Performance Modelling of Content Adaptation for a Personal Distributed Environment

Jie Ding (j.ding@ed.ac.uk)
{LFCS, School of Informatics; Institute for Digital Communications, Joint Research Institute for Signal & Image Processing, School of Engineering and Electronics}, University of Edinburgh, The King's Buildings, Edinburgh, EH9 3JL, UK.

Jane Hillston (jane.hillston@ed.ac.uk)
LFCS, School of Informatics, University of Edinburgh, The King's Buildings, Edinburgh, EH9 3JL, UK.

Dave Laurenson (dave.laurenson@ed.ac.uk)
Institute for Digital Communications, Joint Research Institute for Signal & Image Processing, School of Engineering and Electronics, University of Edinburgh, The King's Buildings, Edinburgh, EH9 3JL, UK.

Abstract. In the new era of wireless, mobile connectivity the possibilities of *anything, anytime, anywhere* access are becoming a reality. In order to deliver services and content to users in a variety of contexts, content adaptation strategies need to be developed and assessed. In this paper we present an approach to performance evaluation and capacity planning, based on a high-level system description formalism, which can be readily extracted from a system design. We demonstrate its usefulness in assessing the design of a content adaptation framework being developed under the auspices of the virtual centre in mobile communications.

Keywords: Performance Modelling, Capacity Planning, Content Adaptation, Personalisation, PEPA

1. Introduction

As networks become more sophisticated, both in terms of their underlying technology and the applications running upon them, it is crucial that users' expectations and requirements are anticipated and met. In particular, users are not concerned with the technological aspects of communications *per se*, however at present they need to be aware of a multitude of detail about equipment and benefits of one communication strategy over another, as well as how to connect systems together to achieve the communication that they desire. As the number of possible communication strategies increases, so does the complexity of negotiating the most appropriate method of delivering content that the user wishes to access. Enabling users to access services in the future requires a means of hiding the complexity involved in the communication of the content, and its delivery mechanism, from the user, empowering the user to access anything at anytime from anywhere.

The virtual centre of excellence in mobile communications (MobileVCE) is addressing this area in the programme entitled "Removing the Barriers to Ubiquitous Services". The programme is investigating the tools and techniques essential to hiding complexity in the heterogeneous communications environment that is becoming a reality. In particular, the work makes use of agents that manage personal preferences, and control the adaptation of content to meet the system requirements for a user to view content they have requested. Interaction between the entities in the user-controlled devices and the network that are required to achieve this then becomes a significant issue.

One important set of expectations are those relating to performance and the responsiveness of applications. However, designing a system to ensure that it will perform satisfactorily can be difficult. Of course, the users' demands on the design must be balanced by the requirements of the system manager, who will seek to develop a system which is robust with well-used resources, without over-provisioning. Thus there is a trade-off between capacity planning and application performance engineering.

In this paper we present an approach to performance evaluation and capacity planning, based on a high-level system description formalism, which can be readily extracted from system design. This allows early analysis of potential designs and configurations to assess their ability to satisfy user expectations with respect to measures such as throughput and response time. From the other perspective, system managers are keen to ensure that resources within a system are efficiently utilised. We will show how the same system model can be analysed to study the degree of replication which may be needed within a network to balance effective use of resources and user satisfaction.

Models are constructed using the stochastic process algebra PEPA [10]. A process algebra is a compositional description technique which allows a model of system to be developed as a number of interacting *components* which undertake *actions*. In addition to the system description aspects the process algebra is equipped with techniques for manipulating and analysing models, all implemented in tools. Thus analysis of the model becomes automatic once the description is completed. In a *stochastic* process algebra additional information is incorporated into the model, representing the expected duration of actions and the relative likelihood of alternative behaviours. This is done by associating an exponentially distributed random variable with each action in the model. This quantification allows quantified reasoning to be carried out on the model. Thus, whereas a process algebra model can be analysed to assess whether it behaves correctly, a stochastic process algebra model can be analysed both with respect to correctness and timeliness of behaviour.

Once such a model has been constructed a variety of different analysis approaches are accessible from the single model:

- The model may be used to derive a corresponding (discrete state) continuous time Markov chain (CTMC) which can be solved for both transient and equlibrium behaviour, allowing the calculation of measures such as expected throughput, utilisation and response time distributions.

- The model may be subjected to scalability analysis in which large numbers of users are injected into the system and a continuous approximation of the CTMC is made and solved as a set of nonlinear ordinary differential equations. This analysis will test the impact of heavy load on the ability of the system to respond to users. Such experiments can be used to investigate the appropriate level of replication of components of the system to cope with anticipated demand.

– Desirable properties of the system can be expressed as logical formulae which may be automatically checked against the formal description of the system, to test whether the property holds. This can be particularly useful in checking that protocols behave appropriately and that certain desired properties of the system are not violated.

In this paper we illustrate the use of PEPA modelling, and the first two analysis approaches, on the content adaptation framework being developed within the project to enable content to be appropriately formatted and sized for a particular users' needs.

The remainder of this paper is organised as follows. In Section 2 we present some background material about the MobileVCE project and the PEPA modelling formalism, its analysis techniques and supporting tools. The architecture of the proposed content adaptation framework is introduced in Section 3 together with its PEPA model. In Section 4 we detail the experiments which have been conducted on the model, assessing the sensitivity of the framework to the performance of individual components, and the scalability of the framework under increasing loads. We discuss the work and future directions and present some conclusions in Section 5.

2. Background

2.1. MobileVCE Project

MobileVCE is the operating name of the Virtual Centre of Excellence in Mobile and Personal Communications Ltd, a collaborative partnership of around 20 of the world's most prominent mobile communications companies and 7 UK universities each having long standing specialist expertise in relevant areas. MobileVCE engages in industrially-led, long term, research in mobile and personal communications [6].

The third programme of the MobileVCE, core 3, developed the concept of a personal distributed environment (PDE)[9]. This concept is a user-centric view of communications in a heterogeneous environment, and is defined as those nodes over which a user has control. The devices need not reside in the same place, or all be connected to the same network, but they cooperate together to fulfil user requests. The PDE concept defines techniques for discovering services and capabilities of devices, determining appropriate routes for communication, implementing security and establishing trust relationships, and negotiation for the provision of services.

Ubiquitous service represents a major future revenue stream for service providers, telecommunication operators and pervasive technology manufacturers, since Bluetooth, WiFi, WiMAX, UWB and more, are bringing the dream of ubiquitous access closer to reality. The “Removing the Barriers to Ubiquitous Services” programme (hereafter called the Ubiquitous programme) aims to build on the personal networking capabilities provided by the PDE and develops this further from three perspectives — user, network and content/service[6].

This paper presents the use of PEPA to analyse the performance of the mechanisms used to adapt content and services to the users’ needs. The system is based, primarily, on the use of a personal assistant agent to specify and control what the users needs and constraints are to receive a particular service or content. This interacts with a content adaptation mechanism that resides in the network. Performance of the system depends upon an efficient negotiation, content adaptation, and delivery mechanism. The analysis will benefit the system not only in seeking the most efficient operation pattern but also the construction and setting of the framework itself. Before turning to the content adaptation, we first give a brief introduction to PEPA in the following subsection.

2.2. Introduction to PEPA

Performance Evaluation Process Algebra (PEPA)[10], developed by Hillston in the 1990s, describes a system as an interaction of the components and these components engage in activities. PEPA is both a timed and stochastic extension of classical process algebras such as CCS [13] or CSP[1]. It can be regarded as a high-level model specification language for low-level stochastic models, which makes it suitable for extracting performance measures as well as deducing functional properties of the system. In this subsection, we represent a brief introduction to PEPA; more details about PEPA can be found in [10].

We assume that there is a set of possible types of action, denoted $\mathcal{A}$. Each activity in PEPA can be defined as a pair (α, r) where $\alpha \in \mathcal{A}$ is the action type and r is the activity rate. This rate specifies the parameter of an exponential distribution so $r \in \mathbb{R}^+$. If a component P completes an activity (α, r) and then behaves as Q, it is denoted $P \overset{def}{=} (\alpha, r).Q$ and the transition may be denoted as $P \xrightarrow{(\alpha,r)} Q$.

The following presents the name and interpretation of combinators used in the PEPA language, which express the individual activities and interactions of the components.

2.2.1. *Syntax*

Prefix: $(\alpha, r).P$
Such a component will subsequently behave as P after it carries out the activity (α, r), which has action type α and a duration which satisfies exponential distribution with parameter r.

Choice: $P + Q$
The component $P + Q$ represents a system which may behave either as P or as Q. The activities of both P and Q are enabled. If one of them is chosen, which depends

on whether its activity is completed first, the other will be discarded and the system will then behave as the derivative resulting from the evolution of the chosen one.

Cooperation: $P \underset{L}{\bowtie} Q$
The cooperation combinator is in fact an indexed family of combinators, one for each possible set L of visible action types. L, the cooperation set, determines the interaction between P and Q. For any activity whose action type is contained in L, P and Q must cooperate to achieve the activity. However, they will proceed independently and concurrently with any activity whose action type is not included in L.

Parallel: $P||Q$
The component $P||Q$ represents two concurrent but completely independent components, meaning the cooperation set is empty. This is simply a shorthand notation for $P \underset{\emptyset}{\bowtie} Q$.

Hiding: P/L
Hiding makes the activities whose action types in L invisible for external observer. The component P/L behaves as P except that any activities of types within the set L are hidden.

Constant: $A \overset{\mathit{def}}{=} P$
Constants are components whose meaning is given by a defining equation such as $A \overset{\mathit{def}}{=} P$, which gives the constant A the behavior as the component P.

2.2.2. *Underlying Markov Process and Performance Measures*

Like all process algebras, PEPA is given a formal semantics. This is a set of transition rules which define the evolution rules for each combinator of the language (see Appendix A for details). From these, given any PEPA expression it is possible to derive all possible states of the model and the transitions between them, the *derivation graph.* From this derivation graph the underlying stochastic process is readily extracted. Due to the memoryless property of the exponential distributions, and since all transitions have an exponential distribution, it is straightforward to see that this stochastic process is a discrete state space, continuous time Markov chain.

In order to facilitate analysis of this underlying model it is stored in the form of its infinitesimal generator matrix. This is extracted automatically by the PEPA tools. By solving the matrix equation characterising the global balance equations we are able to derive the steady state probability distribution for the Markov chain, which the tool relates back to the original PEPA structures. Similarly the matrix may be used as the basis for transient analysis, allowing measures such as response time distributions to be calculated. Such measures can facilitate model verification and system optimization. This will be illustrated in the following sections.

3. Content Adaptation

Users, requesting content from a provider, wish that content to be usable in a specific device or devices. Currently a content provider may provide a number of formats of a particular content to suit a selection of devices, or they may only provide a single format of the data. With the rapidly growing variety of devices that a user may expect to use for delivering a particular content, providing content tailored to each device becomes an infeasible task for the content provider [14]. Thus, in order for a user, who needs the content in a different format, to be able to make use of that content, a transformation needs to take place. In the wider context, not only may transformation from one format to another be required, but additionally the content may need to be modified, for example its bit-rate reduced, in order to meet quality of service constraints. This process is called content adaptation. The adaptation, itself, may take place within the domain of the service provider, the domain of the user, or may take place within the network as a third party service.

C.Canali et al. proposed several distributed architectures for the efficient delivery of content adaptation services, based on which some performance analysis, especially the distribution of the response time are presented [2–4]. The performance evaluation of the adaptation management system is important, not only because it can measure the efficiency of the system's operation and help to seek an efficient working pattern, but also it may direct the modification and improvement of the adaptation framework construction.

3.1. Logical Architecture and Working Cycle

3.1.1. *Logical Architecture*

Figure 1 ([7]) depicts the logical architecture of the content/service adaptation management framework for the MobileVCE Ubiquitous programme. It contains high level representations of the main functional units.

At the User side, there are two key functions: the Personal Assistant Agent (PAA) and the Personal Content Manager (PCM). The PAA is proposed to reduce the perceived complexity of future multi-device personal area networks by proactively managing the modes, the functions, the applications, and the connectivity of user's devices. In addition, employing the PCM can effectively store content throughout user's environments, maximising availability and efficiency as well as retrieving the content in the most appropriate manner. The Device Management Entity (DME) was conceived as part of the MVCE Core 3 PDE program and acts as a platform for the PAA and PCM to operate over.

The Adaptation Manager(AM) consolidates the various types of context not only from the user but also from the Content/Service Provider (CSP), and then assimilates and distils the contexts into relevant rules so that actions may be determined by the decision logic.

The Content Adaptor (CA) organises that actual adaptation processes based on maintaining a profile of all of the available adaptation mechanisms, contacting external adaptation mechanism providers and consolidating their capabilities to meet the adaptation requests, passed down from the AM.

The Dispatcher acts as a buffer, transporting the context information from the PAA to the AM, and delivering the adapted content from the CA to the PDE, which forms the logical interface between the PDE (and Personal Entities) and the Content and Service Adaptation Framework as a whole.

The above introduction is based on [7, 8, 12]. For details, please refer to them.

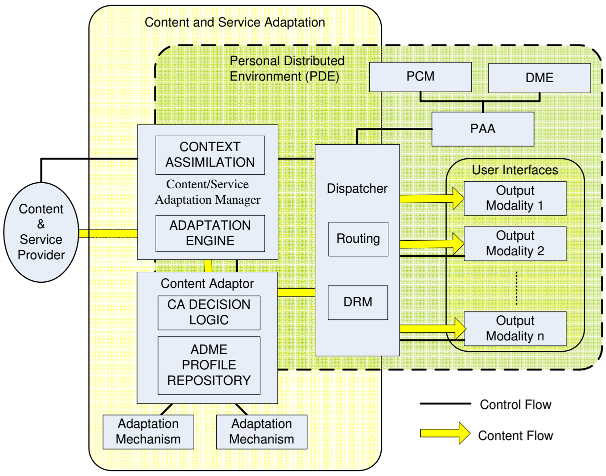


Figure 1. Logical Architecture of Content Adaptation Management

3.1.2. *Working Cycle*

Figure 2 illustrates the operation of the adaptation management - The AM receives the requests for adaptation (with the user's context) from the PAA via the Dispatcher, and the context from the CSP, and ascertains appropriate options for content/service translation from the available options and constraints. The choices or parameters for translation are then given to the CA for adaptation. The newly adapted content or service is then delivered to the user through the Dispatcher. More details are specified in the following working cycle based on Figure 2 (for convenience, we omit the component of the Dispatcher and only use the PDE to represent the all entities of the user's side).

Working Cycle

1. The User makes a content request.

2. The PDE decides whether the request is reasonable (suitable) or not. If not, then return the request, go back to 1. If yes, then it encapsulates it into a context which is expressed as an "Internal Content/Service Request" and sends it to the AM.

3. After receiving the request from the PDE, the AM asks for and receives the "Content/Service Context" from the CSP.

4. The AM assimilates and distills the contexts both from the PDE and the CSP, and subsequently makes a request, called an "Adapted Request", to the CSP. This adapted request asks for the content in an appropriate form, by taking into account the PDE's satisfaction, network conditions, the CSP's repository, and even the competence and capacity of the CA.

5. After receiving this adapted request from the AM, the CSP will make a decision of whether and what kind of C/S should be provided, and then send the C/S source or the information of "no C/S available" to the AM.

6. If the source is received, the Adaptation Decision Engine of the AM will determine the adaptation parameters and then pass them with the source to the CA. Otherwise, in the case of no source available, the Adaptation Decision Engine will forward this information directly to the PDE, then go to 8.

7. The CA starts its "Adaptation" after it receives the parameters and source from the AM, and then passes the adapted content to the user when the adaptation is finished.

8. After receiving the adapted content from the CA or the information that no source is available from the AM, the PDE forwards it to the user's interface. A working cycle is completed and then returns to 1.

3.2. PEPA Model

In this subsection we define the PEPA model based on the architecture and working cycle presented in the previous subsection. The model is comprised of four components, corresponding to the four major components of the architecture, i.e., the PDE, the AM, the CA and the CSP. It is important to note that not all aspects of the components behaviour are represented in detail. The level of abstraction is chosen to be sufficient to capture the time/resource consuming activities and to ensure that the correct interactions between components are maintained.

Each of the components has a repetitive behaviour, reflecting its role within the working cycle. These are represented diagrammatically in Figures 3 and 4. Below we show the PEPA definitions for the PDE and AM components. The CA and CSP components are defined similarly.

PDE : The behaviour of the PDE begins with the generation of a request for content adaptation, represented as activity *pde_ext_cs_req* (working cycle step 1). The rate here reflects the expected rate at which the user will submit requests for content adaptation. The next event is the decision about the feasibility of this request

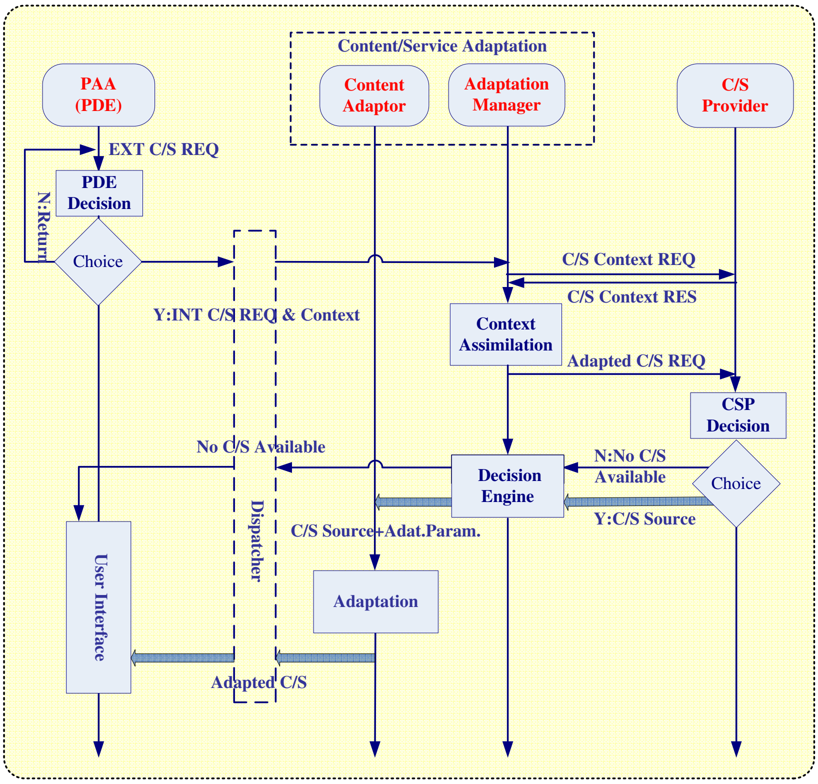
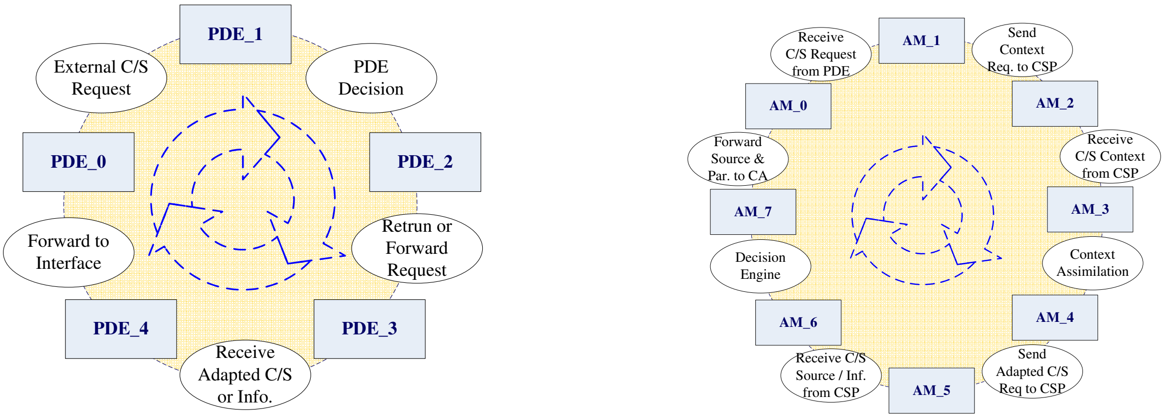


Figure 2. Working Flow of Content Adaptation Management


PDE_1
External C/S Request
PDE Decision
PDE_0
PDE_2
Forward to Interface
Retrun or Forward Request
PDE_4
PDE_3
Receive Adapted C/S or Info.
AM_1
Receive C/S Request from PDE
Send Context Req. to CSP
AM_0
AM_2
Forward Source & Par. to CA
Receive C/S Context from CSP
AM_7
AM_3
Decision Engine
Context Assimilation
AM_6
AM_4
Receive C/S Source / Inf. from CSP
AM_5
Send Adapted C/S Req to CSP

(a). PDE

(b). AM

Figure 3. PDE and AM's Working Cycle

(working cycle step 2). The subsequent activity depends on the outcome of the decision. In PEPA this is represented as two competing activities, both of type *pde_decision*, with rates which reflect their relative probabilities. If a request is passed to the AM (*pde_int_cs_req*), it is assumed to be synchronous request and the PDE waits for a response PDE_3. The model reflects that there are two possible responses, by having two possible activities of receiving the information that no source is available or receiving the adapted content, *csp_no_cs* and *ca_adapted_cs* respectively. In either case the final action of the PDE is to send appropriate information to the user interface, $pde\_interface$ (working cycle step 8).

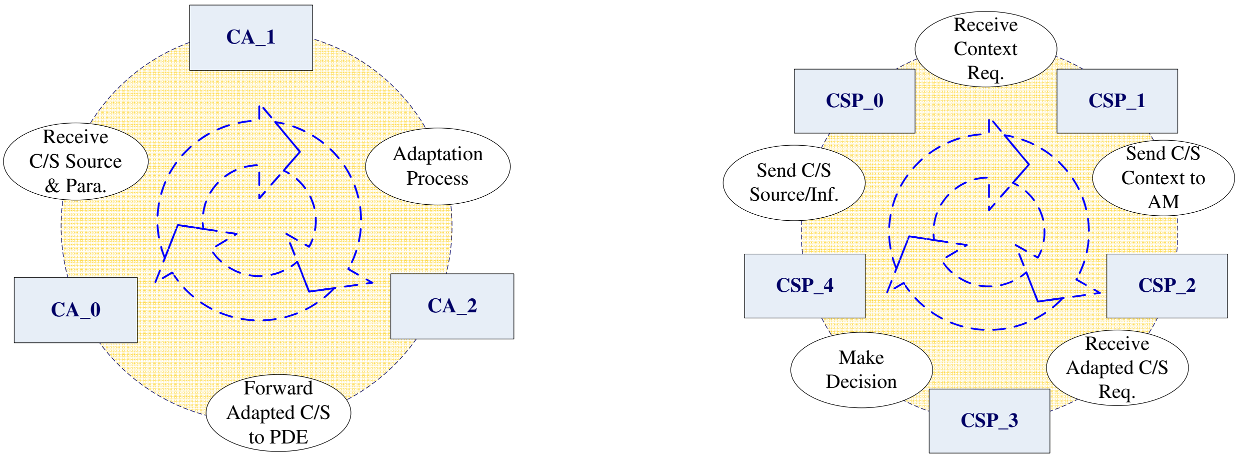
$$\begin{aligned}
PDE_0 &\stackrel{def}{=} (pde\_ext\_cs\_req, r_{pde\_ext\_cs\_req}).PDE_1\\
PDE_1 &\stackrel{def}{=} (pde\_decison, r_{pde\_decison}).PDE_2\\
PDE_2 &\stackrel{def}{=} (pde\_return, r_{pde\_return}).PDE_0 + (pde\_int\_cs\_req, r_{pde\_int\_cs\_req}).PDE_3\\
PDE_3 &\stackrel{def}{=} (csp\_no\_cs, r_{csp\_no\_cs}).PDE_4 + (ca\_adapted\_cs, r_{ca\_adapted\_cs}).PDE_4\\
PDE_4 &\stackrel{def}{=} (pde\_interface, r_{pde\_interface}).PDE_0
\end{aligned}$$

Adaptation Manager : After receipt of a request from the PDE, *pde_int_cs_req*, the AM goes through a sequence of activities corresponding to steps 3 and 4 in the working cycle. The request to the CSP is synchronous and the AM waits for a response. As in the PDE there are two possible response actions, depending on the response from the CSP. If “no C/S available” this is communicated immediately from the CSP to both the AM and the PDE through the shared activity *csp_no_cs* (working cycle step 5). Alternatively the AM will make its determination, *am_determination*, and pass the parameters with the content source to the CA, *am_cs_source_para* (working cycle step 6).

$$\begin{aligned}
AM_0 &\stackrel{def}{=} (pde\_int\_cs\_req, r_{pde\_int\_cs\_req}).AM_1\\
AM_1 &\stackrel{def}{=} (csp\_context\_req, r_{csp\_context\_req}).AM_2\\
AM_2 &\stackrel{def}{=} (csp\_context\_res, r_{csp\_context\_res}).AM_3\\
AM_3 &\stackrel{def}{=} (am\_context\_assimilation, r_{am\_context\_assimilation}).AM_4\\
AM_4 &\stackrel{def}{=} (am\_adapted\_cs\_req, r_{am\_adapted\_cs\_req}).AM_5\\
AM_5 &\stackrel{def}{=} (csp\_no\_cs, r_{csp\_no\_cs}).AM_0\\
&\quad + (csp\_cs\_source, r_{csp\_cs\_source}).AM_6\\
AM_6 &\stackrel{def}{=} (am\_determination, r_{am\_determination}).AM_7\\
AM_7 &\stackrel{def}{=} (am\_cs\_source\_para, r_{am\_cs\_source\_para}).AM_0
\end{aligned}$$

The other two components are defined similarly and can be found in Appendix B.

The final part of the definition of the model is the *system equation* which specifies how the complete model is constructed from the defined components. It specifies how many copies of each entity there are present in the system, and how the components interact, by forcing cooperation on some of the activity types. For our model the system



(a). CA

(b). CSP

Figure 4. CA and CSP's Working Cycle

equation is as shown below, where we indicate that the exact number of independent copies of the PDE component is a variable of some of our experiments.

$$System \overset{def}{=} \Big((PDE \| \cdots \| PDE) \underset{L_1}{\bowtie} (AM \underset{L_2}{\bowtie} CA)\Big) \underset{L_3}{\bowtie} CSP,$$

where

$$L_1 = \{\mathit{pde\_int\_cs\_req}, \mathit{csp\_no\_cs}, \mathit{ca\_adapted\_cs}, \}, \quad L_2 = \{\mathit{am\_cs\_source\_para}\},$$

$$L_3 = \{\mathit{csp\_context\_req}, \mathit{csp\_context\_res}, \mathit{am\_adapted\_cs\_req}, \mathit{csp\_no\_cs}, \mathit{csp\_cs\_source}\}.$$

3.3. Parameters Setting

As in all quantitative modelling it is important that the parameters used within the model are as realistic as possible if the analysis is to generate useful results. In our model each of the activities in the model must be assigned an appropriate activity rate. In order to do this we used published results in the literature, where similar systems have been constructed or modelled [2–4]. The resulting parameter values are shown in Table I, together with the intuitive explanation of each parameter. Note the rate represents how many activities can be completed in unit time, which in our case is one second. The probabilities of the PDE returning the request and the CSP choosing the "no C/S available" response, are both set to 0.1. The final additional parameter is the number of independent PDE entities active within our system. In these initial experiments we assume that this parameter has value 1, unless otherwise stated.

Table I. Parameters Setting

Type	Role	Average Time(ms)	Rate
$pde\_ext\_cs\_req$	user inputs an C/S request	1000	1
$pde\_decision$	PDE judges whether the request is reasonable	30	33.3
pde_return	PDE returns the request	25	40
$pde\_int\_cs\_req$	PDE forwards the internal C/S request	60	16.7
$pde\_interface$	PDE forwards the adapted C/S to user's interface	100	10
$am\_adapt\_cs\_req$	AM sends adapted C/S request to CSP	40	25
$am\_context\_assimilation$	AM's context assimilation	160	6.25
$am\_determination$	AM determines adaptation parameters	50	20
$am\_cs\_source\_para$	AM sends source and adaptation parameters to CA	100	10
$ca\_adaptation$	CA's adaptation process	800	1.25
$ca\_adapted\_cs$	CA forwards adapted C/S to PDE	150	6.7
$csp\_context\_req$	AM asks for CSP's context	40	25
$csp\_context\_res$	CSP sends the context to AM	40	25
$csp\_decision$	CSP determines what should be provided	40	25
csp_no_source	information of no source available	40	25
csp_cs_source	CSP provides C/S Source to AM	100	10

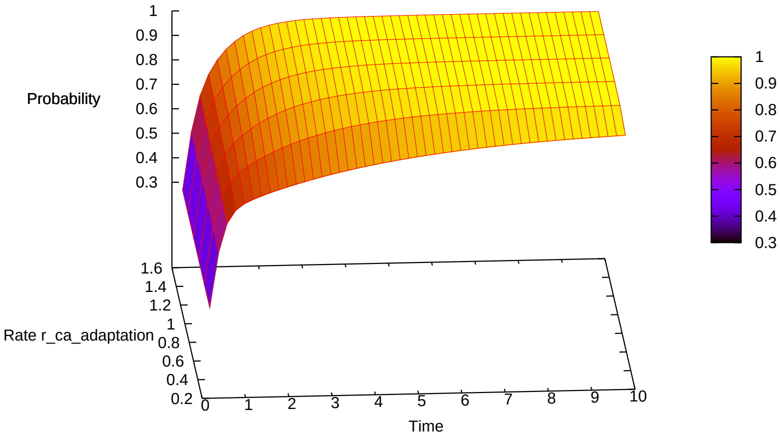
4. Performance Evaluation

Experiments have been conducted using the PEPA Workbench and associated tools. More details on these tools can be found at `http://www.dcs.ed.ac.uk/pepa`.

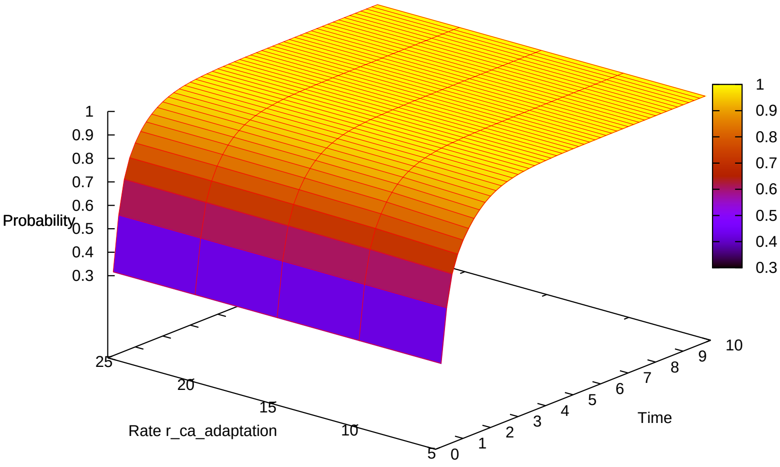
4.1. Response Time

Response time is the interval between the input of a request by the user and the return of the adapted content, i.e., the user's waiting time for the content. Many papers in the literature consider this measure as important since it has a major impact on the user's satisfaction. The response time is determined by a number of factors, namely:

the device capabilities and operating speed of each element in the connection; the interactions between devices; the complexity of the adaptation task; and the network characteristics and status (including network latency and bandwidth).



(a). Response Time vs Adaptation Rate



(b). Response Time vs AM Determination Rate

Figure 5. Response Time Changes with the Rate

We give the cumulative distribution function of the system's response time under our previous parameter setting. These results are similar to the ones in [2–4]. Figure 5(a) shows that the response time has a strong dependence on the content adaptation rate, when the adaptation rate is less than 1, corresponding to an average adaptation time of 1 second. Conversely, Figure 5(b) shows that the AM's rate of selecting adaptation parameters, with average latency in the process of between 40 ms and 200 ms, has little effect on the response time of the system. From a systems perspective, if complexity in the AM can be traded off with complexity in the CA, perhaps by a more involved process of selecting adaptation parameters, the response time could be lowered.

4.2. System's Adaptation Speed and Utilisation Efficiency

4.2.1. *Adaptation Throughput*

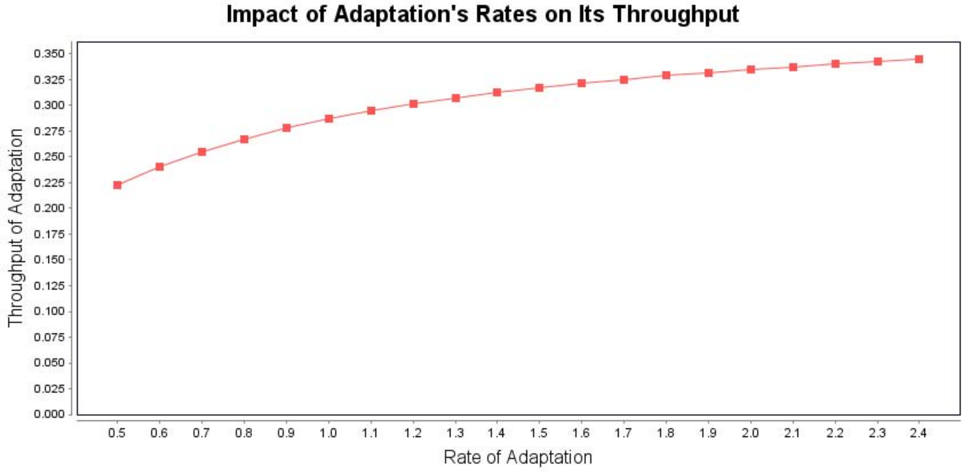


Figure 6. Impact of the Adaptation Rate on Its Throughput

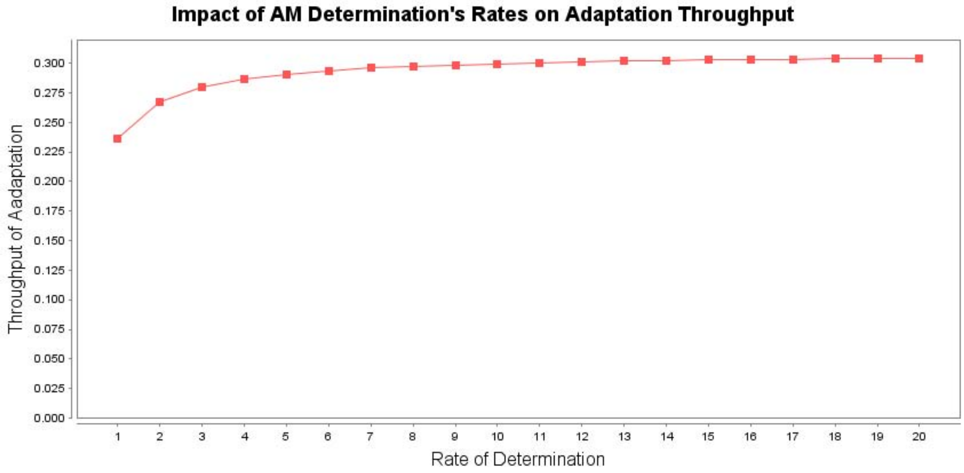


Figure 7. Impact of the AM Determination Rate on the Adaptation Throughput

Following [5, 10], the definition of throughput of an activity is the average number of the activities completed by the system during one unit time (1 second). Thus, the throughput of the CA's "Adaptation" reflects how fast the system runs the adaptation. Obviously, increasing the rate of adaptation or decreasing its delay can improve its throughput. From Figure 6 and Figure 7, it can be observed that the throughput of adaptation is sensitive not only to its own rate but also to the rate of AM's determination when the latter approaches lower rates. At rates of less than 5, which is significantly different from its normal rate of 20, AM's determination has as strong an influence on throughput as the adaptation rate itself. Since the AM's two activities, "Context Assimilation" and "Determination", are sequential and have similar durations, it can be concluded that the AM's context assimilation has a similar influence on the adaptation throughput and response time. Thus, we only focus on the AM's determination and the CA's activities.

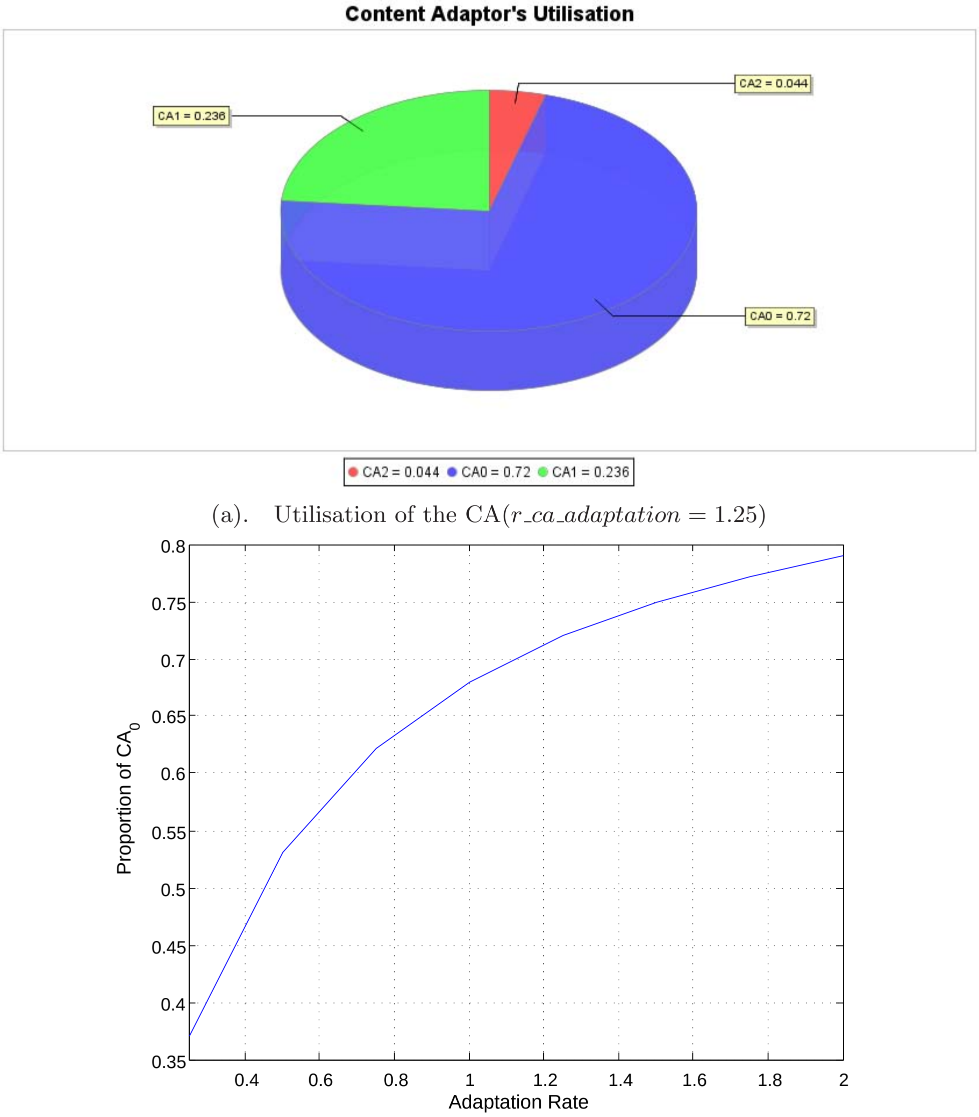
4.2.2. *CA's Utilisation*

The system manager's interests include not only the speed of the system's operation but also the system's utilisation efficiency. Increasing the adaptation rate speeds up the running of the whole system, however this does not imply that the system is more efficiently utilised. To illustrate, we introduce the definition of utilisation[10], i.e., the proportion of time that a component spends in different states. For the CA, there are 3 states, CA_0, CA_1 and CA_2. CA_0 is the state of waiting for and receiving the content source and adaptation parameters from the AM while CA_1 and CA_2 are the states corresponding to the adaptation process and sending the adapted content to the PDE, respectively. If there are no synchronisation points, the proportion of time spent in these 3 states is proportional to their average time for completing the respective activities. In this case we would expect CA_0's occupancy to be small. However, the CA has to synchronise with other events, which means that CA_0 corresponds to the longest time in this component with a proportion of about 72% (see Figure 8 (a)). Moreover, the smaller the adaptation's duration is, the bigger CA_0's proportion (see Figure 8 (b)). When the CA operates without any synchronisation delays, CA_0's proportion could be 9.5% (or $\frac{100}{100+800+150}$), thus the CA has sufficient capacity to serve more requests, and be better utilised.

4.3. Capacity planning

Figure 9(a) illustrates the effect of increasing the adaptation rate. As adaptation rate increases, the adaptation throughput increases, while the response time and the utilisation efficiency decrease. Thus, an improved user experience, as measured by response time, can be obtained through improving the adaptation rate, at the expense of increased redundancy in the content adaptor.

A full network would comprise many PDEs that co-exist and share system resources. This has the effect of changing the load on system components, altering throughput and waiting times. Figure 10(a) shows that the CA's waiting time decreases as the number of the PDEs increases, due to more frequent requests being received, while Figure 10(b) illustrates that the throughput of adaptation is increasing, due to the number



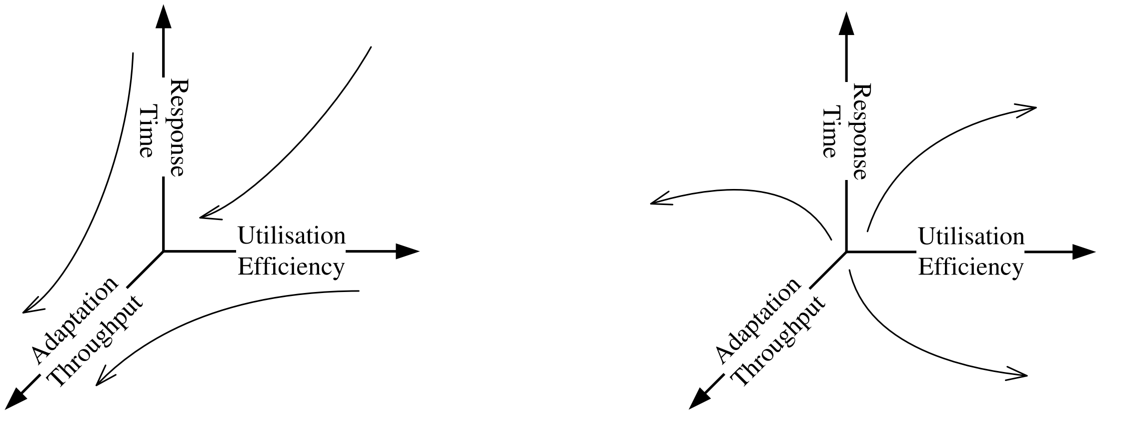
(a). Utilisation of the CA($r\_ca\_adaptation = 1.25$)

(b). Proportion of CA_0 vs Adaptation Rate

Figure 8. Utilisation of the CA

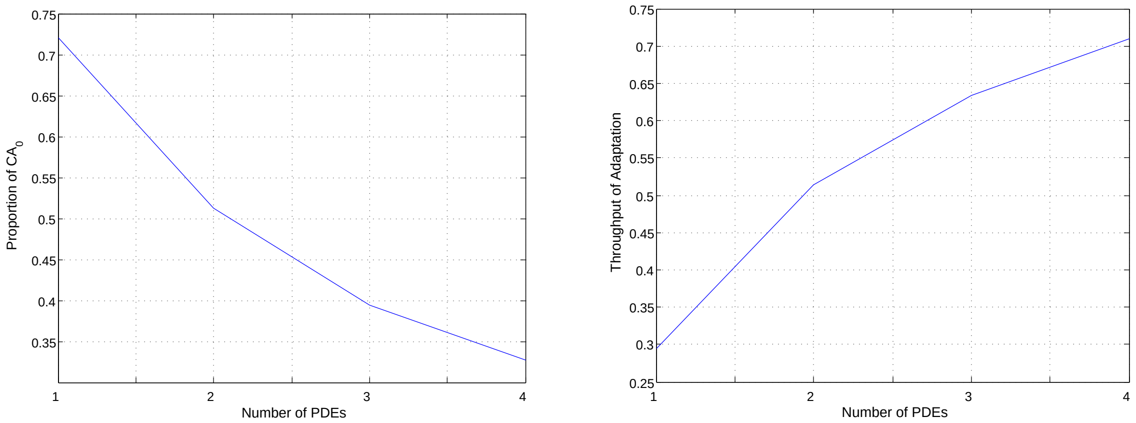
of requests that are being served. For example, four PDEs result in 0.7 adaptations per second being completed compared with less than 0.3 adaptations in the case of 1 PDE.

On the other hand, more PDEs, which make other components more busy, results in longer user waiting times or the system's response time in general (see Figure 11). Figure 9(b) illustrates the effect of increasing the number of PDEs being supported



(a). Impact of Increasing the Adaptation Rate (b). Impact of Increasing the Number of PDEs

Figure 9. Adaptation Rate and the Number of PDEs' Impact on the System Performance



(a). Utilisation Efficiency vs the Number of PDEs (b). Adaptation Throughput vs the Number of PDEs

Figure 10. Throughput and Utilisation Changes with the Number of PDEs

by a system on adaptation throughput, response time and utilisation of the CA. It shows that there is a trade-off between the response time that can be achieved and the loading placed on the adaptation process in terms of achieved throughput and utilisation. This information can be used in the planning process to appropriately dimension a system to achieve its potential.

Unfortunately, the size of the state space of the underlying Markov chain increases sharply with the number of the PDEs (see Table II). If 5 PDEs are involved in the system, there will be 24064 states in the system and the steady state probability distribution (from which the throughput and utilisation originate) cannot be computed easily due to the computational complexity. So, to avoid the state space explosion

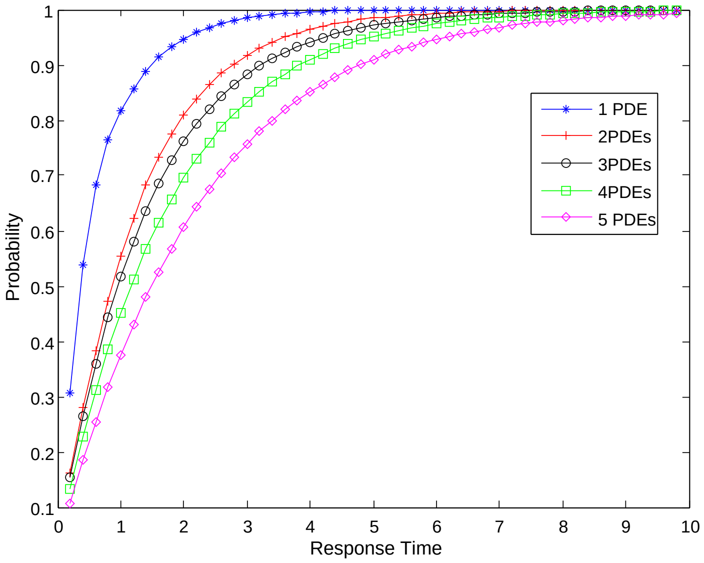


Figure 11. Response Time Changes with the Number of PDEs ($r\_ca\_adaptation = 0.8$)

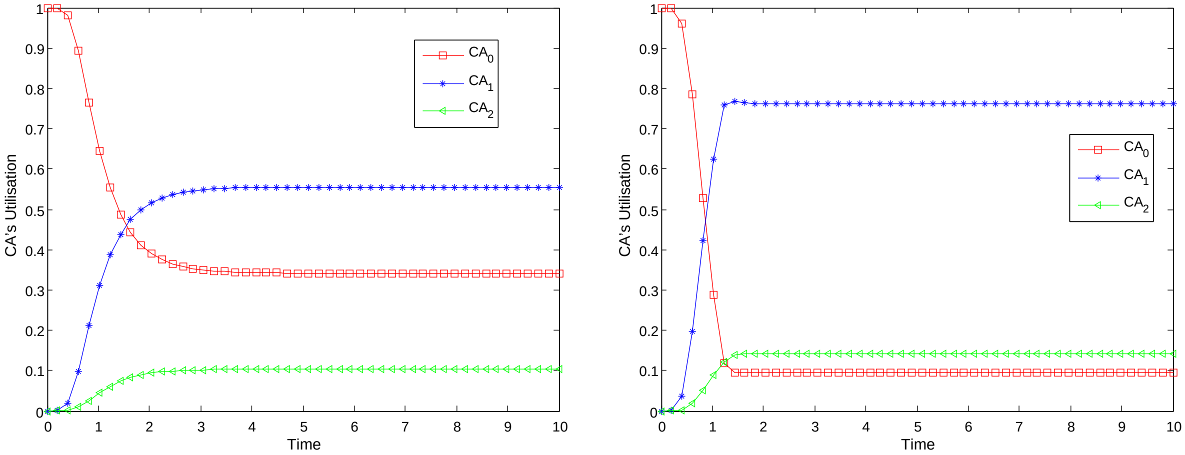
Table II. Size of State Space of Underlying Markov Chains

Number of PDEs	Size of the State Space
1	14
2	112
3	736
4	4352
5	24064

problem of the large scale system we follow the technique in [11], using the ODEs (ordinary differential equations) to approximate the PEPA model.

Consider the case that there are 100 PDEs, 100 AMs, 100 CSPs involved in the system. By varying the number of CAs in the system, the effect on performance can be observed. 20 CAs are enough for the adaptation task where the time spent in the idle state is about 30% (see Figure 12(a)). It seems that 10 CAs are not enough because there is no spare capacity for those CAs since the percentage of time in CA_0 is 9.5% (see Figure 12(b)).

Other experiments (see Table III) show that the number of the AMs has a significant impact on the idle time of the CA while the number of the CSPs does not, except when it is below 20.



(a). 20 PDEs in the system

(b). 10 PDEs in the system

Figure 12. Asymptotic Equilibrium of the CA's Utilisation

Table III. Percentage of Time Spent in CA_0 for Large Scale Systems

PDEs	AMs	CAs	CSPs	Percentage time
100	10	20	10	70%
100	10	20	20	70%
100	10	20	30	70%
100	10	20	40	70%
100	20	20	10	61%
100	20	20	20	40%
100	20	20	30	40%
100	20	20	40	40%
100	30	20	10	61%
100	30	20	20	34%
100	30	20	30	34%
100	30	20	40	34%

5. Conclusions and Future Work

In this paper we have presented the high-level modelling formalism PEPA and demonstrated its application to the content adaptation framework being developed under the auspices of the MobileVCE's "Removing the Barriers to Ubiquitous Services" programme. The constructed model was used as the basis for a number of experiments investigating aspects of the design and how well users' expectations might be met, particularly as the system scaled. Of course, content adaptation is just representative of a whole class of meta-services which may be introduced into the network, offering enhanced value for existing services and content. As applications become more sophisticated in this way it is important to ensure that users' non-functional as well as their functional requirements are met. Performance modelling can play an essential role in making those assessments, before too much commitment is made to one particular framework or architecture, i.e., before an implementation is in place.

On-going work is considering additional analysis techniques which may be brought to bear on models constructed in this way, to enhance the experimentation which can be carried out and the insight gained. For example, expected response time can currently only be calculated based on the discrete state space Markovian models, which limits the size of system which can be considered. We are working to extend this to the ODE models used in the scalability analysis.

As for the content adaptation, the framework presented here is based on a centralised (server-side) perspective of the logical architecture, and this has been reflected in the described working cycle and the PEPA model representing it. However, more sophisticated distributed architectures are also under investigation. For example, the adaptation may be carried out at the users' side, using a combination of server and user-side computation, or elsewhere in the network; it may, in some circumstances be possible to carry out some aspects of the adaptation in parallel, rather than sequentially. In all these cases the PEPA model can be readily adapted to reflect the new configuration and similar experiments to those already presented can be conducted.

Acknowledgements

The authors appreciate Mr Allan Clark's help on the experiments. The work reported in this paper has formed part of the Ubiquitous Services Core Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com. This research has been funded by the DTI-led Technology Programme and by the Industrial Companies who are Members of Mobile VCE. Fully detailed technical reports on this research are available to Industrial Members of Mobile VCE. J. Hillston is also supported by EPSRC Advanced Research Fellowship EP/c543696/01 and EU FET-IST Global Computing 2 project SENSORIA ("Software Engineering for Service-Oriented Overlay Computers" (IST-3-016004-IP-09)). J. Ding and D. Laurenson acknowledge the support of the Scottish Funding Council for the Joint Research Institute with the Heriot-Watt University which is a part of the Edinburgh Research Partnership.

References

1. C.A.R.Hoare: 1985, *Communicating Sequential Processes.* Prentice Hall.
2. C.Canali, S.Casolari, and R.Lancellotti: 2005a, 'Architectures for Scalable and Flexible Web Personalization Services'. In: *Proc. of the International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA'05).*
3. C.Canali, V.Cardellini, and M.Colajanni: 2005b, 'Performance Comparison of Distributed Architectures for Content Adaptation and Delivery of Web Resources'. In: *Proc. of the International Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI'05).*
4. C.Canali, V.Cardellini, and M.Colajanni: 2005c, 'A Two-Level Distributed Architecture for Efficient Web Content Adaptation and Delivery'. In: *Proc. of the IEEE/IPSJ Symposium on Applications and the Internet (SAINT'05).*
5. E.Cecchet, A.Chanda, S.Elnikety, J.Marguerite, and W.Zwaenepoel: 2003, 'Performance Comparison of Middleware Architectures for Generating Dynamic Web Content'. In: *Lecture notes in computer science*, Vol. 2672.
6. http://www.mobilevce.com.
7. J.Bush: 2006, 'Architecture for Ubiqiutous Systems'. Deliverable, MobileVCE Core 4.
8. J.Bush, J.Irvine, and J.Dunlop: 2006, 'Removing the Barriers to Ubiquitous Services: A User Perspective'. In: *First International Workshop on Personalized Networks(PerNets'06).*
9. J.Dunlop, R. C. Atkinson, J. Irvine, and D. Pearce: 2003, 'A Personal Distributed Environment for Future Mobile Systems'. In: *IST Mobile and Wireless Communications Summit.*
10. J.Hillston: 1996, *A Compositional Approach to Performance Modelling(PhD Thesis).* Cambridge University Press.
11. J.Hillston: 2005, 'Fluid Flow Approximation of PEPA Models'. In: *International Conference on the Quantitative Evaluation of Systems (QEST'05).*
12. N.Li and K.Moessner: 2006, 'the MVCE Management Framework for Context-Aware Content and Service Adaptation'. In: *1st International Workshop Semantic Media Adaptation and Personalization.*
13. R.Milner: 1989, *Communication and Concurrency.* Prentice Hall.
14. W.Y.Lum and F. C. M. Lau: 2002, 'A Context-Aware Decision Engine for Content Adaptation'. *IEEE Pervasive Computing* **1**(3), 41–49.

Appendix A

The operational semantics of PEPA are presented as follows.

Prefix:

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Choice:

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E'}, \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} F'}$$

Cooperation:

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F}(\alpha \notin L), \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{L}{\bowtie} F'}(\alpha \notin L)$$

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \quad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} (\alpha \in L), \quad \text{where} \quad R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F)),$$

where $r_\alpha(E)$, the apparent rate of action of type α in the component E, i.e. the sum of the rates of all activities of type α in $\mathcal{A}ct(E)$. Similarly, for $r_\alpha(F)$.

Hiding:

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L} (\alpha \notin L), \qquad \frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L} (\alpha \in L)$$

Constant:

$$\frac{E \xrightarrow{(\alpha,r)} E'}{A \xrightarrow{(\alpha,r)} E'} (A \overset{\mathit{def}}{=} E)$$

Appendix B

Here, for completeness, we present the remaining components of the PEPA model presented in Section 3.2. These components are the Content Adaptor (CA) and the Content/Service Provider (CSP) and their behavior was shown diagrammatically in Figure 4.

$$\begin{aligned}
CA_0 &\overset{\mathit{def}}{=} (\mathit{am\_cs\_source\_para}, r_{\mathit{am\_cs\_source\_para}}).CA_1 \\
CA_1 &\overset{\mathit{def}}{=} (\mathit{ca\_adaptation}, r_{\mathit{ca\_adaptation}}).CA_2 \\
CA_2 &\overset{\mathit{def}}{=} (\mathit{ca\_adapted\_cs}, r_{\mathit{ca\_adapted\_cs}}).CA_0
\end{aligned}$$

$$\begin{aligned}
CSP_0 &\overset{\mathit{def}}{=} (\mathit{csp\_context\_req}, r_{\mathit{csp\_context\_req}}).CSP_1 \\
CSP_1 &\overset{\mathit{def}}{=} (\mathit{csp\_context\_res}, r_{\mathit{csp\_context\_res}}).CSP_2 \\
CSP_2 &\overset{\mathit{def}}{=} (\mathit{am\_adapted\_cs\_req}, r_{\mathit{am\_adapted\_cs\_req}}).CSP_3 \\
CSP_3 &\overset{\mathit{def}}{=} (\mathit{csp\_decision}, r_{\mathit{csp\_decision}}).CSP_4 \\
CSP_4 &\overset{\mathit{def}}{=} (\mathit{csp\_no\_cs}, r_{\mathit{csp}_{\mathit{no\_cs}}}).CSP_0 + (\mathit{csp\_cs\_source}, r_{\mathit{csp\_cs\_source}}).CSP_0
\end{aligned}$$