Gerência de Infra-estrutura e Comunicação na Arquitetura para Aplicações Ubíquas Continuum

Rodolfo S. Antunes[1], Cristiano A. Costa[1], Jorge L. V. Barbosa[1], Cláudio F. R. Geyer[2], and Adenauer C. Yamin[3]

[1] UNISINOS – Universidade do Vale do Rio dos Sinos
São Leopoldo, RS, Brasil
[2] UFRGS – Unifersidade Federal do Rio Grande do Sul
Porto Alegre, RS, Brasil
[3] UCPel – Universidade Católica de Pelotas
Pelotas, RS, Brasil
rodolfo.s.antunes@gmail.com, cac@unisinos.br, jbarbosa@unisinos.br
geyer@inf.ufrgs.br, adenauer@ucpel.tche.br

Abstract Advances in ubiquitous computing foster the development of new and more complex applications. The implementation of these applications benefits from the use of software architectures to abstract the various aspects associated with management of the physical environment. In this perspective, Continuum is a software infrastructure, integrating a framework and a middleware, designed to deal with the requirements of ubiquitous computing. In this paper we focus on describing the design of two Continuum services: *cell information base* and *communicator*. We propose a model for the *cell information base* service that uses a hierarchical management, providing search mechanisms restricted by selection and expected results. In the model proposed for the *communicator* service, two filters are used to constrain the dissemination of notifications in an event-oriented communication mechanism. At the end, we present some results obtained throughout the prototypes implemented for the two services.

1 Introdução

O termo computação ubíqua foi utilizado pela primeira vez em [1]. O autor afirma que é necessária uma maior integração da tecnologia da informação com o cotidiano, de forma que ela se torne transparente, auxiliando os usuários a lidar com a sobrecarga diária de informações. Pesquisas na computação ubíqua englobam, além de aspectos próprios, questões das áreas de sistemas distribuídos e computação móvel [2].

Um estudo sobre os desafios que caracterizam a computação ubíqua é apresentado em [3], sendo utilizado para definir um modelo abrangente de infra-estrutura de software para a computação ubíqua. O modelo tem como elementos principais um *framework*, para auxiliar no projeto de aplicações para a arquitetura, e um *middleware*, responsável por tratar diversas questões da execução

destas aplicações. Este modelo é a base da definição do Continuum [4], uma infra-estrutura de software para aplicações ubíquas, baseada em uma arquitetura orientada a serviços (SOA). Uma SOA é composta por um conjunto de serviços base, que podem ser utilizados para a criação de novos serviços e aplicações através de composição e gerenciamento [5]. O *middleware* do Continuum, neste caso é composto por uma série de serviços plugáveis, responsáveis pelas principais funcionalidades do ambiente de execução.

Alguns serviços que compõe o *middleware* do Continuum possuem implementação funcional [6]. Há, porém, serviços básicos do *middleware*, apresentados em [4], que possuem descrito apenas o modelo de sua interface. Dentre estes serviços, encontram-se: o *cell information base*, utilizado pelo Continuum para a manutenção do ambiente distribuído formado pelas entidades que o executam; e o *communicator*, que provê um sistema simples de comunicação orientada a eventos, seguindo os princípios do modelo publicar-assinar (*publish-subscribe*).

Este artigo apresenta o modelo desenvolvido com o objetivo de guiar a implementação destes serviços, além de resultados obtidos a partir da avaliação de seus protótipos. A seção 2 apresenta conceitos utilizados na modelagem dos serviços. Alguns trabalhos similares são apresentados na seção 3. Os modelos dos serviços *cell information base* e *communicator* são apresentados, respectivamente, nas seções 4 e 5. Detalhes sobre o protótipo são apresentados na seção 6, além de resultados obtidos. Finalmente, a seção 7 apresenta considerações finais.

2 Manutenção de Infra-estrutura e Comunicação no Continuum

Um modelo de abstração é proposto em [4], que permite a representação, no contexto de aplicações ubíquas, de entidades relevantes do mundo real. Também é proposta uma notação baseada na linguagem UML, que pode ser utilizada para diagramar ambientes representados através do modelo. A Figura 1 é um exemplo de um ambiente representado através desta notação.

No modelo, uma *CoDimension* engloba todas as entidades que podem estar contidas no ambiente de uma aplicação. Locais presentes no ambiente são representados através de células, denominadas *CoCells*. Além de estar contida na CoDimension, uma célula pode estar contida em outra célula, permitindo a representação de composições de locais. O nível de abstração representado por uma célula varia com a aplicação desenvolvida.

CoPersons representam pessoas presentes no ambiente, que estão localizadas em uma célula, e podem se deslocar para outras. Objetos representados pelo modelo, denominados *CoNodes*, englobam dispositivos eletrônicos presentes nas células, como computadores, sensores, ou dispositivos móveis. CoNodes padrão representam dispositivos estáticos, tais como computadores de mesa. Além destes, existem tipos particulares de representações empregadas na infra-estrutura. Cada célula possui um nodo especial, denominado *CoBase*, que executa serviços básicos para seu gerenciamento. Dispositivos com características móveis, como

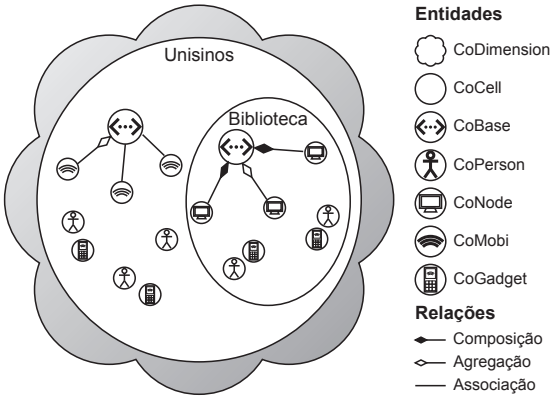


Figura 1. Ambiente modelado através das abstrações do Continuum

notebooks com conexão por rede sem fio, são denominados *CoMobis*. Dispositivos móveis com propósitos específicos, como celulares ou PDAs, são tratados de maneira diferenciada pelo ambiente e são denominados *CoGadgets*.

A manutenção da infra-estrutura modelada, além da consulta aos atributos das entidades gerenciadas, é a função do *cell information base* (CIB). O *communicator*, por sua vez, implementa um serviço de comunicação por eventos que pode ser utilizado pelas aplicações do ambiente.

O modelo publicar-assinar permite que um objeto seja notificado sobre eventos ocorridos em outros, podendo executar operações como resultado da mudança no estado de um objeto [7]. Neste modelo, um provedor publica os eventos disponíveis para observação, enquanto clientes criam assinaturas para o recebimento dos eventos. A comunicação orientada a eventos permite que provedores e assinantes sejam desacoplados nas dimensões de espaço, tempo e sincronização. A anonimidade e assincronismo inerentes deste modelo o tornam ideal para sistemas distribuídos [8]. O objetivo do serviço *communicator* é oferecer um sistema de comunicação com tais características para o ambiente do Continuum.

Sistemas publicar-assinar podem utilizar três metodologias para o gerenciamento de assinaturas [9]. Em um sistema baseado em tipos, as assinaturas são especificadas a partir dos tipos de dados contidos nas notificações. Um sistema de tópicos define assinaturas a partir de tópicos pré-definidos no sistema. Um sistema baseado em conteúdo, por fim, define assinaturas com base nos atributos presentes nas notificações, através de uma linguagem de pesquisa.

3 Trabalhos Relacionados

No âmbito de gerência de infra-estrutura, o *contextual information services* (CIS) [10] é um serviço que provê informações de contexto sobre entidades e recursos em ambientes ubíquos. O principal componente do CIS é o *query synthesizer*, que é sua interface para a execução de consultas. Ele divide uma consulta complexa

em sub-consultas, que são enviadas para as instâncias do serviço que mantêm as fontes de informação. A interface das fontes de informação é a mesma utilizada pelo *query synthesizer*, sendo composta por um método de consulta com uma semântica similar à linguagem SQL. O CIS pode gerenciar vários tipos de informações de contexto, mas não considera a organização dos espaços físicos do ambiente. O CIB, por sua vez, armazena as relações entre os espaços físicos do ambiente, utilizando-as como meio para propagação de suas pesquisas.

A infra-estrutura para computação ubíqua Gaia [11] possui, em seu *middleware*, dois componentes para o gerenciamento de entidades. O *presence service* detecta e mantém informações das entidades físicas e digitais no ambiente, através de um mecanismo de *leasing*, que permite a eliminação de entidades que não renovam seu cadastro. Entidades digitais periodicamente entram em contato com o serviço para renovar seu *lease*, enquanto entidades físicas são observadas pró-ativamente, por meio de sensores no ambiente. O *space repository*, por sua vez, armazena as informações coletadas pelo *presence service*. O meio de armazenamento utilizado pode ser composto de diferentes técnicas, como um banco de dados ou um espaço de tuplas. As consultas ao serviço utilizam a linguagem SQL. Estes serviços levam em consideração dispositivos e aplicações em execução, mas não possuem uma gerência específica dos espaços físicos e suas relações.

No âmbito de comunicação por eventos, o Hermes [12] é um sistema publicar-assinar baseado em um *middleware* formado por uma rede de *overlay*, que interliga os nodos que executam os mecanismos para a comunicação entre os clientes. A rede de *overlay* permite que o Hermes se adapte a mudanças na topologia do *middleware*, além de tornar eficiente o roteamento de notificações. O Hermes utiliza assinaturas baseadas em tipos. Cada tipo de dado dos eventos possui um nodo específico do *overlay* para a gerência de assinaturas, que é encontrado através de uma função *hash* que usa o tipo de dado como argumento. O mecanismo de tipos empregado pelo Hermes traz diversas vantagens para o *middleware*, mas o torna dependente de uma biblioteca de tipos, o que limita sua heterogeneidade.

O Echo [13] é um *middleware* publicar-assinar para comunicação de alto desempenho, que utiliza canais de eventos para a propagação de mensagens. Estes canais utilizam um mecanismo de assinatura baseado em tópicos, ou tipos, e são entidades descentralizadas, mantidas pelos próprios clientes do serviço, que propagam notificações diretamente entre si. As mensagens do Echo utilizam uma biblioteca de tipos própria, que permite a descrição de tipos similares à uma linguagem estruturada. O *middleware* empregado pelo Echo requer o poder computacional dos clientes para sua execução, trazendo problemas para dispositivos limitados. A conectividade intermitente presente em ambientes ubíquos também pode influenciar negativamente no *middleware*.

4 Modelo Proposto de Gerência de Infra-estrutura

Cada CoBase possui uma instância do *cell information base*, que gerencia as informações das células a ele associadas, além dos dispositivos e usuários nelas presentes. Quando há várias instâncias do CIB no ambiente, as informações

coletadas estarão distribuídas entre elas. As relações entre as células formam uma estrutura hierárquica, que tem como raiz a CoDimension. Uma célula c possui uma célula pai c_p, que está um nível acima de c na hierarquia, e um conjunto de células filhas $c_{f_1}, c_{f_2}, ..., c_{f_i}$, que estão um nível abaixo de c.

A hierarquia de células é mantida em uma estrutura de árvore. Quando há uma relação entre duas células gerenciadas por instâncias diferentes do CIB, é armazenado o endereço para acesso entre as instâncias, permitindo sua comunicação. A Figura 2 ilustra uma configuração de diversas instâncias do CIB, e as células por elas controladas. Uma relação interna é mantida por uma instância do CIB. Uma relação externa, por sua vez, é mantida por duas instâncias.

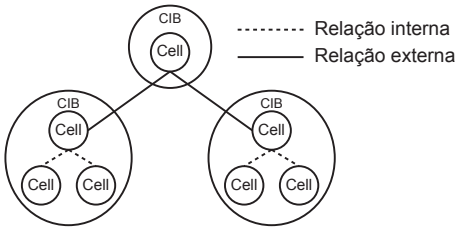


Figura 2. Ilustração das relações entre instâncias do CIB

Cada célula possui um conjunto básico de atributos: um identificador único; uma referência para a célula pai; e uma lista de referências para as células filhas. Estes atributos são atualizados de acordo com mudanças na hierarquia. Outros atributos podem ser associados à célula, de acordo com a necessidade das aplicações que utilizam o serviço. O CIB também armazena informações de usuários e dispositivos presentes no ambiente, coletados diretamente destas entidades. Usuários e dispositivos têm como atributos básicos um identificador único, e o identificador da célula com a qual a entidade está associada, obtido através do CoBase. Atributos adicionais também podem ser incluídos para estas entidades. Todas entidades possuem uma marcação de tempo, atualizada a cada acesso ao serviço. Esta informação é utilizada para tornar inativas entidades que não entraram em contato com o serviço por um determinado período de tempo.

As consultas do CIB têm como base a hierarquia mantida pelo serviço. Cada consulta tem como parâmetro uma célula-alvo, a partir da qual ocorre a propagação para todas as células filhas no ambiente. A propagação se encerra quando o nível mais baixo da hierarquia ou o parâmetro de profundidade máxima da pesquisa são atingidos. Caso uma consulta deva ser propagada para outra instância do serviço, é utilizado o endereço de acesso cadastrado na estrutura. A segunda instância do CIB, neste caso, recebe uma consulta com parâmetros atualizados, que após executada é retornada para a instância original. Cada pesquisa possui outros dois conjuntos de atributos, denominados **atributos de seleção** e **atributos de retorno**. Os atributos de seleção definem um conjunto de filtros que serão aplicados durante a pesquisa. Os atributos de retorno, por sua vez,

especificam quais informações, dentre aquelas cadastradas no serviço, devem ser retornadas. Tomando como exemplo a linguagem SQL, os atributos de seleção são equivalentes à cláusula `where`, e os atributos de retorno à cláusula `select`.

5 Modelo Proposto de Comunicação por Eventos

Cada CoBase presente no ambiente possui uma instância do *communicator* em execução, permitindo que os dispositivos presentes em uma célula possam se comunicar através da publicação e assinatura de eventos. Dispositivos de diferentes células também podem se comunicar através do serviço. Neste caso, os eventos são roteados entre as instâncias do serviço utilizadas por estes dispositivos.

A Figura 3 ilustra as relações entre dispositivos que utilizam o *communicator*. Linhas tracejadas representam relações de comunicação estabelecidas entre um provedor e um cliente. Tais relações são dinâmicas, estabelecidas quando eventos de um provedor se enquadram na assinatura de um cliente. Linhas contínuas apresentam os canais de comunicação de fato estabelecidos pelo serviço. O fluxo de eventos em uma célula é controlado pela instância do *communicator* presente no CoBase. Quando dispositivos de diferentes células se comunicam, o fluxo de mensagens é roteado através das instâncias do *communicator* dessas células.

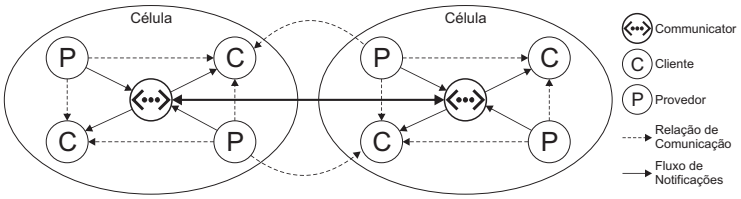


Figura 3. Relações de comunicação entre dispositivos que utilizam o *communicator*

Um dispositivo que irá publicar eventos no *communicator* deve criar um registro no serviço. Este registro é composto por um conjunto de atributos que descreve propriedades gerais dos eventos que serão publicados. Um dispositivo que deseja receber notificações, por sua vez, deve criar uma assinatura, composta por dois filtros, cada um formado por um conjunto de atributos e valores. O primeiro conjunto, denominado **filtro de assinatura**, é comparado com os atributos em cada registro de provedor cadastrado. Nos casos em que os atributos do registro estão de acordo com o filtro de assinatura, é criada uma associação entre esta assinatura e o registro do provedor. Os eventos publicados por um provedor, então, são apenas processados em relação às assinaturas associadas ao seu registro. O segundo filtro, denominado **filtro de eventos** é utilizados na seleção de quais notificações, dentre as recebidas, serão entregues ao cliente.

A Figura 4 ilustra a atuação dos filtros de assinatura e de eventos. Na ilustração, os provedores P_1 e P_3 foram selecionados pelo filtro de assinatura, e

portanto apenas suas notificações são consideradas. As notificações geradas por P_1 e P_3 são então submetidas ao filtro de eventos, que seleciona, com base nos atributos das notificações, aquelas que serão entregues ao cliente. Na ilustração, o filtro de eventos selecionou N_1P_1, N_2P_1, e N_1P_3. As notificações selecionadas são, por fim, armazenadas pelo serviço até que elas sejam entregues ao cliente. Após recuperada por todos os clientes, estas notificações são descartadas.

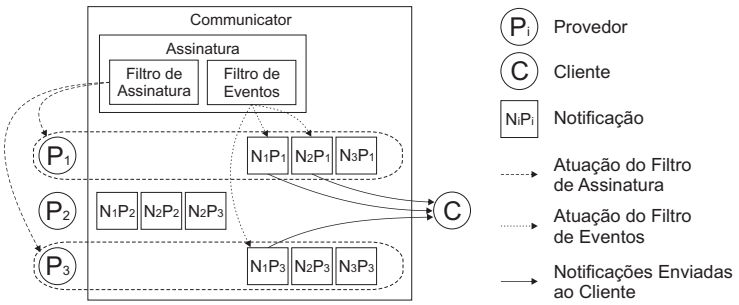


Figura 4. Atuação dos filtros de assinatura e de eventos

Cada novo registro de provedor criado em uma instância do *communicator* é propagado para todas outras instâncias, de forma que os provedores de eventos sejam conhecidos em todo o ambiente. Assinaturas podem ser, então, associadas com provedores de outras instâncias do serviço. Quando tal associação ocorre, a instância onde a assinatura foi criada submete uma cópia desta para a instância onde o provedor se cadastrou. A instância do provedor, neste caso, realiza o processamento do filtro de eventos para cada notificação relacionada com a assinatura, e envia para a instância do cliente cada notificação válida para a assinatura. A instância do cliente, por fim, armazena as notificações para que sejam posteriormente entregues.

6 Implementação e Resultados

A partir dos modelos apresentados, um protótipo dos serviços descritos foi implementado, com o objetivo de permitir uma análise inicial das principais propriedades do modelo proposto, através da execução de avaliações de desempenho nos componentes do serviço. A versão inicial do protótipo foi limitada à execução em apenas um CoBase do ambiente, permitindo a avaliação dos limites impostos pela utilização do serviço de forma centralizada, e também do desempenho de uma instância do serviço em configurações diversas do ambiente.

O protótipo foi implementado na linguagem Java. Os métodos presentes nas interfaces dos serviços utilizam documentos XML para a descrição das entidades gerenciadas. A linguagem de expressões XPath é utilizada para a especificação

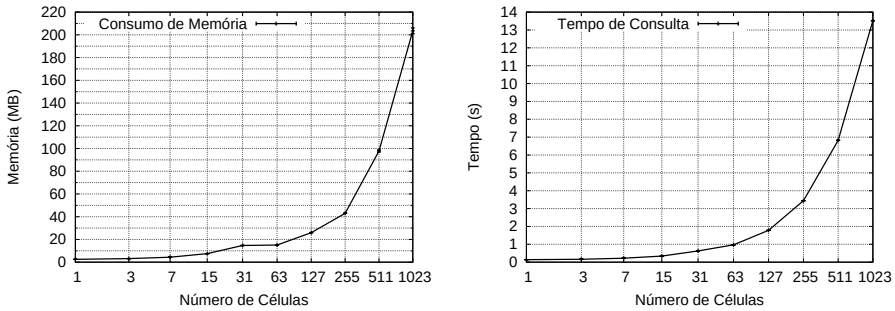
dos filtros utilizados pelos mecanismo de pesquisa do CIB, e também pelo mecanismo de assinaturas do *communicator*. O armazenamento de dados em ambos os serviços foi realizado através de um banco de dados relacional.

Os principais resultados obtidos durante as avaliações do protótipo serão apresentadas a seguir. Os resultados são baseados em experimentos realizados em um computador com um processador Intel Core 2 Quad de 2.4 GHz, 4 GB de memória, executando o sistema operacional Linux. Foram executadas 30 repetições dos experimentos descritos. O intervalo de confiança, que leva em consideração um grau de confiança de 99%, é apresentado nos gráficos, apenas nos pontos em que é observado um intervalo relevante, na forma de barras de erro.

6.1 Avaliação do Serviço *Cell Information Base*

A principal estrutura mantida pelo serviço durante sua execução é a hierarquia de células. A avaliação desenvolvida, neste caso, teve como foco o desempenho do serviço durante a execução de pesquisas em diferentes configurações da hierarquia, contendo de 1 a 1023 células. Cada uma das células de mais baixo nível da hierarquia teve cadastrado um dispositivo, utilizado como alvo das pesquisas realizadas durante a avaliação. Todas as pesquisas utilizadas tiveram como célula-alvo a célula raiz, e a profundidade da pesquisa, bem como o número de resultados retornados, não foram limitados. O método de pesquisa, neste caso, percorre todas as células da hierarquia em sua execução.

As principais métricas avaliadas durante os experimentos foram o consumo de memória da hierarquia e o tempo de execução da pesquisa. A Figura 5 apresenta os resultados obtidos nos experimentos descritos. O eixo horizontal representa o número de células presentes na hierarquia durante o experimento. O eixo vertical da Figura 5(a) representa a memória consumida, em Megabytes, pela máquina virtual do Java. O eixo vertical da figura 5(b), por sua vez, representa o intervalo de tempo necessário para a realização da pesquisa.



(a) Consumo de memória (b) Tempo de consulta

Figura 5. Resultados dos experimentos com o CIB

Os resultados obtidos mostram que ambas as métricas têm um crescimento moderado para hierarquias constituídas por até 63 nodos. A partir deste ponto, tanto o consumo de memória quanto o tempo necessário para a execução de uma pesquisa elevam-se mais rapidamente, sendo necessários 200 Megabytes de memória para o gerenciamento de uma hierarquia de 1023 células, e pouco mais de 13 segundos para a execução de uma pesquisa nesta mesma hierarquia. Tais resultados sugerem que cada instância do serviço não apresenta problemas na gerência de hierarquias menores que 63 células. A partir deste ponto, porém, a sobrecarga gerada tanto no consumo de memória quanto no tempo gasto para a execução de uma pesquisa tornam-se consideráveis. Tal problema pode ser mitigado pelo uso da proposta distribuída apresentada na seção 4, onde mais de uma instância do serviço está presente no ambiente.

6.2 Avaliação do Serviço *Communicator*

A avaliação deste serviço foi focada no desempenho do processamento de notificações recebidas de provedores. O experimento avaliou o tempo necessário para processar uma mensagem de notificação recebida, considerando-se diferentes números de assinantes que devem recebê-la. A Figura 6 apresenta os resultados obtidos nesta avaliação. O eixo horizontal apresenta o número de assinantes utilizados durante o experimento, e o eixo vertical, o tempo necessário para o processamento da mensagem de notificação.

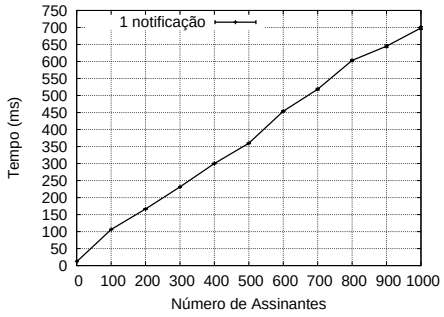


Figura 6. Tempo de processamento de uma notificação no *communicator*

O resultado mostra que o tempo necessário para o processamento uma notificação terá um crescimento linear, com algumas variações, de acordo com o número de assinantes que estão associados ao provedor. Este crescimento mostrou-se em média, próximo aos 70 ms para cada 100 acrescentados ao experimento, representando uma média de 7 ms necessários para verificar se uma mensagem deve ser entregue a um assinante. Este resultado indica que haverá uma maior sobrecarga do serviço conforme o número de clientes aumenta. A distribuição do

serviço, apresentada na seção 5, permite que esta sobrecarga, gerada pelo grande número de clientes, seja distribuída entre várias instâncias do serviço.

7 Considerações Finais

Este artigo apresentou os modelos desenvolvidos para os serviços *cell information base* e *communicator* do *middleware* da infra-estrutura de software para a computação ubíqua Continuum. Estes modelos foram utilizados no desenvolvimento de protótipos destes serviços, utilizados na avaliação do modelo.

Dando continuidade ao trabalho, o protótipo desenvolvido será estendido, para incluir as funcionalidades distribuídas propostas no modelo, permitindo a avaliação final da proposta apresentada. Esta nova versão dos serviços, então, será integrada ao *middleware* do Continuum, para que suas funcionalidades sejam utilizadas nas aplicações do ambiente.

Referências

1. Weiser, M.: The computer for the 21st century. Scientific American **265**(3) (Mar. 1991) 94–104
2. Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE Personal Communications **8**(4) (Ago. 2001) 10–17
3. Costa, C., Yamin, A., Geyer, C.: Towards a general software infrastructure for ubiquitous computing. IEEE Pervasive Computing **7**(1) (Jan. 2008) 64–73
4. Costa, C.: Continuum: A Context-aware Service-based Software Infrastructure for Ubiquitous Computing. PhD thesis, UFRGS (2008)
5. Papazoglou, M., Georgakopoulos, D.: Introduction: Service-oriented computing. Communications of the ACM **46**(10) (Out. 2003) 24–28
6. Costa, C., Kellermann, F., Antunes, R., Cavalheiro, L., Yamin, A., Geyer, C.: Continuum: A service-based software infrastructure for ubiquitous computing. In: 7th International Conference on Pervasive Computing and Communications, Washington, IEEE (2009)
7. Coulouris, G., Dollimore, J., Kindberg, T.: Distributed Systems: Concepts and Design. Addison-Wesley, Harlow (2001)
8. Huang, Y., Garcia-Molina, H.: Publish/subscribe in a mobile environment. Wireless Networks **10**(6) (Nov. 2004) 643–652
9. Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A.: The many faces of publish/subscribe. ACM Computing Surveys **35**(2) (Jun. 2003) 114–131
10. Judd, G., Steenkiste, P.: Providing contextual information to pervasive computing applications. In: 1st International Conference on Pervasive Computing and Communications, 2003, Fort Worth, Washington, IEEE (2003) 133–142
11. Roman, M., Hess, C., Cerqueira, R., Rananathan, A., Campbell, R., Nahrstedt, K.: A middleware infrastructure for active spaces. IEEE Pervasive Computing **1**(4) (Out. 2002) 74–83
12. Pietzuch, P., Bacon, J.: Hermes: A distributed event-based middleware architecture. In: 22nd International Conference on Distributed Computing Systems Workshops, 2002, Vienna, Washington, IEEE (2002) 611–618
13. Eisenhauer, G., Schwan, K., Bustamante, F.: Publish-subscribe for high-performance computing. IEEE Internet Computing **10**(1) (Jan. 2006) 40–47