Distributed Simulation

Optimistic Algorithm with Global Virtual Time

*Thomas Laverghetta (*[*tlave002@odu.edu*](mailto:tlave002@odu.edu)*)*,   
Old Dominion University – Computational Modeling and Simulation Engineering

Please read over HW 6 System Requirements document (which accompanies this report) first to gain understanding of program overview.

# Algorithms

In distributed optimistic algorithms, it is greatly beneficial to know the smallest timestamp of all process in the simulation. With the smallest timestamp, old events can be eliminated, state saves can be deallocated, I/O can be committed, etc. To find the smallest timestamp so to conduct processes, global virtual time algorithm is implemented.

Global virtual time (GVT) is a three-state process to approximate smallest timestamp among all processes in simulation. To approximate GVT, the algorithm breaks the messages into three domains: blue, green, and red. Where the message domain starts with green then translate to red then translate blue then back to red. Figure 1 illustrates the message domain transitions via a state-machine diagram. As seen in the figure, each domain represents a state in the GVT process: GVT Init, GVT Calc, and GVT Broadcast (conducted in green and blue domains).

* GVT Init: GVT Init initializes the GVT process by translating green/blue domain to red domain and gathering number of blue/green messages (which ever was the previous domain) sent to each process. GVT Init is initialized by LP0 when GVT event occurs (for this paper, GVT event is 100-events are executed by LP0), at which, LP0 will send GVT Init message to LP2 and so on until messages go all away around to LP0 again (went around communication ring).
* GVT Calc: GVT Init has gone around ring, LP0 will determine using the gathered number of blue/green messages sent (from GVT Init) to itself from other process if all messages have bee recv’d (no transient messages). If LP0 has not recv’d all msgs, it will continue process and wait for msgs to be recv’d. Once all msgs are/have-been recv’d, LP0 will translate from red to green/blue domain (which ever was not last domain), reset green/blue receiver counters, and LP0 will estimate GVT by taking the min of simulation time and event set time. Then LP0 will send estimate GVT estimate and blue/green msg counter and red msg counter to LP2.
  + LPn will conduct the same calculation as LP0 except including GVT estimate recv’d into its own GVT estimation (i.e., will take the min of simulation and event set time and GVT).
* GVT Broadcast: When GVT estimation has been recv’d by LP0 (GVT Calc was conduct by all LPs), then GVT estimate recv’d is the final GVT approximation. Thus, LP0 will broadcast GVT to all LPs. Then using GVT, LPs will clean up events, state-saves, I/O, etc. up to and including GVT.

After GVT Broadcast, the simulation will wait until next GVT event to start the whole process again (starting from GVT Init).

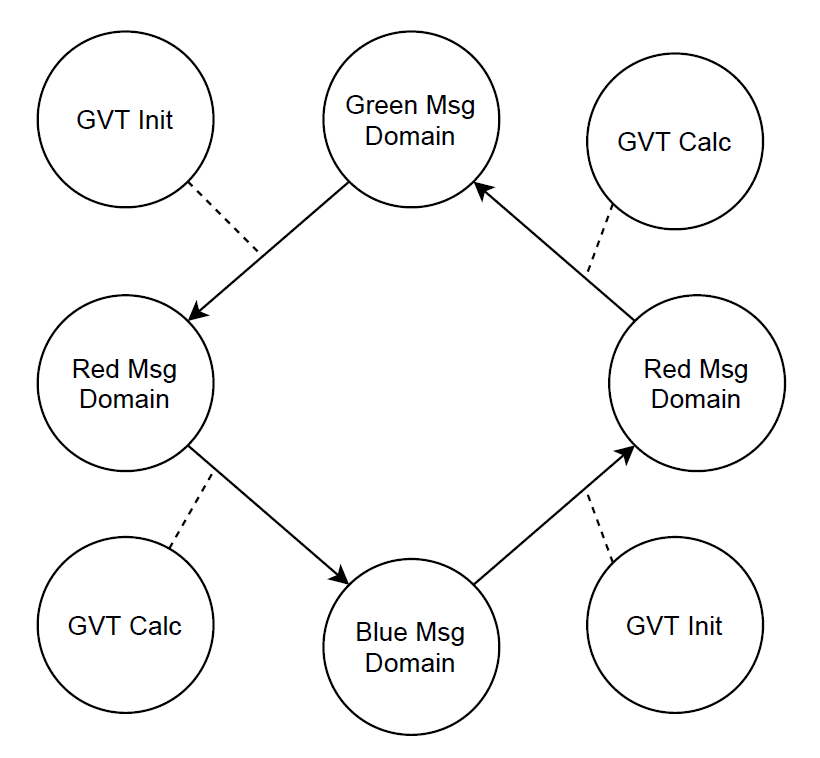


Figure . Message Domain State Machine

## Pseudocode

Given the description of GVT in the previous section, pseudocode can be created so to produce behaviors. Three methods are used to conduct GVT: Send, Receive, and Run Simulation.

Run Simulation will be tasked with determining when GVT event occurs (when to send GVT Init) and sending GVT Init. For this paper, GVT event will be when 100 events are executed. Also, this event is only tracked for LP0 since LP0 initializes GVT Init. Once GVT event has occurred, Run Simulation will initialize green/blue counter array to LP0’s counter values, reset LP0’s counter values to zero, and send GVT Init to next LP. The pseudocode:

*RunSimulation():*

*While true:*

*Start Do-while:*

*Check for new msgs*

*While event set is empty*

*Set simulation time to event set time*

*If simulation time > termination time:*

*Break*

*Get and execute event action from event set*

*If number of events executed >= 100 AND LP == 0 AND all red messages have been recv’d:*

*Reset red recv’d and event executed counters*

*Set current message domain to red*

*Initialize green/blue counter to LP green/blue counter values*

*Serialize green/blue counter*

*Send green/blue counter to next LP*

*END RunSimulation*

Send method will be tasked with counting number of colored messages are sent to each LP. The counter information is used during GVT Init so to combine with GVT Init blue/green counter. It is also during GVT to determine if all red messages have been recv’d. The pseudocode:

*Send(dest, time, ea):*

*Serialize ea, ea id, time, and current color*

*Increment counter for current color domain (green, blue, or red)*

*Send serialized msg to dest*

*END Send*

Receive method will be tasked with processing GVT messages (GVT Init, GVT Calc, and GVT Broadcast) and waiting for transient messages. Starting GVT message processing, there are four scenarios to consider: when GVT Init is recv’d by LP0 (GVT Init when around communication ring), when GVT Init is recv’d by LP other than LP0, GVT Calc is recv’d by LP other than LP0, and when GVT broadcast is recv’d.

* GVT Init recv’d by LP0. In this scenario, LP0 needs to deserialize GVT Init and determine if all green/blue messages have been recv’d. If all have not been recv’d, then it must set a flag stating to check recv’d message to determine if all green/blue have been recv’d (transient messages). If all messages have been received, then LP0 will have to estimate GVT, serialize GVT estimation, serialize red counter array, and send to next LP.
* GVT Init recv’d by LP other than LP0. In this scenario, LP needs to deserialize GVT Init and determine if all red messages have been recv’d (got knowledge about number of red messages at GVT broadcast). If all red messages have not been recv’d, then LP must set flag to check recv’d message to determine if all red messages have been recv’d. If all red messages have been recv’d, then LP translate to red domain, LP will add its green/blue counter values to GVT Init’s values, reset green/blue counter values, serialize counter, and send GVT Init to next LP.
* GVT Calc is recv’d by LP other than LP0 (LP0 did estimation at GVT Init recv’d by LP0). In this scenario, LP will deserialize green/blue counter and will have to determine if all green/blue messages have been recv’d. If all have not been recv’d, then it must set a flag to check recv’d messages to determine if all green/blue messages have been recv’d. If all green/blue messages have been recv’d, then LP will translate to green or blue (which ever was not previous), reset green/blue recv’d counter, will estimate GVT, add red counter values to GVT Calc red counter values, and serialize and send GVT Calc to next LP.
* GVT broadcast recv’d by LP. If LP0, then LP0 will broadcast GVT to all LPs and conduct event elimination (removing old events). Else, LP will conduct event elimination.

Now the transient messages. When a message is recv’d the receive method will check if LP is waiting for red or green/blue messages, and if either are true, it will determine if all messages are recv’d. If they are, then it will conduct the corresponding task associated with that wait. For example, for red msg wait, it will conduct GVT Init processing (process discussed in GVT Init recv’d by LP other than LP0 scenario).

Receive method pseudocode:

*Received(source,tag):*

*If tag != GVT message tag:*

*Recover and deserialize incoming message*

*Increment recv’d color counter given color deserialized*

*If waiting red AND all red messages have been recv’d:*

*Remove waiting red flag*

*Reset red recv’d counter*

*Add green/blue msg out counter values to GVT Init’s out counter values*

*Reset green/blue msg out counter values*

*Set current color to red domain*

*Re-serialize GVT Init with new counter values*

*Send GVT Init to next LP*

*Else if waiting green/blue AND all green/blue msgs have been recv’d:*

*Remove wait green/blue flag*

*Reset green/blue recv’d counter*

*GVT = Min (Simulation time, event set time, GVT)*

*Add red msg out counter values to GVT Calc red msg out counter values*

*// or initialize red msg if LP0*

*Reset red msg counter values*

*If LP is last processor before LP0:*

*GVT Calc = Serialize GVT and red msg counter*

*Else:*

*GVT Calc = Serialize GVT, red msg counter, and green/blue msg counter*

*Send GVT Calc to next LP*

*Else: // GVT msg TAG*

*If LP0 AND current color = Red: // GVT Init recv’d by LP0*

*Deserialize GVT Init*

*If all green/blue msgs recv’d:*

*Reset green/blue msg recv’d counter*

*Change color to blue if green was previous or green if blue was previous*

*GVT = Min (simulation time, event set time)*

*Initialize red msg counter and set to LP0 red out counter*

*Reset red out counter*

*GVT Calc = serialize (GVT, green/blue out counter, red msg out counter)*

*Send GVT Calc to next LP*

*Else: // have not recv’d all green/blue msgs*

*Set flag waiting for green/blue msgs*

*Set GVT to max time // forces estimation to be simulation and even time*

*// in waiting green/blue conditional*

*// GVT Init recv’d by LP other than LP0*

*Else if Not LP0 AND waiting for GVT Init AND current color is green or blue:*

*Deserialize GVT Init*

*If all red msgs have been recv’d:*

*Reset red recv’d counter*

*Change color to red*

*Add green/blue out counters to GVT Init out counters*

*Reset green/blue out counters*

*Serialize GVT Init*

*Send GVT Init to next LP*

*Else: // all red have not been recv’d*

*Set flag to wait for red msgs*

*// GVT Calc recv’d by LP other than LP0*

*Else if Not LP0 AND waiting for GVT Calc AND current color is red:*

*Deserialize GVT Calc*

*If all green/blue msgs recv’d:*

*Reset green/blue recv’d counters*

*Change color to blue if previous was green or green if previous was blue*

*GVT = Min (simulation time, event time, GVT)*

*Add red out counter to GVT Calc red out counter*

*If LP is last LP:*

*GVT Calc = serialize (GVT, red out counter)*

*Else:*

*GVT Calc=serialize(GVT, red out counter, green/blue out counter)*

*Send GVT Calc to next LP*

*Else: // all green/blue msg have not been recv’d*

*Set flag to wait for green/blue msgs*

*// GVT broadcast*

*Else:*

*Deserialize GVT and red out counter*

*Set GVT to LP GVT*

*If LP0:*

*Broadcast GVT and red out counter to all LPs*

*Remove all events less than or equal too GVT*

*END Received*