PROJETO FOR-ALL – COMPUTAÇÃO PARA TODOS, EM TODOS OS LUGARES

Rharon Maia GUEDES; Daniel Fernandes Vilar CARDOSO; Pollyane Carvalho da SILVA; Juliana Maia LEITE; Alisson Vasconcelos BRITO;

CEFET-PB, Av. 1º de Maio,720,João Pessoa-PB, Brasil, +55(83)32083062, rharon.maia@gmail.com
CEFET-PB, Av. 1º de Maio,720,João Pessoa-PB, Brasil, +55(83)32083062, fvcdaniel@gmail.com
CEFET-PB, Av. 1º de Maio,720,João Pessoa-PB, Brasil, +55(83)32083062, pollyanecs@gmail.com
CEFET-PB, Av. 1º de Maio,720,João Pessoa-PB, Brasil, +55(83)32083062, maia_leite@yahoo.com.br
CEFET-PB, Av. 1º de Maio,720,João Pessoa-PB, Brasil, +55(83)32083062, alisson@cefetpb.edu.br

RESUMO

Este projeto denomina-se "For-All – Computação para Todos, em Todos os Lugares", devido à intenção de democratizar o acesso à informação por todas as pessoas e a partir de todos os lugares. O projeto utiliza a tecnologia Sun SPOT (*Sun Small Programmable Object Technology*), que é um dispositivo do tamanho da palma da mão e acionado por software Java, com intuito de formar Redes de Sensores Sem Fio (RSSF) através da tecnologia ZigBee, sendo possível configurar redes adaptáveis capazes de rotear pacotes através de qualquer dispositivo ajustando possíveis falhas. O objetivo está em oferecer os serviços computacionais a qualquer tempo e lugar, com todas suas funcionalidades. Assim um ambiente inteligente é formado, com aplicações "invisíveis" e adaptáveis ao usuário, moldadas à localização do mesmo e às situações do ambiente em que este se encontra, formando assim, um ambiente de Computação Pervasiva e Baseada no Contexto. Para tanto, é necessário que informações sejam coletadas do meio, gerando uma resposta própria e imediata. O trabalho trata-se de um projeto de pesquisa (PIBICT), que foi iniciado no mês de abril sendo capacitado através da doação dos equipamentos pela *Sun Mycrosystens* e do apoio do CEFET-PB. Como metodologia de desenvolvimento, o projeto é formado por uma Arquitetura Orientada a Serviços (SOA) que serve de plataforma para a viabilização de aplicações na área do sensoriamento e monitoramento distribuídos. Na primeira etapa, a camada de Registro de Serviços será desenvolvida. Num segundo momento, aplicações de exemplo serão acrescentadas como estudos de caso. Os experimentos desenvolvidos e apresentados neste trabalho retratam a especificação da plataforma For-All e a implementação de uma aplicação utilizando os serviços básicos de medição de temperatura e luminosidade. Com esses serviços disponíveis, aplicações voltadas ao monitoramento de ambientes para casas inteligentes, sistemas de segurança ou monitoremento de máquinas, por exemplo, podem ser desenvolvidos.

Palavras-chave: Computação Pervasiva, Computação Baseada no Contexto, Sun SPOT, Sensores sem fio.

1. INTRODUÇÃO

O projeto For-All tem a intenção de democratizar o acesso à informação oferecendo os serviços computacionais a qualquer tempo e lugar para todas as pessoas nos mais diversos lugares. Visando à captação rápida de informações sobre mudanças nas características normais de um ambiente, sendo também capaz de tomar decisões em tempo hábil, repassando as decisões tomadas instantaneamente para as pessoas interessadas. Com esses serviços disponíveis, aplicações poderão ser voltadas ao monitoramento de ambientes para casas inteligentes, sistemas de segurança ou monitoramento de máquinas, por exemplo, podem ser desenvolvidos.

O projeto utiliza a tecnologia Sun SPOT, que é um dispositivo acionado por software Java, com intuito de formar Redes de Sensores Sem Fio (RSSF) através da tecnologia ZigBee, sendo possível configurar redes adaptáveis capazes de rotear pacotes através de qualquer dispositivo ajustando possíveis falhas.

The Ninja Architecture for Robust Internet-Scale Systems and Services (O Ninja, Arquitetura para uma escala robusta de sistemas e serviços na Internet), é um projeto similar ao For All em termos de arquitetura, onde procuram habilitar uma inovação ampla em serviços distribuídos na internet. Sua arquitetura consiste de quatro elementos básicos: bases, que são poderosos terminais com ambientes aglomerados usando um software como plataforma que simplificam a construção de serviços; unidades, que são os dispositivos pelos quais os usuários acessam os serviços; active proxies, são elementos de transformação que são usados pelas unidades ou adaptação de serviços específicos; e o path, que é uma abstração embora cada unidade, serviço ou *active proxie* seja compostos (GRIBBLE, 2008).

Nos resultados iniciais, foram implementados apenas dois serviços básicos, os serviços dos níveis de temperatura e de luminosidade. Um terceiro serviço de localização também foi desenvolvido para possibilitar que SPOTs na região seja localizados. Uma pequena aplicação oi desenvolvida onde, através de um computador pessoal, o usuário pode acessar os SPOTs e requisitar um dos três serviços disponíveis.

2. A TECNOLOGIA SUN SPOT

Sun SPOT (Sun Small Programmable Object Technology) é uma plataforma do tamanho da palma da mão e acionada por software Java capaz de feitos incríveis, como equipar dirigíveis e mísseis, controlar mãos e carros robóticos e monitorar um sistema de testes industriais (SUN MICROSYSTEMS, SUN SPOT, 2008).

Baseado em uma CPU ARM de 32 bits e em um rádio de 11 canais de 2,4 GHz, um dispositivo Sun SPOT permite que os programadores compilem aplicativos transdutores sem fio com software Java e usem IDEs familiares, como a IDE NetBeans e o Eclipse, para escrever os códigos. Os desenvolvedores podem escrever seus códigos em Java, carregá-lo no dispositivo do Sun SPOT e executá-lo, podendo desconectá-lo do computador. O dispositivo acionado à bateria inclui um acelerômetro de três eixos, sensores de temperatura e de luminosidade, cinco pinos de entrada e saída de uso geral, oito LEDs de três cores, quatro pinos de saída de alta tensão e uma interface USB. O seu sistema funciona independente de um computador, funcionando sobre o Squawk Virtual Machine (VM), uma pequena máquina virtual na Java 2 Platform Micro Edition (plataforma J2ME), que permite que aplicativos sem fio sejam executados diretamente na CPU sem depender de nenhum sistema operacional (SUN MICROSYSTEMS, SQUAWK, 2008).

Os Sun SPOTs se comunicam entre si usando uma rede ZigBee (ZIG BEE ALLIANCE, 2008), que é uma tecnologia distribuída de redes sem fio que permite uma comunicação confiável, com pouco consumo de energia e baixas taxas de transmissão. Esta tecnologia utiliza mecanismos de conexão e desconexão de um dispositivo em uma rede, identifica e armazena em uma tabela os dispositivos vizinhos, fornecendo identificação e manutenção do encaminhamento. A rede ZigBee especifica como um dispositivo deve se comportar em um dado ambiente.

Desta forma é possível estabelecer redes em malha, conhecidas como redes Mesh, capaz de encaminhar, através de múltiplos saltos, mensagens ao seu destino. Redes Mesh utilizam protocolo de roteamento que fazem várias varreduras das possíveis rotas, considerando a mais rápida e com menos perda de pacotes (RUA, 2006). Esse tipo de rede traz muitas vantagens, tais como custos baixos, simplicidade para o usuário, e robustez, pois ela se adapta às condições da rede, sem intervenção humana.

3. PROJETO FOR-ALL

Na ultima década, os dispositivos computacionais têm estado em crescente desenvolvimento, com isso vem acontecendo a miniaturização de dispositivos e o aumento na capacidade de armazenamento e processamento, fazendo com que essas tecnologias estejam cada vez mais presentes no dia-dia do ser humano. O Projeto For-All explora essa nova área utilizando a Computação Pervasiva (Context-Aware) juntamente com a tecnologia Sun SPOT unida a Zigbee para comunicação, levando em consideração o contexto físico computacional do ambiente e dos usuários para realizar tais processamentos. A facilidade de programação, a arquitetura única, o baixo consumo de energia e a maior segurança foram alguns dos motivos para a utilização da tecnologia Sun SPOT no projeto.

3.1. Uma plataforma pervasiva

À medida que os dispositivos computacionais foram diminuindo novas áreas foram desenvolvidas para englobar tais mudanças. Uma delas é a Computação Context-Aware ou baseada no contexto na qual as tarefas são realizadas de acordo com a situação atual das pessoas ou do ambiente. Esta possibilidade trouxe vários novos termos e conceitos, dentre eles o de "Computação Pervasiva".

3.1.1. O que é Computação Pervasiva?

Podemos entender o conceito de computação pervasiva como sendo a interação entre os dispositivos computacionais e as pessoas, obtida de tal forma a não permitir que seja percebida. Desse modo os dispositivos seriam incorporados ao ambiente para auxiliar atividades humanas e poderiam comunicar-se entre si melhorando o desempenho. Para obter tal resultado a computação pervasiva deve lançar mão das seguintes funcionalidades: sentir, pensar e agir.

Sentir envolve sensores que respondem a estímulos ou sinais como, por exemplo, térmico, luminoso, sonoro, dentre outros. Sua resposta é enviada como um sinal elétrico e pode reagir a qualquer coisa, como uma mudança no ambiente. Pensar representa a tomada de decisão. É onde as informações enviadas pelos sensores são processadas para gerar dados que possam ser usados ou mostrados na aplicação. Já agir são as ações realizadas de acordo com a aplicação e que devem levar em consideração o desempenho, o controle de usuário e o planejamento.

3.1.2. Arquitetura de um Sistema Pervasivo

De forma genérica, um sistema pervasivo baseado em contexto deve possuir a arquitetura mostrada na Figura 1, podendo ser genérica, ou específica para determinada aplicação:

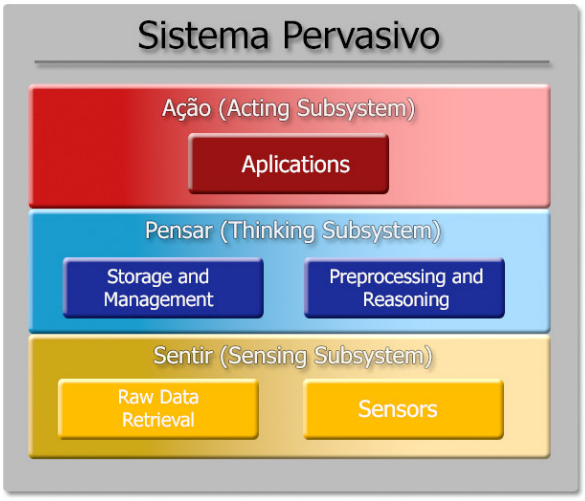

Figura 1 –Arquitetura de um Sistema Pervasivo

3.1.3. Computação Pervasiva e os SunSPOTs:

A Tecnologia Sun SPOT (*Sun Small Programmable Object Technology*) utiliza um dispositivo do tamanho da palma da mão que é acionado por um Software Java, com intuito de formar Redes de Sensores Sem Fio (RSSF). Capacitado com chips ZigBee, tornando possível a configuração de redes adaptáveis capazes de rotear pacotes através de qualquer dispositivo e se adaptar à possíveis falhas em qualquer dispositivo. Aplicações na área de redes de sensores e de computação pervasiva são potencias utilizações da tecnologia Sun SPOT que são exploradas neste trabalho.

A intenção está na oferta dos serviços computacionais a qualquer tempo e de todos os lugares, com todas as funcionalidades desejadas e do modo desejado. Assim um ambiente próprio é requerido, conhecido como ambiente inteligente, com aplicações adaptáveis ao usuário, moldadas a localização do mesmo e as situações do ambiente em que este se encontra. Para tanto, é necessário que informações sejam coletadas do meio, gerando uma resposta própria e imediata. O uso de dispositivos variados e de comunicação distribuída é uma necessidade, como também é necessária uma computação distribuída que forme uma grade pervasiva.

Estas tecnologias unidas provêm diversas possibilidades de aplicações para facilitar o dia-a-dia, onde pessoas vão interagir com vários tipos equipamentos (máquinas, computadores, eletrodomésticos) e também outras pessoas; Sem perceber toda a complexidade por trás dos serviços utilizados, de forma fácil e amigável para o usuário final.

4. UMA PLATAFORMA ORIENTADA A SERVIÇOS

Uma plataforma Orientada a Serviços, é a evolução da computação distribuída com base em pedido/resposta para uma concepção síncrona e assíncrona de aplicações sendo um estilo de arquitetura de software cujo princípio fundamental preconiza que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de serviços (JOSUTTIS, 2007).

Esta é uma promessa de uma nova arquitetura orientada a serviços. Porque cada elemento de um SOA pode ser movido, substituídos, e modificados. Porque cada serviço pode ser montado e remontado em diferentes formas para diferentes propósitos e circunstâncias.

A habilidade de criar processos e compor aplicações a partir desses serviços, combinados com a reutilização e os padrões baseados em interoperabilidade, provêem uma excelente arquitetura necessária para finalidade do projeto – flexibilidade para mudanças (ERICK, 2006).

Na SOA, os programadores criam serviços. As organizações que focalizam os seus esforços de desenvolvimento em torno da criação dos serviços, obterão muitos benefícios. A utilização da SOA nas aplicações, alem de tornar mais fácil a implementação, acarretará em algumas vantagens (KODALI, 2008):

- Aumento da mobilidade. Uma vez que a posição de transparência é uma propriedade das arquiteturas orientadas a serviços, a mobilidade do código transformou-se numa realidade. O aspecto e a dinâmica de conexões de um serviço faz com que o cliente não tenha qualquer preocupação com a localização do serviço. Como conseqüência, uma organização tem maior flexibilidade para mudar os serviços para diferentes máquinas, ou mudar o serviço para um fornecedor externo.

- Maior segurança. A criação de uma camada do serviço, significa que os programadores desenvolvem uma relação adicional, que passa a poder ser utilizada por múltiplas aplicações.

- Prestação de serviços. Os serviços criados serão introduzidos num "catálogo" de serviço. Este "catálogo" é o conjunto de recursos reutilizáveis disponíveis para os clientes e que podem ser utilizados de várias formas. Todos poderão beneficiar de uma maior rapidez no desenvolvimento das aplicações ao recorrerem ao "catálogo" de serviços disponíveis.

- Melhoria da manutenção. A manutenção do software tem como tarefa encontrar e corrigir defeitos no código. Isto é efetuado focalizando o serviço na lógica do negócio. Desta forma, a manutenção é facilitada, pois os programadores detectam e corrigem mais facilmente os erros.

- Melhor escalabilidade. Uma das exigências de uma arquitetura orientada a serviços é a transparência. Para conseguirem esta transparência, as aplicações procuram os serviços no intermediário dos serviços e conectam-se a estes de forma dinâmica em tempo de execução. Esta característica confere escalabilidade ao processo.

A utilização de serviços pode parecer um esforço adicional no início de um projeto, mas estes apresentam vantagens que compensam esse esforço. A capacidade de reutilização, melhor escalabilidade e disponibilidade são fatores importantes que compensam o custo e o esforço extra.

5. ARQUITETURA GERAL

A arquitetura é formada basicamente de Sun SPOTS que possuem serviços nos quais podem ser solicitados a qualquer momento pelo gateway (vide Figura 2). Desta forma, ela é composta por três camadas: a primeira camada é responsável pela coleta de dados (Camada de Sensoriamento) do ambiente que são passados para a segunda camada responsável pela tomada de decisão (Camada de Tomada de Decisão). Na última camada situam-se as aplicações (Camada de Aplicação) que tanto poderão acessar os dados coletados, como configurar a tomada de decisão.

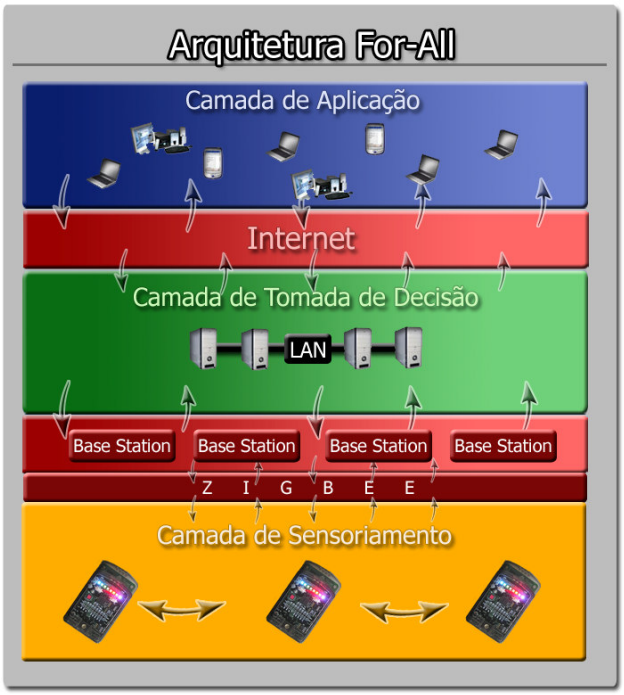

Figura 2 – Arquitetura Geral

Os Sun SPOTs (Camada de Sensoriamento), através da tecnologia Zig-Bee, monitora características relevantes à aplicação (temperatura, movimentação, posição, presença de pessoas etc.) e envia dados pré-processados para a segunda camada que é composta pelas estações base (*base stations*) ligadas a computadores pessoais, que por sua vez, ficam encarregados de tratar e transformá-los em informação para transmitir à terceira camada, responsável por distribuí-las para os usuários da forma desejada.

6. SOFTWARE BASE NOS DISPOSITIVOS

A Figura 3 mostra a arquitetura da aplicação que é executada em cada Sun SPOT.

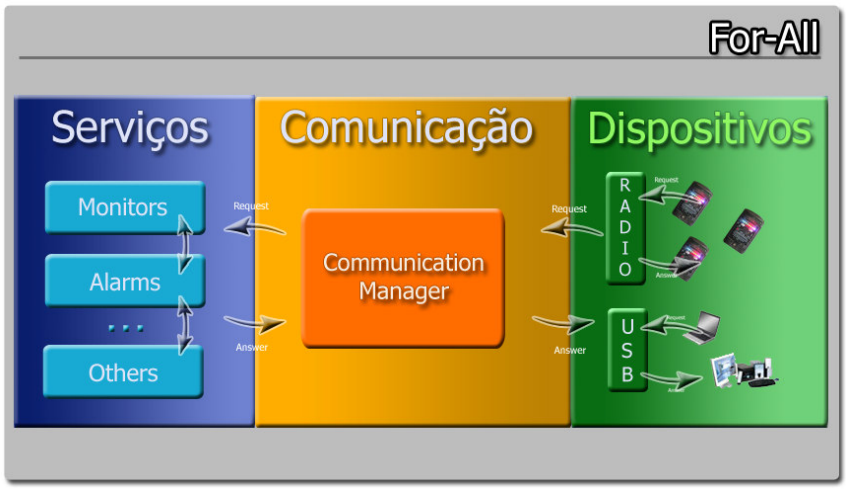

Figura 3 – Arquitetura do Sistema Básico

A camada de serviços separa-os e agrupa-os de acordo com algumas categorias, tais como Alarmes, Monitores e outras. Na camada de Comunicação existe um elemento chamado *Communication Manager* que recebe todas as requisições dos dispositivos (rádio, USB) as quais são analisadas e repassadas ao serviço pertinente. Este funciona como um *thread* individual. Além disso, o *Communication Manager* também é responsável por receber o resultado do serviço e responder pela mesma via da requisição.

7. DESENVOLVIMENTO

A Figura 4 apresenta um Diagrama de Classes simplificado da situação desenvolvimento até o momento. Todos serviços implementados são baseados na mesma interface *ServiceIF*. Ela especifica as principais operações que podem ser realizadas com um serviço. Desse serviço duas variantes foram criadas até o momento, o *AbstractMonitor* e o *AbstractAlarm*. O primeiro especifica como devem ser os monitores, entidades responsáveis por estar sempre monitorando alguma variável através da coleta de dados. A partir dessa classe desenvolvemos monitores de temperatura (*TemperatureMonitor*), luminosidade (*LightMonitor*) e movimento (*MovimentMonitor*). O segundo tipo de serviço implementado é o Alarme, representado pela classe *AbstractAlarm*. Esse serviço é sempre agregado a um monitor. O monitor coleta os dados, e o alarme verifica se eles estão dentro de um limiar aceitável (configurado previamente). Caso os dados coletados passem do limite estabelecido (seja para mais, ou para menos), o alarme dispara um alerta para o mesmo dispositivo que o solicitou. Os serviços são sempre solicitados via Rádio ou USB através da classe *Request*. O alarme lê os dados da requisição dos serviços e utiliza um objeto resposta (*Response*) para gerar o alerta exatamente para o dispositivo solicitante do serviço.

Para envio e recebimento de pacotes, é utilizada a classe Communication Manager, especializada em se comunicar com qualquer dispositivo através das tecnologias disponíveis (Zig Bee ou USB atualmente). Essa classe utiliza outras duas para auxiliar seu trabalho, a *PackageSender* e a *PackageListener*. A primeira recebe objetos *Response* e os envia para o dispositivo gerador da solicitação do serviço (*Request*). Já a *PackageListener* é um thread que monitora todas as conexões físicas do dispositivo esperando requisições. Quando uma requisição por um serviço chega, ela gera um objeto *Request* e repassa-o para o Communication Manager que, por sua vez, consulta o *ServiceRegistry* a procura do serviço solicitado.

Os dispositivos não precisam ter todos os serviços configurados ao mesmo tempo, todo serviço deve ser registrado no *ServiceRegistry* para que o mesmo sempre o encontre quando requisitado pelo *CommunicationManager*. Só assim a requisição é passada para o serviço específico que o executa. Se o serviço gerar uma resposta imediata, a mesma é transformada num objeto *Response* e passada para o *CommunicationManager* que irá utilizar o *PackageSender* para enviá-la ao dispositivo solicitante do serviço.

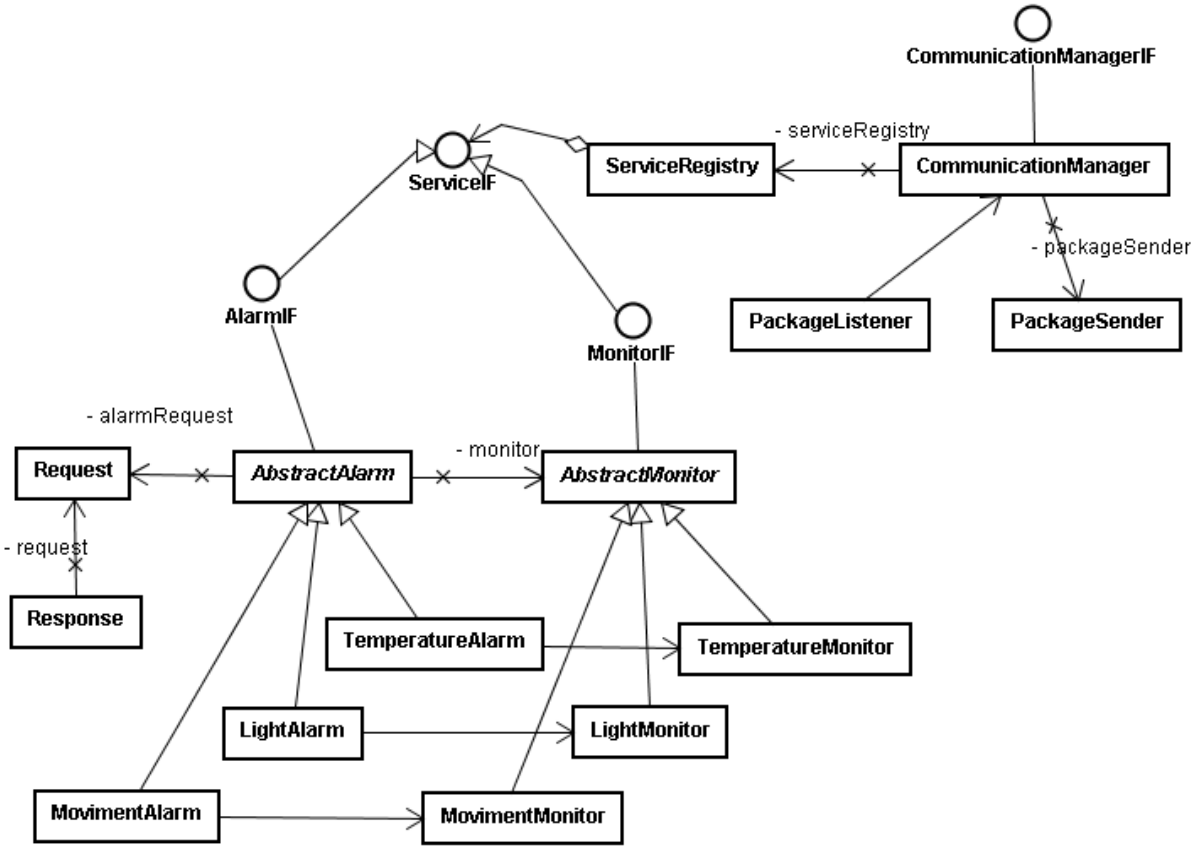

Figura 4 – Diagrama de classes básico

Uma demonstração de como um alarme trabalha pode ser visto na Figura 5. Inicialmente (linha 2) o alarme inicia o monitor respectivo para que o mesmo comece a monitorar a variável desejada (temperatura, por exemplo). De tempos em tempos, o alarme verifica se o valor monitorado (linha 6) está fora da faixa limite (linha 7). Caso isso aconteça, o CommunicationManager é acionado para que um alerta seja enviado (linha 8) ao mesmo dispositivo que solicitou o serviço (representado pelo objeto alarmRequest). O valor ultrapassado (value) também é enviado para que haja uma idéia de quanto o limite estabelecido foi violado.

```
1. public void startAlarm() throws IOException{
2. monitor.startMonitoring();

3. alarm = new Thread(getName()+"Thread"){
4.   public void run(){
5.     while (true){
6.       double value = monitor.getValue();
7.       if(value > getLimit() || value < (-1 * getLimit()) ) {
8.         CommunicationManager.getInstance().sendAlert(alarmRequest,
   value);
9.       }
10.    }
11.};
```

Figura 5 – Código de execução de um alarme genérico (AbstractAlarm)

8. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Os desafios desse projeto são intensos. Vários fatores devem ser considerados. A escabilidade, já que arquiteturas SOA devem ser bem estruturadas prevendo o crescimento no número de equipamentos e de usuários. Outro ponto importante é a segurança. O sistema deve ser ao mesmo tempo disponível a um grande número de pessoas, em todos os lugares a todo tempo, mas deve também ser seguro, protegido contra acessos indevidos. O último ponto que consideramos essencial é a compatibilidade. Em nosso ponto de vista a plataforma For-All, apesar de ter sido iniciada pensando nos dispositivos Sun SPOT, não pode se limitar apenas a esses equipamentos. Devemos prever a inclusão de outros dispositivos de computação e interação. Já está sendo estudada a possibilidade inclusão de outros dispositivos, como computadores de bolso e telefones celulares.

Atualmente apenas os serviços de monitoramento e alarme foram desenvolvidos. Acreditamos que a partir desses, outras variações de serviços poderão ser desenvolvidas. O alarme e o monitor foram implementados e testados em nossos laboratórios apresentando bons resultados. Nosso próximo passo é desenvolver uma aplicação de segurança de edifícios baseado na plataforma For-All utilizando esses serviços implementados.

REFERÊNCIAS

ERICK, U. P., HUGH, T. **Understanding Enterprise SOA,** Manning, 2006.

GRIBBLE, S. D. et. al. **The Ninja Architecture for Robust Internet-Scale Systems and Services**. University of California at Berkeley CA 94720-1776, USA. Disponível em http://www.cs.washington.edu. Acesso em 11 de agosto 2008

JOSUTTIS, M., **SOA in Pratice. The Art of Distributed System Design**. 1. ed. Califórnia: editora O´Reilly, 2007.

KODALI R. R., **What is service-oriented architecture**. Disponível em http://www.javaworld.com. Acesso em 11 de agosto 2008.

MIRANDA, L., **SOA - Arquitectura Orientada a Serviços Disponível** em: http://www.sinfic.pt. Acesso em 11 de agosto de 2008.

RUA D., MARTINS N., REIS P., SOUSA J. P. **Interface USB para recolha de dados de sensores remotos utilizando ZigBee e IEEE 802.15.4**. Departamento de Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, Portugal, 2006.

SUN MICROSYSTEMS, **Sun SPOT Project.** Dispinível em http://www.sunspotworld.com. Acesso em 10 de julho de 2008.

_____. **The Squawk Virtual Machine.** Disponível em http://research.sun.com/projects/squawk. Acesso em 10 de julho de 2008.

ZIGBEE ALLIANCE. **Wireless Control that Simply Works.** Disponível em http://www.zigbee.org. Acesso em 10 de julho de 2008.

7. AGRADECIMENTOS

Agradecemos a Sun Microsystems pela doação dos equipamentos Sun SPOT que viabilizaram a realização deste trabalho.