"SelfService" - A Theoretical Protocol for Autonomic Distribution of Services in P2P Communities

Fabrice Saffre and Håvard Rast Blok
Intelligent Systems Laboratory, BT Exact, Adastral park, Orion 1, pp 12, IP53RE, UK
fabrice.saffre@bt.com, havard.rastblok@bt.com

Abstract

In this paper, we present a theoretical protocol for autonomic distribution of services in a P2P environment, as well as the results of its simulated implementation. Our objective is to demonstrate that it is possible to obtain a distribution of the corresponding software modules that meets the requirement of the community in the absence of any centralized resource management. Instead, an acceptable balance between offer and demand is achieved via a simple "trial-and-error" mechanism, involving a simple, locally applied "reasoning" loop running independently on every peer.

1. Introduction

The peer-to-peer paradigm has recently received increased attention from research and commercial organisations that want to look beyond the early file sharing applications. Legal mass-market P2P products are emerging, links between grid and P2P computing are evident, and the IBM-led autonomic computing initiative [1] can be seen as a key enabler for adaptive and decentralised architectures.

To fulfil the variable needs of users of a modularised application we present a theoretical protocol for autonomic distribution of services in a P2P environment, and results of its simulated implementation. We make the assumption that most users require only a small subset of modules and that, once locally installed, a module can be remotely accessed by other peers. Fig. 1 shows a simplified example of the "ideal" architecture that should emerge in this hypothetical scenario.

Our main objective is to demonstrate that a simple set of local rules can preside to the self-organisation of such a community of peers where every user would experience good quality of service overall, though having installed only a small number of modules on his/her own machine.

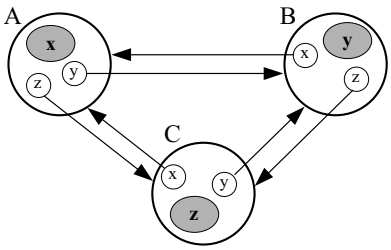

Figure 1: schematic representation of a self-organised, component sharing community of peers. A, B and C are individual machines, x, y and z are application modules.

2. Methodology

The *SelfService* protocol is based on a combination of "trial and error", local memory and broadcast requests. Every newcomer joining the community is assumed to have at least one module installed, and when it needs access to another, it must find a member hosting the corresponding service. If it does not already know a suitable host, a broadcast message is sent to all members of the community. If this fails, the peer has the option to install the module locally.

In the simulation of this protocol we have assumed the application to be comprised of 100 independent modules, with every peer subscribing to an average 10 of these. Each time-step, every member has a probability P of requiring use of one of its enlisted modules These can be found locally or via a directed or broadcast request. Every on-going job has a probability of 0.1 to be completed at the end of every time-step, which results in the average job duration being 10 time-steps. Peers have an average processing capacity of 5 simultaneous jobs (min. 1, max. 10, binomial distribution) independently of their origin. Finally, to emulate the unpredictable reliability of P2P communications, a remote job request only reaches the intended host with probability S.

Due to space constrains, we cannot go into further methodological details in the present paper. These are available on request to the interested reader.

Proceedings of the International Conference on Autonomic Computing (ICAC'04)
0-7695-2114-2/04 $20.00 © 2004 **IEEE**

3. Results

In the first simulated implementation of *SelfService*, we assumed a fault-proof messaging infrastructure where S = 1. The peers benefit from being able to distribute jobs between themselves, and, as shown on fig 2, the number of required modules which are locally installed nearly immediately reaches a plateau of < 14%. This is consistent for all tested values of P. The simulations also show that this load balancing prevents long queues since the peers which possess more processing power effectively relieve their less capable counterparts.

At steady state, only a small fraction of requests give rise to a broadcast message, even though most peers do rely on other members to provide at least some of the modules they require. Fig 2 also shows the evolution of the number of broadcasts over time, which always stabilises below 5% of all requests.

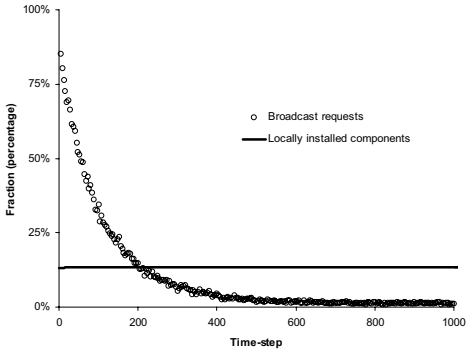

Figure 2: *SelfService* scenario with *S* = 1 and 1000 peers.

In the second scenario the messaging reliability S is set to 0.5 to simulate a more realistic environment. Results show that the faction of locally installed modules is only marginally increased (still < 15%).

The real impact of varying S is best measured by its effect on the number of jobs queuing in the entire system. Indeed, if S < 1, it may require several attempts before a request is successfully passed to the known provider of a service, which can delay the execution of the corresponding job. For P = 0.1 (i.e. when jobs are generated and completed at the same average rate), we find the cumulative length of all queues (i.e. summed over all peers) to be a linear function of S. But for P = 0.2, the signature becomes that of a power law (see fig. 3). This is actually an indication that *SelfService* can make a big difference in a relatively unreliable messaging environment, provided that the total load on the community is relatively high.

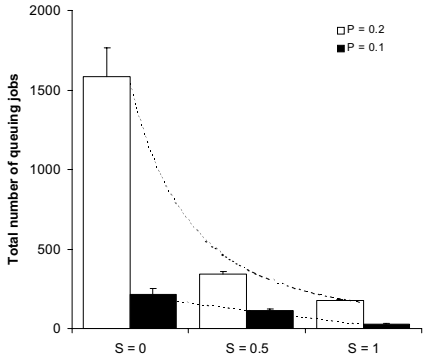

Figure 3: total number of queuing jobs after 255 time-steps, for three values of *S* and two values of *P*.

4. Conclusions and future work

This work demonstrates that it is theoretically possible to obtain adequate load balancing and good utilisation of resources by applying basic decision rules at a very low level.

The resulting "colony" of co-operating peers is not unlike those found in social insects: specialisation means that individuals only possess a small sub-set of all capabilities required for their survival and so heavily rely on each other. It is no coincidence if the so-called "swarm intelligence" paradigm [2] and other nature-inspired heuristics [3] have been applied to grid computing problems by other authors.

Future work will put more emphasis on the economics of distributed services for P2P communities, including relationship with software manufacturers and fairness considerations. We are tempted to argue that this is the equivalent of behavioural ecology for autonomic computing. Indeed, only a good understanding of the long-term dynamics of the interactions taking place between the many "species" involved will allow for the creation of a sustainable P2P ecosystem.

5. References

[1] Kephart, O. and Chess, D. M. , "The Vision of Autonomic Computing", *IEEE Computer*, 36(1), pp. 41-50, 2003.

[2] Babaoglu, O., Meling, H. and Montresor, A. , "Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems", In Proceedings of the 22nd International Conference on Distributed Computing Systems, Vienna, Austria, July 2002.

[3] Abraham A., Buyya R. and Nath, B. , "Nature's Heuristics for Scheduling Jobs on Computational Grids", The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), Cochin, India, December 14-16, 2000.