network, including signaling, isolation, connection and termination issues. It also makes the "translation" between binary 0s and 1s into their equivalent signals on the wire or physical medium. This may require a dedicated hardware "chip" or some portion of a typical microcomputer such as a Universal Asynchronous Receiver-Transmitter (UART).

The MAC Layer is responsible for the assembly and disassembly of *message frames* that are physically signaled by the PHY. The MAC address, and its associated MAC address length, are specific to the MAC layer technology. The MAC layer also implements the appropriate broadcast message concept for that MAC technology. There are specific parameters associated with each MAC layer, such as the MS/TP baudrate and whether the MAC is a master or slave. The MAC

layer is fairly stupid in the sense that it only knows how to send a string of octets to a given destination on the same LAN segment.

The "BACnet-ness" of a device is located within its Application Entity (AE). Each AE has a *device instance number* and a *device name*. Both of these parameters must be configurable within the AE so that they can be unique in any BACnet internetwork that the AE may be installed on. The AE typically also manages the collection of BACnet-visible *objects* that represent information about the device and it's data points, or writable parameters. Keep in mind that the AE may be just a *server entity* that other BACnet devices communicate with, and make requests of. Or the AE may also have a *client entity* that makes requests of other BACnet devices.

BACnet devices of course have other responsibilities besides BACnet! They may be monitoring values, making measurements, performing control activities etc. All of these non-BACnet activities are called the *application layer (AL).*

Since all of these functions are performed inside of the BACnet device, it's really not up to BACnet to dictate how they occur or how they are implemented. But it is useful to keep these entities in mind when talking about BACnet operations, because BACnet does make some assumptions about how internal entities interact with each other.

BACnet Device Binding

Before a BACnet client can talk to a BACnet server, the client has to know where the server device is located. In this context, "located" means what is the (network, MAC) pair to use to talk to the server device? The set of devices (servers) that a client can talk to are called *peer devices*, or just *peers*. We use the term "peers" because there is no implied or enforced hierarchy among BACnet devices. Any BACnet device is allowed to talk to any other BACnet device if it has a need to do so. Although it's possible to organize all potential peers into (network,MAC) pairs, its more common for BACnet clients to associate a given peer device with its device instance. From the client's perspective, it wants to talk to a set of peer device instance numbers. This is generally the best practice because it isolates the client from changes in MAC address or network numbering. The association between a device instance and a (network,MAC) pair is called a *binding* because it binds together the instance with its physical location.

There are two ways that a client can determine its device binding(s). One way is to require that a human physically configure some table of bindings within the device. Since these kinds of bindings are not expected to change, they are called *static device bindings*. The main problem with static bindings is that they must be maintained by a manual process. If a device with some logical function must be renumbered, for example its network number must be changed or its MAC address must be changed, then all of the clients that might make use of that binding must also be changed. Even in a modest size system, this can quickly become a large maintenance issue over time.

The second, and most commonly used, method is called *dynamic device binding (DDB)*. With this doctrine, clients principly use device instance numbers to identify devices. When that device requires communication, the client uses a procedure to *discover* the binding automatically. This discovery is usually managed directly by the AE. The rough flowchart is shown in Figure 2.

1. AE wants to locate device instance X

2. AE sends a Who-Is(X,X) message, typically as a *global broadcast*

3. AE waits for some number of seconds

4. While waiting, if AE receives an I-Am(X…) message then one of the parameters will be the device's MAC address. If the device is on a *different BACnet network number* then that network number will also be included in the I-Am.

5. If the time elapses and no I-Am has been received, then the AE usually tells the client that the device X is not reachable. It is up to local policy in the Application whether to continue waiting and try again, or to give up. If the Application asks the AE again, the AE may repeat the procedure. Some AEs limit the frequency of retries to avoid flooding the network with Who-Is requests.

Figure 2 - Procedure for Dynamic Device Binding

Most AE implementations will cache the information received in such I-Ams into a *peer device table* so that future requests by client application(s) do not require rediscovery. In such cases, it's a good practice to actively manage the peer table. If a "discovered" device fails to respond to some request, such as a ReadProperty, after a few retries, then the peer table should be marked to cause a Who-Is to be transmitted again for that device. This assures that if devices are renumbered in their MAC address or network number, then the clients will automatically reaquire them.

The BACnet Network Layer

So far I've avoided any discussion about how messages get from place to place. I've also used terms like *global broadcast* without discussing what "global" vs. "local" really means. We've also looked at (network,MAC) pairs without talking about when and why multiple networks are needed. All of these issues are the responsibility of the BACnet network layer (NL). The NL is complicated and Clause 6 of the BACnet standard has some ambiguities and use cases that are not clearly described. Let's look at the NL piece by piece to unravel some of its complexity.

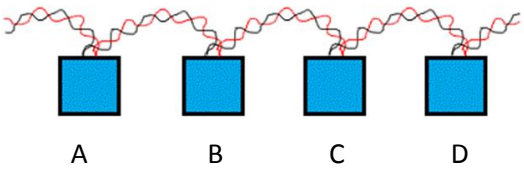
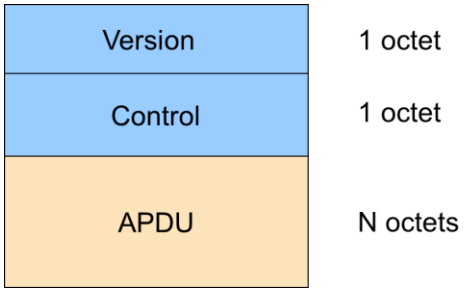


Figure 3 - Typical MS/TP daisy-chain

Figure 3 shows a typical MS/TP daisy-chain network segment. For this example, suppose that device "A" (the client) needs to talk to device "D" (the server). Since both devices are on the same network segment, there is no BACnet "routing" required. Really the only thing that A needs to know is what D's MAC address is.

Unlike some network layer protocols like IP which has a fixed length network layer header, the BACnet NL is designed to stretch and become bigger and smaller as needed. In this example the NL is only 2 octets.

Ther Version is always 01. The Control octet is also known as the network protocol control information (NPCI) octet.



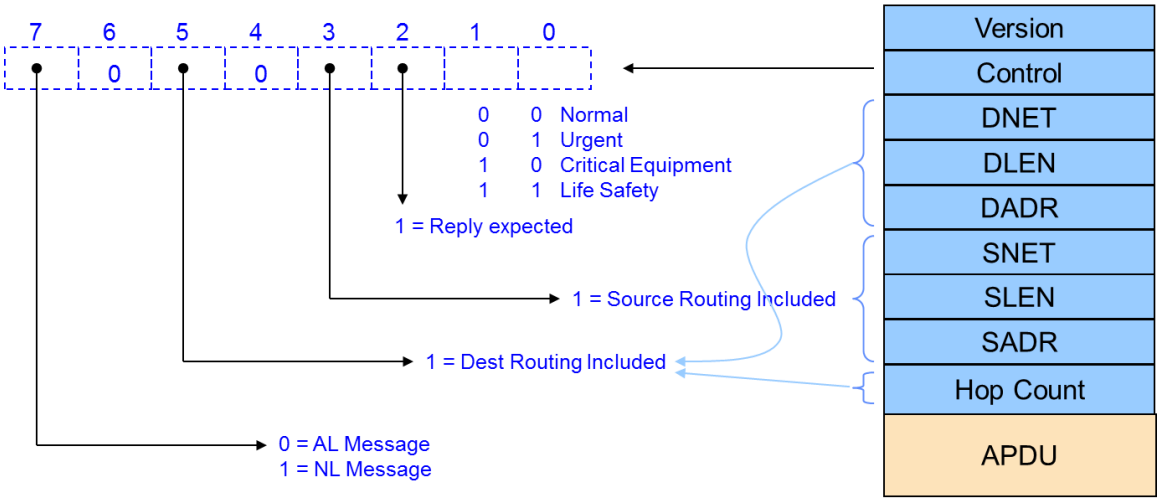


Figure 4 - Structure of the NPCI octet

The NPCI has five functions:

- Bit 7 indicates whether the message is a *network layer message* or an *application layer message*. NL messages convey information between BACnet routers. Most messages are AL messages.
- Bit 5 indicates whether the NPCI is followed by *destination routing information* or not. Specifically whether the DNET, DLEN, DADR and Hop Count are present.
- Bit 3 indicates whether the NPCI, and possibly the DNET, DLEN and DADR, are followed by *source routing information* or not.
- Bit 2 is only important when the destination is an MS/TP segment and indicates whether the router-to-MS/TP should wait for a reply after routing the message onto the MS/TP segment.
- Bits 1-0 are a *network layer priority* that determines the importance of this message within any router that routes the message. The significance of this is, for example, in a router that has an internal queue of messages waiting to be routed onto a segment, a new incoming message with a network priority of 3 (life safety) should be routed first before any queued message with a lesser priority.

In the simple use case of Figure 3 the NL is collapsed to 2 octets. As a result, device D easily knows to respond to the source MAC address, also with no routing required.

In practice most BACnet networks are larger than a single segment. When you have two or more BACnet network segments that have devices which need to talk to each other, then the segments can be joined together by a special kind of device called a *BACnet Router*. A router connects two or more (usually different) LAN types together and automatically routes messages from one LAN to another based on the information in the NL portion of each message. Each BACnet LAN connection to a router is called a *port.* Typically ports are synonymous with physical ports, but may also represent *logical ports*. For example, a router between Ethernet 8802-3 and BACnet/IP may have a single physical Ethernet port connector, but the (very different) 8802-3 and BACnet/IP messages are routed as if the ports were physically separate.

To illustrate this, Figure 5 shows a simple BACnet network with two network segments that we've numbered 1 and 2. They are joined by a router R. For this example, let's say that network 1 is Ethernet 8802-3 and network 2 is MS/TP.

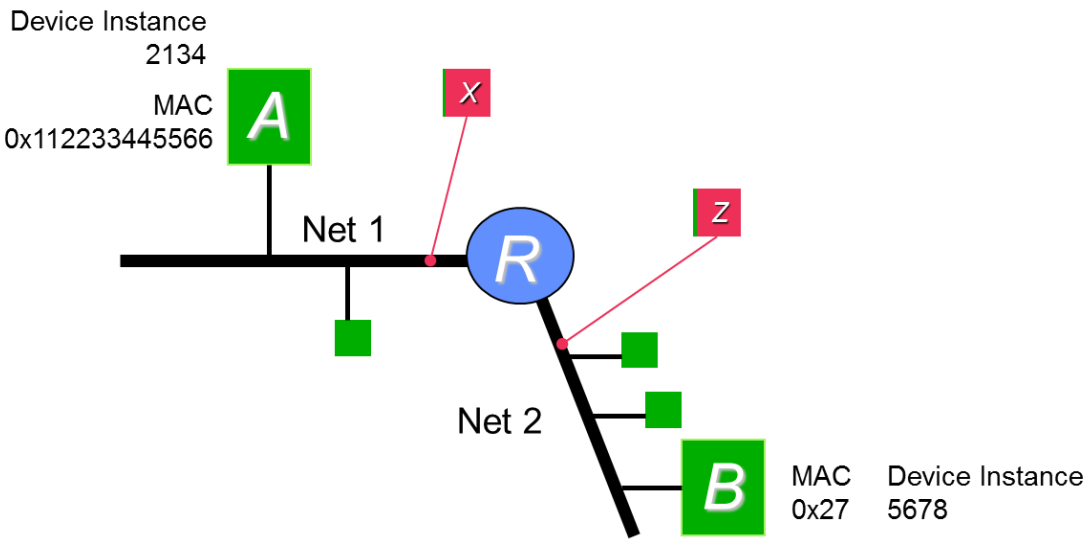


Figure 5 - Simple Two Segment BACnet Network

As a complete example, assume that device A wants to discover B and then send a ReadProperty message to B. The only thing that A knows is that B is device instance 5678.

The first step is that A needs to have a binding for device 5678. So the AE in device A needs to send a Who-Is message with 5678 as the upper and lower bound device instance. Because A knows that there may be more than one segment in the network, it does not want to transmit a broadcast only on segment 1. Instead it wants the broadcast Who-Is to reach all BACnet segments. This is called a *global broadcast*. Global in the sense of the entire BACnet internetwork, not "the whole planet." Because this message requires *destination routing information* the NPCI will need to set its bit 5 (see Figure 4). The DNET will be 0xFFFF which means the global network. DLEN will be 0 which means "broadcast on the final network". There is no DADR because DLEN is zero. Even though "Who-Is" is a kind of request, we do not set the bit 2 reply-expected bit in the NPCI because Who-Is is an *unconfirmed request*. So the outgoing message will include this NL:

Version: 1
Control: 0x20
DNET: 0xFFFF
DLEN: 0
Hop Count: 255
Who-Is message

Note: 2 octets

The NPCI bit 5 is set to indicate destination routing information is present. The DNET is global broadcast network. The DLEN means final broadcast. The Hop Count is initialized to 255. Each time the message passes through a router, the Hop Count is decremented by one.

Figure 6 - Single Hop Network Layer at point X

Of course Figure 6 only shows the NL portion of the message. Keep in mind that this is contained in the payload of the MAC layer. The receiving device will know from the MAC layer who the sending MAC address is. Because this message doesn't have an explicit destination, it is broadcast on the local MAC segment.

Because router R is also on the local segment of A, it will receive the broadcast. R looks at the NL header, specifically the NPCI and realizes that there is destination information attached. Because the destination network is "global broadcast" router R will take this message, but with the Hop Count decremented by 1, and retransmit a copy of the message on all of its router ports except for the port it was received on. However, because the message is "leaving network 1" R must add the original sender's address info to the message. The original sender's network is 1 so SNET is 1. The MAC address has length 6 so SLEN is 6. SADR is the original MAC address 0x112233445566. So at point Z in Figure 5 the message looks this:

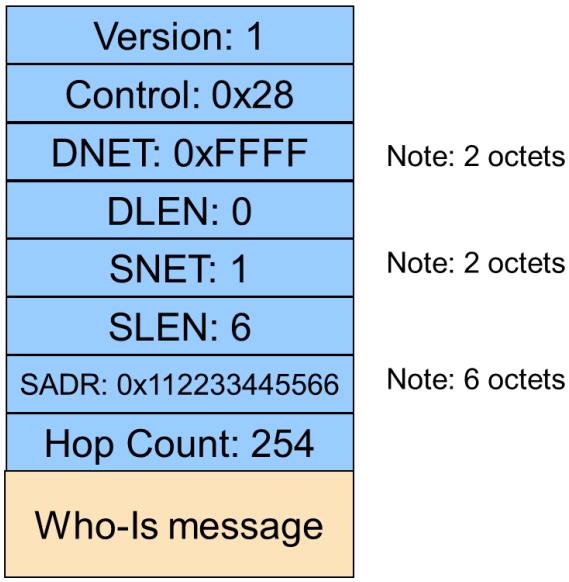


Figure 7 - Single Hop Network Layer at point Z

Device B's AE receives this locally broadcast message. Its NL looks at the NPCI, sees that it includes destination routing information, sees that it is the global network and that it is a broadcast (DLEN=0). Device B's NL passes this on to the AL for processing. The AL recognizes the Who-Is message and compares the provided range of device instances (5678,5678) with the AE's device instance and realizes that device B is in this range. The AE is now obliged to reply "some time later" with an I-Am message sent back to the original sender (device A).

For most of the BACnet LAN types, device B can respond immediately. However, because network 2 is MS/TP, device B is not allowed to reply until a later time when B has the token. This means that B must retain the sender's address information at least in order to build a reply later. This would include:

- The forwarding router R's MAC address on the MS/TP segment
- The SNET, SLEN and SADR of the sending device

Alternatively, device B could build the complete I-AM message and queue it for later transmission.

Eventually device B gets the token and realizes that it needs to send an I-Am message. BACnet dictates that the I-Am should be sent directly to the original sender and should NOT be broadcast globally. Conveniently, the Who-Is already has what we need in it to reply to the sender. The router R's MAC address is the source MAC address which can be turned around and used as the destination MAC address. The NL contains the SNET, SLEN and SADR which can be reused as the DNET, DLEN and DADR. The outgoing message now has an NL like this:

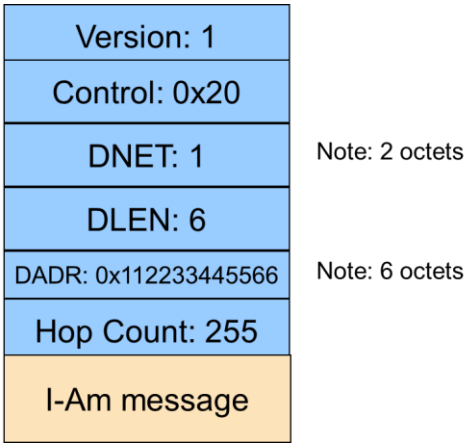


Figure 8 - I-Am Reply to Remote Device

Since B's MAC provided the source MAC address of router R, the NL knows exactly which MAC to send this to. In order to assure that the original sender receives the message we include destination routing information in the outgoing NL.

R receives this message and realizes that the NPCI has destination routing information. It looks at the DNET (1) and looks through its list of (port,network) assignments and sees that network 1 is one of its connected networks. Since this message is about to be routed onto the final network, the destination info can be removed. The DADR is then used as the MAC address for the final retransmission onto network 1. Since the message is leaving network 2, R needs to add source routing information to remember the network number it came from (2) and the MAC address it came from (0x27). So the message R transmits onto network 1 has an NL like this:

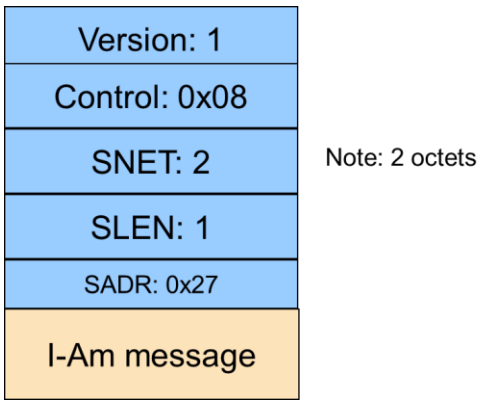


Figure 9 - I-Am Reply on final network

When A receives this message it now has everything it needs to know about B: it's network number (2), it's MAC address (0x27) and the local router address of R.

A can cache this information in its peer table for later reuse.

Now that A knows where B is located, it's a simple matter to send regular questions to B, such as a ReadProperty request. This would look like:

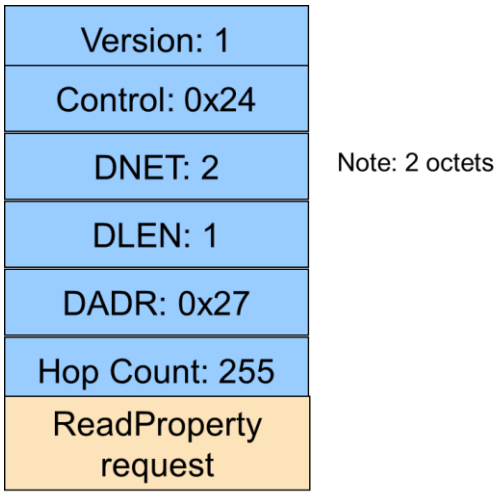


Figure 10 - ReadProperty NL to a Remote Device