824-Queues and Thresholds explained

Background

The NetLinx operating system runs a series of program threads. These threads deal with all of the input and output aspects of controlling the system hardware, along with interpreting the compiled NetLinx code into machine instructions. The threads pass messages amongst themselves by placing them into **queues** which buffer each message until the recipient thread is prepared to process the message. In order to assure proper operation of the system, there are **thresholds** which limit how many messages a thread may place into the **queue**. If that limit is exceeded, the thread which is generating the messages is **pended** (i.e. paused) until the recipient threads have cleaned enough messages out of the queue to drop the number of backlogged messages below the threshold level. This results in a (typically) brief **delay** in the processing of the thread, but does **not** cause any data loss. If the number of messages backlogged exceeds a high value (**200 - 500** messages, depending on the thread), then messages will be lost for that thread.

Queue Usage

There are at fundamentally two types of queues; Transmit and Manager.

• **Transmit**: Below is the list and a brief explanation of each Transmit queue.

o **Axlink Transmit**: This queue buffers the messages that will be sent to the Axlink devices connected to the master.

o **PhastLink Transmit**: This queue buffers the messages that will be sent to the PhastLink devices connected to the master.

o **ICSNet Transmit**: This queue buffers the messages that will be sent to the ICSNet devices connected to the master.

o **ICSP 232 Transmit**: This queue buffers the messages that will be sent to the ICSP devices connected to the master.

o **UDP Transmit**: This queue buffers the messages that will be sent from the master via UDP (ex. NetLinx Device Discovery messages)

o **NI Device Transmit**: This queue buffers the messages that will be sent to the NI device.

o **TCP Transmit**: This queue buffers the messages that will be sent to the AMX IP devices connected to the master.

• **Manager**: Below is the list a brief explanation of each Manager queue.

o **IP Connection Manager**: This queue manages the messages to the TCP Transmit queue and from the AMX IP devices connected to the master.

o **Connection Manager**: This queue manages the messages to the other transmit queues and from the devices connected to the master.

o **Message Dispatcher**: This queue processes ICSP messages and places the message in the next appropriate queue.

o **Device Manager**: This queue is a storage location to keep track of the current state of devices connected to the master, i.e. Online, Channel state, etc..

o **Route Manager**: This queue is used in a Master-to-Master system and maintains a database of routes to a particular master.

o **Notification Manager**: This queue is used in a Master-to-Master system and maintains a database of local devices that need to report a state change to a remote master

o **Diagnostic Manager**: This queue is used to manage messages to NetLinx Studio, NetLinx Diagnostics, etc..

o **Interpreter**: This queue is for messages being generated by external processes.

Thresholds

As mentioned above there are thresholds that limit the cumulative number of messages that can be placed in the queues by pending (pausing) the thread that is sending the message into the queues, until the queues can process the messages. Two of the thresholds are defined below.

• **Interpreter**: The Interpreter Threshold is a count of the number of OUTBOUND message buffers currently queued up in the system. If this threshold is reached, the Interpreter is pended to allow time for the messages to be processed out. Obviously when the Interpreter threshold is set low (minimum of 10) then a strict throttling of the Interpreter occurs as each NetLinx program activity (ex. on/off,send_level, send_string) causes an ICSP outbound message. With a low number, if those requests are not able to be processed in a quick and timely manner (ex. very low baud rate transactions) the Interpreter will be suspended after just a few outbound requests. By increasing the Interpreter threshold (maximum of 2000) the master is allowed more backlogged outbound requests to pile up before suspending the interpreter. One of the times this may be desirable is during startup when the NetLinx program is loading the sections DEFINE_DEVICE, DEFINE_VARIABLE, etc. and when DEFINE_START is executed which, depending on the NetLinx code, can generate large numbers of outbound messages.

• **IP**: The IP Threshold works in the opposite direction. This threshold indicates the number of INBOUND message at which the IP receive threads will be suspended from reading additional data from their sockets. It is this incoming data that causes additional INBOUND events (ex button pushes, level changes, data events). By pending the inbound socket receive threads the master will allow time for the existing inbound messages to be processed by the system, in particular the Interpreter. By increasing this threshold (maximum of 2000) the master is allowed more inbound messages be queued up within the system.

None of these thresholds guarantees that messages won't be lost. For example, if reading messages from a socket is suspended, the sockets buffering will begin filling up. Also, there are multiple queues/process involved in processing an outbound message.

Queue Sizes

Each of the queues listed above have an upper limit (maximum of 3000) on the number of messages that can be buffered. Regardless of the upper limit configured on a queue the initial value is 25. The limit will then increase as needed in increments of 25, this limit will never decrement. Once the upper limit is reached the queue attempts to optimize the messages by removing consecutive redundant messages. NetLinx master ships from the factory with each queue set to a limit that is suitable for most systems. However, there are times when some queue limits may need to be modified to stabilize the performance of a system. There are negative affects of setting the queue limits to high or to low.

Each Queue and Threshold can be tuned to increase performance on a system; however there is no rule of when to change a queue or threshold value or what the value should be. The installer would need to analyze (via logs or telnet interface) the queue usage and decide whether inbound or outbound throttling will be of use. For example, if the Interpreter queue is filling up, perhaps it would help to lower the IP Threshold...but that will only help if the actual inbound traffic is coming from ICSP IP devices and not some other source.

There are commands available in Telnet to assist in the tuning process. The command “show max buffers” displays the highest value the queues have reached. The command “show buffers” displays the current value in the queues. The command “set threshold” is used to change the upper limit of the thresholds. The command “set queue size” is used to change the upper limit of the queues.

NOTE: NetLinx master firmware v.3.21.343 or later enforced a rule that the Device Manager's queue size must be at least as large as the Interpreter Threshold. The Device Manager is the first OUTGOING queue processing messages from the interpreter. In older masters the Device Manager was able to keep up with the outgoing transactions directed by the Interpreter. There are times that applications written on older masters may flood the Device Manager's queue and eventually overflow it during startup when running on the newer faster masters with firmware earlier than v3.21.343. By ensuring that the Device Manager's queue size is at least as large as the Interpreter, then the Interpreter (which is generating the queued messages) will be suspended prior to the Device Manager's queue filling up.

Advanced Explanation of Queues

NotificationManager and RouteManager are both involved in Master-to-Master communication.

The Notification Manager maintains the database of local devices that need to have their state changes reported to other masters. For example, if the NetLinx program in master A has a device event for a D:P:S in system B, it would register a notification request with master B for the device. Then whenever a device event occur in system B for the device in question, the NotificationManager will check its database, see the registered request, and forward the device event to System A to be processed in master A's NetLinx program.

Similarly, the RouteManager maintains a database of routes to a particular master. So if via URL lists, master A connects to master B connects to master C, master A's database would have records so that it knows to route events destined for master C through master A.

The MessageDispatcher is simply an internal processing queue that sits between the Connection Manager and the higher level FW (ex. Device Manager) It processes ICSP message in from the connection manager, looks at the ICSP message contents and determines which higher level queue the message is destined for and places the message on that queue. Think of it as an internal routine point for external ICSP messages coming into a master's higher level brain functionality (Device Manager/Interpreter, ConfigurationManager, DiagnosticsManager).

RELATED LOG MESSAGES

Log message displaying the Interpreter pending and the Axlink Queue optimizing:

Unknown Interpreter pended because 2012 messages back-logged. Last pend was for 653948 mS/max 2992 messages back-logged.

Unknown Optimization tAxlinkTX complete with 1810 messages now in queue. 1190 messages avoided.

Unknown Optimizing tAxlinkTX messages with 3000 messages in queue.

Unknown Interpreter pended because 2005 messages back-logged. Last pend was for 635806 mS/max 2992 messages back-logged.

Log message displaying the Device Manager adding messages to the Interpreter queue, which in turn causes the Interpreter queue size to increase:

(Reader=tInterpreter writer=tDeviceMgr)- CMessagePipe::Max = 325

(Reader=tInterpreter writer=tDeviceMgr)- CMessagePipe::Max = 350

(Reader=tInterpreter writer=tDeviceMgr)- CMessagePipe::Max = 375

Log message displaying the increase of the internal* queue of the Interpreter:

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 250

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 225

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 200

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 175

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 150

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 125

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 100

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 75

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 50

Interpreter ClpEvent::AddInternalEvent - Max Queue Count = 25

*Note: The internal Interpreter queue is for messages being generated by the NetLinx program (waits, timelines, etc...). The upper limit of this queue cannot be modified by the user. However, the upper limit was changed from 300 to 500 in firmware version 3.21.343, to support faster masters.