UNIVERSITY OF TRENTO

Project of Distributed Algorithms

IMPLEMENTATION OF CYCLON OVER THE AKKA FRAMEWORK

Luca Zamboni
XXXXXX

Luca Erculiani
XXXXXX

Abstract

The topic of this work is the implementation of Cyclon, a decentralized peer-to-peer protocol for gossiping over the Akka framework [URL AKKA REF]. The goal of Cyclon is to build a network that can resist against crash of a great part of its node without collapsing in a series of disconnected clusters. This document will explain first the theoretical basis of the protocol, then our implementation of it. The last part of this work will be focused on the statistical result of this project.

1 INTRODUCTION

The goal of this work is to implement a simulation of a peer-to-peer network running the Cyclon gossiping's protocol, for the project of Distributed Algorithms, by Alberto Montresor. The network is strongly decentralized and consist in a large number of peer clients and one or more tracker server, whose function is to allow new peers to connect into the network. Now will follow a list of fast description of the topics of every section of this document.

2 SYSTEM MODEL

The system that we want to model is a dinamic collection of distribuited nodes that wants to partecipate in a epidemic protocol. The number of node isn't fixed and can increase or decrease depending on peers who join and leave the protocol, or maybe crash. The communication between pairs of nodes needs that one of them knows the addres of the other one, and the channel is a best-effort type (potentially a lot of message omission).

2.1 SERVICE SPECIFICATION

Nodes has only a partial views of the network, and this wiew is dinamic. Each node periodically gossip with a random neighbour about its other neighbour. The main idea is that nodes continuously exchange information about other nodes, removing the old ones (so the most probable diaseppeared) and adding the new ones. Each node has a fixed number of neigbhbour and shuffle this with other nodes. For communicating with another node, a peer needs a neighbor descriptor of that node, consisting:

- the address of that node
- a timestamp information about the age of the descriptor
- more additional information, maybe needed by the upper software layer[CITAZIONE MONTRE]

When a new node want to enter in the protocol asks to a tracker server a random subset of peers and start the communication.

2.2 SKELETON OF THE ALGORITHM

Here is presented the structure of the code of an instance running Cyclon. The first algorithm shown is common to a bunch of protocol with the same purpose, like Newscast[CITAZIONE].

upon *inizialization* **do**
  $view \leftarrow$ descriptor(s) of nodes already in the system

repeat *every* Δ *time units*
  Process $q \leftarrow$ selectNeighbor($view$)
  $m \leftarrow$ prepareRequest($view, q$)
  send $\langle$REQUEST, $m, p\rangle$ **to** *q*

upon receive $\langle$*REQUEST*, $m, q\rangle$ **do**
  $m \leftarrow$ prepareReplay($view, q$)
  send $\langle$REPLY, $m', p\rangle$ **to** *q*
  $view \leftarrow$ merge($view, m, q$)

upon receive $\langle$*REPLY*, $m, q\rangle$ **do**
  $view \leftarrow$ merge($view, m, q$)

What make every protocol different is the behaviour of the three functions called in the previous pseudocode. In Cyclon these components acts in this way:

- selectNeighbor() selects the oldest neighobr in the view
- prepareRequest($view, q$) removes $t-1$ random descriptors from the view and return this subset plus a fresh local one
- prepareReply($view, q$) removes and return t freshest neighbors from the local view
- merge($view, m, q$) merges the local view and the one received, if there are duplicates keeps only the freshest. Remove itself and reinsert entries sent to q if space permits

3 ANALYSIS AND RESULTS

In this section we will discuss the results of simulations over the implemented protocol and some tests done under partarticular conditions.

3.1 ANALYSIS METHODS

The output of the program was used to build graphs, using the NetworkX libraries[CITAZIONE] in Python, where nodes are peers and edges are the neighbors of every peer. We useed undirected graphs except for the computations of in-degrees, where we obviously built a directed graph with direction peer $\rightarrow$ neigbor. The switch to a graph structure allowed a series of operations, like the count of the connected components (or clusters), needed to interpret the shape of the network.

We approximated a couple of opearation on the graph, in order to be able to execute it in a reasonable time. In particular the computation of the clustering coefficient was made using a function of NetworkX that make an approximation[LINK TO NETX CLUSTER]. The average path length was approximated computing the average path length of a number n of random couples of nodes.

3.2 STANDARD RUN

In these tests we executed the simulation without interfer during the run, so no crashes or communication failures were induced. This simulations were done bulding a network of 20.000 peers with 20 neighbors and 10 of them exchanged at every cycle. In every simulation the Cyclon graph is compared to a graph with the same number of nodes and the same number of edges randomly distributed.

The first simulation[FIG] compares the approximated average path length of the two graphs. We can see that after few cycles the avg becames comparable with the one of the random graph. This is the optimum result in a network fully decentralized without a hyerarchy.