A General Purpose Programming Framework for Ubiquitous Computing Environments

Graeme Stevenson, Paddy Nixon and Robert Ian Ferguson

The Global and Pervasive Computing Group
Department of Computer and Information Sciences
The University of Strathclyde
Glasgow, Scotland.
{firstname.lastname}@cis.strath.ac.uk

Abstract. It is important to note that the need to support ad-hoc and potentially mobile arrangements of devices in ubiquitous environments does not fit well within the traditional client/server architecture. We believe peer-to-peer communication offers a preferable alternative due to its decentralised nature, removing dependence on individual nodes. However, this choice adds to the complexity of the developers task. In this paper, we describe a two-tiered approach to address this problem: A lower tier employing peer-to-peer interactions for managing the network infrastructure and an upper tier providing a mobile agent based programming framework. The result is a general purpose framework for developing ubiquitous applications and services, where the underlying complexity is hidden from the developer. This paper discusses our ongoing work; presenting our design decisions, features supported by our framework, and some of the challenges still to be addressed in a complex programming environment.

1 Introduction

The vision of ubiquitous computing [1], where large numbers of devices and sensors are embedded into the physical environment, providing contextual services to mobile users and applications, is progressing towards realisation. Increases in the performance of handheld and embedded devices along with improvements in networking technology are aiding this process.

Our previous work, the Strathclyde Context Infrastructure (SCI) [2], was premised on the concept of an overlay network, designed to mirror the view of how context aware services may be deployed, managed and used. The project investigated approaches for the discovery, aggregation and delivery of context information and was also concerned with identifying major research areas within the field of ubiquitous computing.

One of the conclusions drawn from the implementation of that work was the need for a framework to aid general application and service development for ubiquitous computing environments. This paper presents our proposal for meeting this need, discusses our design decisions, and presents some of the challenges that remain to be addressed.

The remainder of this paper is structured as follows: Section 2 motivates and proposes a framework for ubiquitous application and service development along with a brief overview of some the characteristics that we believe make peer-to-peer and mobile agent technologies appropriate for such a project. The high-level design of the proposed framework is then presented in section 3. This is followed by some of the challenges central to ubiquitous computing that still need to be addressed in section 4, and related work in section 5. Finally, the paper concludes with a summary and outline of future work in section 6.

2 Motivation

Building a distributed application is not a simple task. Many factors need to be considered by the developer, such as component failures, network failures, security and remote communication. In ubiquitous computing environments there are extra issues to consider, such as partial views of data, additional modes of failure, personalisation of services and communication with third party, and previously un-encountered entities in a heterogeneous environment.

Due to the complexity of such environments, we believe there is a large benefit to be gained in providing developers with a programming framework to support the development of ubiquitous applications and services. In addition to aiding the creation of software which is capable of dealing with the above issues; it is important that software remains reliable despite non-deterministic changes within the network infrastructure, and that any approach addresses the need for rapid application development.

We propose a two-tiered approach to meet this need: A lower tier for managing the network infrastructure and an upper tier providing the developer framework. The lower tier uses peer-to-peer interactions to form an overlay for message routing, network organisation, and component management; while the upper tier employs mobile agent technology to hide the underlying complexity from the developer, resulting in a high-level, general purpose framework for developing applications and services.

It is important to note that the need to support ad-hoc and potentially mobile arrangements of devices in ubiquitous environments does not fit well within the traditional client/server architecture. We believe peer-to-peer communication offers an appropriate alternative for building a reliable infrastructure for two reasons: Firstly, its decentralised nature offers good scalability characteristics nature and is therefore better equipped to deal with an unpredictable number of users and devices. Secondly, single points of failure and bandwidth/processing bottlenecks are avoided through removing reliance on individual nodes. Although this decision adds complexity to the programmers task, it is believed that the advantages of this design choice greatly outweigh any disadvantages.

The decision to use mobile agents as the basis for the framework was made on several levels. Firstly, by nature they encapsulate units of functionality, making them an easy to work with abstraction for component based programming [3]. Secondly, to enable users to transfer applications from one device to another

without requiring any pre-installation of code on the target device (for example, from a handheld to a desktop PC when a user enters their office). Thirdly, in a decentralised environment, mobile agents provide the ability to perform distributed searches based on user specific criteria without requiring transmission or retention of all results. Finally, mobility allows application components to dynamically move processing closer to the sources of information they require. This has potential uses for global processing of contextual information [4].

3 Overview of the Design

The high level design of the framework and supporting infrastructure can been seen in figure 1. This section gives a detailed overview of some of the features it supports and the design goals that we are aiming to achieve.

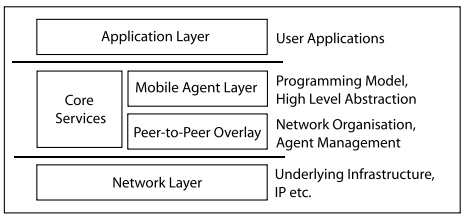

Fig. 1. A high-level overview of the framework

Self Organising Network Infrastructure. Ubiquitous environments are non-deterministic in terms of the number of users they are required to support, availability of resources and connectivity of devices. Due to the high scalability requirements of such environments, it is very unlikely that any one configuration of the above elements will remain static for any length of time. This makes it important that the underlying infrastructure can adapt to changes quickly and efficiently with minimal disruption to users. To facilitate this, we implement a scheme where nodes within the infrastructure are responsible for maintaining information about a small subset of other nodes, reducing the work required to adjust to changes in the environment. This is similar to the approach taken by second generation peer-to-peer systems such as Chord [5] and Pastry [6].

Distributed Processing of Contextual Information. One advantage of using a peer-to-peer model for the network infrastructure is that it provides a way of harnessing resources provided by peers. These resources, which would otherwise be wasted, can be utilised by the infrastructure in different ways.

Idle CPU time across peers can be used to support the processing of the vast amounts of context information that are likely to be generated and used within a wide scale ubiquitous system. Combined with mobility, these features

facilitate the implementation of a distributed engine for context processing whose components can optimise their network position relative to the data they are operating on.

Distributed Storage for User Information. Another peer resource that can be harnessed is disk storage. As we discuss later, personalisation of services is a major challenge that ubiquitous environments need to address. One of the reasons this is a complex requirement is due to the lack of an obvious location to store information about a user. Many users may make use of a communal device or may require access to specific information from a number of independent devices. These reasons make it impractical to store user information at a single location due to memory constraints and fault tolerance requirements. Users are also likely to be mobile, requiring personalisation of services as and where they travel. Requesting that users carry their data with them is both inconvenient and impractical. The framework will address this issue by utilising storage capacity within the infrastructure to provide users with an encrypted, distributed data store, accessible from any node within the infrastructure. We aim to draw upon the experiences of research projects such as Freenet [7] and Oceanstore [8], which have achieved success in providing long-term data storage over volatile peer-to-peer overlays, to achieve this goal.

An Adaptive Protocol Stack. The important advances in technology usually happen at the edges of a network. Often, perhaps for security or data compression reasons, there is a need to develop new protocols. When this requirement clashes with the need for backwards compatibility with legacy components, problems arise for developers. In ubiquitous environments, these problems will be common, as functionality will regularly be provided through the integration of third-party components.

We propose an adaptive protocol stack as part of our framework to solve this problem. This approach involves the use of mobile agents to encapsulate protocols. Using this technique, a protocol stack can be dynamically assembled or altered at runtime based on communication requirements, customised for application specific needs.

For example, consider two components which would normally communicate using HTTP. At some point after deployment, the owner of the first component wishes to implement an upgrade, developing a customised protocol for encrypting data in order to provide security whilst operating in an un-trusted environment. This protocol sits between the HTTP and socket layers. When the updated component wishes to communicate with the second component, the new protocol can be deployed and installed by the second component into its protocol stack at runtime, in order to achieve secure communication. Should the second component refuse to add the protocol to it's stack, the first component can decide to drop its requirements for secure communication and revert to using HTTP with un-encrypted data.

Location Independent Communication between Components. As the basis for ubiquitous computing, communication between distributed application and service components, from a developers perspective, needs to be as straightforward as possible. Once a reference to any component is obtained, interaction should appear as simple as performing a local operation. However, flexibility is also required within the programming model for dealing with a range of failure semantics.

Fault Tolerant Global Referencing for Components. Within the infrastructure, there is a need to aid functionality such as component location, communication and migration at the mobile agent layer. Putting fault tolerant and robust algorithms in place for these features at a lower level reduces the work required of application or service developers.

One of the most common tasks performed within the infrastructure will be to locate a component. From a mobile agent perspective, the suitability of existing algorithms depends very much on the application under development, with existing algorithms exhibiting either a central point of failure or poor scalability characteristics [9]. This is obviously not acceptable within ubiquitous environments.

In contrast, one of the central features of recent peer-to-peer systems is that of efficient distributed object location. Pastry [6], for example, assigns a global identifier to each node and provides guarantees that a message addressed to a node will be delivered to the live node with the identifier closest to the intended recipient.

Our infrastructure will employ an algorithm for locating components that can provide similar guarantees to the above approach, but taking into consideration that due to mobility, components are not fixed to a single node. This requires the additional guarantee that should the node responsible for managing a component's location information fail, there will always be another live node able to provide the same information.

There are two invariants for this algorithm. Firstly, if a component is active, it will always be locatable, regardless of it's position within the network and irrespective of network conditions. Secondly, should a reference exist to a component that has been removed from the network, the infrastructure will be able to correctly detect and report this fact.

In addition to supporting the location of components, similar techniques will also be applied to a components codebase, providing completely decentralised component management at the mobile agent layer. This will result in the removal of all single points of failure exhibited by current mobile agent systems.

Application Specific Approaches to Disconnection Management. In tandem with the above algorithm, the ability to dynamically deploy components without pre-installation of code allows for the development of application specific proxies that can be used to store or react to information should a user lose their connection to the network. Reconciliation can then be performed upon a user reconnecting to the network. In practice, this will involve the specification of

'backup' components within a component's location information, allowing for proxies to be activated and contacted should a component become unreachable.

4 Challenges to be Addressed.

Ubiquitous computing can be regarded as a harder, more complex version of distributed computing. As mentioned above, there are many challenges that need to be addressed towards the goal building a framework for ubiquitous computing. This section gives a taste of some additional issues that need to be taken into account:

No Known Boundaries for Scale. There are no pre-defined scales for modelling ubiquitous environments. Modelling may be required for desks, rooms, buildings, cities, even countries. Traditional network addressing schemes do not scale and for this reason, can not be applied to ubiquitous environments. Work is required to find useable alternatives that are able to support locality within such infrastructures.

Support for Autonomy. This is one of the key issues in ubiquitous computing: How do we build software that is self-managing and able to adapt to its environment, current task or user with a minimum of human interference? One position is that prepackaged applications are undesirable, with research being targeted towards producing mechanisms for the runtime composition of application components based on user requirements and available context information [10].

The Need for Personalisation. Another challenge is the need to personalise services for users. Not only does this include the storing of preferences directly specified by a user but also providing mechanisms that allow applications to derive information from a user's past experiences with similar or related applications. In order to provide truly useful service delivery, information about a user's current context also needs to be taken into account. As discussed above, one of the requirements to support this feature is a facility for storing user information that can be reliably accessed from any node in a network.

Security and Privacy within Ubiquitous Environments. Providing security within a ubiquitous environment is a complex task. For scalability reasons, many techniques used within client/server systems cannot be applied to such decentralised environments, with the emerging paradigm of dynamic trust modelling offering a potential solution. There are many factors that need to be addressed. These include: Providing mechanisms for secure communication (encryption and routing) between components, ensuring the privacy of user data, providing integrity checks that can be used to detect component tampering, letting owners restrict the use of their components and allowing users to control access to the data they make available to applications and services. Providing fault tolerance mechanisms that are able to detect and deal with faulty or malicious nodes are also required.

5 Related Work

While there is no room in this paper for discussion; there are several related projects which focus on providing system support for ubiquitous environments. We briefly mention two:

The *one.world* framework [11] supports a component-oriented approach to application development, focusing on the separation of data and functionality, and exposing change to promote application level handling. Each node in the system may contain many *environments* - the containers for application components and data. Support for the migration of *environments* between nodes is provided to aid service adaption.

Project Aura [12] focuses on user mobility within ubiquitous environments. Each user has a personal *Aura* which marshals the resources required to support their current task within their *physical context*. Through monitoring their *environment*, the *Aura* framework captures information about the user's task, preferences, and intentions, using this information to "shield the user from the heterogeneity of computing environments as well as from the variability of resources." When users move from one environment to another, *Aura* manages the migration of all the information related to their task, and negotiates task support within the user's new environment.

For a complete discussion of related work, we refer the reader to [13].

6 Summary and Future Work

This paper has presented the high level design of a programming framework and supporting infrastructure for developing ubiquitous applications and services. Within this design, peer-to-peer technology is used for network and mobile agent management, while mobile agent technology abstracts over the underlying complexity to provide a framework for fault tolerant application and service development. Key features include robust, decentralised algorithms for location independent component communication; a secure, distributed data store for holding user data; and an adaptive protocol stack for customisable communication between components that eliminates problems with backwards compatibility.

Through the implementation of our previous project, SCI [2], we formed an understanding of the requirements for developing ubiquitous applications, providing the basis for this project. One of the open issues we identified is how best to evaluate solutions to problems within this domain. In addition to furthering our requirements analysis, this is one of the issues we hope to gain insight into through this workshop.

While the limits imposed by available resources will constrain deployment of the infrastructure, we aim to deliver experimental results through a three step process: Reasoning about the scalability of the design by analysing the algorithms it uses for network and component management; performing simulations to examine performance and reliability under a set of hypothetical network conditions; and constructing a prototype of the infrastructure along with sample applications as proof of concept.

Acknowledgements: This work is supported by the EU FP5 GLOSS Project (IST-2000-26070), in collaboration with Trinity College Dublin, Université Joseph Fourier, The University of St Andrews, and through research student scholarships from the University of Strathclyde.

References

1. Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, September 1991.
2. Richard Glassey, Graeme Stevenson, Matthew Richmond, Paddy Nixon, Sotirios Terzis, Feng Wang, and Ian Ferguson. Towards a middleware for generalised context management. In *1st Int. Workshop for Middleware for Pervasive and Ad Hoc Computing, Middleware 2003 companion*, pages 45–52, Rio de Janeiro, Brazil, June 2003.
3. M. Wooldridge and N. R. Jennings. Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, May/June 1999.
4. A Dearle, GNC Kirby, R Morrison, A McCarthy, K Mullen, Y Yang, RCH Connor, P Welen, and A Wilson. Architectural support for global smart spaces. *4th International Conference on Mobile Data Management (MDM 2003), Melbourne, Australia*, 2003.
5. Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.
6. Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.
7. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009, 2001.
8. John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.
9. Pawel T. Wojciechowski. Algorithms for location-independent communication between mobile agents. Technical Report DSC-2001/13, Dpartement Systmes de Communication, EPFL, March 2001.
10. Simon Dobson. Applications considered harmful for ambient systems. Unpublished work.
11. Robert Grimm. *System support for pervasive applications*. PhD thesis, University of Washington, December 2002.
12. J.P. Sousa and Garlan D. Aura: an architectural framework for user mobility in ubiquitous computing environments. In *3rd Working IEEE/IFIP Conference on Software Architecture*, pages 25–31, Montreal, August 2002.
13. Graeme Stevenson. Peer-to-Peer and Mobile Agent Systems: Enabling technologies for pervasive computing? Technical Report Smartlab-2003-02, Department of Computer and Information Sciences, University of Strathclyde, Glasgow, Scotland, June 2003.