Editorial

Issue on Distributed and Self-organising Systems

Matthias Werner

I guess, everybody who works in the area of distributed computer systems knows Lamport's famous definition of distributed systems [1]:

> *A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.*

Well, this quote is aged more than 20 years. There may be cases, where this definition is still valid. But distributed systems are all around, and failure transparency is an objective most designs of distributed systems follow (or, at least, try). Moreover, gaining one or more kinds of transparency is one of the core problems in distributed system design. Beside failure transparency, there are location transparency, name transparency, access transparency, migration transparency, etc. With other words, the designer aims to hide the fact, that the user has to deal with something distributed at all. This is done because distribution is a quite confusing thing. The human structures of thinking are not especially suited for distribution. Of course, we *can* handle it. But it is a well-known fact that programming of a message passing system is much more error-prone than programming a shared memory system, which is in turn more error-prone than programming a non-concurrent system. Thus, frequently, the illusion is constituted that the system the user is dealing with is a classical monolithic system. A lot of approaches and abstractions have been invented to support this illusion. Many of them are so common that we take them for granted, not even noting their existence (unless they fail, acknowledging Lamport's definition), such as RPC, DNS, you name it.

Matthias Werner
TU Chemnitz
Operating Systems Group
E-mail: mwerner@cs.tu-chemnitz.de

However, mock monolithic systems should not be the last word. There are applications where distribution is inherent, i.e. it appears at the level of the application's semantic. In this case, there is still a need for new abstractions, approaches, and mechanisms, which help the programmer to deal with distribution in a better way. Also, there exist approaches, where the programmer does not have to care about the distribution explicitly, since the system cares itself. Such systems exhibit properties that are usually called self-$\star$ properties, where $\star$ is a wildcard for terms like "managing", "controlling", "healing", "organising" and so on, even "self-adaptive" systems have been spotted.[1] Distributed systems with self-$\star$ properties can hide distribution by a wide range of measures, starting from hiding distribution by use of global abstract objectives (which will be broken down and distributed by the system's intelligence), and ranging to hiding the system objectives itself, i.e. the semantic of the overall system. The last is true, e.g., in some cases of systems with swarm or flock intelligence, where the system behavior seems to *emerge* in some magic way.

In this very tension between approaches to camouflage distributed systems as monolithic systems and approaches finding new ways to deal with distribution in general I see the contributions of this issue's papers. I am looking forward to read interesting articles on this domain.

I am looking forward?

That is true. At this very point I have to allude to the fact, that this editorial differs a bit from most other editorials in scientific journals. Whereas other editorials usually serve as introductions to articles in the issue,

[1] In my opinion, "self-adaptive" might be a pleonasm, since "adaptive" implies already a self-reference.

compare their views and approaches, and discusse the relation, you will not find anything similar in this editorial. The reason is simple: At the time I am writing these lines I have no idea how the final version of this issue will look like. The reason in turn is due to the publishing scheme of the ESS Journal. We take advantage of the greater flexibility and agility an e-journal can provide in comparison to the traditional print journal: Instead of imitating the classical creation process of a journal with fix issues we use the "sliding issue" model. Each issue of the ESS Journal starts with an editorial that also serves as a kind of call for contributions. Any contribution that is reviewed and accepted will be published as soon as the final version is received by the editorial board. In this way, an issue is "growing" until the editors decide to close it. With other words, you can already read articles while other articles are still in the process of preparation. In addition, there may exist other topic-specific issues in parallel, where the very same is true.

Sliding issues provide a number of advantages: Firstly, the time to publish a single article is significantly shorter than in traditional journals. (Well, this is an advantage almost all e-journals share.) Secondly, several topic-specific issues can co-exist, what makes it easier to react to a hot or upcoming topic. Thirdly, direct reactions and answer articles to a published article can be found in the very same issue, as well as reactions to the reactions by the original authors. In this way, productive discussions might come up: the best, what can happens to a scientific journal.

From the meta level perspective, sliding issues are another way to deal with distribution we find in our daily life. They became possible due the (meanwhile not-so-new) technology. And while the technology helps to handle the distribution one can find in reality, I hope for fresh ideas to handle distribution within the technology itself.

References

1. L. Lamport. Distribution. E-mail correspondence, May 28 1987. Message-Id: `<8705281923.AA09105@jumbo.dec.com>`.