

Multiple Networks

Support for both token ring and arcnet

Apr 23, 1990

To support the arcnet connection to the new Smart Rack Monitors (SRMs) to be used in the Linac upgrade project, it is necessary to handle both the token ring network and the arcnet network. This note discusses an approach to simultaneous support of both networks.

The approach is designed to share as much code as possible between the two networks. This means trying to preserve the same protocols for both.

Transmission

A parameter in the message to be transmitted from a local station should indicate which network is to be addressed. Where we have a destination lan-node in the protocol, as for the Acnet header-based protocols, a value of the lan byte can indicate the arcnet network. In the case of the "classic" protocol, the "03" byte can become a lan byte and allow use of a similar scheme.

Suppose there is a lan-network table which gives the network that is to be used for a given lan. The lan# is the index in this table, which should be in non-volatile memory (or in prom). By inspection of the destination lan byte, we get the network to which the message is to be sent.

Classic protocol

The classic protocol presents some special problems because it has no lan byte in the network message. There is a destination node byte which is followed by a byte whose value is always 0x03. To remain compatible with present systems, the byte should have this value when transmitted on the network.

When a message is received, the destination node byte is changed to the source node, which is obtained from the hardware protocol. The 0x03 byte is currently changed to the destination node. This last change is made only so that later message processing can determine whether the message was sent in a group-addressed frame. For example, when a data server request is received, a check is made that it was sent to that single node, as group addressing of data server requests is not allowed.

Suppose we modify the logic in this way for the case of the arcnet network. Change the 0x03 byte to a code which specifies the group address tag in the hi bit (1=broadcast) of the byte. There then remain 7 bits which to be used as the network for the reply. It should have a distinctive value that would not conflict with a lan byte value. An example might be the value 0x7x, where the x is the net#, which is not expected to be a valid lan value on token ring. This could allow for 16 networks, which seems to be adequate.

On the transmit side, when a reply is being prepared, the source node of the request is used as the destination node of the reply. Let the 0x03 byte be used to convey the network to be used in the reply. Suppose it is a copy of the other byte that was received (not including the sign bit). Then a value of 0x7x could be an indication to use network#x. Other values would be considered as a lan# and looked up in the lan-network table to get the net#.

The lan byte would be used to direct the OUTPQX routine to place a reference to the message into the proper network pointer queue. When arcnet messages are collected into frames, the byte will be altered to the value 0x03 in the frame buffer so as not to confuse presently installed nodes which require that the byte have that value.

The point is that the \$03 byte in the classic protocol is used internally to remember only whether the message was broadcast (or group-addressed). It always has the value of 0x03 externally. It will now be used also for retaining the lan used for the reply in the classic protocol case on token ring, and it will be used for retaining the net# for the reply for the classic protocol on arcnet.

Acnet header protocol

A special problem occurs with Acnet header-based protocols on arcnet. Since there is no DSAP byte in the frame header, that method cannot be used to distinguish the two protocol types as was done in token ring. So we must determine the protocol type in the arcnet receive interrupt routine by inspection. The first word of a classic protocol message is the message size. The first word of the Acnet header is the flags/msgType word. Since this word does not currently use the hi five bits of the word, we may assign one of these bits for the purpose, say the sign bit. When a frame is sent on arcnet which uses the Acnet header, the code which queues the frame to the arcnet chip should set the sign bit in the first message in the frame. The arcnet receive interrupt routine will look for this bit set (and will clear it) to decide to which message queue the frame reference message should be sent.

OUTPQX

This routine queues a message to the network given by the lan#. For classic protocol messages, the lan# is the "0x03" byte. For Acnet messages, it is specified in the destination node-lan word for request/USM messages and in the source node-lan word (from the point of view of the requester) for a reply message. The lan# indexes into the lan-net table to yield a net#. The net# indexes into a NETABLE which contains key parameters of that network.

The present OUTPQX routine has with it a related OUTPQL routine which was used to tag the last entry placed into the OUTPQ with a destination node# for the case of re-issuing requests to sufficiently tardy nodes. This is a bad solution for this multi-network scheme since it is difficult to recover the output ptr queue that was last used, so it is proposed that this one be dropped. The code in the Update Task which makes this call should prepare the destination node byte in the external request block (type#5) before calling OUTPQX. Then OUTPQX will properly capture the node-lan and preserve it in the output ptr queue entry for use when messages are collected into frames. The problem for which this is a solution is the case of re-issuing an external request message to multiple tardy nodes at once.

Present lan# situation

With the present system software, the lan byte will turn out to be zero by default, as it has not been used yet. Thus, it may be considered that lan#0 is the default lan. And the default network is the one whose net# is in the lan#0 entry of the lan-net table. By changing the lan-net table, one can make the current software use the any "default" network. Later software will allow the local user to set the lan byte arbitrarily, and that will determine the network to be used.

For the case of replies, the network which delivered the request will presumably always be selected for the reply. The request logic must keep a record of the lan# to use for the reply. To cover the classic protocol case on the arcnet network, there is a means of allowing the receive interrupt routine to specify a net# for the reply, as there no lan# which in the frame header.

NetSend

This routine flushes queued messages to the network. For the case of multiple networks, it is proposed that this call flush all output ptr queues to their respective networks.

This call is made after a logical completion of the building and queuing of network messages. It is quite likely that only one output ptr queue is non-empty.

NetXChk

This routine checks whether a message about to be queued to the network is destined for a different node than the first one awaiting to be transmitted. If it is different, then all messages queued to the network are flushed. For the multi-network case, the question is what network output ptr queue to check.

Since this is merely an efficiency issue, in order to more promptly deliver answer response messages packed into frames, we may consider checking only the default network for the time being. An eventual call to NetSend should flush all network output ptr queues.

The point here is that the Network Layer routines tend to hide the concept of the network from the user. The user merely specifies the destination node-lan, and the network used is handled automatically for her.