

ARP and UDP Interaction

Robert Goodwin
Wed, Jan 13, 2010

Network monitoring by MISS

In recent months, the local application `MISS` has been used to monitor missing data request-related datagrams for Linac. It monitors the `ACNETERR` data stream in data server `node0600`, as nearly all Linac devices are described in the Acnet database as having `LIP600` as a source node. One of the errors monitored by the `GETS32` protocol support is whether an expected reply is missing from a contributing node. If it would have been the first reply that is missing, the error code is “36 -8”, and these are the errors being monitored by `MISS` as it runs in test `node0509`, collecting data stream records of Acnet errors at 15 Hz.

When a “36 -8” error is found freshly written to the data stream, `MISS` collects several types of information quickly, including the network diagnostics from both the server node and the “missing” node. All this is assembled into a 2048-byte record and placed into a circular buffer that has room for 8 such entries. When I notice that `MISS` has written a new record, I print it out for analysis. Such records have not been occurring even once a day of late. (As of this writing, more than 6 days elapsed between the last two occurrences.)

The records indicate some commonality; they are not random noise. Recent examples occasionally occur during Big Saves, when a large number of data requests are being handled. It takes from 1-4 seconds to read some 15000 device properties from Linac, so there is indeed heavy network traffic. But more than occurring during Big Saves, these records show occasions when a large data request is handled, one large enough to require fragmentation for delivery on ethernet.

Server node action

When the server node receives a request, it typically includes devices from more than one front end node, so the server node multicasts the same message it just received so that it reaches all Linac front ends. Each front end previews the request message to see whether it needs to be a contributing node; *i.e.*, if any of its own devices is included in the request. If so, it initializes a request to handle it, and it causes an update to be performed right away and its portion of the reply data returned to the server node. The server node sees all the replies from all contributing nodes and copies the reply data into the complete reply buffer. As soon as the last reply comes in, the server delivers the complete reply to the host.

But if a contributing node’s reply seems to be missing after a rather generous couple of 15 Hz cycles of patience, a “36 -8” error condition is declared for that missing node’s data, and the complete reply message is returned with holes for that expected from the missing node. Every time the server node returns error status in a reply message to the host, it writes a record into the `ACNETERR` data stream that includes the time-of-day, the missing node, the status code, the size of the reply data, and the host node. It is this record that the test node running `MISS` monitors.

Fragmentation

It is commonly accepted in IP implementations that when a UDP datagram is to be sent to a node for which there is no current ARP table entry—and an ARP request message must first be sent to find out the target’s physical address—that upon receipt of the ARP reply, the datagram that was saved in a temporary buffer is delivered. The problem comes when the datagram to be sent is large enough so that it must be fragmented. In that case, only one packet buffer holds what

is to be delivered, which likely holds only the final fragment. The end result is that only that final fragment is sent, and therefore, the target node cannot make sense of it at all. It eventually times out waiting for all the fragments and frees up its buffers, also incrementing a diagnostic counter when this happens.

Does vxWorks exhibit this behavior in its implementation of IP? I think so, but I have no hard evidence, so far. I am not sure of the time-out period for an ARP table entry, but it may be 20 minutes, from what I see on the web.

More such errors?

Why do we not have more errors from this problem than we currently see? When the server node gets a large request message datagram from a host, and it finds that more than one node is represented in the device list (by scanning the second word of each SSDN), it sends the datagram to a multicast destination IP address, as stated above. ARP does not come into play for multicast datagrams, so this problem cannot occur for such requests. Only when all devices in the large datagram are from the same node does the server forward the request to that single target node. This may not occur all that often. The maximum GETS32 request message is for 75 device properties. Since each request packet is 20 bytes under this protocol, we have 1500 bytes of request packets alone. Add space for various headers, and of course it must be fragmented.

Only when a request is for data all from the same actual front end will the server node forward a unicast fragmented datagram. Further, only if node0600 at that moment has no ARP table entry for that front end will the problem occur. Nonetheless, it does happen occasionally, and when it does, errors for 75 device-properties are posted to the web, all because of this single request message. (This actually happened on 01/13/10 at 2245.)

What can be done?

One way to limit the chances of this happening, at least as a result of Big Saves, would be to limit the number of device properties in a single request, so that fragmentation cannot occur. If the maximum number were 70, rather than the current 75, there would be no fragmentation of the datagram carrying the request message. The small difference would have almost no impact on the total time required for a Big Save.

Settings restore

Several years ago, there were problems as a result of settings restore, the symptom being that sometimes the host did not receive an acknowledgment of the settings. After some study, it was determined that this same characteristic related to fragmentation was occurring, such that a large datagram containing settings would not make it to the target node. A Sniffer was used that indeed showed an isolated fragment was sent. As a workaround, the host software was modified so that it would not cause fragmentation when it built a SETDAT message. The suggestion above is based on the same approach, to try not to send requests that require fragmentation.