Report on Final Assignment for Distributed Systems II

Daniele Parmeggiani

Parameter	Values
Protocol	Newscast, Cyclon
Graph	geo, random, lattice, star
Nodes	1000
View size	20, 50, 100
View to send size	19, 49, 99
ΔT	1, 4, 10
Disaster intensity	50%, 65%, 80%, 90%, 95%

Table 1. Parameter values used.

Nodes		Latency
global	global	10
geographical	geographical	4
local	local	1
global	geographical	4
geographical	local	1

Table 2. Latencies of edges of a Geo network.

The topic chosen for this project is among those suggested: comparing the Newscast and Cyclon protocols.

In this report, one particularly emphasized aspect is the underlying communication network's latency effect on the protocol. Such was considered after reading this passage in [Voulgaris et al., 2005, §3.2]:

> Nevertheless, ΔT should not be comparatively short to twice the typical latencies in the underlying network, as network delays would unpredictably affect the order in which events are taking place.

It occurred to try and model network latencies in the protocol, which was attempted in the presented implementation.

1. ON THE MODELING OF THE EXPERIMENTS

The experiments are modeled using both agent-based, and discrete event simulation techniques. The reason for using agent-based modeling is self-evident in the nature of the experiment, i.e. instances of some protocol that communicate with each other is a prime situation in which to use agent-based models. The reasoning for choosing to use discrete-events lies in the necessity to model message latency as well, and is presented in detail in the following §1.2.

In this model, we need to consider two kinds of networks, let us call them communication network and overlay network, even though these two terms are used interchangeably for instance in Jelasity et al. [2004]. Respectively, they are intended to model the ISO/OSI Layer 3 network, and the application layer communication; that is, the graph that can be constructed by considering the views of the nodes in the experiment.

The experiments are carried out over both protocols, using the parameters presented in Table 1.

Let us briefly discuss the different graphs mentioned in the table:

- *geo* is explained in detail in the following §1.1,
- *random* is an Erdős–Rényi graph as in $G(n = 1000, p = 10\%)$,
- *lattice* is a ring network of 1000 nodes, and
- *star* is a star network, with a central node and the rest being the $1000 - 1$ nodes connected to it.

Although the suggested parameter value of 100 000 was considered, it was ultimately not used because of the higher memory requirement. Using 1000 nodes allowed the experiment runs to be completed on the system available.

Let us distinguish two terms to be used as time units. We may distinguish the notions of steps, as in steps of MESA's schedule, and of cycles, that is, let a cycle be the amount of steps equal to ΔT. This is useful so as to consider the protocols behavior regardless of the value we may assign to ΔT. It is reasonable to assume that in the steps between cycles, differences in the metrics evaluated (see §1.5) are not great.

1.1 The Geo network

One structured network over which latencies are also modeled is referenced here and in the code as a *Geo* network.

Ideally, this graph is intended to model a network that spans greater geographical regions, having one core "global" network of 6 nodes, 6 geographical networks each composed of 6 nodes, and $\lfloor \frac{1000}{36} \rfloor$ nodes for each local network (that is, a network attached to any of the 36 geographical nodes).

The latency of graph edges is defined based on the class of involved nodes, and is summarized in Table 2.

The total latency between nodes is then computed by using Dijkstra's algorithm.

The interest in this particular network lies in finding a scenario in which partitioning of the network seemed very likely. That is, we're interested in finding the effect of latency over the two protocols, namely whether or not they're able to avoid partitioning in this network, and under what circumstances.

1.2 Messaging and Latency

In the Cyclon and Newscast protocols, nodes exchange messages with each other. In the presented implementation, this is modeled by means of a discrete event simulation.

Essentially, each message is pushed into the recipient node's incoming messages queue (which is a heap queue), sorted by the time the message is to be read.

The reading time is computed by the sender node which uses a pre-computed routing table, which is then used to compute the total latency between the two nodes. In all networks, except the Geo network, all edges have latency $= 1$.

A peculiar aspect of the modeled messaging system is that nodes have an infinite capacity to process messages; that is, no matter how many messages a node has to process at any given step t, at step $t+1$ all those messages will have been processed. While not being an accurate representation of really existing computing networks, the underlying idea is that the computational complexity of processing messages pertaining to these protocols should be negligible.

Throughout an experiment, an agent i (being the unique numeric identifier of each agent) may choose to send a message at step t, if the following holds:

$$(t + i) \bmod \Delta T = 0$$

This yields notable properties:

- the messages respect the cadence of being sent once per cycle, and
- the messages are not sent synchronously.

The latter being an important property mentioned in [Voulgaris et al., 2005, §2.2]. We can notice that this in effect will entail that at each step, the number of messages sent is the same, since there's a uniform probability that an agent will send a message at any of the steps in a cycle. This may not model exactly the behavior of a really running protocol, but no evidence pointed to consider this as a possible refutation of the results presented.

1.3 Disaster and Resurrection

At cycle 100, a disaster is set to happen, meaning that an amount of nodes equals to 1000 $\times$ disaster intensity is set to die: losing its memory and halting communications. Some nodes are exempt from dying: in the Geo network only local nodes can die, and in the star network, the central node cannot die. This is intended to model a supposed higher availability of certain nodes that are more important in the communication network.

Note that regardless of a node's state, routing of messages is not affected. This actually has no implications for the Geo and star networks, since the routing nodes are exempted from dying, but in the lattice and random networks this behavior may sound a little unreasonable. Nevertheless, we're mostly interested in creating an evaluation baseline in these graphs: this improper feature should not render the results meaningless.

At cycle 200, all the nodes that died are resurrected, and at cycle 300 the experiment finishes. Resurrected nodes undergo bootstrapping once again.

1.4 Bootstrapping

By bootstrapping, in the context of the presented implementation, we mean the mechanism for which the view of a node is populated, either at the beginning of the experiment, or when the node is resurrected.

The bootstrapping mechanism differs for each network type. Let us examine them separately.

In the lattice network, every node is bootstrapped with the two other nodes that are adjacent to it.

In the star network, the central node is bootstrapped with 5 random other nodes, the rest are bootstrapped with the central node.

In the random network, each node is bootstrapped with 5 other adjacent nodes, randomly selected.

In the Geo network, the bootstrapping is intended to model some particular features that are considered to be reasonable for such a topology: the local nodes are bootstrapped dynamically, and the 42 global and geographical nodes are bootstrapped statically to know each other; that is, every global node knows about every other global node, and the geographical nodes that are adjacent to it, while the geographical nodes know about the other nodes in their geographical network as well as the global node they are related to. Additionally, the geographical nodes are also dynamically bootstrapped with 5 random nodes in the local network for which they are the "gateway." Finally, the local nodes are bootstrapped with 5 random nodes in their network, as well as their geographical "gateway."

The dynamical behavior of the local nodes is intended to model a bootstrapping mechanism involving a local network broadcast request. Therefore, we can choose some random nodes in the same "LAN" to be responding to this request. In particular, we choose 5 as the number of nodes that will respond, this is because in an actual implementation of such a broadcast we wouldn't want to have the broadcasts to be actually executed by every node, so as to not flood the network with such requests, especially when the protocol is initialized over the network.

So, a somewhat obvious optimization is to have a cool-down period at the beginning of a node's bootstrapping in which the node listens to the possible broadcasts requests occurring, and only after that actually initiate the broadcast. If the initialization is not simultaneous, we can imagine that the first broadcasts would serve a significant portion of the nodes in the local network.

This is not modeled exactly in the implementation presented, but the 5 randomly selected nodes have been considered to be a sufficient depiction of this situation. A more sophisticated implementation would try to increase the probability of a node being randomly selected if it was previously selected by another node.

Metric	Description
Clustering Coefficient	As in [Montresor, 2018, p. 11].
Average Path Length	As in [Montresor, 2018, p. 13].
Degree	As the mean of [Montresor, 2018, p. 15].
Unprocessed Messages	Number of messages that are in the incoming messages queues.
Average Message Latency	Average of amount of time left before a message is read, over the messages in the incoming messages queues.
Partitions	Number of connected components of the overlay network.
Pollution	Percentage of dead nodes in alive nodes views.
Alive Agents	The total number of protocol agents that are running.

Table 3. Metrics collected.

1.5 Collected Metrics

Table 3 shows the metrics that have been collected for each step of the simulation. Each metric is computed over the alive nodes at each step of the simulation. Metrics referring to all nodes, indiscriminate of their status, have also been collected, for each of those listed in Table 3, except for *pollution* and *alive agents*, though they are less interesting.

2. EXPERIMENTAL RESULTS

An experiment run may thus be referenced as a tuple:

$$(P, G, n, V, S, \Delta T, d)$$

2.1 Clustering Coefficient

2.2 Average Path Length

2.3 Degree Distribution

2.4 Robustness to catastrophic failures

2.5 Self-Cleaning

2.6 Partitioned Runs

Experiment runs in which the number of partitions (connected components) was greater than 1 within the first 100 steps of the simulation; thus, before disaster.

Unsurprisingly, most of them took place in the Geo network, the only one actually exhibiting substantial delays in message exchanges. The non-Geo partitioned runs, though, exhibited similarities in that the only settings combination that produced partitioning was $(\text{Cyclon}, *, 1000, 20, 19, 4, *)$. In those six runs,

Although the number of unrecovered partitioned runs is skewed towards a greater number of Cyclon runs, we should not believe this is due to Cyclon's higher susceptibility to partitioning. In fact, Cyclon runs compose XXX% of the total runs performed.

3. CONCLUSIONS

REFERENCES

Mark Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Marteen van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. *Fifth ACM/IFIP/USENIX International Middleware Conference, Toronto, Canada*, 13(2), October 2004.

Alberto Montresor. *Distributed Algorithms. Epidemic Protocols: Beyond dissemination.* October 2018.

Spyros Voulgaris, Daniela Gavidia, and Maarten van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2), June 2005.