Um Modelo de Coordenação Escalável e Proativo para Aplicações Ubíquas

Rodrigo Santos de Souza[1], João Ladislau Lopes[1], Gizele I. Gadotti[2], Marcia Z. Gusmão[2], Adenauer Yamin[2], Cláudio Geyer[1]

[1] Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS

[2]Universidade Federal de Pelotas (UFPel)
Pelotas – RS

{rssouza, geyer, jlblopes}@inf.ufrgs.br, mzgusmao@inf.ufpel.edu.br,

{adenauer,gizele.gadotti}@ufpel.edu.br

Abstract. *In Ubiquitous Computing the systems must provide mechanisms to promote cooperation between the users and between the applications. With this motivation, this paper presents the EXEHDA-TS, which is a distributed mechanism for coordination of applications in Ubiquitous Computing. The proposed software architecture has proactive behavior and allows a dynamic control of communication costs, promoting scalability. The EXEHDA-TS was modeled as an integrated service to the middleware EXEHDA. The prototype was evaluated through case studies in the medical field, considering the challenges inherent in the PERTMED design, satisfactorily fulfilling the requirements of the focused scenarios.*

Resumo. *Na Computação Ubíqua os sistemas devem oferecer mecanismos para promover a cooperação entre os usuários e entre as aplicações. Com esta motivação, este trabalho apresenta o EXEHDA-TS, que é um mecanismo distribuído para coordenação de aplicações na Computação Ubíqua. A arquitetura de software para o mecanismo proposto tem como objetivos um comportamento proativo e um controle dinâmico dos custos de comunicação, promovendo assim sua escalabilidade. O EXEHDA-TS foi modelado como um serviço integrado ao middleware EXEHDA. O protótipo foi avaliado através de estudos de caso na área médica, considerando os desafios inerentes ao projeto PERTMED, cumprindo de forma satisfatória os requisitos dos cenários focados.*

1. Introdução

Na perspectiva da Computação Ubíqua (UbiComp), introduzida por Mark Weiser em [Weiser 1991], o foco das aplicações está no usuário e em suas atividades, devendo prover o acesso às suas informações e ambiente computacional a partir de qualquer dispositivo e a qualquer momento [da Costa et al. 2008, Want et al. 2010]. Nesta perspectiva, os sistemas computacionais na UbiComp devem facilitar o compartilhamento de informações entre os componentes das aplicações distribuídas, promovendo atividades colaborativas entre os usuários e entre seus aplicativos [Grimm et al. 2004, Augustin et al. 2008].

Com o intuito de prover o suporte ao desenvolvimento de aplicações que atendam essas demandas introduzidas pela UbiComp, foi concebido o EXEHDA-TS[1], o qual consiste em um modelo de coordenação pro-ativo, escalável e com comportamento dinâmico, que gerencia um espaço de tuplas compartilhado cujos dados armazenados podem estar distribuídos entre os diversos nodos que constituem a infraestrutura computacional [Souza 2009].

O EXEHDA-TS foi concebido como um serviço a ser integrado à arquitetura de software do *middleware* EXEHDA [Lopes et al. 2007]. Enquanto parte das camadas básicas do *middleware* serve de suporte a outros serviços disponíveis, os quais, em conjunto, constituem uma infraestrutura de desenvolvimento e execução de aplicações ubíquas de propósito geral (ver figura 2).

Na literatura, foram identificados outros mecanismos de coordenação com objetivos semelhantes [Cabri et al. 2006]. Dentre aqueles com suporte às demandas da UbiComp, como por exemplo a mobilidade, a maioria foi concebido com foco em redes *ad hoc*, sendo que alguns ainda limitam o escopo da comunicação aos dispositivos locais. Esta situação constituiu uma das motivações centrais para o desenvolvimento do EXEHDA-TS.

Este artigo está organizado em seis seções. A próxima seção aponta os trabalhos relacionados e apresenta as suas principais características. A seção três contempla uma visão geral a respeito do *middleware* EXEHDA, destacando seus aspectos mais significativos. A quarta seção apresenta a modelagem do EXEHDA-TS e os fundamentos conceituais envolvidos. Na quinta seção é sistematizado o estudo de caso realizado com foco nas demandas da medicina ubíqua. Por fim, na sexta seção, são apresentadas as considerações finais do trabalho desenvolvido.

2. Trabalhos Relacionados

Os projetos apresentados nessa seção foram selecionados por possuírem características próximas à proposta do EXEHDA-TS. Todos implementam modelos de coordenação com base em espaço de tuplas distribuído, com foco em mobilidade e com preocupações em relação à escalabilidade da infraestrutura computacional, aspectos centrais no perfil das aplicações direcionadas à UbiComp.

O Lime [Murphy et al. 2006] é um *middleware* que oferece uma camada de coordenação direcionada às demandas das aplicações voltadas aos ambientes móveis *ad hoc*. Para auxiliar no desenvolvimento das aplicações nesses cenários, o *middleware* Lime propõe um modelo de espaço de tuplas distribuído entre os componentes móveis, podendo ser compartilhado segundo critérios de conectividade. No Lime, cada processo tem seu próprio espaço de tuplas que o acompanha durante as migrações, proporcionando um tratamento para a mobilidade.

EgoSpaces [Julien and Roman 2006] é um middleware baseado em espaço de tuplas semelhante ao Lime, que explora uma arquitetura totalmente distribuída. O compartilhamento dos espaços de tuplas, associados a cada componente de software, se dá de acordo com limites especificados, como IDs dos componentes de softwares, número máximo de saltos entre os nodos e padrões de tupla. Uma característica marcante no

[1]Trabalho integrado ao Projeto Pertmed (*Pervasive Telmedicine*), financiado pelo Finep/MCT

EgoSpaces é o suporte a transações, que permite a execução atômica de uma sequência de operações.

O TOTA [Mamei and Zambonelli 2009] é um *middleware* voltado às aplicações distribuídas. Assim como no Lime, o TOTA também tem como foco redes *ad hoc* bem como seu modelo baseado em espaço de tuplas distribuído. Diferentemente do Lime, em que as operações distribuídas são as de leitura, no TOTA as consultas são realizadas no espaço de tuplas local enquanto as inserções são feitas de forma distribuída de acordo com as regras de propagação associadas a cada tupla.

A reatividade está presente tanto no TOTA quanto no Lime. As aplicações podem realizar subscrições caracterizando o motivo do evento que deve ser gerado. No Lime os eventos são associados às modificações ocorridas no espaço de tuplas, enquanto que no TOTA, também podem ser associados às mudanças na topologia da rede local.

O modelo PeerSpace [Valente et al. 2003], assim como os anteriores, tem foco em redes *ad hoc*. No seu modelo de coordenação, cada nodo possui o seu próprio espaço de tuplas, que pode ser compartilhado por aplicações que estejam em execução no próprio nodo ou em nodos remotos, porém, diferentemente do Lime e do TOTA, o modelo PeerSpace não prevê transparência no acesso aos espaços de tuplas remotos. Além disso, a reatividade é tratada de forma distinta dos outros modelos discutidos. O PeerSpace implementa uma operação reativa chamada de consulta contínua, que trata-se de uma busca distribuída em que a tupla pretendida, quando disponível, é retornada através de um evento.

O desacoplamento de localização e tempo promovido pela abordagem de espaço de tuplas, comum a todos os *middlewares* discutidos, associado ao emprego de eventos que reajam às modificações nos dados armazenados, constituem uma solução de coordenação considerada adequada aos ambientes móveis e dinâmicos, como no caso dos sistemas ubíquos [Cabri et al. 2006]. Esta estratégia é comum a todos os *middlewares* discutidos nesta seção e tem como uma das vantagens o fato de liberar as aplicações do constante monitoramento do espaço de tuplas.

3. Contexto de Pesquisa

O EXEHDA [Lopes et al. 2007] é *middleware* voltado à gerência da execução das aplicações da UbiComp. Foi criado de modo a permitir que as aplicações móveis e distribuídas possam obter informações do ambiente no qual executam, e assim sejam capazes de reagir de maneira adaptativa, possibilitando alteração no seu comportamento conforme ocorrem modificações no ambiente. Na perspectiva do EXEHDA, o *middleware* deve possibilitar que as aplicações dos usuários estejam disponíveis em qualquer lugar, todo o tempo e a partir de qualquer dispositivo.

Para perseguir essa premissa, o EXEHDA provê suporte à mobilidade lógica através de operações sobre a abstração OX (Objeto eXehda), unidade computacional base das aplicações, os quais podem migrar entre quaisquer dispositivos do ambiente ubíquo. Esta possibilidade traz consigo diversos desafios em relação à coordenação, pois a execução de um OX não está associada a um dispositivo físico específico e por consequência não conta com referências usuais, como por exemplo, uma associação fixa a um determinado endereço de rede (IP e/ou *hostname*) [Lopes et al. 2007]. O ambiente ubíquo, gerenciado pelo *middleware* EXEHDA, é constituído por um conjunto de

células (EXEHDAcél) formando uma infraestrutura semelhante a uma grade computacional (ver figura 1).

4. EXEHDA-TS: Concepção e Modelagem

O EXEHDA-TS foi concebido com o objetivo de dotar o *middleware* EXEHDA de suporte para suprir as demandas de coordenação da UbiComp. Em sua modelagem foi explorada a perspectiva de uso de espaço de tuplas com atuação proativa e foram adotadas estratégias que possibilitam o controle dos aspectos de escalabilidade inerentes a UbiComp [Souza 2009].

4.1. Acesso Ubíquo

Uma das principais premissas da UbiComp trata do acesso ao ambiente do usuário independentemente de localização e tempo, o que inclui todos os seus dados [Want et al. 2010]. Para suprir essa demanda, na concepção do EXEHDA-TS foi adotado um modelo em que o armazenamento das tuplas é distribuído fisicamente entre os nodos do ambiente ubíquo gerenciado pelo *middleware*. No modelo proposto, cada OX em execução pode ter o seu próprio espaço de tuplas que é chamado **TSox**. Os diversos TSox podem ser agrupados formando um espaço de tuplas virtual chamado **TSvirt**. O TSvirt consiste de um conjunto de referências à TSox distribuídos pelos nodos do sistema, servindo de base para as operações realizadas sobre o espaço de tuplas (ver figura 1).

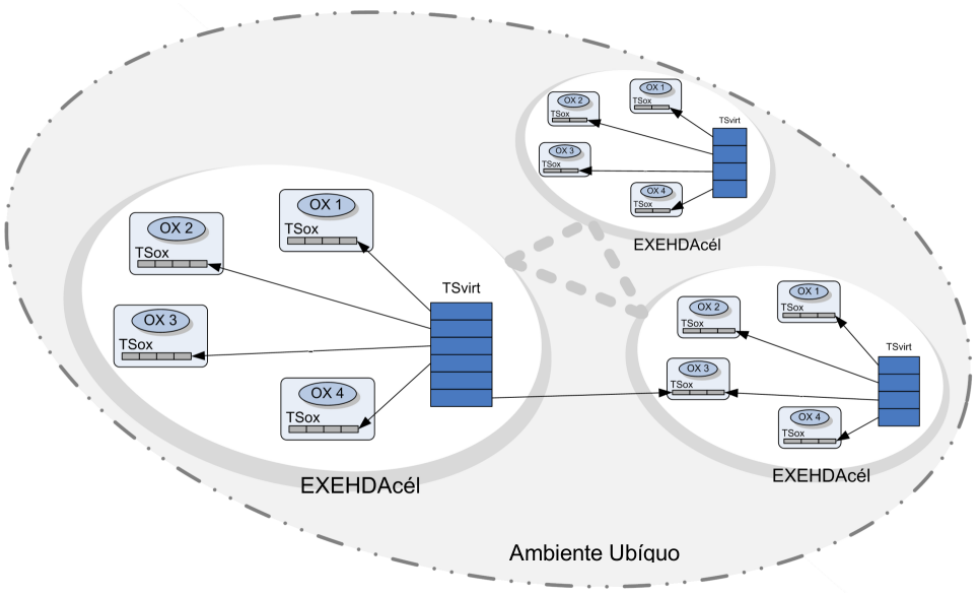

Figura 1. Visão do TSvirt no Ambiente Ubíquo

A abstração introduzida pelo TSvirt proporciona uma transparência em relação aos aspectos de distribuição da infraestrutura computacional. As operações são sempre realizadas sobre um espaço de tuplas genérico, sem qualquer preocupação em relação à sua composição, a qual é gerenciada pelo EXEHDA-TS. A partir desta perspectiva, foi possível dotar o EXEHDA-TS da capacidade de prover acesso ubíquo a dados e concomitantemente atender os aspectos de mobilidade previstos no EXEHDA [Yamin 2004].

No caso de mobilidade física, quando o dispositivo mudar de escopo celular, deve-se registrar a nova localização ajustando-se a referência aos TSox existentes no dispositivo que trocou sua posição. Em relação à mobilidade lógica, o OX deve levar junto consigo o seu TSox durante as migrações, necessitando novamente de uma alteração na sua referência junto ao TSvirt ao qual pertence.

As operações de escrita são sempre realizadas localmente, ou seja, no TSox do próprio OX. Já as operações de leitura podem ser locais ou remotas, porém são realizadas de maneira transparente ao OX requisitante, pois empregam como intermediário o TSvirt, que abstrai a localização física das tuplas. Assim, uma operação de leitura é sempre realizada no TSvirt, sendo que o serviço EXEHDA-TS redistribui a operação a todos TSox que o constituem, sempre que necessário.

4.2. Suporte à Reatividade

Com o intuito de reduzir o custo computacional e o uso da rede, foram modelados para o EXEHDA-TS dois mecanismos que promovem a reatividade, são eles: (i) a subscrição de eventos, e (ii) as consultas ubíquas. Essa estratégia libera os componentes de software do ônus de ter que acompanhar as mudanças nos dados armazenados no espaço de tuplas por conta própria, via de regra através de sucessivas leituras. Essa liberação favorece a escalabilidade.

Através da subscrição de eventos, os componentes de software podem ser notificados sobre a existência de tuplas com informações relevantes imediatamente após elas terem sido inseridas no espaço de tuplas. Para isso, a aplicação deve utilizar uma operação de subscrição (*subscribe*) através da qual é passado como parâmetro um *template* com o padrão da tupla desejada, além de um apontador para a parte do código que deve ser executada quando o evento ocorrer. Dessa maneira, o EXEHDA-TS, através de um módulo gerenciador de eventos, compara cada tupla com os *templates* subscritos à medida que elas são inseridas no espaço de tuplas. Isso irá ocorrer indefinidamente enquanto o OX subscritor existir ou até que seja executada uma operação de cancelamento (*unsubscribe*).

Por sua vez, a operação de consulta ubíqua consiste em uma busca distribuída com geração de eventos. A busca por informações no TSvirt necessita que as operações sejam repassadas primeiramente aos diversos dispositivos em que estão fisicamente localizados nos TSox que compõem o mesmo, em nível celular, e, posteriormente, aos demais TSox distribuídos em outros escopos celulares. Se nenhuma tupla com as características desejadas for localizada após o processo de busca, a operação é mantida ativa nos nodos, a fim de detectar uma possível disponibilização futura, evitando, assim, a execução repetida das complexas consultas distribuídas. Assim como na subscrição de eventos, devem ser repassados como parâmetros na operação um *template* contendo o padrão da tupla desejada e um apontador indicando a parcela de código que deve ser executada pelo evento gerado quando a tupla for localizada. Para que as consultas ubíquas não sejam mantidas indefinidamente ativas à espera de informações, um *timeout* deve ser estabelecido pelo desenvolvedor, e, quando ultrapassado, a consulta é extinta.

4.3. Adaptação Dinâmica

A escalabilidade é uma preocupação sempre presente em projetos de sistemas distribuídos. A larga distribuição do espaço de tuplas pode ser inviabilizada em sistemas

ubíquos se não forem adotados métodos que permitam o controle das comunicações envolvidas [Mamei and Zambonelli 2009]. Nesse sentido, foi adotada a estratégia de dividir com o desenvolvedor das aplicações a decisão a respeito da visibilidade dos TSox a partir dos respectivos TSvirt. Entenda-se por visibilidade do TSox a permissão de que operações de leitura distribuída sejam encaminhadas aos mesmos. Assim, de acordo com as características de cada aplicação, durante a fase de desenvolvimento, podem ser adotadas estratégias que reduzam as comunicações entre as instâncias do EXEHDA-TS, nos diferentes equipamentos.

No EXEHDA-TS, os TSox podem assumir dois modos distintos de visibilidade, os quais podem ser definidos na criação no espaço de tuplas ou em tempo de execução de acordo com a lógica da aplicação. São eles:

- visibilidade permanente: nesse modo de operação, uma vez que a referência ao TSox tenha sido alocada junto ao TSvirt, o mesmo somente deixará de ser visível a partir do TSvirt nos momentos de desconexão, independentemente dos destinos do OX decorrentes de processos de migração;
- visibilidade sob demanda: nesse modo, o TSox fica visível apenas por alguns momentos, como por exemplo, enquanto tiver alguma tupla marcada como importante que ainda não tenha sido lida. Essa estratégia contribui para reduzir os custos de comunicação.

Mesmo que o TSox não esteja visível a partir do TSvirt, este continua tendo o TSvirt como referência para as operações. Portanto, ele continua podendo interagir com os demais componentes de software da aplicação de origem, porém o fluxo de comunicação é reduzido. Essa estratégia foi adotada no EXEHDA-TS com o intuito de promover a escalabilidade.

4.4. Arquitetura do EXEHDA-TS

Para atender as funcionalidades previstas, foi definida para o EXEHDA-TS a arquitetura apresentada na figura 2. Conforme pode ser observado, o serviço EXEHDA-TS consiste em uma estrutura de software integrada ao *middleware* EXEHDA, prestando uma contribuição a coordenação e às comunicações no *middleware*.

A arquitetura do EXEHDA-TS é composta por uma estrutura de dados e um conjunto de módulos, responsáveis pela composição desta estrutura e pela gerência das políticas de acesso às informações. Na base da arquitetura encontra-se o espaço de tuplas distribuído, ou seja o próprio TSvirt. O módulo de Gerenciamento proativo é responsável pela composição do TSvirt através da monitoração das ações dos OX sobre o mesmo. O módulo denominado "Reatividade" no EXEHDA-TS é quem gerencia as subscrições de eventos e as consultas ubíquas. Assim, este módulo monitora as modificações do TSvirt e reage a estas mudanças notificando os OX interessados.

No EXEHDA-TS, as operações mais simples são aquelas realizadas sobre o TSox local, as quais são gerenciadas pelo módulo designado "Acesso Local". Entre essas estão as operações de escrita e também a primeira etapa das operações de leitura. Na medida em que uma operação de leitura não é satisfeita em âmbito local, ela passa a ser realizada nos demais TSox do TSvirt envolvido. Este processo inicia pelos TSox localizados fisicamente no mesmo dispositivo e posteriormente é propagado pela rede aos demais TSox

localizados na célula. Estes processos são gerenciados por um módulo identificado como "Busca Intracelular". A última etapa das operações de leitura são realizadas nos TSox externos à célula. Esta etapa é realizada pelo módulo de "Busca Intercelular".

Um aspecto central na administração de ambientes ubíquos é o compromisso com a escalabilidade do sistema. Sendo assim, foi adotada, na especificação do EXEHDA-TS, uma estratégia de distribuição das tarefas de gerenciamento do TSvirt entre os nodos. Logo, assim como no *middleware* EXEHDA, o serviço EXEHDA-TS é executado em cada nodo do sistema, tendo uma instância nodal e uma base.

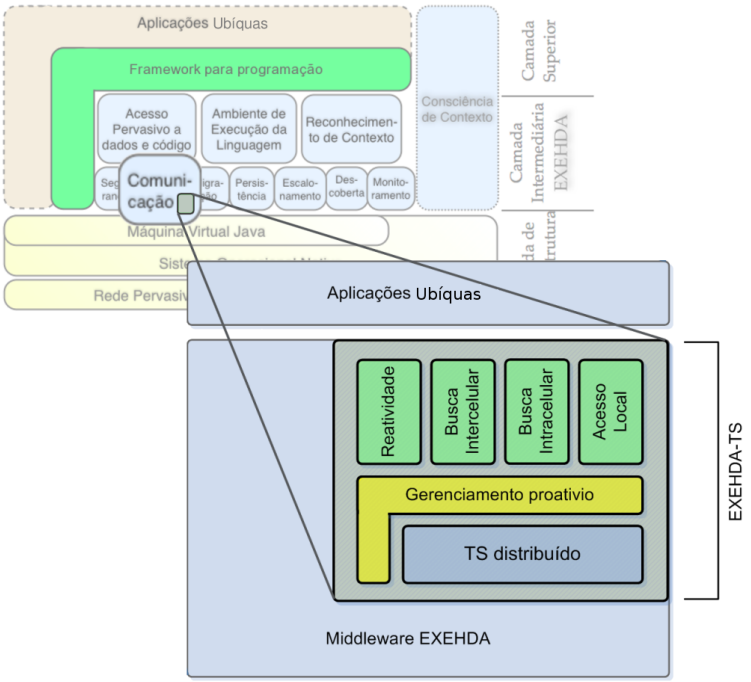

Figura 2. Arquitetura do EXEHDA-TS

5. Estudos de Caso: Medicina Ubíqua

Para validar o modelo proposto, foi desenvolvido um protótipo do EXEDHA-TS utilizando a Linguagem Java, mesma linguagem utilizada na implementação do *middleware* EXEHDA e suportada por sua API.

A avaliação do comportamento do modelo foi realizada utilizando um cenário concebido a partir de situações identificadas no hospital São Francisco de Paula da UCPEL. Esse cenário prevê que um determinado paciente é monitorado através de um conjunto de sensores, e estes se comunicam com o sistema informatizado da unidade médica através de conexões sem-fio. Então em um determinado momento, este paciente, ao se deslocar dentro do hospital para realização de exames, sofre uma crise que é detectada através do sistema de sensoriamento. Em poucos instantes, o médico, de posse de seu *smartphone* nas dependências do hospital, recebe um alerta do sistema informando o problema.

Dentre as características do EXEHDA-TS abordadas neste artigo, para fins de análise, foi utilizado um cenário que permitisse avaliar o tratamento da mobilidade, bem

como a reatividade proporcionada pelo modelo, por estarem entre as funcionalidades mais significativas no âmbito do projeto Pertmed.

Os trabalhos relacionados apresentados nesse artigo, embora tenham propósitos semelhantes ao EXEHDA-TS, foram concebidos com foco em redes *ad hoc*. Isso difere da grande maioria das infraestruturas de rede utilizadas atualmente, como o caso do hospital São Francisco de Paula.

5.1. Análise do Tratamento da Mobilidade

O cenário apresentado caracteriza a mobilidade física do paciente ao se deslocar pelo hospital. Durante o trajeto pode ocorrer perda momentânea do sinal, ou mesmo a troca do ponto de acesso à rede. Isso constitui um cenário em que acontecem desconexões momentâneas do nodo móvel seguidas de uma realocação do dispositivo em relação à rede e, possivelmente alterando seu endereçamento IP. Esse aspecto é gerenciado pelo *middleware* EXEHDA provendo um identificador único de alto nível para o nodo (HostID), abstraindo os aspectos de endereçamento de rede [Lopes et al. 2007].

Durante os eventuais momentos de desconexão, o EXEHDA-TS faz a bufferização das operações que tiverem sido realizadas pelos OXs localizados no equipamento móvel que acompanha o paciente, e, posteriormente, quando a conexão é restabelecida, as operações são processadas. Observa-se que mesmo um dos envolvidos estando momentaneamente impossibilitado de ser acessado, a comunicação é oportunizada pelo EXEHDA-TS, pois o processo mantém suas operações locais. Sendo assim, as informações produzidas durante o período de desconexão podem ser acessadas posteriormente.

O EXEHDA-TS, por sua vez, constitui uma referência para os dados sensoreados independentemente da localização ou da movimentação do produtor em relação à infraestrutura, abstraindo os aspectos de localidade e simplificando as tarefas de desenvolvimento da aplicação.

5.2. Análise do Suporte à Reatividade

A reatividade provida pelo EXEHDA-TS pode ser utilizada como suporte básico ao mecanismo de consciência e adaptação ao contexto do *middleware* [Warken and Yamin 2010, Rodrigues et al. 2011].

A subscrição de eventos, oportunizada pelo serviço, pode ser ativada pelo médico por intermédio da interface do prontuário eletrônico utilizado por ele. Com isso, o *template* fornecido nessa subscrição é inserido em uma tabela alocada na instância nodal do EXEHDA-TS, que, por sua vez, repassa uma cópia à instância base. Esse *template* ainda pode ser propagado às instâncias base do serviço EXEHDA-TS de outras células. Essa estratégia promove a escalabilidade do sistema através da distribuição das tarefas de gerenciamento dos eventos entre os nodos que compõem o TSvirt.

Cada vez que uma tupla associada ao paciente é inserida no TSox, esta é comparada com os *templates* gerenciados pela instância nodal do EXEHDA-TS e posteriormente com aqueles alocados na instância base, que possui todos os *templates* associados aos TSvirts gerenciados pela célula. Quando o *template* submetido a partir do software utilizado pelo médico é localizado, um evento é produzido e a tupla identificada é repassada, possibilitando a geração te uma notificação na interface do prontuário eletrônico. Esse mecanismo é apresentado na figura 3.

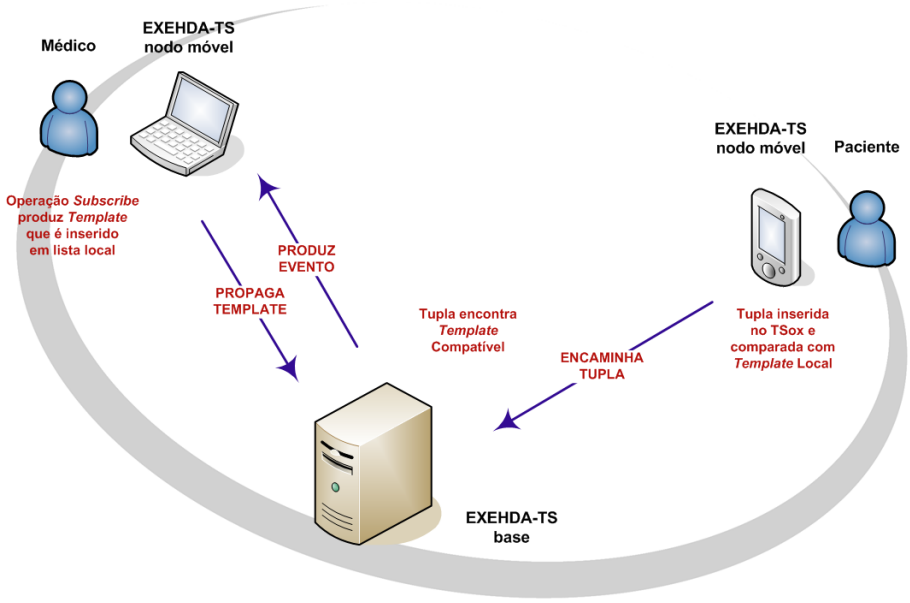

Figura 3. Fluxo de Processamento de Evento

Deste modo, mesmo que o nodo móvel do paciente mude sua posição em relação à infraestrutura da célula, o médico pode permanecer recebendo as notificações do sistema sem que seja necessário nenhum procedimento extra por parte dele. A abstração promovida pelo EXEHDA-TS através do TSvirt permite que a aplicação continue operacional independentemente da dinâmica dos nodos em relação à infraestrutura da célula.

6. Conclusão

Os *middlewares* atuais com foco em ambientes móveis e dinâmicos não suprem as demandas de coordenação do *middleware* EXEHDA. Motivado por isso, foi concebido o EXEHDA-TS, que consiste de um serviço para o *middleware* EXEHDA capaz de gerenciar um sistema de espaço de tuplas de forma escalável e com atuação proativa, direcionado as demandas típicas da UbiComp.

Com o andamento da pesquisa foi evidenciado que para não comprometer aspectos de escalabilidade é necessário que o espaço de tuplas tenha um escopo de abrangência limitado e controlado. Outrossim, a incorporação de mecanismos reativos são fundamentais em cenários de larga distribuição para evitar que as aplicações tenham que monitorar o espaço de tuplas por conta própria, evitando um intenso uso da rede assim como uma elevação no custo das computações.

Por sua vez entende-se que o gerenciamento do comportamento reativo deve ser feito de forma distribuída entre as duas instâncias da arquitetura de software utilizada como suporte (nodal e base), a fim de reduzir os esforços computacionais.

Um maior detalhamento da modelagem do EXEHDA-TS assim como os demais testes que foram realizados estão disponíveis em [Souza 2009].

Referências

Augustin, I., Yamin, A. C., and da Silva, L. C. (2008). Building a Smart Environment at Large-scale with a Pervasive Grid Middleware. In Wong, J., editor, *Grid Computing*

Research Progress, pages 323–344. Nova Science, New York.

Cabri, G., Ferrari, L., Leonardi, L., Mamei, M., and Zambonelli, F. (2006). Uncoupling Coordination: Tuple-Based Models for Mobility. In Bellavista, P. and Corradi, A., editors, *The Handbook of Mobile Middleware*. Auerbach Publications.

da Costa, C. A., Yamin, A. C., and Geyer, C. F. R. (2008). Toward a General Software Infrastructure for Ubiquitous Computing. *IEEE Pervasive Computing*, 7(1):64–73.

Grimm, R., Wetherall, D., Davis, J., Lemar, E., Macbeth, A., Swanson, S., Anderson, T., Bershad, B., Borriello, G., and Gribble, S. (2004). System support for pervasive applications. *ACM Transactions on Computer Systems*, 22(4):421–486.

Julien, C. and Roman, G.-C. (2006). EgoSpaces: facilitating rapid development of context-aware mobile applications. *IEEE Transactions on Software Engineering*, 32(5):281–298.

Lopes, J. a. L. B., Pilla, M. L., and Yamin, A. C. (2007). EXEHDA: a Middleware for Complex, Heterogeneous and Distributed Applications. *Conferência Nacional em Inteligência Computacional Aplicada à Indústria de Petróleo*.

Mamei, M. and Zambonelli, F. (2009). Programming Pervasive and Mobile Computing Applications: The TOTA Approach. *ACM Transactions on Software Engineering and Methodology*, 18(4).

Murphy, A. L., Picco, G. P., and Roman, G.-C. (2006). LIME: A coordination model and middleware supporting mobility of hosts and agents. *ACM Transactions on Software Engineering and Methodology*, 15(3):279–328.

Rodrigues, S. L., Dilli, R. M., Nelsi Warken, Venecian, L. R., Lopes, J. a. L. B., Augustin, I., Yamin, A. C., and Geyer, C. F. R. (2011). Um Framework para o Gerenciamento de Aplicações Direcionadas à Medicina Ubíqua. In *III Simpósio Brasileiro de Computação Ubíqua e Pervasiva - SBCUP*.

Souza, R. (2009). Uma Contribuição à Coordenação na Computação Pervasiva com Aplicações na Área Médica. Master thesis.

Valente, M. T., Pereira, F. M., Bigonha, S., and Andrade, M. (2003). A Coordination Model for ad hoc Mobile Systems. In *Euro-Par*, pages 1074–1081. Springer.

Want, R., Bardram, J., Friday, A., Langheinrich, M., Brush, A. J. B., Taylor, A. S., Quigley, A., Varshavsky, A., Patel, S., Dey, A. K., and Krumm, J. (2010). *Ubiquitous Computing Fundamentals*. Chapman & Hall/CRC.

Warken, N. and Yamin, A. C. (2010). Uma Arquitetura para Controle da Adaptação Dinâmica na Computação Ubíqua. In *II Simpósio Brasileiro de Computação Ubíqua e Pervasiva - SBCUP*, pages 326–335.

Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, 265(3):94–104.

Yamin, A. C. (2004). *Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva*. PhD thesis.