eComP: an Architecture that Supports P2P Networking Among Ubiquitous Computing Devices

Achilles Kameas[1], Irene Mavrommati[1], Dimitris Ringas[1], Prashant Wason[1, 2]
[1]*Computer Technology Institute*
Research Unit 3
Ambient Information Systems Group
[2]*Indian Institute of Technology*
Guwahati
India
Email: {achilles.kameas, irene.mavrommati, riggas, prhwason}@cti.gr

Abstract

In the new paradigm of computer use, the computer ceases to exist as an integrated multi-task device, but disintegrates into a task-oriented collection of networked devices. These devices do not resemble to computers yet they have computational abilities. None of these concepts will be realised without appropriate support from communication technologies – P2P networking being the primary candidate. This paper describes part of the research being conducted in the Extrovert Gadgets project geared towards applying P2P computing solutions to the context of networked everyday objects.

Keywords

Ubiquitous computing, middleware, P2P computing

1 Introduction

In the new paradigm of computer use, the computer ceases to exist as an integrated multi-task device, but disintegrates into a task-oriented collection of networked devices. These devices do not resemble to computers yet they have computational abilities. Thus, the computer becomes ubiquitous as computing services are made available to users throughout their physical environment [1, 2]. EU recently set up the Disappearing Computer IST research framework in order to explore an extension of this concept, where everyday objects are enhanced with computation and communication abilities in order to allow people to form new friendly everyday environments [8].

None of these concepts will be realised without appropriate support from communication technologies, which, more than any other contemporary commodity, symbolize the growing availability and influence of new research in everyday life. Today people work in intensively distributed workspaces and need to communicate via heterogeneous networks and assorted devices. The information revolution has created a new paradigm, transforming the way people work and live, shifting the focus from traditional fixed wired networking to mobile ubiquitous networks, ad hoc networks and finally peer-to-peer computing.

Peer-to-peer networking (P2P) unleashes the power of networks and computing machines through provision of unimaginable diversity and flexibility via the distributed computing paradigm. The history of P2P does not go very far, but still there have already been many innovative and successful demonstrations of its power and abilities, from file sharing (Napster, Gnutella), searching, collaborative projects (Seti@home) to the recent application in building intelligent environments. The advent of new technology has brought a multitude of non-traditional devices (PDA's, mobile phones, embedded systems) within the realm of P2P. The next step in this evolutionary line seems to be the "peering" of everyday artifacts from cups to watches to books, pens, desks, etc. P2P networking emerges as a considerable alternative for implementing networks of collaborating ubiquitous devices.

This paper describes part of the research being conducted in the Extrovert Gadgets (acronym: e-Gadgets) project, which is geared towards applying P2P computing to the context of everyday objects. E-Gadgets is being funded under the IST/FET/ Disappearing Computer initiative and aims to develop and validate GAS (Gadgetware Architectural Style), an architectural style for tangible, communicating artefacts. Also it designs and develops the infrastructure required to deploy GAS and sample artefacts enabling the evaluation of concepts.

The GAS compatible infrastructure is supposed to serve as the required binding middleware between the

ubiquitous computing (possibly smart) devices. As a consequence, part of the project-related research focuses on designing P2P networks for everyday objects, in order to arrive at new concepts, techniques and a reference implementation. This research attempts to answer the many confronted issues and to pave the way for new ideas as already described in [4]. As shown in [3] just the existence of smart devices does not suffice to produce meaningful intelligent environment enhancing the daily experience.

The rest of the paper is organized as follows. Section 2 introduces the terminology of the e-Gadgets project, relates it to the abstractions in the P2P domain and defines the problem in both domains and describes sample scenarios to help the reader understand the concepts. Section 3 elaborates on the problem by presenting the issues that arise when examining its various aspects. Section 4 presents the proposed P2P architecture developed in the project. Section 5 presents several studied technologies and Section 6 concludes the paper, while illustrating future work.

2 eGadgets, gadgetworlds and p2p networks

GAS defines the concepts and mechanisms that will allow people, the users of eGadgets, to define and create Gadgetworlds out of eGadgets, and use them in a consistent and intuitive way.

eGadgets (eGts) are everyday tangible objects enhanced with sensing, acting, processing and communication abilities (Figure 1). Each function or service offered by an eGt is manifested as a *Plug*. Plugs may be considered an extension of pipes in P2P terminology. eGts exhibit a dual presence, both in the real and cyber (digital) worlds. In the real world they appear as "tangible" objects, occupying physical space for a certain amount of time. In the cyber world, eGts appear as "digital objects" or software entities, which are instantiated and run on a processing unit. eGts are functionally equivalent to peers in P2P computing.

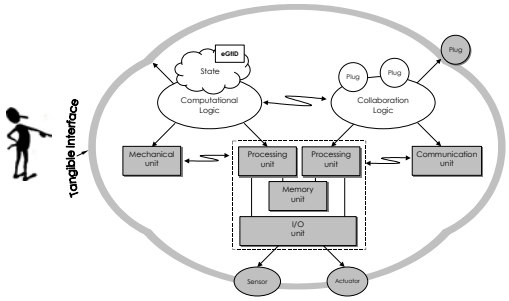


Figure 1: Anatomy of an eGadget (make it larger, across columns)

A *Gadgetworld (GW)* is a specific configuration of associated eGts, which collaborate in order to realize a collective function. Two eGts can collaborate via their Plugs; an association between two compatible Plugs is called a *Synapse*. In that sense, a GW is a set of Synapses. A GW is an enhancement over the group of peers in P2P computing.

2.1 Sample scenarios

In order to demonstrate the concepts underlying the eGadgets-related research, we attempt to implement various everyday activities using eGadgets. As an example, consider a student living in a dormitory, which contains a study desk, a desk lamp, room lamp, clock and a collection of books, some of them on the desk and others on a self. When studying, the student will use the desk as a convenient study place and will use the available books, lights and other objects manually based on his requirements, to make his study experience comfortable.

Inside a GAS enabled student dormitory, the collective functionality of these objects (which have to be eGadgets) can be enhanced to serve the student needs better. At first, the student has to establish a GW among the books, desk lamp, room lamp and desk. When the student opens a book on the desk (wanting to start studying) and the ambient light is too low, then the desk lamp lights up. If the book that is being read moves away from the light source, the light becomes brighter so that the student can still read. When the student closes the book, the light goes off after a short interval. If another book is used for study, the lamp still lights up. In the case the lamp is broken or off the power supply, an alternative light source like the room lamp joins in the GW to provide the required service – ambient light.

More simple or complex scenarios can be supported by the concepts underlying eGadgets, all of which include the intuitive association of eGadgets' services in order to perform a collective function. For example, an alarm clock which gets the daily timetable from a PDA and rings an hour before the lecture, water boiler which starts as the student gets out of bed, lights which adjust to the ambient levels and also to the comfort of any partner who may be sleeping along, etc.

2.2 Definition of the problem

The central technical problem to be resolved in order to have functional GWs relates to the implementation of middleware that will support

eGadget association in an intuitive and robust way. An important module of this middleware is the one that supports networking among eGts. Several other issues, addressed by our research (i.e. people's usage of eGts and GWs, eGt design, realization & packaging, the role of intelligence, etc), are not very relevant to the context of P2P computing and hence will not be discussed in this paper.

3 Refinement of the problem

eGadgets are not used in isolation or for a simple purpose. People associate them into Gadgetworlds, that is, ad-hoc networked clusters that display a collective function. Thus, combined in various ways, these eGadgets collaborate and offer services that are of higher value than the sum of primitive services offered by each eGadget individually.

The factors that refine the problem and shape the solution are both conceptual and technological. The eGadgets project aims to deliver a solution that is generic, modular, scalable, portable and extensible.

3.1 Conceptual factors

Modularity is a key requirement for the infrastructure to be developed, because it enables the improvement of separate modules of the system over time, thus exploiting the advantages of component-based systems. Available technologies and solutions are expected to change over time; the targeted middleware has to be able to encompass the more suitable ones.

Scalability is a priority issue for the eGadgets project. The main target of GAS is to support people in making meaningful uses of eGadget-populated environments, where large number of devices will co-exist without introducing a GAS-imposed constraint on the number of eGadgets that form a Gadgetworld. The functionality of the infrastructure to be developed must be able to scale smoothly from primitive eGadget populated environments, with only a few devices, to very dense ones, where tens of hundreds of devices will share the same resources. The main bottleneck of the system is the network through which the eGadgets communicate. Thus, it is necessary to design a networking module that will not *stress the limits*.

The large variety of daily life objects poses the requirement for *compatibility*: how will compatibility be maintained between implementations of the same object as a different eGadget (for example cups by different manufacturers or various models of same type of lamp with progressively enhanced functionality). A related issue is *extensibility*, which would ensure that new devices could be designed offering new capabilities that older devices could understand and facilitate.

Finally, a generic solution is sought, which will not be constrained to a particular platform. The targeted devices vary vastly in size, computational power and even expected usage in a variety of contexts. Therefore, the system that is designed should not pose severe restrictions, like the assumption of a specific platform. Various manufacturers should be able to implement their consumer solutions on a variety of platforms, not predefined in advance. Thus, eGadgets can be delivered in quite diverse platforms with quite diverse capabilities. Furthermore, devices can be perceived as eGadgets even if they where not designed from the beginning as such. Typical example could be a mobile phone that already has the computational power and ability to communicate via wireless networking, which can be, or can be transformed into, an eGadget if the required software is downloaded to it.

3.2 Technological factors

The main concepts behind the e-Gadgets synapsing concept resemble P2P networking. These have to do with devices that need to communicate and form ad-hoc networks possibly using wireless networking. Moreover, devices need to change locations, while still being able to participate in GWs. On top of these, GWs need to operate in a totally decentralized and self-organizing manner, possibly using a degree of intelligence to enable the cognitive disappearance of computing. The P2P architecture also needs to take into account the hardware constraints of low processing power, limited bandwidth, low memory resources etc. Custom features like power awareness, routing, agents, context awareness [7] etc. also need to be considered to complete the picture. Thus, there exist a number of special issues concerning the networking module that will be selected to implement GAS.

The networking system should guarantee that *roaming* of the eGadget device across different domains or even networks is possible. Given the fact that the eGadgets devices are indented to be the everyday devices of people, they might be quite mobile ones. Thus, for example, an eGadget must be able to function both in its owner's home and office environment without requiring any configuration procedure. Furthermore, since devices are mobile and expected to function even on the absence of infrastructure, *resource discovery* should be an automatic procedure.

Connections between eGadgets cannot be considered persistent in any case. The connected devices can be on the verge of each other's range and thus the connection might be intermittent. *Robustness* against such situations is essential.

The networking system that supports the eGadgets communications is required to be totally *decentralized*. No infrastructure can be expected to be present, since for example the owner of the eGadgets will expect them to function even when he/she takes them along in an excursion, and even if one could be present, there is no guarantee that all eGadgets could reach it at all times. Thus, it is necessary that the networking module can be self-organized in order to offer the required services.

Finally, the networking module of a system that is designed to be platform independent should allow *flexibility* in the platform and programming environment used.

4 The proposed architecture

The infrastructure developed to implement the GAS-related concepts is a true P2P platform geared towards enabling multitude of communication-enabled eGadgets to indulge in peer-to-peer computing. This platform contains the following modules (figure 2):

- Gadget Operating System (Gadget OS)
- GAS OS
- Networking module, named eComP: the extrovert Computing Platform.

This abstraction makes the platform implementing GAS independent of the internal working of the artefact, the communication protocol and the service implementation. In this section, the first two modules are briefly presented, while the design and implementation of the networking module is described in detail.

4.1 The Gadget OS

The Gadget OS is the software layer that offers abstraction from the custom underlying hardware. The Gadget OS handles the resources of the device and hides hardware complexity, by offering access to hardware via standard drivers. At the lowest level every eGadget is a computation-enabled device, thus it requires a Gadget OS to function. Gadget OS makes this P2P architecture generic and extensible since provision of new hardware capabilities requires changes only in the Gadget OS by the manufacturer. The user or the GW designer has only to care about advertising and connecting Plugs.

4.2 The GAS OS

The GAS OS is a software layer that enables eGadgets to be perceived as peers in a P2P system. eGadgets export the services they offer to other eGadgets using Plugs. The GAS OS is responsible for handling the Plugs, advertising them, and thus advertising the capabilities of eGadgets, and forming the connections between the Plugs, the Synapses. The GAS OS layer is there to offer higher-level association services to the eGadgets and to ensure compatibility between solutions offered by different manufacturers. Thus the GAS OS ensures that the eGadgets project solution is an expandable one, since any type of new devices can be introduced in the future and they will be compatible with the existing ones.

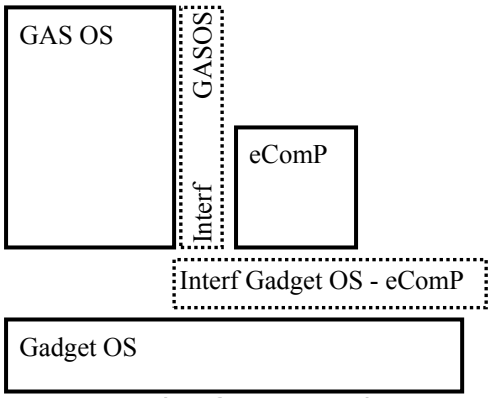


Figure 2: eGadgets platform

4.3 eComP, The Networking Module

The extrovert-Computing Platform, eComP, implements the networking side of the P2P architecture for GAS. It is a decentralized (requiring no fixed infrastructure or the support of any other entity except computing peers) messaging based system abstracting the underlying network and communication protocol and providing services through a well-defined interface. All communication between peers and even within the platform layers takes place via asynchronous XML based messages. The basic services to be required of the module are eGadget and Plug discovery, advertising of resources, message delivery and routing. Towards that end, eComP provides functions for the following:

- Advertise local resources and discover foreign resources
- Listen and reply to resource discovery messages
- Listen for messages addressed to some resource on this eGadget

- Deliver messages to appropriate resources as required by the higher GAS layers via some routing mechanism.
- Static or dynamic configuration via Policies

In the terminology of eComP a resource is any entity that is distinguishable by a unique valid eComP ID. Each eGadget is provided with a universally unique ID that is hard coded in to the device (like the MAC addresses). Resources generated by a peer are local resources for that peer while all other resources are referred to as foreign with respect to that peer. A unique ID derived from the eGadget ID identifies local resources like a plug, gadgetworld or a service.

A user cannot be required to remember this long alphanumeric obscure ID. Hence the mapping of this ID to some familiar, personal and rational identifier (like a textual name for an eGadget) is deemed necessary but has to be taken care of by the higher layers in an application (in GAS this proposition is attended to by the GAS OS). This namespace-based identification scheme makes sure that all the IDs in the platform are unique without the need for complex ID generation algorithms. The eComP ID is central to the working of this platform. All operations are performed with IDs as references to real resources and only during the routing stage the IDs are resolved against the actual network address of that particular resource. Using an ID based routing and naming system allows eGadget to have unprecedented mobility by binding only the ID to the eComP and dynamically resolving the current network address when required. Such a system also solves the configurational problems related to network migration by the eGadget or interface migration (i.e. shift from IEEE802.11 to Bluetooth etc.)

Resources are an integral part of GAS. An eGadget offers some services, which are offered through Plugs that work mainly as a special inter-gadget communication channel. Many eGadgets may join to provide a hybrid service through a new virtual Plug that can be considered as a new resource of the gadgetworld. Thus complex gadgetworlds can be built. eComP has been designed to take care of this resource discovery by advertising and discovering resources.

Another important addition is the concept of local Policies. Policies are defined as an eGadget's pre-defined settings and responses to certain events like advertisement caching, use of memory storage, power, etc. For example a pen/key/wallet eGadget will be supposed to have a scanty memory resource (due to the inherent design constraint of size and negligible requirement) while a book or desk can support relatively large amount of memory. So for a pen eGadget it is impossible to hold a large cache of advertisements and routes other than those very important or those that it frequently uses. On the other hand, desk or book may "decide" to hold lots of information within its cache to facilitate easy & fast discovery of resources. The use of policies helps to configure the "attitude" of the platform on a variety of target gadgets while keeping the code changes to a minimum. Policies can also be dynamic in nature where the context determines the settings in force.

A message is basically an XML data object, which has been optimised to a level more suitable for the case at hand in P2P (or more specifically eComP) by choosing an appropriately designed data structure with negligible parsing overhead. Messages are used for all kinds of communication in eComP from advertising, discovery, routing to the actual data transfer. A message may contain an arbitrary number of sub-sections that are called Elements. In turn, each Element is made up of arbitrary number of sub-sections called Attributes. Messages are transmitted as binary streams and hence can even be compressed or encrypted easily. The ability to inter-convert XML to eComP and eComP to XML messages keeps compatibility with other XML based standards and protocols.

[full justification in the following paragraphs]

eComP abstracts the various functions into different protocols. Since eComP was basically geared towards very small and diverse computing devices, it was required to keep the base set of required protocols and their implementation lean and thin. Only a minimal set of protocols is necessary to be implemented on each peer. The modular architecture allows newer protocols to be added by extending the eComP protocols, or by porting the existing ones. The protocols as well as the architecture within eComP are event based and asynchronous which is a dire need of the ad-hoc and unreliable (in terms of connectivity, QoS, bandwidth, etc.) platforms for which it is envisioned to be implemented. eComP ca be enhanced by add on packages that will add more protocols and functionality to the base set.

The eComP base set is made up of three abstracted protocols. *Discovery protocol* involves sending of discovery messages and receiving of resource advertisements for discovery of anything from peers, pipes, services, groups etc. *Advertisement protocol* is used to advertise local and discover foreign resources. *Routing & Route Discovery Protocol* is responsible for finding a physical route to a resource and appending the route information to the outgoing message. On an intermediate peer, the routing information is updated using this protocol.

eComP has a four-layered protocol stack (Figure 3).

The top layer is the *Interface Layer* that provides a well-defined interface to the outside world to access the networking services while abstracting the underlying details. The second layer is the *Resource Management Layer* (implements Discovery/Advertisement Protocol), which is responsible for discovery and advertising of resources. The third layer is the *Routing Layer* (implements Routing and Route Discovery Protocol), which is responsible for finding dynamic routes, resource ID to network address resolution and finally delivery of the messages through either one-to-one or ad hoc multi hop delivery. As with any network architecture the lowest layer is the *Physical Layer* and is custom implemented for specific underlying network architecture. Each layer provides functionality to the parent layer through well-defined interfaces.

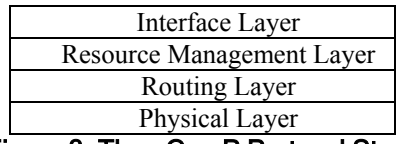


Figure 3: The eComP Protocol Stack

This four-layered protocol stack provides a very efficient solution in a compact package. Between a message delivery operation request at the *Interface Layer and* the final transmission of message bytes via the *Physical Layer* lay a number of sequential events. Firstly, the destination resource's advertisement needs to be discovered. Since resources are expected and designed to seldom change their ID (if at all), any cached advertisements at the resource layer are usually fresh. If no such advertisement is found a discovery message is generated to initiate resource discovery. Once successfully discovered the routing layer is responsible for routing the message to destination by dynamic routing.

Route discovery is a piggyback mechanism. A message traversing between any two eComP modules records the eComP ID of any intermediate eGadget it visits along with the source and destination ID. So by the time a resource discovery response i.e. a resource advertisement arrives at the resource layer, the routing layer already has a latest route to the destination. To deliver the message first a direct end-to-end connection is tried. This step is important since it is not required that the resource advertisement is returned by the resource itself. Hence a shorter or direct route might be available. The eGadgets (? What eGadgets) may also provide a discovery service for known resources by acting as information storehouse. On failure of an end-to-end connection attempt the message is transmitted to the destination using the multi hop ad hoc route just discovered. Since this route is made up of IDs and not network addresses that are assumed to be dynamic, it is more stable even if some intermediate node fails, drops out or changes networks or interfaces. To deliver the message to the next hop the routing layer needs to resolve the current network address of the next eGadget in the list. Since this eGadget is most probably in the vicinity and maybe even in the same gadgetworld, this information is usually present in the cache. Hence each eComP needs to resolve only the next eGadget to send the message to. Since the ID of each resource is fixed and the network locations are resolved only at the routing stage, advertisements never get stale only the routes do.

In case of some runtime failure like eGadgets moving out of range the route can easily be updated using local cache contents of that eGadget where the failure occurs or again a route discovery takes place. While making this design, the only assumption we have made is that each eGadget will have knowledge of at least its neighbours to which it can connect directly. This piggyback routing scheme allows for low overhead route discovery since intermediate eGadgets may also refresh their routing cache while transmitting a message, based on their Policy settings. We believe the system is pretty robust to network failures and is more suited to the project than any other architecture researched. The routing scheme is not completely flat since eGadgets cache routes and hence each gadgetworld turn into a local databanks of routes, and resource information in the form of advertisements. Enhancements to the current system in form of some context aware mechanism, see [7], or other intelligent ideas like mobile agents will be a target of future research.

A layered model helps in making this system portable and can easily be implemented on various wireless network architectures like Bluetooth, Infrared, and IEEE 802.11 etc. We have presently implemented and tested the current version using IEEE 802.11 standard.

4.4 System implementation

The implementation of our system has been done using the Java Programming Language. The implementation is compatible with the Java 2 Micro Edition, CDC Profile, thus can execute on quite resource constraint devices.

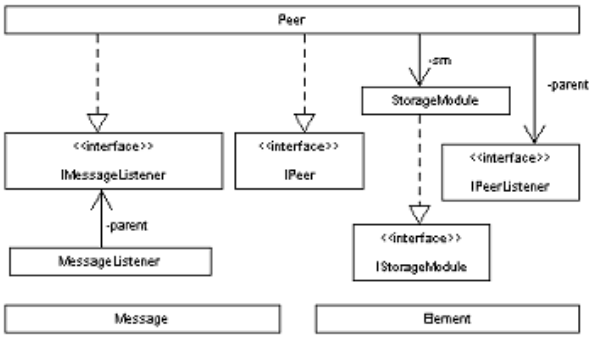
The base class of the eComP system is the Peer. It implements all the functionality that is required in the eGadget environment. The Peer is the representative of

the eGadget in the P2P world.

The Message class is the minimum message exchanged in the eComP system. It is made up of a collection of Elements. Elements have attributes and data. This sets up a XML message without the need for parser, respecting the limited computing power of the devices in question. There are methods like `getElementByName()` and `getAttributeByName()` for convenient access to the data. All data contained in the tags is binary. The Element class corresponds to elements and is made up of an array of elements itself, which are called attributes.

The storage service required by the eComP platform has been abstracted, since various gadgets will have varying storage requirements. It provides the functionality required in order to save, format and retrieve data from the storage module in a manner that is suitable to the file system of the constraint devices, the eGadgets.

Different policies can be implemented on the various devices according to their limitations. Thus a device with low memory may have a policy to reject all unnecessary caching and a device with low processing power may reject all advertisements to other devices.



4.5 Scenario implementation

Coming back to the scenario describe earlier, it is now possible to describe how this can be implemented in a GAS enabled environment. Since the objects that appear in the example are all GAS enabled, they offer some services through Plugs. The books have light sensitive surfaces that enable them to return the level of luminosity on their surface. Also, they have sensors that can detect whether they are open or closed. These two services are exported via two Plugs. The student desk has a Plug that can be connected to a number of books and get event notification about the state of the books. The desk lamp has a Plug that accepts data that can control the light it emits. The desk Plug is connected to this in order to inform the lamp about the state of the books on it. Having the Synapses in place, the user inhabits an environment that offers advanced services to him/her, i.e. studying is concentrated on the mental activity without the overhead of setting the required conditions.

The implementation of the above scenario utilizing the eComP features will be like this: Each object, like the desk, the books, etc, will be perceived as an eGadget in the student dormitory and each service offered by an object will be exported though a Plug. Each entity, eGadgets and Plugs, will have a unique network ID and communication between them will be based on the eComP. The eComP platform will provide the necessary extensibility and scalability, since it offers the ability to have a decentralized, ad-hoc network of devices that can be added/removed, and yet can still be discovered and utilized.

Also, forming the above Gadgetworld is quite simple and fast. The produced outcome, on the other hand, is advanced comfort for the user's environment. The intelligent aspect of the environment will be perceived by the user both when the Gadgetworld is serving him/her and when for some reason the Gadgetworld appears to be out of order, for example if the desk lamp is not able to provide enough light. While the service is running smoothly, the user will notice the improved conditions under which he/she works. When the service offered to the user is on the verge of failing due to some unexpected condition, GAS OS will locate a device that can offer a compatible service. Then, the room lamp will be the device that will adjust the ambient light level to become adequate.

5 Conventional systems studied

The solution that the eGadgets project is offering is designed to be modular. Thus, for the networking module a number of existing solutions might fit in and hence where studied. A short analysis for each of them, along with a *suitability statement* follows.

5.1 Java RMI

Java Remote Method Invocation, [11], allows the creation of distributed objects using Java. [no new paragraph here]Using RMI, eGadgets could export their services as remotely accessible interfaces that could be accessed by other eGadgets. RMI, however, is centered on Java. A non-Java implementation is not possible. Thus, portability and independence of platform are sacrificed.

5.2 Jini

A Jini system [10] is a distributed system based on the idea of federating groups of users and the resources required by those users. [no new paragraph here]Jini-based eGadgets can be perceived as resources that are dynamically added and removed to the system without any required administration and in the same time it is possible to be located by both people and software. The Jini system, though, is Java technology-centered - assumes that Java is the implementation language for all its components and the communication can only be Java RMI. Also, it utilizes of a Lookup Registry in order to enable discovery of the services, which sets the necessity of having a central server somewhere in the environment. Therefore, the Jini approach proves to be unsuitable for the eGadgets environment, since portability to any platform and independence of programming environment is sacrificed and the resulting system is no longer a decentralized one.

5.3 Jxta

Jxta is a set of open, generalized P2P protocols, which allow any connected device on the network to communicate and collaborate as a peer, [12]. Project Jxta comes closest to be selected as the networking module for the eGadget framework. Each eGadget can function as a Jxta peer providing services through its Plugs. The Gadgetworlds can be thought of as an analogy to groups in the JXTA terminology.

But the currently available version of Jxta has been developed with the Java Standard edition in mind. Though ports to other Java editions and implementations on other platforms are available, most of them lose some functionality. Jxme, which is the port of Jxta to Java Micro Edition, is a bare-naked implementation and above that requires the services of a Jxta relay to function as a P2P peer.

5.4 Justification

The amount of stability and dependability required from the P2P architecture is beyond any of the P2P implementations that were studied (it is acceptable to have your PC *hang* once in a while but having your everyday objects *misbehaving* due to an unavoidable crash is certainly out of the question).

Also since there are numerous platforms and hardware involved it isn't very practical to rely on porting a non-custom architecture. Some of the functionality like the high level of security, logging, caching is not very necessary at the present moment, where it is more important to have a generic evaluation of the system. So with the before–mentioned design criteria in the mind we decided to have the best of every world by implementing our own architecture while adopting some of the excellent work done within the Jxta and other project.

6 Conclusions and Future work

The eGadget project is also looking into the role of intelligent agents, which give the eGadgets the smartness to form gadgetworlds independent of human control. For appreciating this idea, consider the scenario again. For some reason let the desk lamp go faulty resulting in no light for the student to study. The desk eGadget need to locate some other eGadget, which provides the same kind of service that is light service. Hence somehow eGadgets need to understand what a service is in terms of its utility and experience for a human. This brings Artificial Intelligence or more specifically machine understanding into the domain of the project. In this new and smart GAS, Agents will function as the brain of the eGadget. The networking module can utilize the services of mobile agents for the route discovery and resource location making it too inherently intelligent. How and in what ways this can be of any benefit to the GAS is a matter of future research.

7 References

1. Weiser, Mark. Some Computer Science Issues in Ubiquitous Computing. ACM, 36(7), July 1993, pp 75-84
2. Abowd, Grocery. Software Engineering Issues for Ubiquitous Computing. ICSE99 pp 75-84
3. Callaghan, Colley, Clarke, Hagras. The Cognitive Disappearance of the Computer: Intelligence Artefacts and Embedded Agents. Workshop on "Cognitive vs. Physical Disappearance of Computers", i3 Spring Days, Porto, 23-25 April 2001
4. Kameas, Mavrommati. Interacting with ubiquitous computer applications: issues and methodology. 1st Panhellinic HCI Conference, Patras, December 7-9, 2001
5. Shafer. Ten Dimensions of Ubiquitous Computing. Keynote presentation at Conference on Managing Interactions in Smart Environments, December 1999.
6. Kortuem, Schneider, Preuitt, Thomson, Fickas, Segall. When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad-hoc Networks. 2001 International Conference on

Peer-to-Peer Computing (P2P2001), Sweden, 27-29 August 2001
7. Gold, Fokus, Tidhar. Towards a Content-based Aggregation Network. 2001 International Conference on Peer-to-Peer Computing (P2P2001), Sweden, 27-29 August 2001
8. The Disappearing Computer initiative: http://www.disappearing-computer.net/
9. e-Gadgets website http://www.extrovert-gadgets.net
10. "Jini Architectural Overview", Technical White Paper, Sun Microsystems. http://wwws.sun.com/software/jini/
11. "Java remote method invocation - distributed computing for java", White Paper, Sun Microsystems http://java.sun.com/products/jdk/rmi/index.html
12. Project JXTA web site http://www.jxta.org/