

Duplicate Replies

Problem analysis

Tue, Oct 29, 2002

For an Acnet server request that is based upon a clock event, duplicate first-time replies are sometimes observed. This note analyzes how this occurs and seeks to devise a solution for it.

If the Acnet request arrives early in the 15 Hz cycle, before the `Server` task runs, and if the request specifies the `0x0C` clock event, which occurs at 15 Hz, the contributing nodes reply right away as soon as they receive and initialize the request. When the last contributing node has replied to the server, the composite reply is delivered to the requesting node promptly. Later in the same cycle, when the `Server` task runs, it decides that the request should be fulfilled, since it is based on an event that is true for this cycle (any cycle for `0x0C`). The data sent for both replies during the same cycle will be the same.

The system code behind this logic exists in order to provide prompt replies to one-shot server requests. When all of the contributing nodes have replied, the first composite reply can be delivered. Since each contributing node will deliver a prompt first-time reply, no delays are introduced for one-shot requests, even when they pass through a server node. (Delays only occur if the server node or a contributing node is busy with some other task.)

Similar logic exists for Classic server requests, so that Classic one-shot requests are answered promptly. In the Classic case, the first-time reply for a periodic request also gets the same treatment. But event-based Classic requests do not. This means that the first reply from a server node cannot occur before the `Server` task time of the cycle, which is typically 40 ms, although it is 48 ms for the Linac server node `0600` as of this writing.

Returning to Acnet protocol, here is the list of references to the `SAGE` field of the `SRB` (Server Request Block) structure.

In `SERVER`, as a server request is initialized, `SAGE` is set to `-2`. In `REQUEST`, the `FTDCNTR` field is initialized to 2 or 3 if a periodic request, or to `-1` if an event-based request.

In `SREPLY`, while processing a reply from a contributing node, if the `SAGE < 0`, count 1 for a first packet received from this node. Reset `SAGE` to the low 15 bits of `CYCLECNT`. If this is a first time reply that completes the reply buffer, clear `FTDCNTR` and call `ACUPSERV` to deliver a prompt reply to the requesting node.

In `ACUPSERV`, if `SAGE >= 0`, it implies that a reply has been received since the request was initialized. Its value in this case is `CYCLECNT & 0x7FFF` as of the time the reply was processed. If the request is not an event-based request, and the number of cycles since that time $>$ the period indicated by the `FTD`, then answers are tardy, a `36 - 7` error will ensue, and `SAGE` is reset to a suitable delay count, which is the #periods in 2 seconds, if the period is less than 30 cycles, or 1, if the period is at least 2 seconds, less 1. If a re-send is required, each successive `SRB` has `SAGE` set to `-1000` in order to prevent multiple re-sends as it continues to look for `SRB` entries that indicate a resend is needed. A re-send is indicated if `SAGE = -2`. Then `SAGE` at the current entry is set to the suitable delay count described above.

After queuing reply message to the network, if a periodic request, set `FTDCNTR` to the period in 15 Hz cycles.

Let's try to follow the logic for a simple case, that of a 15 Hz request sent to node `0600` that asks for data from another node. `SAGE` is initialized to `-2`. Suppose the request arrived early in the cycle, before the `Server` task runs. It is immediately forwarded to the other node, and a reply promptly ensues. In `SREPLY`, `SAGE` is set to the low 15 bits of `CYCLECNT`. Because this is the first-time reply that completes the reply buffer, `FTDCNTR` is set to zero, and `ACUPSERV` is called to deliver the prompt reply to the client. This results in setting `FTDCNTR` to 1. When the `Server` task runs, `ACUPSERV` is called for a second time in the same

cycle. When FTDCNTR is decremented, the result is zero, so another reply (a duplicate reply) is delivered. On the following cycles, there will not be duplicate replies, since the update count (RQUPDTCT) is no longer zero.

In this example, if the request had arrived after the Server task time, a prompt reply would be delivered, but the second reply would not be sent until Server task time during the following cycle; thus, there would not be duplicate replies.

How would the example differ for the case of a 15 Hz event, such as 0x0C? The prompt reply is still delivered, but FTDCNTR is set to -1, which is normal for the event case. At Server task time, the second reply will be delivered, as the event is still true, of course.

A possible solution for this problem: One-shot requests should always get prompt replies. But if the request is either periodic or event-based, only deliver a prompt reply if the Server task has already run during the present cycle. If the Server task has not yet run, omitting the prompt reply will merely defer the first-time reply until the Server task runs. This should eliminate the duplicate reply problem. Here is the assembly code solution at end of SREPLY:

```

MOVE.L A2,A0                ;Ptr to request block
TST.L RQUPDTCT(A0)          ;Count of #times request updated
IF# EQ THEN.S               ;Not yet the first time
  MOVE.L RQABLKPT(A0),A4    ;Ptr to answers block
  MOVE D6,SPARE1(A4)        ;Accumulate #packets received so far
  IF# D6 EQ RQREQ+NRPKTS(A0) THEN.S ;all packets accounted for
  MOVEQ #1,D0                ;MLT mask in message type word in Acnet header
  AND RQAHDR+MSGTYPE(A0),D0
  IF# NE THEN.S              ;not a one-shot request
    CLR FTDCNTR(A0)          ;Update first response in Server task this cycle
    BTST #0,SERVDONE(A5)    ;flag=1 if Server task already run this cycle
    EORI #4,CCR              ;EQ if Server task already run
  ENDIF#
  IF# EQ THEN.S              ;one-shot OR Server task already run
    CLR FTDCNTR(A0)          ;Update first response now,
    BSR.W ACUPSERV           ;just as Server Task normally does.
    IF# NE THEN.S           ;Queued to network
      MOVE.L #$000400+UPDTTASK,D0
      BSR.L TRIGHTASK        ;Flush network queue
    ENDIF#
  ENDIF#
ENDIF#
ENDIF#
ENDIF#
ENDIF#

```

Testing with an event 0x0C request, this code seemed to prevent the duplicate replies. The request arrived early in the cycle, but no prompt reply ensued. At Server task time, however, the reply was delivered to the requesting node. A one-shot request got a prompt reply.