Middleware for Context Sensitive Mobile Applications

K.A.Hawick **H.A.James**

Computer Science Division, School of Informatics
University of Wales, Bangor, North Wales, LL57 1UT, UK
{hawick,heath}@bangor.ac.uk
Tel: +44 1248 38 2717, Fax: +44 1248 36 1429

Abstract

Contextual information such as spatial location can significantly enhance the utility of mobile applications. We introduce the concept of active preferences that represent a combination of user preference information and choices combined with spatial or temporal information. Active preferences set the policy on how a mobile application should customise its behaviour not just for a particular user but as that user moves to different locations and interacts with other mobile users or with fixed location base stations. We discuss technical issues for establishing a middleware infrastructure to aid experimentation and describe our prototype testbed.

Keywords: middleware; location context; personal context; mobile devices.

1 Introduction

Mobile computing tools like personal digital assistants (PDAs) and handheld devices have recently become affordable and powerful enough to run high level software. It seems clear that the ubiquity of such devices will significantly increase and that many of the distributed computing ideas, algorithms and applications that were previously only supportable on conventional computers and networks will now be useful on wireless PDAs. We have experimented with the capabilities of technologies such as: the Compaq iPAQ(Compaq 2002) handheld, which can now run Personal Java virtual machines; with WaveLAN(IEEE 2001) and Bluetooth(Bluetooth 2002) wireless hardware devices; and with embeddable control chips such as the PIC(Microchip 2002) and Tini(Dallas Semiconductors 2002) board. Our goal has been to identify feasible usage patterns and a vision for the sorts of distributed computing applications that will be enabled with such technologies. Our belief is that the hardware is progressing well with commodity priced products becoming available, but that there is still a dearth of software to meet the potential of these devices. In particular we believe suitable distributed middleware will enable many new applications, but also that greater use of and access to contextual information will considerably enhance the uptake of mobile computing.

Contextual information in this context includes spatial location and temporal information about the user but also personal information such as "active preferences". We use this term to mean customisable user information that is expressed with spatial and temporal dependencies. An example of a conventional preference might be paraphrased as "use this as my preferred font". An active preference incorporates conditionals, so an example is "switch off sounds when I am in these locations". It is non-trivial to organise the necessary policy metadata to allow users to enter active preference information, particularly as the complexity of stated policies exceed that containable in the user's head. It becomes important to organise preference information quite carefully and allow the user sophisticated tools to set active preference policies and to query his/her own preference database to check on the consequences.

In this paper we describe our experimental setup which we believe is representative of the sorts of mobile environments and apparatus that will become ubiquitous over the next few years. We discuss the mobile context problem in more depth and suggest some ways of tackling it. We review some of the technologies available to build a middleware solution and describe our prototype system. We have been working with distributed computing middleware for some years now and had originally developed our DISCWorld middleware prototype for conventional computers on fixed networks. Since our system is relatively lightweight and written entirely in Java it has been feasible to adapt it for mobile systems in a way that other metacomputing and grid based software cannot. It is therefore useful for supporting mobile user scenarios.

2 Mobile User Scenarios

Mobile computing technology is developing rapidly and the ways in which we use personal digital assistants and other mobile devices are also still developing. We have seen considerable changes in the possible form factors of such devices from notebook computers, through clam shell PDAs like the Psion series, palm top devices from the Apple Newton through the palm series to the Compaq iPAQ. In a related set of developments we have seen mobile phones grow small and sport proportionately larger displays and some become integrated with PDAs. Add ons such as keyboards, GPRS(Mobile GPRS 2002) communications devices, Bluetooth capabilities and even global positioning devices are now possible and are becoming sufficiently lightweight and low power consuming that we can envisage a single PDA that we are willing to carry everywhere with us being available within a few years. In the meantime, personal area network technologies such as Bluetooth will at least allow us to have a relatively small number of communicating and interoperable devices carried in our pockets.

Communications at the room and building scale allows mobile human users to interact amongst one another via their mobile devices and to interact with their localised environment. Figure 1 shows a key idea for four mobile users, represented in proxy by their personal digital assistants (in this case iPAQs).

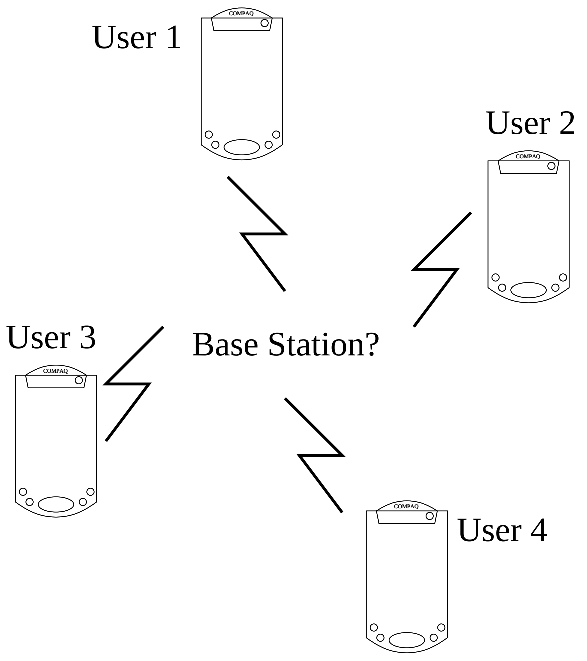

Figure 1: Mobile Human Users Interacting with Wireless PDA with or without base station.

An important technological design issue is whether users/PDAs communicate directly via an *ad hoc* network or whether they have to use an intermediate base station. Wireless technologies such as Bluetooth do allow the formation of temporary *ad hoc* networks. This is useful in the context of a meeting room or in the open air on a field trip or even in a battlefield scenario. Other wireless technologies such as the now near-commodity WaveLAN IEEE 802.11b system generally requires suitable wireless base stations to support mobile wireless devices such as laptops or PDAs. The physical size and power requirements of WaveLAN technology is s till improving and it may be feasible to use network bridge devices to set up completely wireless *ad hoc* networks too.

Several organisations are positing visions of how these communications technologies and mobile devices will change the way we live and work in the very near term future. Hewlett Packard Laboratories' CoolTown (Hewlett Packard 2002) is one well expressed vision of a totally connected individuals, devices and places. Other companies like BT are experimenting with next generation wearable communication devices or Vodaphone with new mobile network communications and applications services for fully connected individuals.

There are many technological problems that need to be solved before such visions can be implemented. Not the least of which problems concern the software infrastructure for devices to interoperate and for users to be able to instruct their devices precisely how they wish them to behave. It is not yet clear how users will wish to interoperate with mobile devices. No doubt some users will wish to have a high degree of "connectivity" while others will want their devices to field calls for them and protect/isolate them from interruptions. It seems likely that we will wish to operate in different modes in different locations and at different times of day and periods in our working or leisure oriented week.

We believe an important aspect of realising CoolTown style visions is to set up experiments in mobile device usage both from a technological capability perspective but also from a social interactive point of view. In the case of many distributed computing technologies the main thrust of development is "cheaper, faster and better". Mobile computing systems are more of an unknown at the present time. It is not clear what "better" is from users' perspectives. There are also interesting generation gap problems in terms of the physical characteristics of mobile devices. Older users may find the new PDAs with touch sensitive keyboards more accessible than the tiny keyboards on mobile phones. Younger users seem to have no problem adapting to the use of such small devices. Different generations of humans have different tolerances for delays and interruptions in the response and behaviour of mobile devices.

It is a considerable challenge for smart devices to interact with humans in a mobile infrastructure. Mobile devices such as robots might use the same infrastructure and will have different tolerances to delays and temporary disconnections. Agent approaches may provide a solution to these problems. Agents may act as proxies for impatient humans and can provide a smart interface to the infrastructure. We discuss this further in section 4.

We anticipate exciting developments in the physical form factors of mobile devices. It would seem important to press on with research and development in the software infrastructure for the devices of the future.

We are experimenting with software and data exchange protocols for making use of spatial and temporal contextual information by mobile applications.

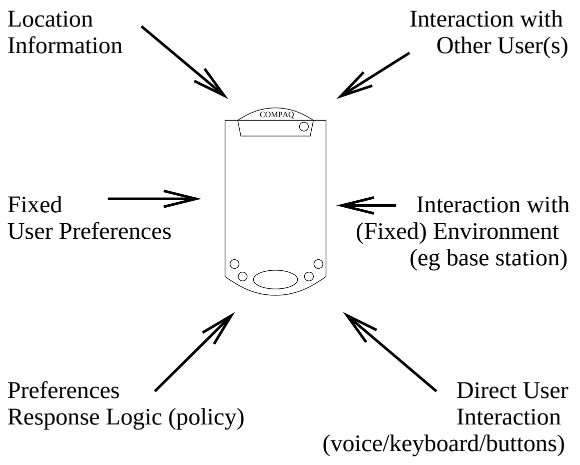
Location Information
Interaction with Other User(s)
COMPAQ
Fixed User Preferences
Interaction with (Fixed) Environment (eg base station)
Preferences Response Logic (policy)
Direct User Interaction (voice/keyboard/buttons)

Figure 2: A PDA Mobile User Device in Context.

Figure 2 illustrates the sorts of information sources that are available to applications running on an iPAQ using today's technologies. In the near future it is likely that voice/speech information will also be readily integrated into the application software stack and the device may also have access to various sensors or other devices connected via the user's personal area network.

It is possible to experiment with existing devices using custom written applications and using the various communications stacks that are available for wireless communications. We believe that a robust middleware infrastructure is the best approach to enabling mobile applications with mobile contextual information. In the section 3 we discuss the contextual data problem in more depth before describing our attempts to support it in middleware in section 7. In section 5 we discuss technological ways of determining approximate location information for mobile users in the absence of a global positioning system or in the absence of applications software stack accessibility to GPS stack information.

3 Context and Location Information

The defining aspect of mobile computer users is that they can be in different locations and may wish different behaviours from their mobile devices dependent upon location. It is desirable that mobile devices be able to cope with temporary disconnection (or "going nomadic" as it is known in the mobile computing literature). It is feasible to have the mobile device and the user's base station interpret the user's location information and perhaps anticipate when a disconnection will occur - due to lack of network coverage for example.

At a simple level we can imagine all mobile devices in the future as having a global positioning capability and being able to know their locations to a few tens of centimetres accuracy. This is likely to be more accuracy than will be needed for the mobile human user. It is likely that map information will not be that accurate and indeed the first few generations of mobile users may have to participate in territory charting surveys gathering new map information which they can transmit back to base.

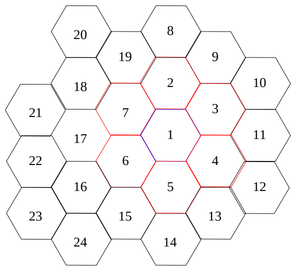

Figure 3: Idealised cell model with regular cell coverage and no interfering obstacles.

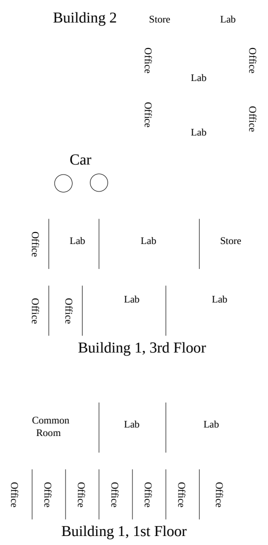

Figure 4: Realistic map of coverage zones in different parts of buildings and even in mobile zones such as vehicles.

Figure 3 shows the classic hexagonal mesh model beloved of mobile telecommunications engineers. This has become the traditional starting point for arranging cell coverage over a geographical area. Cell sizes have shrunk considerably in recent years with appropriate tightening of frequency usage and the advent of smart frequency and cell handover protocols. Cell sizes might now be 100m or less in busy urban areas.

This model is not directly useful to the mobile user however. In practice the world space of interest to the mobile user will be more like that shown in figure 4. The zones are named by the user and have meaning to him/her rather than being anonymous (or purely named) "cell Number 1234873/34" - a name that is only usable by the software itself.

We are interested in accessing location information from user applications programs and with experiments in usability of such mobile awareness in applications. In this section we discuss how such location contextual information might be used in applications and some of the associated software infrastructure issues.

We envisage a model whereby users can either directly or indirectly set up a personal world map of the zones of interest to them. Figure 4 show the work zones of interest to us. We work mainly in one building and have set up wireless networks in the two corridors where our offices and labs are. We are also designing a new technology transfer building that has to be reached by car. We spend most of our working time in these four zones and would like to be able to customise the behaviour of our mobile devices accordingly.

Rule Name	Property	Time
transition enter	transition	specific
transition leave	transition	specific
transition a to b	transition	specific
when in a	location	period
when not in a	location	period
anywhere always	location/time	open
nowhere never	location/time	closed

Table 1: Zone Events for managing context aware information

Table 1 shows the sort of positioning information that a user might want to specify in order to customise his/her mobile device. The location oriented properties or attributes relate to being in or out a specified zone or are associated with transitions to or from one or more zones. The primitive rules specified in table 1 can be combined in various ways using an obvious logic. These combinations provide the basis for a policy language or specification of the users preferences.

Various sorts of mobile applications are discussed in section 6 but one key such application is an instant message manager, that might itself be part of the systems middleware organising message events between applications or between mobile user(s) and base stations. The sort of policy specification discussed above is vital to customise behaviour of the message manager. The user might set various policies about when he does or does not wish to receive messages depending upon his location.

The message manager can operate in various ways using a broadcast/multicast mode or a publish/subscribe model amongst applications modules. We discuss these in section 7.

Managing context information is non-trivial. Many interactive applications such as web browsers or email managers offer their users a "preferences" menu which allows customisation of various static personal data such as email identity, which server to use, which preferred font to use. As applications grow in complexity it is hard to design an intuitive grouping or organisation of preference data. This problem is compounded when mobile/location and time is convolved into the static preferences. We discuss this problem of "active preferences" in section 6.

4 Mobile Robotics Context

Although much of our discussion in this paper has been on supporting mobile human users via their proxy PDAs, one of our closely related projects concerns software infrastructure for interacting robots. Our interest is in swarms of smart robot systems that can cooperate on tasks such as searching or mapping.

Contextual information is important for determining robot behaviour and our middleware is also applicable to robot systems. The robot swarm makes use of spatial and temporal contextual information as well as proximity knowledge concerning what other environmental features and other robots are nearby.

We have experimented with distributed robotics systems based around the Cybot mechatronics. This is described more fully in(Hawick, James, Shepherd &Story 2002). The robot swarm participants require a different sort of contextual information from mobile human users. The time scale and associated response times) and robustness requirements are much more aggressive for agent controlled robots than for humans.

Figure 5 shows the rough schematic for our enhanced Cybot nodes. It illustrates the use of several coupled electronics control devices to support a high level programmable brain (in Java) alongwith mobile/wireless communications capability.

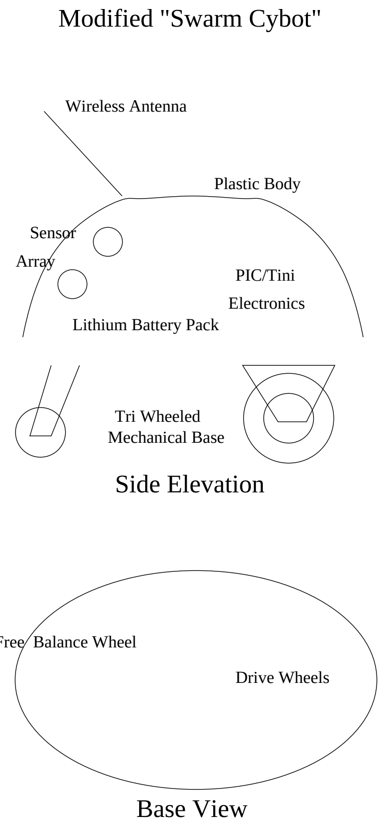

Figure 5: Schematic structure of our modified Swarm Cybot using PIC microcontroller and Tini electronics running the DISCWorld Lite Daemon on the Java Virtual Machine on the Tini board..

Figure 6 shows the Tini(Dallas Semiconductors 2002) control boards upon which we have based our Cybot's enhanced control electronics. The Tini board is capable of supporting a Java Virtual Machine and various network communications protocols. We use PIC microcontrollers to interface between the main Tini brain and the device drivers such as motive units and sensors.

Figure 7 shows a view of the Cybot mechatronics including tri-wheeled base unit, plastic shield and

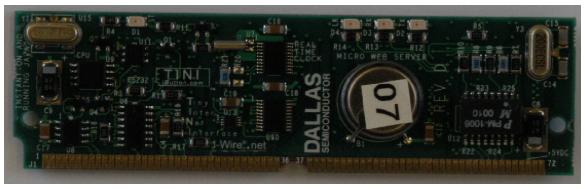

Figure 6: Dallas Semiconductors Tini Board (shown around two thirds actual size), with built in: processor, 1M memory and ethernet link.

sensor array and wireless antennae. The original Cybot design is based on Warwick's "seven dwarf" robots(Real Robots 2002). We use the same mechatronics but are modifying the sensors to include surround ultrasonics and have completely modified the electronics and control systems to use more powerful PIC microcontrollers for the low level device drivers and a Tini processor board that can run a Java Virtual Machine.

We are presently working on software interfaces between a Bluetooth communications module(Cambridge Silicon Radio 2002) and the PIC microcontroller. In the interim we are using relatively lightweight WaveLAN bridge devices to enable IEEE802.11b wireless communications between the Tini's ethernet connection's.

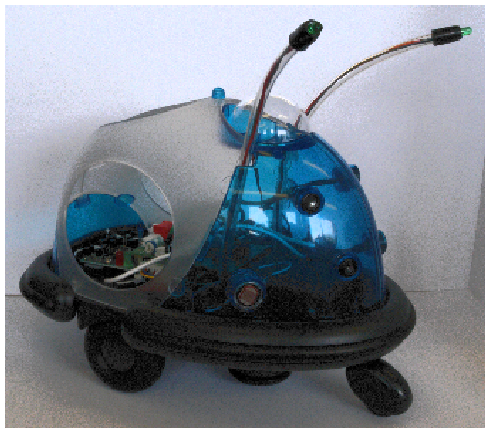

Figure 7: Real Robots "Cybot" Mechatronics.

Figure 8 shows our experimental robot swarm pen in which robots can: interact directly via their on board sensors; be monitored from an "overhead satellite camera"; communicate with one another and with a base station via a wireless communications network. The overhead camera allows us to experiment with mobile robots in an environment where we can employ precise location information. We experimented with a grid pattern drawn on the floor of the robot pen to aid position tracking but it is possible to use simple pattern recognition algorithms for both the pen and individual robots that yields satisfactory spatial resolution(Hawick et al 2002).

The Tini modified cybots provide a relatively cheap hardware platform to allow us to experiment with swarm behaviour. We expect that the cost and power consumption capabilities of PC based systems will allow us to implement a much more complex onboard software environment within a few years.

The mobile swarm of robots requires a similar software infrastructure to a set of mobile human users with PDAs. Although the robots might not set their active preference policies they might have them pre set as part of their behaviour programming. We believe the policy issues and our infrastructure system has some uses in support of artificially intelligent

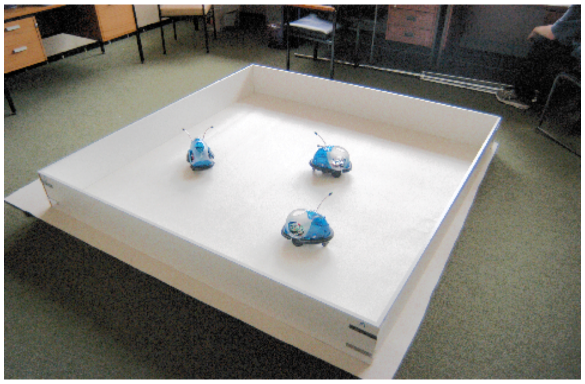

Figure 8: Cybot Pen with WebCam "positioning satellite" above.

robots. Following a layered or component approach to building an AI , a means of specifying policy information is a powerful tool for experimentation. We are experimenting with embedded AI languages such as Prolog in a Java supported environment like that supported on our modified cybots. In section 6 we describe the active preferences and policy issues and how they relate to embedded AI techniques.

5 Mobility Support Technologies

We anticipate Global Positioning Systems (GPS)(Dana 1994) technologies to continually improve so that coverage and resolution, physical weight and cost of the technology all to be sufficient to equip each mobile PDA within a few years. However we need to experiment with location contextual information to develop suitable software infrastructures now. To this end it is possible to encapsulate location contextual information sources and ti make use of the various positional information that we do have access to already.

Source	Resolution	Cost
Manual Entry	Variable	Cheap
Fixed Network	Inaccurate (subnet)	Cheap
Infra Red	Approx (part room)	Medium
WaveLAN Wireless	Approx(net cell)	High
Bluetooth	Approx(room)	High
Cellnet Phone Link	Approx(cell)	High
Global Positioning	Accurate	V. High

Table 2: Sources of Location Contextual Information for Mobile Devices and their typical properties.

Table 2 surveys the location information sources already accessible from an application program.

Our eventual goal is to develop a context-aware system that is able to use highly-accurate positional information to help decide what information may be relevant to a mobile device. In order to do this we need a mechanism that will allow us to read positional information. In this section we describe the experimental testbed we are using to implement the `PositionManager` module of our middleware.

Ideally, we would like to equip all our mobile devices with a highly accurate GPS receiver. However, there are two major obstacles with this approach: the first is commercially-available GPS receivers are only accurate to a range of 3-7 metres (Dana 1994); and the second is that the cost to equip *all* our mobile devices is cost prohibitive. Until the time that the accuracy is improved and the price drops, we are pursuing alternative techniques in an effort to gather positional information and to increase the accuracy of the information already available.

The techniques we are using the implement our `PositionManager` module are: Simple Network Management Protocol (SNMP) and Remote Monitoring (RMON); Bluetooth availability; Infra-red Data Association (IrDA) availability; other network-centric tools and scripts. Before describing the approaches we are taking, it is useful to describe our experimental testbed.

Our testbed currently runs throughout our department's buildings and spills out into the nearby open areas. Several corridors are covered by 802.11b wireless Ethernet reception. In our experience the base stations we are using have a range of approximately 15-20 metres. We have divided our testbed into a moderate number of zones, with as small overlap as possible, loosely based around the area covered by the wireless base stations in the first instance. Other corridors have Bluetooth-enabled computers in offices and laboratories. Each machine (and wireless base station) is connected to a specified port on a switch. Switches are reticulated so as to extend the range of our testbed and cover a small number of different (*untrusted*) subnetworks. We are also about to start experimenting with servers run at staff members' houses utilising 'always-on' internet connectivity.

SNMP and RMON are being used to interrogate network-aware devices such as switches for the devices which are connected to them. Freely available SNMP agents can be used to dump a list of MAC addresses that the switches have learnt about on each of their ports; we have written scripts to search for known mobile devices and to generate approximate positional information. This general positional information may be fed into the `PositionManager` module, where it can be further refined by extra information incident to the system.

The main drawback with this approach is there are different specifications for SNMP Management Information Base (MIBs) and Object Identifiers (OIDs) for most makes (and some models) of network devices. This means we are compelled to write different access routines and scripts for the different devices we wish to query. We are also experimenting with software technologies such as Expect (National Institute of Standards and Technology 2002) to simplify and encapsulate the information gathering phase.

Most of our current mobile devices, Compaq iPAQ PDAs, have Bluetooth capabilities built in. In addition a number of machines have external Bluetooth development kits (Cambridge Silicon Radio 2002) connected via USB or serial connections. We are adding functionality into our `ContextManager Middleware` prototype to detect when another Bluetooth-enabled device comes into range. This allows us to pass extra information to the `PositionManager` module which can be used in conjunction with other information, such as which wireless base station is being accessed, to narrow down the suspected location of the device. We are currently investigating a number of artificial intelligence engines to provide a 'fuzzy logic' approach to increase the accuracy of positional information. We are also attempting to detect the strength of the signal received by the nearby Bluetooth device to make an estimate of its distance from the current device.

The final technology we are investigating for positional information is infra-red (IrDA). In common with Bluetooth, most mobile devices come with IrDA functionality. IrDA's range is up to approximately 1 metre, so if a device is seen to come into range then its location can be quite accurately placed. The attainable bandwidth for IrDA 1.0 is limited to 115kbps, which makes it barely suitable for transferring enough information to establish whether a device is in range, but certainly not enough information for this to be used as the sole means of communication. IrDA 1.1 devices can achieve up to 4Mbps, which allows for more complex information to be exchanged. IrDA's major drawback is that it is necessary to have line-of-sight (up to 15° deviation) between the transmitter and receiver, which makes it less useful for establishing a precise location.

6 Active Preferences

We have seen that standard machines connected to the internet and world wide web are slowly becoming inundated with unsolicited information, from junk email to pop-ups on web pages, and users are continually bombarded with information that is not particularly relevant to them when it is received (for example if you are working in the computer science department, you do not want to be reminded there is a seminar about to start in the history department unless you happen to be interested). Furthermore, when users work via mobile devices, which are often restricted in some senses when compared to desktop machines (in terms of screen resolution, interconnection bandwidth and battery life), users might want to further restrict the information they receive.

The main purpose of this system is to create a software system that can be used to reduce the amount of irrelevant information that is seen by a mobile user. We first define what is meant by contextually aware: by this term we mean that the system is aware of both the user's location, the current time and the information they wish to receive. As shown in figure 9, location and time information is convolved with static preferences make them active.

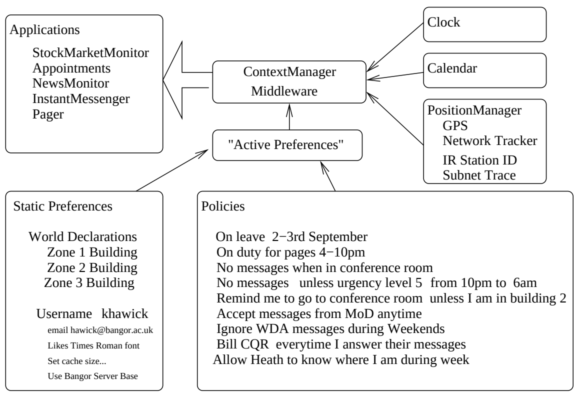

Figure 9: Architecture for combining static user preferences with policies and system information such as position and time to obtain active preferences that can be fed to applications to determine courses of action or to trigger events. Middleware can group this information and provide support services to mobile applications.

Essentially the system is designed to react to events as supplied by applications. When an event occurs, such as an email being received, a file being created, or simply a message being added to a queue, the `ContextManager` consults the other software components (`Clock, Calendar, PositionManager` and `Active Preferences` module and makes a decision on whether the user, or their nominated applications/software components should be made aware of the event. The `PositionManager` module is described in section 5; in this section we focus on the `Active Preferences` module.

Our system can react to events in one of two ways. The system can either be run in a push or pull model. In the push (or multicast) model, whenever an event occurs, it is sent to the `ContextManager` which then starts the process that determines whether the event is propagated to the user's applications. This mechanism is useful if the user is interested in a wide range of events. The pull (or publish-and subscribe) model involves software components registering their interest in events either of a certain type of originating from a certain component. We are currently experimenting with the benefits of these two approaches but will continue to allow users to choose their favoured approach.

The most critical component of the system is the `Active Preferences` module. This module stores a list of user preferences (rules) that are to be consulted when an event arrives. The main challenge is how to structure and represent the rules in such a way that they are easily readable by the user that wants to maintain them, easily extensible for future situations, and are general enough so that preferences can be specified in a way that they will be applicable for many events (if desired).

While there are existing policy specification languages in the research literature, they are often targeted towards particular application domains. For example, the Ponder (Damianou, Dulay, Lupu &Sloman 2001, Dulay, Lupu, Sloman &Damianou 2001) language is designed for security policies on access control mechanisms.

The information retrieval community has considered the use of smart or artificial intelligence techniques for some time(McCune, Tong, Dean & Shapiro 1985, Guach 1992). Systems like RUBRIC users to specify production rules for refining queries over a document space. Similar techniques can be applied over a "circumstance" space which we define to include state settings that arise from spatial and temporal context. Recent work reported in progress(Kim, Lu & Raghavan 2001) reports refinements to the RUBRIC approach to construct rules automatically. We believe this approach has considerable promise.

We are currently experimenting with an XML-based representation of a simple general language coupled with a Java-based Prolog engine (Chirico 2002) to provide the logic interface. Early results are promising, however we are considering the need to name individual users (or their devices), applications and software components, and also having to name the zones in our experimental testbed.

Our system maintains users' preferences as an XML tree. Using such a data structure also allows us to publish system-wide, or application-wide policies that are to be incorporated when making decisions. Other metadata, such as what formats applications can understand, is stored as system-wide preference information. One of our main research issues, especially in the current implementations, is the scalability of our solution and the problem of how to structure the schemas that describe the preferences language.

We are also developing tools that allow users to manipulate their preferences in a general way, and to also verify their saved rule-sets against so-called "what-if" scenarios. For example, given my rule-set what if I receive an email asking me to attend a meeting on Saturday, or while I am driving home from work? This tool will allow consistency checking to ensure users' preferences are not redundant or worse, contradictory. Finally we are adding functionality so that users are able to save a set of cases against which they want to be able to test their preferences whenever they are changed.

7 Middleware and DISCWorld Lite

Mobile devices such as the Compaq iPAQ come with several web based applications and are capable of supporting a full web browser and Java Applet plug ins. The Personal Java Virtual Machine also supports networking, threads and customised class loading. These features allow us to implement a "server" that can run autonomously on mobile devices. A server is important if mobile users are to interact with one another using *ad hoc* networks rather than just via an intermediary base station. A local server also opens up the possibility of running a local middleware that can interact with other peers. Our DISCWorld peer to peer middleware was designed this way(Hawick, James & Coddington 1999) and since it was implemented entirely with Java we have been able to simplify it into a "DISCWorld Lite" that is small enough to run on both PDAs and on the robot controlling Tini devices described in section 4. Our JUMP Java based message passing environment has similar portability although it used certain introspection and object serialisation capabilities that are not available on the KVM supported by the Tini system. We have therefore had to build a very simple point to point message passing system (tentatively called JUMP-Lite).

The general aim of distributed computing middleware is to provide an enabling software layer that sits above the operating system and below the software applications that are making use of the middleware's services.

We have described the additional requirements for our context-aware mobile middleware system. We now discuss the special issues that arise in implementing a set of event and communications management services.

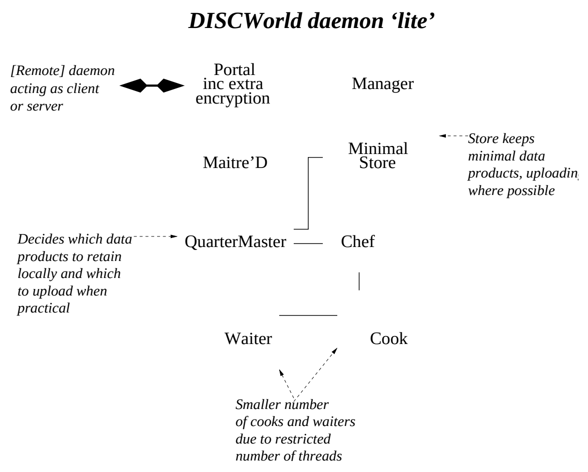
DISCWorld daemon 'lite'
[Remote] daemon acting as client or server
Portal inc extra encryption
Manager
Maitre'D
Minimal Store
Store keeps minimal data products, uploadin where possible
Decides which data products to retain locally and which to upload when practical
QuarterMaster
Chef
Waiter
Cook
Smaller number of cooks and waiters due to restricted number of threads

Figure 10: DISCWorld Lite Internal Daemon Architecture.

DISCWorld was originally designed a s workflow based environment. High level jobs or user queries were satisfied either synchronously or asynchronously by constructing directed acyclic task graphs. Components of the DISCWorld daemon are shown in figure 10. The daemon is guarded by a portal which adds or read security information to inter daemon communications; workflow requests are launched by the maitre'd as waiter threads; the quartermaster organises the retrieval of data from the store or production of new data products by the chef and associated cook threads. A manager object controls policy information used by all other parts of the daemon. This architecture can be extended for mobile users by adding more aggressive cache management and data hoarding components in support of a mobile or temporarily nomadic user. These ideas can be added into the DISCWorld daemon's store.

Event management can be added by providing a whole new daemon component - that of a communications manager that is able to act as a DISCWorld client and launch task graphs itself, but which is able to act a sa broker for messages between other DISCWorld components but also amongst different daemons running on different mobile PDAs or on Cybots.

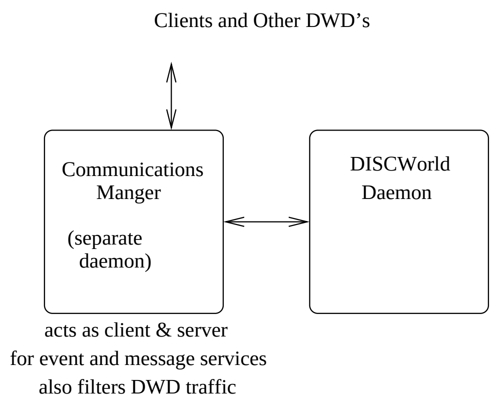
Clients and Other DWD's
Communications Manger
(separate daemon)
DISCWorld Daemon
acts as client & server for event and message services also filters DWD traffic

Figure 11: DISCWorld Communications Manager intercepts DWd traffic and can act as both a client and server, launching DISCWorld task graphs but supporting message registration and notification.

Figure 11 shows how the communications manager interacts with our existing DISCWorld daemon structure.

It has been our experience that experiments in distributed computing often lead to reusable components that can be abstracted and incorporated into a middleware. We hope to integrate the communications manager further into a new DISCWorld daemon.

At present the communications manager provides a simple publish/subscribe model for applications components that wish to be informed of certain events. A simple heartbeat (or polling) mechanism is used to re read policy and location information. An event driven mechanism is likely to be more responsive and consistent although the present system is adequate for our experiments in mobility usage patterns.

The design of the communications manager shares some ideas with that of an instant messenger application. It is able to actively filter or re-route messages or events based on policy information from the active preference settings.

We see the main obstacle to progress as more of a sociological one of agreeing a set of metadata descriptions or keywords to allow users to agree on names for zones and policy components. Setting up an XML based policy description specification/language as described in section 6 seems the best way forwards. It is relatively straightforwards to adapt our middleware and communications manager to use such a specification.

8 Discussion and Conclusions

In this paper we have focussed on technical issues for setting up mobile computing experiments with human users proxied by PDAs or mobile interacting robots.

There are a number of important social and privacy issues related to mobile computing middleware and preference information that we have not addressed.

It is likely that users may well adopt their PDA as the device that holds their master copy of their personal data such as calendar/appointments, bank, tax, finance, household information. Ignoring the problem of ensuring backups and safe copies of this data it is not clear how much of a user's personal information base he/she is happy to release to any one other interactive device or other user or application. In general we might be happy to tell a networked appliance that we wish to buy some product and that here is how we wish to pay. The interesting situation arises if networked appliances get together behind our back as it were and pool their data to build up a profile on us based on all the information snippets they have individually received. It is not clear how to manage this or legislate against it. Mobile networked users and communications networks may make this problem more acute.

It is not yet clear how users will wish to use mobile PDAs. We have found that the present level of technology is now quite useful. Having a genuinely pocket sized PDA that allows networked applications such as web browsing; calendar and appointment agreement; email; and even telnet/ping and low level distributed computing applications that we use in our day to day research is very convenient.

Nevertheless we see a realm of more powerful smart application assistants that can be enabled by providing contextual information, as we have described in the form of active preferences or policies, and which can be combined with position and time information to trigger behaviours on PDAs. We have shown that it is possible to provide useful if approximate location data to the middleware, based upon interception of low level network data.

We strongly believe that smart mobile middleware is the best location in a software stack model to provide these sorts of services. We have described our development prototype based on our DISCWorld system. This is proving a useful research platform and is enabling us to consider usage and sociological issues connected with mobile systems as well as interesting technical and systems research questions.

We have briefly described our mobile robots system and outlined some of the similarities in middleware requirements between PDA enabled humans and mobile communications enabled robots. It is not clear whether precisely the same middleware can serve both needs but we certainly believe it is important that the two systems can interoperate.

The use of AI language techniques combined with user preference and policy statements is proving an exciting research area.

Acknowledgements

The Distributed and High Performance Computing Group is a collaborative venture between the University of Wales, Bangor and the University of Adelaide in South Australia. Thanks to J.E.Story and R.G.Shepherd for assistance in setting up the Cybot robots and for discussion on software architecture for them.

References

Bluetooth. The Bluetooth Core Specification. Available from http://www.bluetooth.com/-dev/specifications.asp Last visited August 2002.

Cambridge Silicon Radio. Casira Bluetooth Software and Hardware Development System. Available from http://www.csr.com Last visited August 2002.

Ugo Chirico, "Java Internet Prolog", Available from http://www.ugosweb.com/jiprolog Last visited September 2002.

Compaq. *iPAQ* Pocket PC. Available from http://athome.compaq.com/showroom/static/-iPAQ/handheld_jumppage.asp Last visited August 2002.

Dallas Semiconductors. "Tiny InterNet Interface" Available from www.ibutton.com/TINI Last visited September 2002.

Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman, "The Ponder Policy Specification Language", in Proc Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, Jan 2001. Springer-Verlag LNCS 1995, pp 18–39.

Peter H. Dana, "Global Postioning System Overview", Available from http://www.colorado.edu/geography/gcraft/-notes/gps/gps_f.html Last visited September 2002.

N. Dulay, E. Lupu, M. Sloman, and N. Damainou, "A Policy Deployment Model for the Ponder Language", in Proc. IEEE/IFIP Int Symp on Integrated Network Management, Seattle, May 2001.

Susan Gauch, "Intelligent Information Retrieval: An Introduction", J. American Soc for Information Science, 43(2):175-182, 1992.

K.A. Hawick, H.A. James, A.J. Silis, D.A. Grove, K.E. Kerry, J.A. Mathew, P. D. Coddington, C.J. Patten, J.F. Hercus, and F.A. Vaughan, "DISCWorld: An Environment for Service-Based Metacomputing," *Future Generation Computing Systems (FGCS)*, 15:623–635, 1999.

K.A. Hawick, H.A. James, R.G. Shepherd &J.E. Story, "Distributed Computing for Robotics", DHPC Technical Report DHPC-114, School of Informatics, University of Wales, Bangor, September 2002.

Hewlett Packard. "A Future Called CoolTown." Available from http://www.hpl.hp.com/news/-cooltown.html Last visited September 2002.

IEEE. Standards for local and metropolitan area networks: Standard for port based network access protocol. *IEEE Draft 802.1X/D11*, March 2001.

Minkoo Kim, Fenghua Lu, and Vijay V. Raghavan, "Automatic Construction of Rule-Based Trees for Conceptual Retrieval", Proc SPIRE2000, Sept 27-29, 2000, Ed. A.Coruna, Spain, IEEE Comp. Soc Press.

Brian P. McCune, Richard M.Tong, Jefrey S. Dean and Daniel G. Shapiro, "RUBRIC: A System for Rule-Based Information Retrieval", IEEE Trans. Software Engineering Vol SE-11, No 9, September 1985.

Microchip, "PICmicro microcontroller", Available from http://www.microchip.com Last visited September 2002.

General Packet Radio Service Available from http://www.mobilegprs.com

National Institute of Standards and Technology. "Expect Home Page", Available from http://expect.nist.gov Last visited September 2002.

Real Robots. "Cybots" Available from http://www.realrobots.co.uk Last visited September 2002.

David L. Wilson. "GPS Horizontal Position Accuracy", Available from http://users.erols.com/-dlwilson/gps.htm Last visited September 2002.