Hierarchy of the CONCORD Platform for P2P Computing

M. Itmi[1], A. Cardon[2], N. El Hami[3], A. El Hami[1], O. Jaumat[1]

1,2. LITIS, INSA de Rouen - St-Etienne du Rouvray. France,

3. ENSA - Kenitra. Maroc
[1](x.y@insa-rouen.fr), [2](cardalain@gmail.com), [3](norelislam@outlook.com)

Keywords: Avahi, P2P, distributed systems, JXTA, Zeroconf.

Abstract: This paper highlights the Concord project. It deals with the development of a distributed platform which main goal is to create a concord into a group of peers in order to distribute a computation that takes time. The protocol of agreement is based on a master-slave architecture.

1 Introduction

In brief, a peer-to-peer network is a network architecture in which no machine has a central role. Each machine must be able to interact with each other independently. In a pure peer-to-peer model, all machines are on the same level, there is no hierarchy between them. To communicate, they use protocols Network discovery and name resolution (if necessary) based on the multicast or broadcast. The broadcast is to send information over the network so that all machines see them and respond. Multicast is used to limit the sending of messages to a group of machines that can understand. For this reason, the use of multicast is preferred for most uses of peer-to-peer.

The research activity dealing with systems of systems leads to consider entities that are autonomous, distributed, proactive, etc. ([5]). Different applications are concerned: autonomous robots, traffic management in big cities, territory competitiveness, etc. In order to conduct our research studies and simulations in such applications we need one hardware and software platform that allows our experiments and tests. This work concerns the software basic platform under study. The objective is to have the ability of setting a P2P system that allows building our solution for distributed computing. This is also an attempt to reduce time processing when we use algorithms suited for distribution, for example in swarm optimization.

In the following, we discuss our choice of Zeroconf, Avahi and its implementation. Then we show the Concord approach which objective is to provide a method of distribution of an algorithm within a peer-to-peer (P2P) network.

2 Building a P2P network infrastructure

The JXTA ([6]) generalized P2P protocols can offer a solution for our platform. Such open platform and its JXSE implementation overcome the platform independence, the interoperability as well as other potential shortcomings of many P2P systems. It gives a set of high-level protocols for dialogue among peers, creating groups, building services and discovery of peers and services. We already conduct in the past some studies thanks to this platform. However JXTA and JXSE are no longer maintained for several years. This leads us to look for a different suite of protocols with the same objective of algorithm distribution. We need a solution with an environment that shows discussion activities and particularly a maintenance activity. Then our choice fell on Zeroconf ([8]) (a suite of protocols for the network discovery), Avahi ([1]) and its implementation. We called Concord the project of building our platform on top of Zeroconf. The Concord mechanism is designed to bring a way for peers to get themselves organized to work together.

In a large network, or when the task at hand requires the coordination of more than two machines, it is often useful to have a central peer helper to manage the effort. This type of peer-to-peer hybrid requires additional protocols to enable the promotion and coordination of peers.

Zeroconf is a suite of protocols to provide a "Zero Configuration" network. Its objective is to allow creating a fully functional LAN without requiring centralized configuration as DHCP and DNS services. In this way, any new machine onto the network will be instantly visible to others, and be able to connect without requiring any configuration.

Zeroconf is located within an IP network, and is thought to function in conjunction with existing services on such a network, such as an HTTP server, printing over IP Remote Control (SSH VNC) or file sharing.

3 Concord's general infrastructure

In Figure 1 we find a general schema of the project architecture. It is composed of three rectangles:

- The Peer Discovery corresponds to an external dependency.
- The Peer Concord rectangle corresponds to the project's main purpose, the Concord mechanism and protocol.
- The Computing Distribution rectangle is an additional layer to make computing distribution easier, the libconcord-compute library, a generic mechanism and implementation to distribute an algorithm computation.

The Concord's main goal is to create a concord into a group of peers in order to distribute a computation in an automated and mostly abstract fashion.

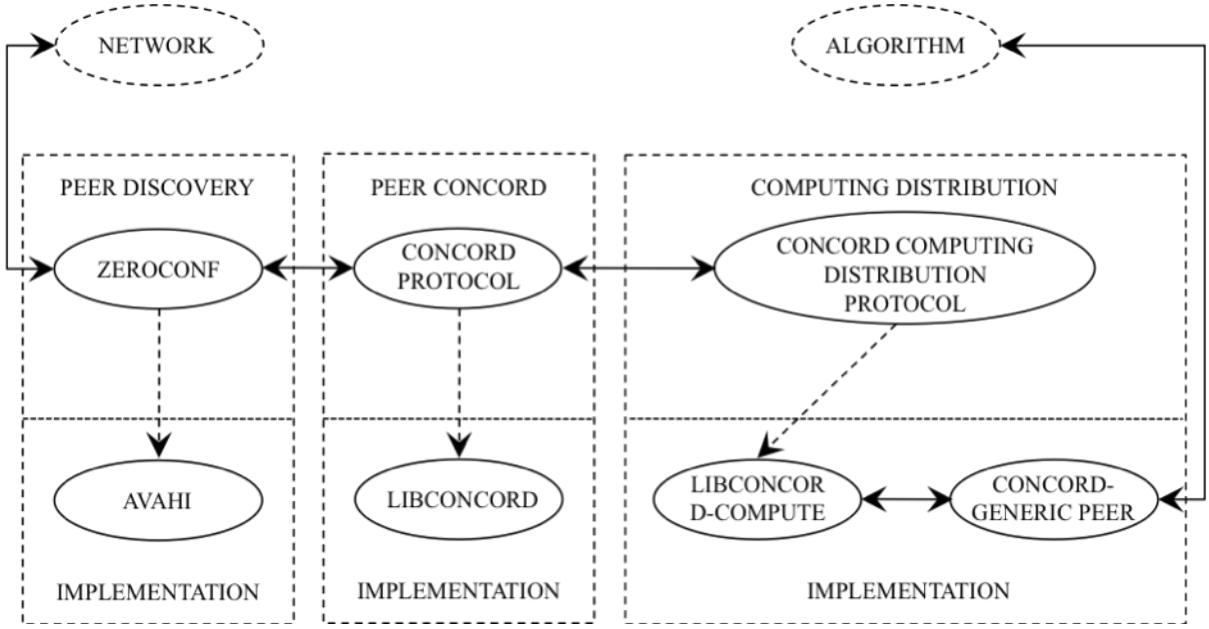

Figure 1. The project architecture

The concord is a conversation, using the Concord protocol, to define which of the peers will be the new master. Only masters advertise themselves on the network, and a peer alone defines itself as a master by default.

4 The Concord mechanism and protocol

The Concord mechanism is defined in two parts:

- The peer discovery
- The concord

The peer discovery is delagated to the Zeroconf set of protocols, specifically to the DNS-SD protocol. In the Concord mechanism, each peer is a service advertised through DNS-SD. When a peer discovers another one, it starts the concord.

The Concord protocol is a set of messages that peers must send to each other in order to find a master peer that will lead the group.

Messages are sent as uint64_t.

The protocol is message-based. Some events in the peer code will trigger a message. When the peer receive amessage, it must act accordingly, possibly sending another (or several other) message(s). The libconcord library is the reference implementation of the Concord protocol.

At this step, we try to organize a group of peers in a master-slave model. A peer will be assigned the role of master and will be in charge of monitoring the work done by his slaves. Figure 2 and Figure 3 show the graphs representing the protocol usage.

The master: The group's master is a peer who has no master. It publishes its presence on the network to indicate that it is available for an agreement. When several groups meet, the masters detect each other and trigger an agreement: their concord. When a peer enters the network, it then considers itself as the default master.

The slaves: All other peers are slaves. Some may also have slaves; however this does not mean their promotion as group masters.

The agreement: It is activated when several masters are found on the same network. Then they stop emitting their presence and agree on the new master to elect among them. The peer so designated will become the master of the group. The latter will announce its presence on the network. Then its new slaves relay their own slaves.

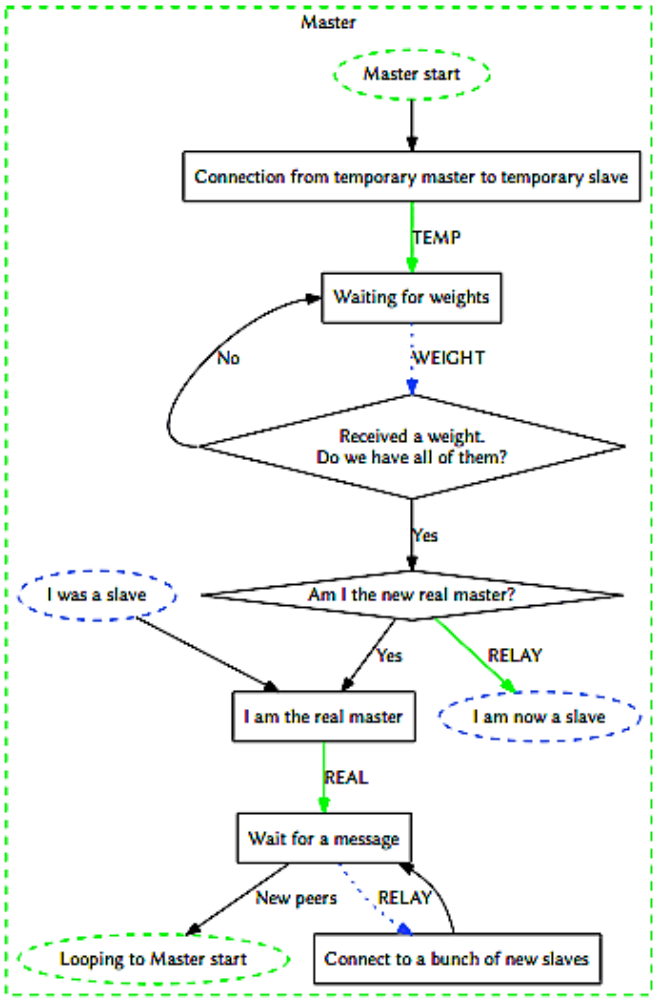

Figure 2. The master activity

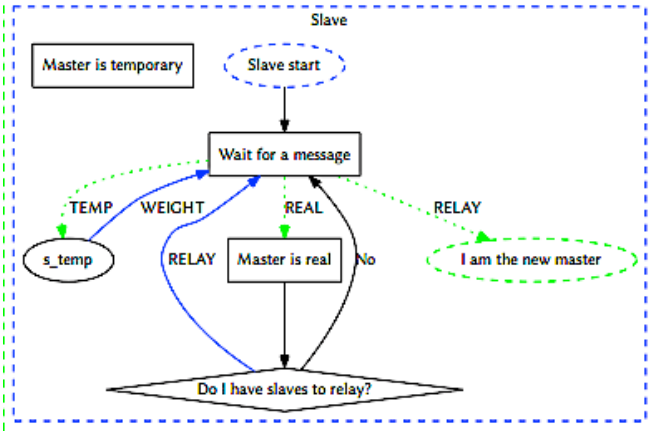

Figure 3. The slave activity

This action will determine the topology of the peer group. In this hierarchy of peers, one of several models can appear. They are forms of trees, with the master as a root: the chain, the binary tree, the balanced tree and the star.

The "simplest" organization is achieved when the master is directly connected to all slaves. This organization is very effective but poorly support scaling: all the slaves saturate the master. This can be avoided by limiting the number of slaves by peer.

5 Concord: a protocol agreement

Concord is a mechanism and protocol based on Zeroconf. Using the discovery permitted by the latter, it offers a solution to the founded peers allowing them to agree on a master. At the software level, it aims to provide an event "a new master was chosen" to peers.

During the design and testing of Concord, some heuristics were found inadequate to the success of the agreement. Mostly, it was forgotten peers, and more rarely messages not arriving for reasons it is not necessary to detail here. However, to compensate for these errors, the choices presented below were made.

Network Discovery: In the agreement, it is difficult to integrate new peers during the process. To avoid this case, the publication of the presence on the network and network discovery are stopped at the beginning of the agreement and recover since its end, but only on the new master. We thus find the complete state of the network at the end of agreement.

Messages agreement: The first architecture considered was to connect each slave to the master. The experiment has shown that this architecture does not allow messages to reach the master in the expected order. Ensuring that the master connects directly to the slaves, the messages now arrive in the expected order.

Disappearance of peers: The disappearance of peers is not directly managed by Concord. Indeed, Avahi provides an event when a peer disappears from the network, but it is not used. This management must be accompanied by checks on the status of the agreement, so as not to be considered as absent peer who disappeared under the agreement. It is however not strictly necessary to take this into account as usual network errors resulting from a loss of connection toward the peer will trigger the appropriate mechanisms.

Management of network errors: When a network error occurs, an even slave will simply begin to reissue its presence, that is to say, become a peer default master. In this way, the agreement will be re-launched itself regardless of the scenario.

Blocking messages: In Concord, sending a message and waiting for a direct response are blocking: no further treatment is possible by hand during this time. Loss calculation is low because Concord uses short messages (usually) spaced in time. So there is no long wait actually detrimental.

6 Computations experiments

The distribution mechanism is as follows:

To be distributed an algorithm must meet two requirements:

- Can be split in independent parts.
- As possible, each part should have as a few interactions with other ones.

Distribution calculation via Concord is to be managed by the program. It must give the master the necessary information, and the master must in turn provide information to its slaves. The direct use of Concord is desirable in some cases, but a large number of calculations share common characteristics that can be abstracted into a separate library.

We would like to establish a generic way to distribute algorithms that are independent. It appears that they are suitable for query-value method. This method is based on the ability of an algorithm to calculate a value from a query (the pair (query, value) is a given). These algorithms usually have a limited number of applications to be processed. By a set of relay requests and values, one can easily distribute the algorithm among the peer group.

A first test module of this architecture was an implementation of the calculation of π using the formula [2], [3]. The second test module was an implementation of the calculation of an integral. These examples are similar but help us to fix some errors. The main application will concern parallel and distributed simulation and optimization algorithms, such as Modified Particle Swarm optimization (MPSO) algorithm used as a heuristic optimization method to find the best distribution of effort needed in collaborative networks [7].

Concord is currently under development in C ([4]). A number of bugs are still present in the code. Simple use cases have been considered and tested, but more complex cases such as the disappearance of peers or a combination of several peer groups already engaged in a calculation that are theoretically supported. Tests of scaling are needed to estimate the robustness of the system on a large number of machines.

7 Conclusion

This paper highlights the current development of a peers concord mechanism and protocol based on Zeroconf discovery. It should allow the peers to agree on a master towards a master-slave architecture. Concord is our current distributed platform that responds to some kinds of scientific calculation that take a long time for calculation, and suited for parallelism and distribution. However it is still a work in progress to finalize.

8 Acknowledgement

The project is co-financed by the European Union with the European regional development fund (ERDF).

References

[1] Avahi website. URL http://www.avahi.org/.

[2] D.H. Bailey, P.B. Borwein, S. Plouffe, “On the Rapid Computation of Various Polylogarithmic Constants”, Mathematics of Computation, vol. 66, pp. 903-913, 1997.

[3] Fabrice Bellard. Pi Formulas, Algorithms and Computations. URL http://bellard.org/pi/.

[4] Quentin Glidic. Mécanisme d’entente de pairs dans un réseau. Rapport de stage master. LITIS Internal report. 2013

[5] Mhamed Itmi and Alain Cardon A. (2012). Autonomy and Control of Adaptive Systems of Systems, International Journal of Modeling, Simulation and Scientific Computing 3(1): 1240002. 21 pages. World Scientific Publishing Company. DOI: 10.1142/S1793962312400028.

[6] Jerome Verstrynge. Practical JXTA II - Cracking the P2P puzzle. Lulu Enterprises, Inc., second edition, July 2010.

[7] Norelislam Elhami, Rachid Ellaia, Mhamed Itim, Hybrid Evolutionary Optimization Algorithm MPSO-SA, International Journal of Simulation and Multidisciplinary Optimimisation (IJSMDO) ASMDO 2010 DOI : 10.1051=ijsmdo :2010004, Volume 4, pp 27- 32 (2010).

[8] Zero Configuration Networking. URL http://www.zeroconf.org/.