ARQUITETURA ORIENTADA A SERVIÇOS EM REDES DE SENSORES SEM FIO - MILAN SERVICES

CLAUDIR AFONSO COSTA

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ARQUITETURA ORIENTADA A SERVIÇOS EM REDES DE SENSORES SEM FIO – MLAN SERVICES

CLAUDIR AFONSO COSTA

ORIENTADOR: ADSON FERREIRA DA ROCHA

CO-ORIENTADOR: HERVALDO SAMPAIO DE CARVALHO

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM - 676/17

BRASÍLIA/DF: SETEMBRO – 2017

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ARQUITETURA ORIENTADA A SERVIÇOS EM REDES DE SENSORES SEM FIO-MILAN SERVICES

CLAUDIR AFONSO COSTA

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

Adson Ferreira da Rocha

ADSON FERREIRA DA ROCHA, Dr., ENE/UNB
(ORIENTADOR)

João Luiz Carvalho

JOÃO LUIZ AZEVEDO DE CARVALHO, Dr., ENE/UNB
(EXAMINADOR INTERNO)

FABIANO ARAUJO SOARES, Dr., FGA/UNB
(EXAMINADOR EXTERNO)

Brasília, 25 de setembro de 2017.

FICHA CATALOGRÁFICA

COSTA, CLAUDIR AFONSO
Arquitetura Orientada a Serviços em Redes de Sensores – MiLAN Services.
xiv, 97p., 210 x 297 mm (ENE/FT/UnB, Mestre, Dissertação de Mestrado –
Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Elétrica
1. MiLAN Services 2. SOA
3. Redes de Sensores sem Fio - RSSF 4. Middleware
I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

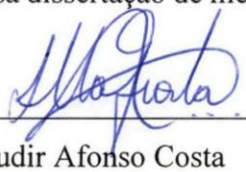
COSTA, CLAUDIR. (2017). Arquitetura Orientada a Serviços em Redes de Sensores – MiLAN Services. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM-676/17, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 99p.

CESSÃO DE DIREITOS

AUTOR: Claudir Afonso Costa.

TÍTULO: Arquitetura Orientada a Serviços em Redes de Sensores sem Fio – MiLAN Services.

GRAU: Mestre ANO: 2017



Claudir Afonso Costa

SQN 410 – Bloco I – Apto 313.

70865-090 Brasília – DF – Brasil.

AGRADECIMENTOS

Agradeço a Deus por ter proporcionado a mim as condições para realizar esse trabalho, não obstante as dificuldades encontradas nesse percurso. Agradeço também ao meu orientador Dr. Adson Rocha, com sua sabedoria, serenidade e paciência. Agradeço ainda ao meu co-orientador Dr. Hervaldo Sampaio de Carvalho, médico e professor de sabedoria e sensibilidade admiráveis, cuja contribuição nesse trabalho foi indispensável.

Dedico esse trabalho à minha família, em especial aos meus pais, Gilson e Aparecida, e também à minha esposa Érica e minha filha Marcela. Todos eles são a base para prosseguir nos estudos e a minha motivação para concluir este trabalho.

RESUMO

ARQUITETURA ORIENTADA A SERVIÇOS EM REDES DE SENSORES SEM FIO – MILAN SERVICES

Autor: Claudir Afonso Costa

Orientador: Adson Ferreira da Rocha

Co-orientador: Hervaldo Sampaio de Carvalho

Programa de Pós-graduação em Engenharia Elétrica

Brasília, setembro de 2017

As redes de sensores sem fio têm progressivamente se tornado um dos grandes objetos de discussão entre as ferramentas que ajudam a unir o mundo físico ao mundo virtual. Os trabalhos nessa linha de pesquisa têm envolvido diversas frentes de trabalho que visam à melhoria do uso dos recursos de rede, do sensoriamento, dos bancos de dados distribuídos, do transporte de dados, do roteamento, das aplicações, e da segurança, entre outros. O MiLAN (Middleware Linking Applications to Network) é um middleware capaz de atender às diversas demandas de aplicações de acesso a redes heterogêneas, adaptando essas redes aos requisitos de Qualidade de Serviços (QoS) das aplicações e otimizando o uso de seus recursos.

Este trabalho apresenta a modelagem e a implementação de uma prova de conceito do MiLAN, bem como de uma camada de serviços para transferência de dados e informações entre a rede de sensores e as aplicações sob a ótica da Arquitetura Orientada a Serviços. A aplicação desenvolvida foi denominada MiLAN Services.

Os resultados obtidos levaram à conclusão de que a MiLAN Services é um middleware capaz de propiciar redes de sensores sem fio com uma redução considerável do esforço de programação para sua utilização pelas aplicações. Além disso, a MiLAN pode auxiliar no aumento da escalabilidade e interoperabilidade de aplicações e de redes de sensores sem fio heterogêneas. Adicionalmente, os resultados mostram que o MiLAN proporciona maior resiliência às redes de sensores sem fio e promove a melhora o uso dos recursos da rede quanto ao uso da energia. Essa característica permite aumentar a vida útil do sistema.

ABSTRACT

SERVICE ORIENTED ARCHITECTURE IN WIRELESS SENSOR NETWORKS – MILAN SERVICES

Autor: Claudir Afonso Costa

Supervisor: Adson Ferreira da Rocha

Co-supervisor: Hervaldo Sampaio de Carvalho

Programa de Pós-graduação em Engenharia Elétrica

Brasília, september of 2017

Wireless sensor networks have progressively become one of the great objects of discussion among the tools that help to unite the physical world to the virtual world. The work in this line of research involves several work fronts, aimed at improving the use of network resources, sensing, distributed databases, data transport, routing, applications, and security, among others. MiLAN (Middleware Linking Applications to Network) is a middleware capable of meeting the diverse demands of heterogeneous network access applications, adapting these networks to the applications' Quality of Service (QoS) requirements and optimizing the use of their resources.

This work presents the modeling and implementation of a proof of concept of the MiLAN, as well as a layer of services for transferring data and information between the applications and the network of sensors from the perspective of the Service Oriented Architecture. This application was named MiLAN Services.

The results obtained led to the conclusion that MiLAN Services is a middleware capable of providing wireless sensor networks with a considerable reduction of the programming effort for its use by the applications. Also, MiLAN can help increase the scalability and interoperability of heterogeneous wireless sensor networks and applications. Moreover, the results show that MiLAN provides greater resilience to wireless sensor networks and promotes better use of network resources for energy use. This feature tends to improve the system´s autonomy.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURAS E ABREVIAÇÕES

CORBA - Common Object Request Broker Architecture
DCOM - Distributed Component Object Model
ECG - Eletrocardiograma
IEEE - Institute of Electrical and Electronics Engineers
JSP - Java Server Pages
MiLAN - Middleware Linking Applications to Network
QoS - Quality of Service
RSSF - Rede de Sensores sem Fio
SDK - Software Development Kit
SOA - Service Oriented Architecture
SOAP - Simple Object Access Protocol
UDDI - Universal Description Discovery and Integration
URL - Uniform Resource Locator
WIST - Web Services Interoperability Technology
WSDL - Web Services Description Language
XML - Extensible Markup Language
XSD - XML Schema Document

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO DO TEMA

O crescente melhoramento em microeletrônica e a proliferação das tecnologias de transmissão sem fio permitiram o desenvolvimento de pequenos dispositivos de baixo consumo de energia, baixo custo e de variadas capacidades de sensoriamento. Esses dispositivos, chamados de nodos sensores, são compostos por sensores, fonte de energia, transceptor e microcontrolador, com processador e memória [1].

As Redes de Sensores Sem Fio (RSSF), compostas por nodos sensores capazes de monitorar diversos parâmetros físicos e de se interagirem de forma colaborativa, podem sem ser utilizadas para as mais diversas aplicações, tais como no controle do trânsito, no auxílio a sistemas agrícolas e no monitoramento remoto de pacientes idosos em suas casas a partir da captura de sinais vitais, de movimento etc.

Por poderem ser constituídas por sensores que utilizem tecnologias distintas de transmissão de dados e em razão das aplicações necessitarem de parâmetros específicos para sua utilização com um respectivo nível de confiabilidade, as redes de sensores sem fio necessitam de mecanismos que as tornem mais integráveis para os diversos usos e mais adaptáveis às necessidades das aplicações com o fito de poderem ser utilizadas na sua plenitude.

1.2 DEFINIÇÃO DO PROBLEMA

Atualmente, não se verifica no rol das tecnologias disponíveis no mercado redes de sensores sem fio de fabricantes e tecnologias distintas capazes de se integrarem de forma colaborativa por meio de protocolo único que objetive, ao mesmo tempo, atender às necessidades das aplicações de forma adaptativa e fornecerem dados e informações sobre a ótica de serviços.

O uso de dados e informações por aplicações externas a uma rede de sensores, em muitos casos, não é trivial e, em geral, exige um grande esforço de programação para isso.

Nas redes heterogêneas, ou seja, aquelas que utilizam tecnologias de comunicação e de gerenciamento diversas a partir de middlewares tradicionais, a integração entre as redes também requer um esforço de programação considerável, quando de sua implementação para utilização por aplicações.

Os middlewares tradicionais para redes de sensores, em regra, não oferecem um gerenciamento dos seus recursos que leve em conta as necessidades das aplicações, otimizando os recursos de rede e, simultaneamente, sendo resilientes.

Nesse sentido, a escolha do conjunto de sensores que melhor atenda aos requisitos de Qualidade de Serviços (QoS) da aplicação ou qual o melhor caminho de transmissão de mensagens na rede visando à economia de consumo de energia da rede, por exemplo, são funcionalidades que o MiLAN propõe, o que minimizaria um grande esforço de desenvolvimento nas aplicações.

O MiLAN propõe um acréscimo da escalabilidade e da resiliência nas redes de sensores heterogêneas e a utilização de seus dados e informações por aplicações também heterogêneas de forma adaptativa. Para maximizar o seu potencial, faz-se necessária a implementação de uma camada que possibilite a interoperabilidade entre as redes de sensores sem fio e aplicações desenvolvidas para as diversas plataformas possíveis.

Com essa abordagem será possível, por exemplo, numa aplicação biomédica, um médico monitorar remotamente um paciente por meio da internet a partir dos dados produzidos pela rede, que podem ser analisados previamente por sistemas externos à rede e apresentados em um nível mais elevado. Tais informações têm o potencial de auxiliar o profissional da medicina para uma tomada de decisão mais efetiva bem como podem promover a melhoria da qualidade de vida e da humanização do paciente, já que é possível reduzir a sua frequência em unidades hospitalares e que as necessidades de intervenção médica podem ser detectadas mais precocemente.

1.3 OBJETIVOS

Objetivo principal: Modelar e implementar uma prova de conceito do MiLAN bem como uma camada de integração que possibilite o consumo dos dados e informações entre a rede de sensores e aplicações heterogêneas por meio de serviços.

Objetivos secundários:

- Implementar uma rede de sensores capaz de fornecer dados e informações da rede expostos como serviços e também consumir serviços externos.
- Implementar uma rede de sensores sem fio que possa operar sobre a ótica do Arquitetura Orientada a Serviços, solução de middleware nomeada de MiLAN Services.

- Implementar as principais funcionalidades do MiLAN quanto ao gerenciamento de rede e atendimento das aplicações conforme seus níveis de prioridade e QoS requeridos.

1.4 ORGANIZAÇÃO DO MANUSCRITO

Este documento está organizado em seis capítulos. Este primeiro Capítulo tem como objetivo contextualizar o leitor sobre assunto abordado realizando uma introdução aos principais temas da área de redes de sensores sem fio de forma bem sucinta e explorados com maiores detalhes nos capítulos adiante. No Capítulo 2 é realizada uma revisão bibliográfica sobre rede de sensores tendo maior foco os temas mais relevantes para o estudo abordado nesse documento. No Capítulo 3 são explorados de forma mais detalhada os fatores que motivaram a realização do estudo e a proposta de alteração no middleware MiLAN, denominado de MiLAN Services. O Capítulo 4 apresenta a implementação realizada. O Capítulo 5 é destinado à exposição dos resultados obtidos com o estudo e também análises desses resultados. No Capítulo 6 são realizadas as conclusões sobre o trabalho e no Capítulo 7 são apresentadas sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

A computação distribuída tem sido objeto de estudo de muitas frentes de pesquisa e tem apresentado grandes desafios desde muito tempo. Inicialmente, a maior dificuldade de se implementar esses tipos de sistemas foi a transferência de dados em redes heterogêneas. Atualmente esses problemas praticamente foram resolvidos e as pesquisas estão voltadas para a resolução de problemas em um nível mais alto como, por exemplo, melhorar a tolerância a falhas por meio de replicação, otimização do acesso a dados por meio de alocação de objetos distribuídos e a abstração da comunicação em alto nível como a orientação a eventos e invocação remota de objetos.

Nesta linha, as redes de sensores sem fio, objeto deste trabalho, estão cada vez mais presentes e imersivas a partir de uma ampla gama de aplicações e demandam estudos em vários campos de pesquisa, tais como, otimização do uso de energia, roteamento, segurança, qualidade de serviços, protocolos de comunicação, dentre outros.

2.1 REDES DE SENSORES SEM FIO

Uma RSSF consiste em um grupo de nodos sensores distribuídos e interconectados por meio de transmissão de rádio, compartilhando dados de sensoriamento e de organização da rede de forma autônoma, podendo capturar várias propriedades químicas, biológicas e físicas como pressão, umidade, temperatura e som.

As RSSF’s podem assistir várias aplicações que vão desde medições no meio ambiente como nível de poluição, volume de chuva e atividade sísmica ao monitoramento dos sinais vitais de um paciente, auxiliando o profissional da medicina em suas decisões remotamente. Um exemplo seria uma RSSF atendendo a uma aplicação biomédica, onde sensores posicionados em um paciente coletam informações como frequência cardíaca, eletrocardiograma, temperatura, pressão e nível de glicemia, tomam decisões no sentido de manter a rede ativa e enviam informações ao médico sobre as condições do paciente.

Uma RSSF pode conter milhares de sensores e serem homogêneas (comporta por uma única tecnologia de gerenciamento e de comunicação) ou heterogêneas ou (compota por tecnologias de gerenciamento e de comunicação distintas), compartilhando dados de forma colaborativa para a sua organização na coleta e envio de informações para autoconfiguração da rede visando escalabilidade, robustez e longa vida de operação [2].

A integração entre as RSSF's e o mundo físico tem sido alvo de inúmeras frentes de pesquisa no sentido de viabilizar soluções que permitam integrar as plataformas computacionais existentes a redes que possam trazer informações do mundo real. Aplicações diversas são beneficiadas com o avanço tecnológico dessas redes e novas aplicações surgem constantemente. Porém, grandes desafios ainda fazem parte dos ambientes de pesquisa no que diz respeito, por exemplo, à redução do consumo de energia da rede, já que energia é um ponto vital para um sistema que depende, na maioria dos casos, de fontes de energia limitadas. Os middlewares, que são pedaços de software que se situam entre as aplicações e a rede, têm sido alvo de pesquisas para auxiliarem não só numa otimização do consumo de energia do sistema como também para controlar e dar maior escalabilidade e flexibilidade às RSSF's.

2.2 MIDDLEWARE PARA REDES DE SENSORES

As RSSF's tendem a se tornarem cada vez mais heterogêneas e a atenderem a um grande número de aplicações, requerendo para isso, em geral, um grande esforço de programação em baixo nível. Dessa forma, o desenvolvimento e implementação de uma arquitetura e protocolos adequados são cruciais para que essa tecnologia se torne comercialmente interessante.

Sistemas operacionais desenvolvidos para nodos sensores, como o TinyOS [3], fornecem funcionalidades gerais para sensores, mas não oferecem um gerenciamento de rede que se oriente às necessidades das aplicações.

A maioria dos middlewares existentes não fazem uma integração entre a configuração no nível de rede e as demandas da aplicação. O middleware *LIME* [4**]** foi projetado para a coordenação de redes *ad hoc* móveis, o seu foco está no modelo da aplicação apresentada pelo programador. As redes *ad hoc* são como um sistema autônomo de componentes móveis (inclusive servindo como roteadores), conectados por links sem fio. Contrasta com redes celulares de um único salto que possuem uma estação de rádio base que funciona como ponto de acesso [5]. Especificamente, o *LIME* fornece uma tupla (*tuplaspace*) *Linda* [6**]** cujo conteúdo se altera conforme a conectividade entre os componentes da rede se alteram. Operações com tuplas se baseiam basicamente na conectividade não havendo nenhuma tentativa no sentido de detectar as propriedades da rede para beneficiar a aplicação. Similarmente, o *Corba* [7**]** e outros middlewares assumem que as conexões são estáticas, não sendo aplicáveis em redes com mobilidade. O ambiente de programação *FarGo* [8] permite que os componentes da rede sejam realocados, respondendo às variações das condições da rede, como disponibilidade de largura

de banda ou a sua confiabilidade. Porém, os componentes do *FarGo* não afetam as propriedades da rede.

O MiLAN, Middleware Linking Applications and Network [12], é um middleware que adapta os componentes da rede às necessidades da aplicação com vistas à otimizar o uso desses recursos. Contrastando os exemplos dados até aqui, as aplicações consideradas no escopo do MiLAN não só devem adaptar-se às alterações da rede como também realizarem modificações na rede para gerenciar o consumo de energia e o desempenho da aplicação.

O MiLAN verifica a dinamicidade e as necessidades das aplicações e utiliza essas informações para configurar a rede, o que auxilia no crescimento do desenvolvimento de aplicações colaborativas. Maiores detalhes do funcionamento do MiLAN são descritos no capítulo 2.

2.3 ARQUITETURA ORIENTADA A SERVIÇOS

A Arquitetura Orientada a Serviços (SOA) é um paradigma para a organização e utilização de serviços distribuídos podendo estar em diferentes domínios e plataformas e pode ser um meio de organização das soluções para promover o reuso, crescimento e interoperabilidade [9].

Dentre os objetivos do conceito SOA destaca-se a interoperabilidade entre diferentes sistemas, plataformas e linguagens de programação. As RSSF's aplicadas no contexto de SOA podem ser acessadas por um grande número de aplicações espalhadas por todo mundo por meio da internet. Assim, cada nodo da RSSF pode ser visto por redes de diferentes domínios como um serviço [10] e [11].

O conceito de SOA pode ser implementado por meio de um ou vários tipos de protocolos e tecnologias como, por exemplo, o Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM) e Simple Object Access Protocol (SOAP). Serviços independentes podem ser "chamados" para realizar uma tarefa de uma forma padronizada, sem que a aplicação requisitante conheça como a tarefa será realizada. No SOA, serviços são descritos, descobertos e invocados de plataformas heterogêneas utilizando padrões como Extensible Markup Language (XML) e SOAP, utilizando-se, por exemplo, tecnologias como o *Web Service*.

Aplicações podem realizar solicitações a serviços fornecidos pelas RSSF's, que estão relacionados a uma tarefa a ser executada na rede, como a coleta de informações por meio de

execução de uma query. Os dados coletados então são processados conforme um plano de execução estabelecido na rede e enviados para a aplicação sem a necessidade de que o usuário tenha conhecimento da configuração ou estrutura da rede, ficando o seu esforço voltado para o conhecimento e o uso dos serviços disponíveis, que pode ser obtido por meio de uma Web Services Description Language (WSDL), que descreve quais e como devem ser utilizados os serviços disponíveis em uma plataforma.

As RRSF's tendem a se tornarem cada vez mais heterogêneas, a exigirem maior nível de segurança e a atenderem a um grande número de aplicações, o que atualmente resulta, em geral, em um grande esforço de programação em baixo nível.

Nas diversas aplicações que utilizam redes de sensores a aplicação do conceito de SOA busca prover um acréscimo na escalabilidade, na facilidade de integração e de implementação. Além disso, a aplicação de SOA nas RSSF's para essas aplicações permite que informações provenientes da rede possam ser agregadas com outros serviços de redes de computação convencionais e sejam transmitidas via internet.

Um meio de organizar os serviços oferecidos pelas RSSF's e interligá-los a aplicações para o compartilhamento de informações e a implementação do paradigma SOA.

Os serviços que podem ser oferecidos pelas RSSF's são um grande potencial para auxiliar a inúmeras aplicações, em especial às aplicações biomédicas, na integração com soluções para a tomada de decisões onde informações de serviços do mundo da computação convencional são agregadas a informações provenientes do mundo físico.

Um médico, por exemplo, pode, por meio de uma planilha eletrônica em seu consultório, obter informações atualizadas de um de seus pacientes, que possui uma RSSF em seu corpo enviando informações pela Web. Esse tipo de aplicação, quando implementada em um modelo de SOA, tem o esforço bastante minimizado na integração com aplicações e protocolos já existentes.

O uso de SOA nesse contexto revela a necessidade de se prover serviços para plataformas variadas e com facilidade de implementação, alcançados por meio de padronização dos protocolos de comunicação, como o SOAP.

Os principais motivos para a arquitetura baseada em SOA nas RSSF's visam a facilitar o gerenciamento dos serviços providos, proporcionando maior escalabilidade e facilitando a sua utilização pelas aplicações distribuídas.

2.4 MiLAN

O MiLAN [12] é um middleware para rede de sensores que tem a capacidade de se adaptar às variações dos componentes da rede de forma a atender as necessidades da aplicação e ao mesmo tempo otimizar o uso dos recursos da rede, sem que a aplicação tenha que implementar alguma forma de gerenciamento da rede.

A proposta do MiLAN no contexto de sistemas distribuídos é a de contribuir para as plataformas de middlewares que se situam entre o sistema operacional e a aplicação, abstraindo funcionalidades de baixo nível como a conectividade da rede e provendo uma interface abstrata de coordenação para os programadores das aplicações.

Em sistemas distribuídos, o funcionamento da rede tem impacto direto no desempenho da aplicação. Isso acontece principalmente quando as redes são sem fio ou são redes móveis ou, ainda, são limitadas pelo tempo de vida da bateria. Nesses casos, a conectividade da rede é alterada com o tempo e deve ser monitorada e gerenciada da melhor maneira, conforme as necessidades da aplicação. Entretanto, múltiplas aplicações utilizando a rede podem ter diferentes critérios de como utilizar os recursos disponíveis. Por exemplo, considere que existam duas aplicações, uma voltada para sistemas de segurança e outra para entretenimento. Ambas consomem os recursos da rede sem fio que fornece áudio e vídeo de baixa definição por meio de dois recursos distintos. Supondo que um terceiro nó foi adicionado à rede e proverá vídeo de alta definição, que a rede não suporta a transmissão simultânea dos dados provenientes das três fontes e que a aplicação de entretenimento utilizaria o vídeo de baixa definição e a aplicação de segurança o vídeo de alta definição, seriam gerados problemas que não poderiam ser facilmente resolvidos utilizando-se middlewares convencionais. Um desses problemas seria como resolver o conflito gerado por se ter aplicações com necessidades diferentes, ou seja, que exigem da rede o acesso diferenciado aos seus recursos.

O MiLAN recebe da aplicação um conjunto de especificações de desempenho e de como as diferentes aplicações devem interagir com a rede de sensores. O MiLAN também monitora a disponibilidade dos componentes da rede, sobretudo o consumo de energia e da largura de

banda. Combinando essas informações, o MiLAN adapta a rede para otimizar o uso dos recursos balanceando as necessidades da aplicação aos custos de desempenho.

2.5 O MiLAN NAS APLICAÇÕES BIOMÉDICAS

Como já mencionado anteriormente, as redes de sensores não se restringem a um grupo de aplicações, mas se estende desde aplicações militares a aplicações biomédicas. Nesta dissertação, foi escolhida para a implementação uma aplicação biomédica de monitoramento cardíaco.

Por meio de uma casa inteligente, por exemplo, médicos podem interagir com os dispositivos projetados para assistir os pacientes diariamente e monitorar e controlar as condições médicas remotamente. No nível mais baixo da aplicação, a casa seria composta por sensores, por exemplo, câmeras, microfones, termômetros; atuadores, como alto falantes, displays e dispositivos médicos, e ainda unidades computacionais, que podem estar embutidas nos sensores, numa unidade digital de assistência ao paciente, ou em computadores de mesa ou notebooks. As unidades computacionais executariam aplicações que capturam dados dos sensores como entrada do sistema e produzem saídas que controlam atuadores e/ou outras aplicações.

Com o intuito de dar um melhor entendimento sobre o funcionamento do MiLAN, é apresentado a seguir um exemplo de casa inteligente que a Universidade de Rochester implementa realizando pesquisas com o objetivo de permitir que pessoas possam monitorar a sua saúde de si mesmas por meio da chamada Start Medical Home[13]. São aquisitados nessa aplicação dados de frequência respiratória, frequência cardíaca, pressão sanguínea, oximetria de pulso, nível de oxigênio e diagrama de Eletrocardiograma (ECG).

A frequência respiratória pode ser conceituada como o número de ciclos respiratórios de uma pessoa que completa num período específico de tempo, expressa comumente em respirações por minuto.

A frequência cardíaca, medida em qualquer região do corpo onde pode ser detectada a pulsação arterial, representa o ciclo cardíaco em um período de tempo, normalmente em batimentos por minuto.

A expressão pressão arterial ou pressão sanguínea representa a pressão exercida pelo sangue contra a parede das artérias e é medida em dois tipos: sistólica e diastólica.

A oximetria de pulso é uma forma de medir quanto oxigênio flui pelo sangue. Um dispositivo chamado oxímetro de pulso pode aferir o nível de oxigênio no sangue, chamado de nível de saturação de oxigênio.

Um eletrocardiograma (ECG) é uma representação da atividade dos músculos cardíacos de acordo com os seus movimentos a partir da variação dos potenciais elétricos gerados pela atividade elétrica do coração.

A fim de determinar como melhor servir a aplicação, o MiLAN deve saber as variáveis de interesse da aplicação, os níveis de QoS requeridos para cada variável e o nível de QoS de cada variável que cada sensor pode fornecer. É válido ressaltar que essas as variáveis de interesse da aplicação e seus respectivos níveis de QoS podem variar conforme o estado atual da aplicação. Durante a inicialização da aplicação essas informações são enviadas pela aplicação ao MiLAN por meio do grafo de variáveis requisitadas baseadas em estado e do grafo de QoS dos sensores. Exemplos desses grafos são mostrados na figura 2.5.1 e na figura 2.5.2.

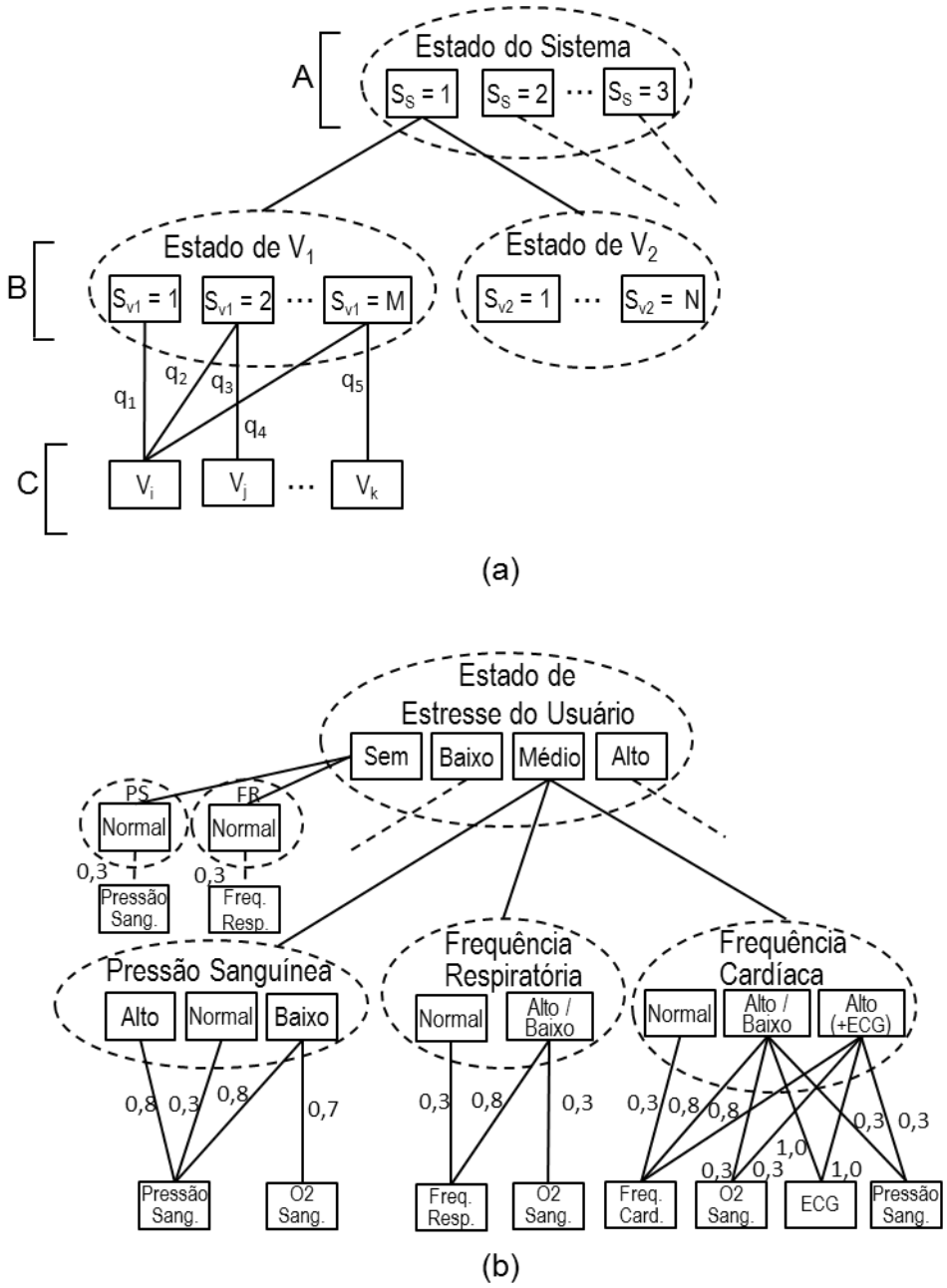


Figura 2.5.1 Grafos dos requisitos de QoS baseados nos estados da aplicação e das variáveis.

Fonte: [12], traduzida e com adaptações.

O grafo representado pela figura 2.5.1a mostra os requisitos de QoS para cada variável de interesse baseada no estado atual do sistema e das variáveis de interesse da aplicação, sendo esses estados baseados no dado de análise da aplicação previamente recebido. Para um estado particular (uma combinação do estado do sistema – nível A e um estado da variável – nível B), o grafo de variáveis requisitadas baseadas em estado define o QoS para cada variável. Em função das variáveis (nível C) poderem ser formadas por múltiplas variáveis de estado (nível

B), o MiLAN deve extrair o QoS máximo para cada variável selecionada para satisfazer os requisitos de todas as variáveis de estado.

A figura 2.5.1b é um exemplo de grafo das variáveis requisitadas para uma aplicação de monitoramento da saúde de um paciente. Essa aplicação tem dois estados, o estado do sistema, que representa o nível de estresse do paciente, e os múltiplos estados para cada variável que pode ser monitorada. O grafo das variáveis requisitadas especifica para o MiLAN o QoS mínimo aceitável para cada variável (por exemplo, pressão sanguínea, freqüência respiratória, etc.) baseando-se no estado atual do paciente. Por exemplo, a figura mostra que quando um paciente está em estresse mínimo e a pressão sanguínea é baixa, o nível de oxigênio no sangue deve ser monitorado com um nível de qualidade 0,7 e a pressão sanguínea deve ser monitorada com um nível de qualidade 0,8.

Para uma dada aplicação, o nível de QoS para cada variável pode ser satisfeito usando dados de um ou mais sensores. Essa aplicação informa ao MiLAN as especificações por meio do grafo de QoS dos sensores, mostrado na figura 2.5.2a. Quando múltiplos sensores são combinados para fornecer uma variável com um melhor nível de QoS é como se esse conjunto formasse um único "sensor virtual". A figura 2.5.2b mostra o grafo de QoS dos sensores para uma aplicação de monitoramento da saúde de um paciente. Esse grafo ilustra as variáveis a serem monitoradas em uma determinada condição do paciente e indica os sensores que podem fornecer as medições dessas variáveis a um dado QoS. Cada linha entre o sensor (ou sensor virtual) e a variável possui um nível de QoS que o sensor (ou sensor virtual) pode fornecer para medir aquela variável. Por exemplo, usando dados do sensor de pressão sanguínea, a frequência cardíaca pode ser determinada com um nível de qualidade 0,7 (que pode ser a acurácia da medição), mas combinando essa medição à de um sensor de fluxo sanguíneo o nível de qualidade seria incrementado para 1,0.

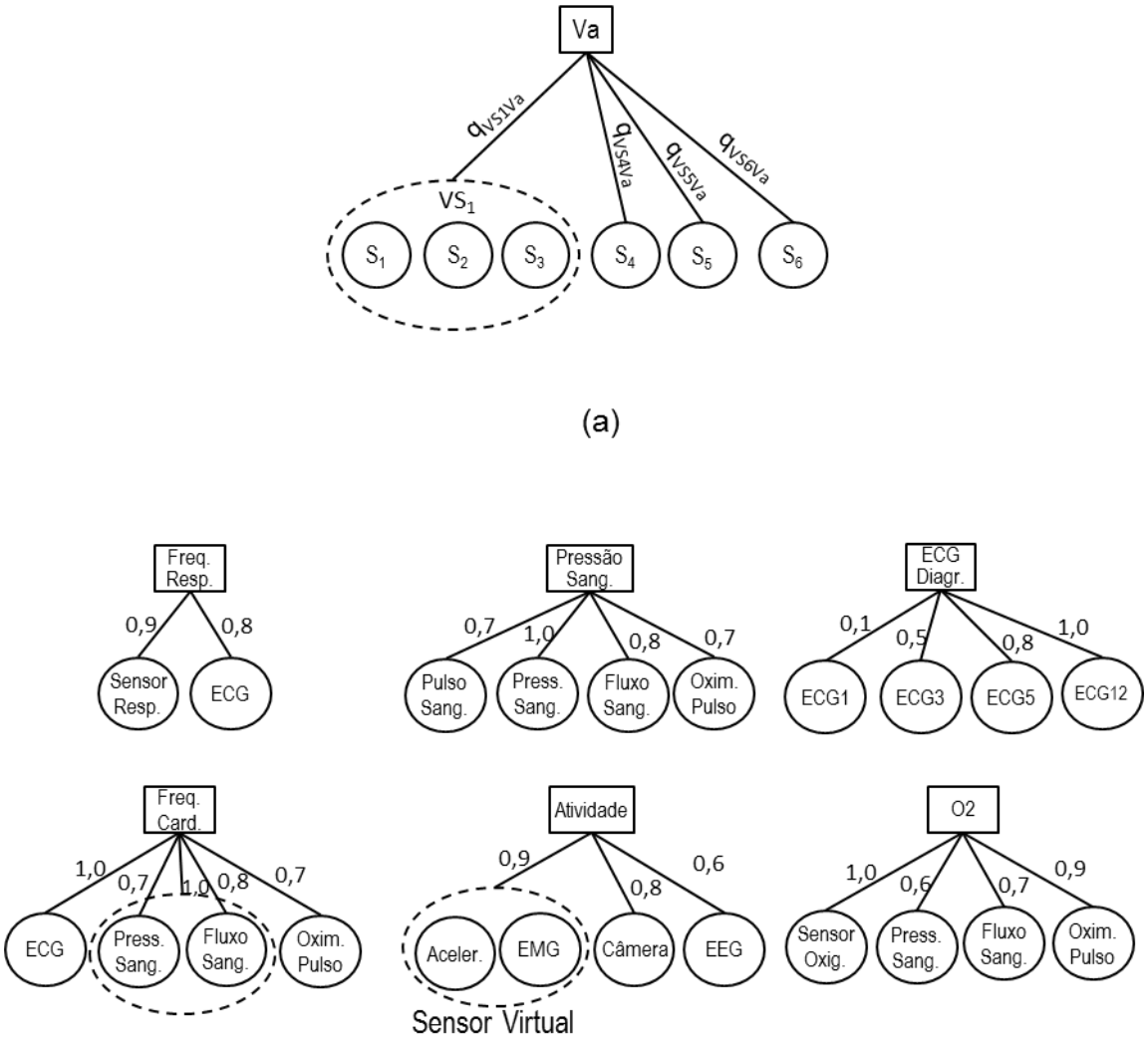


Figura 2.5.2 Combinação de sensores para o fornecimento de uma variável.

Fonte: [12], traduzida e com adaptações.

Para o exemplo dado, os conjuntos de sensores que satisfazem às necessidades da aplicação podem ser verificados na tabela 2.5.1.

Tabela 2.5.1 Conjuntos de sensores que atendem aos requisitos de uma aplicação de monitoramento cardíaco. Fonte: [13], traduzida e com adaptações.

Conjunto	Sensores
1	Pressão sanguínea, frequência respiratória
2	Pressão sanguínea, ECG (3 derivações)
3	Oximetria de pulso, pressão sanguínea, ECG (1 derivação), frequência respiratória
4	Oximetria de pulso, pressão sanguínea, ECG (3 derivações)
5	Nível de oxigênio, pressão sanguínea, ECG (1 derivação), frequência respiratória
6	Nível de oxigênio, pressão sanguínea, ECG (3 derivações)

2.6 ESTRATÉGIA DO MILAN

A estratégia principal de operação do MiLAN é monitorar as condições da rede e otimizar a configuração da rede de acordo com o comportamento da aplicação com base nas especificações fornecidas pela aplicação,

Quando uma aplicação solicita ao MiLAN dados da rede, ela envia um grafo contendo a combinação de sensores que são capazes de fornecer os dados conforme um QoS requerido. O MiLAN então configura a rede selecionando os sensores que fornecerão os dados à aplicação. Ainda, é possível que as aplicações utilizem múltiplas redes de sensores heterogêneas. Por meio de plug-ins, os comandos do MiLAN são convertidos para os protocolos específicos de cada rede.

Diferentemente dos middlewares tradicionais, que não entram nos limites da aplicação e do sistema operacional, o MiLAN controla a rede de sensores possuindo uma arquitetura que se estende em uma pilha de protocolos, conforme ilustrado na figura 2.6.1.

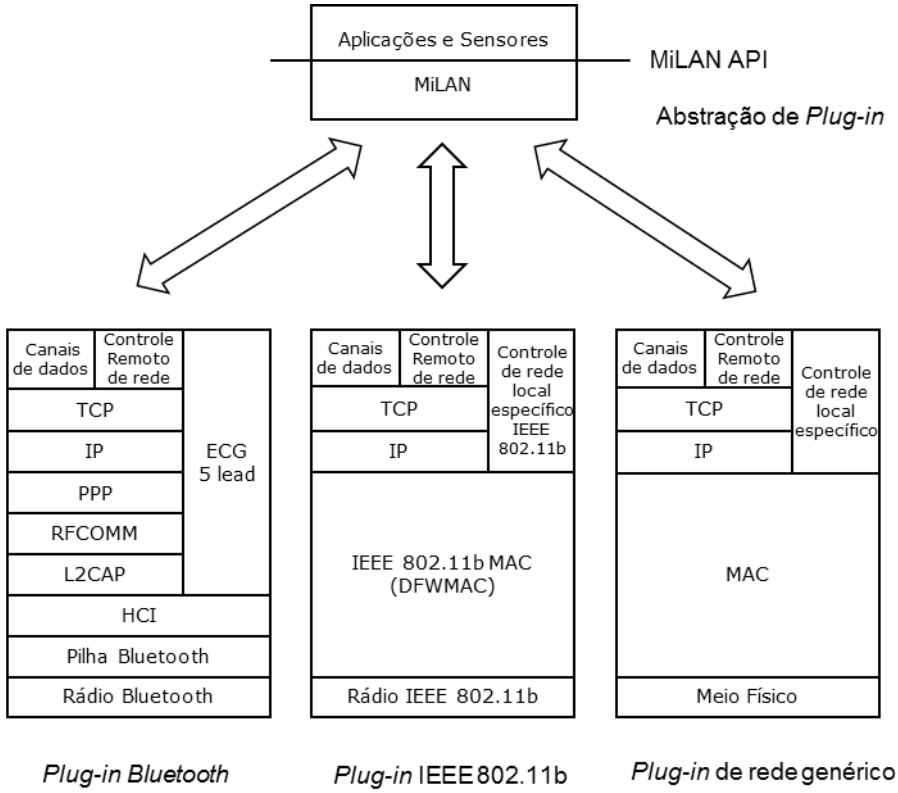


Figura 2.6.1 Pilha de protocolos do MiLAN.

Fonte: [13], traduzida e com adaptações.

O MiLAN fornece Interface de Programação de Aplicações – (API) que representam os requisitos da aplicação para os componentes de baixo nível. Por meio de uma abstração das funcionalidades no nível de rede, o MiLAN executa comandos que determinam a disponibilidade de componentes e que configuram a rede. Os plug-ins específicos da rede convertem os comandos do MILAN nos protocolos específicos da rede [13].

A interface para a aplicação e os componentes de baixo nível permitem que a aplicação forneça um grafo especificando quais componentes de rede podem ser utilizados conforme a sua disponibilidade, que pode mudar ao longo do tempo. Essa interface permite ainda que o MiLAN possa controlar esses componentes de baixo nível.

2.6.1 Seleção do conjunto de sensores considerando o desempenho da aplicação

Considere uma aplicação de monitoramento cardíaco em que muitas variáveis devem ser processadas baseando-se em dados proveniente de sensores. Por exemplo, a frequência cardíaca, a frequência respiratória, o nível de oxigênio no sangue, o fluxo sanguíneo, a pressão sanguínea, o sinal de ECG, a posição e atividade de uma pessoa sendo monitorada; todas essas variáveis são importantes para determinar se o coração está saudável ou com problemas. Se a

aplicação se limita a um único sensor, como por exemplo, o sensor de pressão sanguínea, ele poderia fornecer as demais variáveis acima, com certo grau de acurácia. Por exemplo, o sensor de pressão sanguínea consegue medir diretamente a variável pressão sanguínea com uma acurácia de 100%. Esse mesmo sensor pode fornecer variáveis indiretamente como a frequência cardíaca e o fluxo sanguíneo, entretanto, com uma acurácia inferior a 100%. A acurácia seria incrementada se fossem incluídos dados adicionais de sensores como ECG, frequência cardíaca e nível de oxigênio no sangue [8]. A figura 2.6.1.1 mostra os diferentes parâmetros que são importantes para se inferir a saúde do coração.

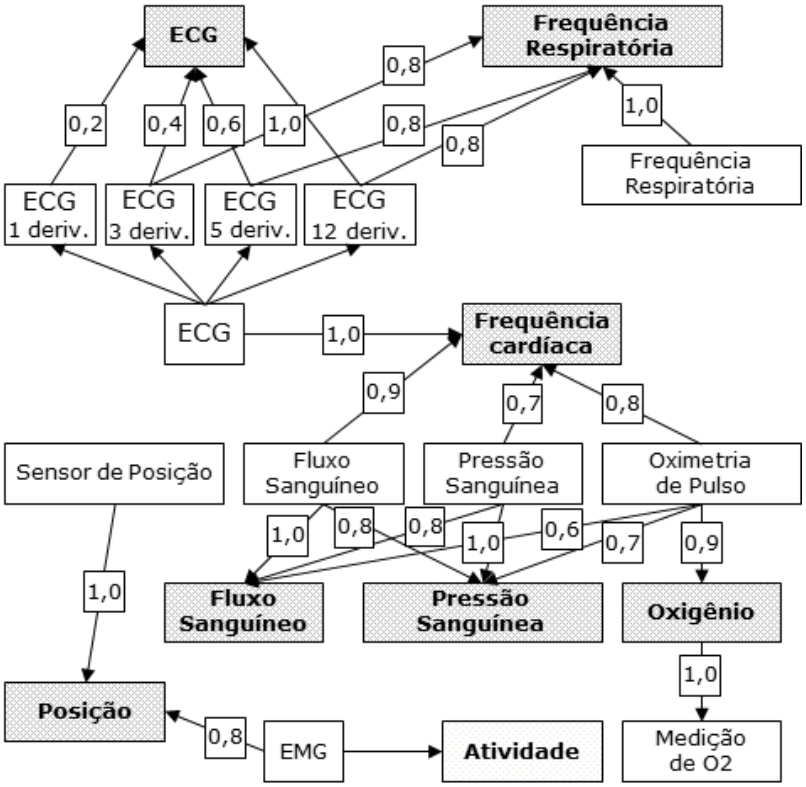


Figura 2.6.1.1 Grafo de variáveis de uma aplicação biomédica de monitoramento cardíaco.
Fonte: [13], traduzido e com adaptações.

Para essa aplicação, os sensores (caixas não axuradas) capturam medições que diretamente ou indiretamente fornecem diferentes variáveis (caixas axuradas) a determinada acurácia.

A solução ideal para a aplicação é extrair quantos dados fossem possíveis para que o monitoramento cardíaco tivesse um aumento da acurácia dos parâmetros medidos. Entretanto, existem algumas restrições que impedem isso. Uma delas é que o tempo de vida do sistema pode ser reduzido se todos os sensores transmitirem ao mesmo tempo. Ao contrário disso, o tempo de vida pode ser estendido se forem requisitados dados de somente um conjunto de

sensores disponíveis, diminuindo a acurácia das variáveis, todavia, permitindo que alguns sensores economizem energia. E, ainda, a capacidade da rede pode não suportar a transferência de dados de todos os sensores transmitindo simultaneamente. Assim, é importante que haja uma forma automática de, dinamicamente, ser escolhido o conjunto de sensores mais apropriado para atender à acurácia definida pela aplicação e, ao mesmo tempo, minimizar o custo de consumo de energia dentro dos limites dos recursos da rede.

Em geral, uma aplicação sabe como extrair um determinado dado por meio de componentes de baixo nível. Essa informação deve ser transmitida ao MiLAN. A proposta do MiLAN é a implementação de um grafo que permite à aplicação especificar as suas informações de desempenho. Nesse grafo, são representados os componentes de baixo nível, por exemplo os nodos sensores, ligados às variáveis que eles podem fornecer. Os *links* que interligam os componentes de baixo nível e as variáveis recebem pesos que determinam a acurácia de cada variável que está sendo medida pelos sensores.

O grafo especificado pela aplicação pode ainda fornecer uma nova variável a partir de duas ou mais variáveis fundidas, funcionando como uma espécie de "sensor virtual". Entretanto isso adiciona ao grafo um maior grau de complexidade, visto que a combinação de variáveis pode crescer exponencialmente.

É possível que a contribuição de cada componente de baixo nível para o desempenho da aplicação varie ao longo do tempo conforme o estado da aplicação. Essa informação deve ser passada ao MiLAN pela aplicação para garantir uma melhor performance da rede. Por exemplo, uma aplicação de monitoramento cardíaco, pode prover três estados de saúde do coração monitorado: saudável, doente, e uma situação intermediária. A aplicação deve informar ao MiLAN quando cada um desses estados acontece e, ainda, qual o mínimo de desempenho, por exemplo a acurácia, deve orientar o MiLAN a escolher o conjunto de sensores para o fornecimento das variáveis.

Suponha que a aplicação envie ao MiLAN as seguintes acurácias requeridas para as variáveis:

- A frequência cardíaca deve ter acurácia maior ou igual a 0,5;
- O nível de oxigênio no sangue deve ter acurácia maior ou igual a 0,7 e
- A pressão sanguínea deve ter acurácia maior que 0,8.

De acordo com o grafo da aplicação fornecido a MiLAN, conforme a figura 2.6.1.1, o MiLAN pode determinar as combinações de sensores que podem satisfazer as necessidades da aplicação, conforme apresenta a tabela 2.6.1.

Tabela 2.6.1 Acurácia de cada conjunto de sensores que satisfazem os requisitos da aplicação. Fonte: [13], traduzido e com adaptações.

Conjunto	Sensores (S_i)	Acurácia Total ($R(S_i)$)
1	A, B, C	1 + 0,9 + 1 = 2,9
2	A, C, E	1 + 1 + 1 = 3,0
3	B, C	0,8 + 0,9 + 1 = 2,7
4	B, C, E	0,8 + 1 + 1 = 2,8
5	C, E	0,7 + 0,9 + 1 = 2,6
6	B, D	0,9 + 0,9 + 1 = 2,8
7	C, D, E	0,9 + 1 + 1 = 2,9

Nessa tabela, A é o sensor de ECG, B é o sensor de oximetria de pulso, C é o sensor de pressão sanguínea, D é o sensor de fluxo sanguíneo e E é o sensor de oxigênio. Foi usada a soma das acurácias individuais para representar a acurácia total do conjunto de sensores $R(S_i)$, entretanto outras formas de se calcular a acurácia total podem ser utilizadas, como por exemplo, se $R(S_i, j)$, é a acurácia obtida usando sensores em um conjunto S_i para se medir a variável *j* então a acurácia total $R(S_i) = \min_j R(S_i, j)$ pode ser mais apropriada para algumas aplicações.

Esses conjuntos de sensores podem ser definidos como F_R, um conjunto de possíveis combinações de sensores que fornecem uma acurácia maior ou igual à acurácia mínima aceitável conforme especificação da aplicação r_j para cada variável de aplicação *j* do conjunto *J*:

$$F_R = \{S_i : \forall j \in J ; \ R(S_i, j) \geq r_j\}. \qquad (2.3.1.1)$$

Para o exemplo acima, r_1 = 0,5, r_2 = 0,7 e r_3 = 0,8 e F_R = {(A, B, C), (A, C, E), (B, C), (B, C, E), (C, E), (B, D), (C, D, E)}. Todos os conjuntos em F_R são aceitáveis, então a pergunta é: qual conjunto S_i em F_R deve ser escolhido? Isso depende de outros parâmetros de sistema como o custo de uso da rede ou o consumo de energia.

2.6.2 Seleção do conjunto de sensores considerando o custo de rede

Uma das complexidades inerentes às redes sem fio é a limitação na largura de banda. Assumindo que todos os sensores estão em uma única rede centralizada, com uma largura de banda tal que limita um total de dados que podem ser transmitidos para a aplicação. Se for assumido, por exemplo, que a todos os nodos estão em uma rede Bluetooth ou 802.11 operando no modo de infraestrutura, todos os nodos transmitem dados diretamente para a aplicação a partir de um nó sorvedouro (por meio de um mestre na rede *bluetooth* ou de um ponto de acesso na rede 802.11).

Entretanto, em um ambiente mais complexo como as *scatternets* das redes Bluetooth ou redes 802.11 operando em modo não estruturado, a capacidade de transmissão de dados pode aumentar em função do reuso espacial e a localização do nodo passa a ser um importante ponto que determina o custo da rede. Assim, é necessário que o MiLAN calcule o custo de rede de acordo com a tecnologia empregada, por exemplo, numa rede *bluetooth* 4.0 deve ser especificado um máximo de 25 Mbps de largura de banda e oito nodos por *piconet* sendo um designado como mestre. *Templates* seriam então empregados para se implementar cada uma das tecnologias, assunto que está fora do escopo dessa dissertação.

Outra preocupação que deve ser considerada pelo MiLAN é o monitoramento contínuo da rede a fim de encontrar novos nodos e verificar quando os nodos não estão mais acessíveis (em função da mobilidade o por não ter energia suficiente para se comunicar com a rede). Quando novos nodos são descobertos por meio do método de descoberta no nível de rede, o MiLAN deve saber as informações de especificação sobre esse nodo a fim de saber qual a forma apropriada de se comunicar com o nodo. Para isso, o MiLAN utiliza o padrão IEEE 1451 que define uma interface inteligente para sensores e atuadores. O padrão IEEE 1451 define uma padronização para sensores especificando os seus tipos, atributos, operações e informações de calibração (9).

No caso do conjunto de sensores possíveis determinados pela análise da acurácia pode ser que haja alguns conjuntos de sensores que atendam às limitações da rede. Esse conjunto de sensores são definidos como um conjunto de sensores possíveis segundo a análise da rede:

$$F_N = \{S_i : N(S_i) \leq n_o\}, \qquad (2.3.2.1)$$

em que $N(S_i)$ é o custo total de se usar todos os sensores em S_i e n_0 é o custo máximo suportado pela rede. Por exemplo, assumindo-se uma rede centralizada, n_0 é a taxa máxima de transmissão que a rede pode suportar, por exemplo, 25 Mbps no caso do *bluetooth* e

$$N(S_i) = \sum_{s_j \in S_i} R_b(s_j), \tag{2.3.2.2}$$

em que $R_b(s_j)$ é a taxa de transmissão do sensor j. Somente conjuntos em F_N podem ser suportados fisicamente pela rede. Conforme a análise anterior, sabe-se que somente os conjuntos em F_R são permitidos pelo ponto de vista da aplicação. Entretanto, se forem combinadas as duas análises têm-se os seguintes conjuntos possíveis de sensores:

$$F = F_R \cap F_N. \tag{2.3.2.3}$$

Então, o MiLAN deve escolher a solução que vier de um dos conjuntos de sensores em F.

2.6.3 Seleção do conjunto de sensores considerando o consumo de energia

O parâmetro final que deve ser considerado para a decisão de quais sensores devem ser escolhidos é a dissipação de energia. Existe um custo de energia para o sensor sempre que ele transmite dados na rede sem fio. Esse custo pode depender da distância entre o sensor transmissor e o sensor receptor se for assumido que é empregado um controle de potência. Depende também do conjunto de sensores escolhido e como ele atua na rede (por exemplo, usando nodo mestre, se existe gateway entre múltiplas *piconets* etc.). Entretanto, foi assumido inicialmente que o custo de energia por nodo é independente de qual conjunto foi escolhido. Esse custo total de dissipação de energia para cada conjunto S_i é:

$$C_p(S_i) = \sum_{s_j \in S_i} C_p(s_j), \tag{2.3.3.1}$$

em que $C_p(s_j)$ é o custo do nodo s_j se ele for selecionado para transmitir dados. Podem ser incluídas formas de ponderar o custo individual de energia dos nodos caso o nodo esteja operando com baixa disponibilidade de energia. A exemplo do método utilizado por Singh *et tal*. [10], outras formas de se medir o custo da energia na rede podem ser empregadas.

2.6.4 Soluções de compromisso

Dentro dos possíveis conjunto de sensores F, o MiLAN escolhe um conjunto de sensores Si, que representa a melhor relação de desempenho/custo como determinado pela acurácia do

conjunto de sensores $R(S_i)$ e pelo custo do conjunto de sensores $C_p(S_i)$. A melhor relação desempenho/custo irá depender da aplicação. Uma possível saída para se definir apropriadamente S_i em um determinado tempo é minimizar o consumo de energia escolhendo o conjunto F, que possui o menor custo $C_p(S_i)$. Essa é uma boa solução de curto prazo, mas a aplicação pode também necessitar de melhor desempenho em longos períodos de tempo. Ao invés de se utilizar um conjunto de sensores que em um dado momento é o melhor para o sistema até que os sensores comecem a esgotar a sua energia e se desligarem, uma melhor forma seria uma visão de grandes períodos de tempo a fim de encontrar um melhor agendamento dos conjuntos de sensores que otimize o consumo de energia de todos os conjuntos. Esse problema de otimização pode ser resolvido usando programação linear e ser resolvido em tempo polinomial [11]. Entretanto, o tempo para se resolver essa otimização pode ser proibitivo dependendo da aplicação, em outras palavras do número de sensores possíveis de acordo com o critério da acurácia.

Uma complicação adicional surge quando são consideradas múltiplas aplicações trabalhando na mesma rede, compartilhando os sensores. Cada aplicação utilizando os recursos da rede pode acessá-la de maneira diferenciada, por exemplo, com credenciais de acesso distintas, que determinam se a aplicação irá acessar um dado com maior ou menor acurácia. Ademais, cada aplicação ainda pode possuir acessos distintos para usuários também distintos. Usando essa informação, o MiLAN pode determinar que componentes do sistema devem se conectar para satisfazer as necessidades de cada usuário e também de cada aplicação.

2.7 AMPLIANDO A CAPACIDADE DE INTEGRAÇÃO DO MILAN

Diante do avanço das tecnologias voltadas para redes de sensores essas redes tendem a se tornar cada vez mais heterogêneas e, quanto maior a complexidade das aplicações, mais os envolvidos sensores possuirão funções diversas.

A implementação de redes de sensores utilizando-se *midlewares* tradicionais em sistemas heterogêneos, constituídos de protocolos distintos, cada aplicação deve conhecer em detalhes como deve se dar a comunicação com a rede, inclusive no nível de gerenciamento, o que é uma complexidade adicional para a aplicação.

Essas redes são o principal alvo do MiLAN, que proporciona a possibilidade de abstrair para as aplicações as funcionalidades de baixo nível, não necessitando da aplicação conhecer detalhes de gerenciamento da rede para se extrair os dados de sensoriamento.

As aplicações têm se tornado cada vez mais complexas na medida em que aumentaram as possibilidades de integrar seus dados a outros gerados por outras aplicações. Ademais, a rede mundial de computadores recebe cada vez mais acessantes e aplicações, desde aquelas que auxiliam os sistemas corporativos e governos a se integrarem com clientes e sociedade, até aquelas voltadas para o entretenimento e comunicação pessoal como as redes sociais, serviços de e-mail, jogos, entre outros.

Assim, propõe-se a inclusão de uma camada de integração ao MiLAN que será responsável por comunicar-se com as aplicações e outras redes sobre a ótica do SOA, proporcionando ao MiLAN o a capacidade de ser implementado por aplicações heterogêneas e, ainda, aumentar o seu alcance na integração com outras redes, principalmente com a internet.

3 MILAN SERVICES

3.1 ESPECIFICAÇÃO E MODELAGEM

Para possibilitar que as diversas aplicações de plataformas heterogêneas acessem uma rede de sensores por meio do MiLAN de maneira que essas redes se adaptem às necessidades das aplicações, inclusive se comunicando com elas com seus protocolos distintos, baseando-se em SOA, é necessário que seja agregado ao MiLAN uma nova camada para integração com as aplicações, como mostra a figura 3.1.1. A plataforma proposta nessa dissertação para atingir esse objetivo é chamada aqui de MiLAN Services.

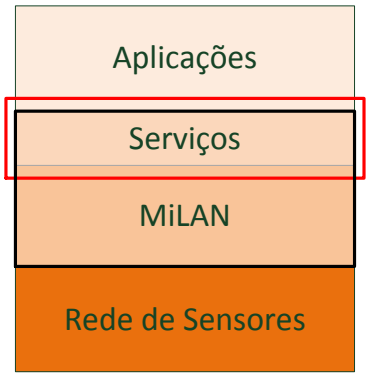


Figura 3.1.1 Arquitetura em Camadas de uma Solução Implementando o MiLAN Services.

Por meio da camada de integração, as aplicações passam a se comunicar com o MiLAN por meio de serviços disponibilizados em um *endpoint*.

Um *endpoint* é composto por um contrato, que especifica as condições de acesso ao serviço, um *binding*, que define a forma que a comunicação deve acontecer, em outras palavras, qual o protocolo de comunicação, e o endereço de acesso.

A aplicação acessa um recurso da rede sem necessitar saber como ele foi implementado consumindo os serviços disponíveis. Da mesma forma, a rede não precisa saber detalhes da aplicação, ficando o negócio da aplicação separado da rede e se comunicando com ela por meio de uma interface de consumo de serviços, como mostra a figura 3.1.2. Todo o controle e gerenciamento de baixo nível da rede continua sendo realizado pela infraestrutura de coordenação do MiLAN, podendo ainda ser disponibilizados serviços que dão informações

sobre o monitoramento da rede, que podem ser consumidos pela aplicação ou pela própria rede como informação em alto nível.

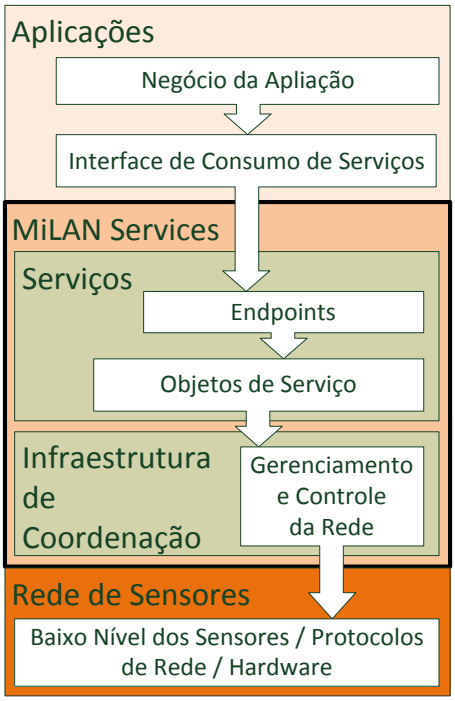


Figura 3.1.2 Integração entre as camadas de uma solução que implementa o MiLAN Services

Para que a aplicação possa se comunicar com o MiLAN para transferir as especificações iniciais do sistema, como os grafos de QoS dos sensores e níveis de priorização, é disponibilizado um *endpoint* de inicialização para a transferência dessas especificações. E, ainda, todas as especificações de quais os serviços disponíveis, quais os protocolos de comunicação e os endereços desses serviços, são disponibilizados no endpoint de inicialização por meio de uma WSDL, ou ainda, a aplicação pode buscar essas informações em um Registro de Serviços ou um *UDDI – Universal Description Discovery and Integration*, que é espécie de "páginas amarelas" onde se publicam informações de serviços em uma rede. A figura 3.1.3 mostra um diagrama de blocos com o *endpoint* de inicialização, as WSDLs de cada objeto de serviço, o Registro de Serviços e demais componentes da rede.

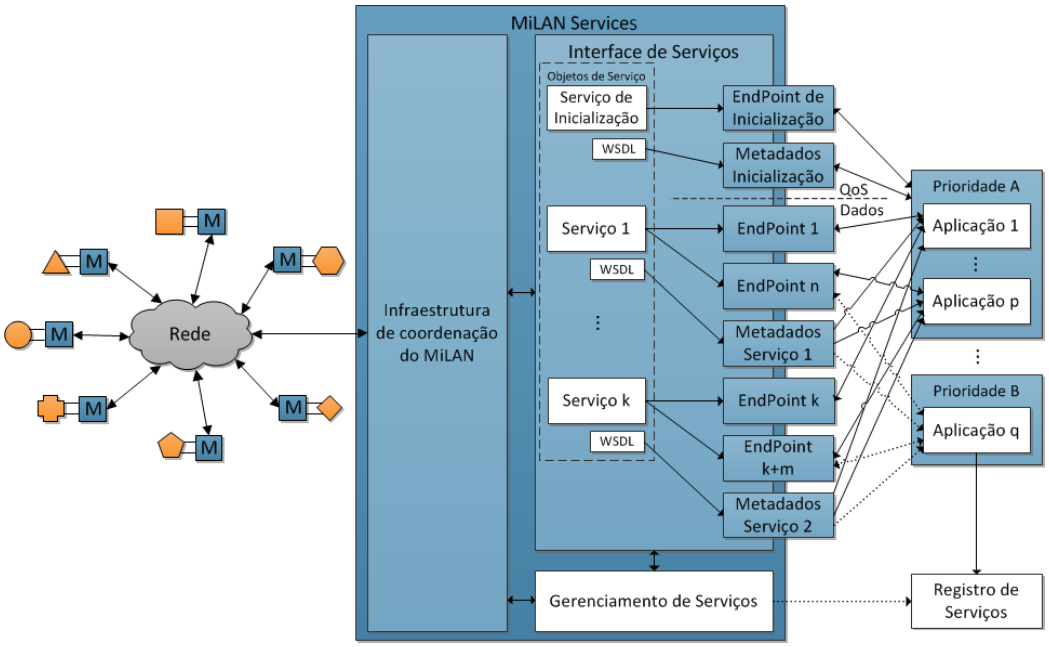


Figura 3.1.3 Diagrama de blocos do MiLAN Services.

Uma das grandes características que tornam o MiLAN Services robusto é a possibilidade de se utilizar diversos protocolos para comunicação com os serviços disponibilizados pela rede.

Outra característica importante do MiLAN Services é que os serviços disponibilizados pela rede são dinâmicos. Isso quer dizer que a aplicação é quem determina quais são os serviços que estarão disponíveis e, caso algum sensor da rede se desassociar e algum parâmetro como QoS não puder ser atendido, um objeto de serviço pode ser excluído, estando disponíveis apenas os serviços que a rede pode prover.

Para melhor entender o funcionamento do MiLAN Services, são descritos a seguir modelos com cenários que representam situações reais que poderiam implementar a plataforma proposta neste documento.

3.1.1 Modelo Genérico

Esse modelo representa o MiLAN Services de uma maneira genérica do ponto funcional. Ele é o modelo mínimo das funcionalidades do MiLAN Services A partir desse modelo é possível abranger todas as formas de implementação do MiLAN Services. A figura 3.1.1.1 mostra um diagrama de blocos desse modelo.

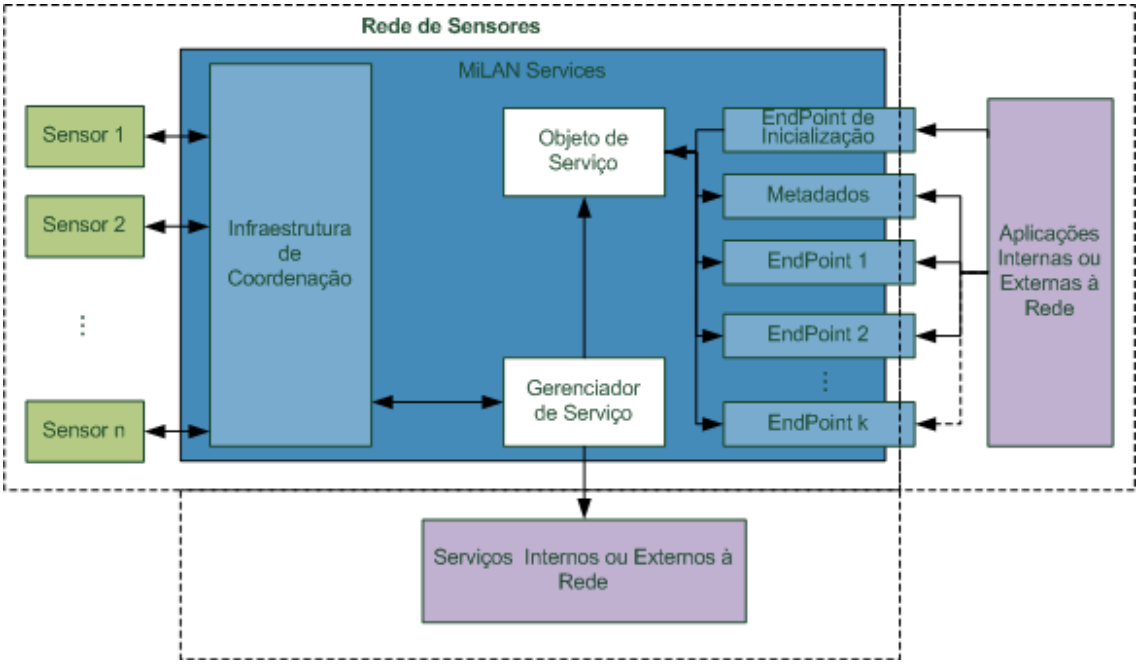


Figura 3.1.1.1 Modelo genérico.

No modelo pode-se observar que ainda permanecem as três camadas básicas de uma aplicação de RSSF: A aplicação, o middleware e a rede de sensores. As aplicações, sejam elas internas ou externas à rede, acessam os recursos do sistema por meio de serviços representados por *endpoints*. Os nodos sensores, que também podem ter aplicações internas (para o processamento distribuído das aplicações internas, por exemplo), podem disponibilizar os dados de sensoriamento e gerenciamento do nodo por meio de serviços em sua própria infraestrutura física ou por meio de outro nodo sensor, que funcionaria como um nó vertedouro.

Cada serviço é visto pelo MiLAN Services como um objeto de serviço. Cada objeto de serviço, criado a partir de especificações da aplicação, pode ser extinto ao longo do tempo, a depender se a rede não consegue atender às especificações fornecidas pela aplicação para a sua criação. Por exemplo, a aplicação definiu que fosse criado um serviço que disponibilizasse o dado de frequência cardíaca a partir de um sensor. Após a criação do objeto de serviço, que disponibiliza o dado de frequência cardíaca, os dados de acesso a esse serviço (ou um método de um serviço) são disponibilizados em uma WSDL. Caso o nodo sensor venha a sair da rede, por exemplo, por descarregamento da bateria, o objeto de serviço é extinto e o respectivo método de serviço é excluído da WSDL.

Serviços externos ou internos à rede podem também auxiliar o Gerenciamento de Serviços na criação de novos serviços. Por exemplo, sendo desejado que o dado de frequência cardíaca seja interpretado para uma informação do tipo frequência cardíaca baixa, alta ou normal, o MiLAN

Services não possui um módulo que realiza essa função, pois não foi projetado para esse fim. Nesse caso, o MiLAN pode buscar essa informação a partir de um outro serviço fornecendo o dado de frequência cardíaca e capturando desse serviço o resultado da interpretação.

3.1.2 Fase de formação da rede

Esta é a fase em que os sensores são associados à rede conforme são ligados. Como a fase de formação de rede é realizada depende diretamente da tecnologia utilizada como Bluetooth, *Zigbee* ou Wi-Fi. O MilLAN Services controla essas redes por meio de seus plug-ins que convertem comandos de gerenciamento e de requisição de dados aos protocolos de cada rede.

3.1.3 Fase de inicialização

Utilizando o *endpoint* de inicialização do MiLAN Services, a aplicação envia o grafo de QoS de sensores, o grafo de variáveis de estado e o seu nível de prioridade. A aplicação também envia ao MiLAN Services um grafo contendo as especificações dos serviços a serem criados. Essas especificações contêm dados como os contratos, os *bindings* e serviços que serão publicados em um registro de serviços como o UDDI. Logo em seguida a aplicação envia o seu estado.

A aplicação também pode enviar uma lista contendo os níveis de prioridade de outras aplicações.

Pode ser previamente definida na rede uma aplicação de gerenciamento, que possui os privilégios de inicialização da rede. Assim, é possível definir melhor como serão as priorizações da rede em relação à todas as aplicações do sistema.

3.1.4 Fase de configuração de nodos sensores

Nessa fase o MiLAN Services inicia o processo de configuração dos nodos sensores. De posse dos grafos de QoS dos sensores e de variáveis de estado o MiLAN Services calcula F_A (aplicação), determinando qual o conjunto de sensores que poderá atender às especificações da aplicação.

Por meio de informações sobre as condições da rede como taxa máxima de transmissão e energia nos nodos o MiLAN Services, calcula F_N (rede) e, realizando a interseção ente F_A e F_N,

configura os nodos sensores a serem utilizados colocando em espera ou desligando aqueles que não serão utilizados.

3.1.5 Fase de criação dos serviços

O gerenciador de serviços do MiLAN Services cria objetos de serviço conforme os grafos de especificações de serviço enviado pela aplicação. Logo em seguida são criados os *endpoints* do serviço com os *bindings*, os endereços e os contratos. O MiLAN Services também gera a WSDL, caso tenha sido demandada pela aplicação e a disponibiliza em um *endpoint* específico. Criados os objetos de serviço, os *endpoints* e a WSDL, o MiLAN Services retorna para a aplicação um *token* de autenticação e os endereços dos *endpoints* criados.

3.1.6 Fase de consumo dos serviços

Uma aplicação solicita realiza uma chamada de método de um serviço do MiLAN Services que pode passar um *token* (caso haja mais de uma aplicação, com a finalidade de identificar as aplicações) de autenticação e informando os parâmetros do método.

O MiLAN Services valida o *token* da aplicação verificando a sua autenticidade e a validade. Logo em seguida verifica se é capaz de prosseguir com a solicitação verificando o método, validando os parâmetros e analisando o nível de prioridade da aplicação. Caso não seja possível prosseguir com a requisição de uso do serviço o MiLAN reporta para a aplicação o código e a descrição da exceção. Caso contrário, o MiLAN Services solicita os dados dos nodos sensores envolvidos na operação e devolve para a aplicação os dados do método de serviço.

3.1.7 Considerações sobre o consumo e manutenção dos serviços

Pode-se utilizar um token de autenticação para que se definida a validade do acesso ao serviço por meio de um selo de tempo. Assim, a aplicação deverá utilizar esse *token* até a sua expiração; após esse prazo, a aplicação deverá realizar um novo processo para consumo dos serviços.

O *token* de autenticação, além de garantir a identificação da aplicação para verificar se ela possui acesso aos serviços, também auxilia na garantia de que seja dado o devido nível de prioridade para cada aplicação.

A WSDL contém a lista de métodos do serviço e como esses métodos podem ser acessados. Como pode haver alterações na rede como a dissociação de nodos sensores, algum objeto de serviço pode ser alterado ou até mesmo extinto. A cada alteração na rede em termos de

associação e dissociação ou o desligamento de nodos sensores em função do nível de energia, o MiLAN Services recalcula o valor de F_N e, por consequência, reavalia quais os serviços ele tem condições de prover conforme o estado da rede, altera os objetos de serviço, seus *endpoints* e a WSDL.

Um serviço criado por uma aplicação pode ser compartilhado com outras aplicações respeitando o nível de prioridade de cada uma delas. Um serviço que provê informação para um sistema local pode ter prioridade em relação a uma aplicação na internet, por exemplo.

Caso uma aplicação já tenha realizado o procedimento de envio dos grafos e lista de priorização, ela poderá iniciar a chamada de métodos dos serviços informando o *token* de autenticação.

3.1.8 Barramento de Serviços

O barramento de serviços no MiLAN Services vai ao encontro do conceito de Orientação a Serviços, centralizando os serviços em um ponto a ser acessado pelas aplicações e permitindo um melhor gerenciamento dos objetos de serviço. Além disso, o barramento disponibiliza uma lista de serviços, a WSDL, para que as aplicações possam ter informações de utilização dos serviços disponíveis. A figura 3.1.8.1 mostra um diagrama de blocos de um modelo com dois barramentos de serviços, um interno à rede, acessível pelas aplicações internas, e outro externo à rede, acessível pelas aplicações externas ou internas.

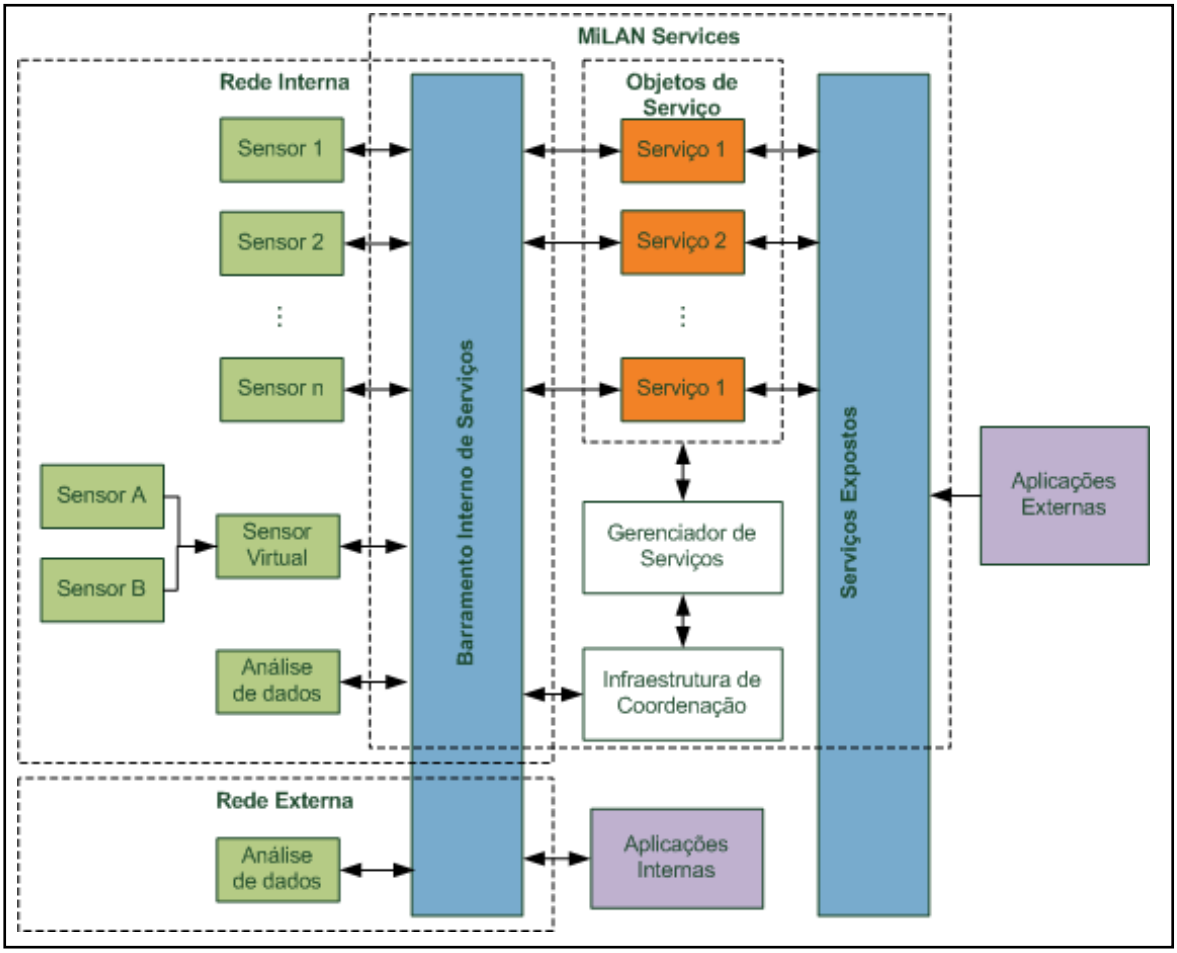


Figura 3.1.8.1 Barramento de Serviços

3.2 CENÁRIOS DE IMPLEMENTAÇÃO

3.2.1 Um nodo sensor como sendo um serviço

Um nodo sensor pode ser visto com um serviço que disponibiliza os dados dos seus sensores para uso de aplicações conforme especificações em uma WSDL. Na visão do nodo, o MiLAN Services verificaria os sensores instalados e em funcionamento e criaria um serviço com base nos sensores disponíveis. Esse serviço, além de fornecer os métodos provendo dados dos sensores, pode disponibilizar também funções de gerenciamento como, por exemplo, o nível de energia da bateria. A figura 3.2.1.1 mostra um diagrama de blocos de um sensor que expõe serviços para uso das aplicações.

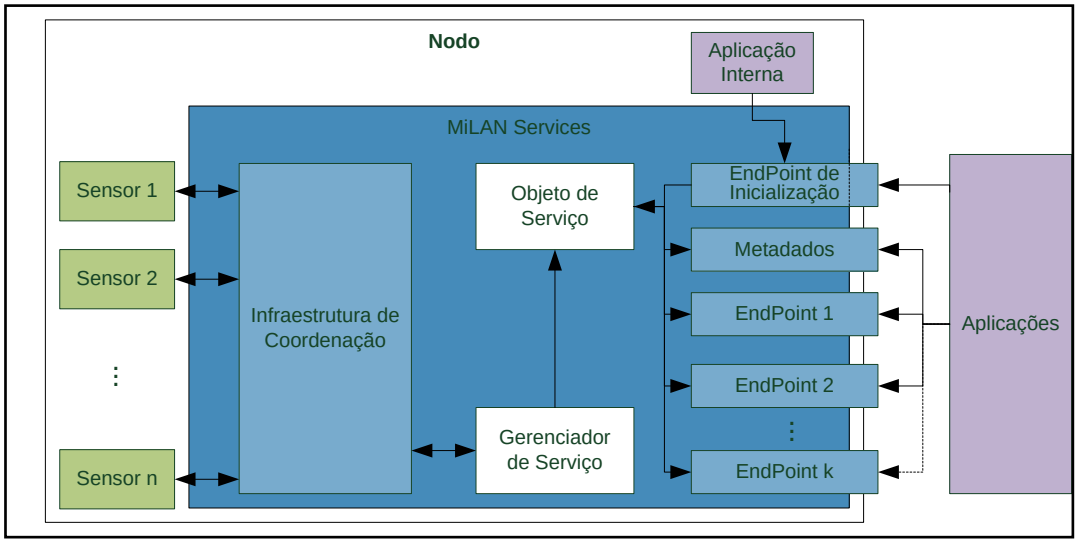


Figura 3.2.1.1 O MiLAN Services do ponto de vista de um nodo e aplicações externas requisitando serviços desse nodo.

A vantagem dessa abordagem é que a criação do serviço no nodo é dinâmica e independente do número e dos tipos de sensores no nodo. Um novo sensor instalado no nodo é visto com um método a ser registrado no serviço. Como cada sensor é acessado por meio de um *plug-in*, não há de se realizar uma nova compilação do MiLAN Services e nem da aplicação que solicita a criação dos serviços. Instala-se o plug-in, caso não exista, que realiza o interfarceamento entre o hardware e o MiLAN Services e adiciona-se as especificações de uso do sensor em arquivo de configuração a ser utilizado pela aplicação interna.

Um exemplo seria um nodo contendo sensores de pressão sanguínea e fluxo sanguíneo. A aplicação de inicialização informa ao MiLAN Services as especificações de serviço assim, a infraestrutura de coordenação procura os sensores instalados, configura esses sensores. O MiLAN Services então disponibilizaria o acesso aos sensores por meio de *endpoints* de serviço.

Na inicialização, o nodo sensor executa uma aplicação interna que envia especificações de criação do serviço para o MiLAN Services por meio do endpoint de inicialização. A infraestrutura de coordenação do MiLAN Services configura os sensores do nodo encaminhando a lista de sensores disponíveis ao gerenciador de serviços que por sua vez cria os endpoints de serviço conforme as especificações.

3.2.2 Composição de serviços e interpretação de dados

Uma aplicação pode solicitar ao MiLAN Services que capture dados de sensores da rede e que faça, por meio de *plug-in* ou de serviço externo à rede, uma análise de dados para disponibilização de informação em um nível mais elevado para a aplicação. Um exemplo seria um serviço que fornece uma avaliação da frequência cardíaca de um paciente com base nos dados de sensores. Esse serviço informaria se a frequência cardíaca do paciente está baixa, normal ou alta. Para que o MiLAN services interprete os dados provenientes desses sensores, ele recebe instruções da aplicação com instruções para interpretação, podendo ser utilizando o interpretador próprio do MiLAN Services, um *plug-in* ou um serviço externo.

Com a combinação de variáveis provenientes de dois ou mais sensores surge um sensor "virtual". O MiLAN Services permite que essa combinação de variáveis seja feita também com variáveis originárias de serviços que interpretam os dados de um ou mais sensores ou serviços da rede. O resultado de qualquer combinação é um serviço disponibilizado por meio de *endpoints*. Com isso, a aplicação não consegue determinar se a variável é proveniente de uma combinação de outras variáveis ou vem diretamente de um sensoriamento do nodo, ficando esse detalhe de constituição da "variável virtual" a cargo do MiLAN. A figura 3.2.2.1 mostra as aplicações acessando o sensor virtual da mesma maneira que acessa um outro serviço no MiLAN Services.

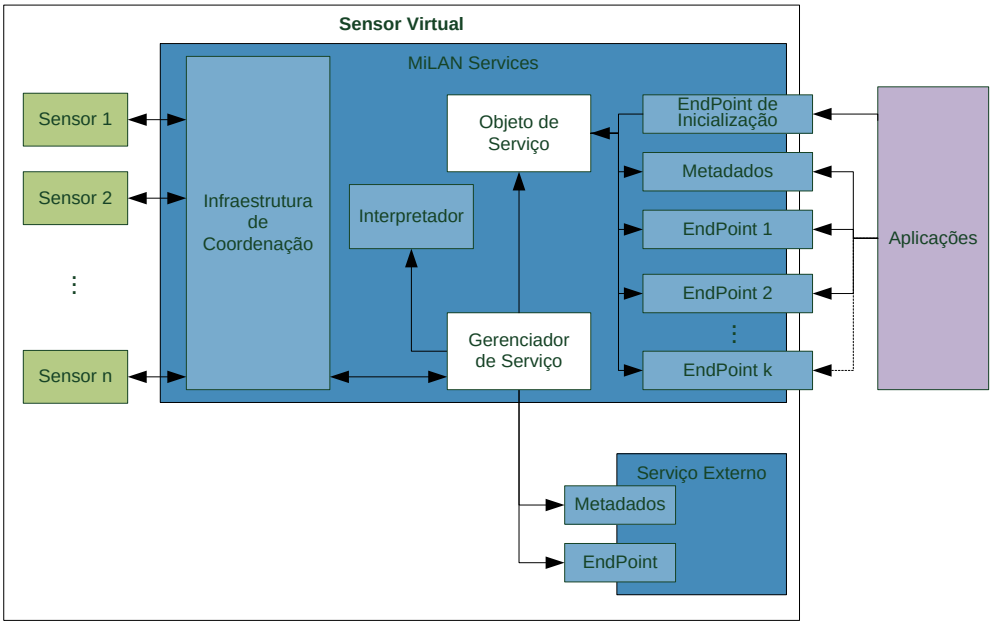


Figura 3.2.2.1 Exemplo de um sensor virtual em que serviços são formados por meio de outros serviços provenientes de sensores internos ou externos à rede.

Além de sensores compondo uma informação a ser disponibilizada como serviço, dois ou mais serviços internos ou externos à rede podem ser compostos gerando um novo serviço. Um exemplo seria uma aplicação externa que solicita a criação de um serviço que informa se a frequência cardíaca do paciente está baixa, normal ou alta. Esse serviço seria formado baseando-se em dados de algum serviço da própria rede que fornece a frequência cardíaca e um serviço externo que informa se um paciente está em uso de beta bloqueador, medicamento que reduz a frequência cardíaca. A composição desses dois serviços resultaria em um serviço que faria a análise da frequência cardíaca com base nos dados dos sensores e de uma aplicação externa. Com base no grafo de interpretação fornecido pela aplicação o MiLAN Services tem condições de informar se a frequência cardíaca está alta, normal ou baixa.

3.2.3 Uma rede de sensores disponibilizando serviços para aplicações externas

A visão de mais fácil dedução de uso do MiLAN Services é a rede de sensores disponibilizando dados por meio de serviços, por exemplo, por um nodo gateway.

Uma aplicação envia para o MILAN Services as especificações de serviço, grafos de QoS e de variáveis de estado, níveis de prioridade das aplicações e, caso necessário, um grafo de interpretação de dados. O MiLAN Services seleciona os sensores a serem utilizados, configura esses sensores, cria os objetos de serviço e disponibiliza os *endpoints* de serviço com a WSDL. A aplicação externa à rede passa então a acessar os dados dos sensores por meio dos serviços disponibilizados.

Um exemplo seria um paciente com vários sensores colocados em seu corpo chegando de uma UTI móvel a um hospital. Inicialmente os equipamentos instalados em uma sala de cirurgia recebem informações do paciente que está entrará em procedimento cirúrgico. Os equipamentos da sala começam a realizar o processo de descoberta dos nodos sensores instalados no paciente por meio dos serviços disponibilizados pelo MiLAN Services. Todos os equipamentos e os nodos sensores formam então uma rede que disponibiliza serviços a serem utilizados tanto por aplicações internas à rede e externas a ela. Um eletrocardiógrafo utilizaria um método desse serviço de UTI para obter dados do sensor de ECG e de oximetria de pulso, por exemplo. Uma aplicação externa a essa rede poderia solicitar dados de vários sensores do paciente pelo serviço de UTI e disponibilizá-los de forma consolidada em um prontuário do paciente.

A figura 3.2.3.1 mostra as aplicações externas à rede acessando a rede de sensores por meio dos serviços.

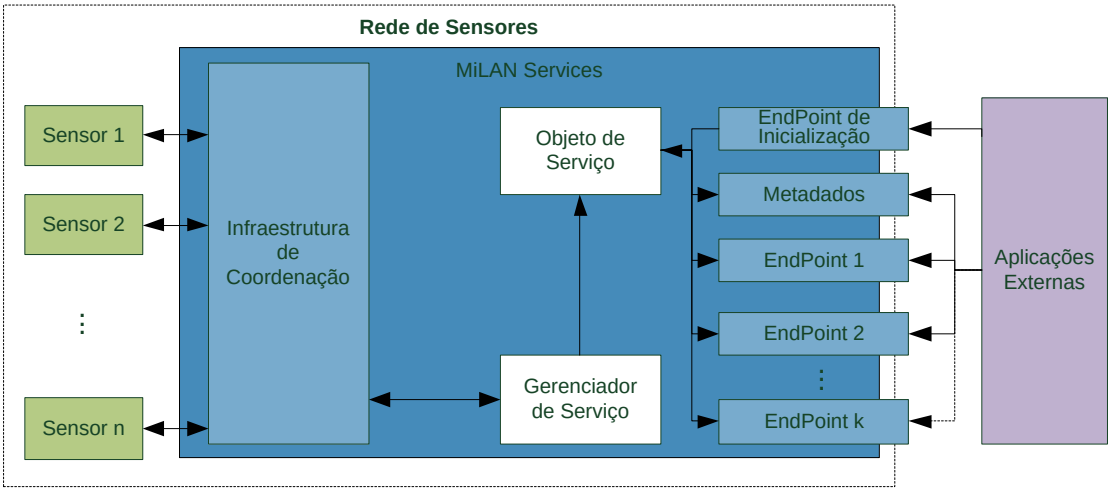


Figura 3.2.3.1 O MiLAN Services do ponto de vista das aplicações externas à rede.

4 IMPLEMENTAÇÃO

4.1 PLATAFORMA DE SISTEMA E AMBIENTE DE DESENVOLVIMENTO

Todo o projeto foi desenvolvido na plataforma Java, máquina virtual versão 6, Servidor Web GlassFish 4.1.1. Também foi utilizado o Kit de Desenvolvimento Java SDK 1.8 e como ambiente de desenvolvimento o Netbeans 8.1. A figura 4.1.1 mostra uma tela do ambiente de desenvolvimento Netbeans com todos os projetos do MiLAN Services.

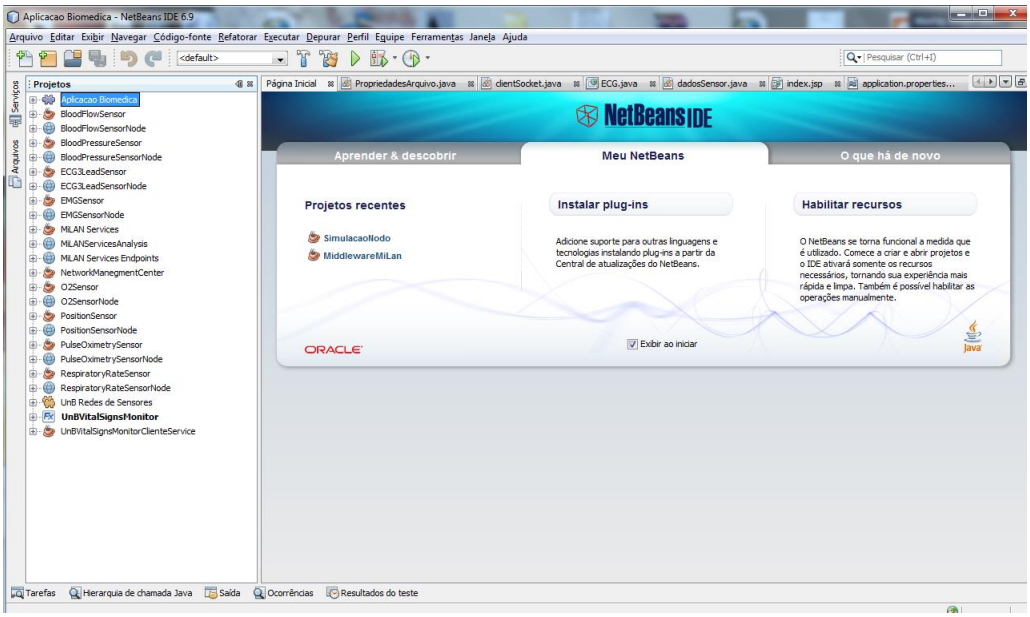


Figura 4.1.1 Ambiente de desenvolvimento Netbeans.

4.2 HARDWARE

Para se obter um ambiente mais próximo de uma rede de sensores sem fio, foram utilizados notebooks com sistemas operacionais Microsoft Windows 10, rede *Wi-Fi*, sendo um dos notebooks utilizado como gateway para comunicação com uma aplicação externa.

A figura 4.2.1 mostra a implementação da rede e a distribuição dos nodos sensores.

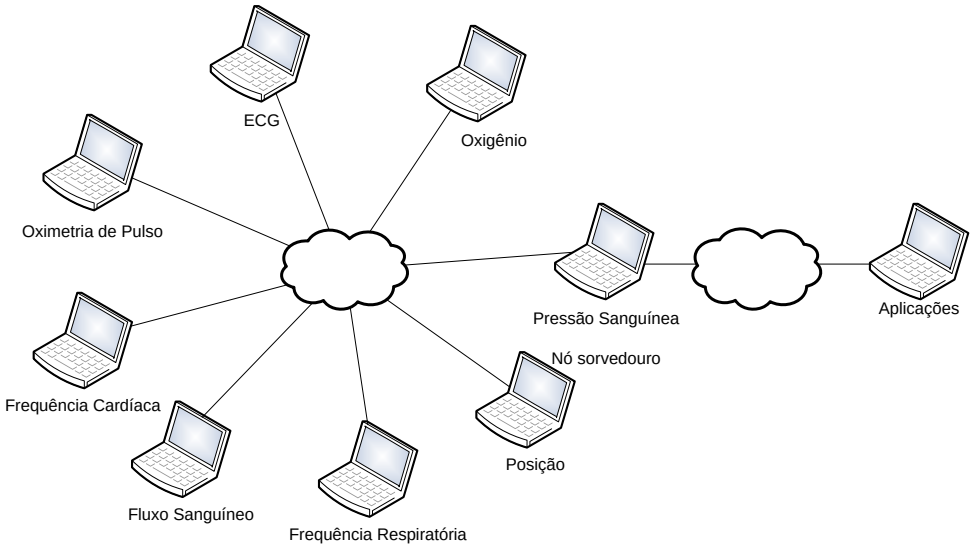


Figura 4.2.1 Implementação da rede de sensores

O nodo sensor de pressão sanguínea também realiza o papel de nó sorvedouro assim, as chamadas realizadas pela aplicação ou encaminhadas a ela passam pelo nó sorvedoruro que separa a rede de sensores sem fio das redes externas.

A escolha dessa arquitetura foi escolhida com o objetivo de se implementar os principais conceitos do MiLAN Services. Assim, houve maior foco na criação de uma estrutura em que fosse implementada uma rede se comunicando basicamente por meio de serviços. Há outras formas de se implementar servidores Web em nodos sensores, como sugere MOKARZEL [18], numa proposta de servidor Web estático.

4.3 NÚCLEO DE GERENCIAMENTO DO MILAN SERVICES

O núcleo de gerenciamento do MiLAN Services tem como função a avaliação dos parâmetros da rede de sensores e das aplicações para que sejam selecionados os nodos sensores que fornecerão dados para as aplicações. Essa análise leva em consideração os níveis de QoS solicitados pelas aplicações, o estado de cada aplicação, o nível de prioridade das aplicações, o estado de cada nodo sensor incluindo o estado da bateria, as características de cada sensor para o atendimento dos níveis de QoS e condições da rede como a melhor rota para transmissão de mensagens e banda de transmissão.

Na figura 4.3.1, pode ser visto o fluxo de mensagens entre processos no núcleo de controle e seleção de nodos sensores no módulo de gerenciamento do MiLAN Services.

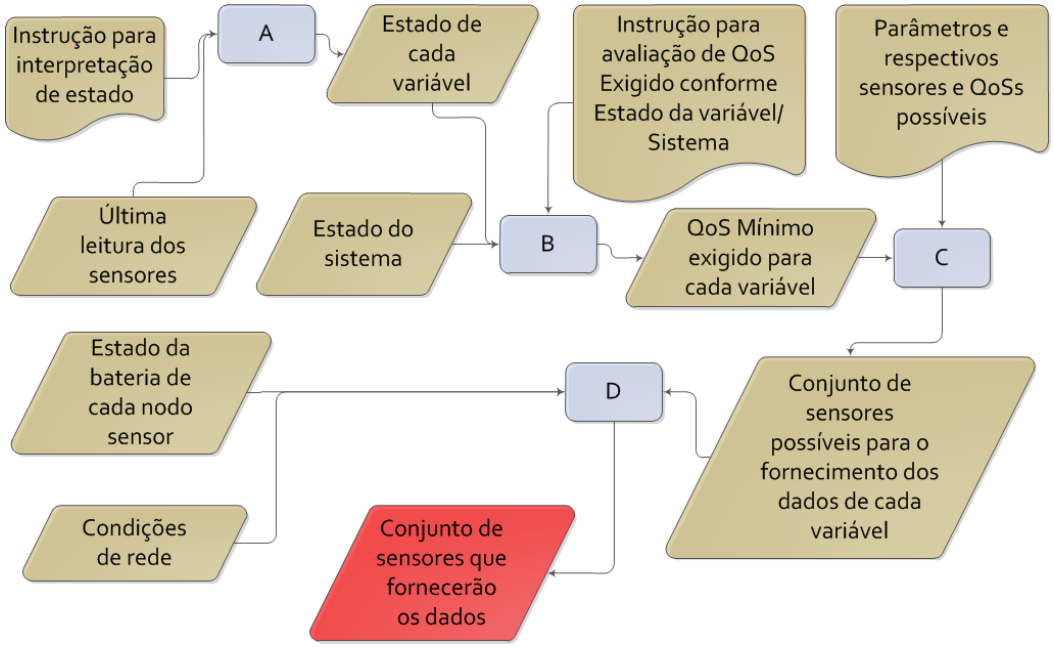


Figura 4.3.1 Processos do MiLAN Services para a seleção de nodos sensores.

As instruções são encaminhadas para interpretação de estado da aplicação e a última leitura dos sensores para o processo A, que é responsável pela definição do estado de cada variável. Por meio do estado de cada variável, do estado do sistema e das instruções para avaliação dos parâmetros de QoS, o processo B analisa qual o QoS mínimo exigido para cada variável. No processo C, tendo como entrada a saída do processo B e a mensagem fornecida pela aplicação que informa quais os nodos sensores podem fornecer que parâmetros a serem sensoriados, é obtido o conjunto de sensores possíveis para o fornecimento dos dados de cada variável. No processo D, por meio da saída do processo D e dados dobre o estado da bateria dos sensores e das condições da rede, são fornecidos o conjunto de sensores que fornecerão os dados para cada aplicação.

A seguir são detalhados cada um dos processos: Processo A, que trata da interpretação dos estados das variáveis; Processo B, sobre a avaliação do QoS mínimo exigido por cada variável; Processo C, que trata da seleção dos sensores possíveis para cada parâmetro de medição e Processo D, sobre a seleção do sensores que fornecem dados para a aplicação.

4.3.1 Interpretação dos Estados das Variáveis

Esse processo recebe como entradas a instrução para interpretação dos estados de cada variável de medição e a última leitura dos dessas variáveis.

A interpretação do estado de uma variável pode realizada de uma forma mais simplificada, por exemplo por meio de um escalonamento de valores em que para cada faixa de valores é atribuído um estado para a variável, ou mais complexa, podendo ser utilizados, pro exemplo, modelos que empregam inteligência artificial. Nas aplicações biomédicas, um paciente pode ter o estado interpretado da sua frequência cardíaca, por exemplo, por um serviço que leva em consideração se o paciente está em uso de betabloqueador, qual a idade, peso, altura e outras informações podendo assim fornecer o estado da variável de maneira mais confiável.

Os parâmetros de entrada desse processo são a mensagem de instruções para interpretação de dados e os valores atuais das variáveis.

A mensagem de instruções de interpretação do estado das variáveis pode conter diretamente a instrução de interpretação ou informar um serviço a ser acessado para interpretar o dado. O apêndice 1, apresenta código em XML do arquivo de interpretação de dados.

Conforme consta do referido apêndice, o estado da frequência respiratória pode ser interpretado diretamente pelas instruções do código XML. Caso a frequência respiratória esteja abaixo de 10 respirações por minuto, o núcleo de gerenciamento do MiLAN Services deve considerar o estado da variável como baixo. Se a frequência respiratória estiver entre 10 e 15 respirações por minuto, o estado da variável será normal. Se a frequência respiratória estiver superior a 15 respirações por minuto, o estado da variável deverá ser considerado como alto.

Para o caso da variável pressão sanguínea e da frequência respiratória, o código XML dá instruções ao MiLAN para consultar um serviço que interpretará o valor da variável. Na instrução é informada a URL do serviço bem como o método Web a ser chamado.

4.3.2 Avaliação do QoS Mínimo exigido por cada variável

Esse estado recebe como parâmetros de entrada os estados de cada variável bem como a instrução enviada pela aplicação para avaliar o QoS a ser exigido para cada variável de medição conforme os respectivos estados e o estado do sistema.

O código XML abaixo representa a mensagem encaminhada pela aplicação para definir quais os níveis de QoS exigidos para os estados do sistema e da variável. No caso da variável Pressão Sanguínea o parâmetro de QoS acurácia deverá ser de 90% quando o nível de estresse do paciente for baixo e quando o estado da variável dor alto, estado esse baseado na última leitura da variável. No código, *systemStates* representa o estado do sistema, *stateVariable* representa

o estado da variável, *variable* é a variável sendo considerada e *QoSParammeters* são os parâmetros de QoS considerados para um determinado estado da variável e um estado do sistema. É possível verificar no apêndice 2 as instruções para avaliação dos estados por meio de arquivo XML utilizado na implementação.

4.3.3 Seleção dos sensores possíveis para cada parâmetro de medição

Têm-se como parâmetros de entrada desse processo o QoS mínimo exigido por cada variável de medição e a mensagem da aplicação que informa quais são os sensores possíveis para realizar uma medição e qual o QoS dessa medição para aquele sensor.

Uma medição realizada por um sensor pode ser utilizada para fornecer mais de uma variável. Por exemplo, por meio de um sensor de fluxo sanguíneo é possível se conhecer a frequência cardíaca de um indivíduo e, por meio desse mesmo sensor, é possível inferir a pressão sanguínea com uma determinada acurácia, a uma acurácia inferior a de um sensor de pressão sanguínea, desenvolvido para esse fim. Essa forma de obter dados indiretamente de sensores é muito útil nas redes de sensores tendo em vista que a rede pode sofrer modificações durante o seu funcionamento como a perda de um nodo ou até mesmo em função de uma otimização do uso dos sensores para economia de energia a rede tenha opções de sensores para o fornecimento de dados e atendendo aos requisitos de QoS das aplicações.

O código XML constante do apêndice 3 informa como cada variável a ser solicitada por aplicações pode ser fornecida. A variável Frequência Respiratória (RespiratoryRate), por exemplo, pode ser fornecida pelo próprio sensor de frequência respiratória a uma acurácia de 100%, pelo sensor de ECG de três derivações, a uma acurácia de 80%, assim como os sensores de ECG de cinco e doze derivações.

4.3.4 Seleção dos sensores que fornecem dados para a aplicação

Neste processo, são levadas em consideração as condições da rede como largura de banda para transmissão para atender ao QoS exigido pela aplicação bem como as condições dos sensores, como o estado da bateria e se ele está ativo ou não. Esses dados são parâmetros de entrada do processo.

Antes de prosseguir com a análise do processo, é importante realizar uma breve discussão acerca do consumo de energia nas redes de sensores sem fio.

Um dos principais desafios, se não o maior dos desafios nas redes de sensores sem fio, é a construção e a operação de uma rede que atenda às exigências das aplicações e tenha o maior tempo de vida útil possível. Com aplicações cada vez mais complexas, que exigem uma quantidade de dados cada vez maior dessas redes, a frequências maiores e com maior acurácia, as redes de sensores sem fio são obrigadas a desprender maior uso energia, que em sua maior parte advêm de baterias, que sofreram pouca evolução tecnológica em relação aos dispositivos eletrônicos de comunicação e processamento. Por isso, o MiLAN Services também tem uma abordagem voltada à otimização do consumo de energia na rede de sensores sem fio.

Como os parâmetros de medição solicitados pelas aplicações podem ser fornecidos por mais de um sensor, o MiLAN Services tem a opção de escolher dentre os sensores possíveis de fornecer os dados aquele que proporcione maior economia de energia para a rede naquele momento, atendendo aos requisitos de QoS exigidos pela aplicação.

O modelo utilizado pelo MiLAN Services para a otimização do consumo de energia baseia-se na utilização dos nodos sensores conforme o seu nível de bateria, conjuntos de nodos e previsão de consumo pela aplicação. Também é considerado se o parâmetro a ser medido pode ser fornecido por mais de um sensor, com o QoS mínimo exigido ou se ele é fornecido por um "Sensor Virtual", que é um conjunto de sensores reunidos para fornecerem um único parâmetro.

Uma vez que um nodo sensor é ligado para fornecer um parâmetro de medição, outros parâmetros de medição podem sem inferidos, evitando-se assim que outros sensores sejam utilizados, o que resulta em menor consumo de energia. Por outro lado, cada sensor pode ter níveis de bateria diferentes, isso implica que o MiLAN Services deve ser capaz de escolher o melhor conjunto de sensores que atendem a um conjunto de parâmetros de medição, ou seja, para melhor otimização dos recursos de rede o módulo de gerenciamento deve realizar o processamento para otimização levando em consideração os conjuntos de sensores que consomem menos energia e que têm maior energia disponível e não apenas o sensor com maior energia disponível.

4.3.5 Camadas do sistema

A figura 4.3.5.1 mostra uma representação em camadas da implementação do MILAN Services.

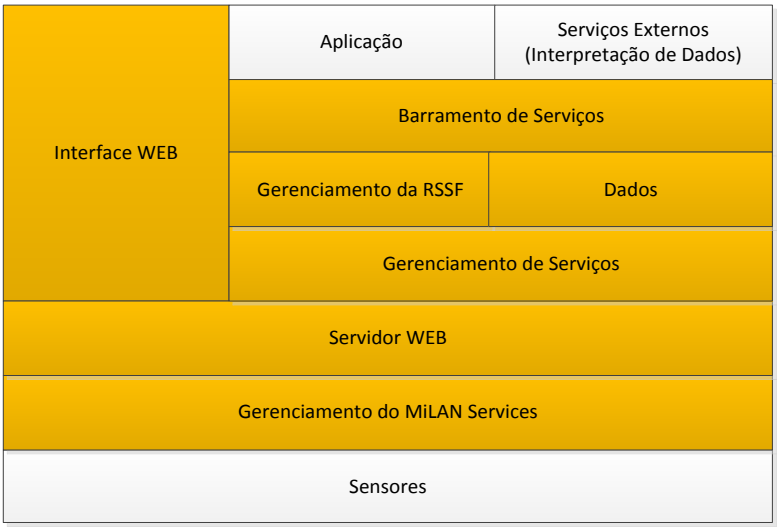


Figura 4.3.5.1 Diagrama de camadas do MiLAN Services.

Os blocos na cor laranja indicam as camadas em que o MiLAN Services atua diretamente. Na base estão os sensores, que são a parte física do sistema e cuja implementação foi simulada por software, o que e será mais bem detalhado adiante. A camada de gerenciamento do MiLAN Services atua na constituição da rede, gerenciamento dos nodos sensores e transmissão de dados de gerenciamento e de sensoriamento. É também nessa camada que são recebidas as mensagens das aplicações que dão as instruções sobre seus níveis de priorização, QoS e seus estados. Essas mensagens são processadas e são selecionados os sensores que melhor atendem às aplicações e que resultem no melhor uso dos recursos da rede. A camada imediatamente superior é a do servidor Web em que a implementação foi escolhido o *Glassfish*. Foi disponibilizada uma interface Web para que o sensor pudesse ser gerenciado, de forma simplificada, por usuários finais sem a intermediação de uma aplicação.

O gerenciamento dos serviços do MiLAN Services é o responsável por ativar ou desativar serviços conforme as condições da rede e requisições das aplicações. São disponibilizados serviços relacionados ao gerenciamento do nodo/RSSF e dados de sensoriamento que são consumidos pela camada de barramento de serviços. É nessa camada que é disponibilizada uma WSDL, que descreve quais são os serviços disponíveis pela rede. A aplicação, de posse da WSDL, sabe exatamente quais são e como consumir os serviços disponíveis. No mesmo nível da camada de aplicação também estão os serviços externos à rede que na implementação são usados na interpretação de dados da rede, tanto dados de sensoriamento como dados de interpretação do estado dos parâmetros de medição.

4.3.6 Simulação do Sensoriamento

Foram implementados simuladores de sensores da forma mais próxima de sensores reais para que a camada superior, de gerenciamento do MiLAN Services, não necessitasse de alterações com a inclusão de sensores reais ao projeto. A figura 4.3.6.1 representa o diagrama de classes dessa implementação.

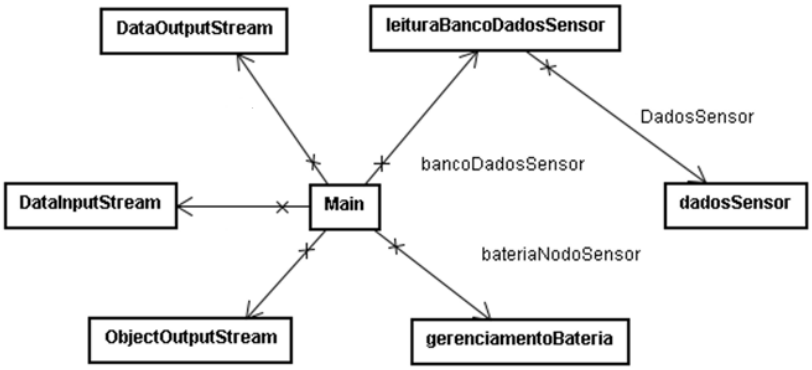


Figura 4.3.6.1 Diagrama de classes do nodo sensor.

Os sensores simulados foram fluxo sanguíneo, pressão sanguínea, oximetria de pulso, oxigênio, posição, ECG, frequência cardíaca e frequência respiratória.

Na classe Main foram implementadas *threads* para que, independentemente de requisições de dados por parte das aplicações, o sensor tivesse um sensoriamento independente. Os dados sensoriados são provenientes de arquivos de texto criados a partir de dados reais de sensores do projeto PhysioNet – Reserch Resource for Complex Physiologic Signals. Cada leitura recebe um selo de tempo, que tem como principal motivo a sincronia dos tempos de leitura de outros parâmetros de medição. Essa sincronização é de fundamental importância na fusão de dados, por exemplo, para que os dados fundidos correspondam ao mesmo instante de medição. Um objeto, nomeado de dadosSensor, é construído pela classe Main, para transmissão de cada leitura para a camada de gerenciamento do MiLAN Services. A transmissão dessa mensagem se dá por meio de sockets com porta e endereçamento definidos por arquivo de propriedade, tanto na camada do sensor quanto na camada de gerenciamento do MiLAN Services. Segue abaixo a figura 4.3.6.2, que mostra o diagrama da classe dadosSensor.

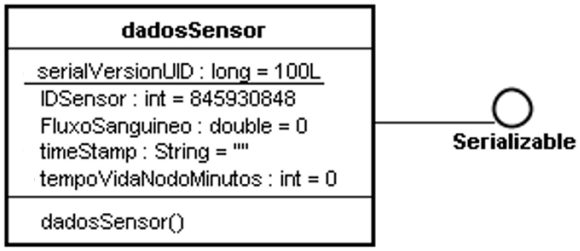


Figura 4.3.6.2 Diagrama da classe dadosSensor.

Na referida classe, apresenta-se as variáveis serialVersioUID, que é um identificador de versão da classe a ser serializada; IDSensor, em que se define o código do sensor; FluxoSanguíneo, em que se registra o valor instantâneo da variável fluxo sanguíneo; timeStamp, em que se registra o selo de tempo do momento da aquisição da variável, e tempoVidaNodoMinitos, que registra o tempo de vida do nodo sensor em minutos. Ainda, a classe contém o método dadosSensor, que pode ser acionado para se informar os dados de sensoriamento do sensor.

A taxa de amostragem da leitura do sensor é definida por meio de mensagem de gerenciamento envidada pela camada de gerenciamento do MiLAN Services.

Para que fosse simulado o consumo do nodo sensor conforme leituras fossem realizadas e requisições de dados fossem enviadas, foi construída uma classe que reproduzisse o consumo da bateria a cada leitura e transmissão de dados. Segue abaixo, figura 4.3.6.3, o diagrama da classe gerenciamentoBateria.

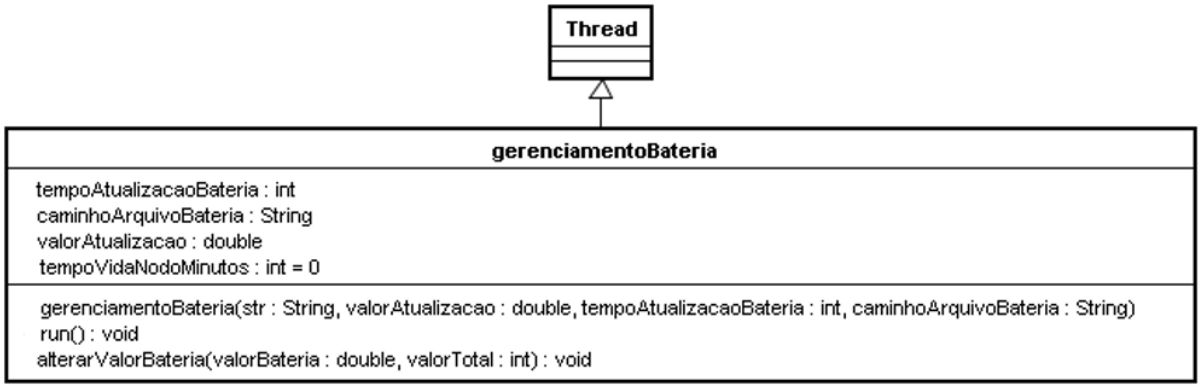


Figura 4.3.6.3 Diagrama da classe gerencamentoBateria.

Como discutido anteriormente, para que a leitura de dados do sensor funcione de maneira independente da requisição de dados do nodo sensor, a uma taxa de amostragem definida, a construção do simulador do nodo foi baseada em *thread*, ou seja, instâncias de execução daquela parte do algoritmo de forma paralela e independente. Esses dados, que são obtidos de arquivos de dados pela classe leituraBancoDadosSensor, representada pela figura 4.3.6.4.

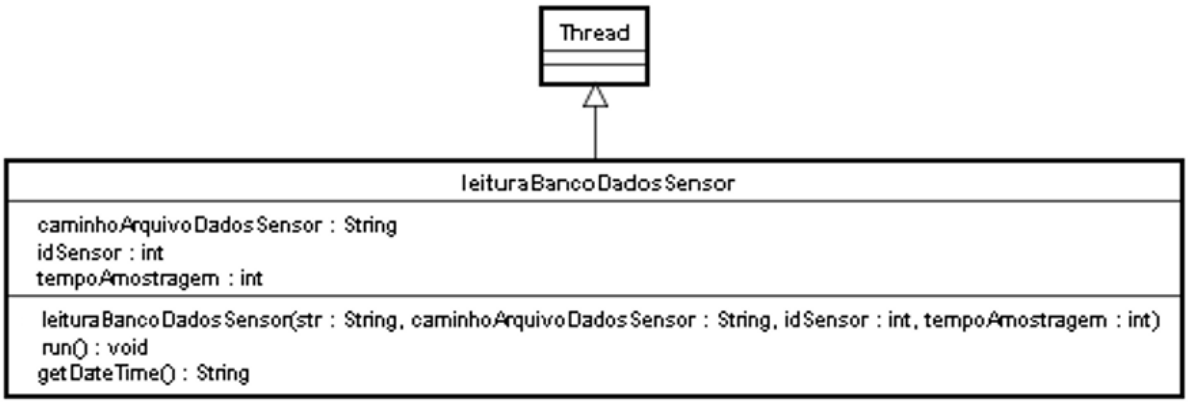


Figura 4.3.6.4 Diagrama da classe leituraBancoDadosSensor.

4.4 MÓDULO DE GERENCIAMENTO DO MiLAN SERVICES

4.4.1 Interface de Gerenciamento do Nodo Sensor

O Módulo de gerenciamento do MiLAN Services exerce duas atividades principais: A de receber os dados provenientes dos sensores e encaminhar para o módulo de gerenciamento de serviços e a de controlar os recursos da rede conforme as prioridades e níveis de QoS requeridos pelas aplicações, além das condições rede.

O diagrama de classes representado pela figura 4.4.1.1 mostra a classe Main do módulo no interfaceamento com os sensores da rede. A classe utiliza objetos de streaming para a transmissão do objeto de leitura do sensoriamento, além de um cliente socket utilizado para a comunicação com os sensores.

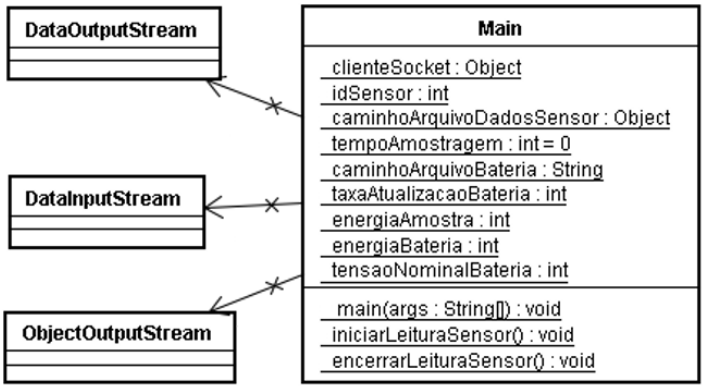


Figura 4.4.1.1 Diagrama da classe Main do módulo de gerenciamento.

Na classe de controle, o código utilizado na implementação do gerenciamento do nodo sensor é abstraído na forma de funções para uso nas camadas superiores. A figura 4.4.1.2 mostra o diagrama da classe controle.

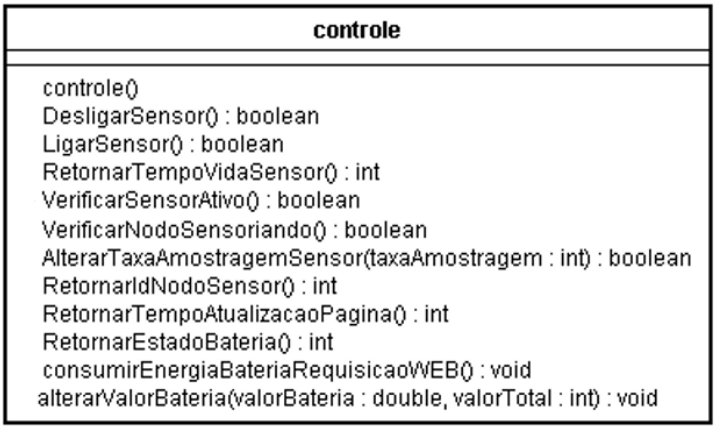


Figura 4.4.1.2 Diagrama da classe controle.

A implementação da rede simulada cuidou de serem definidos, por meio de arquivos de propriedades da aplicação respectiva, tanto para os sensores quanto para o módulo de gerenciamento do MiLAN, parâmetros para a configuração da porta de comunicação do socket, da identificação do nodo sensor, da taxa de amostragem, do valor de energia desprendida para se obter e transmitir a leitura de dados e da tensão nominal da bateria.

Para facilitar a manipulação dos dados do arquivo de propriedade, foi criada uma classe que disponibiliza métodos para que sejam retornados os dados do arquivo. Essa classe foi chamada de Propriedades, como mostra a figura 4.4.1.3.

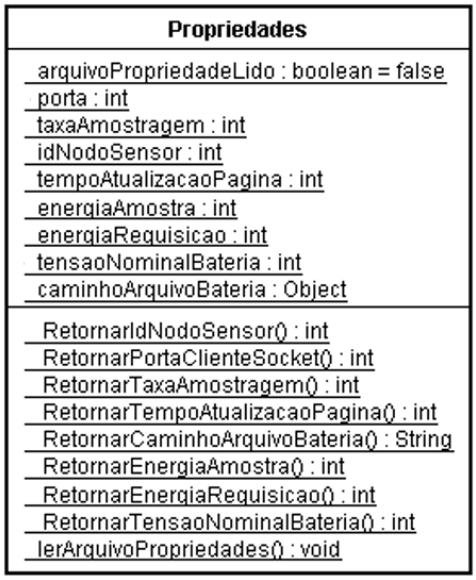


Figura 4.4.1.3 Diagrama da classe propriedades.

Além disso, foi criada uma classe que cuida da abertura, fechamento e tratamento de erros do arquivo chamada de PropriedadesArquivo representada pela figura 4.4.1.4. O tipo de variável

simputStream foi definido para que seja possível a instanciar a variável com dados provenientes de arquivo de sistema.

Figura 4.4.1.4 Diagrama da classe PropriedadesArquivo.

Para abstrair as funções de comunicação entre o gerenciamento do MiLAN Services e o nodo sensor, foi crida a classe *clientSocket*, contida no diagrama ilustrado na figura 4.4.1.5. A classe concentra funções para retornar os dados do nodo sensor e para controlar o nodo como ligar e desligar o sensoriamento e alterar a taxa de amostragem.

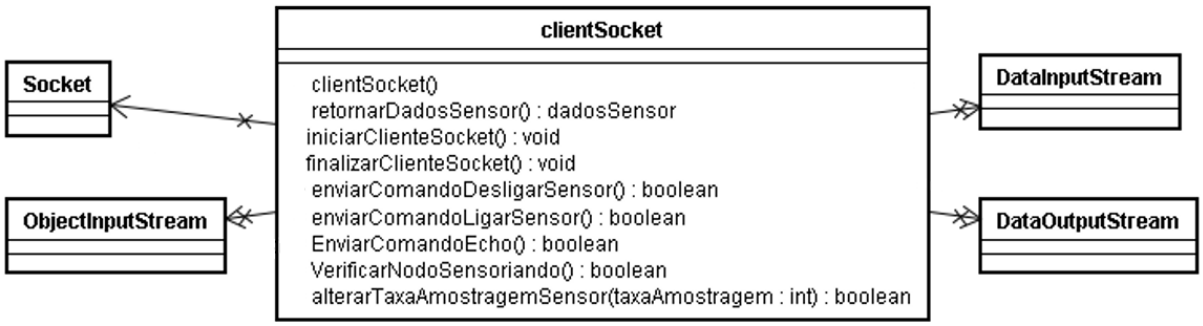


Figura 4.4.1.5 Diagrama da classe clientSocket.

4.4.2 Núcleo de Seleção e Controle de Nodos Sensores

O processo de seleção de nodos sensores, já descrito anteriormente, tem como função selecionar os sensores que fornecem dados às aplicações conforme os níveis de QoS exigidos, estado das aplicações, níveis de prioridade dessas aplicações e condições da rede. A implementação foi realizada em uma aplicação que primeiramente verifica os sensores ativos e a última leitura dos mesmos, verifica o estado da bateria desses sensores, recebe as mensagens da aplicação quanto aos níveis de QoS requeridos, estado da aplicação e prioridades, processa a seleção, liga ou desliga o sensoriamento dos nodos sensores e dispara mensagens para o núcleo de gerenciamento de serviços para atualizações nos objetos de serviço.

O núcleo da seleção de sensores no módulo de gerenciamento de no MiLAN Services é realizado pela classe Main que está representada na figura 4.4.2.1.

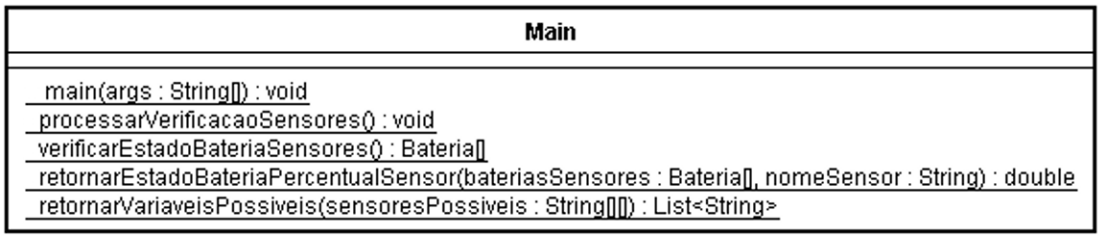


Figura 4.4.2.1 Diagrama da classe Main do núcleo de seleção dos nodos sensores.

Para facilitar a manipulação de mensagens entre métodos no tratamento do consumo da bateria e por se tratar de um objeto complexo, com mais de um tipo de dados a ser tratado, foi criada uma classe Bateria como mostra a figura 4.4.2.2.

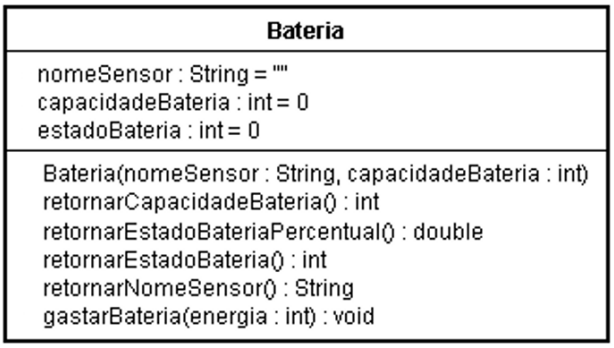


Figura 4.4.2.2 Diagrama da classe Bateria.

A classe SensorQoS, representada pela figura 4.4.2.3, foi criada com o intuito de concentrar em um objeto as funcionalidades relacionadas com a análise do conjunto de sensores possíveis em função do QoS requerido pela aplicação e pelo estado das variáveis.

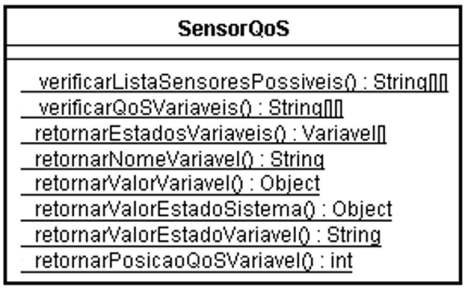


Figura 4.4.2.3 Diagrama da classe SensorQoS.

Seguem algumas classes que foram criadas para representar objetos auxiliares utilizados pela classe Main. A classe EstadoSistema, figura 4.4.2.4, auxilia na informação do estado do sistema, a classe Variável, figura 4.4.2.5, constitui os dados de uma variável de medição, e a classe VariavelSensor, figura 4.4.2.6, que representa a variável de medição de um sensor.

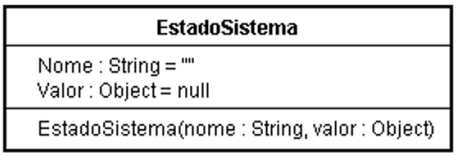


Figura 4.4.2.4 Diagrama da classe EstadoSistema.

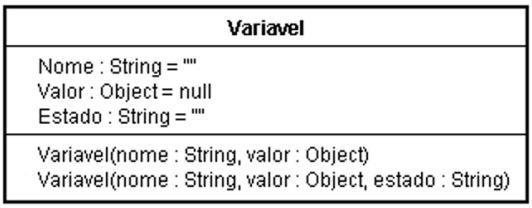


Figura 4.4.2.5 Diagrama da classe Variável.

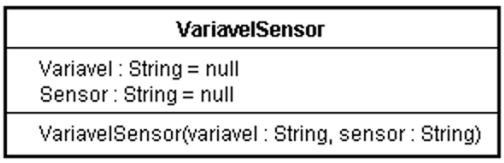


Figura 4.4.2.6 Diagrama da classe VariavelSensor.

4.5 SERVIÇOS NO NODO SENSOR

Cada nodo sensor possui um servidor Web embarcado. A escolha de se ter um servidor Web em cada nodo vai ao encontro do modelo proposto nesse estudo, que é permitir que tanto os nodos sensores quanto toda a rede de sensores sem fio estejam dentro do conceito de SOA.

Com isso, é possível que em uma rede heterogênea vários sensores possam interagir entre si, independente de fabricante, pois os conceitos de serviços são aplicados a todos quando implementado o MiLAN Services. Assim, o conceito de serviços não foi implementado no projeto para que somente dados da rede fossem consumidos por aplicações externas ou internas por meio de serviços, mas também implementado em cada sensor tornando-o como um serviço que disponibiliza métodos para fornecimento de dados de sensoriamento bem como métodos para o seu gerenciamento. A figura 4.5.1 representa uma classe do sensor de fluxo sanguíneo que disponibiliza o método GetBloofFlow para o fornecimento do valor instantâneo do fluxo sanguíneo.

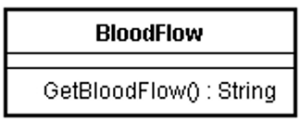


Figura 4.5.1 Diagrama da classe BloodFlow.

A figura 4.5.2 representa a classe BloodFlowAdmin que fornece métodos para a administração do sensor. Entre esses métodos o que permite ligar o sensoriamento, desligar o sensoriamento e retornar o estado da bateria do nodo sensor.

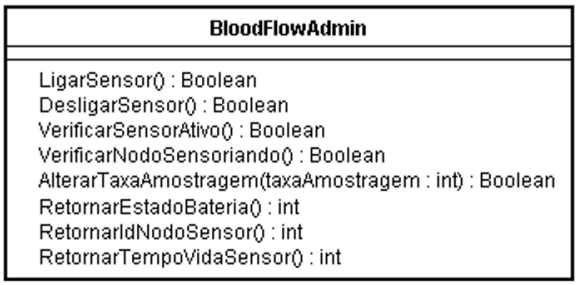


Figura 4.5.2 Diagrama da classe BloodFlowAdmin.

Abaixo a figura 4.5.3 mostra a interface WEB de gerenciamento do nodo sensor. Desenvolvida em JSP, tem o objetivo de disponibilizar recursos simplificados de gerenciamento sem a implementação de aplicações. Foram disponibilizados os recursos de ligar e desligar o sensoriamento do nodo sensor, visualizar a última leitura do sensor com o seu selo te tempo e também o verificar o tempo de vida do nodo sensor.

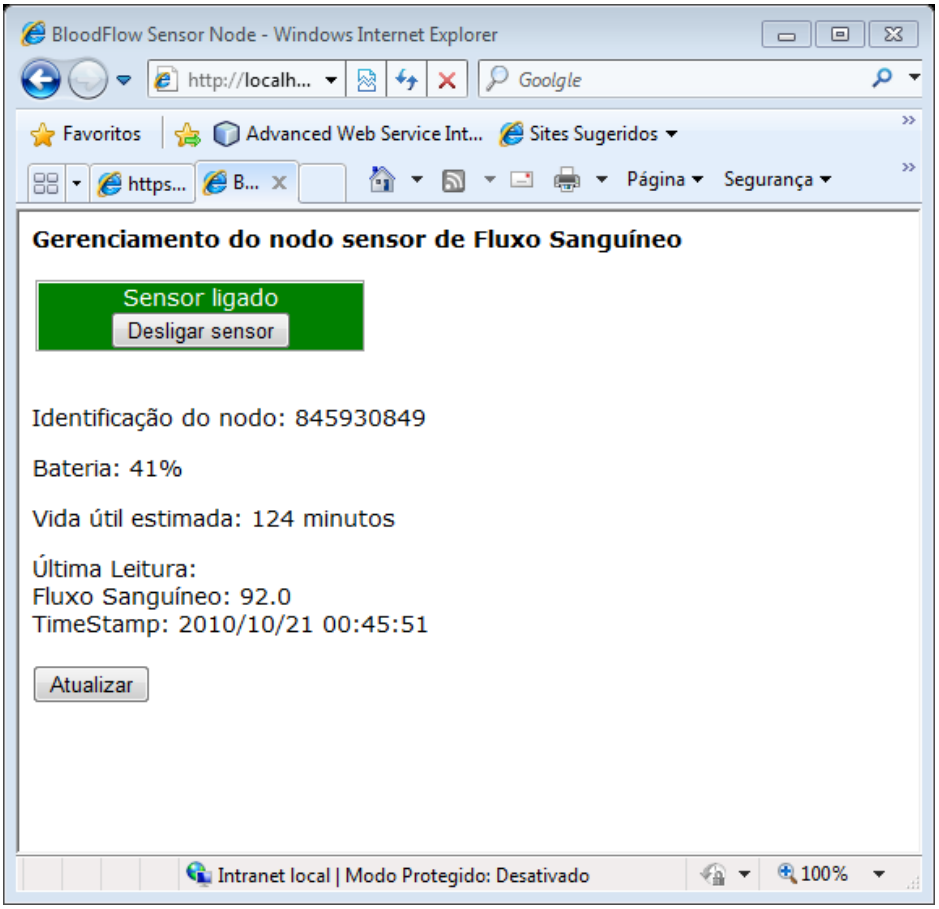


Figura 4.5.3 Interface Web de gerenciamento do nodo sensor de fluxo sanguíneo.

O servidor Web GlassFish disponibiliza um recurso para testar o serviço Web instalado, assim, é possível verificar por meio de uma interface de aplicação os serviços do sensor e realizar chamadas em seus métodos. A figura 4.5.4 mostra essa interface.

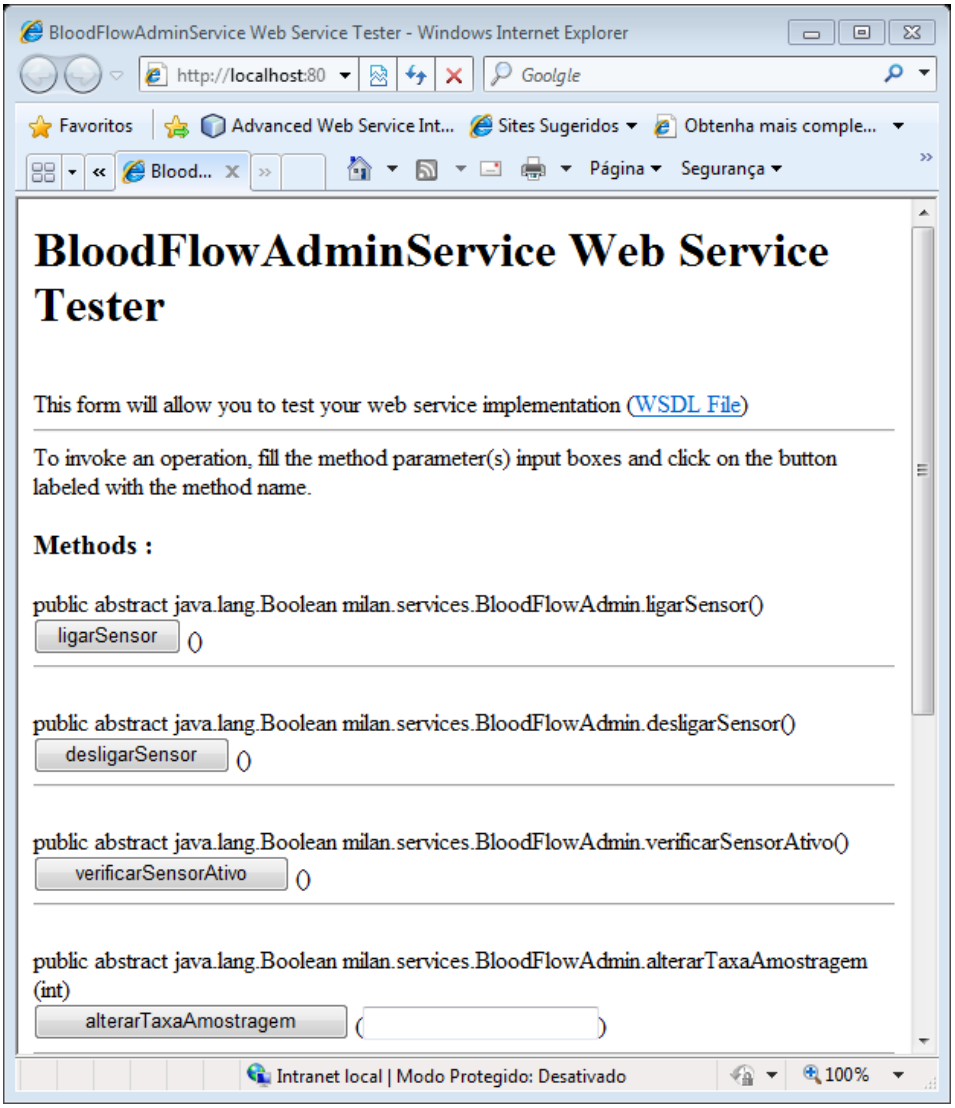


Figura 4.5.4 Interface Web de teste dos serviços de gerenciamento do nodo de fluxo sanguíneo.

O serviço de cada nodo sensor disponibiliza uma WSDL – Web Service Description Language para que a rede e as aplicações saibam como interagir com o nodo. O apêndice 4 apresenta o código XML do WSDL referente ao nodo sensor de fluxo sanguíneo.

Ainda na interface Web Disponibilizada pelo GlassFish, são mostradas como as mensagens SOAP devem ser enviadas e como elas são recebidas. Isso é bastante interessante, pois permite que, com poucas linhas de código e sem o uso de bibliotecas complexas, seja possível criar um cliente de mensagens SOAP e consumir serviços Web. Segue abaixo as mensagens de

requisição e de resposta do método ligarSensor do serviço de gerenciamento do nodo sensor de fluxo sanguíneo:

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header/>
   <S:Body>
      <ns2:ligarSensor xmlns:ns2="http://Services.MiLAN/"/>
   </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:ligarSensorResponse xmlns:ns2="http://Services.MiLAN/">
         <return>true</return>
      </ns2:ligarSensorResponse>
   </S:Body>
</S:Envelope>
```

4.6 BARRAMENTO DE SERVIÇOS

Foi implementado um barramento de serviços no MiLAN Services utilizando os recursos disponíveis da extensão METRO do servidor Web Glassfish.

Como mencionado anteriormente, o MiLAN Services não se prende a uma tecnologia específica para implementação do conceito SOA. A utilização de Web Services no projeto se deu de forma a simplificar a implementação para que fosse possível avaliar aspectos práticos do conceito. A escolha do Glassfish como servidor Web teve como objetivo a utilização de sua extensão METRO, que possibilita a integração de serviços de plataformas heterogêneas como em ambientes Microsoft e Java. Isso é importante para que o MiLAN Services não se prenda a

uma tecnologia, mas a um conceito genérico e isso é possível utilizando-se o METRO tendo em vista que seus serviços são baseados no conceito de *endpoints*. Em outras palavras, por meio de *endpoints* cada serviço pode disponibilizar "portas" de comunicação com aplicações independente de tecnologia, bastando apenas que ela seja conhecida pela rede e pelo usuário que irá consumir o serviço, que haja um contrato e que seja especificado o endereço de acesso. Esse tripé é a base para o conceito de *endpoints*.

Na implementação foram utilizados os recursos já existentes no Glassfish, portanto, não foram desprendidos esforços que necessitam de maior detalhamento, pois a criação de serviços necessita simplesmente de uma implementação de um método de uma classe que leva um atributo de Web Service. A figura 4.6.1 mostra os métodos construídos na classe MiLANServices, que por meio dos recursos do Glassfish, disponibiliza serviços para o barramento de serviços.

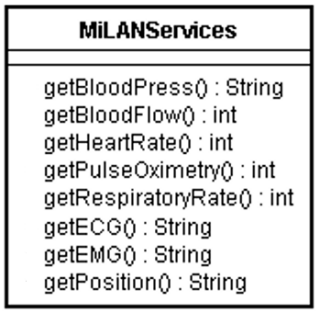


Figura 4.6.1 Diagrama de Classe MiLANServices descrevendo os métodos do serviço.

É possível também testar os serviços do barramento por meio de uma interface Web disponibilizada pelo servidor Web. Por meio dessa interface é possível disparar os métodos Web e obter a lista de serviços, WSDL. A figura 4.6.2 ilustra a interface de teste do serviço.

Figura 4.6.2 Interface WEB de teste dos serviços da rede de sensores sem fio.

O apêndice 5 apresenta partes do código WSDL dos serviços disponibilizados no barramento de serviços em que verifica: as definições de segurança e o esquema XSD, os dados necessários para utilização dos métodos Web do serviço bem como a estrutura que define a sua resposta, os dados de endereçamento de cada operação do serviço MiLANServices, para a transmissão de mensagens de requisição e de resposta, os protocolos de comunicação para cada operação (para o caso do projeto em comento, utilizou-se o SOAP) e o endereço de acesso ao serviço MiLANServices, especificado, para o caso em referência, o *endpoint* MiLANServicesService, pela porta 8080 (porta de internet).

4.7 INTERPRETAÇÃO DE DADOS

Dados provenientes dos sensores podem receber uma interpretação a ser disponibilizada na forma do serviço no MiLAN Services. Essa interpretação pode ser realizada por um aplicativo que usa de recursos como simples cálculos ou recursos mais complexos como o uso de inteligência artificial. Como não é esse o objetivo de funcionamento do MiLAN Services, mas ao mesmo tempo deve ser levado em consideração no projeto, foi implementado o recurso de utilização de serviços externos para manipulação ou interpretação de dados. Assim, o MiLAN Services, conforme instruções enviadas pela aplicação, ao receber um dado de um nodo sensor, pode realizar a chamada de um serviço externo repassando o dado do nodo e esse serviço interpreta e retorna a informação ao MiLAN Services. O MiLAN Services por sua vez cria um serviço em sem barramento de serviços com a informação.

Assim, foi implementado um serviço no projeto que realiza a função de um serviço de interpretação de dados. Foram criados alguns métodos para simplesmente testar a funcionalidade de uso de serviços externos e pode ser perfeitamente substituído por serviços mais complexos sem ônus para a rede de sensores bastando apenas constar nas instruções de interpretação de dados da aplicação. Segue abaixo a representação da Classe Analysis, figura 4.7.1, e AnalysisPort, figura 4.7.2, que são parte da estrutura do sistema de interpretação de dados.

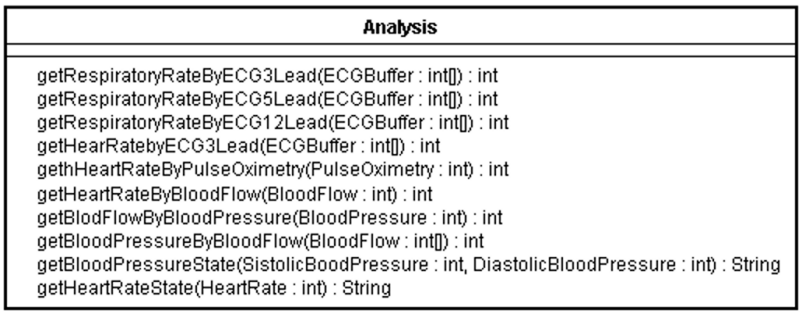


Figura 4.7.1 Diagrama de classes Analysis descrevendo os métodos do serviço de interpretação de dados.

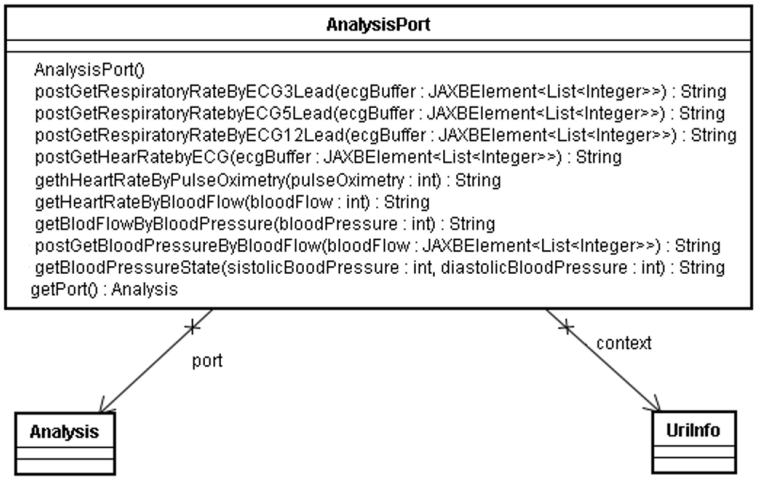


Figura 4.7.2 Diagrama da classe AnalysisPort do serviço de interpretação de dados.

4.8 APLICAÇÃO

A aplicação foi desenvolvida utilizando-se a tecnologia Java FX visando a uma interface rica, o que possibilite melhor uso de recursos gráficos. A figura 4.8.1 mostra a interface gráfica da aplicação.

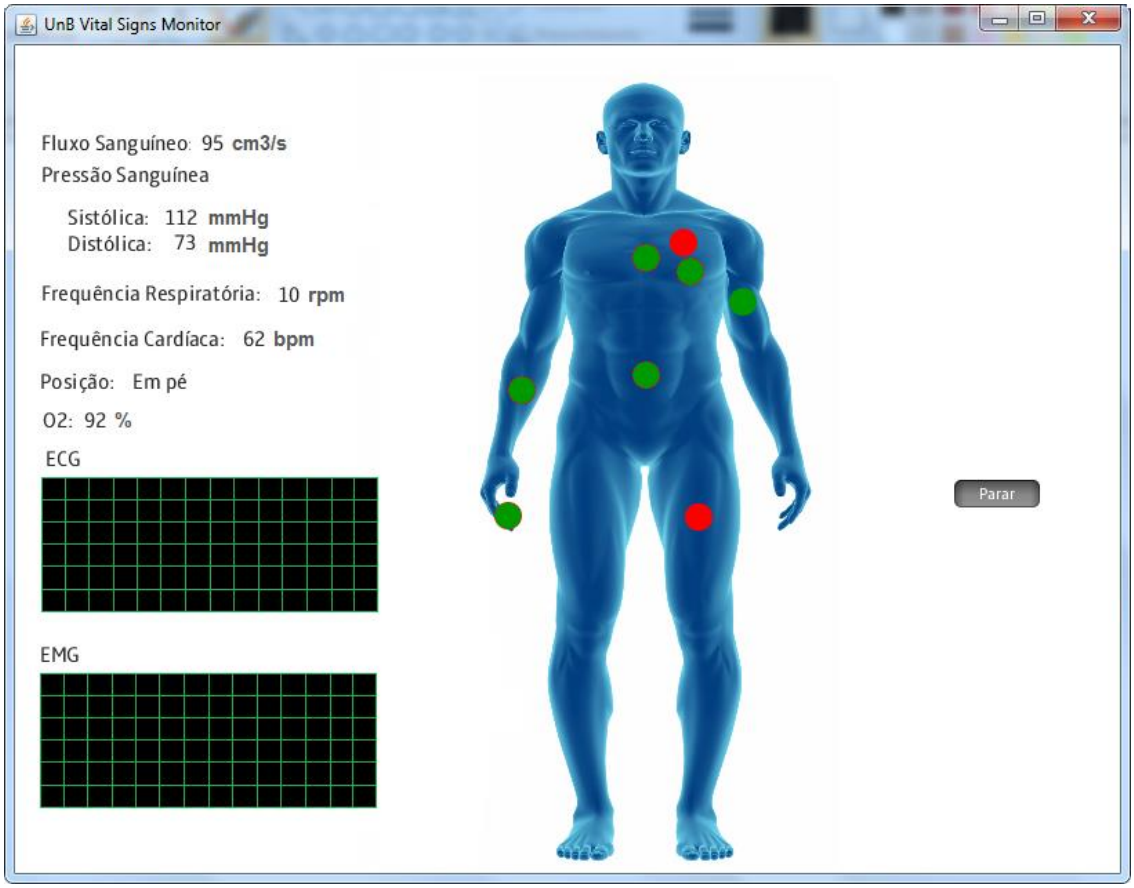


Figura 4.8.1 Interface gráfica da aplicação usuária da rede de sensores sem fio.

Foi criada um cliente de serviços Web que é utilizado pela aplicação, implementada na classe UnBVitalSignsMonitorClientService. Na aplicação é possível verificar os sensores ativos e inativos e verificar os dados dos sensores. Os parâmetros de inicialização como o nível de QoS exigido para cada parâmetro de medição, o nível de prioridade da aplicação, o grafo de QoS e sensores e o estado da aplicação também são enviados pela aplicação na forma de mensagens XML. Toda a comunicação entre a aplicação e a rede de sensores é realizada por meio de Web Services, por intermédio do MiLAN Services.

5 RESULTADOS

A metodologia para aferição dos resultados do MiLAN baseou-se em dois parâmetros: o número de conjuntos de sensores possíveis para atender às variáveis exigidas pelo sistema em uma determinada combinação de estados das variáveis e o número médio de sensores por conjunto. Para o primeiro caso, quando maior o número de conjuntos de sensores possíveis, maior a resiliência da rede, ou seja, mais tolerante a falhas é o sistema. Para o segundo caso, quanto menor o número de sensores por conjunto, menor é o consumo de energia, e por consequência, maior será o tempo de vida útil da rede, pois outros sensores não utilizados poderão ser recrutados a medida que outros sensores não tiverem energia suficiente para atender à rede.

Na simulação, foram três as variáveis exigidas pela aplicação: pressão sanguínea, frequência cardíaca e frequência respiratória.

O nível de serviço exigido para cada variável depende do estado do sistema e do estado de cada variável. Para fins de simplificação, o estado do sistema foi fixado em "baixo estresse", que é um dos três possíveis estados (baixo, moderado e alto).

Os estados possíveis para cada variável podem ser baixo, normal e alto (L, N e H, respectivamente). Dessa forma, o número de combinações possíveis para os estados das variáveis podem ser 27, conforme apresentam as tabelas 1 e 2 do apêndice 6. Um exemplo de combinação seria: pressão sanguínea alta, frequência respiratória normal e frequência cardíaca alta (HNH).

Os resultados são comparados com os de uma rede convencional. Para isso, foi considerado que os conjuntos de sensores possíveis para atender às três variáveis definidas pela aplicação são aqueles originalmente constituídos para atender a um parâmetro específico, sem a utilização de sensor virtual, fusão de dados ou outro recurso indireto de aquisição de dados, como ocorre no MiLAN. Dessa forma, a variável pressão sanguínea dever ser aferida pelo sensor de pressão sanguínea, a saturação e O2 deve ser obtida pelo sensor SPO2, a frequência cardíaca pelo sensor de oximetria de pulso e assim por diante.

5.1 AVALIAÇÃO DOS RESULTADOS

Para todas as combinações de estados baixo, normal e alto para as ariáveis pressão sanguínea, frequência cardíaca e frequência respiratória obtêm-se os conjuntos de sensores possíveis para atender aos requisitos da aplicação conforme apresenta o gráfico constante da figura 5.1.1.

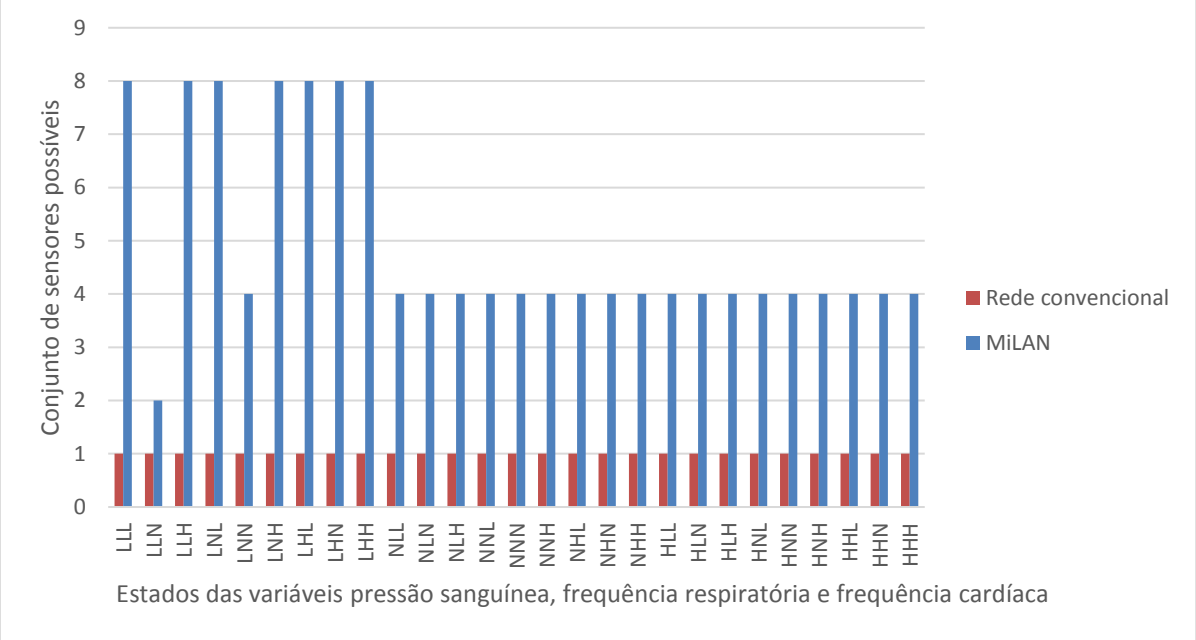


Figura 5.1.1 Comparação entre a rede MiLAN e uma rede convencional em relação aos conjuntos de sensores possíveis

Conforme se verifica do gráfico acima, na rede convencional, é possível estabelecer apenas um conjunto de sensores para cara combinação de estado das variáveis ao passo que na rede MiLAN alcança-se até oito conjuntos possíveis por combinação de estado. Isto implica dizer que há maior adaptabilidade da rede MiLAN a possíveis falhas em sensores podendo ser selecionados outros conjuntos de sensores, sendo, portanto, mais resiliente quando comparada a uma rede convencional.

O gráfico apresentado na figura 5.1.2 mostra o número médio de sensores por conjunto de sensores possíveis para cada combinação de estado das variáveis.

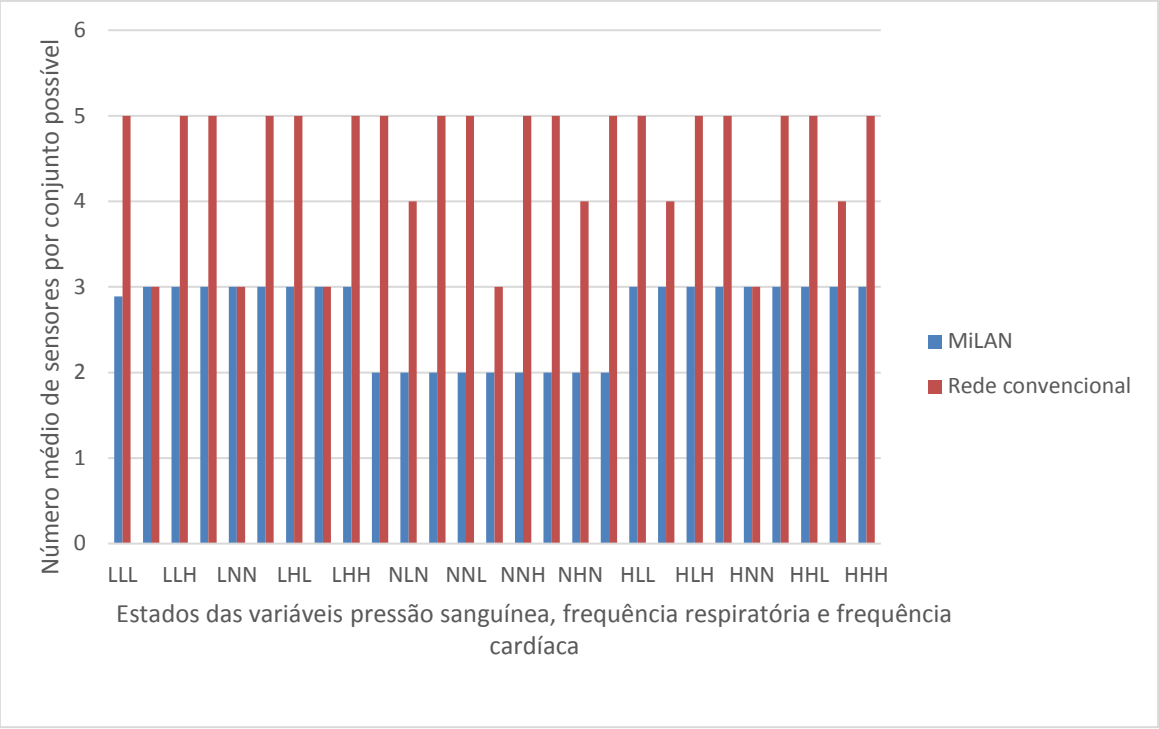


Figura 5.1.2 Comparação entre a rede MiLAN e uma rede convencional em relação ao número médio de sensores por conjunto

Verifica-se no gráfico acima que o número médio de sensores por conjunto de sensores possíveis na rede MiLAN é menor que em uma rede convencional. Para a aplicação em comento, o número médio de sensores em todas as combinações de estado das variáveis para o MiLAN é de 4,48 e para uma rede convencional é de 2,66. Pode-se concluir, portanto, que o MiLAN promove o uso mais eficiente dos recursos de rede e, portanto, para o caso do uso da energia dos nodos sensores, aumenta a vida útil da rede.

5.2 LIMITAÇÕES DO TRABALHO

Não foram avaliados no trabalho os impactos decorrentes do uso dos recursos da rede a partir de mais de um usuário. Num sistema multiusuário, são necessárias abordagens que levem em conta esquemas de priorização, já que alguma aplicação pode ser mais crítica que outra, que alguma aplicação pode consumir mais recursos e que, o critério de seleção do conjunto de sensores deve considerar maneiras de aumentar a vida útil da rede. Portanto, o método de hierarquização das aplicações tem impacto direto no consumo de energia dos nodos sensores.

Outra questão é que quando na não existência de um conjunto de sensores que atenda aos requisitos da aplicação, seja pelo nível elevado de QoS requisitado ou pela perda de nodos sensores, pode haver a possibilidade de a aplicação reduzir os requisitos de nível de serviços afim de manter um mínimo de informação que pode ser utilizada pelo sistema. Todavia,

ressalta-se que essa abordagem dependerá da aplicação pois dados provenientes da rede que possuam QoS inferiores aos definidos pela aplicação podem resultar em tomadas de decisão que menor nível de assertividade.

Verifica-se no trabalho que as principais vantagens do MiLAN estão associadas ao uso mais eficiente e ao aumento da resiliência da rede de sensores bem como à simplificação da implementação de redes de sensores heterogêneas. No entanto, há possíveis limitações no que diz respeito a aplicações multiusuários. Uma abordagem que leve em conta a ponderação em relação ao grau de prioridade e de uso de recursos da rede de sensores pode ser necessária para a hierarquização das aplicações usuárias da rede.

6 CONCLUSÕES

Foi modelada e implementada com sucesso uma prova de conceito do MiLAN bem como uma camada de integração de serviços utilizando-se de ferramentas amplamente difundidas, que possibilitou exemplificar o consumo de dados e informações entre a rede de sensores simulada e a aplicação biomédica utilizando-se serviços.

A rede de sensores implementada utilizou os conceitos de SOA por meio de objetos de serviço que são expostos por *endpoints* com endereço, *bindings* e contratos, e são listados por uma WSDL, no padrão Web Service Interoperability Technologies – WIST, o que possibilita maior integração com diferentes plataformas.

As principais funcionalidades do MiLAN foram implementadas, entre elas o gerenciamento da rede para atender às especificações das aplicações, que informam ao MiLAN Services os níveis de prioridade e de QoS requeridos.

Demonstrou-se que o MiLAN proporciona maior resiliência às redes de sensores sem fio bem como otimiza o uso dos recursos rede, mormente quanto ao uso da energia que, por consequência, aumenta a vida útil do sistema.

Verifica-se a necessidade crescente de que as redes de sensores sem fio se integrem cada vez mais com um maior número de aplicações e que os dados por elas fornecidos ultrapassem a barreira da rede e passem a trafegar em redes diversas, inclusive na internet.

A inclusão de uma camada de serviços ao MiLAN sobre a ótica de arquitetura orientada a serviços provê a essas redes a possibilidade de serem mais flexíveis, mais integradas e com uma maior gama de recursos, já que serviços externos podem ser consumidos pela rede e, portanto, provendo serviços a partir de sensores virtuais.

Além disso, constatou-se que a implementação de SOA em redes de sensores oferece um ganho de escalabilidade e diminuição do esforço de programação na implementação de aplicações que utilizem os serviços da rede.

Assim essa dissertação mostrou que o MiLAN Services é uma forma viável de se unir o mundo físico ao mundo virtual, provendo por meio das redes de sensores, serviços dinâmicos e flexíveis.

7 TRABALHOS FUTUROS

Em uma era em que cada vez mais diversos os dispositivos que fazem parte do cotidiano das pessoas passam a se conectarem, tal como nos paradigmas redes vestíveis e internet das coisas, muitos aspectos relacionados a essas redes, inclusive nas redes de sensores sem fio, se tornam objeto de estudo, tais como, protocolos de comunicação, segurança, roteamento, eficiência energética, dentre outros.

No MiLAN Services, protocolos para disponibilização e uso de serviços com menores *oveheads* trariam melhores benefícios em termos de consumo de energia da rede.

Ainda nesta linha, a implementação de algoritmo que selecione o conjunto de nodos sensores considerando uma melhor a avaliação sobre os métodos e a implementação de algoritmos no MiLAN Services para seleção de conjunto de nodos sensores levando em consideração a análise de consumo ótimo de energia da rede.

Necessário se faz avaliar algoritmos que selecionem o conjunto de sensores para atender ao QoS das aplicações e, simultaneamente, avaliem as condições da rede em termos de transmissão e de energia disponível em cada sensor de tal forma que maximize a vida útil da rede.

Uma avaliação dos impactos da implementação de protocolos de segurança no que diz respeito à garantia da integridade das mensagens trafegadas na rede, à autenticidade das mensagens e à autorização para a realização de determinadas tarefas na rede são assuntos de grande relevância nas RSSF's, haja vista a sua escala de aplicação cada vez mais crescente e as restrições inerentes a essas redes.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] THAM, C. K.; BUYYA, R. SensorGrid: Integrating Sensor Networks and Grid Computing. Computer Society of India, Julho de 2005.

[2] CERPA, A.; ELSON, J.; HAMILTON, M.; ZHAO, J.; ESTRIN, D.; GIROD, L. Habitat monitoring: application driver for wireless communications technology. Workshop on Data communication in Latin America and the Caribbean. San Jose, Costa Rica, Abril de 2001.

[3] HILL, J., SZEWCZYK R.; WOO, A., HOLLAR, S., CULLER, D.; PISTER K. System Architecture Directions for Network Sensors. 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 93–104, Cambridge, MA, USA, November 2000.

[4] MURPHY A. L.; PICCO, G.P.; ROMAN, G. C. Lime: A Middleware for Physical and Logical Mobility. In Proceedings of the 21st International Conference on Distributed Computing Systems, pages 524–533, April 2001.

[5] CORDEIRO, C. M.; AGRAWAL, D. P.. Ad hoc and Sensor Networks, Theory and Applications. World Scientific, 2006.

[6] GELERNTER, D. Generative Communication in Linda. ACM Transactions on Programming Languages and Systems, 7(1):80–112, January 1985.

[7] OBJECT MANAGEMENT GROUP. The Common Object Request Broker: Architecture and Specification, Revision 2.3. 492 Old Connecticut Path, Framingham, MA 01701, USA, 1999.

[8] HOLDER, O.; BEN-SHAUL, I.; GAZIT, H. System Support for Dynamic Layout of Distributed Applications. 19th International Conference on Distributed Conputing, páginas 403–411, 1999.

[9] COMITÊ DE ESPECIFICAÇÃO 1. Modelo de Referência para Arquitetura Orientada a Serviço 1.0. OASIS. Dezembro de 2012.

[10] CHU, X; BUYYA, R. Service Oriented Sensor Web. Grid Computing and Distributed Systems Laboratory, Dept. of Computer Science and Software Engineering, University of Melbourne, Australia, Janeiro de 2007.

[11] REZGUI, A.; ELTOWEISSY, M. Service-Oriented Sensor-Actuator Networks. IEEE Communications Magazine 2007.

[12] HEINZELMAN, W. B.; MURPHY, A. L.; CARVALHO, H. S.; PERILLO, M. A. Middleware to Support Sensor Network Applications. University of Rochester, Janeiro de 2004.

[13] MURPHY, A. L; HEINZELMAN, W. B. MiLAN: Middleware Linking Applications and Networks. University of Rochester, November 13, 2002.

[14] CONWAY, J.C.D.; FERNANDES, A.O.; COELHO C.J.N. Jr.; ANDRADE L.C.G.; SILVA D.C.; CARVALHO H.S. Wearable Computer as a Multi-parametric Monitor for Physiological Signals. In Proceedings of the IEEE International Symposium on Bioinformatics and Bioengineering (BIBE), páginas 236–242, 2000.

[15] LEE, K. IEEE 1451: A Standard in Support of Smart Transducer Networking. In IEEE Instruments and Measurements Technology Conference, May 2000.

[16] SINGH, S.; WOO, M., e RAGHAVENDRA, C.S. Power-Aware Routing in Mobile Ad Hoc Networks In Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom' 98), October 1998.

[17] KHACHIYAN, L. A. Polynomial Algorithm in Linear Programming. In Doklady Akademii Nauk SSSR 244, 1979.

[18] MOKARZEL, M. P. Internet Embedded TCP/IP para Microcontroladores. 1ª Ed. São Paulo, Erica, 2004.

[19] CHU, X.; BUYYA, R. Service Oriented Sensor Web. Grid Computing and Distributed Systems Laboratory, Dept. of Computer Science and Software Engineering, University of Melbourne, Australia, January de 2007.

[20] LÓPEZ, E. A.; GARCÍA-MACÍAS, J. A. Tiny SOA: a service-oriented architecture for wireless sensor networks. Verlag London, 2009.

[21] ALSHININA, R.; ELLEITHY, K. Performance and Challenges of Service-Oriented Architecture for Wireless Sensor Network Computer Science and Engineering Department, University of Bridgeportks. USA, March 2017.

[22] GUSTAFSSON, J.; KYUSAKOV, R.; MÄKITAAVOLA, H; DELSING, J. Application of Service Oriented Architecture for Sensors and Actuators in District Heating Substations. Division of EISLAB, Luleå University of Technology. Luleå, Sweden, 2014.

[23] SAWANT S. A., ADINARAYANA J., DURBHA S. S.; TRIPATHY A. K., Sudharsan D. Service Oriented Architecture for Wireless Sensor Networks in Agriculture. Centre of Studies in Resources Engineering, Indian Institute of Technology Bombay, Maharashtra, India, September 2012.

APÊNDICES

A - CÓDIGO XML DE INSTRUÇÕES PARA INTERPRETAÇÃO DE DADOS

```
<?xml version="1.0" encoding="utf-8"?>
	<stateVariables>
		<variable type="RespiratoryRate">
			<range min="" max="10">low</range>
			<range min="10" max="15">normal</range>
			<range min="15" max="">high</range>
		</variable>
		<variable type="HeartRate">
			<ServiceAnalysis
url="http://localhost:8080/MiLANServicesAnalysis/AnalysisService"
method="GetHeartRateState" source="HeartRate"></ServiceAnalysis>
		</variable>
		<variable type="BloodFlow">
			<range min="" max="80">low</range>
			<range min="80" max="100">normal</range>
			<range min="100" max="">high</range>
		</variable>
		<variable type="BloodPressure">
			<ServiceAnalysis
url="http://localhost:8080/MiLANServicesAnalysis/AnalysisService"
method="GetBloodPressState" source="ECG3Lead">
				<subvariable type="Siastolic"/>
				<subvariable type="Diastolic"/>
			</ServiceAnalysis>
		</variable>
		<variable type="SPO2">
			<range min="" max="90">low</range>
			<range min="90" max="100">normal</range>
		</variable>
	</stateVariables>
```

B - CÓDIGO XML DE INSTRUÇÕES PARA AVALIAÇÃO DOS ESTADOS

```
<?xml version="1.0" encoding="utf-8"?>
<systemStates>
        <systemState type="UserStressState" state="Low">
        <stateVariables>
    <stateVariable type="BloodPressure" state="High">
     <variables>
      <variable type="BloodPressure">
       <QoSParammeters>
        <QoSParammeter type="Accuracy">0.9</QoSParammeter>
       </QoSParammeters>
      </variable>
     </variables>
    </stateVariable>
    <stateVariable type="BloodPressure" state="Normal">
     <variables>
      <variable type="BloodPressure">
       <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
       </QoSParammeters>
      </variable>
     </variables>
    </stateVariable>
    <stateVariable type="BloodPressure" state="Low">
     <variables>
      <variable type="BloodPressure">
       <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
       </QoSParammeters>
      </variable>
      <variable type="SPO2">
       <QoSParammeters>
        <QoSParammeter type="Accuracy">0.7</QoSParammeter>
```

```
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="RespiratoryRate" state="Normal">
  <variables>
    <variable type="RespiratoryRate">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="RespiratoryRate" state="High">
  <variables>
    <variable type="RespiratoryRate">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
      </QoSParammeters>
    </variable>
    <variable type="SPO2">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="RespiratoryRate" state="Low">
  <variables>
    <variable type="RespiratoryRate">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
      </QoSParammeters>
```

```
    </variable>
    <variable type="SPO2">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="HeartRate" state="Normal">
  <variables>
    <variable type="HeartRate">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="HeartRate" state="High">
  <variables>
    <variable type="HeartRate">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
      </QoSParammeters>
    </variable>
    <variable type="SPO2">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
    <variable type="ECG" >
      <QoSParammeters>
        <QoSParammeter type="Accuracy">1.0</QoSParammeter>
      </QoSParammeters>
```

```
      </variable>
      <variable type="BloodPressure" >
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.3</QoSParammeter>
        </QoSParammeters>
      </variable>
    </variables>
  </stateVariable>
  <stateVariable type="HeartRate" state="Low">
    <variables>
      <variable type="HeartRate">
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.8</QoSParammeter>
        </QoSParammeters>
      </variable>
      <variable type="SPO2">
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.3</QoSParammeter>
        </QoSParammeters>
      </variable>
      <variable type="ECG" >
        <QoSParammeters>
          <QoSParammeter type="Accuracy">1.0</QoSParammeter>
        </QoSParammeters>
      </variable>
      <variable type="BloodPressure" >
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.3</QoSParammeter>
        </QoSParammeters>
      </variable>
    </variables>
  </stateVariable>
</stateVariables>
```

```
        </systemState>
        <systemState type="UserStressState" state="Medium">
                <stateVariables>
                        <stateVariable type="BloodPressure" state="High">
                                <variables>
                                        <variable type="BloodPressure">
                                                <QoSParammeters>
                                                        <QoSParammeter
type="Accuracy">0.9</QoSParammeter>
                                                </QoSParammeters>
                                        </variable>
                                </variables>
                        </stateVariable>
                        <stateVariable type="BloodPressure" state="Normal">
                                <variables>
                                        <variable type="BloodPressure">
                                                <QoSParammeters>
                                                        <QoSParammeter
type="Accuracy">0.5</QoSParammeter>
                                                </QoSParammeters>
                                        </variable>
                                </variables>
                        </stateVariable>
                        <stateVariable type="BloodPressure" state="Low">
                                <variables>
                                        <variable type="BloodPressure">
                                                <QoSParammeters>
                                                        <QoSParammeter
type="Accuracy">0.9</QoSParammeter>
                                                </QoSParammeters>
                                        </variable>
                                        <variable type="SPO2">
                                                <QoSParammeters>
```

<QoSParammeter type="Accuracy">0.4</QoSParammeter>
</QoSParammeters>
</variable>
</variables>
</stateVariable>
<stateVariable type="RespiratoryRate" state="Normal">
<variables>
<variable type="RespiratoryRate">
<QoSParammeters>
<QoSParammeter type="Accuracy">0.3</QoSParammeter>
</QoSParammeters>
</variable>
<variable type="BloodPressure">
<QoSParammeters>
<QoSParammeter type="Accuracy">0.3</QoSParammeter>
</QoSParammeters>
</variable>
</variables>
</stateVariable>
<stateVariable type="RespiratoryRate" state="High">
<variables>
<variable type="RespiratoryRate">
<QoSParammeters>
<QoSParammeter type="Accuracy">0.8</QoSParammeter>
</QoSParammeters>
</variable>
<variable type="SPO2">
<QoSParammeters>

<QoSParammeter type="Accuracy">0.3</QoSParammeter>
</QoSParammeters>
</variable>
</variables>
</stateVariable>
<stateVariable type="RespiratoryRate" state="Low">
<variables>
<variable type="RespiratoryRate">
<QoSParammeters>
<QoSParammeter type="Accuracy">0.8</QoSParammeter>
</QoSParammeters>
</variable>
<variable type="SPO2">
<QoSParammeters>
<QoSParammeter type="Accuracy">0.3</QoSParammeter>
</QoSParammeters>
</variable>
</variables>
</stateVariable>
<stateVariable type="HeartRate" state="Normal">
<variables>
<variable type="HeartRate">
<QoSParammeters>
<QoSParammeter type="Accuracy">0.3</QoSParammeter>
</QoSParammeters>
</variable>
</variables>
</stateVariable>
<stateVariable type="HeartRate" state="high">

```
                    <variables>
                        <variable type="HeartRate">
                            <QoSParammeters>
                                <QoSParammeter
type="Accuracy">0.8</QoSParammeter>
                            </QoSParammeters>
                        </variable>
                        <variable type="SPO2">
                            <QoSParammeters>
                                <QoSParammeter
type="Accuracy">0.5</QoSParammeter>
                            </QoSParammeters>
                        </variable>
                        <variable type="ECG" >
                            <QoSParammeters>
                                <QoSParammeter
type="Accuracy">1.0</QoSParammeter>
                            </QoSParammeters>
                        </variable>
                        <variable type="BloodPressure" >
                            <QoSParammeters>
                                <QoSParammeter
type="Accuracy">0.3</QoSParammeter>
                            </QoSParammeters>
                        </variable>
                    </variables>
                </stateVariable>
                <stateVariable type="HeartRate" state="Low">
                    <variables>
                        <variable type="HeartRate">
                            <QoSParammeters>
                                <QoSParammeter
type="Accuracy">0.8</QoSParammeter>
```

```
                                        </QoSParammeters>
                                    </variable>
                                    <variable type="SPO2">
                                        <QoSParammeters>
                                            <QoSParammeter
type="Accuracy">0.3</QoSParammeter>
                                        </QoSParammeters>
                                    </variable>
                                    <variable type="ECG" >
                                        <QoSParammeters>
                                            <QoSParammeter
type="Accuracy">1.0</QoSParammeter>
                                        </QoSParammeters>
                                    </variable>
                                    <variable type="BloodPressure" >
                                        <QoSParammeters>
                                            <QoSParammeter
type="Accuracy">0.3</QoSParammeter>
                                        </QoSParammeters>
                                    </variable>
                                </variables>
                            </stateVariable>
                        </stateVariables>
  </systemState>
        <systemState type="UserStressState" state="High">
   <stateVariables>
     <stateVariable type="BloodPressure" state="High">
       <variables>
        <variable type="BloodPressure">
          <QoSParammeters>
            <QoSParammeter type="Accuracy">0.8</QoSParammeter>
              <QoSParammeter type="FrequenciaAmostragemMin">0.5</QoSParammeter>
          </QoSParammeters>
```

```
    </variable>
  </variables>
</stateVariable>
<stateVariable type="BloodPressure" state="Normal">
  <variables>
    <variable type="BloodPressure">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="BloodPressure" state="Low">
  <variables>
    <variable type="BloodPressure">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
      </QoSParammeters>
    </variable>
    <variable type="SPO2">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.7</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="RespiratoryRate" state="Normal">
  <variables>
    <variable type="RespiratoryRate">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
```

```
  </variables>
</stateVariable>
<stateVariable type="RespiratoryRate" state="High">
  <variables>
    <variable type="RespiratoryRate">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
      </QoSParammeters>
    </variable>
    <variable type="SPO2">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="RespiratoryRate" state="Low">
  <variables>
    <variable type="RespiratoryRate">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
      </QoSParammeters>
    </variable>
    <variable type="SPO2">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="HeartRate" state="Normal">
  <variables>
    <variable type="HeartRate">
```

```
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="HeartRate" state="High">
  <variables>
    <variable type="HeartRate">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
      </QoSParammeters>
    </variable>
    <variable type="SPO2">
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
    <variable type="ECG" >
      <QoSParammeters>
        <QoSParammeter type="Accuracy">1.0</QoSParammeter>
      </QoSParammeters>
    </variable>
    <variable type="BloodPressure" >
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.3</QoSParammeter>
      </QoSParammeters>
    </variable>
  </variables>
</stateVariable>
<stateVariable type="HeartRate" state="Low">
  <variables>
    <variable type="HeartRate">
```

```
          <QoSParammeters>
            <QoSParammeter type="Accuracy">0.8</QoSParammeter>
          </QoSParammeters>
        </variable>
        <variable type="SPO2">
          <QoSParammeters>
            <QoSParammeter type="Accuracy">0.3</QoSParammeter>
          </QoSParammeters>
        </variable>
        <variable type="ECG" >
          <QoSParammeters>
            <QoSParammeter type="Accuracy">1.0</QoSParammeter>
          </QoSParammeters>
        </variable>
        <variable type="BloodPressure" >
          <QoSParammeters>
            <QoSParammeter type="Accuracy">0.3</QoSParammeter>
          </QoSParammeters>
        </variable>
      </variables>
    </stateVariable>
  </stateVariables>
 </systemState>
</systemStates>
```

C - CÓDIGO XML DE INTRUÇÕES PARA FORNECIMENTO DE VARIÁVEIS

```
<?xml version="1.0" encoding="utf-8"?>
<variables>
  <variable type="ECG">
    <sensorsQoSs>
      <sensorQoS>
        <sensors>
          <sensor type="ECG"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">1.0</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
      <sensorQoS>
          <sensors>
          <sensor type="SensorECG"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">1.0</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
    </sensorsQoSs>
  </variable>
  <variable type="RespiratoryRate"  defaultValue="15">
    <sensorsQoSs>
      <sensorQoS>
        <sensors>
          <sensor type="RespiratoryRate"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">1.0</QoSParammeter>
        </QoSParammeters>
```

```
      </sensorQoS>
      <sensorQoS>
        <sensors>
          <sensor                                                    type="ECG"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.8</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
      <sensorQoS>
        <sensors>
          <sensor                                                    type="SensorECG"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.8</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
    </sensorsQoSs>
  </variable>
  <variable type="HeartRate"  defaultValue="70">
    <sensorsQoSs>
      <sensorQoS>
        <sensors>
          <sensor                                                    type="RespiratoryRate"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">1.0</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
      <sensorQoS>
```

```
      <sensors>
        <sensor type="ECG"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">1.0</QoSParammeter>
      </QoSParammeters>
    </sensorQoS>
    <sensorQoS>
      <sensors>
        <sensor type="SensorECG"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">1.0</QoSParammeter>
      </QoSParammeters>
    </sensorQoS>
    <sensorQoS>
      <sensors>
        <sensor type="BloodFlow"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.9</QoSParammeter>
      </QoSParammeters>
    </sensorQoS>
    <sensorQoS>
      <sensors>
        <sensor type="BloodPressure"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.7</QoSParammeter>
```

```
      </QoSParammeters>
    </sensorQoS>
    <sensorQoS>
      <sensors>
        <sensor type="PulseOximetry"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
      </QoSParammeters>
    </sensorQoS>
    <sensorQoS>
      <sensors>
        <sensor type="SensorSPO2"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.8</QoSParammeter>
      </QoSParammeters>
    </sensorQoS>
  </sensorsQoSs>
 </variable>
 <variable type="BloodFlow"  defaultValue="92">
  <sensorsQoSs>
    <sensorQoS>
      <sensors>
        <sensor type="BloodFlow"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">1.0</QoSParammeter>
      </QoSParammeters>
    </sensorQoS>
    <sensorQoS>
```

```
        <sensors>
          <sensor type="BloodPressure"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.8</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
      <sensorQoS>
        <sensors>
          <sensor type="PulseOximetry"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.6</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
      <sensorQoS>
        <sensors>
          <sensor type="SensorSPO2"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.6</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
    </sensorsQoSs>
  </variable>
  <variable type="BloodPressure"  defaultValue="50;40">
    <sensorsQoSs>
      <sensorQoS>
        <sensors>
```

<sensor type="BloodFlow" service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
</sensors>
<QoSParammeters>
<QoSParammeter type="Accuracy">0.8</QoSParammeter>
</QoSParammeters>
</sensorQoS>
<sensorQoS>
<sensors>
<sensor type="BloodPressure"/>
</sensors>
<QoSParammeters>
<QoSParammeter type="Accuracy">1.0</QoSParammeter>
</QoSParammeters>
</sensorQoS>
<sensorQoS>
<sensors>
<sensor type="PulseOximetry" service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
</sensors>
<QoSParammeters>
<QoSParammeter type="Accuracy">0.7</QoSParammeter>
</QoSParammeters>
</sensorQoS>
<sensorQoS>
<sensors>
<sensor type="SensorSPO2" service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
</sensors>
<QoSParammeters>
<QoSParammeter type="Accuracy">0.7</QoSParammeter>
</QoSParammeters>
</sensorQoS>

```
  </sensorsQoSs>
</variable>
<variable type="SPO2"  defaultValue="92">
  <sensorsQoSs>
    <sensorQoS>
      <sensors>
        <sensor type="PulseOximetry"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">0.9</QoSParammeter>
      </QoSParammeters>
    </sensorQoS>
    <sensorQoS>
      <sensors>
        <sensor type="SPO2"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">1.0</QoSParammeter>
      </QoSParammeters>
    </sensorQoS>
    <sensorQoS>
      <sensors>
        <sensor type="SensorSPO2"/>
      </sensors>
      <QoSParammeters>
        <QoSParammeter type="Accuracy">1.0</QoSParammeter>
      </QoSParammeters>
    </sensorQoS>
  </sensorsQoSs>
</variable>
<variable type="Activity">
  <sensorsQoSs>
    <sensorQoS>
```

```
        <sensors>
          <sensor                                                    type="EMG"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">1.0</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
    </sensorsQoSs>
  </variable>
  <!--variable type="Temperature">
    <sensorsQoSs>
      <sensorQoS>
        <sensors>
          <sensor type="SensorTemperatura"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">1.0</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
    </sensorsQoSs>
  </variable-->
  <variable type="Position">
    <sensorsQoSs>
      <sensorQoS>
        <sensors>
          <sensor                                                    type="EMG"
service="http://localhost:8080/MiLANServicesAnalysis/AnalysisService?wsdl"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.8</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
```

```
      <sensorQoS>
        <sensors>
          <sensor type="Position"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">0.9</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
      <sensorQoS>
        <sensors>
          <sensor type="SensorAcelerometro1"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">1.0</QoSParammeter>
        </QoSParammeters>
      </sensorQoS> <sensorQoS>
        <sensors>
          <sensor type="SensorAcelerometro2"/>
        </sensors>
        <QoSParammeters>
          <QoSParammeter type="Accuracy">1.0</QoSParammeter>
        </QoSParammeters>
      </sensorQoS>
    </sensorsQoSs>
  </variable>
</variables>
```

D - DESCRIÇÃO DO SERVIÇO DO NODO SENSOR DE FLUXO SANGUÍNEO

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-
  hudson-752-.
-->
<!--
Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-
  hudson-752-.
-->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
    utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
    xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://Services.MiLAN/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://Services.MiLAN/"
    name="BloodFlowService">
 <wsp:Policy xmlns:wsat="http://schemas.xmlsoap.org/ws/2004/10/wsat"
    wsu:Id="BloodFlowPortBinding_GetBloodFlow_WSAT_Policy">
  <wsat:ATAlwaysCapability />
  <wsat:ATAssertion xmlns:ns1="http://schemas.xmlsoap.org/ws/2002/12/policy"
    wsp:Optional="true" ns1:Optional="true" />
   </wsp:Policy>
<types>
<xsd:schema>
  <xsd:import namespace="http://Services.MiLAN/"
    schemaLocation="http://localhost:8080/BloodFlowSensorNode/BloodFlowService?xsd=1"
    />
   </xsd:schema>
   </types>
<message name="GetBloodFlow">
  <part name="parameters" element="tns:GetBloodFlow" />
```

```
    </message>
<message name="GetBloodFlowResponse">
  <part name="parameters" element="tns:GetBloodFlowResponse" />
    </message>
<portType name="BloodFlow">
<operation name="GetBloodFlow">
  <input wsam:Action="http://Services.MiLAN/BloodFlow/GetBloodFlowRequest"
      message="tns:GetBloodFlow" />
  <output wsam:Action="http://Services.MiLAN/BloodFlow/GetBloodFlowResponse"
      message="tns:GetBloodFlowResponse" />
    </operation>
    </portType>
<binding name="BloodFlowPortBinding" type="tns:BloodFlow">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
<operation name="GetBloodFlow">
  <wsp:PolicyReference URI="#BloodFlowPortBinding_GetBloodFlow_WSAT_Policy" />
  <soap:operation soapAction="" />
<input>
  <wsp:PolicyReference URI="#BloodFlowPortBinding_GetBloodFlow_WSAT_Policy" />
  <soap:body use="literal" />
    </input>
<output>
  <wsp:PolicyReference URI="#BloodFlowPortBinding_GetBloodFlow_WSAT_Policy" />
  <soap:body use="literal" />
    </output>
    </operation>
    </binding>
<service name="BloodFlowService">
<port name="BloodFlowPort" binding="tns:BloodFlowPortBinding">
  <soap:address location="http://localhost:8080/BloodFlowSensorNode/BloodFlowService" />
    </port>
    </service>
    </definitions>
```

E - WSDL DO BARRAMENTO DE SERVIÇOS

Código XML do WSDL referentes serviços disponibilizados no barramento de serviços.

As linhas iniciadas com o caractere “+” possuem itens recolhidos do ramo XML. Aquelas com o caractere “-” é referente a um ramo expandido.

Definições de segurança e o esquema XSD:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
 Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-
   hudson-752-.
 -->
<!--
 Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-
   hudson-752-.
 -->
- <definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
   wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
   xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://Services.MiLAN/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://Services.MiLAN/"
   name="MiLANServicesService">
- <types>
- <xsd:schema>
  <xsd:import namespace="http://Services.MiLAN/"
   schemaLocation="http://localhost:8080/MiLAN_Services_Endpoints/MiLANServicesSe
   rvice?xsd=1" />
  </xsd:schema>
  </types>
```

Dados necessários para utilização do método Web getPosition, que retorna a posição do paciente, e estrutura que define a resposta do método Web:

```
- <message name="getPosition">
<part name="parameters" element="tns:getPosition" />
    </message>
- <message name="getPositionResponse">
  <part name="parameters" element="tns:getPositionResponse" />
    </message>
```

Demais métodos Web construídos no serviço e apresentados no WSDL:

```
+ <message name="getBloodPress">
+ <message name="getBloodPressResponse">
+ <message name="getBloodFlow">
+ <message name="getBloodFlowResponse">
+ <message name="getHeartRate">
+ <message name="getHeartRateResponse">
+ <message name="getPulseOximetry">
+ <message name="getPulseOximetryResponse">
+ <message name="getRespiratoryRate">
+ <message name="getRespiratoryRateResponse">
+ <message name="getECG">
+ <message name="getECGResponse">
+ <message name="getEMG">
+ <message name="getEMGResponse">
```

Dados de endereçamento de cada operação do serviço MiLANServices, para a transmissão de mensagens de requisição e de resposta:

```
- <portType name="MiLANServices">
- <operation name="getPosition">
  <input wsam:Action="http://Services.MiLAN/MiLANServices/getPositionRequest"
      message="tns:getPosition" />
```

<output
wsam:Action="http://Services.MiLAN/MiLANServices/getPositionResponse"message=
"tns:getPositionResponse" />
</operation>
+ <operation name="getBloodPress">
+ <operation name="getBloodFlow">
+ <operation name="getHeartRate">
+ <operation name="getPulseOximetry">
+ <operation name="getRespiratoryRate">
+ <operation name="getECG">
+ <operation name="getEMG">
</portType>

Protocolos de comunicação para cada operação (no caso deste projeto, SOAP):

- <binding name="MiLANServicesPortBinding" type="tns:MiLANServices">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="getPosition">
<soap:operation soapAction="" />
- <input>
<soap:body use="literal" />
</input>
- <output>
<soap:body use="literal" />
</output>
</operation>
- <operation name="getBloodPress">
<soap:operation soapAction="" />
- <input>
<soap:body use="literal" />
</input>
- <output>
<soap:body use="literal" />
</output>

```
    </operation>
+ <operation name="getBloodFlow">
+ <operation name="getHeartRate">
+ <operation name="getPulseOximetry">
+ <operation name="getRespiratoryRate">
+ <operation name="getECG">
+ <operation name="getEMG">
  </binding>
```

Endereço de acesso ao serviço MiLANServices. Disponibilizado acesso por meio do *endpoint* MiLANServicesService pela porta 8080 (porta de internet):

```
- <service name="MiLANServicesService">
- <port name="MiLANServicesPort" binding="tns:MiLANServicesPortBinding">
  <soap:address
       location="http://localhost:8080/MiLAN_Services_Endpoints/MiLANServicesService"
       />
     </port>
     </service>
     </definitions>
```

F - COMBINAÇÕES POSSÍVEIS PARA OS ESTADOS DAS VARIÁVEIS

Tabela F.1 Conjuntos de sensores possíveis em uma rede convencional para atender aos requisitos da aplicação para cada combinação possível de estados das variáveis.

Estados das variáveis	Conjuntos de sensores
LLL	BP, SP, RP, HR, ECG
LLN	BO, RP, HR
LLH	BP, SP, RP, HR, ECG
LNL	BP, SP, RP, HR, ECG
LNN	BO, RP, HR
LNH	BP, SP, RP, HR, ECG
LHL	BP, SP, RP, HR, ECG
LHN	BO, RP, HR
LHH	BP, SP, RP, HR, ECG
NLL	BP, SP, RP, HR, ECG
NLN	BP, RP, SP, HR
NLH	BP, SP, RP, HR, ECG
NNL	BP, SP, RP, HR, ECG
NNN	BO, RP, HR
NNH	BP, SP, RP, HR, ECG
NHL	BP, SP, RP, HR, ECG
NHN	BP, RP, SP, HR
NHH	BP, SP, RP, HR, ECG
HLL	BP, SP, RP, HR, ECG
HLN	BP, RP, SP, HR
HLH	BP, SP, RP, HR, ECG
HNL	BP, SP, RP, HR, ECG
HNN	BO, RP, HR
HNH	BP, SP, RP, HR, ECG
HHL	BP, SP, RP, HR, ECG
HHN	BP, RP, SP, HR
HHH	BP, SP, RP, HR, ECG

Tabela F.2 Conjuntos de sensores possíveis em uma rede MiLAN para atender aos requisitos da aplicação para cada combinação possível de estados das variáveis.

Estados das variáveis	Conjuntos de sensores	Estados das variáveis	Conjuntos de sensores	Estados das variáveis	Conjuntos de sensores
LLL	BP, PO, ECG	NLL	PO, ECG	HLL	BP, PO, ECG
	BP, PO, RR		SP, ECG		BP, PO, RR
	BP, SP, ECG		PO, RR		BP, SP, ECG
	BP, SP, RR		SP, RR		BP, SP, RR
	BF, PO, ECG				
	BF, PO, RR				
	BF, SP, ECG				
	BF, SP, RR				
LLN	BP, PO	NLN	PO, ECG	HLN	BP, PO, ECG
	BF, SP, ECG		SP, ECG		BP, PO, RR
	BF, SP, RR		PO, RR		BP, SP, ECG
			SP, RR		BP, SP, RR
LLH	BP, PO, ECG	NLH	PO, ECG	HLH	BP, PO, ECG
	BP, PO, RR		SP, ECG		BP, PO, RR
	BP, SP, ECG		PO, RR		BP, SP, ECG
	BP, SP, RR		SP, RR		BP, SP, RR
	BF, PO, ECG				
	BF, PO, RR				
	BF, SP, ECG				
	BF, SP, RR				
LNL	BP, PO, ECG	NNL	PO, ECG	HNL	BP, PO, ECG
	BP, PO, RR		SP, ECG		BP, PO, RR
	BP, SP, ECG		PO, RR		BP, SP, ECG
	BP, SP, RR		SP, RR		BP, SP, RR
	BF, PO, ECG				
	BF, PO, RR				
	BF, SP, ECG				
	BF, SP, RR				
LNN	BF, PO, ECG	NNN	PO, ECG	HNN	BP, PO, ECG
	BF, PO, RR		SP, ECG		BP, PO, RR
	BF, SP, ECG		PO, RR		BP, SP, ECG
	BF, SP, RR		SP, RR		BP, SP, RR
LNH	BP, PO, ECG	NNH	PO, ECG	HNH	BP, PO, ECG
	BP, PO, RR		SP, ECG		BP, PO, RR
	BP, SP, ECG		PO, RR		BP, SP, ECG
	BP, SP, RR		SP, RR		BP, SP, RR
	BF, PO, ECG				
	BF, PO, RR				
	BF, SP, ECG				

	BF, SP, RR				
LHL	BP, PO, ECG	NHL	PO, ECG	HHL	BP, PO, ECG
	BP, PO, RR		SP, ECG		BP, PO, RR
	BP, SP, ECG		PO, RR		BP, SP, ECG
	BP, SP, RR		SP, RR		BP, SP, RR
	BF, PO, ECG				
	BF, PO, RR				
	BF, SP, ECG				
	BF, SP, RR				
LHN	BP, PO, ECG	NHN	PO, ECG	HHN	BP, PO, ECG
	BP, PO, RR		SP, ECG		BP, PO, RR
	BP, SP, ECG		PO, RR		BP, SP, ECG
	BP, SP, RR		SP, RR		BP, SP, RR
	BF, PO, ECG				
	BF, PO, RR				
	BF, SP, ECG				
	BF, SP, RR				
LHH	BP, PO, ECG	NHH	PO, ECG	HHH	BP, PO, ECG
	BP, PO, RR		SP, ECG		BP, PO, RR
	BP, SP, ECG		PO, RR		BP, SP, ECG
	BP, SP, RR		SP, RR		BP, SP, RR
	BF, PO, ECG				
	BF, PO, RR				
	BF, SP, ECG				
	BF, SP, RR				