Scientific Report of Visit to the Distributed Programming Laboratory of École Polytechnique Fédérale de Lausanne

João Barreto
INESC-ID/IST
Distributed Systems Group
Rua Alves Redol N 9, 1000-029 Lisboa
joao.barreto@inesc-id.pt

Contents

1 Purpose of Visit

The evolution of the computational power and memory capacity of mobile devices, combined with their increasing portability, is creating computers that are more and more suited to support the concept of ubiquitous computating. As a result, users are progressively using mobile devices, such as handheld or palmtop PCs, not only to perform many of the tasks that, in the past, required a desktop PC, but also to support innovative ways of working that are now possible. At the same time, novel wireless communication technologies have provided these portable devices with the ability to easily interact with other devices through wireless network links.

The visit aimed at exploring two essential research directions entailed by ubituitous computing environments. First of all, support for loosely-coupled interaction using the publish-subscribe communication paradigm. Secondly, optimistic consistency protocols in ubiquitous computing environments, as a means of providing highly available and decentralized access to shared objects.

Within these two main directions, the visit focused on the following three particular issues:

1.1 Effective Distributed Subtyping for Loosely-Coupled Interaction using Publish-Subscribe Models.

Testing if a type of an object is a subtype of another type is traditionally called a "subtyping" test or a "type inclusion" test. Usually, object-oriented languages implement this test via native language keywords or via specific methods. For instance, Java and C# use respectively the *instanceof* and *is* keywords, whereas Smalltalk uses the *isKindOf:* method.A common assumption made in object-oriented languages is that the code of the type of the object must be available when performing a subtyping test. This assumption has been carried over in all object-oriented distributed systems we know of.

Relying on the code of the type of the object in order to accomplish any subtyping test involving the object might look reasonable. Indeed, in object-oriented languages, the object is somehow responsible for storing data whereas its type for storing its behavior. However, this perspective is restrictive in dynamic distributed environments where objects are remotely exchanged, which is the case in the increasingly popular event-based (also called publish/subscribe) systems. In such systems, whenever an object is received, a type inclusion test has to be performed. This is key for a remote process to throw away objects it is not interested in. Unfortunately, if the received object is of an unknown type, its code has to be downloaded before the subtyping test can be performed. This is particularly cumbersome and inefficient if the probability that the unknown object is of interest is low. Instead, it makes sense to incorporate some (encoded) type information into the object.

Efficiently encoding type hierarchies without heavily impacting remote communication is not a trivial task. Ideally, the encoding should not depend on the size of the hierarchy nor introduce any global reconfiguration if new types are added at runtime. To avoid reconfiguration, the encoding of the core type hierarchy (the set of types present at the initialization of the system) should remain the same throughout the lifetime of the system.

Moreover, it should be possible for a process to perform subtyping tests against a new type added at runtime, and consequently be able to receive and correctly treat objects of such type. Existing encoding schemes are all devised for a centralized setting and cannot be distributed as such, and are furthermore hardly adaptable to a distributed environment.

1.2 Efficient update commitment algorithms.

In the context of ubiquitous computing, as in any distributed environments, data replication is a fundamental mechanism, for a number of highly desirable properties. It enhances performance and scalability by enabling local data replicas to be accessed rather than a centralized physical data source, potentially located on a remote server. Replication improves fault tolerance if replicas are ensured to be kept consistent: should one replica be lost, the contents of any other consistent replica can be used to recover the lost data. Finally, it increases availability as applications may continue to access their local replicas even in case of failure or network inaccessibility of other replicas.

In particular, optimistic replication protocols are of extreme importance in mobile and other loosely-coupled networks that will expectedly prevail in ubiquitous computing environments. The nature of such networks calls for decentralized replication protocols that are able to provide highly available full access to shared objects. Such requirement is accomplished by optimistic replication strategies, which, in contrast to their pessimistic counterparts, enable updates to be issued at any one replica regardless of the availability of other replicas. The replication protocol is then responsible for disseminating such updates among the remaining replicas.

As a trade-off, the issue of consistency of an optimistically replicated system is problematic. Since replicas are allowed to be updated at any time and circumstance, updates may conflict if issued concurrently at distinct replicas. Some optimistic replication protocols ensure that, from such a possibly inconsistent *tentative* state, replicas evolve towards an eventual consistent *stable* state. For this end, a distributed consensus algorithm is executed so as to reach an agreement on a common order in which tentative updates should be committed. Such consistency approach can be regarded as a hybrid strategy: at the immediate update time, only weak consistency guarantees are provided upon the tentative replica values; on an eventual point in the future, however, a strongly consistent stable value is attained.

There are many scenarios where users and applications, in order to benefit from high availability, are willing to work with temporarily tentative data, provided that a commitment agreement regarding such data will eventually be reached. Consider, for instance, a worker carrying a laptop that becomes disconnected from his corporate fixed network file server after leaving his office and heading to a work meeting. If, by chance, he comes to think of some changes to a report that is currently replicated at his laptop cache, he might expect to be able to modify it at that moment, even if tentatively.

Furthermore, consider that the mobile worker happens to meet the remaining of his team colleagues while away from the office who, in an ad-hoc fashion, establish a short term work group to review the shared report. If every worker is carrying some mobile device with a cached replica of the report, a collaborative activity might be carried if each worker is able to perform updates to its local replica and have such updates propagated to the remaining replicas. A set of causally related tentative updates will result from such activity. Hopefully, if no update is concurrently issued from outside the group, such tentative work will be eventually committed.

Hence, the high availability provided by an optimistic replication strategy is especially interesting in such scenarios as the previous ones. However, the usefulness of one such approach strongly depends on the ability of the underlying replication protocol to efficiently achieve a commitment decision concerning the tentatively issued data. Users are typically not inclined towards working on tentative data unless they trust the protocol to rapidly achieve a strongly consistent commitment decision regarding such data.

More effective alternatives to the primary commit scheme ought to be studied, in order to devise solutions that impose lower commitment delays while ensuring high availability.

1.3 Exploitation of ubiquitous resources for update commitment.

Existing results suggest that the inherent mobility of users on an ubiquitous environment, comprised of radio-equipped mobile or stationary devices, should be an efficient mechanism for the dissemination of consistency information, complementing the direct interactions between replica holders. Nevertheless, the memory requirements imposed to such devices by the consistency protocol must be sufficiently low so as to be tolerated by them.

This means that a solution to be devised must allow the existence of nodes that will only be able to hold, possibly for short periods of time, simple consistency data structures such as replica identifiers or timestamps. Such novel consistency protocol should be evaluated so as to infer its benefits in comparison to conventional optimistic consistency protocols.

2 Work carried out during visit

2.1 Effective Distributed Subtyping for Loosely-Coupled Interaction using Publish-Subscribe Models

Regarding objective 1.1, the work carried out along with Sébastien Baehni and Rachid Guerraoui devised the first algorithm to encode a type hierarchy in a dynamic distributed environment. The algorithm, which we denote by DST, supports multiple subtyping as well as concurrent additions of new types at runtime.

In particular, my contribution during the visit included the following aspects:

- Optimizations of the type encoding length and performance;
- Optimization of event serialization and deserialization performance;
- Implementation of remote transference of events using the devised type information encoding and associated subtyping algorithm;
- Experimental evaluation of the algorithm's implementation.
- Co-writing of the paper and general reviewing.

2.2 Optimistic consistency protocols

The work carried out on the subject of optimistic consistency protocols (objectives 1.2 and 1.3) consisted of the following:

- Talk given to the Laboratoire de Programation Distribuée and discussion about the ongoing work on this subject;
- Research of related work.

3 Main results obtained

As described in the paper included in Appendix A, we believe that DST could be incorporated into modern (typed) event-based (publish/subscribe) distributed systems, e.g., through their (de)serialization mechanism. This would increase the capabilities of those systems (in terms of bandwidth and CPU processing) with the ability to add (and express interests in) new types at run time, while efficiently performing distributed subtyping tests without global reconfiguration.

We devised a complete set of Java APIs that implement our DST algorithm as well as a new serialization (deserialization) mechanism based on our approach.[1] The serialization scheme we introduce encodes an object sent over the wire with limited information, that is still sufficient to perform subtyping tests without having to download the code of its type nor to deserialize the object.

Our performance measurements show that DST is more efficient than a standard code downloading approach as well as a naive approach using

[1]Both are available at *http://lpdwww.epfl.ch/sbaehni/dst.tgz*

strings to encode the type information both in terms of length of message sent in the network and in processing time for performing subtyping tests over an object of no interest. Moreover, DST is comparable, in terms of encoding length, to the best currently known centralized subtyping algorithm.

4 Future Collaboration

The visit described in this report will be extended for an additional period of six weeks. The purposes of such extension are to continue the work already carried out during the MINEMA Exchange visit, with the same research objectives in mind.

5 Projected publications resulting from the grant

A scientific paper has resulted from the grant period (included in Appendix A), which has been submitted to the 19th International Symposium on Distributed Computing (DISC 2005) and is currently under review. The title of the paper and its authors are as follows:

Distributed Subtyping
Sébastien Baehni, João Barreto and Rachid Guerraoui, 2005.