ECHELON

The LonWorks™ Company

LonWorks™ Connections

Bernd Gauweiler,
November 1999

Abstract

LonWorks networks benefit from an almost intuitive means for devices to communicate with each other, the network variable connections[1]. Whilst using and taking advantage of such connections is fairly straight-forward, the process of creating such connections can be a tricky business if optimum results are desired.

This document summarises the concepts and techniques provided by the LonTalk® protocol, discusses the impact and side-effects to the network integrator and the device manufacturer using today's tools and techniques, and finally discusses a few example connection scenarios.

LonTalk Protocol Basics

The LonTalk protocol offers a comprehensive range of service types, addressing modes, and other transport properties for communication between two or more points. The following summarises those properties that are relevant to the subject of this document. A discussion follows in the next section, allowing for the experienced reader to skip over this summary.

Service Types

The following service types are supported by the LonTalk protocol:

- Acknowledged (short: ACKD)
- Unacknowledged but repeated (short: UCKD/RPT)
- Unacknowledged and not repeated (short: UCKD)
- Request/Response (short: REQ/RESP)

Sending a message using acknowledged service causes all receivers of said message to reply with an acknowledgement (also known as the "ACK"). The propagation of an ACK confirms that the incoming message has been received free of errors, and that it entered the application input queue. The sender waits for all expected ACK messages to come back within a time period described by the transaction timer (aka "Tx Timer"), and enters a configurable number of retries upon failure of completion. The sender has a choice of either retrying the complete transaction or issuing a reminder to individual devices which failed to ACK in time.

The LonTalk programming model supports means to programmatically determine success or failure of completion of such an acknowledged transaction for most cases.

---

[1] Although most of this document also is applicable to implicitly addressed (i.e. bound) explicit message tags, this document only refers to network variable connections.

Unacknowledged services do not include such an acknowledgement. The sending device will repeat the outgoing message for a configurable number of repeats, using a configurable time between these attempts (Repeat Timer). A special form of UCKD/RPT service is the unacknowledged but not repeated service UCKD, which uses the same mechanisms but a repeat count of zero.

Network variable updates sent to one or more destinations typically use acknowledged or unacknowledged-repeated services.

For UCKD/RPT or UCKD services, transaction completion control is limited to local completion. Success is reported as soon as the original packet and all repeats went out on the network, but does not include information about the receipt of such message on the destination nodes.

Last not least, the request/response communication mechanism is very similar to acknowledged service with two key differences. Firstly, the acknowledgement is now called a response and carries data. Secondly, the transaction is initiated in the opposite direction: an ACKD transaction is initiated by the data source in order to provide up-to-date data for the data consumer, whereas an REQ/RESP transaction is initiated by the data consumer in order to obtain up-to-date data from the data source.

Addressing Modes

The LonTalk protocol supports the following means to identify the destination address of a given packet:

- NEURON ID addressing
- Subnet/Node ID addressing
- Domain broadcast addressing
- Subnet broadcast addressing
- Group addressing

NEURON ID addressing is a unicast addressing mode which uses the physical address of a given LonWorks device to specify the desired destination. This addressing mode should only be used by the network management tool in case of a communications problem and during the initialisation phase. A message using NEURON ID addressing can be seen on the entire network, and it can also be sent out with destination subnet information, allowing infrastructure devices to route the packet accordingly.

Subnet/Node ID addressing, the second unicast service in the LonTalk protocol, uses a device's logical address to denote the destination. Such a logical address is assigned to the device by the network management tool and must be unique in the entire network. Subnet/Node addressing allows for traffic segmentation by LonWorks routers, so that such a packet only becomes exposed to the relevant parts of the network.

Very much like most NEURON ID addressed messages, domain broadcast messages can be seen on the entire network. In contrast to NEURON ID addressed messages, however, domain broadcast messages will be received and interpreted by all LonWorks nodes in this domain.

Subnet broadcast messages are more economic than domain broadcasts, but this addressing mode requires all addressees to belong to a common subnet. Configured or

learning routers can isolate such a packet from the rest of the network, and prevent it from being exposed to the entire system.

Group addressing is the most complex type of multicast addressing provided by the LonTalk protocol. In principle, any number of LonWorks devices in any location on the network can form a group of devices, and LonWorks routers will route or filter group addressed messages accordingly. A few key concerns should be noted, though: firstly, it should be noted that LonWorks networks are limited to 256 groups per domain. Secondly, it should be noted that the group membership information is kept in the device's address table[2], along with the other addressing records. Last not least, it should be noted that the LonTalk protocol restricts the size of groups using ACKD services to 64 members including the sender device.

Possible Combinations

The following table illustrates the possible combinations of the above service types and addressing modes, as far as the LonTalk protocol is concerned. It should be noted that only a subset of these combinations is available in real implementations. This will be discussed later in this document.

	ACKD	UCKD/RPT	UCKD	REQ/RESP
NEURON ID	✓	✓	✓	✓
Subnet/Node Id	✓	✓	✓	✓
Group	✓ (Note A)	✓	✓	✓ (Note C)
Subnet Broadcast	✓ (Note B)	✓	✓	✓ (Note B)
Domain Broadcast	✓ (Note B)	✓	✓	✓ (Note B)

Table 1: Addressing modes versus service types

Note A: Limited to a total of 64 members

Note B: Application's transaction completion control will only track the first response being received

Note C: For a group with a known size, this is limited to 64 members. For an open group (i.e. a group of unknown size), see note B above.

Discussion

Although the LonTalk protocol as such allows for the service types, addressing modes, and combinations of these to be used as indicated above, these techniques and their impact shall be discussed in this section.

Practicable Combinations

[2] A separate document in the same series, "The Address Table", provides more information on the subject.

Although the LonTalk protocol implements an almost orthogonal set of transport mechanisms, a few considerations and side effects limit those combinations to the ones shown in the table below.

Table 2 lists the practicable combinations of service types and addressing modes in a LonWorks network. It should be noted that this is in line with the feature set of the binding algorithm available as part of LNS for Windows:

	ACKD	UCKD/RPT	UCKD	REQ/RESP
NEURON ID	✓	✓	✓	✓
Subnet/Node Id	✓	✓	✓	✓
Group	✓ (Note A)	✓	✓	✓ (Note C)
Subnet Broadcast		✓	✓	
Domain Broadcast			✓	

Table 2: Practicable combinations of addressing modes and service types

The LNS binder does not allow for all possible combinations for the following reasons:

- Acknowledged service causes misleading results in the transaction completion control algorithm when being combined with broadcast addressing. The application will be notified upon successful completion as soon as the first reply (ACK) has been received by the originator
- Request/Response service causes misleading results in a transaction completion control algorithm, too. The first response being received on the originator will cause the transaction completion control algorithm to report success, and all other responses will be ignored.
- The use of unacknowledged/repeated service is undesirable in combination with domain broadcast addressing: such an attempt would easily cause a lot of network-wide traffic, causing congestion in the router devices and in the devices' receive transaction database. This would probably cause the network to stop functioning for a noticeable period of time for each unacknowledged/repeated domain broadcast, and is therefore neither recommended nor supported by the LNS binding algorithm.

What Does ACK Acknowledge?

Using acknowledged service seems to be an attractive option, because the LonTalk protocol automatically takes care of transaction completion control and also provides mechanisms to notify the application layer on success or failure[3]. The obvious question, however, is what precisely the receipt of an ACK acknowledges.

[3] See the NEURON C Reference Guide, functions nv_update_fails(), nv_update_succeeds(), and nv_update_completes(), for details on completion control and notifications.

In case of the receiving node being a regular LonWorks device running an application local to the NEURON chip, the ACK will be returned to the originator by the time an incoming network variable update message has entered the application input buffers[4].

For the initiator of the original update, receipt of an ACK notifies of the correct receipt of the original message on the destination device, including the availability of space in one of the input queues.

Receipt of an ACK, however, does not confirm that the destination device's application was able to process this network variable update and to respond accordingly. Apart from the application program being poorly written, there are a couple of valid reasons for this to happen, even on a well-designed and well-behaved system. For example, the application could simply be busy processing other network or I/O related events.

Using an analogy, the acknowledged service type can be compared with a registered letter service with H. M. Royal Mail (or U.S. Mail, for that matter). Upon receipt of such a letter, the recipient signs the receipt. The mail service returns this signed note to the sender of the letter. Receiving this confirmation acknowledges the correct delivery and receipt of the original message, but it does not include confirmation that the receiver actually opened, read and understood the letter or successfully took the appropriate action as a result of the message being received.

In case such comprehensive confirmation is required by the originator, confirming that the appropriate action has been performed or initiated both correctly and timely as a result of the original network variable update, a more complex mechanism is required between the data source and the data destinations. An application layer acknowledgement should be considered in such a case, very much like what the LonMark Interoperability Association introduces in the basic object types "Close Loop Sensor" and "Closed Loop Actuator". This allows for true confirmation and true transaction control. On the other hand, such a control mechanism is not easy to implement in case of multicasts, since the originator needs to know about the number of intended recipients.

The Pros and Cons of ACKD Service

As discussed above, acknowledged services provide an easy-to-use and *almost* reliable mechanism for transaction completion control and management. In case truly reliable and complete transaction management and control is desired, an application layer response is the more appropriate approach, though.

Here are a few more disadvantages of acknowledged service:

- Lock-out upon failure

  This effect becomes effective in case a transaction fails to complete straight away, i.e. the ACK is not being received by the originator. In this case, the originator waits for a maximum of T = number of retries * TxTimer interval until the transaction completes with a failure. The NEURON chip, which is used on most LonWorks

[4] Hosted devices using MIP or NSI firmware are irregular in that sense. For a hosted device, an ACK will be returned as soon as the incoming packet left the NEURON chip upwards towards the application, i.e. as soon as the incoming update entered the host-side driver. Subject to the implementation details, a more or less complex administration and routing layer follows on the host before the packet actually reaches the application.

devices, only supports one concurrent outgoing transaction per queue[5]. Thus, the NEURON chip will not be able to propagate any other packets until the transaction completes successfully or T expires.

- Throughput limitations

  Due to the administrative overhead of ACKD service compared to UCKD/RPT, a single LonWorks device can propagate much more UCKD messages than ACKD ones. This is even true in case of good communications, because the sender device still needs to wait for the ACK to come back from the recipient. Tests have shown that a single device was able to propagate almost twice as many UCKD packets than ACKD ones when the maximum packet size of 240 bytes was used, and about four times as many with packet sizes used by network variables[6].

- Peak traffic demand

  In an acknowledged multicast, the potentially huge number of acknowledgements caused by an acknowledged network variable update causes a peak demand in network bandwidth. This has a couple of possibly unwanted effects:

  a) Routers on the way between the sender and the acknowledging nodes must be suitably configured and could cause loss of ACK messages due to a lack of buffers, thus causing the originator to send reminders. This will delay completion of such a transaction in an otherwise healthy and well-functioning network.

  b) The potentially huge number of ACK messages issued almost simultaneously necessarily creates a high momentary collision rate. This delays completion of such a transaction in an otherwise healthy and well-functioning network.

- Redundant configurations cause multiple duplicates

  Networks using redundant router configurations unavoidably produce duplicates of the original (acknowledged) message. It must be noted that each acknowledged message being received by a LonWorks device must be replied with an ACK, even if the original packet is been recognised as a duplicate. This is because the receiver can't tell whether the duplicate is caused by the network topology or a result of the originator re-trying. For a network which uses redundant router configurations, this will unavoidably result in a huge number of ACK duplicates, thus causing even more traffic and potential congestion.

On a positive note, here are some advantages of acknowledged service:

- Correct channel backlog estimation

  Using acknowledged services allows all nodes on the network to correctly compute the number of expected packets, hence to correctly adjust the size of the randomising window. This is important for the p-persistence of the LonTalk protocol's media access algorithm, and especially beneficial for multicast transactions. Although the broadcast addressing mode supports a channel backlog property, this number may not be known (in which case the backlog for this transaction defaults to 15). Or, the expected backlog might be in excess of 64 (in

---

[5] There is a separate queue for priority and non-priority messages, and the NEURON chip supports one concurrent outgoing transaction per queue.

[6] That is up to 31 byte data.

which case the randomising window would have reached its maximum size of 1008 β_1-slots), thus unavoidably causing an unwanted probability of collisions.

- Support for authentication

  Authentication can be applied to all forms of acknowledged transactions (i.e. ACKD packets as well as request/response transactions), but it can not be used with any type of unacknowledged service.

- Bandwidth consumption

  Assuming good communications, ACKD transactions for unicast connections only require two packets per transaction (The ACKD message itself and the ACK coming back). UCKD/RPT service would typically use 3 repeats, thus causing a total of 4 packets (the original message plus 3 repeats) per network variable update.

Acknowledged vs. Unacknowledged

In light of the above considerations, it is worthwhile spending some time on how unacknowledged services perform compared to acknowledged ones. A key aspect is the probability of delivery and completion.

As discussed in the LonTalk protocol specification, the probability of delivery of an unacknowledged message with three retries is in excess of 99.99%.

Figure 1 illustrates the probability of successful delivery for an UCKD/RPT message as a function of the number of repeats, where the parameter is the percentage of transmit errors. Even under poor communications conditions, suffering from 3.125 % collision rate and 10% transmit and propagation errors, the overall probability of delivery for at least one of the packets still is in excess of 99.99% with three or more repeats.

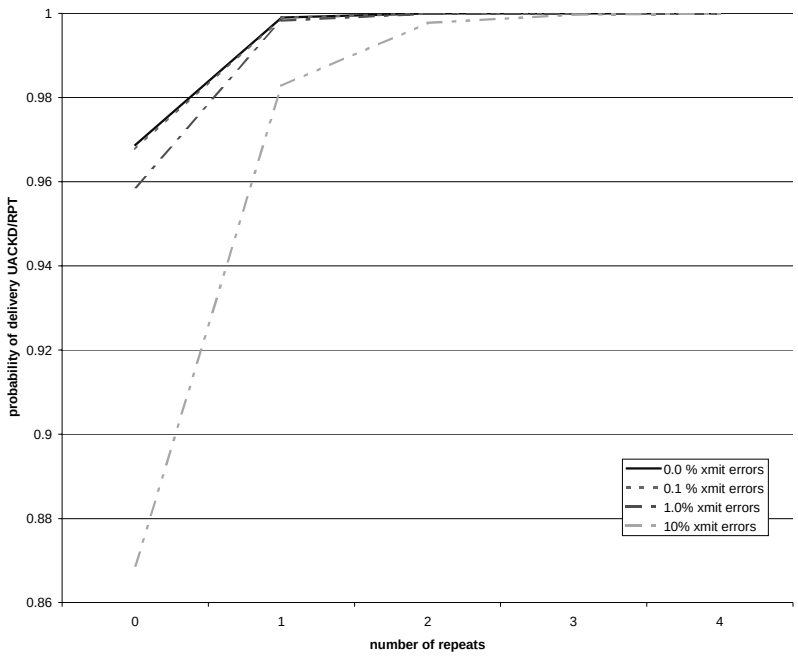

Figure 1: Probability of delivery of an UCKD/RPT message

Figure 2 shows the probability of completion of ACKD multicast messages under the same poor communications: 3.125% collision rate and 10% transmit and propagation errors, not taking any routers into account.

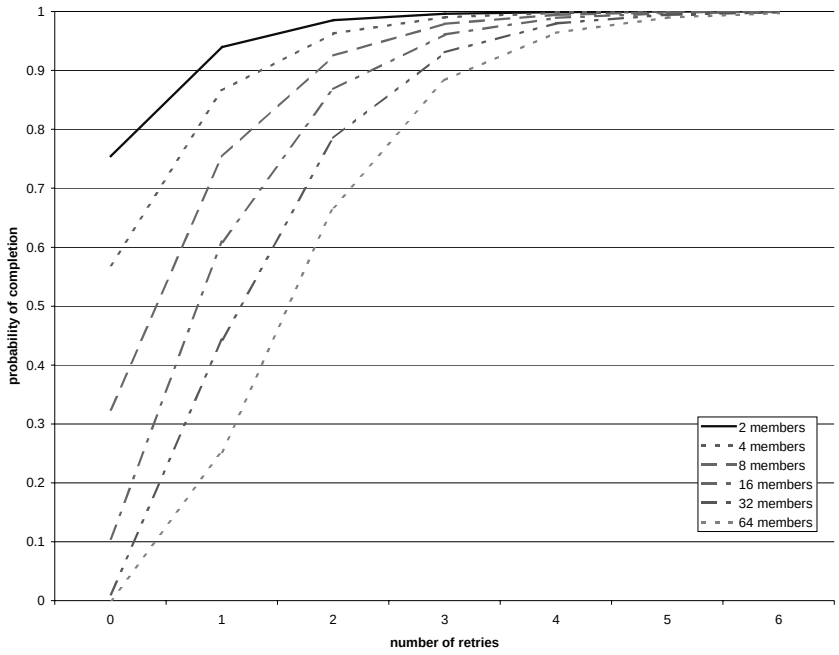

Figure 2: Probability of completion for an ACKD multicast under poor communications conditions

Compared to above figure 1, ACKD service does not perform just as good. Especially under poor communications conditions, large group multicasts with 3 retries do not achieve the same probability of completion than UCKD/RPT service does.

The overall result of all these considerations is hardly surprising. Compared to acknowledged services, UCKD/RPT services

- offer a higher probability of delivery
- have less impact on the sending device
- cause less congestion

Use of Groups

As long as acknowledged services shall be used, only two addressing modes are available: subnet/node ID addressed unicasts, and group-addressed multicasts for groups with up to 64 members[7]. Apart from being limited in size when acknowledged services shall be used, groups themselves are a very limited resource in a LonWorks network: a given LonWorks domain can only have up to 256 different groups. Although modern binding tools include algorithms to re-use and overload group identifiers and thus being more and more economical with this rare resource, it is still worth being aware of groups whilst creating connections.

A group addressing mode is required in the following cases:

[7] The NEURON ID addressed unicast is not listed here, since this should only be used by the network management tool during the installation phase.

1. Acknowledged multicast
2. Polling fan-in connections

The most frequent case where groups are used is acknowledged multicasts like the one shown in figure 3: all three LonWorks devices shown will become member of a group, so that a single network variable update can be received (and ACK'ed) on both receivers A and B.

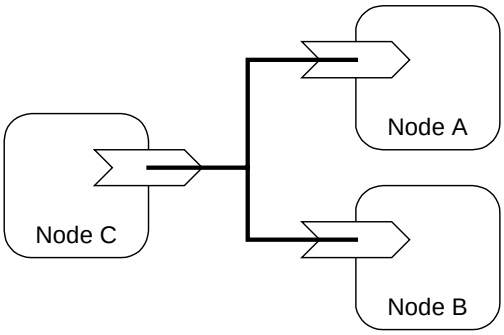

Figure 3: Multicast Connection

Polling fan-in connections, however, are a more hidden way to exhaust group identifiers in a LonWorks network. The principle design of such a configuration is shown in figure 4:

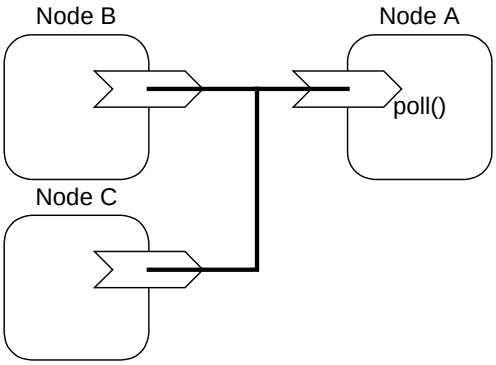

Figure 4: Polled fan-in connection

Without node A being polling, the connection shown in figure 4 would use two unicasts for devices A and B, one each. Due to the use of the poll() function in the node A application, the connection will use a group addressing mode (and a request/response service on this group every time the poll() function is being executed).

Connection Management

The following discusses measures and precautions to be considered by both, the device manufacturer and the network integrator, in order to achieve "good solutions". Such a solution is making most efficient but also most economic use of the LonTalk protocol and the resources in a LonWorks network. It also attempts to preserve the most limited resources as much as possible so that these are available when needed.

Device design time

During device development time, quite a few thoughts should be spent on how the connections are to be made once the product is finished. With regard to network variable connections, some key considerations are:

1. How does the device obtain input variable values after start-up?
2. How does the device implement transaction control?
3. Which amount of flexibility shall be allowed to the integrator?

For the problem of receiving input variable values after start-up, the following approaches are possible:

1.a) Polling remote data via local input network variables

1.b) Relying on incoming network variable updates and default values

Polling local input network variables, although being attractive on a first glance, is not the best choice in most cases. It has the following disadvantages:

- Calls to the poll() function consume address table space. The amount of address table space being consumed is a function of the actual connection scenario and can be anything from one address table entry for the entire device up to one address table entry per input network variable. Since the address table is limited to no more than 15 entries, this quickly causes most severe problems as soon as such a device shall be used in a real network.
- Calls to poll() might cause the LonWorks binder to allocate a group where otherwise no group might have been required (See figure 4 in above discussion for an example). Since group identifiers are limited to no more than 256 per domain, such a technique will typically consume a considerable amount of said identifiers.
- Calls to poll() also limit the number of remote output network variables being bound to this very local input network variable to 63. To ensure transaction control, the binder does not tolerate poll commands on open groups, subnets, or domains.
- Polling large groups necessarily cause a large number of responses. This might well require a very powerful application, able to process these responses fast enough to avoid packet loss due to a shortage of application input buffers.
- Multiple devices polling input network variables after power-up might cause excessive temporary network traffic.

It should be noted that the issue of peak bandwidth demand can be solved with adding a random delay between power-up and issuing the poll command, however, all other disadvantages can only be solved in refraining from using polling techniques in the first place.

Alternatively, all devices on the network might propagate the best possible known output values after power-up (still using a random delay to avoid congestion). As long as the required input data has not arrived, each device will use fail-safe output defaults. This allows for the network to start-up in the most economic fashion, and is also in line with the common technique of heartbeats.

The device manufacturer needs to consider local demands into transaction control, too. The basic options are

2.a) Trust statistics and do not perform explicit transaction completion control in the application code. Rely on the network and the LonTalk protocol to deliver the packets as desired

2.b) Not to trust the protocol and the network blindly, but to watch success or failure of transaction completion locally and to respond to an error condition accordingly

2.c) Not to accept any potential data loss unless there is a severe and unrecoverable malfunction

Option 2.a is the most commonly used. From an application developer point of view, this means to simply update an output network variable but not to care for nv_update_succeeds() or nv_update_fails() functions. Although being a "fire & forget" strategy, this works surprisingly well and is suitable for most tasks, especially when being combined with heartbeats. In particular, this approach leaves all decisions about service types and addressing modes to the integrator, and does not restrict the appliance of the device in question.

Option 2.b means to take advantage of nv_update_succeeds() and nv_update_fails() functions to obtain tight control over the transaction completion. Upon failure, the device could raise an alarm, enter an error mode, or otherwise respond accordingly. It should be noted that option 2.b would make most sense if acknowledged services are being used, and the relevant output network variables should therefore be declared using the optional network variable connection information bind_info(ackd (nonconfig)).

Last not least, option 2.c is the most powerful solution using application layer acknowledgements in the sense of a feedback network variable as introduced by the closed loop actuator and sensor objects in the LonMark standards[8]. Figure 5 shows the principle of such a feedback loop. Assuming the sensor device detects a catastrophic malfunction in the oil burner, the sensor will want to notify the actuator (an alarm buzzer and strobe light) reliably. It will therefore propagate the alerting network variable repeatedly until the buzzer device replies to the sensor's input network variable, confirming that the message has been understood and correct action has been taken.

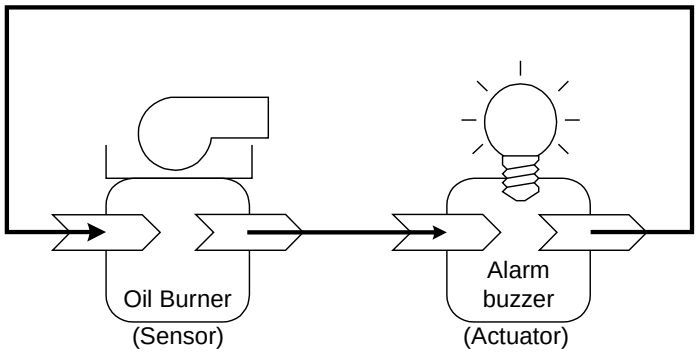

Figure 5: Feedback network variables

[8] See the LonMark Interoperability Association Application Layer Guidelines for details

Such a transaction completion control algorithm can easily be implemented in a NEURON C application if the loop only includes unicasts. The implementation is trickier for a sensor device in case multicasts are used, as shown in figure 6 below.

The oil burner would expect 2 feedback replies for each output network variable output. Note that application feedback loops are ideal candidates for UCKD/RPT services in case of unicast feedbacks, and ideal candidates for ACKD groups in case of multicast feedbacks. Acknowledged groups provide the number of feedback updates to be expected, unlike any other kind of multicast connection.

Needless to say that such an approach achieves optimum completion control on the expense of device and network resources.

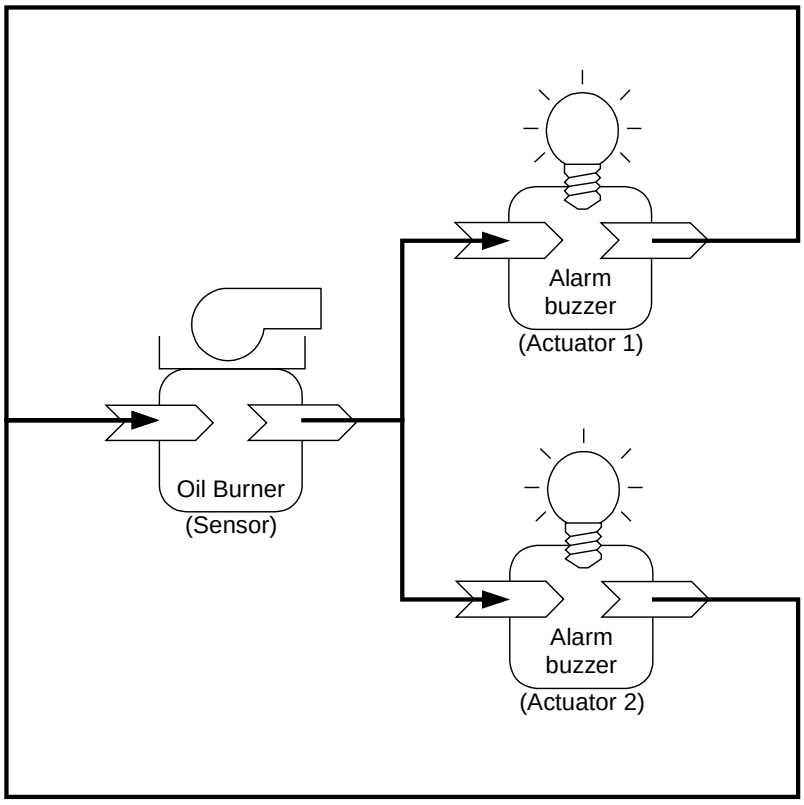

Figure 6: Feedback loops with multicasts

Another important consideration for the device developer is support for network variable aliases. Being trivial to implement for the NEURON C developer[9], networks variable aliases allow for much more connections and allow for connections which would otherwise be impossible[10]. The examples later in this document include cases where aliases are beneficial, and it is strongly recommended to include support for aliases in any newly developed LonWorks device.

[9] Add "#pragma num_alias_table_entries X" with $0 \le X \le 62$ at the expense of 4 byte EEPROM space per entry.

[10] Please refer to the document "LonWorks Aliases Made Easy!" for further information on aliases

Network design time

At network design time, the key problem with regards to connections is to achieve the most reliable solution with the most economic use of resources on the network and the devices. For the purpose of this document, network resources can be thought of as network variable selectors, group identifiers, channel bandwidth, etc, whereas address and alias table space are the critical device resources.

When a connection is being made, the LonWorks binder first ensures compliance with the LonWorks connection rules and binding constrains:

Rule 1: Standard network variables within one binding must have the same type

Rule 2: Network variables within one binding must have the same length

Rule 3: At least one network variable in a connection must be an input network variable and one of them must be an output network variable

The above binding rules are enforced by binding tools, in order to ensure proper function, to add extra security to the binding process and in order to avoid data loss or data misinterpretation.

There are, however, some binding limits that are enforced by the LonTalk™ protocol. Those limits are usually referred to as "binding constrains". Though there is no recommended way of bypassing the above three rules, there are some binding constrains which can be lifted by the use of alias network variables. These constrains are:

Rule 4: Network variables within one connection must share one and only one network variable selector, which must be unambiguous.

Rule 5: Multiple input network variables on a given node can not share a network variable selector. The same applies to output network variables on nodes earlier to firmware version 6[11], and to all polled output network variables.

Rule 6: Each network variable has one and only one network variable selector. In addition, each output network variable (or polling input) has one address table entry associated with it. Input network variables may receive updates using zero, one, or multiple address table entries.

Once the binding tool approved the compliance of the desired connection with said rules and constrains, it will attempt to implement the requested connection. The network management tool might provide an optional set of connection properties, also commonly referred to as a connection description template CDT.

In such a case, the binder will verify whether or not the desired connection can be achieved with the given CDT and it will confirm the availability of all required resources on the network and the devices, prior to actually implementing the connection.

The CDT, however, can also allow more freedom of the binder to programmatically make decisions on service types and addressing modes. If so, the binder will follow a policy starting from AKCD unicasts and decending down to UCKD domain broadcasts. The binder's preference for addressing is in the order of

Subnet/Node, Group, Subnet broadcast, Domain broadcast

[11] NEURON™ Chip 3120 based single-chip solutions might still include firmware versions prior to version 6, please check with manufacturer for details.

The LonWorks binder, however, does not have a build-in preference policy regarding the service type. It defaults to ACKD, but the binder will not change the service type, since the choice of service type has a potentially huge impact on the correct function of application devices as discussed above.

The consequence is that the binder may be more or less restricted in the way it determines the addressing mode by enforcing or allowing a service type. Table 2 earlier in this document shows these restrictions: using ACKD service will only allow for unicast and group multicast connections for up to 64 members, UCKD/RPT allows for unicast, group multicast, and subnet broadcasts, whereas UCKD allows for all addressing modes including domain broadcasts.

The consequences are:

- The binder will be able to create the largest number of connections if UCKD service is allowed
- The binder will not use domain broadcast addressing if UCKD/RPT service is requested
- The binder insists on group addressing for ACKD multicasts
- The binder prefers the use of groups over broadcasts

A LNS based binding tool may also support advising the (LNS) binder on enhanced policies and decisions. Such advice is provided via two optional parameters, the AliasOptions and the BroadcastOptions parameter.

The AliasOptions preference can be AliasForSelectorConflicts (Default) or AliasForUnicasts. By default (AliasForSelectorConflicts), the binder will allocate one or more input or output aliases to overcome selector conflicts as described in rules 4 to 6 above.

Option AliasForUnicast allows to split a single multicast into multiple unicast connections using one or more aliases to the primary output network variable. This can be used to avoid joining or creating groups, as shown in figures 7 and 8 below.

The benefit of using alias based connections for monitoring purposes obviously is that the monitoring tool does not have to join the group in the first place. The existing group Node 1,2,3 will remain as it is, and Node 1 will also propagate network variable updates to the monitoring tool using an ACKD unicast via subnet/node id addressing. Hence, much less transactions are needed to accomplish this connection in case of the situation shown in figure 8. Also, address table space on the monitoring node is preserved (becoming a member of a group requires an address table entry to accommodate the group membership information).

The policy "AliasForUnicasts" can also be used to avoid multicast connections in the first place. The building automation example in the examples section below introduces a binding problem which can be avoided by splitting a multicast connection into multiple unicasts.

As far as the broadcast related options are concerned, the LNS binder supports three flavours: BroadcastNever, BroadcastAlways, and BroadcastGroup. Whereas the options "never" and "always" are intuitively understood, the third option requires some explanation. The BroadcastGroup option allows the binder to automatically use broadcast

addressing in case a multicast is required and group identifiers are not available any more; provided the service type specified is sufficient (i.e. UCKD/RPT or UCKD).

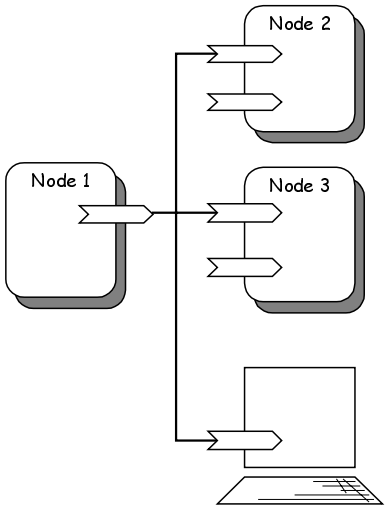

Figure 7: Monitoring as member of a group

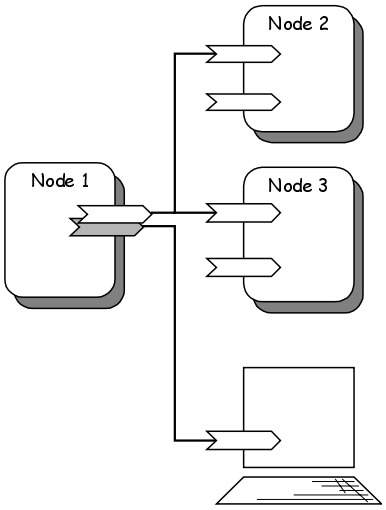

Figure 8: Monitoring via network variable alias

These two parameters allow for the following 6 combinations[12] of binder policy :

	BroadcastNever	BroadcastGroup	BroadcastAlways
AliasForSelectorConflicts	(A)	**(C)**	(E)
AliasForUnicast	(B)	(D)	(F)

Table 3: Combinations of LNS binder policy options

A: Combination A allows for subnet node addressing and multicast group addressing, but it does not allow for any broadcast addressing scheme.

B: Combination B allows for unicasts using subnet node addressing. In theory, combination B also allows for group addressing with one of the unicasts, but the binder will not allocate a group identifier to a unicast (group size must be > 2).

C: Combination C is the default CDT for LNS tools. Assuming ACKD service, this combination allows for subnet/node addressing and group addressed multicasts. Allowing UCKD/RPT also includes subnet broadcast addressing into the list of possibilities for the binder to choose from, and specifying UCKD service also allows the use of domain broadcast addressing.

D: Combination D splits a multicast into multiple unicasts using subnet/node addressing, allocating aliases to the output network variable. In theory,

[12] Note that details of the exposure of said binder policy options are subject to the implementation of the very network management tool. The tool may hide these options entirely, or offer access to these options using different names. If in doubt, you should contact the manufacturer of your preferred LNS network management tool.

combination B also allows for group addressing with one of the unicasts, but the binder will not allocate a group identifier to a unicast (group size must be > 2).

E: Combination E is the recommended option if multicast connections must be build and group based addressing shall be avoided.

F: Combination F splits a multicast into multiple unicasts, but uses broadcast addressing for each of these unicasts. This has the advantage of a potential re-use of address table space on the sending node, on the expense of network bandwidth.

Example: Building Controls

In this example, consider a lighting system in an office building as shown in figure 10. The example includes 9 lanterns in the ceiling, A to J. There are also 4 occupancy sensor devices R to U, connected to the four surrounding lights each. Thus, occupancy sensors R and S necessarily are both connected to lanterns B and E, sensors R and T share lanterns D, E, and so forth.

This common binding scenario presents an interesting problem.

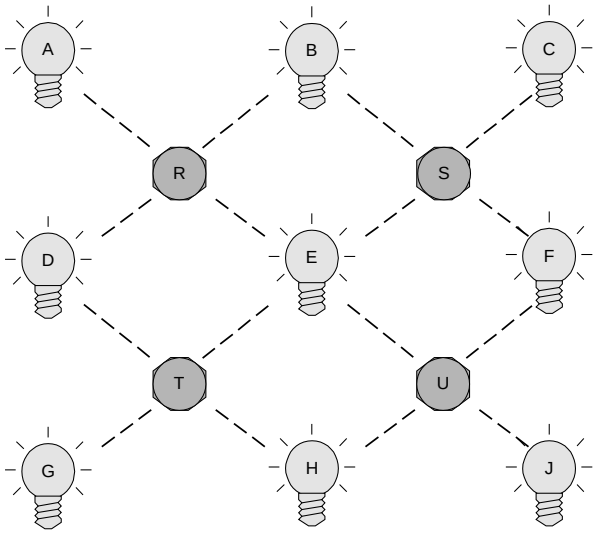

Figure 10: Ceiling lighting with occupancy sensors and bindings

Assuming each occupancy detector has one relevant output network variable only, a multicast connection will be made to connect sensor R to lights A, B, D, and E. By default, the binder will use a group to accomplish this. A group ID will be allocated and one address table entry will be used on each of the five participating devices.

When connecting the sensor S, another group will be created to bind S to B, C, E, and F. This requires another address table space on each of these five devices, and yet another group identifier.

On a large floor with by far more than four occupancy sensors, each light (apart from the ones in the outermost row and column) will have to have four address table entries to accomplish these connections. This may or may not be affordable, but it is also likely that the total number of available groups is used up pretty quickly.

256 groups per domain would only allow for 256 occupancy sensors, that is 16 columns with 16 in each row. Given today's building applications, it seems likely that the above

problem quickly uses up all available groups, and alternative connection solutions are desirable.

The use of broadcast addressing looks like being a way around the use of group identifiers. After all, most of the lights will reside in the same subnet than the occupancy sensors, and subnet broadcasts and UCKD/RPT services seem like an attractive option. Unfortunately, this will not work:

In order to honour rule 6 of the binding constrains introduced above, the binder would have to allocate the same selector Z to both intersecting sensors (e.g. R and S). This is required so that the input network variable on lights D and E does not violate rule 6.

This has an ugly side-effect: each network variable update from, say, sensor R will not only effect the lights A, B, D, and E, but also lights C and F. The effect is known as a network variable leakage due to intersecting broadcasts.

The LNS binder is smart enough to detect this problem beforehand and refuses to connect the second (and any further sensor) due to the detection of a broadcast leakage.

A solution to the problem is to advise the binder to use aliases for unicasts instead of using multicast connections. Since each unicast can have a selector of its own, the leakage problem disappears and the group identifiers remain being available for other connections.

This is done on the expense of alias table space and address table space on the occupancy sensor devices, and also on the expense to network bandwidth: the different unicasts will be processed one after the other, causing more network traffic than a single multicast would have produced.

Solving Problems

When LonWorks connections are being made, frequently seen problems are shortages of group identifiers, aliases, or address table space. Currently available technology should take advantage of the LNS 2 binder algorithms, which include aggressive algorithms to avoid such problems. However, in large systems or in systems accommodating challenging scenarios, the system integrator might still be facing such a problem.

It should be noted that not all available network management tools expose the properties as discussed in the following. Please refer to the manufacturer of your network management tool for details.

Shortage of Groups

A single domain has up to 255 groups. Apart from using binding tools with aggressive group overloading and group re-use algorithms, the integrator (or the automated integration tool) can assist the binder so that groups are only used when they are really needed. A general approach will be discussed in the next chapter, however, here are some suggestions to overcome group shortages:

- Acknowledged groups can be replaced by acknowledged unicasts, using the "AliasForUnicast" binder option. This requires sufficient address table and alias table space on the sending device. It should be noted that splitting a multicast into

multiple unicasts necessarily extends to total transaction time and the total packet count, compared to a single multicast

- UCKD/RPT groups might be replaceable by UCKD/RPT subnet broadcast, assuming all destination devices belong to the same subnet (Note that the source device does not have to belong to the same subnet). This is usually the case if all destination devices belong to a common channel, and if this channel accommodates less than 128 devices.
- Polling fan-in connections always require a group. Such a connection should be avoided, and it is advisable only to use polling strategies in rare cases. Unfortunately, this can only be modified by the device manufacturer, but not at connect-time.
- In case the desired connection must be using a group (for example, for routing purposes), the existing connections can be re-considered. One might be able to identify an existing group (e.g. an ACKD multicast connection) which can be changed to UCKD/RPT subnet broadcasts or individual unicasts as discussed above. This will free a group identifier for the very connection in question.

Shortage of Address Table Space

Generally speaking, address table space can be conserved by using addressing mechanisms with a broad scope. For example, if a device would propagate all outgoing packets using domain broadcasts, this device would only ever need one address table entry per target domain (i.e. a maximum of two, since the NEURON chip can only be member of up to two domains).

A good way to conserve address table space therefore is to avoid groups; replacing them by UCKD/RPT messages using subnet broadcast addressing. Not only that this policy saves group identifiers, it also creates an address table entry which is highly likely to be re-usable by other connections originating from the same device.

The address table does not only accommodate the destination address, though. The second set of data being kept in each address table entry is the set of transport properties like the repeat count, the transmit timer, etc.

Connections should therefore be made with the default transport properties as assigned to by the network tool. The tool will calculate the required parameters as a function of the network topology and the connection shape, and it will typically result in a minimum number of variations.

In summary, address table shortages can be avoided or overcome by the following means:

- Use broad addressing scope to obtain re-usable address table entries
- Avoid groups (for the group membership information itself is kept in the address table on each device which is a member of said group)
- Keep variations on transport properties to a minimum

When groups must be avoided by splitting a single (group-addressed) multicast into multiple unicasts as explained above, this does not only require alias table space but might also quickly consume a huge amount of address table space on the sender device. This is because each of those unicast connections potentially requires one address table entry. The optional binder parameters allow for a combination of

AliasOptions = AliasForUnicast

and BroadcastOptions = BroadcastAlways

This combination (combination F in table 3) can assist to avoid the effect of address table consumption as a result of using multiple unicasts instead of a single (group-addressed) multicast. The binder will split the multicast into multiple unicast, but it will also use broadcast addressing for each of these unicast messages. Assuming at least two targets reside on the same subnet, this will result in re-use of address table entries.

Please refer to another LonSupport courtesy paper, The Address Table [2], for more details on the subject.

Shortage of Aliases

Generally speaking, it is pretty difficult to overcome a shortage of aliases at connect-time. The device manufacturer might not have enable support for the maximum of aliases (62 on a NEURON based device and 4096 on a hosted node), but sadly, this can not be changed at connect-time.

The only mechanism to avoid a shortage of aliases is to avoid connections which require aliases in the first place. For example, avoiding intersecting broadcast scenarios is such a case. An intersecting broadcast situation can be avoided by using group addressing. Consequently, aliases would not be required to solve the problem.

Please refer to the relevant LonSupport courtesy paper, LonWorks Aliases Made Easy! [1], for more details on aliases.

An Alternative Binding Strategy

As pointed out earlier, the LonWorks binder does not automatically modify service types, and it used ACKD services by default. Although this being a very sensible thing to do in most cases, the natural consequence is that the binder, unless otherwise being told, prefers the use of group addressing over other multicast addressing modes.

As pointed out in the above, the LNS 2 binder includes sophisticated and aggressive algorithms to overcome and avoid shortages of groups, address table space, and network variable selectors. In very large networks, it might be desirable to assist the binder so that group identifiers can be preserved for the cases where they really are required. This will lead to an increased number of connections being made, but this includes more intervention by the system integrator or the automated network management tool.

The following strategy avoids the use of groups, using subnet broadcast addressing for multicasts wherever possible. To achieve this, multicasts will always use UCKT/RPT service, and only unicast connections will use ACKD services. The latter is done in the grounds of network bandwidth conservation in a good communications situation, as explained above.

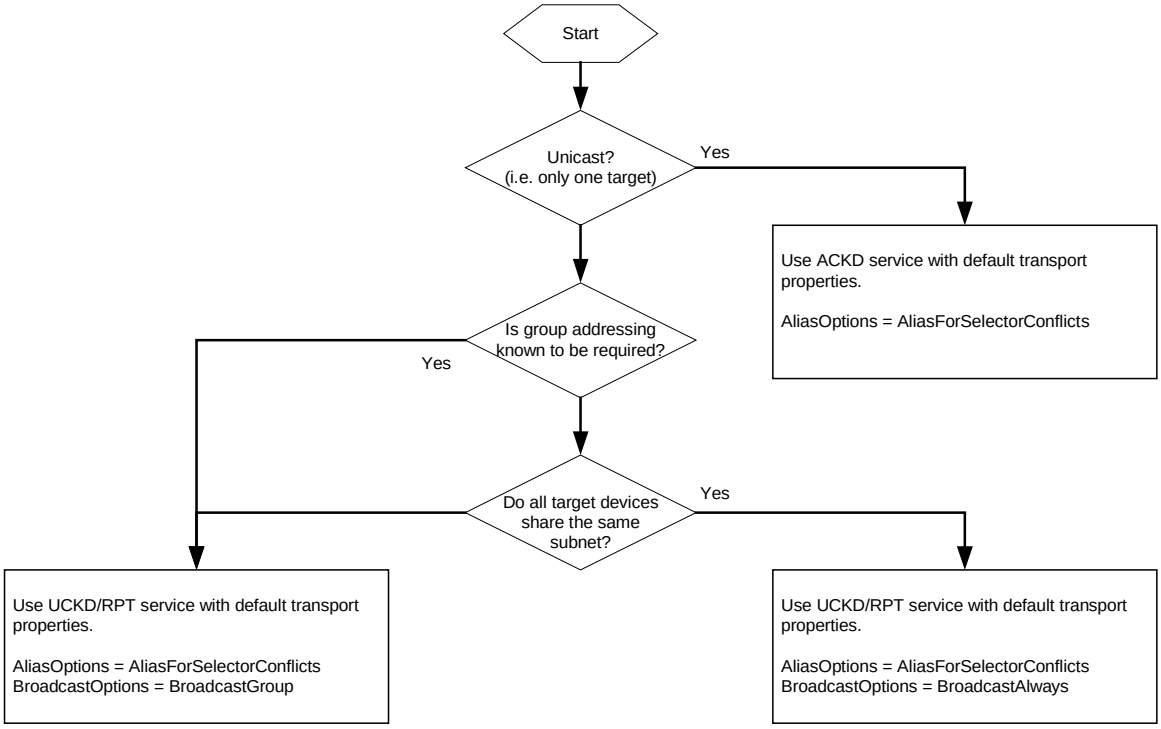

figure 11: Alternative binding approach

It should be noted that the proposed binding policy rules out the use of domain broadcast addressing due to UCKD service being too unreliable for most. Once a connection scheme has been selected as indicated by the flowchart diagram in figure 11, a possible connection failure can be handled as discussed in the relevant section above.

Further reading

1. *LonWorks Aliases Made Easy!* (LonSupport courtesy paper, version 2 or better)
2. *The Address Table* (LonSupport courtesy paper)
3. *Connection Errors* (LonSupport courtesy paper)
4. *The EIA 709.1 LonTalk specification* (see http://global.ihs.com for purchase)
5. *The NEURON C Programmer's and Reference Guides* (Echelon documents, part of echelon's NEURON C development products)
6. *LonMark Interoperability Association Application Layer Guidelines* (www.LonMark.org)

Revision History:

-

Disclaimer:

This document is given to you as a courtesy from LonSupport. Note that this document is not an Echelon Engineering Bulleting, and that it is not part of an Echelon User's Guide or similar documentation. Echelon Corporation and Echelon Europe, Ltd., assume no responsibility for any errors contained herein. No part of this document may be reproduced, translated, or transmitted in any form without permission of Echelon Europe, Ltd.

Echelon Europe, Ltd.
16, The Courtyards
Hatters Lane
Watford
Herts WD1 8YH

United Kingdom

+44 – 1923 – 430 – 100 phone
+44 – 1923 – 430 – 200 technical support phone
+44 – 1923 – 430 – 300 fax
LonWorks@echelon.co.uk email
LonSupport@echelon.co.uk technical support email

