Self Tuning With Self Confidence*

Miguel Matos
University of Minho
mm@lsd.di.uminho.pt

José Pereira
University of Minho
jop@di.uminho.pt

Rui Oliveira
University of Minho
rco@di.uminho.pt

1. Introduction

Recent research on managing complex computing systems has focused on the autonomic computing vision: Systems should manage themselves according to an high level administrator's goal [5]. As an example, system components should monitor the enviroment and self-tune to meet quality of service expectations, without requiring manual intervention in the selection of concrete configuration options or in coordinating the reconfiguration process.

In a distributed system, the usual approach is to have several alternative protocols for each key distributed system function, each fit for a different performance tradeoff. Depending on the assessment of the environment and on the policy set by the administrator, the ideal protocol is chosen and configured. The result should be a simple feedback control loop that provides the desired self-tuning capability.

This approach has however several disadvantages. First, having multiple implementations, that may be seldom used, increases the complexity of the system and has implications in the reliability of the software. Second, the switchover mechanism is itself complex and requires distributed agreement, thus introducing a large overhead and often the need to "stop the world" while in progress. Also, by switching among a limited set of options, the system can cope only with a limited number of scenarios. And finally, pratical implementations often rely on a centralized coordinator component, which is in itself an obstacle to dependability. In short, all these issues combined make current systems fall short of the autonomic computing vision.

In this paper we show how two fundamental building blocks for dependable distributed systems – consensus and reliable multicast – can be extended to support autonomic principles. In fact, we point out that existing protocols [6, 1] support decentralized decisions and fine grained adaptation with a single algorithm thus surpassing the impairments pointed above. Furthermore, the correctness of the algorithm is always ensured in the given models thus guaranteying that it behaves properly even if the adaptive decisions taken are not adequate. Finally, we speculate on realizing actual systems based on this approach and on extending it to other distributed systems building blocks.

*This work was partially supported by project "P-SON: Probabilistically Structured Overlay Networks" (POS_C/EIA/60941/2004).

2. Case Study: Consensus

The consensus problem [8] abstracts agreement by a set of processes in a distributed system in the presence of faults. This has been shown to be the key issue in many building blocks for dependable distributed systems, such as the replicated state machine, atomic multicast, or view synchrony. Solving consensus efficiently has thus been the focus of a number of research efforts.

As an example, a key performance factor in an asynchronous message passing model is the way how votes are collected to form a quorum. In certain protocols, the votes are collected by a coordinator and the result broadcast after a decision has been reached [2], thus minimizing bandwidth used, while on others, votes are directly broadcast to all the participants which independently count them [7], thus reducing decision latency.

A particularly interesting tradeoff is achieved by the mutable consensus protocol [6]: The same protocol allows runs in which votes are collected by a single coordinator, runs in which votes are broadcast to all participants, as well as a number of message exchange patterns in between. In fact, different message exchange patterns can be induced by judiciously chosen delays, and thus the protocol, assuming only an asychronous system, is still correct regardless of the delay configuration.

3. Case Study: Reliable Multicast

The reliable dissemination of information to a very large number of destinations is at the core of distributed systems middleware, such as publish/subscribe and media streaming. The goal is to ensure that messages are reliably delivered according to some criteria, namely regarding how different receivers get the same data, while making efficient use of resources.

The usual approach is to build a spanning tree and then use it to relay messages [4]. Collection of acknowledgments and retransmissing to repair from losses, as required to meet reliability criteria, are also performed along the tree. The tree can also be optimized to make use of higher capacity links and nodes, thus making better use of available resources. The downside is that the cost incurred in setting up and maintaining the structure is prohibitive when reconfiguration is frequent, such as, when nodes continually enter and leave the system.

A radically different approach is provided by gossip-based or epidemic multicast [3]. Although the basic protocol is disarmingly simple – each node relays each message to a small random subset of neighbors – it can be shown that all destinations receive it at least once with high probability. Besides avoiding the cost of setting up and maintaining the spanning tree, the inherent load balancing and redundancy make it highly resilient to reconfiguration and message loss. On the other hand, it fails to take advantage of links and nodes with higher capacity as each node is chosen randomly and thus over time each node will process approximately the same number of messages. Furthermore, as each message is transmitted multiple times this approach leads to a large bandwidth consumption.

It has however been shown that by judiciously scheduling the transmission of message payload in epidemic multicast, it is possible to combine the advantages of both approaches [1]. In detail, when relaying messages each node may omit the payload and transmit it later only upon request. This ensures the desirable resilience properties of epidemic multicast. However, depending on the policy used to schedule payload transmission, it can be observed that redundant retranmissson of payload can be entirely avoided without negative impact in end-to-end latency and that some nodes and links contribute with much higher probability of payload transmission: The result is thus a probabilistic structure that emerges from the operation of the gossip protocol itself (based on efficiency criteria) instead of being imposed by construction.

4. Discussion

The protocols discussed in previous sections [6, 1] share a number of desirable features for self-tuning systems. In detail:

Single Protocol. Proof of correctness and correct implementation of the protocol needs to be done only once, since a single protocol is able to deliver a wide range of performance tradeoffs.

Out-of-Model Tuning. Tuning for performance is done by adjusting parameters which have been abstracted in the system model used for designing for correctness, resp. delays in an asynchronous system and packet scheduling in an epidemic. This separation of concerns can easily be enforced even in implementations by validating the output of the policy module.

Local Adaptation. Since protocol corretness is oblivious to tuning, the protocol will still be correct if each participant independently adjusts local parameters with no global coordination whatsoever. This reduces overhead and obviates the need for a switching mechanism.

Progressive Adaptation. Even if there are only a few configurations, tradeoffs in-between can still be achived if different amount of nodes choose different configurations or if nodes alternate between different configurations.

Low Adaptation Latency. Since adaptation is local and progressive, the feedback control loop has low latency and fine grained control over the system, and is thus much more likely to achieve stable configurations.

There are however some outstanding challenges in realizing a dependable distributed autonomic system using the suggested approach. The first is to derive the performance of the proposed protocols in a wide range of environments and configurations. The second is to determine to what extent can effective global performance be achieved based on strictly local tuning decisions. Finally, whether the proposed approach is applicable to other distributed systems problems.

References

[1] N. Carvalho, J. Pereira, R. Oliveira, and L. Rodrigues. Emergent structure in unstructured epidemic multicast. In *IEEE/IFIP Intl. Conf. Dependable Systems and Networks (DSN)*, 2007.

[2] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.

[3] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie. From epidemics to distributed computing. *Computer*, May 2004.

[4] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Trans. Networking*, 5(6), Dec. 1997.

[5] J. Kephart and D. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[6] J. Pereira and R. Oliveira. The mutable consensus protocol. In *IEEE Intl. Symp. Reliable Distributed Systems (SRDS)*, 2004.

[7] A. Schiper. Early consensus in an asynchronous system with a weak failure detector. *Distrib. Comput.*, 10(3):149–157, 1997.

[8] J. Turek and D. Shasha. The many faces of consensus in distributed systems. *Computer*, pages 8–17, 1992.