JET PROPULSION LABORATORY INTEROFFICE MEMORANDUM
SFB: 363-87-005

AUGUST 12, 1987

COUNTEREXAMPLES

Steve Bellenot

This memo could also be titled “failures” in that the examples below have shown that certain policies fail. I list these failures here so that if anyone asks “Have you tried ... ?” You can answer “No, but we know it won’t work because of ...”

Another way of viewing these examples is that they show that simulations can be schizoid. That is simulations can have two or more natures which work against each other.

A NODE WHICH IS BOTH FARTHEST AHEAD AND FURTHEST BEHIND.

This was observed on node 0 while running Commo12 using commo12b.cfg (see my IOM 363-87-003). Node 0 was farthest ahead in that it was running objects with LVT over 400 (self-propelled message generators). However, node 0 was also the furthest behind in that there were objects (commo objects) that were blocked awaiting query replies at LVT around 10.

In particular, this shows that you can’t just stop running objects when there is little free memory. Even on nodes which have lot’s of self-propelled objects greedily consuming all of your memory, this could be the node furthest behind. Thus GVT will not advance, no memory will be garbage collected, and the simulation will hang.

It turns out that this node has the least to do. Indeed, given enough memory, the message generators quickly run to the end of the simulation. These generators get rolled back, but they “resend” the same messages. This is the node with the most calls to set null, the routine which TW tells the Tester it has no objects to run.

LIMITING THE NUMBER OF SAVED STATES PER OBJECT.

Trying to synchronize by limiting the number of saved states per object can work. But it could slow you down a lot. Commo14 is the example this time. Since in Commo14 almost every object has an event at almost every time, GVT can only advance x virtual time units, where x is the number of saved states permitted.

Making x small enough to synchronize the simulation while GVT < 250, will greatly slow you down when GVT > 250 where the simulation moves much faster. When GVT < 250, each GVT update only increases GVT by at most 3 on the Suns and by at most 11 on 32 hypercube nodes. Thus making x small has little effect on the rate GVT increases while GVT < 250. However, when GVT > 250, some GVT updates increase GVT by 50 or more. Thus some objects can need 50 or more saved states so that GVT can make these big increases in one GVT interval. If x were 5, it might take 10 GVT intervals to get GVT to increase by 50, a waste of 90%. On the otherhand, making x large doesn't synchronize the simulation.

AN INPUT BUNDLE CAN HAVE ONLY QUERY REPLY MESSAGES.

A “bundle zap” in Commo12 produced an input bundle (the collection of messages for one object with a given receive time, say RT) consisting of three query reply messages. The query messages which caused these query replies and an event message were in the output bundle with send time RT. The event message (with receive time RT) which caused all this activity was later annihilated.

This shows some care is needed at go_forward time. (Or we could get rid of query messages.) There is a science fiction story called “by his bootstraps” which comes to mind.