Proceedings of the Symposium

2nd PERADA Workshop on Pervasive Adaptation

A symposium at the AISB 2009 Convention (6-9 April 2009)
Heriot-Watt University, Edinburgh, Scotland

Symposium Chairs
Prof. Nick Taylor
Prof. Emma Hart

Published by SSAISB:
The Society for the Study of Artificial Intelligence
and the Simulation of Behaviour
http://www.aisb.org.uk/

ISBN - 1902956796

2nd PERADA Workshop on Pervasive Adaptation

A one-day symposium at AISB 2009 (6-9 April 2009).
http://www.perada.eu/perada-events/20-symposium-pervasive-adaptation-aisb-2009

PROGRAMME CHAIRS

Prof Nick Taylor, Heriot-Watt University, UK
Prof Emma Hart, Napier University, UK

INTRODUCTION

The vision of a technology-rich future in which computing is truly ubiquitous poses significant engineering challenges. Multitudes of heterogeneous devices will be required to operate in an ever-changing networked environment which has no central control point or controller. These dynamically created systems will have to continuously organise and adapt; adaptation of individual components will lead to adaptation of the system as a whole and to the emergence of new system behaviours. For this vision to be realised, new approaches to both hardware and software are required to endow systems with the capability to achieve these goals. Systems in this context may range from small ad-hoc collections of mobile devices in a local environment to massive collections of devices which self-organise into tribes of societal artefacts. Such systems will exhibit emergent behaviours, arising from interactions within the system. Behaviours will emerge over a range of timescales, ranging from short to very long and should be capable of adapting as the underlying network itself adapts. Emergent behaviours should be useful; how can this be controlled and quantified?

Secondly, there will also be an emergence of information within the system; this occurs at a number of different levels: Local information is held by individual devices - as networks of devices are formed, a meta-level of information emerges which is held by the network as a whole; this comes from sharing, aggregating, comparing and interpolating information and experiences held by individuals. This 'meta-information' is a rich source of information, which can then be exploited by the network. However, it also raises a number of questions. For example, how can we encapsulate and exploit that information e.g. to influence individual local decisions? How do we decide at the local level what information to remember and which to forget taking into account the effect on the global knowledge? How do we aggregate and reconcile possibly conflicting information held by individuals across the network?

This workshop, the 2nd in a series of workshops organised by PerAda, the EU Coordination Action for Pervasive Adaptation, focuses on how emergence is achieved in Pervasive Adaptive systems. The 1st PerAda event was held in Venice, October 2008 at the Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems. Five talks will be presented at this workshop, three of which were invited from experts in the field. The papers address topics ranging from practical proposals for facilitating emergence via architectures, agent and communication protocols, through to future visions of biocybernetic systems and systems which enhance quality of experience for users. We thank both the authors and invited speakers for contributing to the 2nd PerAda workshop AISB and their role in making the workshop a stimulating and successful event.

TOPICS

Topics of interest include but are not limited to:

- New paradigms for capturing and exploiting memory in pervasive adaptive systems
- Algorithms and methodologies facilitating and studying emergence in pervasive systems
- Emergence of behaviour over multiple time-scales
- Quantifying emergent information in pervasive systems
- Exploiting local and global information in a pervasive system
- Aggregating information in a pervasive system
- Security and trust issues associated with the information held in a pervasive system
- Hardware issues

PROGRAMME COMMITTEE

Professor Damal Arvind, University of Edinburgh
Professor Dave Corne, Heriot-Watt University
Mr Kevin Doolin, Waterford Institute of Technology, Ireland
Dr Yanguo Jing, London Metropolitan University
Dr Christian Jones, University of the Sunshine Coast, Australia
Dr Mark Neal, Aberystwyth University
Professor Ben Paechter, Napier University
Dr Jon Timmis, University of York
Dr Roger Whitaker, Cardiff University

Table of Contents

Pervasive Adaptation for Enhancing Quality of Experience

Arvind Bhusate and **Jeremy Pitt**[1]

Abstract. A visit to a public collection is potentially one of the most effective and entertaining ways of knowledge acquisition open to the general public. Such collections are nevertheless under increasing pressure, for various commercial and technical reasons, to improve the qualaity of experience (QoE) afforded to users. In this work, we propose to (try to) enhance QoE in a visit to a museum through a combination of a sensor-saturated environment with intelligent decision-making and policy-based computing, to deliver personalised, context-sensitive services to the visitor. The fusion of sensor data and user actions to adapt the behaviour of, for example, displays and exhibits is addressing a specific challenge of pervasive adaptation, i.e. how to coordinate the complex interaction of users, devices, and sensors to deliver an improved service or experience.

1 INTRODUCTION

A visit to a public collection, such as a museum, is potentially one of the most effective and entertaining ways of knowledge acquisition open to the general public. Such collections are nevertheless coming under increased pressure to improve the quality of experience afforded to visitors. The reasons for this are due both to competition from alternative forms of entertainment, and also from the expectation levels raised by those forms. We believe though that the appropriate response is no to attempt to make the visit a comparable 'experience', but instead to capitalise on potential opportunities in exhibition delivery: with respect to the visitor-museum relationship, the visitors-museum relationship and finally the visitor-exhibit relationship [3].

With respect to this latter relationship, in any visit there will be those exhibits in which the visitor is more interested, but can't interact with because restrictions are in place, e.g. it is behind a glass cabinet. Even if there is any form of interaction or direct experience with the exhibit, it is currently un-recorded; yet this direct experience is potentially of most value to both the museum and visitor. The visitor's enjoyment, engagement, learning and recall are all at risk as the dialogue between the visitor and exhibit is limited to mainly viewing and reading. We need to find ways in which this limited dialogue can be expanded.

Possible ways of enriching this dialogue include tracking visitors to provide personalisation, encouraging visitors to browse and explore the exhibits in a meaningful fashion, and finding an appropriate trade-off between directed, structured or didactic routes through the exhibits, and open, unstructured paths which end up with the same pedagogic content. There is also a requirement to make subject matter tangible: in addition to digital content (which would be readily available over the internet anyway) the visitors have to be exposed to the 'real things'. We propose to 'follow' the visitors in their passage through the museum, complement digital information and/or virtual exhibits with real life replica models, and personalise the interaction, and so improve the overall quality of experience accordingly.

We propose to use a combination of pervasive computing and policy-based computing to (try to) enhance *quality of experience* (QoE) in a visit to a museum, through a combination of a sensor-saturated environment with intelligent decision-making with respect to deontic policies, in order to deliver personalised, context-sensitive services to the visitor. In this paper, we begin by offering an appropriate definition of QoE; specify the architecture of a platform to provide, measure and evaluate QoE; and describe the decision-making support which adds the 'intelligence' that in its interpretation of sensor data and user activity adapts the display and exhibit behaviour in order to enhance QoE (cf. [9]).

We then describe the iCars Exhibition, a combination of a sensor-saturated environment with intelligent decision-making and policy-based computing, to deliver personalised, context-sensitive services to the visitor. The fusion of sensor data and user actions to adapt the behaviour of, for example, displays and exhibits is addressing a specific challenge of pervasive adaptation, i.e. how to coordinate the complex interaction of users, devices, sensors, and intelligent systems to deliver an improved service or experience.

2 QUALITY OF EXPERIENCE

2.1 Background

Quality of Experience (QoE), sometimes also known as "Quality of User Experience", is a subjective measure of a person's experiences with a place, object etc. More precise definitions include:

- the degree to which a system meets the target users tacit and explicit expectations for experience [2];
- the measured level of quality of a particular user experience when compared to a specific target, using a specified metric and method or tool [8].

An experience could be of watching a video over the internet, dining in a restaurant and so on. User experience is context-dependent and tends to include wider human experience dimensions such as pleasure, fun, and other emotions [2]. In this work, a user's experience is expressed as emotions, attitudes, thoughts and perceptions felt by users across the usage life cycle and which affects the users behaviour.

Many other studies and research have been carried out on trying to understand, define and measure such a subjective area of study. Beauregards et al [2] describes Quality of Experience by breaking

[1] Department of Electrical & Electronic Engineering, Imperial College London, UK, SW7 2BT; email: {a.bhusate,j.pitt}@imperial.ac.uk

it down into two distinct parts, the first being the User Experience and the second being User Experience Quality. User Experience is described in the form of a loop, which is made up of three major components: perceptions, emotions, thoughts & attitudes and finally behaviour.

Alternatively, Alben [1] defines several criteria, which include: understanding of users, learnable, usable, needed, mutable, effective design process, appropriate, aesthetic, and manageable. Corrie et al [5] discusses four domains, which include: task domain, needs domain, services domain and technology domain. Finally, Forlizzi and Battarbee [7] uses five aspects of experiencing a product, which are: physical, sensual, cognitive, emotional and aesthetic.

2.2 QoE for a Public Collection

All of these studies are informative, but demonstrate that a definition of QoE is highly context-dependent. To derive a workable definition for our puposes, we need to

- Define the experience: This involves describing a typical yet detailed scenario (situated in an exhibition space), which aids to set the physical environment, describes the sorts of exhibits we envisage the visitor will encounter and describes the type of visitors this experience is aimed towards.
- Understand the experience: By identifying the roles, goals, relationships and interactions of the, and between the, visitors, objects and environment we can identify the factors which can affect their experience.
- Design and realise the experience: Using the above understanding we can create a requirements specification of the components which need to be developed to realise this experience.
- Measure the experience: Finally using the factors we can identify the measures/criteria and create a hypothesis to evaluate against.

In the following sections, we briefly consider each of these steps.

2.2.1 *Define the Experience*

We envision a system in which there is a sensor-saturated environment, where the environment is equipped with cameras, the visitor is equipped with an indentifying device, and the exhibits equipped with a range of sensors to determine users' actions, e.g. those which change the state, orientation or movement of the exhibit. This envisionment is illustrated in Figure 1, and foreshadows the iCars Exhibition presented in Section 5.

2.2.2 *Understand the Experience*

Assuming this 'experience envisionment', which is quite abstract, we derive from it more concrete components: visitor, exhibition space and exhibits. For each we identify their roles, goals, relationships and interactions amongst and between one another respectively. Finally the identify the measurable factors constituting QoE.

Visitor. This exhibition space is designed and created for children (both male and female) between the ages of 11-13. It addresses issues which are dealt with in school curriculums today so that the children can relate and connect with the information. The children have the opportunity to browse the exhibition and explore the exhibits. The children are empowered by a museum (in this case the exhibition space we have defined) which defines what they can and cannot do (using powers and permissions, the basic building blocks of deontic policies) with regards to interactions with the different exhibit types.

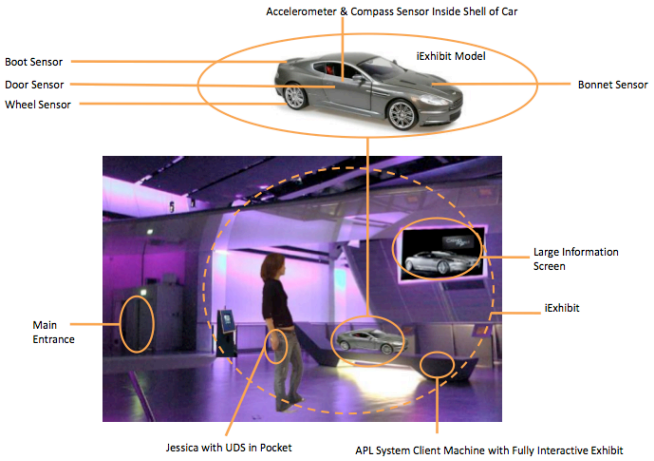


Figure 1. Interactive Exhibit

Exercise of these powers and permissions is dependant on their usage and behaviour. The children also have an opportunity to interact with one another to obtain a shared experience, however we do not model or define any governing mechanisms for this.

Exhibition Space. The exhibition space is browsed by the visitors. The aim of the exhibition is to educate children in an entertaining manner about the history and culture of cars and how they connect to important issues of security, safety & security and environment. The exhibition space houses many types of exhibits and has an overall quality associated with the many experiences it provides. A museum housing an exhibition space has the responsibility and obligation to manage and organise exhibits and visitors.

Exhibits. The exhibits are explored by the visitors. Amongst the multiple types of exhibits there are actually three main types we are interested in evaluating the QoE. Each exhibit may be traditional (non-interactive), partially interactive and fully interactive. Thus each exhibit provides an experience and has an associated quality. Exhibits have a responsibility to serve content to a visitor. However, certain exhibits can be empowered which gives them options of how it can serve its content based on the visitors empowerment levels, actions, usage and behaviour. For example an exhibit may decide that a users behaviour has been excellent and therefore should be rewarded. This would actually mean that the exhibit itself decides to increase the permission levels for the user as it has gained trust. Since this is based on trust it can potentially put the security of the exhibit and other similar exhibits at risk.

Factors. Using this scenario together with the identified visitors and objects, the relationships and interactions amongst them we define the factors which could affect the several smaller experiences present and the overall experience quality for a visitor. This is an essential step in realising and evaluating this experience. In this context, we determine that the quality of experience is made up of three main groups, which are:

- human perception factors: ambience, aesthetics, comprehension, content, accessibility, tangibility, interactivity, usability, and collaborativity;
- technology factors: functionality, personalisation, security, trust & privacy, response time, synchronisation, decision, intelligence, latency, and errors;
- human psychological factors: comprehension, emotions, stress, anxiety, patience, fatigue, behaviour, and expectation & demand.

2.2.3 *Design and Realise the Experience*

The substance of this section is described in the following sections, on the system architecture (Section 3), decision support (Section 4), and their incorporation in the interactive iCars Exhibition (Section 5).

2.2.4 *Measure the Experience*

The measurement of experience, according to factors identified in the three main groups above, is an outcome of the evaluation trials and experiments, and is addressed in Section 5 below.

3 SYSTEM ARCHITECTURE

This section presents the overall system architecture, called the APL platform. It is generic in design, but is targeted at a specific exhibition, i.e. the iCars Exhibition. This comprises a server and four constituent exhibits: three cars, one non-interactive, one partially interactive, and one fully-interactive; and an interactive quiz exhbit. However, the APL platform is generic and requires the designer to instantiate the APL platform for a specific planned exhibition.

3.1 APL Platform: Logical View

The APL Platform's purpose is two-fold. Firstly, it presents a means to provide an experience and capture this experience. By offering a richer experience it aims to of enhance the quality of that experience in a museum, by providing interactivity through the unobtrusively embedded sensors and a mechanism to dynamically deliver information in various forms governed by user/system policies and intelligent decision making.

Secondly, it is used to help evaluate the users experience by collecting, processing and storing user interaction data later to be retrieved for analysing their experience. The APL Platform keeps track of the users exploration in terms of the exhibits they have viewed, the duration they have spent at each, their focus at the time of exploration, and the interactions which took place between the exhibit and them. It also acts as a guide to aid the user in their learning. This functionality allows the user to focus on just exploring and learning and not worry about what they have seen, what they still need to see etc. This opens avenues for building a longer lasting relationship between the museum and user as they can continue their visit from where they left off.

Therefore, at a high level of functional abstraction, the APL Platform provides a user with a variety of personalised services and interaction mechanisms in a ubiquitous computing environment. It provides the administrator a means to monitor and evaluate a users experience in an unobtrusive manner.

The APL Platform is (logically) made up of software components, collectively known as the APL System, and hardware components, which in the iCars Exhibition include the APL Exhibit Extension, ID Devices, video cameras and quiz pads. The APL System software consists of two parts: the server-side component, namely the APL System Server and the client-side component, namely the APL System Client. The client side entity directly provides the personalised services and content using its decision-maker in conjunction with fuzzy inference systems plus defined policies (see the next section). However the server side component is working together, along with other user-defined policies to cater for this personalisation of services and content. The APL System Client communicates with the APL System Server and the Exhibit Extension (if any) using Bluetooth.

3.2 APL Platform: Physical View

The *physical* architecture of the APL platform, i.e. in terms of machines, for the iCars Exhibition, is illustrated in Figure 2. Recall there were four constituent exhibits, therefore one machine for each, each running a separate instance of the APL System Client, and each with its own local policies and content database, display screen to present information about the exhibit which it is associated with in the form of text, images, videos and CG models, dependant upon the type of exhibit, video camera embedded in the display screen (to be unobtrusive) which captures the user's focus and attention (its operation is automated and is triggered by the users presence). In the case of Exhibits 3 and 4, which are interactive, there is an Exhibit Extension which includes all the sensors and hardware required to implement an interactive exhibit. The visitor carries an ID device (IDD) which is used to make the various components of the platform aware of the presence of whoever is logged into the system with it. It is carried by a registered user to identify their presence. The IDD sends data via radio to the APLSystemClient.

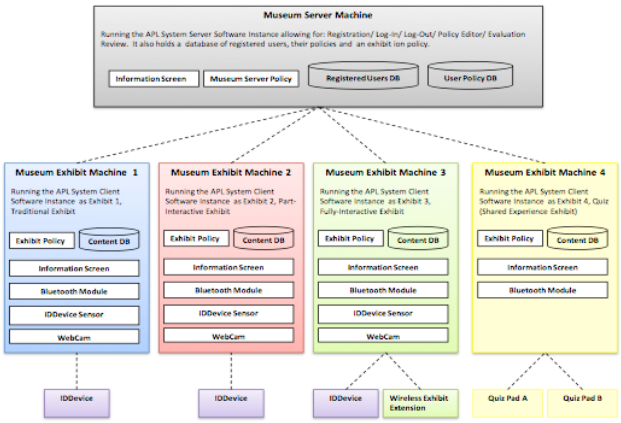


Figure 2. iCars Exhibition Physical Architecture

In more detail, the function of each APL Platform (software and hardware component) is as follows.

The APL System Server allows for user registration, user log-in/log-out, and user/system policy creation and editing with a built in policy configuration toolkit. Registration allows a user to register to the APL Platform, where an account (folder and UserID) is created for them. This account initially stores a newly created policy and will hold future SessionInteractionData and VideoData files. Along side this, their name, DoB and UserID are also stored in a Registered Users Database. Log In simply associates an IDDevice to a user and broadcasts their policy over the Bluetooth Communication Link to all available exhibits allowing them to start exploring. Log out simply disassociates a user from an IDDevice after they have finished exploring and makes the IDDevice available to another user.

The Policy Configuration Toolkit allows for the creation and modification of user/system (exhibit) policies catering for the personalisation and added value of the service received by the user. The APLSS also manages and monitors Exhibits (APLSystemClient instances) and their policies.

An APL System Client instance is able to detect the presence of users that are registered and logged in via their IDDevice. The APLSC is receives user policies from the APLSS and then loads them ready to be used when the corresponding user is detected. It creates a session for this detected user and then captures the users in-

teractions (particular to the fully-interactive exhibit) and video data. It also provides information using a variety and mixture of multimedia content including a virtual CG model dependant on the exhibit type. The automation and control of these processes is carried using an intelligent decision maker which uses both the user and exhibit policies.

APL System Client Software retrieves and loads appropriate user policies dependant on the user(s) detected. It also collects data from the user exhibit interactions, processes it and encapsulates it into the users session file. After the user leaves the exhibit, their policy and session file is sent to the APL System Server Software. It also makes decisions about what content to serve and when, based upon the users actions, overall behaviour and several policies including user and exhibit policies. It provides information content about the exhibit to the user via its associated information screen.

4 DECISION SUPPORT

4.1 APL System Client 'Intelligence'

We aim to enhance QoE through some 'intelligence' in the fully-interactive exhibits. This is derived from fusing the behaviours of users derived from sensors with other data, using a fuzzy inference system (FIS). The output of the FIS, in conjunction with user policies, is interpreted by the Decision Maker, for example to change the view on the associated display. This is illustrated in Figure 3.

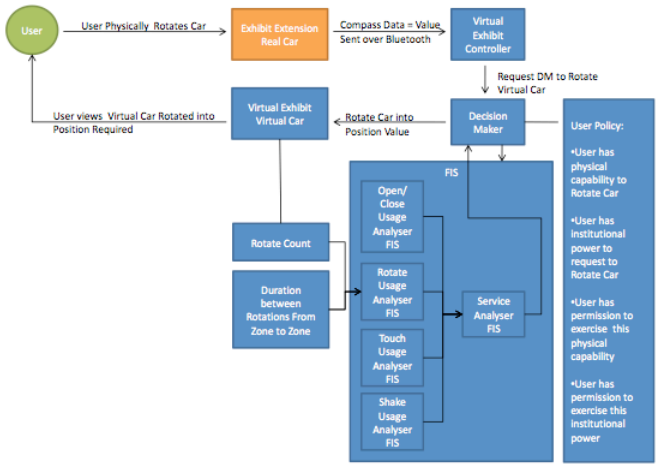


Figure 3. Use Case for User 'Rotate Car' Action

In the rest of this section, we review the three main components of the decision support system: the APL policies, the fuzzy inference system, and the decision maker.

4.2 APL Policies

An APL Policy can be written for several actors of the APL Platform. These policies encapsulate setup information for each entity. They also indentify and define the responsibilities in terms of obligations and sanctions for the system components; and actions in terms of physical capability, institutional power and permissions for the subjects (users). In particular:

- a policy represents the museum server in terms of its setup and main responsibilities, e.g. delivering system and subject policies as and when requested, keeping track of users movements etc.
- a policy represents the exhibit in terms of how it must be set up, its responsibilities which importantly govern what reactions to carry out on certain actions from the user.
- a policy represents the user in terms of the actions they can/cannot perform whilst browsing or exploring an exhibit. A user can only have one policy file associated to them.

There are a number of alternative policy languages available (e.g. [6, 10, 11]), but these were aimed at different domains (management, security, web services), none quite captured this full range of representation, and did not distinguish between being physically capable, being conventionally empowered to bring about (his/her actions had conventional significance in the context of the museum), and whether or not the user was permitted to perform either physical or conventional acts.

For this reason, we designed our own policy language, based on a museum/exhibit ontology of actions and objects, policy information model, dictionary and grammar, collectively defining the APL Policy Language.

This language is used to write APL Policies. describe and in turn govern the operational (functional) behaviour of an APL System Client (Exhibit). This will allow for control of the content that will be presented to a user, helping to protect the exhibit from any potential harm etc.

There are currently two types of APL policies: subject policies and system policies. The subject policies contain details about the users (visitors or administrators), and also defines their access control for applying actions on exhibits. The details of the visitor include their personal information and the actions they are permitted to apply using the concepts of physical capability, institutional powers and permissions to exercise the former two. The system policies contain details about the various system components, and defines the systems control for automation of tasks such as delivery of content, denial or service, warning or aiding between system user and system system type interactions. The system uses obligations and sanction policies to control the delivery of content to the user, the general running of the system and finally what sanctions to apply when the user breaks a policy. The elements of either policy type can be specified in any order. The elements of either policy types can contain many other policy types. A system has many default system policies. Some system policies can be overruled if stated in the system policy by an administrator policy for example. Access control is a good example of this.

We use in our policies the distinction, which is standard in the study of legal, social and organizational theories, between three types of 'can'. This is the physical capability to manipulate objects (e.g. manually rotating an object), the institutional power to establish conventional facts (e.g. a command to rotate an object), and the permission to perform either type of action.

- *Physical Capability (Can_1) Policy*. We assume the user always has physical capability. However, they might not have permission to pick up and perform a physical action (rotate, touch, open/close etc) on the physical model of the exhibit if they do, it could sound an alarm alerting a nearby member of staff to attend the scene. Or the virtual exhibit could simply not respond leading the user to boredom.
- *Institutional Power (Can_2) Policy*. Power to request to rotate the CG model of the car (create a conventional fact), whether these requests are acknowledged and served by the system depends on if the user has the permission to exercise this power, if so the model will move appropriately. If this power is not granted the requests

cant be made, however will be logged as part of the data collection. The CG model will be frozen.

- *Permission (Can_3) Policy*. The Museum gives the user permission to exercise his/ her physical capability and institutional power on the physical exhibit in order to control the virtual exhibit i.e. open/close, rotate, touch and shake.

Permission can apply to both the physical capability and the institutional power. You can have permission to make a request, and you can have permission to pick up an object, or not, in both cases. The difference is, we can intercept in the system itself, the institutional power and permission, and grant/withdraw these accordingly. If the user has permission to make a request, a rotate request for example, the request that is made will be honoured and the relevant information will be provided (rotation of the CG model will occur, textual information will be displayed, video will start etc. at appropriate positions (zones) the model is rotated into). If the user does not have the permission to be able to perform *can*_2, the request won't be honoured and the CG model will appear frozen.

We can *not* affect the physical capability of the users (short of "getting medieval" on transgressors); we can however monitor and control the permission to physically interact with exhibits, but can only do anything about it if we alert a Docent (i.e. back in the "real world", outside the system itself).

We will grant institutional power to specific users one at a time, so their movements do actually count as requests to the controller to perform actions of the appropriate type. This will also guide the user through the experience that was intended (so as for the user to systematically learn and query the exhibit) rather than just letting the user randomly explore it.

4.3 Fuzzy Inference System(s)

Fuzzy Logic is a formalism which facilitates reasoning about imprecise facts, uncertainties, and value judgments. Fuzzy Logic is the basis of fuzzy inference systems, although there are different types of fuzzy systems as there are various different ways in which outputs can be determined.

In general, to build a fuzzy system, an engineer might start with a set of application- dependent fuzzy rules as specified by a domain expert. Fuzzy rules are expressed in the form "if ... then ..." that convert inputs to outputs, where both inputs and outputs are members of fuzzy sets (a fuzzy set is a set in which objects are members to some degree).

Given a set of such rules, it may be that a particular range of inputs fire (activate) any given subset of those rules. The rules which are fired then contribute proportionally to the fuzzy output: this is calculated by applying the implication method of fuzzy logic to the activated rules and aggregating all the results. The process of defuzzification converts the aggregated output into a 'crisp' value (the usual method is a centroid calculation, i.e. finding the centre of an area under a curve).

This entire process, called fuzzy inference, thus converts quantitative inputs into a precise output using qualitative statements.

Several FISs are used to provide the user with a service (present information to the user), help and finally protect the exhibit if negative intentions are sensed by inferring a users usage type and behaviour. The FIS is used to determine if: the user needs help, is mistreating the exhibit, or should be granted permission to be able to apply more advanced actions.

If the user is determined to be mistreating the exhibit or not adhering to boundaries specified by instructions or warnings etc, the FIS will sanction the user. The severity levels of the sanctions are in the following order (1 being lowest level of severity):

1. Warning of removal of permission
2. Removal of permission
3. Warning of removal of power
4. Removal of power

For example, a user has the physical power to touch the steering wheel, they have the institutional power to touch the steering wheel and request information about it. However, they do not have the permission to touch the steering wheel when a door is not open. In this case the user will just be helped with a message such as: this is not allowed, please open a door first. If this was the users first minor offence and is not doing any harm, then the FIS doesn't log this as a problem. If the user does this again, the FIS will see this as a warning should be provided as still no harm is done. But if the user decides to do this again the FIS will apply a heavier sanction and remove the permission of being able to perform Touch actions at all.

4.4 Decision Maker

The DM uses several APL policies and FISs to make decisions about how the system (exhibit) should be controlled to cater for the user in the best possible way as to keep the users QoE high and to keep the systems (exhibits) safe from misuse. The policies include MuseumSystem, exhibit, subject, exhibit extension and VirtualModel policies. The FISs include TouchAction, ShakeAction, RotationAction, Open/Close usage FISs. The decision maker of course gets raw data from a variety of sensors such as user presence, compass, accelerometer, touch, open/close sensor readings from the exhibit extension, time etc. to infer or process data to something meaningful which can then be used to make decisions.

The combination of FISs work together to infer behaviours, such as if the user is 'normal' or whether they are potentially 'dangerous' and misusing the exhibit. This inferred behaviour may actually not mirror the reality, i.e. this misuse may not be intentional and therefore the system should not penalize the user on just some of the interactions and therefore must not appear to be a rigid system. The FISs use multiple rules to work out from many actions if the user is in need of help or should be heavily sanctioned for dangerous/mischievous behaviour. Therefore the feedback/response from the system varies from help and mild warnings, to strict warnings and even sanctions (denial of further interactions and/or even complete removal from using the exhibit). The administrator is able to intervene at anytime with the decisions of the system if something is clearly going wrong.

5 THE iCARS EXHIBITION

To evaluate the enhancement of QoE, we have built a usability room that is a microcosm of an exhibition space in the Science Museum, London. This is the iCars Exhibition with the four exhibits, History, Security & Safety, Environment and Quiz, arranged as shown in Figure 4. 'Visitors' start in the south-west corner and view consecutively the History (non-interactive) exhibit, the Safety (partially interactive) exhibit, and the Environment (fully interactive) exhibit, before addressing the Quiz (fully interactive) exhibit.

The Environment Exhibit is an APL Exhibit Extension of an instance of the APL System Client. The exhibit itself is made up of two parts: real and virtual. The real part of the exhibit is an Aston Martin DBS scale model, which is fitted with a variety of sensors:

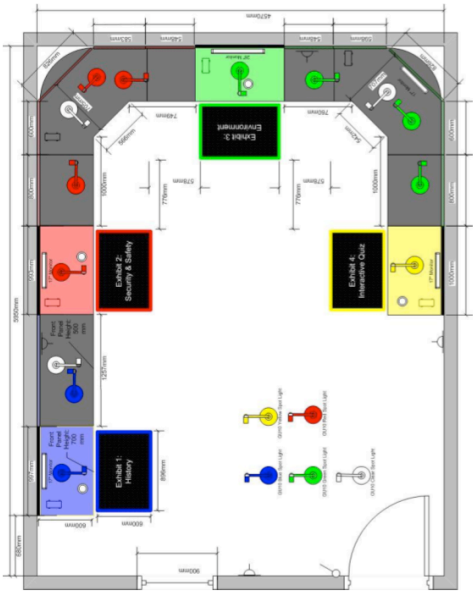


Figure 4. Floorplan of the iCars Exhibition

- Touch sensor: identifies which non-moving parts of the exhibit are being explored;
- Door/bonnet/boot sensors: identify which moving parts of the exhibit are being explored;
- Compass sensor: identifies the position of the car in the Z plane;
- 3-axis accelerometer sensor: identified movement of the car as a whole in the X, Y and Z planes.

These sensors record touch, rotations, open/close and shake actions. Detected actions can be fused with data in order to control the content delivery to the user using the information plaque, this being a CG model of the car (i.e. the virtual part). The car itself, and corresponding displays, are illustrated in Figure 5.

Figure 5. Exhibit 3: Model Car and Display

We are now in a position to commence trials and experiments to determine any enhancements of QoE enabled by the iCars Exhibition. These trials are scheduled to take place in March 2009.

6 SUMMARY AND CONCLUSIONS

In summary, we have observed that one of the most pressing problem facing modern museums is to engage their visitors in a way which is more interactive than traditional exhibits and information displays; yet is more engaging that just viewing the a computer-generated version of the exhibit on a web page, say; and yet remains informative, instructive, and memorable. We defined a notion of Quality of Experience (QoE) in this context, and developed a system designed to enhance QoE in a visit to a public collection.

This system operates in a sensor-saturated environment, with sensors in the exhibition space and in the exhibits themselves. In addition to this the client-server APL platform provides a generic infrastructure to define exhibits. Then, by using fuzzy inference and user-defined policies, we could use an 'intelligent' decision maker to customise displays and exhibit behaviours. It was this personalisation that we though would be the keystone to enhancing QoE. We have defined and built an experience, iCars Exhibition, a microcosm of a real museum exhibition space, which at the same time allows us to evaluate any enhancement.

It is too early to draw any conclusions, as the research programme is entering its evaluation phase. The system will be evaluated as it currently stands, but one direction of further work is to enhance the decision-making with a 'learning' component. The idea here is to match a current trajectory of behaviour against a history of previous interactions to create a sense of 'expectation', which can be used to provide some form of pro-active behaviour [4]. We would claim that this fusion of sensor data and user actions to adapt the behaviour of, for example, displays and exhibits is addressing a specific challenge of pervasive adaptation, i.e. how to coordinate the complex interaction of users, devices, sensors, and intelligent systems to deliver an improved service or experience.

ACKNOWLEDGEMENTS

We would like to thank the EU DANAE Project, the EU PANORAMA Coordination Action, and an Imperial College Studentship for various aspects of support for this research.

REFERENCES

[1] L. Alben, 'Quality of experience: defining the criteria for effective interaction design', *ACM Interactions*, **3**, 11–15, (1996).
[2] R. Beauregard, A. Younkin, P. Corriveau, R. Doherty, and E. Salskov, 'Assessing the quality of user experience', *Intel Technology Journal*, (2007).
[3] A. Bhusate, L. Kamara, and J. Pitt, 'Enhancing the quality of experience in cultural heritage settings', in *Proc. 1st European Workshop Intelligent Technologies for Cultural Heritage Exploitation, 17th European Conf. Artificial Intelligence*, pp. 1–13, (2006).
[4] C. Castelfranchi, R. Falcone, and M. Piunti, 'Agents with anticipatory behaviors: To be cautious in a risky environment', in *ECAI 2006, 17th European Conference on Artificial Intelligence*, eds., G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, pp. 693–694. IOS Press, (2006).
[5] B. Corrie, H. Wong, T. Zimmerman, S. Marsh, A. Patrick, J. Singer, B. Emond, and S. Noël, 'Towards quality of experience in advanced collaborative environments', in *Third Annual Workshop on Advanced Collaborative Environments*, (2003).
[6] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, 'The ponder policy specification language', in *POLICY*, pp. 18–38, (2001).
[7] J. Forlizzi and K. Battarbee, 'Understanding experience in interactive systems', in *DIS '04: Proceedings of the 5th conference on Designing Interactive Systems*, pp. 261–268. ACM, (2004).
[8] T. Hall and L. Bannon, 'Designing ubiquitous computing to enhance children's interaction in museums', in *Proceedings of the 2005 conference on Interaction design and children*, pp. 62–69. ACM, (2005).
[9] A. Isard, J. Oberlander, I. Androutsopoulos, and C. Matheson, 'Speaking the users' languages', *IEEE Intelligent Systems Magazine*, **18**(1), 40–45, (2003).
[10] L. Kagal, T. Finin, and A. Joshi, 'A policy language for pervasive systems', in *Fourth IEEE Int. Workshop on Policies for Distributed Systems and Networks*, (2003).
[11] A. Uszok, J. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken, 'Kaos policy management for semantic web services', *IEEE Intelligent Systems*, **19**, 32–41, (2004).

Adaptive Mesh Architectures for Speckled Robots

D. K. Arvind[1] **and A. P. Kulkarni**[2]

Abstract. The primary purpose of humanoid telepresence cyber-physical systems is to faithfully reproduce user intent in the remote robot. One of the main challenges lies in maintaining bipedal balance during complex behaviour such as walking, or negotiating obstacles whilst operating in a changing environment. The aim is to achieve this dynamic balance during human-controlled motion, and determine an effective solution to the humanoid telepresence problem by using a distributed sensing and actuation architecture. Preliminary results are presented for the stability of a Kondo KHR-1HV robot equipped with wireless motion capture Orient-2 sensors (each equipped with 3-axes gyroscope, magnetometer and accelerometer) for controlling six servos in the joints in the upper and lower legs and connected as a mesh, which demonstrate that distributed robotic control is better able to recover gracefully from command latency and local failures.

1 INTRODUCTION

Cyber-Physical Systems (CPS) [2] consist of physical resources which are coupled closely with networks of embedded sensor, actuator and computational devices for monitoring and controlling them. Examples of CPS include autonomous collision avoidance, robotic surgery, assistive technologies for pervasive healthcare, and tele-operations in hazardous environments. This paper considers the interplay between the low-level distributed sensing/actuation/computational devices and the higher-level control networks. The CPS of particular interest is humanoid telerobotics which seeks to achieve real-time use by a human operator of a anthropomorphic robot in a remote location. In a previous paper [1], the authors had described a specknet-based [3] cyber-physical framework for humanoid telerobotics.

Figure 1 shows a model of the humanoid telerobotics CPS consisting of a user interface for the operator which is realised as a distributed on-body sensor network for motion capture [4], a forward transmission link, a robotic "avatar" with mesh-connected sensor-actuator network, appropriate inverse kinematic mechanism translating user signals into avatar movements, and a reverse feedback link from the avatar to the operator.

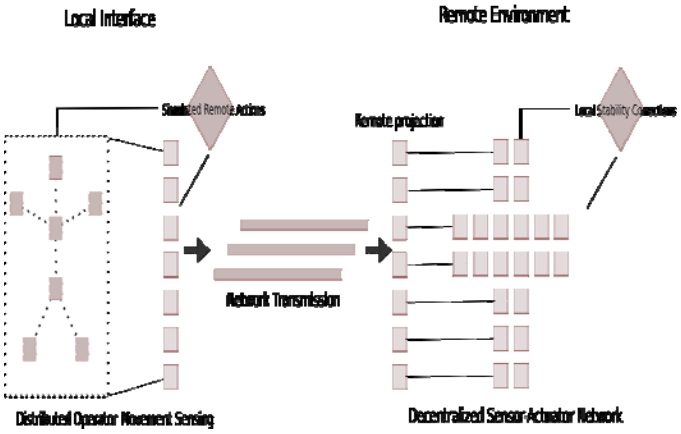


Figure 1. A distributed cyber-physical architecture for telepresence robotics. The human interface for the operator is realised as a distributed sensor network with the remote interface implemented as a mesh-connected sensor-actuator network [1].

One of the issues, when realising cyber-physical architectures, concerns the resolution of dynamic control; in other words, how does one maintain balance when making complex imitative motion, i.e. where in the control loop should low-level movement decisions be made?

The cyber-physical approach uses a distributed wireless sensor network on the operator, to sense movement at the source, and a sensor/actuator network on the robot to manipulate and control it. Three implementations were explored: distributed sensing with distributed direct actuation; distributed sensing with centralised actuation; and, distributed sensing with distributed cooperative actuation [1].

This breakdown corresponds naturally to three architectures for the management of low-level control schemes.

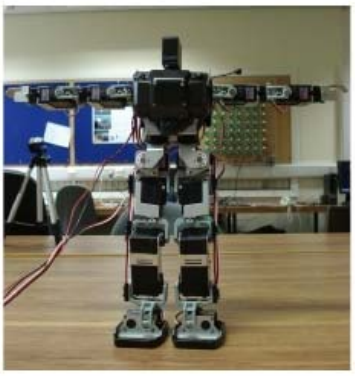
1. User-Side: Body-based wireless sensor networks on the operator detect local orientation of the limbs and

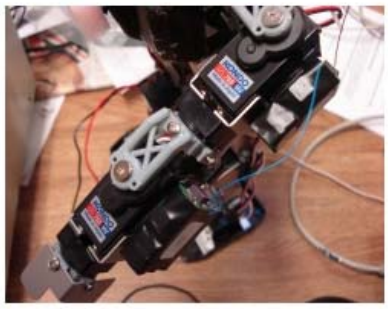
[1] Research Consortium in Speckled Computing, School of Informatics, Univ. of Edinburgh, EH8 9AB, UK. Email: dka@inf.ed.ac.uk.
[2] Dept. of Industrial Engineering and Operations Research, Univ. of California, Berkeley, CA 94720, USA. Email: anandk@berkeley.edu.

transmit these signals independently to the corresponding servos on the robot (point-to-point control).

2. Intermediate: Network-resident inverse kinematics programs on an intermediate layer between user and robot read movement information from the user and determine the low-level instructions sent to the servos. Predictive filters observe operator actions and transmit sanitised instruction data to servos for the purposes of protecting the robot from damage caused by motion outside its range or activating prerecorded movement patterns.
3. Robot-side: Individual wireless sensor-actuators devices on the robot's body accept input from the remote operator and come to a local consensus on how best to achieve user-transmitted goals, but make and execute these decisions locally.

Figure 2. From top-left, clockwise (1) KHR-1V robot, (2) Orient wireless sensor device, (3) local sensing, computation, and collaboration to active the servos to maintain balance, (4) Orient sensor connected to the servo on the robot.

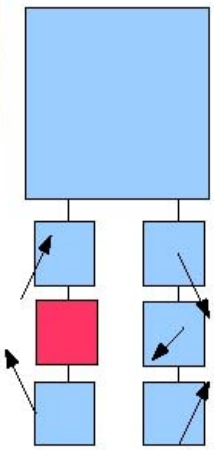




In [1], the three schemes were evaluated on a theoretical model of the telerobotic CPS. In the case of the first scheme, the orientation of each Orient-2 sensor mounted on the body of the operator is mapped directly onto the movements of each servo. Using the model for evaluation with order-of-magnitude estimates of each quantity, even before substituting parameters in the model measured experimentally, it was observed that latency effects of a few decaseconds caused by dropped packets and transmission delays lead to failure [1]. It was concluded that distributed sensing with direct one-to-one control of each servo by body-mounted sensors is an inappropriate strategy by itself. In the absence of any additional local feedback, this method caused stability failure.

2. MESH-CONNECTED RECOVERY SYSTEM

It is proposed to investigate experimentally the third scheme outlined above, one of mesh-connected, wirelessly-communicating sensor-servos on the rigid body of the robot which co-operate to maintain its balance. Each servo/Orient combination communicates orientation information to its neighbours and decides its movement at each stage in response to the changing environment and external stimuli. As any servo deviates from the target position transmitted by its corresponding node on the operator's body, the surrounding servos on the robot-side respond to maintain dynamic balance within the system while remaining faithful to the individually targeted positions transmitted by the operator's Orient devices. Servos react fractionally in opposition to the movement of surrounding servos, with the goal of maintaining an overall stable position of the structure. The resulting mesh network of joint servo-sensors is more resilient to failures of individual servos and to the delays in commands reaching individual servos.



3. THE ORIENT WIRELESS MOTION CAPTURE SYSTEM

The Orient Motion Capture System [4] developed by the Research Consortium in Speckled Computing at the University of Edinburgh demonstrated for the first time, fully wireless, full-body, 3-D motion capture in real time using on-body network of fifteen custom-designed Orient inertial sensor devices. The system is free of infrastructure such as cameras[5][6], magnets, or audio receivers, and does not require special suits to be worn to contain the wires as in the cases of Xsens's Movens [7] and Animazoo's IGS-190 [8].

The compact Orient device measures 36x28x11mm and weighing 23gms (including the custom-designed Perspex casing and Velcro straps) and contains three 3-axes sensors: gyroscope, accelerometer and magnetometer and a temperature sensor. The nine sensors are sampled at a frequency of up to 512 Hz, and a positional update rate of up to 64 Hz is attained over the wireless network of 15 devices for full-body motion capture using a modest low-power 250 kbs radio. This is achieved thanks to an efficient local orientation estimation algorithm in the Orient device firmware which runs on a 16-bit dsPIC processor, and which reduces the communications data by 79% compared to existing methods [4].

Its onboard ADC is used to sample the inputs of the analog sensors: rate gyroscopes, magnetometers and accelerometers in each axis, plus temperature monitoring to allow compensation of the thermal response of the sensors. When multiple Orient devices are used together, their measurements are synchronised and their results transmitted across the radio channel in sequence, so that a complete frame's data can be assembled at the base-station within milliseconds. The base-station has USB, Bluetooth and WiFi interfaces which can bridge to a mobile phone or PDA for viewing the visual representation of the results.

The Orient devices capture orientation data at the maximum update rate of 64 Hz for 15 devices for around 150 minutes from a full battery charge. The 120mAh lithium polymer battery and charger are integrated into the device, with charging as simple as plugging in a lead, even when the device is held in a strap for use. The whole device can be placed into a low-power, sleep mode for weeks at a time, whilst being ready for use within a couple of seconds of being woken by a radio signal.

A simple calibration process requires the operator to hold briefly, just for a few seconds, a pre-determined stance which enables the alignments between the Orient devices and the operator's body segments to be automatically accounted for.

The MotionViewer software provides a user-friendly interface for the operator to interact with a network of Orient devices. The software comprises of five main subsystems: Device Interface, Forward-kinematic rigid-body model, Project Management, Real-time visualisation, and a Plugin API.

The Device Interface is used to configure individual Orient devices and set up a network to perform motion capture. The base-station with its USB, Bluetooth and WiFi interfaces acts as a bridge between the Orient devices and the host which could be a PC, PDA or a mobile phone. The interface is designed to be usable as a library for stand-alone applications, in addition to its use in MotionViewer.

4 EXPERIMENTAL RESULTS

In the following experiments, the operator is strapped with Orient devices for motion capture.

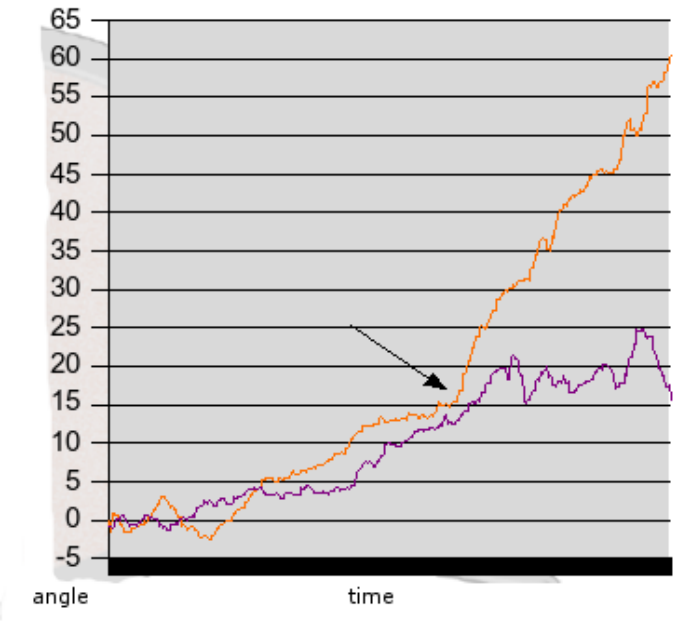


Figure 3. Graph of the angles of displacement of the torso (y-axis) of the robot over time (x-axis) for the two cases: with (purple, lower line) and without (orange, upper line) assistance of a collaborative mesh network. The arrow points to the moment when the robot topples.

The telerobot used in the experiments was a commercial off-the-shelf Kondo KHR-1HV [9] humanoid desktop-scale robot. The servos on the robot are almost identical and could be approximated to behave as a distributed robot application consisting of homogenous components. Each Orient sensor node directly controlled the orientation and motion of a single servo on the robot body. In order to simplify the hardware implementation, the effect of directly integrating the sensor node with the servo was simulated by routing instructions via a central computer connected to the robot. As the transmission speed of instructions from the robot to reach and activate the servos was negligible (less than a millisecond) in comparison to other latencies in the system, this simulation does not affect the authenticity of the results.

Early results are presented for testing experimentally the hypothesis outlined in the third scheme in Section 2 in which the Orient sensor devices on the robot-side collaborate locally to maintain its balance. The operator with the Orient sensors strapped to his body controls a remote robot with a decentralised network of Orient devices linked to the servos. Each sensor-actuator combination re-aligns itself to match positions with its counterpart on the user's body several times per second using simple update rules, which permits dynamic responses to the user's actions. In the absence of direct position instructions from the operator, the sensors communicate locally to maintain the balance of the robot. The resulting emergent behaviour based on the simple rule of making fractional movements in the servos in the direction opposite to their neighbours is in practice more resilient than external movement directives.

The effectiveness of the mesh-networked sensor-actuator combination was tested in maintaining balance of the robot due to unexpected environmental impact resulting in extended packet delays from the operator. An Orient device was also mounted on the torso of the robot to measure and record its angle of displacement from the vertical. Mechanical failure and packet losses were simulated by disconnecting the power supply to the Orient device in one of the sensor-actuator pairs on the robot's "knees". External pressure was steadily applied on the robot's torso until it toppled. This was carried out in two scenarios: with the sensor-actuator pairs collaborating as a mesh network; and with the actuators acting in the normal manner controlled by a single built-in gyroscope.

The angle of the robot's torso with the normal was plotted over time as a graph shown in Figure 3. The orange (upper) and purple (lower) lines refer to the cases of without, and with the mesh-connected collaborative network to maintain balance. Without mesh recovery the robot toppled quite rapidly at the point in time indicated by the arrow in the graph. In contrast, with the mesh recovery activated, the servos shared the neighbours' orientation data and moved fractionally in the opposite directions. The outcome as shown in the purple line is the neighbouring sensor-actuator pairs compensating successfully for the "failed knee" to maintain a temporary stable feedback loop with the angle with the normal being maintained steadily around 20 degrees. This preliminary result favours the idea of a mesh recovery telerobotic system for maintaining balance using local sensing and control while satisfying external movements sent by the remote operator.

5 CONCLUSIONS & FUTURE WORK

A couple of questions of theoretical interest immediately present themselves:

1. In the case of arbitrary rigid body structures, does there exist a set of rules for local responses for the servos that keep the system stable?
2. Is a meshed interconnectivity necessary for larger systems or is the awareness of the nearest neighbours sufficient?

The second scheme outline in Section 2 assumes the availability of information and data on the wider Internet in the humanoid CPS, as well as a large collection of possible users. An interesting inquiry would be to explore how this data can be used in the other layers to help smooth the telepresence experience. For example, access to online movement recordings and motion-capture databases may allow lower-level portions of the telerobot system to better coordinate its movements.

The ``physical'' layer need not be isolated to the robot's end. More sophisticated interfaces might utilise distributed actuation on the operator's end in the form of wireless haptic devices.

ACKNOWLEDGEMENT

The authors wish to thank the Speckled Computing group at the University of Edinburgh for useful discussions. APK was supported by the Research Consortium in Speckled Computing during his visit to the University of Edinburgh during the period, July – December 2007. The Research Consortium in Speckled Computing is funded by the Scottish Funding Council as part of the Strategic Research Development Programme (R32329), and UK Enginneering and Physical Sciences Research Council (EPSRC) as part of the Basic Technology Programme (C523881).

REFERENCES

[1] D. K. Arvind and A. P. Kulkarni. Specknet-based Cyber-Physical Frameworks, in Proc. Int. Workshop on Cyber-Physical Systems Challenges and Applications (CPS-CA'08), Santorini Island, Greece, June 11 2008, in conjunction with the 4th IEEE Int. Conf. on Distributed Computing in Sensor Systems (DCOSS'08).

[2] http://www.nsf.gov/pubs/2008/nsf08611/nsf08611.htm accessed on 1 March 2009.

[3] D. K Arvind, "Speckled Computing", Proc. 2005 Nanotechnology Conference, Vol 3, pp 351-4, ISBN 0-9767985-2-2, Anaheim CA, USA, May 2005, NSTI.

[4] A. Young, M. Ling, and D. Arvind. Orient-2: A Wireless Real-time Posture Tracking System Using Local Orientation Estimation. *Proc. 4th International Workshop on Embedded Networked Sensors*, Cork, June 2007.

[5] Vicon, http://www.vicon.com

[6] Motion Analysis Corporation
http://www.motionnalysis.com
[7] Animazoo, http://www.animazoo.com
[8] Moven, http://www.moven.com
[9] Kondo http://www.kondo-robot.com

Conceptualisation of an application of adaptive synthetic socioeconomic agents for intelligent network control

Doug Legge, Atta Badii
The Intelligent Media Systems & Services Research Centre
University of Reading
(www.imss.reading.ac.uk)
(d.j.s.legge, atta.badii)@reading.ac.uk

Abstract— The deployment of Quality of Service (QoS) techniques involves careful analysis of area including: those business requirements; corporate strategy; and technical implementation process, which can lead to conflict or contradiction between those goals of various user groups involved in that policy definition. In addition long-term change management provides a challenge as these implementations typically require a high-skill set and experience level, which expose organisations to effects such as "hyperthymestria" [1] and "The Seven Sins of Memory", defined by Schacter and discussed further within this paper.

It is proposed that, given the information embedded within the packets of IP traffic, an opportunity exists to augment the traffic management with a machine-learning agent-based mechanism.

This paper describes the process by which current policies are defined and that research required to support the development of an application which enables adaptive intelligent Quality of Service controls to augment or replace those policy-based mechanisms currently in use.

***Index Terms*—Quality of Service, multi-agent, ontology.**

I. INTRODUCTION

Within Service Orientated Architecture (SOA) network topologies that implement Quality of Service (QoS) mechanisms for communication, there exists an issue with the human-oriented requirements analysis and ongoing policy change management procedures to ensure acceptable user perception of the application(s) classified. This is in part due to that initial "application requirements analysis", which can be complicated and involve many differing user-groups, each with distinct and differing options, perceptions and goal orientated requirements (design by committee). The provision of an ongoing change management function to support business or network changes means many corporate, particularly Small-to-Medium-Enterprise (SME) businesses face real challenges regards resource, skill-set, business priorities and timeframe.

It has been well documented within the network (e.g. IETF, IEEE) and vendor community (e.g. Cisco Systems, Juniper Networks) network metrics and sensitivities required to define appropriate QoS policies, but it is the goal of the enterprise to interpret these various best-practises, in such a way that they can support the "business requirements" of the applications being deployed. Therefore the application requirements analysis must tie those technical-criteria and business-goals into a defined policy.

This typically necessitates a business analyst, questioning groups of collaborative parties, possibly including: an application development team; user group; management group; network team; technical support group and technical management group, to establish the properties of the application or applications under development, characteristically influenced by:

- Protocol sensitivities (e.g. voice compared with ftp)
- Application sensitivity (e.g. real-time ticketing system, see table A.1.1. at appendix A)
- Concurrent user access
- Infrastructure (e.g. the abilities of the supporting LAN, WAN & client > server environment)
- Topology (e.g. the geographic, political & corporate distribution of the intended user group)
- Commercial business priority (e.g. willingness to support priority of application *x* over application *y*) and any associated surcharge or penalty

From this requirements analysis a policy can be proposed, to implement mechanisms by which to classify, mark and "rate limit" traffic. Classification is the process of identifying traffic and categorising that traffic into different classes such that prioritisation can be allocated. Typical traffic descriptors include: Class of Service (CoS); incoming interface; IP precedence; Differentiated Services Code Point (DSCP); source or destination address; application; and Multiprotocol Label Switching (MPLS) Experimental Bits (EXP). After the traffic has been defined and classified, it is accessible to QoS mechanisms for treatment. Classification should take place at the network edge as this reduces the transfer of packets which may only be dropped later due to their low classification marking value. Layer 2, Class of Service (CoS) mappings can be implemented, however a disadvantage of CoS markings is that frames lose their CoS markings when transiting a non-802.1Q or non-802.1P link, including any non-Ethernet WAN link. Therefore a more ubiquitous permanent marking such as IP DSCP, should be used for network transit. In a Cisco network device environment this would take the form:

Classification
class-map name
match traffic type

```
description :an example of which could be match protocol http
url "*important*"
 !
```

Marking/Policing

```
policy-map name
        class name
                set mark
description :an example of which could be ip dscp af21
 !
```

Rate Limit

```
service-policy name (applied at an interface level)
        set constraint
description :an example of which could be setting the
bandwidth available, or the queuing mechanism to be applied,
such as: fair-queue or LLQ
 !
```

Assured Forwarding, defined in RFC 2597 [2], delivers a Per Hop behaviour (PHB) for applications that require a better reliability than the best-effort service, which identifies four classes of service, and within each class, three drop precedence's, detailed in Table A.1.3 at appendix A.

The analysis may require a review of all existing business applications, such as: finance; sales; billing; customer service; email; call-logging systems, which were previously mapped into class-groups based on the business and network audit of their requirements, sensitivities and operational business criticality. This is due to best practise guidelines that recommend that no more than 3 applications are mapped to the mission-critical, and transactional service classes. The greater the bandwidth saturation of the prioritised classes the greater the impact on those applications allocated with a lower priority classification. The resulting application requirements proforma (see table A.1.2 at appendix A) must then obtain executive endorsement, including an indication of any supporting changes or impacts (e.g. application *x* may have to be demoted to allow sufficient resource for this new delivery). A full policy to support a business scenario above is outside the scope of this paper, however a typical policy structure, based on Cisco Internetwork Operating System (IOS) syntax, defined for applications in use within the IMSS department at the University of Reading might show:

```
ip cef
!
class-map voip-rtp
        match protocol rtp audio
class-map http-blackboard
        match protocol http url "*blackboard*"
class-map http-reading.ac.uk
        match protocol http url "*reading.ac.uk*"
class-map match-any NetMeet
        match protocol rtp payload-type 4
        match protocol rtp payload-type 34
 !
policy-map imss-list
 !
   class voip-rtp
        set ip dscp EF
   class NetMeet
        set ip dscp AF41
   class http-blackboard
        set ip dscp AF 21
   class http-reading.ac.uk
        set ip dscp AF23
   class class-default
        set ip DSCP default
 !
interface Fastethernet 0/0
!
   service-policy input imss-list end
!
```

What this policy, called imss-list, specifies is:

- Match the protocol rtp audio and mark with DSCP Expedited Forwarding (EF)
- Match the protocol NetMeet (rtp payload-type 4 and payload-type 34) and mark with Assured Forwarding (AF) 41
- Match any http traffic that contains the URL string "blackboard" anywhere in the URL and mark with Assured Forwarding (AF) 21
- Match any http traffic that contains the URL string "reading.ac.uk" anywhere in the URL and mark with Assured Forwarding (AF) 23
- Match all other traffic and mark with DSCP default which is DSCP BE (Best Effort)

These application sensitivities, class-groups and associated addressing, once captured and signed off by the relevant management, are mapped into policies and applied to the interfaces of the networking equipments deployed in an infrastructure, further detailed in section 3. However as previously highlighted, following deployment adjustments can be made which effectively break those deployed policies; an example of this would be the change of the application server IP address used within the prioritisation policy. In addition, over a period of time, applications usage can change: more users attempt to concurrently connect; the application is enhanced to support additional features not covered (e.g. by port address) within that initial policy; or in fact the application sensitivities might have been incorrectly identified within the initial requirements analysis and all of these areas must be captured and considered to ensure the intended SLA can be effectively delivered.

This research aims to develop an agent-based society, where agents gain an understanding of that network on which the applications are delivered their individual application sensitivities and how well the network supports these sensitivities, and where these agents negotiate with other, or with application-specific or peering-agents, to achieve a suitable *classification* setting, such that the application can

match both an acceptable user perception, termed by the author's as Quality of Experience (QoE) or any higher level Service Level Agreement's (SLA) defined within the business or with third-party service providers (e.g. network provider). This is the correct and "best-effort" prioritisation, where best-effort is the ability to deliver the traffic for which an agent is negotiating within the constraints of its classification sensitivities and SLA, *but no better*, as this is resource that other agents must bid for as part of their outcomes.

This research will investigate adaptive learning techniques, including, but not exclusive to: neural; Genetic; and Bayesian networks; and consider adaptations of Swarm Intelligence models. Whilst the prospect of autonomous packet-based agents, for example: Reynold flocking boid's or Dorigo ant-hill colony algorithm, with outcome-based tasking, flocking throughout the network gaining that knowledge required to deliver a positive outcome, is fascinating, the current accepted view that Swarm Intelligence relies on dumb agents that select the most attractive free task, and that they do not typically negotiate for tasks, or in the coordination of these tasks as demonstrated by [3], which would appear to be a constraint requiring further investigation.

It is therefore proposed that this research focus on the delivery of an Application Specific Interface Card (ASIC) hardware based machine learning mechanism, which provides QoS based, not on human definition but actual application specific requirements against real-time network resource availability, and that this is then extended to provide inter-agent negotiation through peering arrangements of available resources. A major focus of the research will be that of "reuse", where reuse is defined as an activity that focuses on the recognition of commonalities of systems within and across research domains, e.g. should there be a requirement to provide the agents with an understanding of delay's occurring within the network, then an existing mechanism, such as an RSVP PATH message, is investigated for suitability or adaptation prior to any proprietary implementation.

II. Problem domain

End-users, of an application or system, require a positive Quality of Experience (QoE) in their interaction with technologies. Whether through: an operationally efficient system; ease of use, intuitiveness; interface design, functionality or aesthetics; or a new technology, such as; the quality of a High Definition (HD) TV service. In the field of data communications our focus as service and communications' providers is typically on the delivery of such services, within a set of Service Level Agreements (SLAs). These SLAs require consideration in a number of areas: from the applications or services being delivered; the infrastructure over which these services will be delivered; through interaction with other operators or applications, which may contend the delivery network; the skill and experience of the network support staff; right up to the selection of Customer Premises Equipments (CPE) including Personal Computers (PCs), any required user training, and ongoing support. All of these items can not only affect the delivery of this positive experience, but likewise have an impact on the operational expenditure (OPEX) of the provider, should it be deemed viable to deploy more or bigger supporting infrastructures to achieve this required experience. One of the biggest issues faced, not just by Communications Providers (CPs) but by enterprise IT departments, is the management of user contention, between the applications and services being delivered and that bandwidth available, particularly given the trend towards converged (data & voice) networks. The continued deployment of QoS mechanisms, introduces a number of reflections:

1. Prioritization mechanisms essentially provide "managed unfairness". It could be said that they, "rob Peter to pay Paul", that is they allocate applications a percentage of the available bandwidth, or apply queuing mechanisms, which during peak-demand periods, the service of such designated applications can negatively impact those other applications, allocated to lower priority queues or which have restricted bandwidth access imposed.

2. Whilst current QoS mechanisms capture the fact that applications do not typically require all 100% of the bandwidth allocated, 100% of the time, which allows available resource to be accessed by applications of a lower priority. These current QoS mechanisms are prone to ongoing change management issues due to their dependence on identifying the application(s) being classified by metrics such as: source/destination IP address; and IP port number. That is, should these identifiers change, as they typically do in enterprise environments and the change management procedures in place do not capture and incorporate these requirements within the classification policy, then the policy will break down.

3. The deployment of QoS mechanisms, whilst being done to manage the requirement for further bandwidth, can still result in escalating OPEX for organisations through the requirement to supply relevant skill-sets and mechanisms to visualise, monitor, trend, analyse, troubleshoot, and resolve issues. The delivery of a true end-to-end (E2E) management process, for most SME organizations is still:
    a. complex to design & configure
        i. ensure all system areas are incorporated (from ISO layer 1-7; client; server; supporting Infrastructure)
        ii. ensure all user groups perception of the solution are the same
        iii. multi-vendor environments requiring distinct language & even revision specific skill-set (e.g. IOS, JUNOS, v11.x, v12.x)
    b. complex to manage
        i. ongoing (change) management of deployed

policies and associated documentation

c. complex to visualize
   i. those policies deployed
   ii. define technical & management reports of sufficient depth
d. can be extremely expensive to purchase and maintain
   i. requires dedicated, trained personnel to ensure full ROI

The author is aware from his own experience, both as a technologist and as a team manager, that whilst it can be easy to define a human as an expert, through certification for instance, it becomes increasingly difficult for humans to retain that initial expertise through memory alone. We rely on memory to perform an astonishing variety of tasks in our everyday lives and this loss can be attributed to a number of conditions in a work environment:

a. Advancement in career typically results in a promotion away from that initial area of expertise, resulting in advancement in those new areas, but a loss of proficiency in those original skills. This is typically true of engineers who are promoted into management positions who must then lead other engineers in the delivery. The more time an engineer spends in this role, the more they find their hands-on skills diminish. A lot of knowledge engineers gather over the course of their tenure is held privately, this can result in issues regards a full and complete analysis of implemented policies should that engineer be unavailable.
b. All engineers' will reach a time in their life when earlier instinctive knowledge is lost. Marshall [1] highlights that irrelevant or out-of-date information needs to be pruned from the memory, because if humans remembered every detail, a condition known as "hyperthymestria", then this would result in mass confusion. Schacter in his book "The Seven Sins of Memory", which was the basis for Marshall's article in the New Scientist, details research into the ways humans forget and why, which can be summarised as (the sins):
   i. Omission (forgetting) - which includes; *Transience*, a gradual loss of memory in both short and long term memory; *Absent-mindedness*, a breakdown between attention and memory, either at the encoding or retrieval stage; and *Blocking*, a failure to retrieve or access deeply encoded information [4]
   ii. Distortion in contrast is responsible for; *Misattribution*, forgetting the source or assigning the memory to the wrong source of the information; *Suggestibility*, the implantation of memories; *Bias*, misremembering or influence of memories as a result of current knowledge, emotions or beliefs.
   iii. Intrusion responsible for *Persistence*, memories that should be forgotten but cannot be forgotten characteristically those resulting from strong emotional experiences.

Therefore, if machine learning was applied, not just to those tasks appropriate to the configuration and control of traffic (e.g. Cisco CCNA/CCNP/CCIE, Juniper JNCIA/S/E or similar), but to the business and technical applications analysis and classification, then whilst those skills remained relevant, memory loss would not be a factor in the system forgetting its goal orientated tasks.

Therefore this research looks to provide consistent and ongoing management of complex networks; which typically require Quality of Service (QoS) policy application, including, but not exclusive to:

- requirements analysis
- application classification and prioritisation
- policy definition & consistent management
- resource allocation
- network access control mechanisms
- topology, delay, learning and ontology engines to support automated inference
- negotiation, reputation & repudiation
- human visualisation reporting

III. PROPOSAL

In a traditional network environment, the policy rules, once defined from the business & technical analysis stage, would be applied to a hardware device such as a router depicted in Figure 3.1, these policies are now exposed to those issues presented within this paper, and for which the authors' now define their proposed mediation.

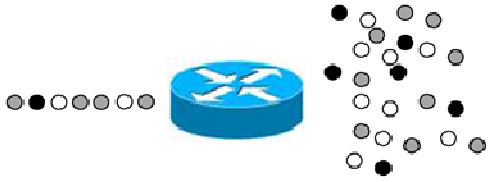


Figure 3.1. Typical Hardware Implementation

It should be pointed out that the *agents-unit*, used in this paper are adaptations of an "A Reflexive Autopoietic Unity" (Figure 3.2), presented by [5] in their paper "Organizations as Self-organizing and Sustaining Systems: A Complex and Autopoietic Systems Perspective". The "Autopoietic Unity" was originally presented by [6] in their book a "The Tree of Knowledge", and here they make the distinction, highlighted by Goldspink to the author [7], that an Autopoietic Unity demonstrated in Figure 3.2, is self-reproducing. However the author's do not contend that the agent-unit, based in-silico, has or even requires self-reproduction of its components, purely that the representation as shown in The Tree of Knowledge allows for a simple yet aesthetic representation, but this

adaption by the authors' to demonstrate their "learning-process" has been done in such a way as to ensure no confusion with the original representation or any biological inference.

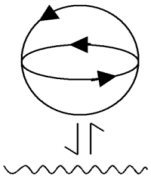


Figure 3.2. Autopoietic Unity [4] from [5].

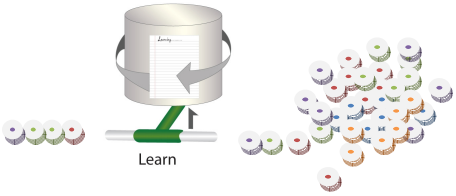


Figure 3.3. Agent Performs Supervised Learning

To overcome those issues identified, the adaptive synthetic socioeconomic agent would initially acquire data from the network, represented by the network cable and tap. The arrow represents that the agent now completes a comparison of that manual policy statement reasoning against its own knowledge base, indicated by the symbol: ⛁ . At this stage the explicit IF *x* THEN *y* actions based on that manually designated service-policy, represented by: 🗎, are completed and the agents uses internal processes (**Learn**): Egocentric representation - what is my local environment (e.g am I a router or switch and what interfaces do I have?) and; Topographic representation - what other devices are in the environment (routing map) to establish that manual outcome against its proposed outcome, thus establishing an experience of the positive/negative outcomes of this matching. A *Prediction System* evaluates the proposed outcome, this outcome then being compared to that actual episodic outcome, such that an error can be established, from here the system adapts to develop its outcome. A *Meta-prediction system*, looks at how well the system is learning, not the prediction of the outcomes of proposed actions, but its progress along the prediction error curve. To ensure that any service impact during this learning process is limited it should be possible to mirror (port-mirror) that inbound traffic such that the router processes *live* traffic, whilst the agent processes a *copy* of that same data.

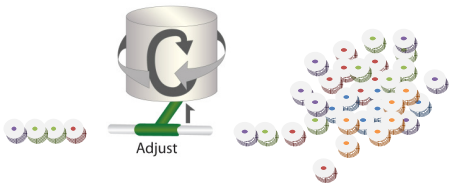


Figure 3.4. Agent Performs Unsupervised Learning

In Figure 3.4 the agent is now (**Adjust**)ing internal metric tables, represented by the internal arrow, such that it can remark, for example: the Differentiated Services Code Point, of those packets to be transmitted in a way that is believed will be beneficial to that traffic for which it is responsible. It performs implicit (**Reason**)ing, which could be weighted such, that whilst in a training mode, the learning and outcome was of more importance that the delivery of an outcome within any perceived budget (e.g. round-trip-time). In addition the agent should be updating topographic and ontological representations and potentially completing human visualisation (e.g. setting snmp traps and outputting for management reporting). However at this stage the agent is working in isolation to other agents and still completes a positive/negative outcome matching against that manual policy definition.

Figure 3.5. Agent Performs Reinforcement Learning

Figure's 3.5 & 3.6 show negotiation taking place between agents (**Negotiate**), such that resource allocation can be transacted, agreements made and following the transaction a reputation (positive or negative) achieve, allowing (**Grow**)th.

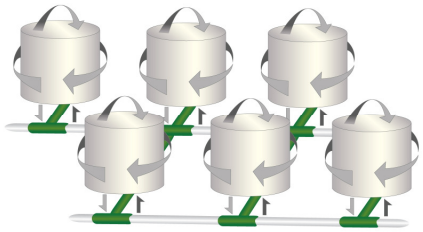


Figure 3.6. Inter-agent Negotiation, Reputation and Repudiation

The agents now display an Inference, the ability to connect existing knowledge, within the knowledge store, to create new knowledge through the chaining of existing rules or patterns [1], allowing a classification or ability to use data, to create knowledge, and not just be a digital filing cabinet. Finally the system reaches a state of *Formal Operations* [8] displaying an emergent behavior [9]. Emergence is the property in which the interaction of the parts creates some greater property of the whole which cannot be fully explained from the measured cause-and-effect behaviours of the components. The system is able to think logically about potential events or abstract ideas, can handle errors or mismatches and can build further on its semantic dictionary, constructing a library of "what-if" patterns for yet-to-happen (anticipated) events. This formal operating state, defined by Piaget [10], see Figure A.3.1.at appendix A, and which according to research by Bradmetz, few humans achieve, draws parallels with the work of

Artificial Life, which analyses and creates lifelike organisms in digital or robotic form. This Constructionist approach to education, where learning takes place inside the learner rather than being imposed by the teacher [11] aligns with the authors' goal of the system starting to self-organise.

IV. FUTURE WORK

Having established the framework on which the research will be based the authors present the following as further research milestones:

1. Experimentation and Simulation – This next paper will detail the work undertaken to present that observed traffic data to an artificial intelligence, machine-learning, application. In particular the authors are keen to demonstrate that any hypothesis of "optimization" should be based on an analysis and comparison of that performance which existed a priori and any associated proof be based on well formed and accepted methodologies. To this end the selection of the "learning-function" will be supported by appropriate statistical modeling to ensure that any experiments::

- conform to Shewhart's rule [12] to preserve all the evidence in the original data. This is important should the dataset, which [13] refers to as the *frame*, be used across experimental iterations, or multiple testing software' or new elements within that data are found in later experiments which can then be tested to ensure any comparison made to previous results is supportable
- an indication of what randomisation was selected
- an indication of what the results refer to and the robustness of the conclusions
- how the authors propose to use these conclusions to support further experimentation

This paper will then progress to describe those mechanisms required to support that proposed experimental & simulation environment, shown in Figure A.4.1 at appendix A, which includes:

- Generation – the creation of that data which will be input to the "learning mechanism"..
- Capture – how that network traffic data generated can be captured such that it can be analysed to ensure that it is exactly as sent (generated).
- Benchmark – benchmarked regards that performance provided by the system at that time, regards: load; transaction rate; pre-QoS; post-QoS; QoS policy applied; learning-mechanism; etc.
- Learning-mechanism – the learning types: neural; Bayesian; Kohonen; Hopfield; Boltzmann etc that allow for an optimum outcome
- Meta-prediction – how well is system the learning? As the system analyses patterns, triggers' or recall of other patterns are invoked. The time required for each iteration, and the number of iterations per pattern before the system to reaches an optimum (local or global) outcome, such as a point in the error curve, which can be measured and analysed. An additional reflection being what didn't work, as well as what did work. Later research papers will then progress those areas such as:

- Agent language – the language required for the agents to communicate
- Agent protocols – the elements which the agents adopt to ensure the global outcome (e.g. normalization) can be achieved over any local outcome requirements. This should include: non-repudiation; reputation and consider that social aspect of the agent society., that is those elements critical to ensure proper interaction to enable that "global" outcome (e.g. normalisation of the network environment). Whether the agents have, require or will achieve characteristics, and what those characteristics are. Could there be the equivalent of spectrum disorder (e.g. an autistic agent, which demonstrates those well known human autistic traits as [adapted from 14]:
    a. insistence on sameness; resistance to change
    b. difficulty in expressing needs
    c. difficulty in communicating with others
    d. unresponsive to normal teaching methods
    e. sustained odd behaviour
    f. obsessive attachment to objects (resource)
    g. apparent over-sensitivity or under-sensitivity
    h. noticeable over-activity or extreme under-activity
    i. non-responsive to cues

- Experimentation control – those controls and verification metrics suitable to support the results of the simulation as being correct and consistent
- Representation lexicon – those characters adopted for the representation of the work

Figure A.4.2. at appendix A, demonstrates a proposed distribution and agent type which is summarized in Table 4.1 below, and further detailed at A.4.1., Appendix A:

Agent Type	Description
Type 1 agent	Agent, typically software based on a PC/laptop device, resides in an internal Autonomous Agent Domain (iAAD). Communicates with Type 3 or 4 agent its location potentially via mechanisms such as its DHCP registration. Has local mechanism that records traffic sent & received and the delays seen in these transmissions.
Type 2 agent	Agent, typically software based on a telephony device, resides in an internal Autonomous Agent Domain (iAAD). Communicates with Type 3 or 4 agent its location potentially via mechanisms such as its DHCP registration. Has local mechanism that records traffic sent & received and the delays seen in these transmissions.

Type 3 agent	Agent, typically software based on a layer 2 (switch or access point) device, resides in an internal Autonomous Agent Domain (iAAD). Has local mechanism that records traffic sent & received and the delays seen in these transmissions. Type3 agents communicate full or local summary topology maps to Type 4 agents. Type 3 agents would typically have policing controls which allow the NBAR recognition of Scavenger services and take appropriate action (e.g. drop), of value where no Type 2 agents exist. During initial research the default mechanism is to pass all data traffic marked as COS/TOS 0 and all voice traffic as COS/TOS 5, this is to allow focus on the learning and optimisation with Type 4 devices, however further research will be focused on providing agent negotiation for Type 3 agents. It is feasible that a switch could perform all the duties of a layer 4 router. It is yet to be defined whether such a switch performing this function would be counted as a Type 4 agent device.
Type 4 agent	Agent, ASIC hardware based on a layer 3 (router) device, resides in both an internal Autonomous Agent Domain (iAAD) and if connected to a Communications Provider Type 5 agent an external Autonomous Agent Domain (eAAD). Type 4 agents (single or High Availability (HA) pair) are at the core of this current research. The main components of a Type 4 agent are proposed as: Topology engine: for internal mapping local interfaces and of the agent-based network, by collating all Type 1, 2, 3 and other Type 4 agent locations, potentially via existing mechanisms, such as: those devices DHCP registration; spanning-tree; or routing updates. This engine also collates all Type 1, 2, 3, 4 traffic sent/received/delays transmission reports such that internal E2E can be recorded. In addition it should be defined whether Type 4 agents collate summary information from locally attached Type 5 device(s) which provide an E2E baseline on which negotiations can be based, or Type 4 devices create a tunnel through the Type 5 devices for this purpose. Type 4 agents are the main focus of the initial research and it is currently still to be proved via simulation whether the agent performing their own proprietary communications and topology map building is optimal, or whether this topology is built by participating/listening to local routing protocol update information (e.g. OSPF, IGRP, iBGP, etc updates).
Type 5 agent	Agent, ASIC hardware based on a layer 3 (router) device, resides in both an internal Autonomous Agent Domain (iAAD) for Intra-service communications and if connected to a CPE Type 4 agent, or other Communications Providers Peering Partners an external Autonomous Agent Domain (eAAD). It is proposed that Type 5 agents are Communication/Service Provider agent devices which are able to either: a) consolidate, summarise and distribute topology maps, not only of their own internal topology and delay, but can consolidate and summarise that detail of their attached customer base b) or allow the construction of Type 4 agent tunnels for this purpose.
Type 6 agent	Agent, ASIC hardware or software based on a server device, or per application software agent install, resides in an internal Autonomous Agent Domain (iAAD). It is proposed that Type 6 agents are responsible for: a) the administration of E2E agent negotiation on behalf of the applications services installed b) they are to record and summarise data relevant to delay resulting from excessive internal processing

Table 4.1. Agent Architecture Summary

V. CONCLUSION

The authors conclude that this paper presents a cohesive programme of research intended to support, initially enterprise organisations in the requirements specification, deployment and ongoing service management of quality of service policies, but that eventually this methodology could prove scalable enough to support service provider of Internet environments.
Architecture Summary

VI. APPENDIX A

The supporting figures, diagrams and tables for this paper, can be found at: <http://www.imss.rdg.ac.uk/Publications/DLeggeABadii2009AppendixA.pdf>,
whilst the original un-summarised paper can be found at: http://www.imss.rdg.ac.uk/Publications/DLeggeABadii2009Full.pdf.

VII. REFERENCE

[1] Marshall, J., (2008), Unforgettable, New Scientist, 16th January 2008, page 30

[2] Heinanen, J., Baker, F. Weiss, W., Wroclawski, J., (1999) Request for Comments: 2597, Assured Forwarding PHB, IETF Network Working Group, June 1999, http://www.ietf.org/rfc/rfc2597.txt, (Accessed 17th February 2009).

[3] del Acebo, E., and de la Rosa, P., (2008), Bar Systems: A Class of Swarm Intelligence Optimization Algorithms, AISB Conference, Aberdeen (April 2008).

[4] Wallisch, P., The Seven Sins of Memory, (2007), [Internet], http://lascap.de/The_Seven_Sins_Of_Memory.pdf, (Accessed 20th February 2008, not currently online).

[5] Goldspink, C., (2008), Conversation relating from his presentation: Agent Cognitive Capabilities and Orders of emergence: Critical Thresholds relevant to the Simulation of Social Behaviours, AISB '08, University of Aberdeen (April 2008).

[6] Humberto R. Maturana and Francisco J. Varela (1987), The Tree of Knowledge, reprinted by arrangement with Shambhala Publications, Inc., Boston, www.shambhala.com.

[7] Goldspink, C., & Kay R., Organizations as Self-organizing and Sustaining Systems: A Complex and Autopoietic Systems Perspective, International Graduate School of Management, University of South Australia, joint impact, email from author [5] & meeting held in Guilford, October 2008.

[8] Eysenck, M., (2001), A2 Psychology: Key Topics, London: Psychology Press

[9] Highsmith, J., A., III, (1999), Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, Dorset House Publishing, New York.

[10] Wagner, K., Van, Background and Key Concepts of Piaget's Theory, Stages of Cognitive Development, http://psychology.about.com/od/piagetstheory/a/keyconcepts.htm, (Accessed 17th February 2009).

[11] Challoner, J., (2002), Artificial Intelligence, London: Doring Kindersly

[12] Shewhart, W. A., Economic Control of Quality of Manufactured Product, http://en.wikipedia.org/wiki/Shewhart, (Accessed 8th September 2008).

[13] Deming, E., W. (1975) On Probability As a Basis For Action. *The American Statistician,* Volume 29, No.4., http://www.stat.osu.edu/~jas/stat600601/articles/article1.pdf (Accessed 19th February 2009).

[14] Characteristics of Autism, (2008), [Internet], Autism Society of America, http://www.autism-society.org/site/PageServer?JServSessionIdr012=sqyplynxu1.app24a&pagename=about_whatis_char, (Accessed 8th September 2008).

A Dynamic Connection Capability for Pervasive Adaptive Environments Using JCSP

Anna Kosek[1], Aly Syed [2], Jon Kerridge [1] and Alistair Armitage[1]

Abstract. The house, office or warehouse environment is full of devices that make users' life and work easier. People nowadays use personal computers, laptops, Personal Digital Assistants, mobile phones and many more devices with ease. The mechanism to connect, enable co-operation and exchange data between devices will help to use devices' full capabilities. This paper investigates the usability of Communicating Sequential Processes for Java in pervasive systems and the adaptation possibilities offered by this environment. It focuses on dynamic connection capabilities. The paper also describes an experiment that organizes an adapting pervasive environment which uses dynamic connections for data flow.

1 INTRODUCTION

The number of devices surrounding us increases every day. The devices that we use have become more complicated and have more capabilities and functions. A mobile phone is now not only used to make calls, but also to store and edit files, take photos and videos, play music, browse the web and much more. Some of those functions are repeated in many devices. Most of the devices that we use are autonomous and making them co-operate with other devices is often difficult and time consuming.

The mechanism to connect devices, make them co-operate and use each other's functions in a simple and transparent way would help to manage a network of devices used every day. The method of making many computers physically available, but effectively invisible to the user is called ubiquitous computing [1] or pervasive computing. The concept was introduced by Weiser in 1991 and many researchers have been investigating it since.

Making connections is a crucial capability when considering a network of devices. Dynamic connection requires device and service discovery. A device arriving in an unknown network has to have mechanisms to adapt to the environment and discover distributed capabilities that can be useful. The same device in a different location can work differently depending on services that are available in a particular space.

This paper describes dynamic connection capabilities for pervasive adaptation using Communicating Sequential Processes for Java (JCSP) [2]. In Section 2 the background to the research is provided. In section 3 pervasive software requirements are presented and compared with JCSP properties. Section 4 describes JCSP channels in more detail. An experiment using dynamic connections for synchronization is presented and evaluated in Section 5 of this paper. Conclusions and future work are stated in Section 6 of the paper.

2 BACKGROUND

The background section of this paper presents the research areas of pervasive computing, ad-hoc networks and adaptation. Fundamental concepts of CSP and a simple example of a CSP based system are also described.

Pervasive (Ubiquitous) Systems

The ubiquitous computing concept is an approach to making many computers physically available, but effectively invisible to the user [1]. Devices in the system vary in size, capability and function, each suited to a different task. The term 'ubiquitous' suggests that devices will finally become so pervasive in everyday objects, that they are hardly noticed [3]. Ubiquitous computing is also called pervasive computing [4].

Devices in pervasive computing systems are enabled with communication capability and are designed to be useful in a single environment such as home or hospital. The idea is to integrate all the devices that are connected to the network to co-operate and use each other to achieve an expected performance and capability. In this system computers are no longer tools but assist humans in their everyday activities. All the computers in the environment will not be considered as autonomous but parts of a larger system that is targeted to users' needs. Devices in pervasive system have to be aware of their location and surroundings. If a computer knows its position, it can adapt its behavior to the environment [1].

Network Structure

The intelligent environment should work with any set of devices, and enable collaboration of available devices depending on their capabilities. In the event of a device failure, the system should reconfigure itself and continue to work. A central control workstation or remote control device is not the goal of a pervasive system. It is likely that a mixed control system will be required that is neither fully distributed nor fully centralized but the particular mix of control function needs to be determined and may vary with the application environment.

The pervasive adaptive environment is often a mix of fixed and mobile devices. Some devices can stay in the environment for a long period of time; some can visit a specific space only briefly. As a central control system is not present, devices in the network have to monitor the network topology. These kinds of networks are called ad-hoc networks and defined as a collection of mobile and fixed devices communicating over wireless links [5]. A mobile ad-hoc network should be a set of devices that recognize each other, decide about inter-network connections, organize a virtual topology, exchange and use resources and capabilities.

Devices Types

A pervasive adaptive environment can consist of different types of devices. Equipment varies from very small devices like sensors, capable of sending only a precise type of signal or FPGA's programmable for specific tasks, to PDA's and powerful stationary and mobile computers. Rapid technology advances affect device configuration, therefore, it is difficult to pinpoint what constitutes a big or a small device, but for the sake of argument we define a small device to have 100 MHz processor and less than 64kB RAM. A medium device is defined to have 200-400 MHz processor and less than 100MB RAM. A large device is considered to have more than 400 MHz processor power and more than 100MB RAM. Some parts of the pervasive adaptive system can be very simple and be based on primitive signals, other parts must be powerful enough to run algorithms associated with learning and analyzing data. Therefore communication capability must be based on uncomplicated data structures and acknowledgments, so a wide variety of devices can communicate.

Adaptation

The ability to adapt is an important requirement for a pervasive environment. New services, functionalities, interaction mechanisms or devices can be added to the pervasive system requiring them to be adapted to the specific characteristics of the environment [6]. Mobility of devices makes a system's network topology even more dynamic. Mobile devices have to be able to detect change of location and exploit knowledge about their current situation; this is called location-awareness [7]. All mobile devices have to detect and react to the environment and therefore adapt to a new space to improve quality of communication. On the other hand existing elements from the system might be adapted to a user's requirements or changing conditions in the environment [6]. This characteristic of pervasive systems is called context-awareness.

Adaptation should be supported by a flexible and dynamic system. Dynamic communication is a very important capability for pervasive adaptation. When a mobile device arrives in a new location communication has to be established. Depending on the situation and devices present in this environment connections have to be created or destroyed. The ability to manage connections between many devices in an intuitive and transparent way is very useful for pervasive adaptation.

Concurrent Systems

A system is called concurrent when there is more than one process existing at a time [8]. A concurrent style of programming enables the creation of a system with processes working in parallel. One of the key aspects of parallel system design is that simple processes can be composed into larger networks. The concurrent style of programming better reflects the nondeterministic environment surrounding us and can be used to design and construct pervasive adaptive systems.

CSP

Communicating Sequential Processes (CSP) is a formal language consisting of mathematical models and methods to construct component processes to interact with each other by communication [8] and provides a mathematical notation for concurrency theory [9].

CSP applications are process-oriented. A system consists of processes that are sequences of instructions. Processes run separately and can communicate with other processes using channels [10]. Processes can work on different processors or even communicate across different devices. A channel is a point-to-point connection between two processes [10]. A simple version of a CSP channel consists of an in-end and an out-end performing a one-way communication. One process writes to a channel and the other reads from it. To establish a connection it is necessary to have at least two processes, one process is connected to an in-end of the channel and the other process is connected to the out-end of the channel as is shown in Figure 1.

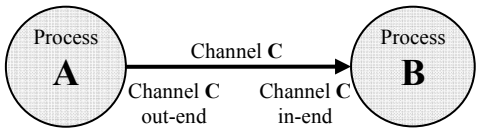


Figure 1. A simple example of a CSP network diagram.

In Figure 1 a channel is represented by an arrow to show the direction of communication. If bidirectional communication is needed an additional channel in the opposite direction should be added. Channels are unbuffered, so process A writes to the channel C and waits in an idle state for process B to be ready to read. In this way channels are unbuffered and communication occurs only when both processes are ready [10]. This simple channel is constructed in a way that data sent over it cannot be lost, provided the underlying message transport layer guarantees message delivery. In CSP based systems there are no buffers needed in the channels. Therefore the size of buffers becomes a property of a device that needs them for its own data handling, which makes system design simpler and more robust, and makes it easier to calculate memory needs.

A CSP process can have many channels connected to it. With the CSP 'alternative' programming structure it is possible to distinguish from which channel communication appeared. Therefore alternative captures non-deterministic channel behavior and permits selection between one or more input communications.

Summary

This section described a basic background of the research area. The fundamentals of ubiquitous computing, adaptation and ad-hoc networks have been described. CSP concepts have been explained as an introduction to Section 3 of this paper. A pervasive adaptive system presents many challenges and, to categorize and understand them better, an infrastructure defining software requirements is needed.

3 INFRASTRUCTURE FOR PERVASIVE SYSTEMS

Software requirements for pervasive computing were described in [11]. The software environment appropriate to support pervasive system must sustain application requirements such as:

mobility and distribution, adaptation, interoperability, component discovery, development and deployment, scalability and context awareness [11]. In this section we will describe those requirements and explain how CSP for Java (JCSP) [2]can be used to achieve pervasive system goals. JCSP is an environment allowing the development of Java applications following CSP principles. Use of Java based language enables executing processes on any device that can run a Java Virtual Machine (JVM).

Mobility and Distribution

Mobility and distribution are a natural requirement for a pervasive system. A pervasive environment is highly mobile, as users and devices can change their location; it is required for software to be mobile and distributed. Mechanisms associated with this requirement should be deployed in software and transparently for component developers [11]. Mobility should be achieved without thinking about synchronization or data migration [11]. Software for pervasive systems should have those mechanisms already deployed.

Π-calculus, a calculus for communicating systems, is a model for ubiquitous computing that proposed formal mobility constructs [12]. JCSP, following π-calculus model, offers mobility of processes, channels and code [13-15]. Process mobility enables creating processes and sending them over a channel. A mobile process can be connected and run in parallel with different processes on the node it was sent to. Mobility of channels enables sending channels with processes without a need of recreating the connection, therefore the input or output end of the channel changes its location [14]. Code mobility enables sending appropriate Java classes that are used by mobile process but are not present in a new location. Mobility in JCSP is presented and fully explained in the papers [13-15].

Adaptation

Adaptation in a pervasive environment is not only the ability to react to a changing environment but also to users' intents and needs. The pervasive infrastructure should apply adaptation to individual software components [11]. Depending on situation, some bindings should be reconfigured by adding, removing or replacing components [11]. Those processes can be reconfigured depending on adaptation techniques.

One of the main advantages of CSP system design is that simple processes can be composed into larger networks [10]. In computer science this approach is called separation of concerns [16], where concerns can represent features or functionalities of the system. In this design approach functionalities are implemented separately and composed into a larger architecture. As components of the system with different functionalities are implemented separately, the system can be more easily reconfigured by reordering and reconnecting processes, or by adding and removing processes. This type of adaptation is also supported by JSCP mobile processes and channels by moving components of the system between different devices.

Interoperability

A pervasive infrastructure is required to integrate diversity of components programmed using different languages into an infrastructure that can successfully interact and cooperate [11]. Integration of different components is a very difficult task and has been researched in Component-Based Software Engineering [17]. The ability to unify the connection mechanism would be very useful when considering a network of different components.

JCSP is equipped with network capabilities using the Communicating Process Architectures Universal Network Protocol (CPAUNP) to communicate between processes [18]. The protocol enables communication independent from the data being sent, and work towards a standardized mechanism for connecting various CPA based network architectures is in progress [18]. In particular, the protocol enables communication between Java and non-Java based devices and the simpler devices do not have to implement a complete parallel environment.

Component Discovery

A component and service discovery framework is used to organize dynamic client-server applications [19]. There are many approaches to component and service discovery [19-22]. One approach is to have a centralized repository that keeps track of all the devices and services available in the network. Any member of the network can register itself with a repository. For example JINI is a Java-based environment [21] that provides a set of application programming interfaces and network protocols for service discovery [22] and is supported by a local repository Central JINI Lookup Service. This approach is suitable for wired networks where there are always at least one device remaining in the network making the repository available for new devices [20]. In ad-hoc networks, the topology is dynamic and devices have to perform their tasks with any set of equipment. There is no central control and assuming that the repository is always available would only make an ad-hoc network less flexible. A device that performs service discovery would have to find a repository server every time it visits some network. Typical devices in ad-hoc networks can have small memory capabilities, storing information about all needed resource servers would be difficult.

Use of a distributed approach is a solution that would better fit ad-hoc network characteristics. Konark is a service discovery and delivery protocol developed in University of Florida [20] that enables integration of ad-hoc network of devices using a distributed peer-to-peer mechanism that enables devices to advertise and discover services available in the network. Service discovery is based on Web Server mechanisms using SOAP messages; therefore all the devices are using a small version of a HTTP server [20]. This distributed approach is more suitable for ad-hoc networks, but still unsuitable for devices with small memory capabilities.

In JCSP, component discovery can be performed in a simpler way. For a device and service discovery the most important issue is establishing connection. In JCSP to create a connection between two nodes only the IP address and port number are required, therefore device discovery amounts to finding those two properties. This can be performed using simple, low level sockets and broadcast capability for a local network. This is explained in Section 5 of this paper. When this information is available, the next stage to perform service discovery is to exchange information about capabilities and functions deployed in devices. This approach relies on the dynamic connection capabilities in JCSP, which are the main focus of this paper and described in Section 4 and 5.

Development and Deployment

Components in pervasive systems are required to adapt to changing environmental conditions that requires the ability to redeploy and adapt at a runtime, without restarting devices or installing new versions of components. Rapid development using multi-agent systems can be a solution to reconfiguration problems [11]. An agent is a physical or virtual unit that can identify its environment and communicate; an agent is autonomous and has the ability to accomplish tasks and achieve its goals [23].

The interpretation of agent ideas in JCSP is presented in paper[24]. The mobile agent, based on the actor model [25], is a unit that is able to move around the network, connect to some or all of the nodes and perform some pre-defined tasks. A JCSP mobile agent is a specialization of the mobile process [24] described earlier in this paper. JCSP is a framework that can offer capabilities for rapid development using an interpretation of multi-agent systems.

Scalability

The number of devices and users in a pervasive environment is not limited. Therefore the number of interactions increases and also the number of devices increases. As a result scalability is a problem for pervasive systems [4].

CSP usability for a complex emergent system was described in [26]. The study was to model artificial blood platelets using CSP concepts. This software technology scales to millions of processes per processor [26], which shows that CSP based systems are capable of dealing with scalability trivially. Since in CSP processes offering some services are composed using channels into a system that offers a natural separation of concerns at the level of services. Another advantage is that composing new system using existing components becomes easier and maintenance requires less effort.

Scalability also depends on device capabilities, and hence a limitation of the pervasive system. In dynamic adaptive pervasive system the number of threads can vary depending on a situation. A JCSP system controlling LEGO NXT robot was presented in paper [27]. In JVM a process is represented as a thread. Due to JVM and device capabilities limitation the number of simple threads was restricted to 90. The number of threads on a JVM is limited by the memory used for a thread [27]. As a result of this, if the system runs on a limited device, only limited operations can be done.

Context Awareness

Another pervasive software requirement is context awareness. Transparency of the application can be achieved by enabling a pervasive system to make decisions based upon context taken from environment and user inputs [11].

A context aware system has to take into consideration many signals from different sensors and services providing information [11]. As the nature of the environment is nondeterministic, components from the pervasive system have to be able to make nondeterministic choice between received signals.

In JCSP a nondeterministic choice is performed by alternating upon channel inputs. JCSP based systems can wait for many inputs and trigger actions according to signal received. The concurrent nature of a system based on channels and processes simplifies sending and receiving signals.

Context awareness is also associated with adapting to changing environmental conditions. The topology of the environment is also dynamic because of mobility of devices. Pervasive devices should recognize changes in the environment and adapt to it. The JCSP dynamic connection capability is a main subject of this work.

Another important issue when considering pervasive and mobile environment is energy management. Most of mobile devices are battery charged and saving energy when device is not in use would be an advantage.

CSP based processes are designed to stay in idle state when they wait for a communication from other processes and use no CPU power. Only a communication from particular channel, which process waits for, can wake it up. This way many processes in CSP based system wait for a communication form a channel or a set of channels in a state that consumes less energy than programs that would continuously send signals to sender to check its availability.

Summary

The JCSP features, described in this section, are useful when constructing a pervasive system. Mobility and distribution are supported by mobility of channels, processes and code. Adaptation techniques can be applied with ease by reconfiguring a JCSP system by reordering and reconnecting processes. Interoperability may be achieved by using the Communicating Process Architectures Universal Network Protocol (CPAUNP) between different CSP based components. Development and deployment can be accomplished by using JCSP dynamic connection capabilities and mobility of components. Scalability was shown on an example of a CSP based system presented in [26]. Context awareness by adapting to environmental conditions can be resolved using JCSP dynamic connections, which is the main focus of this paper and described in the following sections.

4. JCSP CONNECTIONS

JCSP offers local and network channels. Local channels are used to connect processes working in one device, so are executed in one JVM. This kind of channel is based on CSP principles. Network channels are used for communication between nodes on a network. JCSP connections can be divided into two categories: static and dynamic. This section introduces JCSP network channels and dynamic connection capabilities.

JCSP Network Channels

Network channels enable communication between two network nodes. Communication that uses network channels is based on CPAUNP that was developed and verified at Napier University[28]. To establish connection between two nodes only an IP address and port number of the remote node are required.

Managing the connection is easier if it is possible to check the state of the channel. The JCSP networking package enables this at the application or user process level. If the connection is lost the process trying to send data is informed, so new connections between processes can be established. If the connection was already established, the process trying to create it again is informed by the mechanisms embedded in the network architecture.

Dynamic JCSP

The JCSP package [29] enables dynamic creation of the basic components of CSP based systems: processes and channels.

Dynamic Channel Creation

In JCSP dynamic channels can be created if they are needed to establish a new connection between two processes. Both local and network channels can be dynamically created. The number of dynamic channels on a node can be large, with this number of connections it is possible to dedicate one connection to one task or type of signal (e.g. turn on/off a light). This way we can establish many connections from one node or process to another. Therefore control over many functions in the device can be performed using simple non-deterministic alternation on channel inputs. Dynamic channel creation is a very useful capability when establishing a connection between devices that know very little about each other. The connection will help devices to explore each other's potential.

Dynamic Processes Creation

JCSP libraries enable dynamic creation of an instance of a process that can be activated and connected to a network node using dynamic channels. A new process can be initialized with some data, then perform a defined task working concurrently with processes on the node. The process can be disconnected and destroyed when its task is finished. JCSP also enables dynamically created processes to be mobile[13], which makes JCSP based systems even more dynamic.

Verified Model

Concurrent systems can cause many problems, and it is necessary to have a good understanding how they behave and how to accommodate non-determinism and avoid livelock and deadlock during the design stage of the system [10], especially with systems that have dynamic topology. A tool for software verification would be useful for testing purposes and improve software reliability.

Spin is an open-source software tool that can be used for formal verification of concurrent and distributed software systems [30]. Spin can be used both as a simulator or verification tool. Spin model is a logic model checker, and accepts a specification language called Promela (Process Meta-Language). Spin is similar to JCSP model, so processes and channels are included. Importantly, Spin incorporates a concept of mobility in its design that makes its use more appropriate for the mobile distributed systems that will occur in pervasive adaptive systems. Moreover a Spin model can verify a system as a whole, including dynamic and static sections. This capability is useful when checking if a system with dynamic topology is deadlock and livelock free.

Summary

A pervasive adaptive environment can be built using components of a system working concurrently. Therefore a CSP model can be used when building a pervasive adaptive system. As deadlock and livelock can be avoided by design and the software can be verified using a Spin model, the advantages of the CSP model can be used when designing and implementing a pervasive adaptive system. Every device from the system can run processes communicating with processes on different devices. Processes on one device can be grouped in sets and cooperate. Every process can be responsible for different device capabilities, communication or synchronization with different devices.

5 DYNAMIC CONNECTIONS EXPERIMENT

This section presents an experiment with dynamic connections for synchronization using a JCSP based system. Synchronization on connection is an important issue when considering connections between devices. CSP processes can be dedicated to controlling the quality of communication between processes. Those additional processes will exclusively manage only one connection between devices.

The devices in this experiment are placed in different rooms. These devices have various capabilities and different types of communication links: wired or wireless. The experiment is to enable communication between devices from different rooms.

These tasks can be achieved using the CSP concepts and functions that JCSP offers. Priority and scheduling on channels can be accomplished by using guards and timers.

Aim of the Experiment

The aim of this experiment is to create a set of processes working on different machines performing: device and service discovery, synchronization on connection and dynamic creation of exclusive channels for data flow.

Method

Tasks of the system are to discover existing devices in rooms, create connections to perform service discovery and send commands between devices. Moreover the system is designed to create exclusive connections between devices for data flow. There are devices from the system that are created to enable communication between rooms.

Description of the system

The experiment will be carried out with two rooms and three types of devices. Rooms have different sets of devices. There are three types of devices: A, B and C shown on Figure 2.

Device type A is a device that needs to send some type of data. The device can connect to devices from Room 1 including device type B. Device type A has no knowledge about services offered in other rooms, but can recognize that device type B from its room can provide useful information about devices in other rooms.

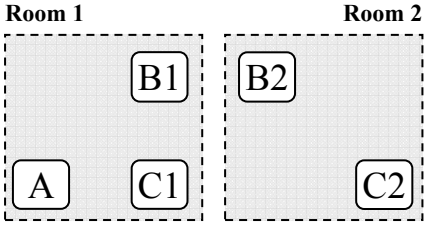


Figure 2. System design datagram.

Devices of type B are present in both rooms. A type B device is aware of devices and their capabilities in the same room as themselves and can also connect with other type B devices. Device type B performs an information service about devices outside its own room, and on request can find specific devices and arrange exclusive connections between devices. Device type

B is not a repository; devices from the system can connect to other devices from the room without asking device type B. Every device has its own device discovery capability, but connection outside the room is performed using device type B. Therefore device type B is another capability of the system, extending the connection scope, but communication inside of the room can occur without it.

Device type C is able to receive streams of data, and has specific channel inputs for processing particular types of data. In this experiment there are devices of type C in both rooms. Their capabilities are different, so depending on the data type to be sent, one of them will be used.

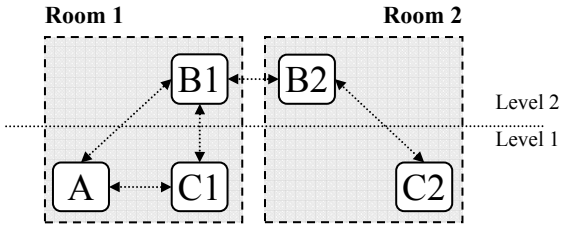


Figure 3. Connections established using a discovery service.

All the devices from the system can recognize the presence of different devices. A device discovery service is present in every device and provides information about all of the existing devices (Figure 3). Devices as shown in Figure 3 are divided into levels. Levels are used to manage initial connections between devices. Devices from level 1 can only connect other devices from the same room and have to request connection with devices from level above. This way device from lower level requests a connection with devices from one level above, and that manages the order of communication and reduces number of requests sent during the initialization stage. In Level 2 there are type B devices with high priority, that recognize devices from its own room and other devices from Level 2. In Level 1 there are devices type A and C that have only knowledge about devices in their own room and store this knowledge as a local repository.

The system consists of processes running concurrently on different devices communicating over TCP/IP links. Every device is equipped with main process responsible for key system decision making procedure and set of device discovery and information processes: Discovery Server and Discovery Client connected in an internal network (Figure 4).

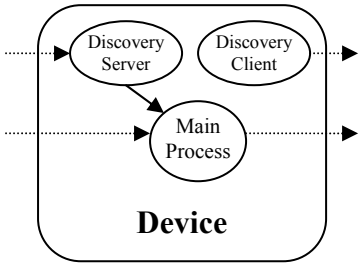


Figure 4. Devices internal network.

The network of processes in a device is initialized with localization and device capability data. The main process is responsible for establishing initial connections between the device and others, managing signals and data received by the device, send and receive commands. Moreover main processes in devices type B are responsible for dynamic creation of an instance of a process that can be connected to the network of processes already existing on the device, to manage exclusive connection between devices type A and C. The device discovery and information processes are responsible for informing other devices about their existence and receive information about other devices.

Network diagram

Let's consider a scenario that device A wants to send a particular type of data for processing to device C2. Device A first searches its own repository in order to find a suitable device. If an appropriate device is not present the device asks device B1 to provide a channel input, of a particular type, for sending data. Device B1 searches its repository and asks device B2 to fulfill a request. B2 checks the capabilities of devices in Room 2 and finds that device C2 is available to perform a request (Figure 5).

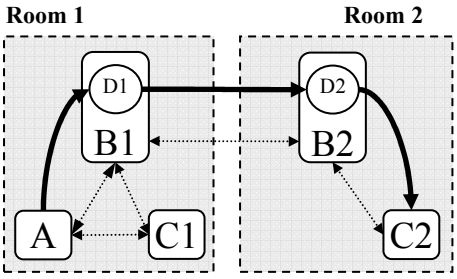


Figure 5. System design datagram.

Device B2 sends a request to device C2 for a channel input and asks it to be ready for input data. Next it creates a new process D2 (Figure 5) that will be exclusively responsible for a connection between B1 and C2. Device B1 also dynamically creates a process D1 so device A can connect to device C2. When an exclusive connection is ready B1 sends directions for device A to send data (Figure 5). When processes D1 and D2 finish passing data they are disconnected and destroyed.

In the system devices A and C2 can communicate although they are in different rooms. Devices B1 and B2 are responsible for initiating connections not maintaining them. When an exclusive connection is established processes D1 and D2 are responsible for it, devices B1 and B2 can perform different tasks and respond to requests from different devices.

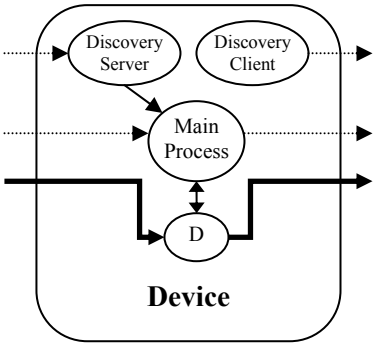


Figure 6. Exclusive connection management.

In the device internal network process D is connected to the main process (Figure 6). The exclusive connection initialized by the main process is now managed by dynamically created process D.

The device discovery capability is present in every device. As shown in Figure 6 there are two processes in every device responsible for finding new devices and recognizing that a device is no longer in use. Process DiscoveryClient sends an UDP (User Datagram Protocol) packet with information about itself to a broadcast address. DiscoveryServer process creates socket and waits for a communication. When communication occurs DiscoveryServer remembers the IP address of the sender and, if it's a new device, stores it in a local list. Processes run in parallel and every device can simultaneously send and receive packets.

Equipment and Libraries

Devices from the presented system are connected to an ad-hoc network and run Java scripts, communicating over TCP/IP network provided by a wireless router. Every device is a Personal Digital Assistant (PDA) Dell Axim X5 with Microsoft® Pocket PC operating system, processor Intel® PXA255/400MHz, RAM capacity of 64MB and IBM J9 Java Virtual Machine. Devices used in the system are in medium size, as it was defined for the purpose of this paper. The reason to use those machines is to show that the system can be run on devices with limited memory capabilities.

The Personal Digital Assistants used in the experiment have Standard Java 2 Platform Micro Edition (J2Me) 1.3 libraries.

Resulting system

The system consists of application classes with code and only limited additional JCSP libraries. The size will be a sum of those two sizes and is presented in Table 1.

Application classes:	48.5 KB
Additional JCSP libraries:	628.0 KB
Total:	**676.5 KB**

Table1. Size of the system

There are many packages in JCSP libraries, available at [2], the system only uses three of them: *jcsp.lang*, *jcsp.net2* and *jcsp.util*. The size of those libraries can be reduced, to separate classes that are used in the application. The JCSP Robot Edition (JCSPre) is a set of classes collected to support the system that controls LEGO NXT Robots [27]. The size of the *jcsp.lang* package was reduced from 223 KB to 20KB.

Devices in the system have different capabilities and perform various sets of tasks. Sizes of particular sets of classes for different devices are presented in Table 2.

Device type:	Application classes:	Additional JCSP libraries:	**Total:**
A	21.7 KB	628.0 KB	**649.7 KB**
B	24.4 KB	628.0 KB	**652.4 KB**
C	19.6 KB	628.0 KB	**647.6 KB**

Table 2. Size of the system on particular devices

The static size of the system on a particular device is no more than 700 KB (Table 2), assuming the presence of Standard JavaMe 1.3 libraries. The system is small in size, although it operates at a high level of abstraction. As the system is dynamic and adapts to a particular situation and changes in environment, the size of the system at runtime is larger than its static size.

Summary

We present a concept in which, the idea of a system is based on simple routing techniques. All devices from a proposed system have information about other devices' availability and can connect within a room with other devices. Only when a device with required capabilities is not present in the room, or a device cannot be directly connected, a device type B is used to help to create a dynamic connection between devices. This kind of connection will be virtual and devices on both sides will not be aware that it is not direct.

The system presented in this experiment dynamically creates a network of devices in rooms that discover other devices, connect appropriate ones, perform service discovery and creates dynamic connections to perform some tasks using available devices. The system has no central control and devices have different capabilities and can perform different tasks. Adaptation technique is designed to connect appropriate devices and enable creation of exclusive connections between devices for data flow. Device type B adapts to a request from devices type A by creating a mobile processes to manage requested connection type.

The system is small in size, but does operate at a high level of abstraction and organizes a dynamic adaptive pervasive environment. It performs device and service discovery, creates dynamic connections, supports sending commands between devices and manages an exclusive connection for data flow.

6 CONCLUSION AND FURTHER WORK

Pervasive adaptive computing creates new challenges for software development. This paper introduces JCSP dynamic connection capabilities and explores its usability for pervasive adaptation. We have described the infrastructure for software development in pervasive systems and useful JCSP capabilities have been presented.

Scalability in a CSP based system was demonstrated by a system modeling artificial blood platelets [26]. Mobility and distribution are supported by JCSP through mobility of channels, processes and code. Adaptation can be achieved by reconfiguring a JCSP system that allows adding, removing, reordering and reconnecting processes. Interoperability can be achieved using CPAUNP protocol between different CSP components not necessarily Java based. Readapting to changing environment at runtime can be accomplished by using JCSP dynamic connection capabilities and mobility of components.

The experiment of a simple pervasive adaptive environment was presented. The application dynamically creates a network of devices in rooms that discover other devices, connect appropriate ones, perform service discovery and create dynamic connections to perform data flow using available devices.

JCSP may not be the answer to all the challenges in pervasive adaptive computing, but some of it features might be useful. The system built using JCSP concepts is small in size and is appropriate for devices with small memory capacity. In CSP based systems simple processes can be composed into larger networks. As the system can be verified using the Spin model it

is appropriate also for large scale architectures consisting of small devices.

Interoperability is still an unsolved problem, but research in this area is in progress. JCSP advantages can be used in some parts of a pervasive system. The ability to create a system consisting of various types of components, implemented in different languages would be useful.

The size of the system presented in the experiment is measured in a static way, but the actual size of it at the runtime has not been calculated. The size of the system is crucial when considering using devices with vary small memory capabilities, so experiments to measure the runtime system size have to be performed.

ACKNOWLEDGMENT

This work is based on *jcsp.net2* package with the Communicating Process Architectures Universal Network Protocol developed by Chalmers [28] at Napier University.

REFERENCES

[1] M. Weiser, The Computer for the 21st Century, Scientific American, 1991, pp. 66-75.

[2] P.H. Welch, P.D. Austin, The JCSP Home Page. http://www.cs.ukc.ac.uk/projects/ofa/jcsp/, 1999.

[3] G. Coulouris, J. Dollimore, T. Kindberg, Distributed Systems: Concepts and Design, Pearson Education, 2005.

[4] M. Satyanarayanan, Pervasive Computing: Vision and Challenges, IEEE personal communications, 2001, pp. 10-17.

[5] C. Elliott, B. Heile, Self-Organizing, Self-Healing Wireless Networks, Aerospace Conference Proceedings, IEEE, 2000, pp. 355-362.

[6] A. Rashid, G. Kortuem, Adaptation as an aspect in pervasive computing, OOPSLA 2004 Workshop on Building Software for Pervasive Computing, Vancouver, British Columbia, Canada, 2004.

[7] R.H. Katz, Adaptation and mobility in wireless information systems, IEEE Communications Magazine 40, 2002, pp. 102-114.

[8] A.W. Roscoe, C.A.R. Hoare, R. Bird, The Theory and Practice of Concurrency, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.

[9] C.A.R. Hoare, Communicating Sequential Processes, Prentice Hall International Series in Computer Science, 1985.

[10] J. Kerridge, Lecture Notes and Forthcoming Text Book, Private Communication, Napier University, Edinburgh, 2007.

[11] K. Henricksen, J. Indulska, A. Rakotonirainy, Infrastructure for Pervasive Computing: Challenges, Workshop on Pervasive Computing INFORMATIK 01, Vienna, 2001.

[12] R. Milner, J. Parrow, D. Walker, A Calculus of Mobile Processes, Part I, Information and Computation 100, 1992, pp. 1-40.

[13] K. Chalmers, J. Kerridge, jcsp.mobile: A Package Enabling Mobile Processes and Channels, Communicating Process Architectures, 2005.

[14] K. Chalmers, J. Kerridge, I. Romdhani, Mobility in JCSP: New Mobile Channel and Mobile Process Models, Communicating Process Architectures, 2005.

[15] J. Kerridge, K. Chalmers, Ubiquitous Access to Site Specific Services by Mobile Devices: the Process View, Communicating Process Architectures, 2006.

[16] H. Ossher, P. Tarr, Using multidimensional separation of concerns to (re)shape evolving software, Communications of the ACM 44(2001) pp. 43-50.

[17] G.T. Heineman, W.T. Councill, Component-Based Software Engineering: Putting the Pieces Together, Addison-Wesley Professional, 2001.

[18] K. Chalmers, J. Kerridge, I. Romdhani, A Critique of JCSP Networking, in: I. Press, (Ed), Communicating Process Architectures, 2008.

[19] G.R.I. Golden, M. Spencer, Service and Device Discovery, McGraw-Hill Professional, 2002.

[20] S. Helal, N. Desai, V. Verma, L. Choonhwa, Konark - A Service Discovery and Delivery Protocol for Ad-Hoc Networks, Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE, 2003.

[21] R. Singh, P. Bhargava, S. Kain, State of the art smart spaces: application models and software infrastructure, Ubiquity 7, 2006, pp. 2-9.

[22] R. Gupta, S. Talwar, D.P. Agrawal, Jini Home Networking: A Step toward Pervasive Computing, IEEE Computer Society 5, 2002, pp. 34-40.

[23] J. Ferber, Multi-Agent System: An Introduction to Distributed Artificial Intelligence, Harlow: Addison Wesley Longman, 1999.

[24] J. Kerridge, J.O. Haschke, K. Chalmers, Mobile Agents and Processes using Communicating Process Architectures, Communicating Process Architectures, 2008.

[25] G. Agha, C. Hewitt, Concurrent programming using actors: Exploiting large-scale parallelism Foundations of Software Technology and Theoretical Computer Science 206/1985, Springer Berlin / Heidelberg, 1985.

[26] P.H. Welch, F.R.M. Barnes, F.A.C. Polack, Communicating Complex Systems, 11th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'06), 2006.

[27] J. Kerridge, A. Panayotopoulos, P. Lismore, JCSPre: the Robot Edition to Control LEGO NXT Robots, Communicating Processes Architectures, York, UK, 2008.

[28] K. Chalmers, Investigating Communicating Sequential Processes for Java to Support Ubiquitous Computing, Napier University, 2008.

[29] P.H. Welch, J.R. Aldous, J. Foster, CSP Networking for Java (JCSP.net), in: P.M.A. Sloot, C.J.K. Tan, J.J. Dongarra, A.G. Hoekstra, (Eds), International Conference Computational Science - ICCS 2330, Springer Berlin / Heilderberg, Amsterdam, The Netherlands, 2002, pp. 695-708.

[30] G.J. Holzmann, The SPIN Model Checker: Primer and Reference Manual, Addison-Wesley, 2003.