Middies: Passive Middleware Abstractions for Pervasive Computing Environments

Daniel Cutting, Adam Hudson, Aaron Quigley
University of Sydney, Sydney NSW 2006 Australia
{dcutting,ahudson,aquigley}@it.usyd.edu.au

The authors would like to acknowledge the support of the Smart Internet Technology CRC.

1. Introduction

As the pervasive computing paradigm develops, it is necessary to have means of communication for applications executing over the many disparate devices that form pervasive computing environments. The dynamically changing set of resources and services available in such environments stems both from the nature of distributed systems and the mobility of devices that constitute temporary parts of the system.

Our work draws on concepts from many existing middleware systems, primarily concentrating on tuple spaces such as Linda [2] and LIME [3], publish-subscribe systems such as Elvin [4], and blackboards such as Hearsay-II [1].

We present our model of *middies*, generalised passive middleware abstractions that can be used to construct middleware systems with operations and semantics desirable to the applications with which they will be used in a pervasive computing environment.

2. Model

Our model is based on a layered architecture which is intended to run on all devices that wish to use the middleware.

By considering the various middleware systems mentioned above, it is possible to identify several general abstractions which make up the top-most Middie Layer (ML).

Firstly, each of the three middleware systems has a concept of a logically centralised server to which the communicating parties connect. In the case of Linda, this is the tuple space. With publish-subscribe systems it is the subscription server, and with blackboards, it is the blackboard itself. We thus generalise this to the concept of shared *spaces*, which are logically centralised and can be referenced by any clients by simply providing a common name.

Similarly, each system has some unit of storage, be it a tuple, an event, object or an hypothesis. This can be generalised to a *block*, which we define simply as a flat, unstructured array of bytes.

Some systems are reactive and need some mechanism for notifying clients of events. This is particularly evident in publish-subscribe systems. We employ the concept of *reactors* which can be registered with spaces essentially as callbacks to be activated under certain conditions.

Each system also has means of extracting data from the shared space. In tuple spaces, tuples are matched to templates. In pub-sub systems, events matching a subscription as defined by a subscription language are propagated to clients. A generalised middleware system needs a *matcher* abstraction, which can be used to compare two blocks and return one if it is deemed to 'match' the other.

A Distribution Layer (DL) beneath the ML allows a logically centralised space to be physically stored across multiple devices. This is achieved by a distribution policy that uses contextual information to determine the best placement of blocks around the network.

A Resource Layer beneath the DL coordinates the devices within the network and abstracts the network addressing and service discovery by way of a service discovery protocol.

References

[1] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[2] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[3] A. L. Murphy, G. P. Picco, and G.-C. Roman. LIME: A coordination middleware supporting mobility of hosts and agents. Technical Report WUCSE-03-21, Washington University, April 2003.

[4] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings AUUG97, Brisbane, Australia, September 1997*. Distributed Systems Technology Centre, University of Queensland, Australia, 1997.