Description of the Self-Chord P2P Application

Agostino Forestiero, Carlo Mastroianni
ICAR-CNR, Rende(CS), Italy
{forestiero,mastroianni}@icar.cnr.it

***Abstract*—This paper presents the implementation of the "Self-Chord" P2P system. Self-Chord inherits the ability of Chord-like structured systems for the construction and maintenance of an overlay of peers, but features enhanced functionalities deriving from ant-inspired algorithms, such as autonomy behavior, self-organization and capacity to adapt to a changing environment. Self-Chord has three main advantages with respect to classical structured P2P systems, especially in the context of Grid and Cloud Computing: (i) it is possible execute range queries efficiently; (ii) the load balance among peers is improved; (iii) maintenance load is reduced because resources are spontaneously redistributed when peers disconnect or reconnect to the systems, or when resources are added/modified. In this paper, we summarize the main functionalities of the Self-Chord application, and illustrate the configuration and parameter settings, with the help of an example.**

***Index Terms*—Bio-inspired systems, multi-agent systems, peer-to-peer, self-organization.**

I. Introduction

This paper presents and describes the implementation of Self-Chord. Self-Chord is a P2P system, previously introduced in [1] and [2] – and published on `http://self-chord.icar.cnr.it` – which inherits from Chord the ability to construct and maintain a structured ring of peers, but features enhanced functionalities achieved through the activity of ant-inspired mobile agents.

As opposed to Chord [3], Self-Chord decouples the naming of resources and peers, resulting in two sets of keys/indices that can have different cardinalities. Keys can be assigned to resources depending on the requirements of every specific application domain and on the desired granularity of resource categorization, with also the possibility to give them a semantic meaning. Moreover, Self-Chord does not place resource keys to specified hosts, as Chord does: this feature is actually unnecessary and limits the system flexibility. Conversely, Self-Chord focuses on the real objective, which is the reordering of keys over the ring, and their fair distribution to the peers.

The technique adopted in Self-Chord is based on statistical operations of mobile agents whose behavior is inspired by ant colonies [4]. Ants hop from peer to peer, pick/drop resource keys, and sort them on the underlying P2P structure. Sorting of keys ensures their fast discovery. Moreover, the keys of similar resources are assigned to the same or neighbor peers, which enables the efficient execution of range queries.

This kind of approach is inherited from recent proposals that aim to use bio-inspired algorithms and swarm intelligence techniques to enhance the self-organizing properties of distributed systems [5], [6], in particular of P2P networks.

After summarizing the key functionalities of Self-Chord, by means of a graphical example, this paper presents the Self-Chord application, developed at the ICAR-CNR institute.

II. Key Points of Self-Chord

Self-Chord uses the ring-shaped overlay of Chord, the forefather of structured P2P systems, to establish P2P interconnections, but distributes resource keys among peers using the operations of ant-inspired mobile agents. The principle of the mechanism is illustrated in Figure 1. In this example, keys are assigned integer values in the range [0..63], and the key space is toroidal, so that key 0 is the successor of key 63. Each peer computes its centroid as the key value that minimizes the sum of the distances between itself and the keys stored in the local area. In the example, an ant arrives at the peer with centroid C=24, and picks key 55, because its value is dissimilar to the centroid. Actually, the pick operation is subject to a Bernoulli trial, whose success probability is inversely proportional to the distance between key and centroid. The agent, carrying the picked key, exploits the long distance links of the underlying Chord structure, hops to a region of the ring where peers are supposed to have centroid values close to the key, and then tries to drop the key in this new region. Notice that these operations assume that keys and centroids are already sorted in the ring. The power of the ant algorithm is that, by making this assumption, the keys will actually be ordered, even starting from a completely disordered network. In stable condition, the sorting of keys guarantees their logarithmic discovery. The base principles for this behavior are the self-organizing nature and the positive feedback mechanism of ant algorithms.

Figure 2 shows a snapshot of Self-Chord in a stable condition, in which both keys and centroid are ordered. In this sample scenario with 16 peers, peer indexes are defined over 7 bits, while values of resource keys are between 0 and 63. At the interior of the ring, the figure specifies the indexes of the peers, whereas at the exterior it reports, for every peer, the keys stored by the peer (only the first three keys are shown for simplicity) and the peer centroid c. It can be noted that both the values of centroids and peer indexes are sorted in clockwise direction, but they are not related to one another. Indeed, different approaches are used to sort them: the peer indexes are sorted by the Chord management operations, whereas the resource keys are sorted by the self-organizing operations of the Self-Chord agents.

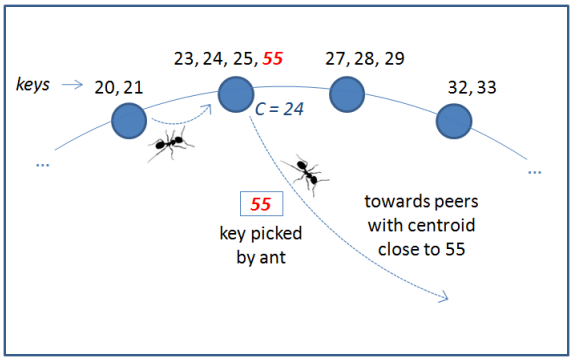

Fig. 1. Example of Self-Chord operation. A mobile agent arrives at a peer with centroid C=24, picks key 55, and then hops to a region of the ring where peers are supposed to have centroids close to 55.

Self-Chord features the following benefits with respect to Chord:

(i) In Self-Chord, peer indexes and resource keys are defined independently and there is no obligation to assign a key to a well specified peer. This feature enables the definition of "classes" of resources; a class being defined as a set of resources that share common characteristics, and are mapped to the same key value by

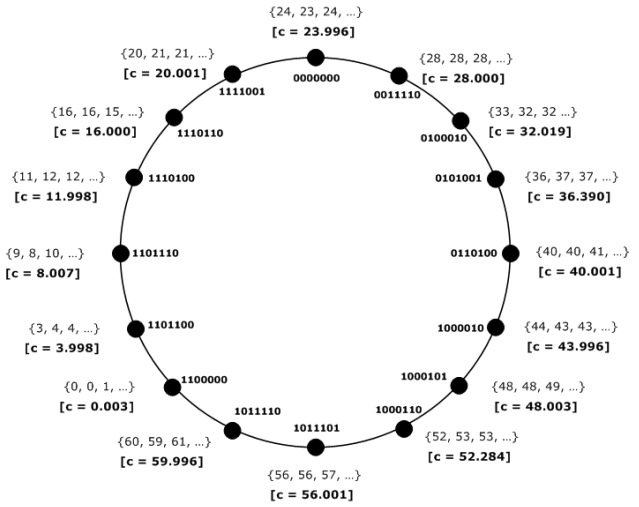

Fig. 2. Sample sorting of resource keys in the peers of Self-Chord. For each peer, its index, a number of stored keys and the centroid are reported.

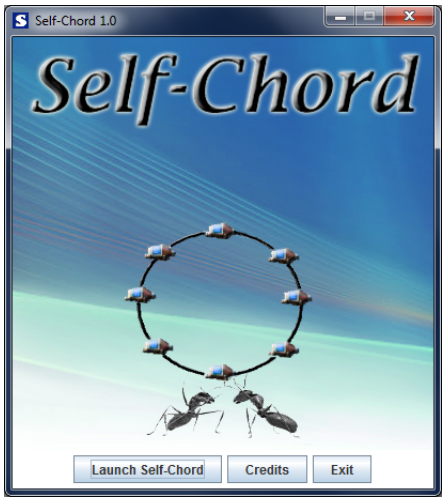

Fig. 3. The GUI Welcome Panel of Self-Chord.

a hash function. A user can issue "class" queries, i.e., explore the network to find a number of resources belonging to the same class and then select the most appropriate for his/her purpose.

(ii) Structured systems like Chord can produce imbalance problems depending on the location of peers and the statistical distribution of the values of resource keys. In Self-Chord, the keys are fairly distributed over the peers, irrespective of the location of peers and the distribution of key values, thus fostering a better balancing of storage responsibilities.

(iii) In Chord, appropriate operations are necessary when a peer joins the ring or when new resources are published: these resources must be immediately assigned to the peers whose indexes match the resource keys. These operations are not necessary in Self-Chord, because the mobile agents are always active and will spontaneously reorganize the keys. This assures scalability (keys are continuously reordered as the network grows) and robustness with respect to environmental changes.

III. The Self-Chord Application

The Self-Chord software was developed by the Distributed Systems Group of ICAR-CNR. The software, downloadable from `http://self-chord.icar.cnr.it`, provides an advanced graphical utility to let the user publish the resource metadata and search resources over the P2P network by name or by key. The current version of the Self-Chord software (both prototype and simulator) can be downloaded and used under the GNU licence (http://self-chord.icar.cnr.it). The available Self-Chord prototype does not need installation procedures, but it is sufficient to download and unzip the package, execute the file `compile.bat` and then run the file `run.bat`. The following software is required:

- Java 2 Standard Edition 5.0;
- log4j. (An Apache package included in the zip file);

A. Parameter Setting

The basic parameters of the Self-Chord software can be set in the file `selfchord.properties`. They are:

mode This parameter indicates the operation mode of agents. If the value is "log", the mode is logarithmic and the agents move the keys through the peer finger tables. If the parameter is "linear", the mode is linear, and the agents move the keys across adjacent peers. The logarithmic is the default mode and guarantees a faster reordering of keys. The linear mode is slower but assures a better load balancing.

k_t **and** k_l These are the parameters used in the take and leave probability functions (see [1] and [2]). They must be set to numerical values between 0 and 1.

range_of_keys This is the number of classes in which resources are categorized. Each resource is assigned a key with value between 0 and $range_of_keys - 1$ by a hash function.

sleep_time (ms) This is the time that an agent waits before moving to the next peer. The velocity of reordering can be scaled through this parameter.

p_gen This is the probability that a new peer generates an agent. The default is 1. The number of agents can be tuned through this parameter.

B. Executing Self-Chord

To run the Self-Chord software a double click on the file run.bat, included in downloaded zip file, is sufficient. Figure 3 shows the welcome panel of the prototype.

The main Self-Chord Window, shown in Figure 4, allows the user to manage the local peer, publish resources locally and search for remote resources. The `Connection Information` box can be used to create a new Self-Chord network or to join an existing overlay. This peer will be the contact node for the second peer. Each subsequent peer can use any existing peer as contact node. The TCP port can be chosen by the user to match the configuration of the local firewall. By choosing appropriate ports, it is possible to create multiple peers on the same host. This gives the possibility of testing a Self-Chord network with many peers by using just one or few hosts. To join the network, a new peer must specify the contact information (IP address and port) of any peer already connected to the network, its own contact information, and finally click the Join button. The first peer of the network must specify its IP address and TCP port and click the Create button.

The `Peer Information` box specifies the URL, the ID and the centroid of the current peer and the IDs of the two adjacent peers.

The `Insert Resources` box is used to insert the name of a new resource that must be published by this peer. The resource key is calculated by a hash function and shown in the same panel.

The `Resource` box shows information about the resource keys stored in the local peer. The content of this box is continuously updated by the agents that execute take and leave operations to reorder the keys on the network.

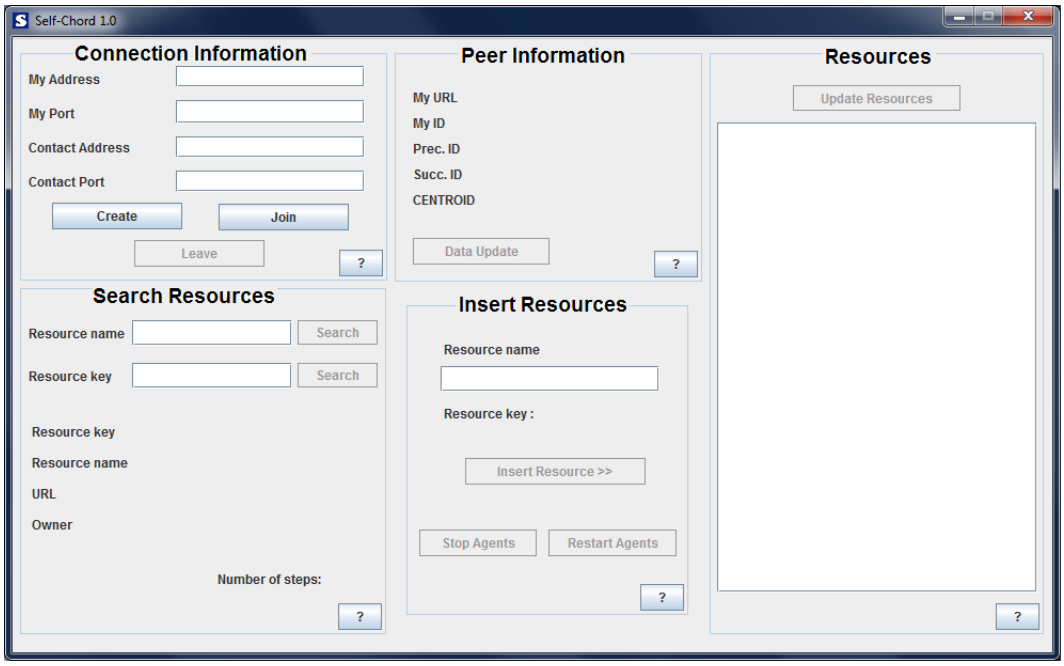

Fig. 4. The main Self-Chord Window.

The `Search Resources` box is used to specify a resource to discover on the network. The target resource can be specified by name or by key. The URL field specifies the address of the peer that stores the discovered key, whereas the Owner field indicates the address of the peer that stores the actual resource. The number of steps that was necessary to discover the resource key is also indicated. Thanks to the key sorting, it is logarithmic with respect to the number of peers connected to the network.

References

[1] A. Forestiero, C. Mastroianni, and M. Meo, "Self-Chord: a Bio-Inspired Algorithm for Structured P2P Systems," in *Proc. of the 9th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2009)*, Shanghai, China, May 2009.

[2] A. Forestiero, E. Leonardi, C. Mastroianni, and M. Meo, "Self-Chord: a Bio-Inspired P2P Framework for Self-Organizing Distributed Systems," *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1651–1664, October 2010.

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. of the Conference on Applications, technologies, architectures, and protocols for computer communications SIGCOMM'01*, San Diego, CA, USA, 2001.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. New York, NY, USA: Oxford University Press, 1999.

[5] D. C. Erdil, M. J. Lewis, and N. Abu-Ghazaleh, "An adaptive approach to information dissemination in self-organizing grids," in *Proc. of the International Conference on Autonomic and Autonomous Systems ICAS'06*, Silicon Valley, CA, USA, July 2005.

[6] S. Y. Ko, I. Gupta, and Y. Jo, "A new class of nature-inspired algorithms for self-adaptive peer-to-peer computing," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 3, no. 3, August 2008.