

Missing Datagrams

Local application

Fri, Mar 4, 2005

In the course of analyzing missing replies to RETDAT requests made through the Linac server node0600, a local application that captures network activity around the time of the missing reply may be helpful. This note describes a local application, called MISS, that can perform this diagnostic service, and its evolution.

Requests that pass through the server node are monitored for missing replies from the contributing nodes that are referenced in the request. For periodic requests, a couple of 15 Hz cycles of grace are permitted. For clock event-based requests, a reply is expected on any cycle for which the event is true as known by the server node. Replies typically arrive during the first 20 ms or so of each cycle. The deadline time imposed by the server node is 48 ms after the start of the cycle. For Linac nodes, the start of the cycle is 3 ms after the time of a Booster reset clock event, a time that was chosen to be 1 ms after the 50 μ s Linac beam pulse. When the server node is about to deliver a reply to an Acnet client, it scans all the status words in the reply message. The first one it finds that shows a nonzero status causes that reply to be logged in a data stream called ACNETERR. The record that is logged includes the original Acnet client node number, the total size of the reply, the first nonzero status word, the node that caused the error status, and the current date and time in the usual 8-byte BCD format.

Each node houses a data stream called NETFRAME used to hold recent network datagram activity. Each record includes the target node number, the size of the datagram, the address of the received/transmitted datagram, and the current time in BCD.

Consider the following parameters for this LA:

<i>Parameter</i>	<i>Meaning</i>
ENABLE B	Usual enable Bit# for this LA instance
SNODE	Server node# whose ACNETERR data stream is to be monitored
RNODE	Matching requesting node#
MSIZE	Matching total request size
MSTAT	Matching Acnet status word
MNODE	Matching missing node#

The LA initiates a 15 Hz periodic request for the latest ACNETERR data stream records. In processing the records in the replies, it looks for a record matching the parameters specified. (A matching parameter not specified allows for any.) If a match is thereby found for all 4 parameters, a one-shot request is made to the node# which the record indicates is missing to retrieve its recent network activity. On the following cycle, this data is called for and logged along with the original record from ACNETERR. Then the next ACNETERR record is processed.

The objective of the resulting log is to show whether a missing datagram was recorded as being transmitted in the network diagnostics of the missing node.

Details

Consider using state machine logic to guide the activities of the LA. When initializing, issue a request for the data stream index of ACNETERR in the target server node. On the next cycle, call for that response, then issue a request for such data stream records to be returned at 15 Hz. On subsequent cycles, process the records returned each cycle. For each that is found to match the input parameters, place the matching record into a queue of records to be processed. Then revisit each queued record, issuing a request to the missing node's recent

NETFRAME data stream records. (The data stream index for this can be assumed to be 0.) On the next cycle, Copy the queued record and the reply records into a log record. The queued record requires 16 bytes, and each NETFRAME data stream record also fits in 16 bytes. If we ask for the most recent 15 records, say, each log record therefore occupies 256 bytes. If we allow for 16 log records, we need 4K bytes; for 32 log records, we need 8K bytes.

The state logic for this might work as follows:

Initialization:

Send request for index for ACNETERR data stream in target server node. Set State0.

State0:

Collect the index from the target server node of its ACNETERR data stream. If valid, send request for ACNETERR data stream records, say, 4 each 15 Hz cycle. Set State1.

State1:

Collect the data stream records and scan each one, looking for a match with the input parameters. For each match found, copy into a matched record queue, unless it is full.

Review the queue. If not empty, send a request for NETFRAME data stream records from this indicated missing node. (Assume the data stream index for this one is 0.) Set State2.

State2:

Collect the network diagnostic records, and together with the matched record, write a new log entry. Set State1.

On any cycle, if the target server node# changes, send a request for the index of its ACNETERR data stream. Clear matched record list. Set State0.

Two list numbers are needed for this data request logic. One is for looking up a data stream index. The other is used by one-shot requests to collect recent network diagnostic records.

Note that in reviewing the logged network diagnostic records, we can expect that the cycle on which we can hope to see a transmitted reply datagram is actually two cycles earlier than that of the last network diagnostic record. (This should be the case for an event-based request, in which the server node expects to see replies from all contributing nodes arrive during the same cycle.) In any case, one should be able to match the time of the ACNETERR record logged with the network diagnostics in the same log entry.

Post-implementation notes

During the course of using the MISS local application, several additions were made to enhance its usefulness. One was to include the Booster reset event for the most recent 16 cycles, ending with the cycle on which the network diagnostics are requested via a one-shot. This provides a context for analyzing the logged results.

Another addition was to include the most recent RETDAT request activity as logged in the RETDLOG data stream. In the systems under test, this data stream is always DSTRM entry 7. The most recent 4 requests are logged. Recall that one can omit recording requests in this log from up to three nodes. This was done for the DAE used in node061C (0C14) and that used for node062C (0C18). This avoids filling up the data stream queue with requests that are probably not of interest for this study. (The DAE routinely issues requests every 80 seconds for replies to be returned every 15 seconds.)

Another addition was more detail for the record of recent clock events. But since there are many clock events, all those > 0x6F are ignored as are all 0x18 events. (The 0x18 event seems

to occur about 3 ms after Booster reset time, the time that Linac nodes start 15 Hz activities.) The most recent 32 such filtered events are recorded. These events are available from the EVTLOG data stream, which is installed for all Linac nodes as DSTRM entry 4. It is useful to be able to assume this in order to avoid adding further delays before diagnostics can be captured for the log. All this required the use of two more list numbers to support the three one-shot requests that are issued at the same time.

Results analysis

With this tool, some logged records showed that on an occasion when node0600 was missing a reply from a contributing node, the missing node did not even try to transmit a reply; nothing was in the network diagnostics on that cycle. Examining the record of recent clock events, several examples of duplicate 0x10 events were seen. This was quite a surprise, since that event indicates a Booster reset event that is neither 0x11 nor 0x12. Further, it seems that there was no 0x52 event recorded on the cycle in question. No wonder that a reply was missing; the contributing node did not see the event on which the reply is based.

Examining the buffer into which the unfiltered EVTLOG records were captured, more detail about the duplicate 0x10 events was seen. Here is an example:

<i>Event</i>	<i>Timestamp, μs</i>
1D	874A95
AF	874A96
0C	874A97
10	874A98
10	874A9A
AA	874A9C
83	874AB9
AB	874ACB

Note that the first 6 events occur bunched up. (The closest in time that two events can occur is 1.2 μ s because of the 10 MHz carrier.) Also note that the time stamp of the second 0x10 is different from the first; in fact, it is the same as normally applies for the 0x52 event. It appears that the second 0x10 event was seen in lieu of an expected 0x52 event. It may be that this is a hardware anomaly, something that is almost perfect, but not quite.

In an effort to further examine the missing 0x52 events, all Linac nodes were set so that their "selected" event was 0x52, and a new local application called EVTC was written to monitor the associated selected counter in all nodes via a one-shot request sent every minute. This has shown other nodes for which missing 0x52 events are seen. In some cases, it seems that one or more nodes are seen to have picked up an extra event 0x52. This could happen if a duplicate 0x52 occurred, perhaps. If this meant that an expected 0xAA would be missed, we would not notice it, as there are no requests active that are based on that event. Note that no observable symptom would occur as a result of a duplicate 0x52; setting a bit in the event bit map array twice does not change anything.

But there seems to be more than merely missing clock events to account for the missing datagrams that are logged into the ACNETERR data stream in node0600. There are cases in which the network diagnostics indicate that the missing node actually tried to send the reply datagram, but node0600 still did not see it. This can happen for an event-based request, but it can also happen for one-shot requests, and perhaps for periodic requests as well. The cause of this is not so clear, but we need to look more critically at the network. All PowerPC nodes in Linac are currently using 100 Mb ethernet, the only operational front ends to do so. It is possible that something is not perfect with the network. This is work in progress.

Later post-implementation notes

After using the MISS local application more to capture missing datagrams, the log records was enhanced further by adding the recent NETFRAME diagnostic records captured from the server node. The final collection of diagnostics logged for each anomalous occurrence is as follows:

- Copy of the ACNETERR record (16 bytes)
- Last 16 Booster reset clock events as known by the node running MISS (16)
- Last 32 filtered clock events as known by the missing node (32)
- Last 4 RETDAT requests received by the missing node (64)
- Last 24 NETFRAME records from the missing node (384)
- Last 96 NETFRAME records from the server node (1536)

All together, the log record occupies 2K bytes. Eight such records can be logged before the circular buffer log "wraps." With this detail, one can easily distinguish the case of a missing reply due to a failure to detect the associated clock event, and the case of a datagram that was sent by a contributing node but was not received by the server node. In actual runs, we detected evidence of both problems. According to the latest understanding, those of the latter case appear to be caused by nodes, primarily two, that were set for 100 Mb ethernet full duplex. After changing them to 100 Mb half duplex, no records caused by those two nodes have been seen after 3 days. The former case may involve the Digital PMC board logic, which is still being investigated at this writing.