CS 425 Distributed Systems

MP 2 Report

Anish Shenoy (ashenoy3) & Omkar Lokhande (lokhand2)

For this MP, we had to write a program that implements a distributed group membership service. For failure detection and dissemination, we implement the gossip protocol, in which every machine picks K machines out of the N machines in its membership list and sends its own membership list to them. The receiving machine then merges the membership list with its own list, making changes when a new machine joins the group, a machine leaves the group voluntarilyor if a machine crashes.

A new machine first sends the introducer its single entry membership list. The introducer receives this list and merges it with its own list. However, it continues normal gossiping in which it randomly selects a machine from its list and sends it its membership list. The new machine keeps sending its list to the introducer till it receives a gossiped message from one of the members of the group, after which it too starts the normal gossip protocol.

Our implementation is scalable because every machine sends a fixed number of messages K every T_{gossip} seconds, or 1 gossip period. Thus, since K does not change with the size of the group, every machine sends and receives the same number of messages on average for each gossip period, regardless of the size of the group. The membership list, however grows linearly with the size of the group.

There are three main parameters used in this protocol: (i) the gossip time T_{gossip} - the time period for gossiping, (ii) the failure time T_{fail} – the time each machine waits to get a newer heartbeat sequence from the machine, failing which the machine is marked as having 'Failed', and (iii) the cleanup time T_{clean}– the additional time a machine waits before deleting a 'Failed' entry. This is done to avoid the 'ping-pong' effect as discussed in class.

MP1 was used to grep the log files at various stages of development to make sure the algorithm was working properly and also to get the results in the following studies.

The following settings and assumptions were used to produce the results that follow: $T_{gossip} = 500$ ms, $T_{fail} = 1500$ ms, $T_{clean} = 1500$ ms. These time values were chosen to have the lowest possible bandwidth usage and false detection rates that met the MP specifications.

1. **Marshalled message format**: The membership list consists of (i) an ID - containing the IP of the machine and joining timestamp, (ii) a heartbeat sequence, and (iii) last update time and (iv) status of the machine. Each row of the list represents a single machine, and the list is converted to a string (each row being delimited by a newline character) before being sent as a UDP packet.

2. 'Failure detection' is the deletion of the entry corresponding to a machine from another machine's membership list

Results

1. **Background bandwidth usage**: This was measured to be 1.993 kbps with 4 machines, and no membership list changes, over a period of 10 minutes. We did not drop any received messages, and observed an almost zero false detection rate, while meeting the detection speed requirements.

2. **Expected bandwidth usage when a node joins/leaves/fails**: The joining process has no significant overhead as explained earlier. In case of a failure, the propagation is achieved through the usual gossiping again and no extra gossips are sent to treat this scenario. In case of a voluntary leave, a machine marks its status as 'V' before sending out a normal gossip message. This is treated as a timeout by other machines and they wait for T_{clean} seconds before deleting it from their list. Again no additional messages were required for voluntary leave, over and above the normal gossip messages.

3. **False positive rates**: The results have been summarized in tables below.

Table 1: False Positive Rate Analysis

droprate	$n = 2$	$n = 4$
3%	No false positives for a long time	(0.060, 0.050, 0.040, 0.053, 0.043)
10%	(0.010, 0.005, 0.010, 0.010, 0.0025)	(0.065, 0.13, 0.105, 0.08, 0.065)
30%	(0.050, 0.050, 0.060, 0.045, 0.075)	(0.407, 0.493, 0.45, 0.433, 0.417)

Table 2: False Positive Rate Analysis

droprate	$n = 2$	$n = 4$
3%	No false positives for a long time	(μ, S) = (.049, .008), CI = (0.039,0.059)
10%	(μ, S) = (.0075, .0035), CI = (0.0031,0.0119)	(μ, S) = (0.089, 0.028), CI = (0.054, 0.123)
30%	(μ, S) = (0.056, 0.012), CI = (0.0412, 0.0708)	(μ, S) = (.44, 0.034), CI = (0.3978, 0.4822)

Discussion

In general it can be observed that the false positive rates are higher at a higher message drop rate which is to be expected. The data for $N = 2$ has been taken at $T_{fail} = 1000$ ms, whereas the data for $N = 4$ has been taken at $T_{fail} = 1500$ ms. The data at $N = 2$ was taken this way to ensure non-zero false positive cases in a reasonable amount of time. We observed that decreasing T_{fail} tends to increase the false positive rate as nodes are marked as failed and deleted at a faster rate. High variance in the data also indicates that there is a lot of randomness in the data or maybe the experiments should have been carried over a longer period of time. Interestingly despite waiting for a long period, we were unable to detect any false positives for $N = 2$ and a 3% drop rate. So, it can be concluded that at practical values of the message drop rates (like 3% in this case, practically it is lower than that), the system is not prone to false positive cases in case of a small number of nodes.