MIGRANTS
Self-moving Object Clusters

SIGOPS'96 Contribution
Extended Abstract

Eric Jul
DIKU
University of Copenhagen
Copenhagen, DENMARK
eric@diku.dk

1 Extended Abstract

In a world connected by an *Information Superhighway* there will be many smaller highways, roads, and even paths to users. Users will roam around and touch upon the access paths at many different point with varying path bandwidths. Upon each access, the user will expect to see data, objects, and computations essential to his or her needs.

We propose to provide such access using *Migrant* which, in a nutshell (excuse the pun), encapsulates small units of data, objects, and computation. The central idea behind migrants is that of carving out a subset of objects in an object based system where the subset represents some small part of one (or more) applications and letting this subject, called a *migrant* chase down the user wherever the user accesses the WorldWide net.

A migrant can be considered a small agent containing essential data (and computations) which furthermore are aware of the network and are willing to travel as to find the user. For example, a simple migrant could contain a mail message plus information on where the migrant might rendez-vous with the user. On the way, the migrant may decide to replicate itself, so as to better show up wherever the user connects to the network next.

Migrants must be application defined because they need to contain a suitable summary of some subset of the application. The application programmer defines *distilling* functions as to create migrants. The idea is that migrants are mini-replicas and are themselves replicable—an ability which is useful when trying to chase down the users. For example, a migrant may replicate itself by being sent to multiple plausible locations of the user.

Migrants can also produce migrants of their own, e.g., a mail message migrant can produce a smaller migrant (max. 160 characters) that can be transmitted via the Short Message Service (SMS) of the GSM cellular system.

Migrants can also be created at the user end, e.g., a user can create a migrant using a cellular phone by sending a SMS message. Upon receipt the migrant expands itself into a

real task, e.g., sending a real mail message, accessing a database, requesting the creation of a file migrant (e.g., to order retrieval of a file), etc.

At the implementation level, migrants can be considered a form of active messages in that they contain the necessary code to identify and hook up with the application or user (or other migrant) that they are looking for. The underlying support layer is to provide transportation, location information, and location hints that the migrant can use in chasing down its destination. Thus the underlying transport layer must make parts of its information available to migrants. In many cases, a hunting migrants will not have to do much chasing because the underlying system has up-to-date location information and network connections to the destination. However, by exposing location information, migrants can then apply their own policy concerning the chasing to be done. This can include the further replication of the migrant and the initiation of further communication over the available communication lines.

We are developing language constructs for specifying migrants within the object-based system Emerald. A major problem with semi-automatically defined migrants is to ensure that enough other Emerald objects are included in the migrant for it to be able to function. In this sense, migrants are akin to mobile objects in Emerald where sub-data structures are moved along with their containing objects. However, for migrants it is often necessary to provide proxies or dummy stubs for references outside the migrants. A part of the project is to study the necessary types of references required in this environment. The original single type of object reference in Emerald does not suffice. Migrants also need handlers (upcalls) to enable them to react when they arrive at a new location, when a potential destination has been found, etc.

Migrants also need to match themselves against potential destinations. The exact matching used is still an open question. One solution is to rely on the underlying language reference mechanism (so matching is reduced to using the identity operator). This, however, places the real matching burden on the underlying system which is a poor chose for several reasons. The matching is fixed and thus inflexible; in some cases, the migrants will want to chose their own policies. We are also considering using the Emerald type system to perform part of the matching in a more secure manner by employing the Emerald **restrict** mechanism. One advantage of using the type system for matching is that the migrant and the destination do not have to use the same compiler; the Emerald type system uses structural equivalency and thus independent Emerald compilers can be used.

Emerald has a clustering mechanism in the form of the *attachment* concept. We are considering modifications to this concept as to provide for migrant clustering. Migrants need to cluster small and very specific objects and need some control over which of its sub-data objects are replicated.

Currently, the Migrants project is in the preliminary stages; its future depends on funding that will prevent student migration!