A framework for adaptive applications

Extended abstract

Christer Åhlund¹, Mario Toffia, Stefan Elf, Håkan Nordmark
{christer, solo, self, hakann}@cdt.luth.se
Luleå University of Technology / Centre for Distance-spanning Technology
Skería 3, SE-931 87 Skellefteå, Sweden

Abstract:

The adaptive applications framework presented here aims at enabling hosts in a network to share objects such as applications or data. The framework consists of a set of components needed at each host in the network. They are the repository, the monitor, and the mobile agents.

The repository acts as a database for storage of objects and object semantics. An adaptive application, which itself is an object, first tries to access an object from the local repository. If it is not found, the search is extended to the repositories of co-operating hosts by the use of mobile agents. An object or a proxy object is returned to the requester. The application can be executed on the local host or on the remote host depending on predefined network and host load conditions.

The framework that has been prototyped will form a base for further work, which includes exploring the use of semantic modelling for pre-caching and intelligent download of resources.

Keywords: adaptive application, semantic model, repository, mobile agent

¹ The paper will be presented by Christer Åhlund.
1 Introduction

When applications or resources become distributed several problems appear. Among these are the difficulties in locating the application, to determine if it has the desired properties, to take into account the network characteristics, and in what manner to bring the application to the user.

This rises several issues e.g. regarding how we can let the application gather knowledge about its own performance and in what way we can build effective “search engines” not only for users, but also for applications and software developers.

We cannot know in advance whether the network conditions are such that it is most favourable to execute the application remotely and exporting the interface to the user, rather than first download the whole application and execute it locally. This line of reasoning applies equally well to data. The decision will then be whether to treat data locally or remotely.

There are a number of technologies for media- and software-on-demand. We find the use of network characteristics in combination with the use of resource semantics particularly interesting. The need for flexibility and for fully understanding the mechanisms that are needed, is why we have chosen to design this framework. A framework such as the one described here will also aid the developer in creating adaptive applications.

The remainder of this paper is organised as follows. Section 2 describes the design goals and the design of the framework. In section 3 we conclude and propose future work. Section 4 contains references.

2 Design

2.1 Design goals

The work described in this paper presents a framework that addresses the issues of locating and executing adaptive applications or accessing data objects. The application may consist of several objects or just one object that performs a specific function. When the network load is deemed high, applications should favour remote execution through a proxy object, rather than downloading the whole application.

The primary design goal is to obtain a flexible framework upon which adaptive applications could be developed. The design should allow for easy extension towards more advanced methods of finding objects, performing downloads, and for executing applications.

2.2 Application and data repository

The desired object, an adaptive application or data, can be located on any of the participating hosts. A semantic description is used to find an application. To the client the location of an application is transparent. Presently, a simple predetermined semantic description regarding the supported functionality is used. This is stored in the repository, followed by the Java class code and the initial state of the application.
An application must register itself with the repository on the local host. The framework supports this process. Among the attributes being registered are the application’s name, semantic, class code, state, unique identifier, and registration lifetime.

When applications copied as a result of a client request they will also be stored in the local repository of the new host. This is done on account of there being some likelihood that the application will be requested more than once. When the lifetime expires, the application will be removed from the repository, and thus made unavailable for further use.

The repository serves as a persistent storage for applications. A client can store an application locally. The class code and the current state of the application will be stored together with a name and semantic description.

2.3 Locating resources

As a client requests an application a search will primarily be made in the local repository. If a matching application is found it will be returned to the client. If a match is not found in the local repository, a request to all repositories is made using mobile agents [1]. Each mobile agent will query a repository and respond with the matches found, possibly with a degree of uncertainty depending on whether the semantic description matches exactly or not. The requesting client must then select the most appropriate application.

2.4 Executing applications

For the client to be able to execute the application, either the application itself or a proxy object is delivered. Predefined measures of the network load and the CPU load of the involved hosts are monitored. If it is suitable, the application will be copied to the client host prior to execution. If not, the application will execute on the host upon which it was found. For this purpose, there is a monitor observing the CPU-load on each host. It also measures the network load between the local and the remote hosts by taking snapshots of the round trip times.

The application will be executed locally on the client host or on a remote host, based on the client’s desire, and/or the advice from the application itself based on the current network capacity and the process utilisation on the involved hosts.

3 Conclusions and future work

An adaptive applications framework has been designed and a prototype has been implemented. The framework will allow us to develop our ideas regarding timely and intelligent download of applications, smart location of applications and resources etc. It is built on three core services, which are needed on every participating host; repository, monitor, and mobile agent.

The repository is used by applications and data resources to register themselves in the framework, and by clients to find an application. The agents will support a client in its quest for an application or a data resource on different hosts in a distributed system. The monitor observes the current load of participating hosts and takes a coarse measure, by using the round trip time, of the load on
the network. This information will help the requesting client to decide how and where the application should be executed.

The most obvious further plan is to extend the presently very simple semantic function to something more intelligent. There are two paths that we can follow and both should be considered. On one hand we can create a measure by defining a semantic for adaptive applications [2]. The administrator of an adaptive applications network can decide on which semantic properties are the most obvious, valuable, and descriptive for a given application.

On the other hand the construction and maintenance of such a predefined semantic may prove to be a tedious task. It should be possible to develop algorithms that describe semantic relationships between applications and object. Such work has been done e.g. regarding disconnected mobile work [3]. The framework would take advantage of this model in order for it to draw conclusions regarding crucial elements of the applications and of the network prior to downloading the actual application. Such conclusions could be based on current measurements and collected historical data alike.

We cannot wait a very long time before the application starts to execute. Instead, critical components must be downloaded first, allowing the first user interactions to take place. In the meantime, and in the background, the rest of the application will download.

It will of course be very interesting to compare our framework to other.

4 References

