Predator : A distributed location service and example applications

J.N.Weatherall [*], A.Hopper [†]

September 16, 1999

Abstract

This paper introduces a simple, distributed, location service, suitable for deployment on a wide variety of heterogeneous platforms and which is scalable to cope with location forwarding on a global scale. Also described are two existing applications of this service, one in particular being a wireless-via-wired routing service for mobile devices, suitable for deployment both in-building and over a wide area.

1 Introduction

Recent work in distributed computing has focused on the problems of mobility both of software components of systems and of the devices on which they operate.

There are desirable features of conventional distributed middleware architectures such as CORBA[12] which we would like to be able to integrate seamlessly with the more stringent requirements of mobile systems.

One of the primary difficulties in such integration is that of location, both of distributed services and of mobile devices by clients. Common approaches to device location, for example IP Routing[5] and DNS[10], require that the name of a machine reflects its physical location to some degree.

Similarly, distributed architectures built on IP and DNS usually require that programs exporting services not migrate between devices, if clients are to be able to access those services. Mobile services will therefore be considered as equivalent as regards location to mobile devices for the purposes of this report.

[*] Laboratory for Communications Engineering Cambridge University Engineering Department, Cambridge, England

[†] AT&T Laboratories Cambridge, 24a Trumpington Street, Cambridge, England

2 Existing Mobile Location Schemes—Pure vs Impure Names

The issue of transparent location is tackled by a number of different schemes, which can be divided, for the most part, into three general categories;

1. Schemes such as the Globe Object Model's[7] location service, that use *pure* names[11] which convey no information to the client as to how the corresponding entity should be located. These systems are seldom as scalable as we might like, since most implementations require that some central node is aware of every object in existence.

2. Schemes such as Mobile-IP[4] and the CORBA LifeCycle [13] service, that use *impure* or *composite* names, most often comprising a home agent location and a key. These systems can fail when the home agent is unavailable for some reason and can suffer from performance degradation when the mobile entity is far away from its home.

3. Hybrid schemes such as ALICE[3], in which the mobile entity's home is effectively mobile itself and the composite name of the entity is transparently munged to reflect its current home, older names of the entity being forwarded to it's current location transparently. This approach is more efficient than 1 and more manageable than 2 but suffers similar robustness problems to the latter. It is also unclear as to how long these surrogate homes should hold forwarding information for.

3 The Predator Model of the World

The Predator system assumes a world of roaming mobile devices with intermittent connectivity, via *contact-points*, to a backbone of static, wired network infrastructure.

Mobile services can be modelled in the same way, since their contact-points are effectively the devices on which they operate (which may in themselves be mobile).

The core of the Predator system is its location service, on which the Predator IIOP-Forwarding service and PicoNet[1] Routing service are built as applications. For convenience, this is constructed as a CORBA service and deals only with CORBA Object references. In order to track other mobile entities, such as mobile devices, a wrapper layer must be constructed to abstract the mobile devices into CORBA objects, which can then be registered with the Predator Location service on their behalf.

CORBA was chosen as the base middleware for Predator because of its platform and language interoperability and its popularity as an open standard for component software operation. In choosing CORBA, very little restriction was placed on how wrapper layers must be constructed for mobile entities. Although in principle the same techniques could be applied to, for example, a Java RMI environment, it was felt that this might prejudice the system by polluting it with platform-specific optimisations.

4 The Predator Location service

The Predator Location service (from here on referred to as the Location service) is run on all machines wishing to export or import mobile objects. Location service nodes are arranged in a simple, hierarchical tree structure representing the *global domain*, with similar, hierarchically structured trees representing the *sub-domains* implemented as *distributed objects* over the global or *base* tree.

4.1 Search Tree Structure

Each node of the Location service has a concept of a parent node, which it may query for locations of devices which cannot be found locally, and may be aware of its peers and/or its ancestors in order to allow for failure recovery features.

Nodes in the base tree are grouped according to physical locality. For example, all the Location service nodes running on the PCs in a room may be grouped under a single parent node which represents that room. The parent nodes for each room will then in turn be grouped under a node responsible for the building containing those rooms and so on.

Figure 1 shows a simple search tree structure that might be used to handle mobile devices in a small building.

4.2 Sub-trees as Distributed Objects

In the Globe object model, the top-level entities are known as *distributed objects*.

A distributed object consists of multiple *implementation objects* running in separate address spaces on a number of machines and communicating with each other via the network in order to behave as a single entity.

Distributed objects *expand* and *contract* as implementation objects are added and removed. How the objects cooperate to achieve the desired behaviour is up to the objects themselves and may be tailored to the specific task.

Using the Predator Location service, mobile objects are accessed via a *logical* hierarchical naming scheme. The key difference between Predator and DNS-style

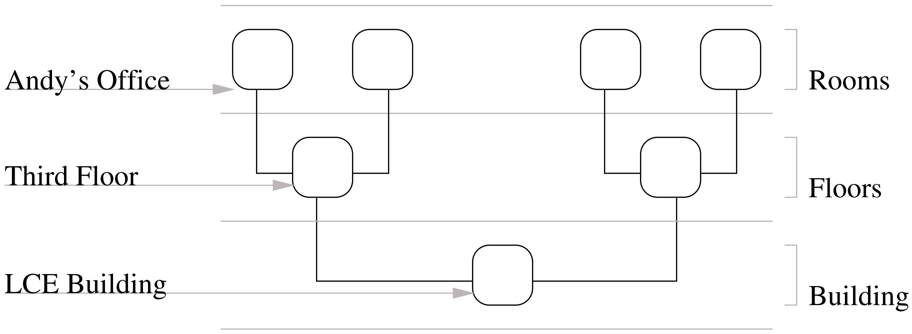

Figure 1: A Hierarchical Tree Structure for a Small Building

systems is that this logical hierarchy is in no way tied down to the physical hierarchy of the system.

Instead, the Predator service is implemented as several layers of search trees, one for each domain in the hierarchical namespace. At the base layer, there is a single tree to represent the global domain. At the next layer up, the search trees implement different sub-domains, which may represent different tasks running on the system. For example, there may be trees implementing "Paging", "IIOP-Forwarding" and "Email" domains. Tree nodes implementing these domains will exist at any particular point in the network *if and only if* mobile objects belonging to them exist at that point.

Figure 2 shows example physical and logical layouts of a backbone network and some mobile objects. Figure 3 shows the structure of the search tree corresponding to the "Paging" domain, overlayed on the physical network layout.

The "Paging" domain can be considered as a distributed object, since it is made up of a number of distributed nodes working to provide a single, unified function—that of locating objects used for the task of "Paging". The "Email" domain can be considered, similarly, as a distributed object, even though it is only implemented in one place. The root domain is special in that it exists everywhere and its structure reflects the underlying geographical structure of the wired network.

4.3 Using the Predator Location Service

4.3.1 Locating a Mobile Object

When a client wishes to locate an object using the Location service, it passes the appropriate hierarchical name to the base Location service on the local machine. For convenience, our implementation runs on a well-known port number and uses

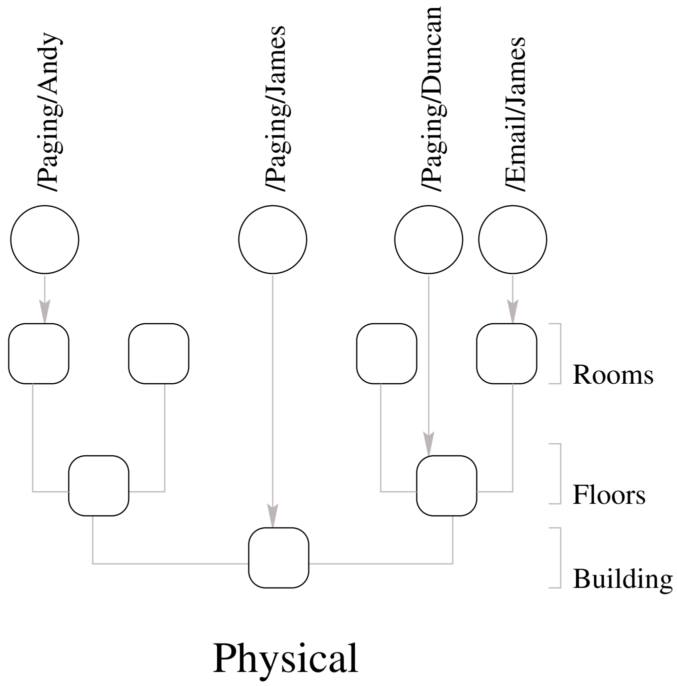

Figure 2: Physical and Logical Layouts of an Example Network

a well-known CORBA object key.

The base Location service will attempt to resolve the first part of the name (if the desired name was "Paging/Andy", for example, the first part would be "Paging") in the global domain, either locally or by querying its parent node. If the attempt fails then an exception will be thrown to indicate that no object exists with the desired name.

If the attempt to resolve the first part of the name succeeds then the returned Location service node (which could be anywhere in the world) is queried to resolve the next part of the name. This continues until the entire name has been resolved, at which point the desired object will have been found.

Figure 4 shows an example search in which client object (1) attempts to locate mobile object "Paging/Duncan". The "Paging" domain is implemented as a tree of co-operating nodes—those labelled P1, P2 and P3 in the diagram. Once a client can see *any* node which is part of the "Paging" domain, it can then locate *any* object that belongs to that domain, simply by searching the domain's tree in the same way as it would search the global tree.

- The client (1) asks the Location service at node "R1" to find the object "Paging/Duncan" (2).

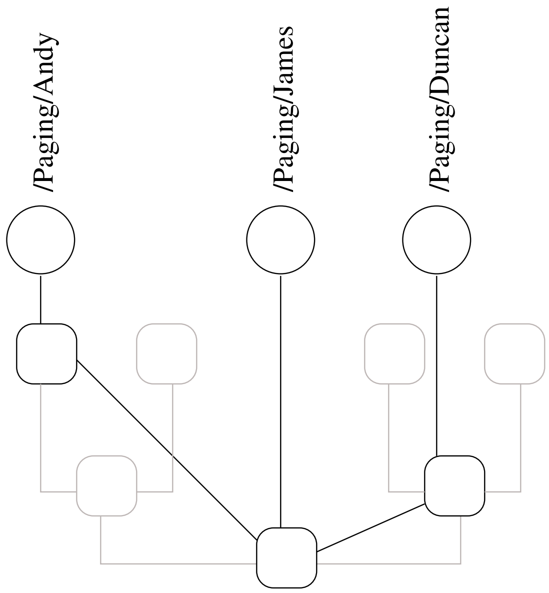

Figure 3: Layout of the Paging Domain Search Tree

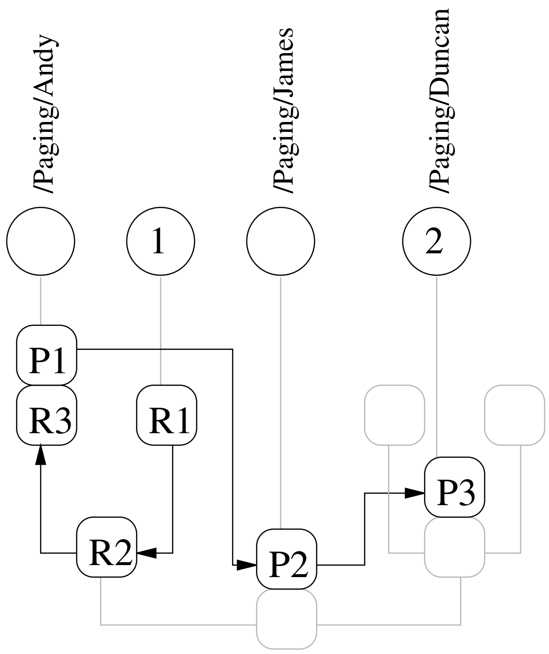

Figure 4: An Example Search of a Layered Search Tree

- Node “R1” first tries to locate the nearest node in the “Paging” domain.
  - “R1” cannot see any nodes in the “Paging” domain locally, so it falls back to its parent, “R2”.
  - “R2” has been told by “R3” that it knows of a node in the “Paging” domain, so “R2” forwards the request to “R3”.
  - “R3” can now return a reference to “P1”, direct to node “R1”. Note that “P1” is not the base of the “Paging” domain’s tree but the node in that tree that is closest to “R1”. This helps us avoid long-distance searches throughout a tree when the desired mobile object is actually local to us.
- Node “R1” now asks “P1” to find the object called “Duncan” in the “Paging” domain. A similar search is then performed of the “Paging” domain to that performed on the global domain, returning “P3” as the current location of the object called “Duncan”.
- The resulting object reference is returned to the client and may be used until it fails for some reason, at which point the client can fall back to another search for the name “Paging/Duncan”. Caching schemes can help reduce the search path on subsequent attempts to relocate an object that has moved.

4.3.2 Registering a Mobile Object

Registering an object with the Location service under a particular name follows the same basic pattern to locating an object. The primary difference is that nodes of all the required domain trees will be created local to the object being registered, if they don’t already exist there.

For example, consider the case in which a new object is to be registered under the name “Email/Andy”, as in Figure 5a;

- The new object (1) calls down to its local Location service node and asks the Location service to register it as “Email/Andy”.
- The Location service performs a normal search, to locate the nearest node in the “Email” domain and finds node “E1” [5b].
- If the nearest node is not local then a new “Email” domain node is created locally and linked to the existing one.
- If there are no existing “Email” domain nodes then one is created locally.
- The object is now registered with the local “Email” domain node, “E2”, under the name “Andy” [5c].

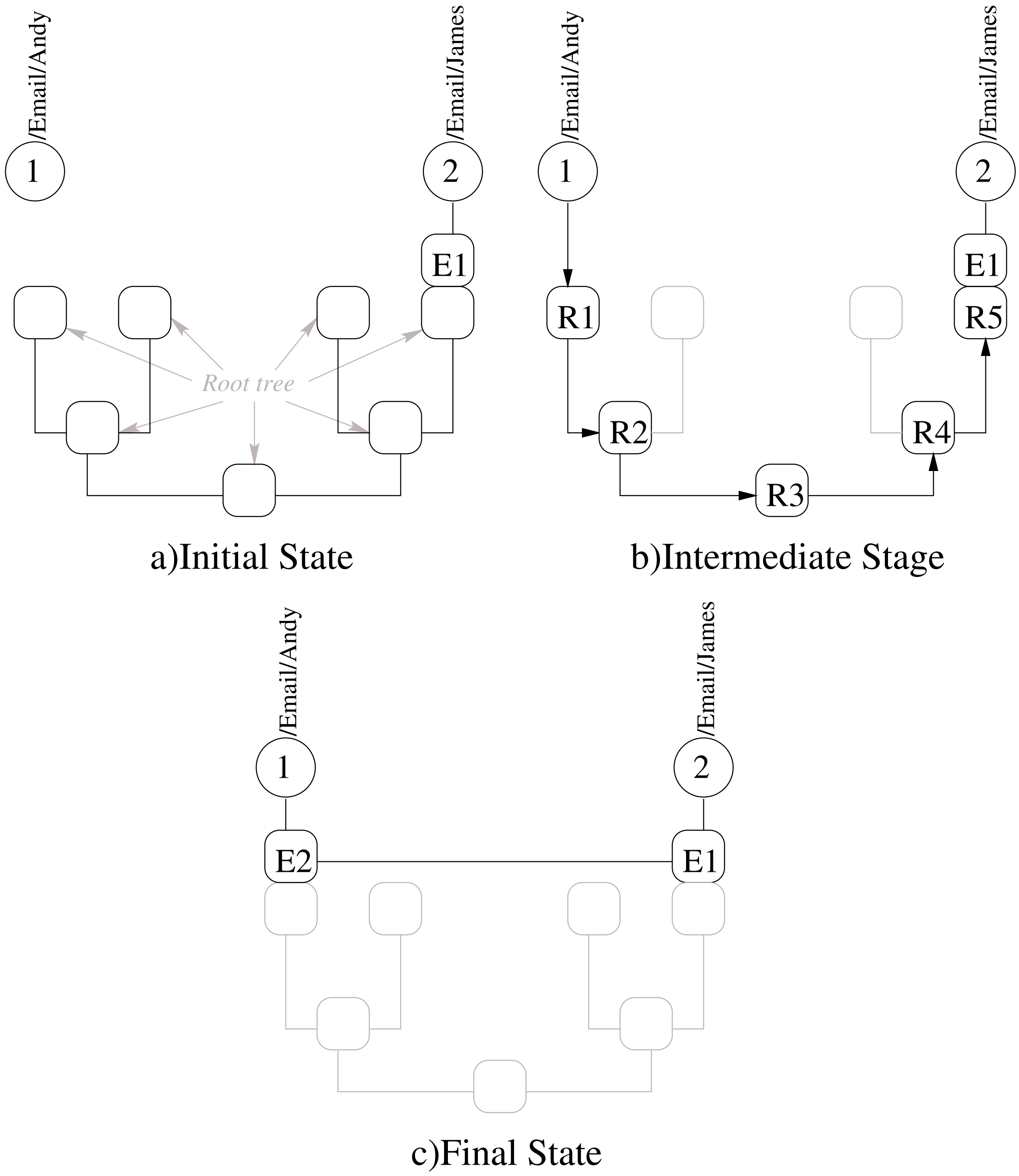

Figure 5: An Example Registration of a Mobile Object

5 Location service implementation

The present implementation of the Location service is a minimal one, with the following limitations:

1. Automatic failure recovery and configuration of nodes in the base tree is not currently implemented since, on the prototype's current scale, such features are neither required nor easily tested.

2. All key names are currently flat. For example, the hierarchical name "PicoNet/ORL/64" is implemented as the flat, munged name "pNET00000064". Work is in progress to implement hierarchical keys and to automate redistribution of domain trees.

3. No security is currently available. Some way of limiting who is allowed to bind particular names is required, in order to avoid namespace clashes and denial-of-service attacks, for example. Use of the CORBA Security Service may go some way towards providing this security.

6 The Predator IOR-Forwarding service

The Predator IOR-Forwarding service, allows applications to generate *global object references* and to pass these to clients in place of the actual object references involved.

When a CORBA service wishes to export a mobile object, it calls into the Forwarding service local to it and requests that the object's reference be bound to a global object reference in the Location service. The service can then pass the global object reference to client applications, in place of the object's real reference. An instance of the Forwarding service must be running on all client and server machines in this model, although it is possible to get round this requirement to support legacy systems at the cost of scalability.

Migration of an implementation to a new site becomes straightforward using the Forwarding service, since the new implementation can be created, initialised and bound to the underlying Location service and the old instance, if required, can simply be torn down. Clients using compliant ORB implementations will detect the failure of the original object and automatically revert to using the global IOR, thus implicitly accessing the Location service to obtain the new instance's IOR. The client process can remain completely unaware that anything has changed.

7 The Predator PicoNet Routing service

7.1 PicoNet

PicoNet is a lightweight, low-power wireless communications system for embedded and mobile devices. The low-power requirement, needed in order to allow

PicoNet to be embedded in even the tiniest devices and therefore be truly ubiquitous, imposes consequent bandwidth and latency penalties on the system. As a result, PicoNet is a technology aimed primarily at low-bandwidth control and negotiation situations, rather than mass data transfer situations as are catered for by systems such as BlueTooth[2] and HomeRF[8].

In its current incarnation, PicoNet allows short range point-to-point communication between devices. While this is ideal for many location-sensitive tasks, it restricts the available applications somewhat—if the recipient of a message is not within about 5 metres of the sender then the message cannot be sent.

7.2 The PicoNet Routing Architecture (RLink)

The Predator PicoNet Routing architecture (RLink) assumes a backbone of wired machines (in our case desktop PCs), with a PicoNet gateway node attached to at least one machine in each room, or more if the room is particularly large. The mobile PicoNet nodes then traverse this backbone as they are carried around by their owners and, while they are still able to communicate directly with each other over short distances without requiring any backbone infrastructure, they can usually fall back to routing via the wired network if the device they wish to contact is not locally available.

Location of PicoNet nodes in this system is, once again, performed by the Predator Location service. It is used to retrieve the IOR of a CORBA object which is willing to act as a gateway to the desired node. If the node has moved away by the time the message is passed to a gateway by a client then the gateway will try to re-locate the client and pass the message on to the new gateway. This avoids repeated traversal of the search tree, since the receiving node is unlikely to have moved very far from the original gateway.

Several assumptions are necessary in the implementation of this scheme;

- Firstly, it is assumed that *only* messages with a specific destination address should be routed through the wired network. The PicoNet system also allows broadcast requests for services to be made but in general such requests implicitly expect that only local instances of the desired service will respond.

- Secondly, it as assumed that *all* messages with a specific destination address may be routed through the wired network.

One useful result of these two assumptions is that although a PicoNet device can make a plea for any nearby instance of a desired service, it may continue to refer to that service instance specifically, by its address, until the transaction is complete, whereupon it may fall back to requesting any local instance again. This session-based mode of operation allows transactions to continue to completion even when the nodes involved are no longer co-located, provided they are both near a Routing gateway.

7.3 Experiences Using RLink

The RLink wireless-through-wired router has been used as the basis for several prototype applications. Two simple examples are discussed briefly below.

7.3.1 Active Door Badges

For some years the AT&T Labs in Cambridge have had an infrastructure of sensors for the Active Badge system[15]. Various automated applications have been constructed using this technology such as Teleporting[14] but the major use remains simply that of finding out where people are so that they can be contacted.

In particular, users visiting a coworker's office will often find that that person is busy or absent and in the latter case they will resort to the Active Badge system to find them. In the former case, it would be desirable for there to be some indication on the door into a worker's room that they are busy, and perhaps some way for people to register the fact that they would like to talk to them when they're done.

To this end some PicoNet devices with small LCD displays and a few buttons have been constructed and used as door signs. In normal operation, such signs will display static information on the room and its usual occupants, and the display hardware is optimised power-wise for this case. However, since the door signs have access to RLink gateways local to them, when one of their buttons is pressed they can fetch current location data for the usual occupants of the room and display it, avoiding the need for the visitor to go to a PC terminal to find the person they wish to visit.

Similarly, it is possible to send pieces of text to be displayed on a door sign in addition to the other information, providing for indications such as "This Room is Busy" or "Wet Paint", for example.

In this case, wireless PicoNet devices were able to transparently access wired backbone services without regard for the location of either party, through use of the Predator and RLink services.

7.3.2 Generic Remote Controller

As part of a demo for a separate project, a CORBA-based mobile streaming architecture was built in our lab. Among other control methods experimented with, a door sign PicoNet node was reprogrammed to instead support a simple audio-player interface, with Play, Pause, Stop, Fast Forward and Rewind buttons.

In addition to supporting the control operations normally associated with an infra-red remote, the PicoNet controller was also capable of retrieving information from the audio player, such as the name of the track currently being played.

For the purposes of our demo, the audio player had actually been a standard PC with an archive of MPEG files stored on it. Because we used the RLink architecture to connect the two, users could roam between rooms and continue to control their audio player from the new location, seamlessly.

In addition, the controller could, without modification, control a real CD player or similar device, provided the device had a PicoNet node attached and was exporting a simple audio-player interface. Control of such a device could either be local or transparently remote, via the RLink service. This feature was especially important, since in using the Predator and RLink services to provide access between remote devices and to backbone services, we hadn't sacrificed the ability to operate in an ad-hoc manner.

8 Conclusion

The Predator Location Service aims to tackle the shortcomings of pure-name, impure-name and current hybrid-name approaches to addressing large numbers of highly mobile objects or devices over a static, wired network infrastructure. It improves on these approaches in the following ways:

- By partitioning the overwhelmingly large global search domain into more manageable logical domains we avoid the scalability issues associated with flat name space systems such as the Globe Object Model.

- By allowing logical domains to themselves be mobile we avoid imposing artificial constraints on the mobility of objects while at the same time providing a means to optimise the search space for related objects. This gives clear advantages over schemes such as DNS, by allowing the topography of the search tree to alter to match the requirements of its clients.

- By the use of two orthogonal trees–the hierarchical logical namespace and the hierarchical physical search space–we allow searches to be optimised to avoid traversing large distances unnecessarily when locating nearby nodes. In addition to supporting far greater scalability, this approach avoids the problems often associated with impure-name and hybrid-name schemes. In particular, there is no single point of failure as in home-agent systems like

Mobile-IP, nor is there a need to maintain forwarding addresses for mobile objects indefinitely as in systems such as ALICE.

- By constructing the search trees for sub-domains as we do, we avoid some of the worse pathological cases possible with a more general tree structured architecture. While pathological cases still exist, they are rare and introduce only minimal extra cost into the system. In general the locality heuristic inherent in the Predator system is appropriate for the target application domains we are interested in.

The transparent IIOP forwarding features of Predator may be applied to distributed systems such as DAWS[6] to simplify the tasks of component migration and failure recovery. This service's co-existence with the RLink routing service illustrates the generic nature of the Predator service. The Predator service could equally well be used to enhance the mobility capabilities of other architectures such as Jini[9].

Some work using the Predator service as a basis for enhancing the usefulness of a wireless ad-hoc technology in an office environment has been briefly described. Research continues into aspects of interoperation between PicoNet devices, in particular for purposes of control, both of backbone services and of local devices.

9 Acknowledgements

The authors wish to thank Ant Rowstron, Duncan Grisby, Sai-Lai Lo and Alan-Jones for their patience in proof-reading the many drafts of this paper. Thanks are also due to Paul Osborn and Gray Girling of the AT&T Laboratories Cambridge, for their help and advice in building the PicoNet side of the system.

References

[1] Frazer Bennett, David Clarke, Joseph B. Evans, Andy Hopper, Alan Jones, and David Leaske. Piconet - embedded mobile networking. IEEE Personal Communications, Vol. 4, No. 5, pp 8-15, October 1997.

[2] BlueTooth website. http://www.bluetooth.com.

[3] Raymond Cunningham. Architecture for Location Independent CORBA Environments. Dissertation for MSc in Computer Science, Trinity College, Dublin, September 1998.

[4] C. Perkins (editor). IP mobility support. RFC 2002, October 1996.

[5] F. Baker (editor). Requirements for IP Version 4 Routers. RFC 1812, June 1995.

[6] Duncan P. Grisby. *A Distributed Adaptive Window System.* PhD thesis, Computer Laboratory, University of Cambridge, 1999.

[7] Franz J. Hauck, Maarten van Steen, and Andrew S. Tanenbaum. A location service for worldwide distributed objects. Technical report, Dept. of Math. and Computer Science, Vrije Universiteit, Amsterdam, The Netherlands, 1997.

[8] HomeRF website. http://www.homerf.org.

[9] Jini Connection Technology. http://www.sun.com/jini/overview.

[10] P.V. Mockapetris. Domain Names - concepts and facilities. RFC 1034, RFC 1035, November 1987.

[11] Roger M. Needham. Names. In Sape J. Mullender, editor, *Distributed Systems*, chapter 12. Addison-Wesley Publishing Co., 1993.

[12] Object Management Group. *The Common Object Request Broker: Architecture and Specification*, 2.2 edition, 1998.

[13] Object Management Group. LifeCycle Service Specification. In *CORBAServices: Common Object Services Specification*, chapter 6. OMG, 1998.

[14] Tristan Richardson, Frazer Bennett, Glenford Mapp, and Andy Hopper. Teleporting in an X Window System Environment. IEEE Personal Communications Magazine, Vol. 1, No. 3, pp 6-12, Fourth Quarter, 1994.

[15] Roy Want, Andy Hopper, Veronica Falcao, and Jonathon Gibbons. The Active Badge Location System. ACM Transactions on Information Systems, Vol. 10, No.1, pp 91-102, January 1992.