PolarSoft® Inc.
914 South Aiken Ave
Pittsburgh PA 15232-2212
412-683-2018 voice
info@polarsoft.com
www.polarsoft.com

Source: **Brian Conroy on BACnet-L (12-Jan-2004)**

Question: *Just wondering about the traffic issues brought about by having all nodes on an MS/TP network acting as master nodes. The reason I want to do this is so that all devices can issue "i-am" requests. Without the ability of all devices to issue "i-am", I feel that a BACnet network becomes much less flexible in terms of cross-vendor operation, and even in terms of dynamically adding devices made by the same vendor.*

Does anyone out there have any experiences that they can relate about any possible traffic problems caused by having (possibly) 32 master nodes on the network. This seems like overkill , but my desired implementation really requires that all devices are able to issue "i-am" requests.

Alternately, anyone have any clever ideas about other ways to overcome this dynamic discovery limitation? Carl Neilson mentioned in an earlier email that "There are changes that are either under public review, or that have passed public review that provide mechanisms to work around this limitation." Any more news on these changes?

Carl Neilson observes:

All Delta Controls MS/TP product are masters and we have no problem with network traffic or response times and our installation are routinely larger than 32 nodes per network.

In our experience, the most important factors for MS/TP network performanace are:

1) *Baud rate - Products should support 76.8K (or at least 38.4K). Baud rates below 38.4 are too slow.*

2) *Execution of ReadPropertyMultiple - All of our products support ReadPropertyMultiple, even our smallest sensors. From our expereince this alone will make an enormous difference to the network throughput. If you had to choose between supporting 76.8 and RPM, support RPM.*

3) *Response time - We have occasionally encountered products that take way too long to respond to requests, which do not use the Reply_Postponed mechanism. While the standard allows for up to 300 ms for a device to generate a response, any device that takes this amount of time will kill the network. In general it is better that a device reply with a Reply_Postponed sooner which will allow the network to continue working instead of waiting out the whole time period that the standard allows.*

Answer: **David Fisher**

Carl's advice is good, but in my view there are a bunch of other issues to consider as well.

1. In our experience, the number one cause of delay and slowness in MS/TP networks, is the improper configuration of MaxMaster. Many people either choose to not implement a writable MaxMaster (very bad) and thus default it to 127, or they simply don't configure a default setting of 127 to something lower.

QUESTION

This has the often terrible side effect of causing a periodic poll of 127-N (where N is the highest actual master MAC address in use) devices with PollForMaster frames. Since each such poll must wait a minimum of 20ms (or longer) for a reply this can cause huge periodic gaps of inactivity. If you have a network of, say, 30 units then about every 100 token cycles there will be 97 PFMs issued.

2. One of the most important issues for MS/TP interoperability is the Tusage_timeout. The standard (unfortunately) allows as little as 20ms and as much as 100ms for this value. The way it is worded makes it somewhat ambiguous for the implementor.

   In general, you should respond as quickly as possible to PollForMaster, Token Pass and such frames. For maximal interoperability though, you have to be prepared to interact with peers who are at the slower end of this parameter. Some MS/TP implementations are fussy about this and give up long before 100ms. This leads to interoperability problems in some scenarios. Our advice: respond quickly, be patient when polling.

   This setting has a huge effect on overall network throughput, much more so than the factors Carl advised about because this setting imposes hard baud-rate-independent required latencies.

3. RPM is a good and a bad thing. Unless you implement segmentation, which most devices, especially small devices, don't do, RPM is of limited utility. It is good in those cases when the reply will fit in a single PDU. Unfortunately there are some common scenarios when that isn't the case. For example, when reading the Object_List property of a device object which has more than about 93 objects. There are many other common examples.

   The successful use of RPM as a throughput accelerator depends on very sophisticated clients. If your network (or your customer's network) traffic is dominated by such clients then RPM is a valuable tool. But otherwise, it's only a small part of improved performance.

4. To Reply_Postponed or not to reply quickly, that is the question. I agree with Carl that a snappy reply is always a good thing. But I don't agree that always postponing is the ideal policy. If a device always postpones its reply, then (on the average) it will take N/2 token cycles of latency to ever get a reply from that device. While this might be in some circumstances "fairer" to the network overall, it is almost never "faster" or "higher throughput". Carl is definitely right that a large number of pokey responders is bad. But there is a large continuum between the maximum reply time and the latency of one or more token cycles. I think the point is that one should labor to make the latency between the receipt of a request PDU by the MSTP MasterNodeStateMachine and the processing and reply to that request by the upper layers, as short as possible. In those devices where this is feasible, I don't think it's unreasonable to forego Reply_Postponed. But that's if you can reply in a sprightly timeframe. Let's think 5-10ms not 255...

   The reason to consider this carefully is that the decision to implement an "always postpone" doctrine vs. a direct reply doctrine has big implications on the architecture of the processing pipeline in your implementation, and in particular the number of and kind of buffers you require. If you want to keep buffering requirements to a minimum, you will be hard pressed to choose the doctrine of procrastination.

