INSTITUTO SUPERIOR TÉCNICO
UNIVERSIDADE TÉCNICA DE LISBOA

Adaptabilidade e Modelos de Coerência de Dados em Redes Móveis

Proposta de dissertação para obtenção do grau de Doutor em Engenharia Informática e de Computadores

Candidato: João Coelho Garcia

Orientador: Doutor Paulo Jorge Pires Ferreira

Comissão: Doutor José Manuel Alves Marques
Doutor Mário Gaspar da Silva
Doutor António Rito Silva
Doutor Paulo Jorge Pires Ferreira

Resumo

A crescente mobilidade dos computadores, faz emergir a adaptação ao ambiente como um requisito importante de um sistema moderno. A adaptação ao ambiente só é possível se previamente houver a capacidade de percepcionar o contexto, captando as propriedades computacionais, físicas e até mesmo do utilizador.

Esta dissertação aborda o problema da aplicação da percepção do contexto ao aspecto particular da gestão de dados. Os principais contributos que se pretendem alcançar são a definição de uma arquitectura que simplifique a aquisição de informação de contexto pelas aplicações, a extensão dessa arquitectura com um mecanismo de previsão do contexto futuro e a definição de um modelo que automatize e/ou parametrize as operações principais envolvidas na gestão de dados em sistemas distribuídos: a replicação, a gestão de cache, a submissão de actualizações e a resolução de conflitos.

A arquitectura de percepção de contexto apresentada especifica textualmente as propriedades e eventos do contexto e permite às aplicações desenvolverem as suas descrições específicas dos eventos e situações que lhes interessam.

O algoritmo de previsão de contexto futuro será integrado na arquitectura de percepção de contexto de forma a potenciar a informação disponível às aplicações.

Resultados e Planeamento

Publicações

- Concurrency Control for Distributed Co-operative Engineering Applications. João Garcia, Paulo Ferreira. SAC 2002. Madrid. Março 2002.
- CORBA Persistence Object Service using PerDiS. Edi Ray Zavaleta Olea, João Garcia, Paulo Ferreira. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002). Orlando, USA. Junho 2002.
- Environment Awareness: Telling applications about the present and the future, João Garcia, Luís Veiga, Paulo Ferreira. Relatório Técnico INESC ID Lisboa. 2002.

Sistemas e Protótipos

- Arquitectura do *Environment Awareness System* (descrito na secção 3.2)
- Protótipo de uma aplicação do EAS, em desenvolvimento (descrito na secção 3.3)

Planeamento

- Estudo do Trabalho Relacionado e Planeamento de Investigação
  - Início: Março de 2002
  - Conclusão: Julho de 2004
- Avaliação de Proposta de Dissertação
  - Data: Julho de 2004
- Investigação e Concretização de Soluções
  - Data de Início: Julho de 2004
  - Data de Conclusão: Julho de 2006
- Escrita da Dissertação
  - Início: Julho de 2006

  - Conclusão: Dezembro de 2006

- Provas
  - Data: Dezembro de 2006

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O mundo dos computadores nasceu de costas voltadas para o mundo físico. Os computadores foram criados para lidar com a mais abstracta criação/descoberta do homem: a Matemática. Algumas das mais importantes descobertas da informática, como os sistemas operativos e as pilhas de protocolos de comunicação, são mecanismos bem sucedidos de esconder a realidade física do mundo computacional.

Até há poucos anos, os utilizadores eram os únicos intermediários entre os computadores e o mundo físico. Sempre que as circunstâncias em torno de um computador se alteravam, tinha de ser um utilizador ou administrador de sistema a reconfigurar o computador, quer se tratasse de uma mudança na rede a que o computador estava ligado, de uma mudança de fuso horário por se ter levado o computador para outro fuso ou uma afinação da velocidade de processamento para poupar a bateria de um computador portátil.

No entanto, com o aparecimento, por um lado, de diversos sensores ou meios de contactar sensores em computadores e, por outro lado, de mecanismos de computação e comunicação associados a objectos do mundo físico, prevê-se uma convergência cada vez maior entre o mundo físico e o mundo virtual.

É expectável que, no futuro, cada vez mais objectos do mundo físico tenham alguma forma de ligação com o mundo digital. Essa ligação pode ser feita de formas tão simples como com mecanismos passivos de identificação e/ou localização até formas tão sofisticadas como a integração com pequenos computadores com capacidade de processamento, armazenamento e comunicação.

Também se tem observado uma crescente inclusão nos computadores de mecanismos para percepcionar o contexto ao seu redor. O contexto computacional é percepcionado usando protocolos de localização de serviços enquanto que o contexto físico é percepcionado usando sensores e antenas (temperatura, GPS, carga

da bateria, luminosidade, etc...). Isto é possível devido, por um lado, à constante descida do custo destes sensores, mas também é motivado pela mobilidade de muitos dispositivos. Quanto mais variadas puderem ser as circunstâncias de utilização de um dispositivo mais interessante é caracterizar o contexto que o rodeia.

Como consequência da aproximação entre estes dois mundos, existe um número crescente de aplicações informáticas que a exploram e prevê-se que um número ainda maior o possa vir a fazer no futuro. Da perspectiva da gestão de dados, esta convergência tem dois aspectos particularmente interessantes. Por um lado, existem objectos com os quais pode ser interessante os computadores interactuarem mas que até agora não era possível (sensores, recursos físicos). Por outro lado, a sofisticação que muitos dispositivos (telemóveis, consolas de jogos, electrodomésticos, automóveis, postos multimédia em transportes colectivos) estão a atingir configuram-nos como potenciais plataformas computacionais e participantes em partilha de recursos com outros computadores.

Uma das áreas onde tem havido desenvolvimentos significativos é no domínio dos identificadores electrónicos. É viável hoje em dia ter autocolantes com circuitos de identificação por rádio frequência (RFID) a preços extremamente baixos. De facto, já é possível dotar um escritório com pastas de documentos todas dotadas de RFID e de um conjunto de antenas sendo assim possível localizar qualquer documento através de um interface computacional. A identificação RFID é um exemplo de uma tecnologia de comunicação que, caso seja utilizada para identificar recursos computacionais e periféricos (tomadas eléctricas, tomadas de rede, ecrãs, etc...), pode ser também usada para permitir aos computadores avaliarem melhor o seu contexto operacional.

Outra observação relacionada com a ubiquidade é a da proliferação das capacidades computacionais de muitos dispositivos que utilizamos no nosso quotidiano. Existem cada vez mais dispositivos com significativas capacidades de computação e comunicação. Actualmente é comum existirem utilizadores que recorrem às máquinas fotográficas para transportar ficheiros entre computadores ou que utilizam consolas de jogos para navegar na *World Wide Web*.

1.1 Exemplos Motivadores

Há pois cada vez mais oportunidades e situações onde os utilizadores podem tirar partido dos seus dados digitais. Estas oportunidades levantam, no entanto, novas questões quanto à disponibilidade e à coerência dos dados levados até esses dispositivos.

Não são só os dispositivos mais simples que estão a ver aumentadas as suas

capacidades computacionais, mas também se observa uma proliferação de sensores capazes de detectar propriedades do contexto mais complexas. Um exemplo simples das sinergias que podem resultar da adição de novos sensores a dispositivos é o acoplamento de um sistema GPS[1] a uma agenda. Com este conjunto, pode, por exemplo, ter-se uma agenda que notifica os utilizadores não só quando se está no momento de fazer uma tarefa, mas também quando se está no local onde pode ser realizada uma dada actividade.

Outro cenário interessante de utilização de informação de contexto (este será explorado mais adiante no capítulo 3) é o que motiva a criação de uma aplicação que seja capaz de, em função da agenda futura de um utilizador, copiar para a sua agenda electrónica todos os ficheiros necessários para as reuniões que terá até regressar ao seu escritório. Enquanto o utilizador está deslocado, caso ele pretenda realizar alterações nalgum desses ficheiros, será escolhido o computador mais adequado daqueles que se encontrem nas redondezas da agenda electrónica e será escolhida a ligação de rede ideal para propagar essas actualizações até ao servidor de ficheiros do escritório.

Observamos pois uma homogeneização de funcionalidades dos dispositivos em que um número cada vez maior dispõem de processador, capacidade de comunicação e capacidade de armazenamento. Enquanto que há alguns anos não encontrávamos computadores pessoais com capacidade de comunicação sem fios nem telefones com a possibilidade de carregarem e executarem código hoje em dia as diferentes capacidades tendem a expandir-se a todos os dispositivos. Note-se que isto não quer dizer que todos os dispositivos sejam iguais. Se as funcionalidades tendem a homogeneizar-se, continuamos a observar à nossa volta uma enorme variedade em termos quantitativos que vai desde o auricular de telemóvel com ligação *Bluetooth* até ao supercomputador.

Outra consequência da miscigenação entre os mundos computacionais e físico é um relaxamento das formas de organização das redes informáticas. Enquanto que as redes constituídas por computadores estacionários têm topologias estáticas que permitem a introdução de forma simples de mecanismos de configuração, administração e partilha de dados, as redes constituídas por dispositivos móveis têm uma natureza muito mais dinâmica. Se, da perspectiva dos utilizadores, esta nova realidade permite uma grande flexibilidade geográfica e funcional na utilização dos computadores, do ponto de vista da concepção de software de suporte às aplicações (sistemas operativos, *middleware*), torna-se mais complexo fornecer um ambiente de execução de aplicações coerente e adaptado a este novo ambiente variável.

[1] *Global Positioning System*

1.2 Objectivos

A percepção tem inúmeros aspectos que serão detalhados no capítulo 2. Desde os diferentes sensores que podem ser acoplados a um dispositivo, até à forma como essa informação é armazenada, combinada e até usada para realizar previsões. Mas a informação de contexto só tem utilidade quando serve para aumentar a funcionalidade de um dispositivo e das aplicações nele executadas.

Assim, o objectivo desta dissertação consiste em desenhar e implementar uma infra-estrutura de *software* que permita às aplicações melhorar a gestão de dados e a utilização de recursos computacionais em computação móvel e ubíqua.

Em particular pretendo desenvolver uma infra-estrutura de *middleware* que utilize toda a informação de contexto necessária para que um utilizador, ou conjunto de utilizadores de dispositivos computacionais, tenham um acesso o mais coerente e completo aos seus dados independentemente de circunstâncias adversas, tais como uma localização remota ou uma baixa conectividade.

Esta dissertação centra-se em torno de três questões no âmbito do suporte de sistema para aplicações móveis:

- A percepção do contexto e a adaptação de aplicações e dados a ambientes móveis.
- A concepção de modelos de coerência de dados para ambientes móveis.
- O contributo da informação de contexto para a gestão de réplicas.

1.2.1 Percepção do Contexto

A primeira questão prende-se com o facto da computação móvel se caracterizar por uma grande variedade de condições ambientais, quer em termos físicos quer em termos computacionais, e ser portanto de todo o interesse adaptar as aplicações e os dados ao seu ambiente de execução. A adaptação das aplicações permite que estas tenham o melhor desempenho possível face às circunstâncias que as rodeiam: regulando o seu consumo de recursos no dispositivo onde se executam, extraindo o máximo de informação do contexto físico em seu redor e tirando o melhor partido possível dos recursos computacionais disponíveis, por exemplo utilizando o canal de comunicação com maior largura de banda, gerindo a energia até à próxima ocasião de recarga ou armazenando dados em discos partilhados na sua imediação.

O próprio processo de percepção do contexto levanta problemas. É necessário que se acorde um protocolo de comunicação entre o computador que tenta percepcionar

o contexto e os sensores e serviços ao seu redor. É ainda necessário que o computador consiga armazenar, filtrar, processar toda a informação resultante dessa percepção para posteriormente poder tomar opções a partir dessa informação. Um aspecto particularmente interessante da percepção do contexto é a possibilidade de realizar previsões do contexto futuro em função do histórico recolhido.

Por exemplo, se um computador mantiver um histórico da qualidade da rede de comunicação entre si mesmo e um outro computador remoto, pode, usando um modelo de previsão, detectar o padrão temporal dessas variações de qualidade passando assim a conhecer o momento ideal para fazer transferências de dados com esse servidor.

A adaptabilidade dos dados torna-se necessária quando um programa se executa em diferentes computadores e os dados que é possível (ou relevante) armazenar e apresentar ao utilizador variam de dispositivo para dispositivo. Nestes casos, seria interessante criar mecanismos para automaticamente seleccionar e filtrar repositórios de dados adaptando-os a dispositivos com menos recursos.

1.2.2 Modelos de Coerência de Dados para Computação Ubíqua

A segunda questão é motivada fundamentalmente pela constatação de que na maioria dos ambientes móveis é utilizada a distribuição e replicação de dados para lidar com o problema da conectividade fraca ou intermitente entre dispositivos que partilham dados. A utilização de replicação implica a necessidade de dispor de um modelo de coerência de dados para estruturar as relações entre essas réplicas. Existe uma quantidade significativa de trabalho realizado no domínio dos modelos de coerência para redes tradicionais. A extensão dos modelos existentes na direcção dos novos ambientes móveis pode envolver pelo menos três componentes, eventualmente complementares: uma de sistema, outra organizacional e outra ainda semântica.

A componente de sistema prende-se com a pretensão de aproximar o mais possível as redes móveis das redes tradicionais usando combinações e evoluções de técnicas existentes (*caching*, recolha prévia de dados, compressão de comunicações, transmissão selectiva de dados) para ultrapassar os problemas de comunicação inerentes à mobilidade.

Na prática, este objectivo corresponde a responder às seguintes questões de uma perspectiva das redes móveis ubíquas:

- Que dados externos devem ser copiados para cada dispositivo? É possível prever que dados vão ser usados e que ligações vão estar disponíveis caso se

quisesse obter os dados mais tarde?

- Em que formato devem os dados ser copiados? Em que outros dispositivos serão utilizados e quais as capacidades desses dispositivos?
- Caso existam alterações quando devem ser submetidas as versões originais dos dados?
- Como enviar as actualizações?
- Como resolver conflitos que ocorram na reconciliação?

Estas questões já foram, total ou parcialmente, respondidas para redes estáticas (ver secção 2.6), mas a sua abordagem para redes móveis e/ou ubíquas obriga a que sejam reequacionadas. Por exemplo, a maior parte dos sistemas que lidam com este tipo de problemas considera um ambiente de rede mais tradicional em que os dispositivos têm capacidade ilimitada e a avaliação do acesso a uma rede de computadores é binária (ligado/desligado) e sem horizonte temporal.

Na vertente organizacional reconhece-se que, num sistema móvel, a replicação, e consequente divergência de réplicas, são inevitáveis e pretende-se extrair da estrutura organizativa da rede e dos seus utilizadores (relações entre réplicas, dispositivos e utilizadores, trabalho definitivo vs. trabalho tentativo, âmbito de visibilidade das réplicas, etc...) informação que apoie a resolução de conflitos e a organização de versões divergentes.

Finalmente, na componente semântica o objectivo é obter o máximo de meta-informação acerca dos dados de forma a conseguir raciocinar sobre qualquer conflito entre réplicas e chegar da forma o mais automática possível à sua catalogação e eventual resolução.

1.2.3 Modelação de repositórios e réplicas

O prolongamento natural deste tema, e que constitui a terceira questão antes referida, é o de utilizar a informação de contexto recolhida para auxiliar um mecanismo de gestão de réplicas. Tipicamente, estes mecanismos operam sobre registos de operações realizadas em cada réplica e uma descrição do domínio sintáctico dos dados para tentar reconciliar réplicas divergentes automaticamente. Será interessante investigar em que medida é que a informação de contexto poderá contribuir para uma mais eficiente reconciliação entre réplicas.

Em síntese, esta dissertação, por um lado, discute modos de adaptar o contexto de execução de aplicações e os respectivos dados às características físicas e computacionais dos ambientes móveis e, por outro lado, investiga novos modelos de

coerência de dados e mecanismos que evitem e/ou permitam a resolução, com base em informação e meta-informação semântica e organizativa, conflitos entre réplicas de dados.

Capítulo 2

Trabalho Relacionado

Neste capítulo é exposto o trabalho existente sobre os temas da percepção do contexto e da gestão de dados replicados. Começo por definir o conceito de contexto e as suas especializações. Apresento os principais de protocolos de interligação entre dispositivos e de localização de serviços.Seguidamente apresento as infra-estruturas mais relevantes que permitem a sistemas informáticos percepcionar o seu contexto.

Por último, traço uma panorâmica dos principais sistemas que desenvolveram mecanismos para a gestão de dados em dispositivos móveis. A catalogação destes mecanismos é relevante porque a análise destes sistemas contribuirá certamente para delimitar os parâmetros envolvidos na gestão de dados e quais as suas interacções com o contexto circundante.

2.1 Definição de Contexto

Devido à sua mobilidade e à variedade de dispositivos existentes nos ambientes computacionais actuais, a adaptabilidade é uma característica cada vez mais importante nos computadores. A percepção do contexto ou sensibilidade ao contexto (*context awareness*) é um aspecto chave da adaptabilidade.

Vários autores ([SAW94], [Abo99], [HHS+99], [HIR01], [CCLG02]) têm tentado definir o que é o contexto de um sistema computacional e o que significa uma aplicação ser sensível ao contexto. Essas definições podem ser resumidas afirmando que: Um sistema sensível ao contexto é aquele que automaticamente adapta o comportamento das aplicações à sua percepção das circunstâncias que o rodeiam.

Se um sistema se quer adaptar ao ambiente ou contexto ao seu redor, tem primeiro de ter a capacidade de o percepcionar. Antes de abordar os sistemas existentes

que realizam percepção do contexto, consideremos primeiro o conceito de contexto, as suas possíveis especializações e os mecanismos para o percepcionar.

2.2 Tipos de Contexto

O contexto em torno de um dispositivo computacional pode ser dividido em contexto computacional, físico e dos utilizadores [SAW94]. O contexto computacional engloba as características do próprio sistema, das redes de comunicação acessíveis e dos dispositivos colocados nessa rede.

O contexto físico consiste nas características ambientais que podem ser detectadas pelo dispositivo. A variedade das propriedades detectadas depende grandemente dos sensores associados ao computador.

O contexto do utilizador é composto pelas circunstâncias emocionais (o estado de espírito do utilizador), sociais (a actividade que ocupa o utilizador e quem o acompanha) e intencionais (o que o utilizador pretende fazer).

2.2.1 Contexto Computacional

A adaptação ao contexto computacional é dos aspectos mais explorados na adaptabilidade dos computadores já que são as restrições de recursos computacionais que mais tem limitado a computação móvel. Existem propriedades do contexto computacional que são de fácil avaliação já que são valores conhecidos do sistema operativo como a memória total, a memória disponível ou a autonomia energética. Ainda acerca da alimentação de dispositivos computacionais, o crescente uso de rótulos de identificação por rádio-frequência (RFID) poderão contribuir para os utilizadores serem avisados de tomadas eléctricas para carregarem os seus dispositivos. Este mecanismo, na fronteira entre o contexto físico e o contexto computacional, poderá ainda ser expandido a outros aspectos como a detecção de pontos de acesso a redes fixas de comunicação.

Outras propriedades computacionais têm de ser avaliadas usando programas de medida como é o caso da velocidade de processamento ou a qualidade de serviço da rede de comunicação.

Resta ainda avaliar aquilo que outros computadores podem fornecer. Os recursos partilhados entre computadores surgem geralmente sob a forma de serviços. Os serviços em torno de um computador são detectados geralmente usando um protocolo de detecção de serviços (ver 2.4).

2.2.2 Contexto Físico

As propriedades do contexto físico cuja percepção mais tem sido utilizado para aplicações sensíveis ao contexto são [CK00]:

Localização Relativa A capacidade de compreender onde se encontra um dispositivo é das mais interessantes para adaptar aplicações. Existem diversos sistemas de localização relativa [HB01] desde cartões até sistemas sem fios via infra-vermelhos, rádio ou micro-ondas.

Localização Absoluta A localização absoluta dos dispositivos é, em geral, menos interessante do ponto de vista do desenho de aplicações, mas pode ser mapeada em localizações relativas e tem a grande vantagem de ser actualmente um esquema verdadeiramente global por via do sistema GPS[1]

Acesso a Rede de Comunicação Actualmente, observa-se uma proliferação de redes sem fios. Principalmente, existem redes de Ethernet sem fios (seguindo o protocolo 802.11) e redes de telefonia sem fios (seguindo o protocolo GPRS). Torna-se pois uma propriedade de contexto importante saber se existe alguma rede de comunicação fixa ou sem fios disponível e qual a mais conveniente.

Tempo Finalmente, uma propriedade do contexto físico que há muito está disponível nos computadores é o tempo que é usado para todo o tipo de aplicações desde datar documentos até medir a duração de tarefas e implementar agendas.

2.2.3 Contexto de Utilizador

O contexto do utilizador é o aspecto do contexto mais difícil de percepcionar já que envolve quase sempre interpretar as acções e situações sociais humanas e, por isso, é o tipo de contexto onde se têm feito menores progressos na sua percepção. Esta é uma área de charneira com os estudos de interfaces pessoa-máquina e poder-se-ão aí obter ensinamentos úteis para a computação com percepção do contexto.

Os aspectos mais visíveis do contexto do utilizador como a presença ou não de um ou mais utilizadores ou informação proveniente da agenda do dono de um dispositivo até já foram utilizados em aplicações sensíveis ao contexto (e.g. o quadro DUMMBO desenhado para o *Context Toolkit* , [SDA99]). As propriedades que levantam mais desafios para o futuro são pois a compreensão das emoções e intenções do utilizador. Por exemplo, é simples detectar que se encontram várias pessoas numa sala, mas isso

[1] *Global Positioning System*

corresponde a uma reunião? Um utilizador utiliza muitos menus de uma aplicação sem seleccionar um comando. Estará desorientado?

Para além destes três tipos de contexto – computacional, físico e de utilizador –, o contexto pode ainda ser classificado quanto à forma como são organizados os dados recolhidos. A forma mais evidente de o fazer é criando um histórico dos valores de propriedades de contexto observadas.

Outra operação habitual realizada com dados de contexto é a sua composição.A composição de diferente informação de contexto é fundamental para se poder obter informação de contexto mais complexa que frequentemente é a que tem mais significado para os utilizadores.

Uma forma ainda mais sofisticada de complementar a recolha de dados de contexto é a previsão de contexto. A partir do histórico do contexto, usando modelos estatísticos de previsão, é possível detectar padrões temporais na propriedades de contexto e a partir desses padrões estimar os seus valores futuros.

2.3 *Middleware*

Um componente determinante para a percepção do contexto e também para as transferências de dados necessárias para a gestão de dados replicados é a escolha de uma tecnologia de *middleware* que abstraia da forma mais eficiente possível a invocação remota de serviços e o acesso a dados partilhados.

2.3.1 PARCTAB T-RPC

Como experiência pioneira, o projecto PARCTAB escolheu para mecanismo de interacção no seu sistema distribuído o T-RPC, um desenvolvimento das tradicionais *Remote Procedure Calls* realizado no âmbito do projecto. O objectivo desta extensão ao RPC era abstrair a invocação remota de uma função da localização do dispositivo onde se encontra o componente invocado e de eventuais erros de comunicação que sucedam.

O T-RPC não é um mecanismo de invocação genérico mas sim uma biblioteca com a capacidade de realizar um conjunto de chamadas pré-definidas que foram as que os autores consideraram necessárias para interagir com os dispositivos PARCTAB (ver secção 2.5.1). As chamadas para um dado dispositivo são sempre direccionadas para o seu agente de software que se encontra fixo numa estação de trabalho que se encarrega de repetir a chamada até conseguir contactar o PARCTAB correspondente. O agente tem em linha de conta o facto de o dispositivo móvel estar

acessível ou não e encarrega-se também de garantir a ordenação de invocações e a eliminação de duplicados.

2.3.2 Espaços de tuplos

As redes de comunicação sem fios ou intermitentes favorecem um mecanismo de comunicação independente da existência de ligações. Assim, os mecanismo de comunicação síncronos habituais nos sistemas distribuídos tradicionais devem ser substituídos por mecanismos de comunicação assíncronos dos quais os espaços de tuplos são um bom exemplo.

As origens dos espaços de tuplos remontam à linguagem de coordenação Linda. Os espaços de tuplos são repositórios de tuplos que são criados por processos e colocados nesse espaço por uma primitiva de escrita. O espaço pode ser acedido concorrentemente por diversos processos que lêem tuplos usando primitivas que consultam ou retiram os tuplos. Os tuplos são anónimos e a sua selecção é feita por emparelhamento de padrões. Este modelo de comunicação é pois ideal para a distribuição no espaço e no tempo. A adaptação do modelo do espaço de tuplos a um ambiente móvel distribuído já foi abordado por em diferentes sistemas [WMLF98, Wal98, PMR99] que se ocuparam com o problema da persistência e da distribuição. . No entanto, a maior parte destes sistemas foca-se principalmente na persistência. Relativamente, à distribuição a maior parte dos sistemas usa os tuplos como um modo simples de particionar e reagrupar dados separando-os em espaços de tuplos diferentes.

2.3.3 *Globe*

O sistema Globe [SHHT99, BST99] preconiza uma abordagem para simplificar o desenvolvimento de aplicações distribuídas garantindo que os objectos que as constituem são replicados por todos os computadores onde possam ser necessários, sendo a política de replicação controlada pelas aplicações. Com esta abordagem de considerar um objecto uma entidade distribuída, o desenvolvimento de aplicações distribuídas é levado um pouco mais longe na medida em que as aplicações podem abstrair-se dos problemas de comunicação, *caching* e replicação, podendo, no entanto, influenciar o comportamento em termos de consistência entre as réplicas do objecto.

Porém, a plataforma Globe está vocacionada para aplicações distribuídas fixas, já que não contempla a possibilidade de existirem problemas de comunicação entre os diferentes nós envolvidos.

2.4 Protocolos de Descoberta de Serviços

A descoberta de serviços é, como já referi, um aspecto fundamental do contexto computacional. Existem diferentes abordagens ao problema de um dispositivo detectar uma rede em que se possa integrar, registar-se nessa rede e publicar e usufruir dos serviços que se encontrem noutros dispositivos dessa rede. Na Figura 2.1 podemos observar um modelo genérico do que é a subscrição de serviços e de que apresento de seguida algumas das variantes mais divulgadas.

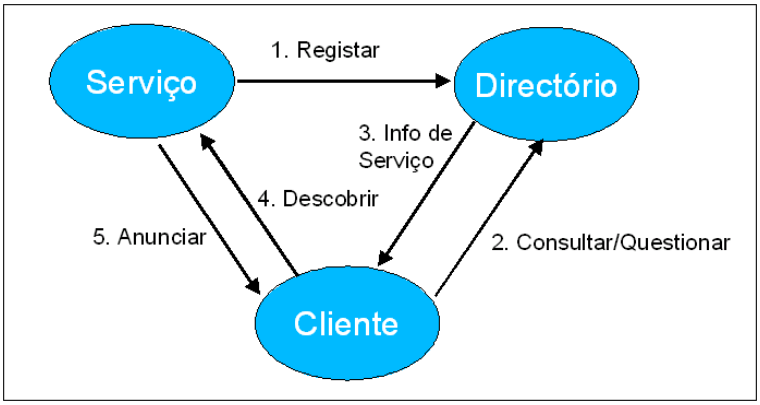

Figura 2.1: Arquitectura genérica de um mecanismo de localização de serviços

2.4.1 SLP - *Service Location Protocol*

O protocolo SLP [SLP] é a principal referência em termos de localização de serviços. No modelo SLP existem três tipos de entidades:

- SA (*Service Agent*)
- DA (*Directory Agent*)
- UA (*User Agent*)

O SA representa um serviço na rede de comunicação e anuncia os serviços que disponibiliza por multicast ou difusão (ver Figura 2.2). Para além disso, recebe e responde aos pedidos de utilização desses serviços (SrvRply). Os pedidos de utilização de um pedido (SrvRqst) são emitidos pelos UA.

De forma a interacção entre SA e UA ser mais dinâmica existem os DA que são directórios de serviços. Os DA recebem as mensagens de anúncio de serviços

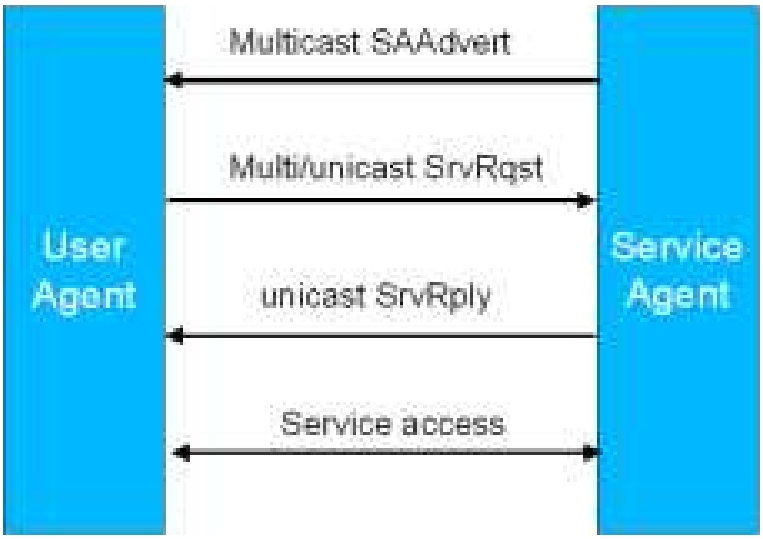

Figura 2.2: Interacção entre um cliente e um servidor no SLP

enviadas pelos SA e criam uma lista de serviços disponíveis. Quando surge um SrvRqst, os DA respondem com o endereço de um SA que conste do seu directório.

De forma a simplificar a localização dos DA, que funcionam como ponto de entrada para a utilização de serviços, os endereços dos DA são distribuídos por DHCP quando os UA se tentam registar na rede. Os serviços são representados por URLs e por um conjunto de atributos que descrevem os seus parâmetros.

2.4.2 UPnP - *Universal Plug and Play*

O protocolo *Universal Plug and Play* (UPnP) da Microsoft pretende ser uma extensão para além dos limites do computador do modelo *Plug and Play* onde o sistema operativo detecta e instala qualquer periférico que seja ligado ao computador. O UPnP divide a rede em dispositivos (*devices*), serviços (*services*) e pontos de controlo (*control points*). Os dispositivos são cada um dos aparelhos ligados à rede. Os serviços são o menor unidade que pode ser invocada por outro dispositivo e os pontos de controlo são dispositivo especiais que centralizam a informação acerca dos serviços na rede.

Todos os dispositivos são identificados usando *Dynamic Host Configuration Protocol* (DHCP). Os serviços fornecidos por um dispositivo são anunciados usando o protocolo *Simple Service Discovery Protocol* (SSDP) que permite encontrar os pontos de controlo da rede através de difusão sobre UDP/IP. O anúncio dos serviços de

um dispositivo inclui um *Uniform Resource Locator* (URL) que permite ao ponto de controlo obter a descrição dos serviços expostos pelo dispositivo. Os serviços são invocados usando o *Simple Object Access Protocol* (SOAP). O modelo UPnP é ainda complementado pelo mecanismo de cada ponto de controlo saber quais os tipos de serviço que lhe interessam.

Assim, por exemplo, numa casa onde todos os electrodomésticos suportem UPnP, se se ligar um controlo remoto, ele usará o SSDP para difundir mensagens de interesse em conhecer todos os componentes da aparelhagem multimédia podendo a partir daí controlar todos os dispositivos que respondam.

2.4.3 SDP - *Service Discovery Protocol*

O protocolo *Service Discovery Protocol*, SDP, foi criado especificamente para as redes Bluetooth. Todos os dispositivos que suportem Bluetooth têm de suportar SDP. O SDP dá suporte à interacção entre dois dispositivos Bluetooth que pretendam partilhar um serviço.

A interacção entre um dispositivo que procura um serviço e outro que o oferece é feita em três passos de pergunta e resposta que são o pedido da lista de serviços (ou de um serviço específico) do dispositivo, o pedido dos atributos do serviço e o estabelecimento de uma ligação Bluetooth para a utilização do serviço. No SDP, os serviços que um aparelho Bluetooth pode prestar estão pré-definidos e correspondem a todas as funcionalidades que pode ter um aparelho Bluetooth.

Actualmente estão definidos 25 serviços que vão desde o pedido do uso da porta série até à distribuição de vídeo.

2.5 Sistemas de Percepção do Contexto

[CK00] faz uma vista geral dos projectos de investigação existentes na área da percepção contextual na computação móvel.

2.5.1 PARCTAB

O projecto PARCTAB [WSA+95, Sch95] foi a primeira experiência com aplicações sensíveis ao contexto. Usando um pequeno assistente móvel, denominado PARCTAB (ver Figura 2.3) com capacidade de comunicação por infra-vermelhos foram criadas 4 aplicações sensíveis a uma propriedade de contexto nomeadamente a localização relativa. As aplicações sensíveis à localização eram:

- Selecção de objectos próximos do utilizador, neste caso selecção de impressoras.
- Reconfiguração dos directórios do sistema de ficheiros.
- Selecção de um quadro virtual num programa de desenho cooperativo.
- Mensagens de aviso.

Figura 2.3: O assistente móvel PARCTAB

O sistema funciona em células de localização e comunicação por infra-vermelhos designadas nano-células. Em servidores especializados encontra-se um programa que gere toda a informação relativa a um utilizador, o seu *user agent*. Este programa interage com os programas que se executam no PARCTAB ou em estações de trabalho, os *device agent*. Dado que as aplicações realizadas se baseiam na localização, existe uma representação do espaço onde se movimentam os PARCTAB à qual podem ser submetidas questões; é o *active map*. As interacções entre os *device agents* e o *user agent* são feitas por mensagens unidireccionais não fiáveis ou por invocações remotas usando o T-RPC (ver 2.3.1).

Esta experiência é interessante pela qualidade de desenho do dispositivo que era simples mas com um interface de utilizador e aplicações que faziam um aproveitamento engenhoso e sensato das suas capacidades. É também interessante ver a quantidade de informação de uma única propriedade de contexto, se bem que a propriedade escolhido para este projecto pioneiro tenha sido a localização que é das mais relevantes. Trata-se porém de um trabalho que não levava em linha de conta a ubiquidade do sistema, nem propunha nenhuma arquitectura para extender a sensibilidade ao contexto a outras propriedades do contexto.

2.5.2 *Context Toolkit*

[SDA99] descreve um sistema para que as aplicações se possam abstrair das complexidades de interagir com dispositivos de aquisição de informação de contexto e só tenham de interagir de forma normalizada com os *widgets* de contexto deste sistema.

Os *widgets* de contexto são componentes de software que representam uma propriedade de contexto e são compostos por geradores – componentes que obtêm de um sensor as leitura de uma propriedade de contexto – e interpretes – componentes que convertem a informação do gerador numa propriedade de contexto abstracta.

O sistema está implementado em Java e a comunicação remota é feita por HTTP trocando mensagens em XML para garantir a interoperabilidade com componentes, por ex. sensores, heterogéneos.

As aplicações realizadas são, tal como no caso do PARCTAB, relacionadas com a percepção da localização: um directório indicando se os utilizadores se encontram ou não num dado edifício e um quadro branco com gravador de imagens que foi ampliado com a capacidade de detectar reuniões.

O principal problema do Context Toolkit é não estar integrado com um mecanismo de descoberta de serviços e não permitir a construção dinâmica de contexto, dado que não possui uma linguagem de representação de propriedades, eventos ou situações de contexto.

2.5.3 Solar

Em [CK02], é proposta uma plataforma para suportar aplicações sensíveis ao contexto para computação ubíqua denominada Solar.

Este trabalho foca-se principalmente no problema dos mecanismos e do custo computacional envolvido em processar e resumir a informação proveniente dos sensores que fornecem informações de contexto elementares.

Solar usa uma linguagem de especificação que permite a cada aplicação definir como é que pretende processar a informação de sensores elementares através de um grafo de operadores. Os operadores estão pré-definidos de forma a serem o mais elementares e genéricos possível. Dado que as suas entradas e saídas são constituídas por eventos genéricos podem ser combinados num grafo arbitrário.

Pressupondo que este processamento se revelará pesado, Solar está implementado de forma distribuída, sendo que cada novo grafo submetido por uma aplicação pode ser instanciado num novo nó.

2.6 Coerência de Dados para Dispositivos Móveis

Nesta secção apresento alguns sistemas que se ocuparam de aspectos que penso que serão cruciais para o desenvolvimento de um modelo de gestão de dados replicados com percepção de contexto, nomeadamente, a cópia prévia de dados para um dispositivo móvel (*hoarding*), a gestão de cache, o envio de actualizações de dispositivos móveis para a sede dos dados e a reconciliação entre réplicas divergentes.

2.6.1 CODA

CODA [Sat02] foi um sistema pioneiro na tentativa de solucionar alguns dos problemas da gestão de dados em computação móvel. Foi desenvolvido a partir do sistema de ficheiros AFS (*Andrew File System* da Universidade de Carnegie Mellon) e a sua primeira preocupação era a de permitir a utilizadores com portáteis recolherem os seus dados antes de se desligarem da rede. O sistema CODA evoluiu ao longo de diversas versões e foi incluindo novas funcionalidades.

Naturalmente, o primeiro aspecto com que os autores do sistema tiveram que lidar foi a resolução de conflitos. Dado que o sistema permite a cada utilizador criar uma cache de ficheiros no seu dispositivo móvel, quando os ficheiros são reintegrado existe a possibilidade de existirem conflitos entre o utilizador móvel e os utilizadores que acederam aos dados fixos. Para abordar esse problema, o CODA passou a registar um diário de todas as operações realizadas sobre os dados móveis de forma a poder realizar uma reintegração dos dados mais informada e evitar alguns conflitos que podem ser seriados se os dados e as datas que os motivaram estiverem registados.

Outra observação importante no CODA foi a de que muitas vezes os utilizadores móveis não se encontram desligados mas têm uma ligação com pouca largura de banda. Para aproveitar essas situações o CODA passou a enviar mensagens de invalidação de cache aos utilizadores móveis sempre que o servidor central recebesse uma escrita de um ficheiro e quisesse invalidar a cache dos clientes móveis. Seguindo essa lógica de aproveitar ligações ocasionais com baixa largura de banda, o CODA passou a escalonar assíncrona e tentativamente o envio de quaisquer actualizações para o servidor central. Assim, sempre que o cliente móvel disponha de uma ligação, mesmo que fraca, tentará enviar as actualizações para o servidor central de forma a reduzir a probabilidade de haver uma divergência entre os dados centrais e os dados em cache no cliente móvel.

Finalmente, uma extensão bastante interessante ao CODA foi a adição daquilo que os seus autores designaram cache translucida. Colocados perante o problema

2.6.2 Seer

No projecto Seer [Kue97], procurou-se determinar automaticamente os ficheiros de que um utilizador necessita antes de desligar um dispositivo móvel do seu servidor de ficheiros. Os ficheiros a serem copiados para o dispositivo móvel são determinados em função da distância semântica entre os ficheiros que por sua vez é determinada usando sofisticados algoritmos de localidade temporal. Aqui aposta-se na interpretação do contexto de utilizador, neste caso o acesso a ficheiros, para determinar de que forma os dados de um utilizador estão relacionados e evitando assim que se tenham de dar indicações ao sistema como sucedia no CODA.

2.6.3 Bayou

O Bayou [TTP+95, PST+97] é um repositório transaccional para aplicações cooperativas. Os aspectos mais inovadores do Bayou são o ambiente de trabalho que possibilita, e o seu protocolo de actualização da base de dados. Este sistema tem a particularidade de gerir os conflitos introduzidos numa base de dados por aplicações que se executam concorrentemente em computadores móveis fracamente ligados.

O Bayou pressupõe que todos os dados do sistema estão armazenados numa única base de dados que é replicada em todos os nós servidores do sistema. As aplicações operam sobre a base de dados do computador onde se executam e as diversas réplicas da bases de dados são mantidas coerentes através de trocas esporádicas, ponto-a-ponto entre dois nós, e incrementais de actualizações das réplicas da base de dados. O sistema permite ainda que uma aplicação cliente noutro computador se ligue a uma qualquer das réplicas da base de dados.

O facto da coerência da base de dados do Bayou se manter através das interacções entre quaisquer réplicas permite-lhe suportar qualquer topologia, larguras de banda ou graus de fiabilidade de comunicação.

O protocolo de actualização das réplicas da base de dados baseia-se na troca entre dois computadores dos diários de escritas das respectivas réplicas. Dado que cada escrita está numerada por ordem crescente é fácil concluir quais as porções dos diários a trocar e dado que a troca é feita ordenadamente é facilmente interrompida e retomada tornando assim a actualização incremental.

Adicionalmente, existe um mecanismo para a resolução de conflitos. Cada escrita, quando é propagada para outro nó, é acompanhada de um predicado lógico de verificação de dependência (*dependency check*) que indica se a sua integração na base de dados gera ou não um conflito com outras escritas e de um procedimento (*merge procedure*) para a tentativa de fusão entre esta escrita e outra com a qual

possa estar em conflito. O carácter genérico destes predicados permite implementar diversas semânticas de confirmação de transacções se bem que exija um esforço de programação adicional.

Note-se que um conflito pode só se dar após vários encontros entre nós do sistema, numa fusão de actualizações que não foram realizadas por nenhuma das réplicas que a está a realizar. Ou seja, um nó só tem conhecimento do cancelamento de uma transacção nele realizada quando voltar a ter acesso à base de dados actualizada com todas as actualizações realizadas concorrentemente por outros nós do sistema e constatar que as actualizações por si introduzidas não foram retiradas.

Capítulo 3

Percepção do Contexto e Aplicações

Um dos principais obstáculos à proliferação das aplicações que tiram partido de informação de contexto é o grau de complexidade das interacções entre as referidas aplicações e os mecanismos de aquisição dessa informação.

Comecei por desenvolver um trabalho que explorava o conceito de utilizar informação de contexto para a gestão de dados era [GF02] que é descrito na secção seguinte. Aí é utilizada a informação de contexto que consiste em saber qual o grau de certeza que o utilizador tem na alteração de dados que está a realizar para controlar o controlo de concorrência aplicado aos dados e determinar o âmbito de visibilidade das alterações.

Concebi a arquitectura de um sistema de percepção do contexto [GVF02], a que chamei *Environment Awareness System* (EAS), que é descrita na secção 3.2.

Por último descrevo na secção 3.3 um conjunto de aplicações que demonstram a utilidade da percepção do contexto para a gestão de dados. Neste caso, trata-se de dar resposta a um cenário em que uma pessoa pretende sair do seu escritório com todos os dados necessários para todas os compromissos que tenha até regressar. Os dados são escolhidos com base no assunto dos ficheiros que se encontram no PC do utilizador e os compromissos são conhecidos interagindo com a agenda do utilizador.

Estes trabalhos espelham as duas principais linhas de acção que pretendo prosseguir. Por um lado, continuar a desenvolver um sistema de percepção do contexto que simplifique o desenvolvimento de aplicações sensíveis ao contexto. Por outro lado, investigar os aspectos do contexto mais relevantes para a gestão de dados em computação móvel e ubíqua.

3.1 Controlo de Concorrência para Aplicações Cooperativas Distribuídas

Comecei por explorar a adaptação da gestão de dados com informação de contexto na sequência do trabalho realizado para o projecto PerDiS [FSB+00] . Este projecto pretendia criar uma plataforma de *middleware* para aplicações cooperativas de desenho de engenharia e a minha contribuição inicial consistia em desenvolver os mecanismos transaccionais do sistema.

O tipo de informação de contexto que foi explorada relaciona-se com a visibilidade que os utilizadores das aplicações pretendem que as alterações por eles efectuadas tenham. O grau de exposição, mais local ou mais global, é um modo eficaz de representar o grau de confiança do utilizador nos resultados do seu trabalho. Assim, quanto maior o grau de confiança, maior o âmbito de visibilidade.

No caso explorado em particular, o domínio de dados escolhido foi o dos projectos de construção civil realizados em empresas virtuais[1].

Numa empresa virtual pretende-se ter acesso rápido e consistente aos dados partilhados entre os diferentes participantes. Devido à natureza dos projectos, a colaboração no projecto toma muitas vezes uma forma padronizada. Tipicamente, uma equipa faz o desenho inicial, realizando muitas actualizações durante um período de tempo limitado. Seguidamente, o desenho é passado para outra equipa, possivelmente num local diferente, que avalia um aspecto técnico diferente. Neste tipo de projectos, tende a haver grande localidade espacial e temporal. Também há concorrência real, incluindo conflitos de escrita, que não podem ser ignorados. Por exemplo, é frequente pessoas ou equipas diferentes trabalharem em desenhos alternativos.

Nestas circunstâncias, o modelo ACID[2] não está bem adaptado à área de aplicações de desenho porque pode levar a que sejam abortadas transacções que reflectem um longo período de trabalho do utilizador o que pode ser muito perturbador. Nalguns casos, uma transacção que, de acordo com a semântica ACID, seria abortada pode ser concluída com sucesso se tomarmos em consideração a semântica da aplicação.

Neste trabalho, propus um novo mecanismo para reduzir a necessidade de abortar um conjunto de modificações (não necessariamente realizadas dentro de uma tran-

[1] Uma empresa virtual é a infra-estrutura informática que emerge quando se federam, normalmente de forma parcial, os sistemas de informação de diferentes empresas que participam num consórcio de construção civil

[2] ACID designa as propriedades de uma transacção clássica: Atomicidade, Coerência, Isolamento, Durabilidade

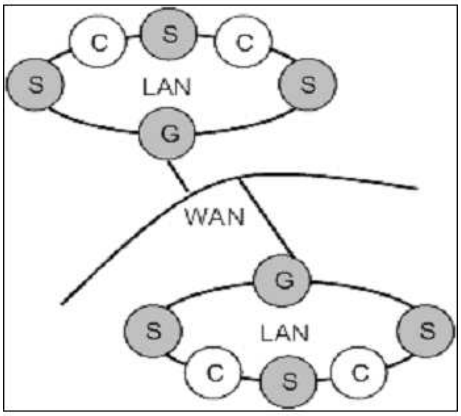

Figura 3.1: Arquitectura LAN/WAN. S designa um servidor, G designa uma *gateway* e C designa uma estação de trabalho.

sacção), a que chamei âmbito de visibilidade. Este mecanismo permite ao utilizador especificar a consistência dos dados partilhados em relação a diferentes conjuntos de nós das redes de computadores de uma empresa virtual.

Este mecanismo mapeia-se facilmente no modo de colaboração numa empresa virtual descrito acima. Por exemplo, um arquitecto que efectua esboços quanto à colocação de salas/gabinetes/etc... num dado edifício, pretende que os dados respectivos sejam visíveis apenas pelos colegas mais próximos de modo a possibilitar a discussão e trabalho cooperativo. Quando as alterações em causa atingem um estado mais estável, então faz sentido que se tornem visíveis para outros participantes na construção do edifício em causa, por exemplo, os engenheiros.

Outro mecanismo desenvolvido foi denominado cópia privada (secção 3.1.3.1) e permite distinguir entre dados que são acedidos para consulta informal e dados cuja consistência é crítica para o trabalho em curso.

3.1.1 Ambiente

Geralmente em empresas virtuais, os membros de uma equipa trabalham numa única rede local, enquanto que o trabalho cooperativo entre equipas é realizado numa WAN e é claramente menos interligado do que aquele realizado dentro de uma equipa.

No PerDiS é feita uma distinção clara entre LANs e WANs (ver Figura 3.1).

Uma LAN é composta de um grupo de nós que podem tomar os papéis de cliente (execução de aplicações), servidor de dados (participante em transacção)

e/ou coordenador de transacções.

Todos os dados tem uma base, que é o nó onde foram criados. A base é parte do esquema de nomes e é usado como ponto inicial para a localização de dados. Para cada LAN, existe um nó especial, denominado a *gateway* da LAN, que gere todas as interacções com nó exteriores a essa LAN.

3.1.2 Arquitectura do PerDiS

Para aliviar os programadores de aplicações do trabalho de lidar com o problema de existirem escritas concorrentes sobre um conjunto de dados, uma aplicação PerDiS é composta por uma sequência de transacções. A aplicação pode ler ou escrever na memória sem interferência de outras aplicações concorrentes.

As transacções são realizadas sobre um repositório multi-versões com garantias ACID e que gera notificações quando surge contenção. Existem dois conjuntos diferentes de mecanismos transaccionais: um para as LANs e outro para as WANs.

Esta distinção influencia a sincronização, o método de confirmação e a *cache* de dados. Consequentemente, o mecanismo transaccional escolhido delimita o tipo de eventos de contenção que são detectados e de que são notificados os utilizadores:

- Dentro de uma LAN, os dados são partilhados usando um mecanismo de memória distribuída e partilhada. Dado que a partilha de memória é feita à escala de uma página de memória, a detecção de contenção também acontece nessa escala.
- À escala WAN, os dados são partilhados trocando e fazendo *cache* de ficheiros (guardados na *gateway*), as actualizações são registadas na LAN local e as transacções são confirmadas enviando as actualizações usando o protocolo de confirmação em duas fases (*two-phase commit* [GR93]. A esta escala existem notificações sempre que são pedidos ficheiros concorrentemente, que são submetidas actualizações de ficheiros partilhados por outros utilizadores e que é confirmada uma transacção de outro utilizador que use dados concorrentemente.

Cada *gateway* tem uma *cache* de ficheiros. Esta *cache* funciona como o interface entre o funcionamento em modo LAN e WAN. Este efeito é obtido redireccionando todos os pedidos de dados exteriores à LAN para a *gateway*.

A *cache* de ficheiros da *gateway* tenta reduzir o tempo de acesso aos ficheiros mantendo-os na LAN ou tentando obtê-los o mais rápido possível na sua origem.

Esta *cache* é uma cache transaccional porque não só gere os ficheiros como também verifica se os ficheiros que serve constituem vistas coerentes com os dados que estão a ser usados pelas transacções em curso na LAN.

As aplicações podem realizar transacções tanto pessimistas como optimistas. As transacções pessimistas restringem a concorrência, trancando os dados de forma conservadora. Isto garante que as transacções não serão abortadas devido a conflitos. Por outro lado, as transacções optimistas não sincronizam os dados, porque pressupõem que haverá pouca contenção entre transacções concorrentes.

Consequentemente, no caso optimista, transacções em conflito podem ter de ser abortadas a menos que sejam usados mecanismos adicionais como o âmbito de visibilidade e a cópia privada.

3.1.3 Controlo de Concorrência com Percepção do Contexto

A noção do âmbito de visibilidade foi motivada pela necessidade de permitir aos utilizadores níveis diferentes de isolamento dos dados consoante o grau de consolidação do seu trabalho.

Por exemplo, considera-se uma ferramenta de CAD[3] cooperativo usada na concepção de um edifício. Suponha-se que a equipa responsável pelas canalizações tem uma equipa a trabalhar na sua rede local, que está a reestruturar a canalização do edifício. Provavelmente, a equipa não quererá que os outros parceiros vejam os passos intermédios desse trabalho enquanto não estiver realizada toda a reestruturação.

Neste caso, o âmbito de visibilidade LAN limitaria a visibilidade dos acessos aos dados à rede LAN dessa equipa. O resultado final só seria visível para toda a empresa virtual quando fosse realizada uma confirmação de transacção à escala WAN.

O âmbito de visibilidade permite que os utilizadores definam para cada transacção quais os âmbitos de visibilidade dos dados de origem e de confirmação da transacção.

São suportados três níveis de visibilidade: WAN, LAN e PC.

Cada nível fornece domínios de consistência diferentes:

- WAN - As actualizações de ficheiros confirmadas ficam disponíveis para todos os nós da empresa virtual. Este nível implica uma modificação efectiva dos dados no seu nó de origem.

[3] *Computer Aided Design*

- LAN - As actualizações confirmadas são ficam visíveis para os outros nós dentro da mesma LAN; não ficam visíveis a nível WAN quando forem usados numa transacção de visibilidade de destino WAN.
- PC - As actualizações são estritamente locais ao nó onde a aplicação que as realizou se executa e só serão visíveis para outros nós quando forem incluídas em transacções de visibilidade de destino LAN ou WAN.

Quando um utilizador requer um ficheiro com visibilidade PC, LAN ou WAN, só é garantido que esse ficheiro será coerente relativamente aos outros ficheiros com essa mesmo âmbito de visibilidade.

3.1.3.1 Cópia Privada

Os dados podem ser adquiridos para leitura ou escrita. Para além disso, o sistema permite que as aplicações requeiram dados num modo designado cópia privada. Esses dados estão consistentes com a transacção no âmbito da qual foram requeridos e podem ser usados para consulta ou trabalho tentativo mas não poderão ser submetidos no fim da transacção.

Por outras palavras, o programador (ou o utilizador da aplicação) podem aceder aos dados da forma que quiser (leitura ou escrita) mas esses dados não farão parte dos conjuntos de leitura ou escrita submetidos com a transacção. Esta funcionalidade é mais um bom exemplo de adaptações que se podem introduzir na gestão de dados em função de informação de contexto, neste caso, contexto de utilizador acerca da intenção do utilizador. O seguinte exemplo ilustra o utilidade do acesso a dados em modo de cópia privada.

Num projecto de CAD, um engenheiro frequentemente tem de consultar diversas plantas para chegar à parte do projecto onde pretende intervir ou para consultar dados acessórios. No entanto, isto não significa necessariamente que o seu trabalho dependa ou seja influenciado pelos dados consultados. De facto, o mais provável é a leitura desses dados não ter um impacto semântico na transacção que se executa. Porém, se os dados tivessem sido adquiridos para a transacção de uma forma tradicional ACID, surgiriam conflitos desnecessários com outros engenheiros que actualizassem os dados. Note-se que os conflitos que se evitam não colocando as cópias privadas nos conjuntos de leitura ou escrita da transacção não são conflitos reais. Se o engenheiro do nosso exemplo decidisse modificar mesmo dados lidos em cópia privada então o trinco teria de ser actualizado para passar a ser um trinco transaccional de escrita.

Por último, note-se que é necessário cuidado para não utilizar esta técnica de forma errada: é fundamental não requerer em modo de cópia privada dados cuja

escrita tem de ser realmente incluída na transacção. Por exemplo, considere-se um utilizador que percorre uma árvore de directórios para escolher o ponto onde vai gravar um ficheiro. É correcto e eficiente, deixar o utilizador percorrer a árvore requerendo os dados em cópia privada, mas, quando a directoria de escrita for seleccionada, todo o caminho desde a raiz da árvore até à directoria seleccionada tem de ser percorrido e trancado de novo para garantir que o ficheiro não é escrito numa directoria que, por exemplo, já não existe.

3.2 Arquitectura de um sistema de percepção do contexto

O *Environment Awareness System* (EAS) [GVF02] é um módulo de *software* que dá às aplicações um mecanismo simples e estruturado de sondarem o contexto físico e computacional do dispositivo onde se executam e doutros nas suas imediações.

Os componentes do EAS são (ver Fig. 3.2):

- Interface de programação de aplicações que permite obter valores para as condições ambientais e requerer notificações assíncronas;
- Registo das notificações assíncronas programadas;
- Plano das acções de sondagem do contexto a realizar;
- Repositório dos eventos passados, escalonados e previstos;
- Modelo de previsão do contexto futuro;
- Vários módulos de percepção do contexto[4];
- Interface de Invocação Remota.

O ambiente de execução do EAS é baseado em eventos. Um **evento** é um par <*atributo,valor*> que representa uma alteração de uma propriedade do contexto. Consequentemente, tem de existir uma relação entre cada sensor acessível ao dispositivo e um atributo que descreve a propriedade que o sensor detecta. Um **atributo** é o nome de uma propriedade do contexto como "Localização Absoluta"ou "Qualidade de Rede". Cada atributo toma valores num domínio pré-determinado, e.g. "Qualidade de Rede"pode ter um dos valores "Inexistente", "Fraca", "Média"ou "Boa".

[4]Environment Perception Modules (EPM)

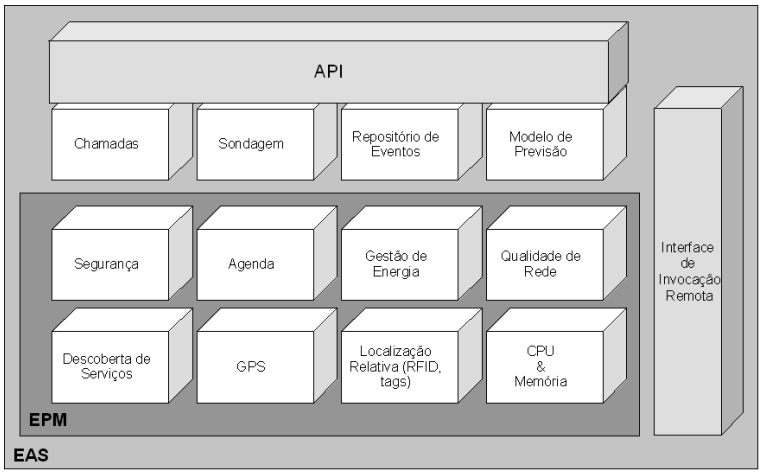

Figura 3.2: Arquitectura do EAS

Um valor descreve o estado de uma propriedade como "443.23N 217.98W"(um possível valor para "Localização Absoluta").

As aplicações podem formular **questões** ou requerer **notificações**. As questões reportam-se ao contexto actual ("'Localização' = 'Lisboa' ?") ou ao contexto passado e retornam à aplicação a indicação se as propriedades questionadas se verificam (ou verificaram caso seja uma questão sobre o contexto passado). As questões podem ter como objectivo a obtenção do valor de uma propriedade de ambiente num dado momento ("Temperatura, 2004/04/09-19:45 ?").

As notificações são um pedido para que ocorra uma notificação quando certas condições se verificarem no futuro. Pode ser associada uma validade às notificações. Relativamente ao alvo, quer as questões quer as notificações podem ser dirigidas ao dispositivo local ou a um dispositivo remoto. No âmbito de uma questão ou notificação, os valores de atributos podem ser especificados como um intervalo.

Outro conceito incluído no EAS é o de **situação**, ou seja situações classificadas. Qualquer situação pode ser classificada relativamente às propriedades do contexto que descreve ou em relação a outras situações. As aplicações que usam o EAS podem submeter situações rotuladas, que são armazenadas e que vão assim formando um dicionário no EAS e podem ser referenciadas em interacções (questões, notificações ou definições de outras situações) posteriores.

Tipicamente, as propriedades de contexto são os elementos de topo de uma hierarquia de situações (ver Figura 3.3) já que é em função dessas propriedades que se define uma situação. A organização hierárquica das propriedades de contexto e dos eventos combinam, por um lado, uma forma simples de manipular informação sobre

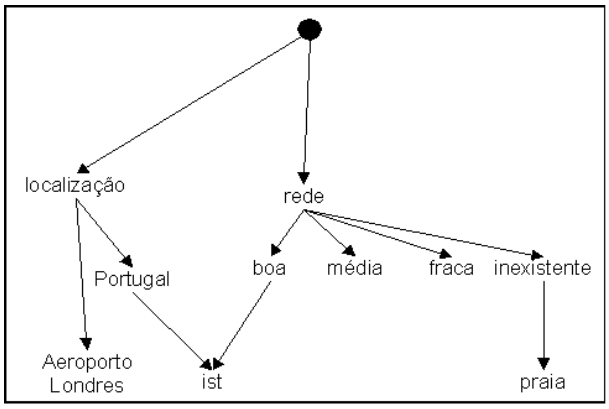

Figura 3.3: Exemplo de uma hierarquia de situações

o hardware e, por outro lado, um mecanismo expressivo para descrever situações às quais as aplicações querem reagir.

3.2.1 Interface de Programação de Aplicações

É usando o interface de programação de aplicações ou API[5] que as aplicações interagem com o EAS, submetendo questões, registando notificações ou manipulando a hierarquia de situações. Os métodos mais relevantes da API do EAS são[6]:

- **CallbackReturnRecord RegisterCallback (Situation, ExpiryDate, GeoReference, MethodReference)**: O método RegisterCallback permite às aplicações registar uma notificação para uma dada situação que é passada como parâmetro e que a aplicação pretende detectar. A situação é expressa como uma lista de pares < atributo, valor>. A notificação inclui uma validade e uma referência geográfica podendo assim restringir a notificação no tempo e no espaço. É também passada uma referência para o método da aplicação a ser chamado quando a situação em causa se verificar.

- **QueryReturnRecord Query (AttributeList, DateRange, GeoReference)**: O método Query é usado para questionar as condições de contexto actuais. A questão é composta por um conjunto de atributos, uma data e uma referência geográfica.

[5] *Application Programming Interface*

[6] A hierarquia de classes usada no EAS foi concebida usando nomes ingleses dado que a arquitectura do sistema foi descrita num artigo submetido a uma conferência internacional

- **RegisterSituation (Label, ParentList, AttributeRange)**: RegisterSituation permite às aplicações adicionar novas situações rotuladas à hierarquia de situações existente. A aplicação fornece o rótulo da situação, os nós que ficam acima da situação na hierarquia e uma lista de intervalos de características de contexto. Para cada característica relevante é fornecido o intervalo dentro do qual se considera que se verifica a situação.

- **SituationList QuerySituation (Label)**: QuerySituation é o método utilizado para navegar no grafo de situações. Este método recebe o rótulo de uma situação e devolve a lista de todos os descendentes dessa situação.

- **RemoveSituation (Label)**: RemoveSituation remove do grafo de situações uma situação em particular identificada pelo seu rótulo.

- **RemoveCallback (CallbackHandle)**: RemoveCallback é utilizado para remover uma notificação antes dela expirar. A notificação a ser removida é identificada por um objecto CallbackHandle, que foi anteriormente incluído no objecto CallbackReturnRecord devolvido à aplicação quando esta invocou RegisterCallback.

Qualquer situação usada para realizar uma questão ou registar uma notificação será convertida numa sequência de valores (ou intervalos) numéricos que descrevem os valores pretendidos para as propriedades de contexto. As questões são avaliadas imediatamente enquanto que as notificações são armazenadas.

3.2.2 Notificações

As notificações feitas pelo EAS de volta para as aplicações são um componente essencial para permitir às aplicações serem notificadas das condições futuras: localização, alimentação, rede de comunicação, etc... As notificações são associadas a alterações nos valores de uma ou mais propriedades de ambiente. As notificações são registadas de acordo com as propriedade de contexto que referem.

Cada vez que o EAS detecta uma alteração de uma propriedade de contexto, realiza uma busca no conjunto de notificações registadas à procura de notificações que tenham de ser desencadeadas por essa alteração. As notificações são armazenadas numa estrutura de dados indexada por propriedade de ambiente e serão referenciáveis por todas as propriedades nelas incluídas. Assim, sempre que ocorra uma alteração numa propriedade, e.g. qualidade da rede de comunicação, todas as notificações que refiram essa propriedade serão revistas para verificar se todas as suas condições se verificam e se, consequentemente, a aplicação que as submeteu deve ser notificada.

Se a situação requerida pela notificação já está prevista para um momento futuro quando a notificação é registada, o EAS devolverá à aplicação o primeiro momento quando a situação ocorrerá.

3.2.3 Sondagem

Periodicamente, o EAS tem de consultar os dispositivos físicos que fornecem os valores para as propriedades de contexto. Muitas das propriedades de contexto são características de dispositivos computacionais que não mudam frequentemente, se é que mudam. Portanto, podem ser armazenadas no EAS e só serem recalculadas quando o hardware local é reconfigurado. Existe um escalonamento dos momentos seguintes em que cada um dos dispositivos disponíveis tem de ser sondado para actualizar as propriedades que lhe dizem respeito. As acções de sondagem podem ser desactivadas nos períodos durante os quais não existem notificações registadas.

3.2.4 Repositório de Eventos

O repositório de eventos armazena todas as alterações de propriedades de contexto que são detectadas pelo EAS. Os eventos são rotulados como eventos observados, calendarizados, previstos ou prováveis. Os eventos observados são o resultado de sondagens realizadas pelos EPMs (ver secção 3.2.7); os eventos previstos são calculados pelo modelo de previsão (ver secção 3.2.5); os eventos calendarizados estão inscritos explicitamente na agenda do utilizador e os eventos prováveis são derivados de informação de agenda. O repositório de eventos está indexado temporalmente de forma que o estado actual de todas as propriedades seja facilmente obtido consultando o último momento em que cada propriedade se alterou.

3.2.5 Modelo de Previsão

Foi incluído um modelo de previsão de modo a prever situações futuras e permitir às aplicações tomar medidas relativamente a condições de contexto futuras. A previsão é realizada com base em eventos passados recorrentes e num modelo de previsão ARIMA [BJ90] para séries temporais de variáveis discretas, que, com base no registo de eventos, detecta padrões temporais para cada uma das propriedades de contexto e situações registadas e insere novos eventos rotulados como "previstos"no repositório de eventos. As séries temporais, que constituem os dados de entrada do modelo de previsão, são geradas analisando o registo de eventos passados e derivando os valores das propriedades para os momentos de amostragem. O número de previsões

e a sua precisão pode ser ajustada às possibilidades de cálculo e armazenamento dos dispositivos.

3.2.6 Interface de Invocação Remota

Um EAS fornece um serviço que pode ser invocado por outros EASs de outros dispositivos. Isto abre a possibilidade de partilha de informação entre um grupo de dispositivos de forma a que, em alguns casos, outros dispositivos confiáveis possam funcionar como extensões de um dispositivo. De uma forma geral, o interface de invocação remota é utilizado para realizar questões que digam respeito a outros dispositivos em torno daquele onde se executa a aplicação que submeteu a questão. Um exemplo interessante desta funcionalidade é , numa situação em que só exista uma rede local, questionar os dispositivos em torno daquele no qual se executa a aplicação para decidir qual terá uma ligação de rede Internet mais cedo e ,portanto, será o ideal para, mais tarde, enviar email ou ficheiros para máquinas exteriores à rede local actual.

3.2.7 Módulos de Percepção do Contexto

Os módulos de percepção do contexto (EPM[7]) são os componentes do EAS que interagem com os sensores do dispositivo computacional e avaliam as propriedades de contexto actuais. Eles representam um obstáculo operacional considerável à construção do EAS devido à grande heterogeneidade dos dispositivos e consequentemente à grande variedade de módulos que será necessário desenvolver. Por exemplo, um programador de aplicações gostaria de avaliar a ocupação de memória simplesmente pedindo ao EAS o valor actual da propriedade "Ocupação de Memória". No entanto, o mecanismo para obter as características da memória varia de uns sistemas operativos para outros. Dada a variabilidade de sensores, gestores de dispositivo e sistemas operativos, conceber EPMs para todos os sensores pode-se revelar uma tarefa muito trabalhosa. Refira-se porém que muitos dos módulos de sistema operativos e sensores, cada vez mais, são acessíveis através de APIs normalizadas (e.g. ACPI [8] para gestão de energia), o que simplifica grandemente o código do EPM. Nos casos em que essas APIs normalizadas não existem, pretendo apostar em desenvolver os EPMs para um sistema operativo de referência de entre os seguintes: Windows CE, Palm OS, Linux, Symbian. De qualquer forma, o EAS tem sempre de ter um modo de se adaptar à ausência de um EPM e de rejeitar questões e notificações que

[7] *Environment Perception Modules*
[8] *Advanced Configuration & Power Interface*

o refiram. Algumas das propriedades de contexto mais relevantes e os mecanismos para as obter são:

Capacidade de Processamento A capacidade de processamento é normalmente avaliada usando uma aplicação de avaliação. Duas das mais reconhecidas são LINPACK [Don01] e HINT [GS95]. Ambas devolvem resultados numéricos que podem eventualmente ser adaptados para uma escala qualitativa. A avaliação da capacidade de processamento é uma operação dispendiosa mas que só raramente tem de ser realizada.

Memória A informação acerca da memória de um dispositivo é importante para avaliar a sua capacidade computacional. Muitas aplicações podem realizar adaptações ao seu funcionamento dependendo da disponibilidade de memória, por exemplo, optando ou não, por manter dados em cache, pré-calcular valores temporários, gerar gráficos de grande detalhe, etc... Em geral, a informação acerca da memória é fácil de obter junto do sistema operativo.

Gestão de Energia A gestão de energia é provavelmente a área em que tem havido mais investigação sobre a adaptabilidade, já que os limites de armazenamento de energia são talvez o maior obstáculo a uma utilização ainda mais difundida dos dispositivos móveis. O EPM a desenvolver no EAS será um interface com o ACPI [acp99].

Localização Absoluta O GPS[9] é o meio mais difundido de localização geográfica. Actualmente, o GPS tem uma precisão que permite localizar um dispositivo computacional com grande precisão e utilizar essa informação para aplicações dependentes da localização e/ou para configurar um dispositivo em função das sua localização.

Localização Relativa Para além dos mecanismos de localização absoluta, há um conjunto crescente de locais onde são utilizados sistemas de localização relativa. Muitos destes sistemas podem ser usados de forma inversa para permitir a um dispositivo saber onde se encontra.

Conectividade A conectividade é uma das características mais variáveis num ambiente móvel. Para além da variabilidade na qualidade de serviço, os computadores estão, hoje em dia, equipados com um conjunto crescente de interfaces de rede (Ethernet, Modem analógico, GSM, IRDA, Bluetooth, etc...) o que levanta a questão não só de saber da qualidade da ligação actual, mas também de saber qual o interface que deve ser escolhido. As métricas que escolhi no EAS foram medir a qualidade de rede do nó em causa até à *gateway* e entre o

[9]Global Positioning System

nó em causa e um conjunto de pontos de interesse para as aplicações de modo a avaliar quer a qualidade geral dos interfaces quer a qualidade de serviço ponto a ponto.

Agenda Electrónica A maior parte dos computadores dispõem de *software* de gestão de dados pessoais, vulgo agenda electrónica. As agendas electrónicas são fontes preciosas de informação acerca dos hábitos dos utilizadores de dispositivos móveis. Mais, à medida que vão sendo acumulados dados acerca das condições de ambiente habituais nos locais frequentados pelo utilizador dos dispositivos, podem-se deduzir as condições de contexto futuras para os momentos em que são marcados eventos na agenda.

Serviços A localização e utilização de serviços disponibilizados por outros computadores é outro dos grandes desafios quando se lida com a variabilidade dos ambientes móveis. O objectivo no EAS é fornecer um conjunto de EPMs de interligação com os principais protocolos de localização de serviços existentes – Jini, UPnP, SDP, SLP (já descritos na secção 2.4) – que poupe às aplicações o trabalho de experimentarem os diferentes protocolos, determinarem quais estão activos e quais os serviços disponibilizados.

Naturalmente, os EPMs apresentados acima são uma fracção dos que se podem imaginar que têm ou virão a ter utilidade. É fácil imaginar outras situações em que outros dados de contexto e outros serviços se tornem relevantes: avaliar a iluminação para gerir baterias carregadas por painéis solares, recorrer a dispositivos de processamento de som em dispositivos próximos, etc... Naturalmente, à medida que a implementação do EAS evoluir será possível incorporar essas novidades e os dados fornecidos pelos EPMs correspondentes no seu modelo de eventos.

3.2.8 Aplicações

Como acabei de exemplificar, à medida que se recolhem os dados elementares acerca do contexto, emerge uma grande quantidade de informação. Colocando questões simples ao EAS, podem-se criar muitas aplicações úteis. Abaixo, descrevo alguns exemplos dessas aplicações. Refira-se ainda que o estudo destas aplicações contribuirá para a compreensão da influência da informação de contexto para a gestão de dados na computação ubíqua e móvel.

3.2.8.1 Agenda ciente da localização

É comum as agendas electrónicas notificarem os seus utilizadores quando chega a data ou hora de um evento. Pretende-se realizar uma extensão desse conceito, criando uma agenda que avise o seu utilizador não só das datas dos eventos mas também da localização de tarefas a realizar. Por exemplo, o utilizador anota que tem de enviar uma carta por correio (o que requer uma ida a uma estação dos correios). Logo, sempre que o utilizador se encontre próximo de uma estação, a sua agenda pode avisá-lo desse facto. Isso requereria adicionar a localização da estação à hierarquia de situações e registar uma notificação. Essa notificação seria invocada na situação específica em que se conjugassem várias propriedades de contexto: o utilizador estar num dado raio da estação de correios, a estação estar aberta, caso fosse possível detectar o modo de deslocamento do dispositivo, o utilizador não se estivesse a deslocar num meio de transporte.

3.2.8.2 Interface de rede ciente das actividades futuras

O objectivo desta aplicação é não só minimizar o custo associado ao uso de um interface de comunicação mas também garantir que os dados são propagados atempadamente para outro dispositivo móvel (ou para um servidor fixo). Consideremos que existe um conjunto de dados que têm de ser transmitidos ou recebidos de outro dispositivo, então o EAS devia ser capaz de notificar o utilizador (ou uma aplicação) do melhor momento para realizar essa transferência. Este tipo de funcionalidade requer não só que o sistema conheça a sua localização actual e as redes de comunicação nela disponíveis mas também quais as ligações que terá (ou prevê que venha a ter no futuro. Poder-se-á então registar uma notificação levando em conta a qualidade da ligação e ainda o prazo limite pretendido para a realização da transferência.

3.2.8.3 Gestão de energia ciente das actividades futuras

A gestão de energia pode ser adaptada tendo em conta a localização futura de um dispositivo. O objectivo neste caso é garantir que, ou o dispositivo tem as suas baterias suficientemente carregadas, ou, caso não tenha, se encontra num local onde as pode recarregar. Mais, a aplicação de gestão de energia deverá ser capaz de notificar o utilizador da necessidade de carregar a bateria em função da sua localização futura. Neste caso particular, o utilizador seria notificado em função de estar num local com possibilidade de carga da bateria, do estado actual da bateria, do seu consumo habitual e dos períodos previstos passados em locais sem rede eléctrica.

3.3 DocManager: uma aplicação de gestão automática de documentos

O objectivo enunciado em 1.2 de criar um modelo para a gestão de dados que inclua informação de contexto deriva da convicção de que essa informação é crucial para que se consiga atingir uma situação ideal em que os utilizadores não se preocupem com nenhum aspecto da gestão de dados e se limitem a utilizar os seus dados independentemente do dispositivo que estejam a aceder e do contexto envolvente. Ou seja, não parece ser possível automatizar a gestão dos dados de um dispositivo móvel sem se conhecer o ambiente que o envolve.

O protótipo que está em desenvolvimento pretende justamente exercitar um cenário em que um utilizador partilha dados entre a sua estação de trabalho e um dispositivo móvel (e.g. um PDA[10]). O dispositivo móvel será levado com um conjunto de dados e depois será reposto e os dados devolvidos à estação de trabalho.

Consideremos que esse utilizador tem de abandonar o seu posto de trabalho para participar numa reunião, por exemplo, sobre orçamento do ano seguinte. Ele gostaria de aceder a todos os seus ficheiros sobre orçamentos sem ter de explicitamente os copiar para o seu PDA no momento de sair da sala. Uma vez na sala de reuniões, gostaria de poder actualizar um ficheiro ou aceder no PC ligado ao projector vídeo à apresentação que tem de fazer. Finalmente, quando regressar ao escritório gostaria que as actualizações (e novos ficheiros) fossem automaticamente copiados para o seu PC.

Muitos sistemas já abordaram a maior parte dos aspectos envolvidos em tornar este cenário uma realidade. O foco deste protótipo e de grande parte desta dissertação é contribuir para que este cenário passe de tecnicamente realizável para completamente transparente para os utilizadores.

A arquitectura concebida para este protótipo tem três níveis(ver Figura 3.4):

Servidor de contexto – Uma implementação simplificada do EAS (ver 3.2) que fornece a informação de contexto estritamente necessária para o funcionamento do protótipo.

MetaOffice – Um serviço tanto local como remoto permitindo procurar, ler e escrever ficheiros Microsoft Office.

DocManager – A aplicação que detecta a saída e chegada do utilizador e executa a recolha e actualização dos ficheiros.

[10]Personal Digital Assistant

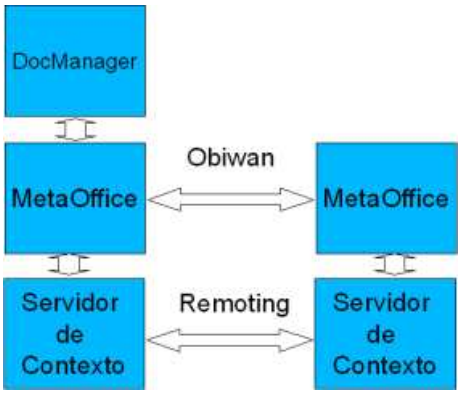

Figura 3.4: Arquitectura genérica do DocManager

Os componentes deste sistema estão a ser desenvolvidos para o sistema operativo Microsoft Windows XP e Microsoft Windows CE em C# usando as funcionalidades de invocação remota de objectos fornecidas pela Microsoft .Net Framework.

3.3.1 Servidor de Contexto

O servidor de contexto é uma versão simplificada do EAS e funciona como um repositório de informação de contexto.

3.3.1.1 Recolha de Informação

O primeiro passo para a obtenção de informação de contexto é identificar as potenciais fontes de informação. O servidor de contexto considera três tipos de fontes de informação de contexto:

- Aplicações locais: As aplicações locais podem ser excelentes fontes de informação de contexto. Em particular, a agenda pessoal dos utilizadores contém informação acerca das actividades e localizações futuras desses utilizadores.
- Sensores locais: Sensores locais podem fornecer elementos como a posição geográfica do dispositivo, a existência de uma rede de comunicação, a carga da bateria do dispositivo, o tempo, etc...
- Informação remota: A informação proveniente de computadores remotos pode ser fornecida por protocolos de gestão de rede ou de localização de serviços, ou directamente por outros servidores de contexto existentes noutros dispositivos.

Os dados recolhidos podem ser questionados por aplicações locais que queiram detectar informação de contexto. Os dados são transformados de formatos variados para o formato textual uniforme. No caso deste cenário, o servidor de contexto recolhe a seguinte informação de contexto:

- Existência ou não de rede de comunicação
- Localização do dispositivo (junto do posto de trabalho do utilizador ou não)
- Servidores de ficheiros usados pelo utilizador

3.3.2 MetaOffice

MetaOffice é um gestor de replicação de documentos. A sua funcionalidade consiste em procurar, recolher e repor os documentos de um dado utilizador de múltiplas fontes. MetaOffice é um serviço que recebe pedidos tanto locais como remotos e, com base na localização dos repositórios de documentos de um utilizador, realiza buscas e transferências de documentos Microsoft Office. O MetaOffice recorre à plataforma Obiwan para invocar outros serviços MetaOffice noutros computadores e para realizar a gestão dos documentos.

3.3.2.1 Obiwan

Obiwan [VGSF00] é uma plataforma de *middleware* vocacionada para a gestão de replicação. Neste protótipo é utilizada como um mecanismo para transferir documentos entre os seus dispositivos originais e os dispositivos móveis onde são transportados.

3.3.3 Aplicação de exemplo: DocManager

A aplicação usada para demonstrar as possibilidade dos dois componentes descritos acima, MetaOffice e servidor de contexto, chama-se DocManager. É uma aplicação simples cujo objectivo é transferir de e para o dispositivo onde se executa todos os documentos Microsoft Office cujo tema coincida com o tema das reuniões seguintes de um utilizador. O DocManager realiza este processo em três passos simples:

- Usar o servidor de contexto para consultar a agenda do utilizador e obter os assuntos das reuniões que serão realizadas longe do seu posto de trabalho.

- Desencadear a transferência de documentos para o dispositivo onde se executa recorrendo às funcionalidades do MetaOffice, momentos antes de o utilizador desligar o dispositivo da rede de comunicação. Este passo levanta uma questão algo complexa, que é a de determinar quando é que se vai dar a perda de ligação com a rede de comunicação. Não havendo nenhuma solução óbvia para esta questão, uma opção é indicar ao utilizador que não é possível retirar o seu PDA sem realizar a sincronização dos dados. Outra hipótese é realizar a sincronização dos dados quando o PDA foi ligado à rede. Neste caso, a instância do MetaOffice que se executa no posto de trabalho do utilizador, teria de invalidar a cache de ficheiros no PDA. Nesse caso, a sincronização teria de ser repetida para que o PDA estivesse carregado com os dados correctos no momento da interrupção da ligação de rede. Enquanto essa sincronização posterior não estiver realizada, o utilizador deve ser notificado que o PDA não deve ser desligado.

- Usar o servidor de contexto para detectar que o dispositivo onde se executa foi ligado de novo à rede de comunicação e desencadear no MetaOffice a cópia para os seus repositórios de origem de quaisquer documentos que estejam no dispositivo.

3.4 Conclusões

Parte dos mecanismos de controlo de concorrência desenvolvidos destinavam-se a serem utilizados em interacção com informações de contexto, neste caso contexto de utilizador, relacionado com a percepção que o utilizador tem do grau de consolidação ou qualidade do seu trabalho. Dado que o contexto psicológico dos utilizadores é dos tipos de contexto mais difíceis de detectar automaticamente, nas aplicações deste projecto as indicações da intenção do utilizador eram dadas explicitamente. No entanto, regressando à definição de contexto (discutida em 2.1), o trabalho em torno destes mecanismos de controlo de concorrência põe em evidência os dois aspectos fundamentais, e indissociáveis, das aplicações sensíveis ou cientes do contexto: percepção e adaptação. É não só necessário conseguir detectar o contexto das aplicações – neste caso, contexto de utilizador – como também ter algo que adaptar no sistema – a sincronização na aquisição dos dados e a visibilidade do resultado do trabalho.

O desenho do EAS (3.2) surge na sequência da constatação que uma aplicação que fosse sensível a muitos dos aspectos de contexto que é interessante percepcionar seria extremamente complexa. O EAS liberta pois as aplicações da tarefa da percepção do contexto, ficando a seu cargo a componente de adaptarem o seu comportamento ou o do sistema onde se executam às alterações de contexto.

Finalmente, o cenário exercitado com a aplicação DocManager, quando concluído pretende vir a pôr em evidência aspectos típicos do problema da gestão de dados em dispositivos móveis e contribuir para a clarificação da relação entre as operações elementares da gestão de dados e a percepção do contexto.

Capítulo 4

Trabalho Futuro

Neste capítulo descrevo as etapas seguintes a desenvolver para a conclusão desta dissertação a partir o trabalho já desenvolvido e que foi descrito no capítulo 3.

4.1 Conclusão da arquitectura EAS

De um ponto de vista estrutural, o avanço mais relevante a conseguir é o refinamento da arquitectura do sistema EAS 3.2 até ao ponto onde estejam claramente definidos os seus componentes e a sua realização seja possível, dado que ainda existem diversas questões fundamentais por responder:

- Necessidade e modo de distribuição do EAS: Alguns componentes da arquitectura prevista para o EAS podem vir a requerer uma capacidade de processamento superior àquela que se pode esperar de um dispositivo móvel. Como tal poderá ser necessário incluir no EAS a possibilidade de o implementar de forma distribuída. Dispositivos que não tenham capacidade de suportar alguns componentes poderiam recorrer aos componentes equivalentes de computadores mais potentes. Por exemplo, um dispositivo móvel poderia submeter o seu registo de actividade a outro computador para que esse execute o modelo de previsão e lhe devolva os resultados dessa execução.

- Periodicidade e escolha dos aspectos a incluir no modelo de previsão: Outro aspecto por avaliar no modelo de previsão do EAS é o detalhe e a periodicidade da sua execução. Relativamente ainda aos custos de execução do algoritmo de previsão pode revelar-se demasiado dispendioso recalcular frequentemente previsões para diferentes propriedades de contexto. A periodicidade desses cálculos e a selecção das variáveis a estudar tem de ser determinada. Uma

hipótese avançada na arquitectura do EAS é só se calcularem previsões para propriedades para as quais existam questões ou notificações. No entanto, este critério é muito ingénuo, pois quando é submetida uma questão já deviam existir previsões que dificilmente poderão ser calculadas naquele momento.

- Mecanismo de troca de informação de contexto: Existem diversos mecanismos possíveis para realizar a troca de informação entre os EAS de computadores diferentes. Esta troca de informação é fundamental para se ampliar a quantidade de informação conhecida num nó específico e para permitir a colocação de questões a dispositivos remotos. Uma hipótese é a utilização do protocolo SNMP [SNM]. Esta abordagem tem a vantagem de ser simples e normalizada o que permitiria obter informação de contexto de um leque mais variado de computadores. A principal desvantagem de se utilizar o SNMP é a impossibilidade de invocar serviços em computadores remotos. De momento, não é evidente a necessidade de o fazer no âmbito do EAS. No entanto, caso se verifique essa necessidade, será necessária escolher um protocolo de invocação remota de objectos para ser utilizada no EAS. Nesse caso, ter-se-á de optar entre manter a troca de informação de contexto no EAS ou migrá-lo para o protocolo de invocãção remota escolhido.

- Armazenamento de eventos: Não está também definido o modelo de dados e o formato em que os eventos do EAS serão armazenados. A tecnologia mais natural para o fazer será, em princípio, uma base de dados relacional, mas o motor de base de dados pode revelar-se demasiado pesado para alguns dispositivos. Alternativamente, poder-se-ão manter em memória estruturas de dados com os eventos e serializar essas estruturas de dados para ficheiros quando se pretender torná-las persistentes. Outra dimensão na qual esta questão tem de ser estudada é na quantidade de dados a manter. Parte da solução deste problema pode ser limitar o espaço de tempo ou propriedades para as quais se mantêm dados ou então escolher uma solução distribuída em que existem nos fixos que, não só realizam as previsões, como propus acima, mas também armazenam os históricos de outros dispositivos com menos capacidade de armazenamento.

- Algoritmo de distância entre dispositivos: A mobilidade dos dispositivos torna a questão da localização geográfica premente e existem muitos trabalhos em torno da localização de dispositivos. No entanto, se se pretender que o EAS constitua um mecanismo genérico de partilha de recursos entre dispositivos, outra medida importante é a da distância física entre dispositivos. Não está ainda definido qual o algoritmo de medida de distância entre dispositivos que será escolhido.

4.2 Modelo de gestão de dados para computação ubíqua

Um dos objectivos desta dissertação é determinar qual o possível contributo da informação de contexto para um modelo de gestão de dados/réplicas. Como se viu em 2.6, existem muitos sistemas com diversos modelos de coerência de dados. Todos esses sistemas realizam os diferentes passos necessários para a gestão de dados replicados (leitura, cache, actualização, reconciliação) usando fundamentalmente três tipos de informação:

- As preferências do utilizador ou referências explícitas entre dados são usadas para seleccionar os dados a recolher.
- A capacidade de armazenamento dos dispositivos é usada para delimitar a quantidade de dados a recolher.
- As datas em que os dados foram modificados são usadas como base do protocolo de actualização (e eventualmente reconciliação).

4.2.1 Relação entre operações elementares de gestão de dados e aspectos de contexto relevantes

A abordagem que proponho seguir para identificar novas relações entre a informação de contexto e as decisões que é necessário tomar para gerir dados replicados consiste em fazer um levantamento das aplicações cientes do contexto, existentes ou desenhadas (e.g. a aplicação DocManager descrita na secção 3.3), e determinar em que momentos realizam essas decisões, quais as informações de contexto que utilizam e que outras informações poderiam ser adicionadas.

A partir desse levantamento, será possível isolar as diferentes operações e informações de contexto relacionadas. Daí, poder-se-á definir uma relação entre as propriedade de contexto e o momento e modo de realização das operações de gestão de dados.

4.3 Desenvolvimento de um modelo de previsão de contexto

Pretendo ainda abordar a criação de um modelo de previsão de condições futuras de contexto. A previsão do contexto futuro é extremamente útil para realizar a

adaptação de aplicações. Por exemplo, se se usar o histórico de localização de um dispositivo para prever a sua localização futura podem-se definir planos para realizar transferências de dados e/ou carga de baterias nos locais onde houver acesso às redes de comunicação e eléctrica. Outro caso em que a previsão é útil é na ocupação de memória. Se se conhecer o padrão de utilização de memória do computador pode-se dimensionar *a priori* a cache de dados.

A previsão pode ser feita usando modelos de previsão temporal como os modelos ARIMA (*Auto-Regressive Integrated Moving Average*) [BJ90] [Tra01]. Estes modelos permitem capturar o comportamento temporal e realizar previsões com base em séries temporais de variáveis discretas. Assim, usando registos periódicos de medidas de características de contexto poder-se-á prever o contexto futuro dos dispositivos computacionais.

Os modelos ARIMA cobrem uma grande variedade de padrões estacionários, não estacionários e periódicos. Analisam a correlação entre os dados observados derivando modelos seguindo um conjunto de passos – construção de um modelo tentativo a partir da identificação de padrões, ajuste dos parâmetros do modelo a partir dos dados observados e finalmente verificação diagnóstica para verificar a adequação do modelo.

Apêndice A

Glossário

ARIMA - *Auto-Regressive Integrated Moving Average*

EAS - *Environment Awareness System*

EPM - *Environment Perception Module*

GPS - *Global Positioning System*

LAN - *Local Area Network*, uma rede local fiável e confiável tipicamente dentro de uma organização ou departamento protegida por uma *firewall*

Middleware - Camada de *software* tipicamente colocada entre as aplicações e o sistema operativo e que esconde às aplicações aspectos da manipulação de dados como a distribuição, a persistência, a concorrência ou a tolerância a faltas. aplicações e o sistema operativo destinada a fornecer às aplicações funcionalidades relacionadas com a distribuição.

PDA - *Personal Digital Assistant*

WAN - *Wide Area Network*, uma rede de grande escala que excede a rede de uma organização.

Bibliografia

[Abo99] Gregory D. Abowd. Software engineering issues for ubiquitous computing. In *Proceedings of the 21st international conference on Software engineering*, pages 75–84. IEEE Computer Society Press, 1999.

[acp99] *Advanced Configuration & Power Interface*. http://www.acpi.info, February 1999.

[BJ90] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[BST99] A. Bakker, M. Van Steen, and A. Tanenbaum. From remote objects to physically distributed objects. In *7th IEEE Workshop on Future Trends of Distributed Computing Systems*, pages 47–52, Cape Town, South Africa, December 1999.

[CCLG02] Ozan Cakmakci, Joelle Coutaz, Kristof Van Laerhoven, and Hans-Werner Gellersen. Context awareness in systems with limited resources, 2002.

[CK00] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

[CK02] Guanling Chen and David Kotz. Solar: An open platform for context-aware mobile applications. In *Proceedings of the First International Conference on Pervasive Computing (Short paper)*, pages 41–47, June 2002. In an informal companion volume of short papers.

[CM00] Paul Castro and Richard Muntz. Managing context for smart spaces. *IEEE Personal Communications*, October 2000.

[Don01] Jack J. Dongarra. Performance of various computers using standard linear equations software. Technical Report Technical Report CS-89-85, Oak Ridge National Laboratory, USA, October 2001.

[FSB+00] Paulo Ferreira, Marc Shapiro, Xavier Blondel, Olivier Fambon, João Garcia, Sytse Kloosterman, Nicolas Richer, Marcus Robert, Fadi Sandakly, George Coulouris, Jean Dollimore, Paulo Guedes, Daniel Hagimont, and Sacha Krakowiak. PerDiS: design, implementation, and use of a PERsistent DIstributed Store. *Recent Advances in Distributed Systems, Springer Verlag LNCS, Eds. S. Krakowiak and S.K. Shrivastava*, 1752, February 2000.

[GF02] João Garcia and Paulo Ferreira. Concurrency Control for Distributed Cooperative Engineering Applications. In *ACM Symposium on Applied Computing*, Madrid, 2002.

[GR93] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Data Management Systems. Morgan Kaufmann, 1993. ISBN 1-55860-190-2.

[GS95] John Gustafson and Quinn Snell. HINT: A new way to measure computer performance. In *Proceedings of the 28th Annual Hawaii International Conference on Systems Sciences*, volume 2, pages 392–401. IEEE Computer Society Press, 1995.

[GVF02] João Garcia, Luís Veiga, and Paulo Ferreira. Environment Awareness: Telling applications about the present and the future. Technical report, INESC ID Lisboa, July 2002.

[GWvB+01] Steven D. Gribble, Matt Welsh, J. Robert von Behren, Eric A. Brewer, David E. Culler, N. Borisov, Steven E. Czerwinski, Ramakrishna Gummadi, Jon R. Hill, Anthony D. Joseph, Randy H. Katz, Z. M. Mao, S. Ross, and Ben Y. Zhao. The ninja architecture for robust internet-scale systems and services. *Computer Networks*, 35(4):473–497, 2001.

[HB01] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.

[HHS+99] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. In *Mobile Computing and Networking*, pages 59–68, 1999.

[HIR01] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Infrastructure for pervasive computing: Challenges. In *GI Jahrestagung (1)*, pages 214–222, 2001.

[HK99] Todd D. Hodes and Randy H. Katz. Composable ad hoc location-based services for heterogeneous mobile clients. *Wireless Networks*, 5(5):411–427, 1999.

[Kue97] Geoffrey H. Kuenning. SEER: PREDICTIVE FILE HOARDING FOR DISCONNECTED MOBILE OPERATION. Technical Report 970015, 20, 1997.

[PMR99] G. Picco, A. Murphy, and G.-C. Roman. Lime: Linda meets mobility. In *21st Int. Conf. On Software Engineering (ICSE-99)*, pages 368–377. ACM Press, May 1999.

[PST+97] Karin Petersen, Mike Spreitzer, Douglas Terry, Marvin Theimer, and Alan Demers. Flexible update propagation for weakly consistent replication. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, pages 288–301, Saint-Malo (France), December 1997. ACM.

[RHR+94] Peter L. Reiher, John S. Heidemann, David Ratner, Gregory Skinner, and Gerald J. Popek. Resolving file conflicts in the ficus file system. In *USENIX Summer*, pages 183–195, 1994.

[Sat02] M. Satyanarayanan. The evolution of coda. *ACM Transactions on Computer Systems (TOCS)*, 20(2):85–124, 2002.

[SAW94] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[Sch95] William Noah Schilit. *A system architecture for context-aware mobile computing*. PhD thesis, Columbia University, 1995.

[SDA99] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *CHI*, pages 434–441, 1999.

[SHHT99] M. Van Steen, F. Hauck, P. Homburg, and A. Tanenbaum. Globe: A wide-area distributed system. *IEEE Concurrency*, 1(7):70–78, March 1999.

[SLP] Service location protocol (SLP) , v.2. RFC 2068.

[SNM] A simple network management protocol (SNMP). RFC 1157.

[Tra01] Nancy Ngoc Tran. Automatic arima time series modeling and forecasting adaptive input/output prefetching, 2001.

[TTP+95] Douglas Terry, Marvin Theimer, Karin Petersen, Alan Demers, Mike Spreitzer, and Carl Hauser. Managing update conflits in bayou, a weakly connected replicated storage system. In *Proc. 15th ACM Symposium on Operating Systems Principles*, pages 172–183, Cooper Mountain Resort, Colorado (USA), December 1995. ACM.

[VGSF00] Luís Veiga, João Garcia, João Silva, and Paulo Ferreira. Distributed object invocation in OBIWAN. In *Proc. of the 9th ACM SIGOPS European Workshop*, Kolding, (Denmark), September 2000.

[Wal98] J. Waldo. JavaSpaces specification 1.0. Technical report, Sun Microsystems, March 1998.

[WMLF98] P. Wyckoff, S. W. McLaughry, T. J. Lehman, and D. A. Ford. TSpaces. *IBM Systems Journal*, 3(37):454–474, 1998.

[WSA+95] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An overview of the PARCTAB ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28–33, Dec 1995.