UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

MASSAHIDE DE OLIVEIRA NAMBA

Modelagem e Especificação de um Middleware para Redes de Sensores Sem Fio Aplicado à Saúde

Goiânia
2011

MASSAHIDE DE OLIVEIRA NAMBA

Modelagem e Especificação de um Middleware para Redes de Sensores Sem Fio Aplicado à Saúde

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Computação.

Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Leandro Luís Galdino de Oliveira

Coorientador: Prof. Dr. Iwens Gervásio de Sene Junior

Goiânia
2011

Dados Internacionais de Catalogação na Publicação (CIP)
GPT/BC/UFG

N174m

Namba, Massahide de Oliveira.
Modelagem e especificação de um middleware para redes de sensores sem fio aplicado à saúde [manuscrito] / Massahide de Oliveira Namba. - 2011.
xv, 108 f. : il., figs, tabs.

Orientador: Prof. Dr. Leandro Luís Galdino de Oliveira; Coorientador: Prof. Dr. Iwens Gervásio de Sene Junior.
Dissertação (Mestrado) – Universidade Federal de Goiás, Instituto de Informática, 2011.
Bibliografia.
Inclui lista de figuras e tabelas.
Apêndices.

1. Middleware – Saúde – Cuidados pessoais. 2. Redes sem fio. I. Título.

CDU: 004.75:613

MASSAHIDE DE OLIVEIRA NAMBA

Modelagem e Especificação de um Middleware para Redes de Sensores Sem Fio Aplicado à Saúde

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Computação, aprovada em 29 de Agosto de 2011, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Leandro Luís Galdino de Oliveira
Instituto de Informática – UFG
Presidente da Banca

Prof. Dr. Iwens Gervásio de Sene Junior
Instituto de Informática – UFG

Prof. Dr. Renato de Freitas Bulcão Neto
Instituto de Informática – UFG

Prof. Dr. Hervaldo Sampaio Carvalho
Faculdade de Medicina – UnB

Massahide de Oliveira Namba

Para Deus, sempre presente em minha vida. Para minha mãe e meu pai, Devanice e Massahide (*in memoriam*), que me apoiaram constantemente, e aos amigos pelo companheirismo e por terem suportado o caos e a inconstância da vida de um mestrando e profissional compulsivo.

Agradecimentos

Agradeço em especial aos professores Leandro Luís Galdino de Oliveira e Iwens Gervásio Sene Junior, pelo apoio inconcusso ao longo do desenvolvimento deste trabalho. Agradeço também aos vários colegas que me foram companheiros e me inspiraram a transposição de obstáculos abrindo exceções cruciais para o sucesso deste trabalho em demonstração do seu grande companheirismo, sem fazer nomeações, haja vista suas importâncias individuais e únicas. Por fim, agradeço a todos aqueles que me apoiaram e entenderam as minhas dificuldades no período de construção deste trabalho.

Por toda a sua vida avance diariamente, tornando-se mais capacitado a cada dia que passa. Isso não tem fim.

Yamamoto Tsunetomo,
Hagakure – Na sombra das folhas.

Resumo

de Oliveira Namba, Massahide. **Modelagem e Especificação de um Middleware para Redes de Sensores Sem Fio Aplicado à Saúde**. Goiânia, 2011. 108p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

As soluções na área da saúde focam em diagnósticos, produtividade e cuidados pessoais. Desta forma, as Redes de Sensores Sem Fio – RSSF – permitem aos profissionais de saúde o monitoramento mais eficiente das pessoas. Para que seja mais eficiente a integração entre as RSSF e as aplicações e interfaces que os profissionais de saúde utilizam, é necessário um componente de software que facilite e mitigue os problemas do desenvolvimento dessas aplicações, a este componente denominamos middleware. Da perspectiva do desenvolvimento de software, este middleware oferece principalmente a vantagem de poder focar o trabalho nas exigências do software, livrando o programador da preocupação com características de hardware e interconexão de dispositivos na RSSF. A fim de facilitar o uso de aplicações na área de saúde, este middleware propõe características que são introduzidas para simplificar e garantir a transmissão de informações, além de proporcionar o uso confiável das mesmas. Nesta dissertação é proposto o middleware Kratos que permite portabilidade e execução concorrente das aplicações por meio de interfaces padronizadas com dispositivos móveis, possibilitando assim o seu uso em redes de sensores aplicadas a saúde de forma transparente. Para validar o uso do Kratos e suas funcionalidades foi proposto um cenário de aplicação para o monitoramento de sinais eletrocardiográficos (ECG), que realiza o monitoramento e envio de alertas para dispositivos móveis de uma equipe médica. A implementação do sensor de ECG bem como toda a modelagem do middleware está em conformidade com a família de padrões IEEE 1451.

Palavras–chave

middleware, saúde, monitoramento, redes de sensores sem fio, IEEE 1451

Abstract

de Oliveira Namba, Massahide. **Modeling and Specification of a Wireless Sensor Network Middleware Intended for Health**. Goiânia, 2011. 108p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Solutions in healthcare are focused on diagnostics, productivity and personal care in this way, the Wireless Sensor Networks – WSN – allow health professionals to monitor people more efficiently. To make integration between WSN and the applications and interfaces that health professionals use more effective its needed a software component that facilitates and mitigates the problems of developing these applications, this component is called middleware. From the perspective of software development, this middleware mainly offers the advantage of focusing the work on the requirements of the software, making the programmer not to worry about the characteristics of hardware and interconnection of devices in the WSN. In the way to facilitate the use of applications in healthcare, this middleware offers some features that are introduced to simplify and ensure the transmission of information in addition to providing reliable use of them. In this work we propose a middleware named Kratos that allows portability and concurrent execution of applications by standardized interfaces to mobile devices, allowing their use in sensor networks applied to health in a transparent way. To evaluate the use of Kratos and its functionality was proposed an application scenario for monitoring electrocardiographic signals (ECG), which will conduct the monitoring and sending alerts to mobile devices from a medical staff. The implementation of the ECG sensor and all the modeling of middleware is in compliance with the IEEE 1451 family of standards.

Keywords

middleware, health, monitoring, wireless sensor networks, IEEE 1451

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

1.1 Contexto

Os sistemas de saúde caminham para uma crise iminente devido à grande pressão causada pelas necessidades crescentes a seus serviços. Isto pode ser visto pela sobrecarga, escassez de pessoal qualificado e orçamento escasso. É primordial a busca por avanços tecnológicos que auxiliem os sistemas de saúde a se tornarem mais acessíveis e eficientes, e provavelmente contornando a possível crise desse sistema [46]. A área de medicina computacional é um campo interdisciplinar desafiador com necessidades crescentes de tecnologia que suporte a alta demanda por agilidade e eficiência, de forma a reduzir tempo e custo e aumentar a qualidade dos serviços de saúde. Levando em conta essas demandas, a interoperabilidade e a integração podem ser identificadas como peças chave para o desenvolvimento de tecnologias para a área de saúde, pois com elas se conseguirá um gerenciamento de conhecimento mais eficiente [29]. A computação pode favorecer o desenvolvimento de soluções para a medicina computacional enfatizando aspectos importantes para esta área como acurácia, disponibilidade, robustez, além dos aspectos demandados citados anteriormente [24].

De acordo com dados da Organização Mundial de Saúde (OMS), em "Estatísticas Mundiais de Saúde 2011" [63], existem evidências para a mudança do foco de atenção nas doenças crônicas não transmissíveis (DCNT), como as doenças cardiovasculares, o diabetes, certos tipos de câncer e as doenças respiratórias crônicas. Esta classe de doenças respondem por aproximadamente dois terços da mortalidade prematura. Segundo Maher et al. em [38] existem três elementos para reduzir a carga de morbidade, incapacitação e mortalidade prematura relacionadas com as DCNT: (1) identificar e tratar fatores de risco alteráveis, (2) prevenir DCNT comuns e (3) diagnosticar, tratar e acompanhar pacientes com DCNT comuns usando protocolos padrão. Estes elementos se encaixam bem na categoria de sensoriamento e computação ubíqua, facilitando a inferência de diagnósticos através da representação de conhecimento, que seja integrado, por exemplo, com centros de pesquisa biológica.

O anseio pela comunicação sempre foi fundamental para a construção de um

mundo cada vez mais tecnológico. Esse anseio faz com que a distância entre as pessoas diminua a cada dia chegando a uma integração quase que completa entre as pessoas e as tecnologias que são propostas. Em função disso uma tecnologia que tem uma importância peculiar é a de redes de computadores. Através delas o mundo conseguiu se interligar tornando tudo mais acessível. Em seguida, as redes sem fio vieram agregar vantagens de interligar as pessoas. Com essa tecnologia várias barreiras foram quebradas, principalmente as barreiras físicas, auxiliando as redes cabeadas com o problema de conectividade.

Nesse contexto surgem as Redes de Sensores Sem Fio (RSSF), diferindo das demais por serem formadas por nodos sensores e atuadores, que consistem em dispositivos computacionais de tamanho reduzido, baixo custo, baixo consumo energético e capacidade de sensoriamento multifuncional, os quais comunicam-se com uma estação base. Estes dispositivos computacionais são capazes de monitorar, e em alguns casos modificar, condições do ambiente no qual estão inseridos, assim como processar os dados coletados e comunicar-se com outros dispositivos da RSSF.

As RSSF podem ser utilizadas para monitorar um ambiente específico, que pode ser um corpo humano, uma área de preservação ambiental ou uma área sob vigilância militar, por exemplo [49]. As redes de sensores utilizam mecanismos de comunicação específicos, pois os nós necessitam comunicar entre si para atender aos objetivos da rede, que é auto-organizável [35]. A utilização de redes de sensores depende de fatores de gerência, tais como o arranjo entre os nós, a vida útil das baterias entre outros fatores que devem ser articulados de modo que se tenha um aproveitamento total de recursos. Dessa forma, a vida útil e a eficiência dessas redes aumentam consideravelmente, o que resulta em economia de recursos e garantia de confiabilidade.

As necessidades de uma RSSF para a área de saúde podem diferir em vários aspectos em relação às RSSF *ad hoc*, de propósito geral. Uma RSSF para a área de saúde pode, por exemplo, ser instalada permanentemente na infraestrutura de um hospital, contando com a completa integração entre os dispositivos sem fio, de forma a garantir disponibilidade para os vários serviços de emergência. Outro requisito importante neste ambiente é a segurança da informação trafegada entre os sensores, pois esta informação não deve ser exposta de forma alguma a uma parte não autorizada, permanecendo totalmente confidencial, e muito menos pode ter sua integridade comprometida, sendo suscetível a alterações maliciosas não autorizadas [54, 50]. As soluções na área da saúde focam no auxílio a diagnósticos, produtividade e cuidados pessoais, desta forma, as RSSF permitem aos agentes de saúde o monitoramento mais eficiente dos seus pacientes [53]. Para que seja mais eficiente a integração entre as RSSF e as aplicações e interfaces que os profissionais de saúde utilizam é necessário um substrato que facilite e mitige os problemas do desenvolvimento dessas aplicações, que denominamos middleware.

1.2 Motivação

Estudar as redes de sensores para desenvolvimento de um middleware aplicado a área de saúde.

Modelar um middleware que permita a interoperabilidade e execução concorrente das aplicações através de interfaces genéricas, possibilitando o seu uso em redes para a área de saúde.

1.3 Uma visão geral do middleware

O desenvolvimento do framework em uma linguagem de domínio público possibilita ao usuário a criação de módulos que podem ser integrados adaptando-se a diferentes aplicações da área de saúde. Também é possível ressaltar que a maioria dos frameworks existentes são direcionados a uma aplicação específica ou então genéricos, entretanto até o desenvolvimento deste trabalho não foi encontrado um middleware específico para a área de saúde que seja interoperável com diversas classes e modelos de sensores. Essa integração de novos módulos é possível através da definição de interfaces e adequação a padrões, em especial a norma IEEE 1451, que fará a interação com o usuário adaptando o middleware às expectativas da aplicação.

Os requisitos do middleware serão apresentados no capítulo 5, dentre os quais o requisito chave é a interoperabilidade que faz com que o middleware seja capaz de gerenciar redes com diferentes tipos de sensores, como por exemplo de sinais eletrocardiográficos (ECG), sinais eletromiográficos (EMG) ou de resistência galvânica da pele (GSR), além de gerenciar dispositivos de fabricantes diversos que estejam em conformidade com a norma IEEE 1451 e também comunicar-se com outros sistemas existentes no ambiente hospitalar.

1.4 Sumário da dissertação

No Capítulo 2 estão relacionadas as tecnologias utilizadas na estruturação do trabalho, que serão necessários para a melhor compreensão do mesmo.

No Capítulo 3 é apresentada o estado da arte de soluções correlatas, exemplos de outros middlewares, como funcionam, em que são baseados, quais as suas áreas de aplicação, entre outras informações.

O Capítulo 4 mostra alguns dos padrões relacionados a RSSF que foram utilizados na modelagem do trabalho.

O Capítulo 5 apresenta detalhes das características do middleware: funcionamento, objetivos, limitações e o que traz de inovação em relação aos middlewares já exis-

tentes, além de uma visão geral que possibilita uma compreensão satisfatória do projeto. Neste Capítulo também é apresentada a modelagem do sistema.

No Capítulo 6 são apresentadas as considerações finais e trabalhos futuros.

CAPÍTULO 2

Tecnologias e aplicações das redes de sensores sem fio

Neste Capítulo será realizada uma revisão das principais tecnologias e conceitos relacionados com este trabalho. O objetivo é contextualizar o leitor ao tema abordado pela proposta.

2.1 Redes de Sensores sem Fio

Redes de sensores sem fio estão em amplo desenvolvimento, dado o número de pesquisas publicadas com este tema, e estão sendo utilizados cada vez mais em diferentes áreas. O avanço das área de microprocessadores, sensoriamento, sistemas eletromecânicos, juntamente com o avanço na área de comunicação sem fio tem um papel fundamental nesse desenvolvimento.

A utilização de sensores, que são dispositivos que recebem e respondem a estímulos, também vem crescendo e contribuindo para o desenvolvimento de redes de sensores sem fio. Os sensores podem medir grandezas físicas, como por exemplo temperatura e pressão, e converter os dados obtidos em sinais elétricos para que possam ser processados por dispositivos computacionais.

Os sensores chamados “inteligentes”, por poderem se comunicar e se reagrupar numa rede, estão cada vez mais presentes em diferentes áreas, monitorando processos físicos, químicos e biológicos, por exemplo [35].

Esse tipo de rede possui um grande número de nodos (sensores ou atuadores). Assim, é necessário um mecanismo para realização de uma autoconfiguração. Isto se deve pelo fato de uma rede de sensores sem fio ser autônoma e exigir uma grande capacidade de cooperação para executar as tarefas exigidas pela rede.

Um dos pontos importantes da autoconfiguração é que os sensores de uma rede necessitam estar sempre atentos às mudanças que possam acontecer na rede. Por exemplo, um sensor pode passar do estado ativo para o estado inativo e a rede precisar ser reorganizada. O contrário também pode acontecer, um sensor que não participava da rede

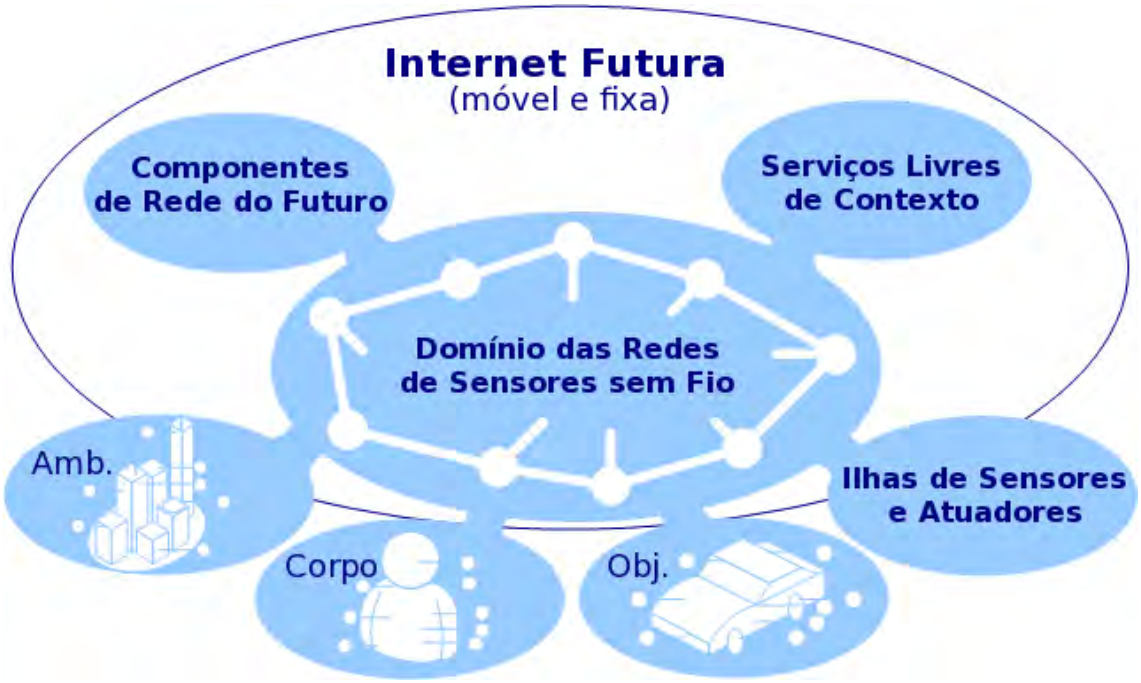

Figura 2.1: *Exemplo de RSSF, adaptado de [10]*

pode passar a participar, sendo necessário o reconhecimento deste pelos outros e os dados recolhidos por ele passarem a contribuir no resultado final buscado pela rede [11].

Uma ilustração dos possíveis usos de RSSF é encontrada na Figura 2.1. Numa rede de sensores sem fio, um nodo pode ser composto por vários sensores e estes podem ser de diferentes aplicações como sensores de temperatura, de presença, de pressão, de acústica, dentre outros. Estes sensores, ao pertencer a uma rede, devem ser capazes de detectar um evento, processá-lo e tomar uma decisão, como passar ou não uma informação captada para os outros sensores ou outros nodos. Para essa comunicação, os sensores são organizados em grupos chamados clusters. Em cada grupo, pelo menos um sensor é responsável por fazer essa difusão [42].

Do ponto de vista de organização, as redes de sensores sem fio são idênticas às redes móveis *ad-hoc*, pois possuem elementos computacionais que se comunicam diretamente através de enlaces de comunicação sem fio. Porém, as redes móveis *ad-hoc* ou MANET (*Mobile Ad-hoc Network*) têm como objetivo a comunicação entre elementos computacionais para executar tarefas distintas individualmente, enquanto que nas redes de sensores sem fio os sensores comunicam entre si a fim de executar uma função de forma colaborativa.

2.1.1 Estabelecimento de uma rede de sensores sem fio

Uma RSSF é estabelecida em três fases: Fase de distribuição dos sensores, fase de pós-distribuição e fase de redistribuição ou adição de novos sensores [35].

- **Fase de distribuição:** os sensores são colocados no local de monitoramento de duas formas, a primeira é lançando-os em massa no local. Este método é usado em caso de uma rede com grande quantidade de sensores ou quando o campo é de difícil acesso. Os sensores também podem também ser colocados um a um, quando se trata de redes menores como por exemplo em sensores de temperatura no corpo de um paciente em uma Unidade de Terapia Intensiva (UTI).
- **Fase de pós-distribuição:** após a distribuição dos sensores é necessário fazer manutenções rotineiras. Em casos de redes pequenas e acessíveis a uma manutenção manual este trabalho fica fácil. Quando se trata de uma rede maior onde os sensores foram distribuídos em massa, a auto-organização é extremamente importante. A preocupação com a rede após a distribuição dos sensores se faz necessária devido às mudanças que a rede está sujeita, como alteração na localização dos sensores, interferências, esgotamento de energia, entre outras.
- **Fase de redistribuição:** nesta fase, novamente a auto-organização difere as redes de sensores sem fio das redes convencionais. Pode ser necessário adicionar mais sensores à rede ou mudá-los de posição a fim de substituir sensores defeituosos ou aumentar a área de cobertura.

2.1.2 Características das redes de sensores sem fio

De acordo com a área de aplicação, as redes de sensores sem fio apresentam características específicas. Isso garante a resolução de problemas também específicos, pois cada rede é projetada para uma finalidade, utilizando sensores apropriados [35].

A seguir, algumas características importantes das RSSF:

- O **endereçamento dos sensores** depende de como e para quê estão sendo utilizados. No monitoramento das funções vitais do corpo humano, por exemplo, é necessário o endereçamento de cada sensor, pois cada um monitora uma parte específica do corpo e cada informação é importante individualmente. Já no monitoramento de uma floresta, os sensores não precisam ser necessariamente identificados individualmente, pois o que se espera são dados da região em geral, onde todos os sensores juntos enviam dados para a obtenção de uma só informação. Enfim, o método de endereçamento depende da lógica de negócio da aplicação.
- A **agregação dos dados** é outra característica importante. É a capacidade de uma rede de sensores sem fio agregar os dados coletados. Se a rede possui essa capacidade, ela pode combinar os dados ainda na rede antes de enviar à estação base. Isso é interessante por diminuir a quantidade de mensagens enviadas, poupando recursos e prolongando a vida útil da RSSF.

- A **mobilidade dos sensores** indica se os sensores de uma rede podem ou não se mover no sistema de coleta de dados. Os sensores colocados no corpo humano para verificar a temperatura são considerados sensores estáticos por não se moverem. Por outro lado, sensores colocados na superfície de rios para medir o nível de poluição da água são móveis.
- A **quantidade de sensores** também é uma característica relacionada à aplicação da rede de sensores sem fio. Por exemplo, uma rede que monitora ambientes externos como umidade de florestas ou temperatura num deserto necessita de uma grande quantidade de sensores.
- Uma característica bastante importante é a **limitação da energia disponível** para os sensores. Geralmente os sensores são colocados em locais de difícil acesso onde a manutenção é algo complicado de se fazer. Um exemplo disso são sensores instalados em balões para medir a temperatura em diferentes altitudes. Até mesmo sensores instalados em pessoas para monitorar os batimentos cardíacos precisam de uma vida útil considerável. Dessa forma, ao se projetar uma rede de sensores sem fio, é necessário pensar na fonte de energia que irá alimentar os sensores. Aplicações, protocolos e algoritmos devem ser escolhidos de maneira que se conheça a quantidade de energia consumida para que se tenha um maior aproveitamento dos sensores.

  Bateria, rádio, processador e sensores compõem o modelo de energia. Este modelo pode ser visto como um provedor de energia para elementos consumidores que dependem de uma bateria com capacidade finita. Os consumidores são modelos de processadores, rádio, e sensores. Cada consumidor notifica o provedor do seu consumo de energia e esse informa a quantidade de energia disponível.

  - A bateria é o armazenador da energia que será consumida pelo sensor. Tem capacidade finita e uma taxa de consumo.
  - O rádio representa o sistema de transmissão e recepção, amplificador e antena. Geralmente a transmissão de dados consome mais energia que a recepção.
  - O processador é o elemento de processamento central do sensor. O consumo de energia depende de vários fatores, como a velocidade de clock, ou seja, quanto menor a sua frequência, menor o consumo.
  - O sensor é o dispositivo de sensoriamento. O consumo depende de algumas variáveis como o tipo de grandeza a ser medida e do modo de operação.
- A **auto-organização da rede** é uma característica que faz das redes de sensores sem fio diferentes das demais. Pela natureza das redes de sensores sem fio, essas redes devem ter a capacidade de se ajustar a possíveis alterações sem a interferência humana. Nesse tipo de rede, os sensores podem ser perdidos, por diversas razões. Falta de energia, destruição física e falta de comunicação com o restante dos

sensores são exemplos. Também pode acontecer de um sensor que não fazia parte da rede passar a fazer. Assim, através de mecanismos de auto-organização, a rede continua funcionando e executando sua função. Isso deve ser periódico e automático por meio de algoritmos. Dessa forma, a capacidade de auto-organização é uma característica fundamental para as redes de sensores sem fio e as tornam diferentes em relação a outras redes, fazendo com que elas se tornem viáveis, pois uma das grandes vantagens desse tipo de rede é poder coletar dados em locais remotos e de difícil acesso, onde uma manutenção manual seria totalmente inviável, o que faz da auto-organização uma característica tão importante.

- As redes de sensores sem fio são particulares também pelo fato de executarem **tarefas colaborativas**. Isso quer dizer que a rede detecta e executa eventos de interesse global do sistema que vão além de promover a comunicação entre os sensores e entre os sensores e a base de dados, como por exemplo a identificação de melhores caminhos para transmissão de dados, difundindo esta informação pela rede.

2.1.3 Funcionalidades das redes de sensores sem fio

Estabelecimento da rede, manutenção, sensoriamento, processamento e comunicação são as principais funcionalidades das RSSF. Suas ocorrências são simultâneas e podem estar ativas em diferentes momentos do tempo de vida das redes de sensores.

Manutenção de uma rede de sensores sem fio

A manutenção é uma funcionalidade utilizada em todas as outras fases: estabelecimento, sensoriamento, processamento e comunicação. É funcional durante toda a vida da rede e tem por finalidade prolongá-la, atender aos requisitos da aplicação, além de prever o que pode acontecer ao longo de sua vida. Durante todo o tempo de vida os sensores atingem níveis de energia que podem comprometer de forma parcial ou total sua capacidade. A manutenção pode ser reativa, preventiva, corretiva ou adaptativa, dependendo do estado da rede e de cada sensor, sendo absolutamente importante para manter a rede em atividade.

Sensoriamento

Esta fase está relacionada com a percepção do ambiente e a coleta de dados. Inclui a determinação de distância do que passará pelo sensoriamento, ruídos do ambiente, tipo de dados coletados, volume de informação e frequência de amostragem.

O sensoriamento deve também determinar as áreas de sobreposição de sensores. A detecção destas áreas pode resultar na alteração do estado de um sensor. Caso a área

de percepção de um sensor tenha intersecção com a área de outro sensor, um destes pode ser desativado, sendo necessário somente as informações de um sensor. Isso resulta em economia de energia e um maior tempo de vida da rede.

Outra função do sensoriamento é avaliar se o número de sensores ativos da rede é suficiente para a execução da tarefa, pois os sensores podem falhar devido à falta de energia, destruição ou inoperância temporária. Nesse caso, um rearranjo da rede é feito para mantê-la ativa até que não seja mais possível realizar a tarefa a qual a rede foi designada.

Processamento

O processamento em redes de sensores pode ser de suporte ou de informação. O primeiro é o processamento funcional dos sensores, que está relacionado com o gerenciamento, comunicação e manutenção da rede. Já o processamento da informação diz respeito à compreensão, correlação, criptografia e assinatura digital, agregação dos dados, entre outros.

2.1.4 Restrição da energia dos nós em redes de sensores sem fio

As redes de sensores sem fio possuem recursos bastante limitados. Além da capacidade computacional e memória serem reduzidas, as reservas de energia, que são provenientes de bateria, são pequenas, o que é um problema, pois dependendo da aplicação, os sensores são colocados em áreas remotas de difícil acesso, ou então implantados no corpo humano, como pode acontecer no caso da área de saúde, o que dificulta a manutenção e a troca de baterias. Dessa forma, as reservas de energia de uma rede de sensores é algo que merece atenção e é um dos aspectos mais importantes no projeto de rede de sensores sem fio, pois define, além de vários outros fatores, o tempo de vida da rede [26, 35].

Para ter acesso à quantidade de energia disponível em cada sensor ou em cada parte da rede de sensores é utilizado o mapa de energia. A utilização desse mapa auxilia no prolongamento de vida da rede, auxiliando o usuário a determinar quais regiões da rede estão mais propensas a falhas por falta de energia. Uma das maneiras mais notórias de construir um mapa de energia é fazer que cada nodo envie periodicamente sua quantidade de energia aos nodos de monitoramento. Uma maneira visual de representar o mapa é com uma imagem de níveis de cinza, onde as áreas escuras representam mais energia e áreas claras, menos energia, como exemplificado na Figura 2.2.

Através do mapa de energia é possível organizar a rede de modo que ela continue em atividade, trocando sensores que estão na eminência de ficarem sem energia ou colocando novos sensores gradativamente na rede para que os resultados não sejam

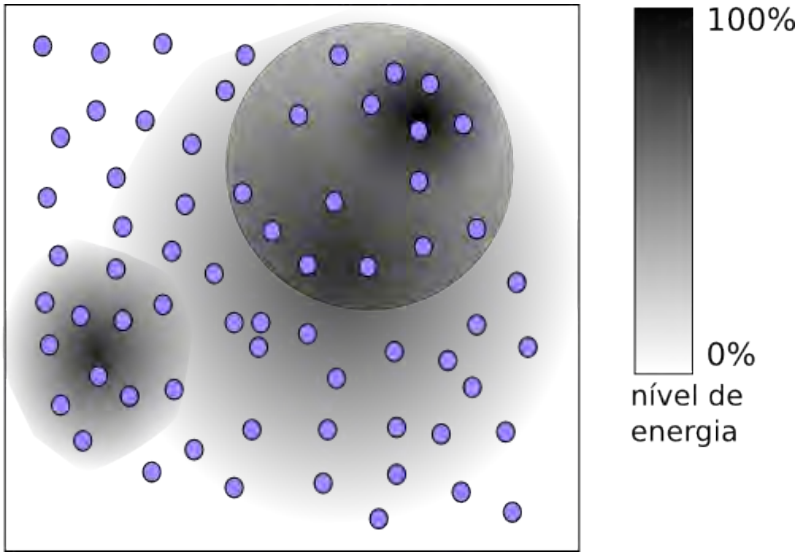

Figura 2.2: *Exemplo de representação visual de níveis de energia em RSSF*

prejudicados. A localização do nodo sorvedouro, que é o nodo que recebe os dados dos outros nodos, também pode ser definida pelo mapa de energia. Por exemplo, se o roteamento for nodo-a-nodo, quanto mais próximo do nodo sorvedouro um sensor estiver, mais energia ele irá consumir, pois mais dados ele terá que receber dos sensores mais distantes para transmitir ao nodo sorvedouro. Assim, o nodo sorvedouro pode mudar de posição, sendo colocado numa região da rede onde os sensores estão com mais capacidade energética, fazendo com que a vida da rede seja prolongada.

Algoritmos de roteamento também podem fazer uso do mapa de energia escolhendo rotas que utilizem regiões da rede com maior quantidade de energia, poupando assim os sensores com poucas reservas de energia.

Portanto, a energia consumida por uma rede de sensores sem fio é algo absolutamente importante para um projeto e tem sempre que ser levada em consideração, pois para cada aplicação, para cada algoritmo e protocolo utilizado, é necessário um estudo do consumo de energia para garantir um melhor aproveitamento da rede, obtendo assim um melhor resultado da obtenção dos dados.

2.2 Comunicação em redes sem fio

As tecnologias de redes sem fio estão cada vez mais presentes no cotidiano, ou seja, na vida pessoal ou profissional essa tecnologia está sendo cada vez mais utilizada para a diversão e para o trabalho, além de também haver dispositivos para cuidados médicos, segurança pessoal e outras aplicações, todas sem fio. Dessa forma, é cada vez mais difícil pensar na vida sem redes sem fio.

Alguns fatores importantes que devem ser considerados na análise do avanço das redes sem fio são:

- A não dependência de conectividade física, então as chances de uma rede sem fio sobreviver a desastres naturais como furacões, terremotos, enchentes dentre outros são muito maiores;
- Alto nível de disponibilidade com as tecnologias sem fio.

Assim, para diferentes aplicações são utilizadas diferentes protocolos de comunicação de rede sem fio. Hoje há uma quantidade razoável dessas tecnologias, com foco, finalidades e abrangências diferentes. Nesse contexto, é interessante ressaltar algumas tecnologias específicas de redes sem fio para uma maior compreensão do que se pode encontrar atualmente [42].

2.2.1 Bluetooth

O Bluetooth é um protocolo muito utilizado em redes pessoais e em pouco tempo atingiu um grande número de usuários. Essa grande utilização se deve à praticidade fornecida a quem a utiliza.

Por sua abrangência variar de 10 a 100 metros, essa tecnologia é geralmente implementada em dispositivos móveis como telefone celulares, estando disponível para usuários comuns, o que é uma grande inovação, pois geralmente as tecnologias de redes sem fio são primariamente utilizadas para fins comerciais de grande porte.

A tecnologia Bluetooth está tornando possível uma série de inovações. A interação homem-máquina está cada dia maior com sua ajuda. Com isso, a cada dia os fabricantes de dispositivos móveis estão aderindo a esse protocolo e fazendo com que um número maior de usuários tenha acesso ao mesmo.

Essa tecnologia foi desenvolvida inicialmente em 1994 pela Ericsson Mobile Communications com o intuito de unir o mundo da computação com o das telecomunicações através de uma interface de rádio de baixo custo e de baixo consumo entre telefones móveis e seus acessórios. Em 1998, a Ericsson, Nokia, IBM, Toshiba e Intel criaram um forte grupo formado por líderes de mercado em telefonia móvel, computação e processamento denominado Special Interest Group (SIG) com a finalidade de criar um padrão para comunicação sem fio para distâncias entre 10 e 100 metros. Logo depois, em 2003, esse grupo foi expandido, agregando outras grandes empresas totalizando 1790 fabricantes [42].

Características do bluetooth

Atualmente existem três classes definidas para o Bluetooth. De acordo com a classe é possível determinar a distância e o nível de potência de cobertura. O protocolo é

transparente para o usuário e a conexão é estabelecida no momento em que um dispositivo Bluetooth é encontrado. Isso é extremamente importante, pois facilita o manuseio pelo usuário final, que são os proprietários de dispositivos móveis e quem desejar estabelecer uma comunicação entre dispositivos, seus acessórios e também entre dispositivos e computadores. As características dessa tecnologia podem ser vistas na Tabela 2.1:

Tabela 2.1: *Classes de dispositivos Bluetooth*

Classe	Potência	Nível de potência	Cobertura
1	Alta	100mW (20dBm)	até 100 metros
2	Média	2.5mW (4dBm)	até 10 metros
3	Baixa	1mW (0dBm)	0.1 a 10 metros

Fonte [5]

Ao estabelecer comunicação, quando um conjunto de dispositivos Bluetooth forma uma rede, dá-se o nome de piconet. E quando é formado um grupo de piconets dá-se o nome de scatternets. Um mesmo dispositivo pode fazer parte de diferentes piconets. Na conexão, existem os mestres, que são os que iniciam a troca de dados e os escravos são os que sempre respondem aos mestres. Apesar de ser possível um mesmo dispositivo fazer parte de diferentes piconets, ele pode ser mestres de apenas um. Isso tudo são regras de comunicação que formam o protocolo Bluetooth.

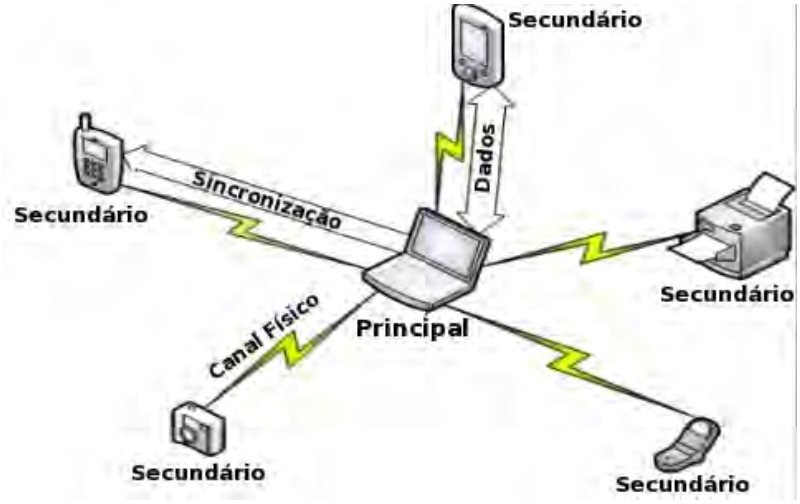

Figura 2.3: *Exemplo de uma piconet, adaptado de [44]*

Assim, o Bluetooth forma uma rede espontânea e *ad-hoc*. Essa expressão vem do latim e significa literalmente “para isto”. Podemos então deduzir que o Bluetooth é uma tecnologia desenvolvida para fins específicos. Nas Figuras 2.3 e 2.4 são exemplificadas algumas redes Bluetooth.

O Bluetooth também tem como objetivo a otimização de diversos dispositivos móveis e segue alguns princípios como: manipulação de voz e dados, evitar interferências

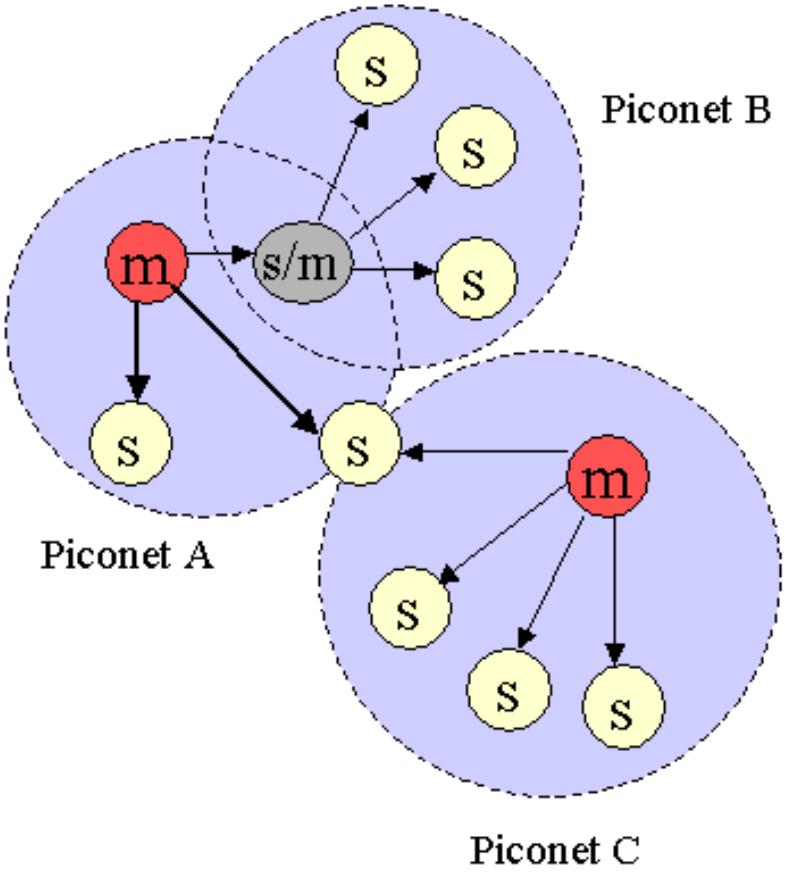

Figura 2.4: *Exemplo de dispositivos que participam de mais de uma piconet [39]*

de outras origens em uma banda aberta, baixo consumo, baixo custo, entre outros. O baixo consumo e custo se devem principalmente ao número mínimo de estágios analógicos, usando circuitos digitais sempre que possível.

2.2.2 ZigBee

O padrão ZigBee foi desenvolvido como alternativa para a comunicação sem fio em redes que não necessitam de soluções complexas para o seu gerenciamento. Sendo uma tecnologia relativamente simples, que faz uso de um protocolo específico, visando oferecer flexibilidade aos dispositivos que controla, essa tecnologia se fez barata, reduzindo custos de aquisição, manutenção e mão de obra.

O padrão ZigBee oferece interfaces com velocidades de conexão entre 10Kbps e 115Kbps e com um alcance de transmissão entre 10 metros e 100 metros, o que depende diretamente dos equipamentos e de características ambientais como interferência eletromagnética e de obstáculos físicos, por exemplo.

Os dispositivos baseados nessa tecnologia operam na faixa ISM (industrial, científica e médica), que não requer licença para funcionamento, isso inclui as faixas

2,4Ghz (Global), com taxa de transferência de dados de 250Kbps, 915Mhz (Estados Unidos da América), com 40Kbps e 868MHz (Europa) com 20Kbps.

Para a alimentação dos dispositivos, os módulos de controle que utilizam essa tecnologia podem ser alimentados até mesmo por baterias comuns e sua vida útil depende da capacidade da bateria e também da aplicação a que se destina. Dessa forma, o protocolo ZigBee foi projetado para suportar aplicações com o mínimo de consumo [67].

Estrutura

Em uma rede ZigBee, pode-se identificar dois tipos de dispositivos, conforme ilustrado na Figura 2.5: FFD e RFD, que serão especificados a seguir:

- **FFD (*Full Function Device*):** Este dispositivo desempenha a função de coordenador da rede e tem acesso a todos os outros dispositivos, podendo funcionar em toda a topologia do padrão. Trata-se de um dispositivo de construção mais complexa.
- **RFD (*Reduced Function Device*):** Este dispositivo é limitado a uma configuração com topologia em estrela e não pode atuar como coordenador da rede. É um dispositivo de construção mais simples e pode comunicar-se apenas com um coordenador de rede.

Podemos perceber pela Figura 2.5 que em uma rede com topologia em estrela precisamos ter ao menos um dispositivo FFD, sendo que os demais podem ser RFD, já em topologias ponto-a-ponto e árvore, todos os dispositivos têm que ser FFD.

Características do padrão ZigBee

O padrão ZigBee apresenta as seguintes características:

- Implementação simples e consumo de baixa potência, com interfaces de baixo custo;
- Dois estados principais de funcionamento: *active* para transmissão e recepção e *sleep* quando não está transmitindo;
- Simplicidade de configuração;
- Densidade elevada de nodos na rede. É possível o funcionamento com um grande número de dispositivos ativos;
- Protocolo simples que permite a transmissão confiável de dados com certos níveis de segurança.

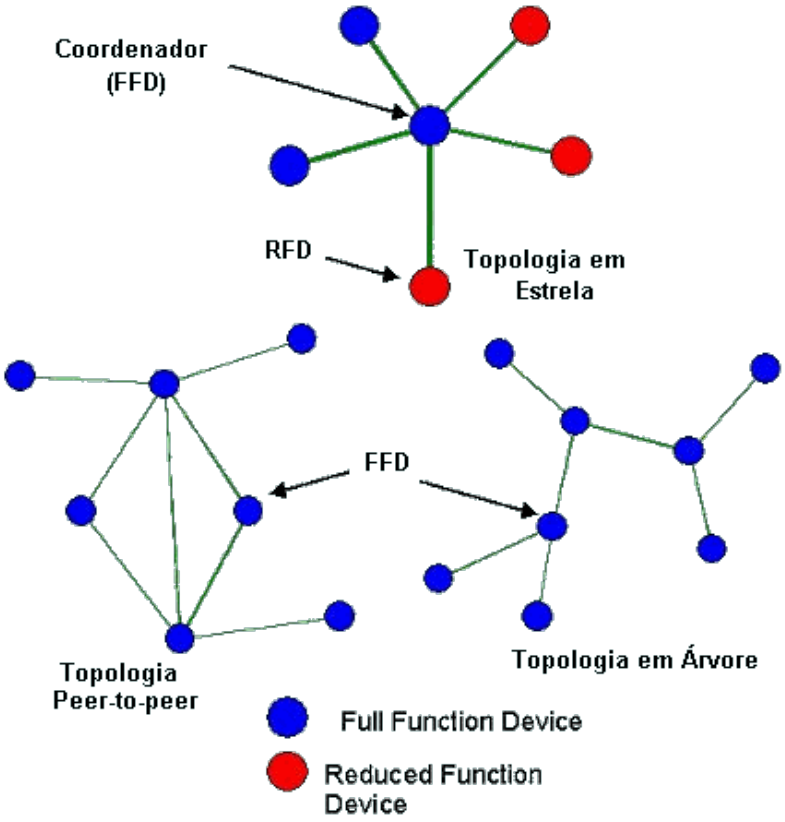

Figura 2.5: *Topologias para redes ZigBee [48]*

Camadas de protocolo

A arquitetura do protocolo ZigBee é composta por camadas, onde uma camada executa serviços específicos para a camada superior. Embora se baseie no modelo OSI de sete camadas, a arquitetura protocolar apenas define as camadas de interesse para atingir as funcionalidades desejadas [8].

A camada física (PHY) acomoda as necessidades de interfaces de baixo custo, possibilitando elevados níveis de integração. O uso da técnica de transmissão de seqüência direta (DSS) permite que os equipamentos sejam muito simples, possibilitando assim implementações mais baratas.

A camada de controle de acesso ao meio (MAC) permite topologias múltiplas com baixa complexidade. Permite também que um dispositivo com funcionalidade reduzida opere na rede sem a necessidade de grandes quantidades de memória. Pode também controlar uma grande quantidade de dispositivos sem precisar deixá-los "em espera", como em algumas tecnologias sem fio.

A camada de rede (NWK) possibilita o crescimento da rede sem a necessidade de equipamentos de transmissão potentes e opera grandes quantidades de nós com latência relativamente baixa. A NWK é responsável pelo início ou fim da ligação de um dispositivo à rede, a detecção de novos dispositivos e a atribuição de endereços. Nesta camada que

estão presentes os mecanismos de descoberta de rotas e encaminhamento de informação assim como de configuração de novos dispositivos. Também é de responsabilidade da camada NWK implementar de pilhas de protocolo que permitem balancear os custos das unidades em aplicações específicas, o consumo das baterias, buscando produzir soluções com o perfil específico de custo-desempenho para a aplicação.

A camada de aplicação (AP) é a interface efetiva entre o ZigBee e o usuário, contendo a maioria dos componentes definidos na especificação do ZigBee.

Os perfis de dispositivos (*Device Profiles*) são agrupamentos convencionados de dispositivos com protocolos de aplicação comuns.

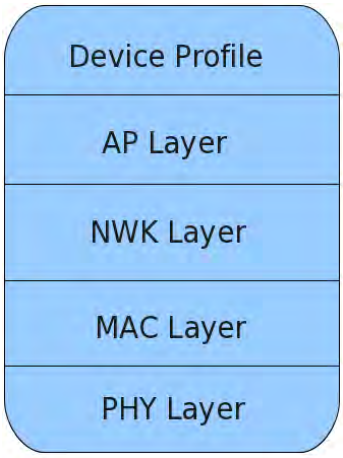

Figura 2.6: *Camadas do protocolo ZigBee*

A Figura 2.6 mostra as camadas da arquitetura do protocolo padrão ZigBee, discutidas anteriormente.

Os custos de acesso em redes sem fio ainda são proibitivos para muitos usuários de redes e de sistemas de automação. A tecnologia ZigBee surge nesse cenário como uma alternativa viável que possibilita a utilização dos sistemas de controle sem fio para esse tipo de aplicação em dispositivos mais simples.

2.3 Considerações finais

As Redes de Sensores Sem Fio (RSSF) foram discutidas neste Capítulo por serem a principal tecnologia utilizada neste trabalho, principalmente pelo fato de serem formadas por dispositivos que interfaceiam o mundo físico com o mundo lógico e ainda possuem uma capacidade, mesmo que pequena, de processamento de dados.

Cada vez mais a utilização de RSSF está fazendo parte das diferentes áreas da pesquisa, indústria, saúde, agricultura, ambiente, segurança, tráfego, militar, entre outras áreas. Através desse tipo de rede é possível monitorar diferentes variáveis em ambientes internos ou externos como prédios ou florestas, tráfego de veículos, segurança

em estacionamentos ou em centros comerciais. As redes de sensores sem fio são também muito utilizadas em diversos setores como na indústria de aviação para monitorar um grande número de variáveis como velocidade, pressão e temperatura, substituindo-se os cabos. Em indústrias de extração de petróleo e gás os sensores são amplamente utilizados, proporcionando segurança e um melhor aproveitamento destes recursos. Na produção industrial, o uso de redes de sensores sem fio acarreta ganhos na produção, maior quantidade de dados para análises e melhoramentos de processos, além de mais segurança em casos de monitoramento de áreas de difícil acesso ou perigosas. Num campo de batalha é possível monitorar a movimentação de inimigos, um gás venenoso ou radiação [1, 35].

Na área de saúde pode-se monitorar órgãos e sinais vitais, facultando o acompanhamento da saúde individual 24 horas por dia, dentro ou fora de uma unidade hospitalar sem muitos inconvenientes, estabelecendo um gerenciamento de saúde proativo e melhor acompanhado em vários casos. Os sensores atualmente podem coletar sinais fisiológicos significantes, como frequência cardíaca, pressão arterial, temperatura corporal e cutânea, sinais eletrocardiográficos (ECG), sinais eletromiográficos (EMG), resistência galvânica da pele (GSR) etc. Desta forma, pode-se considerar as RSSF para a saúde principalmente como um novo meio de monitorar doenças crônicas, idosos, pessoas em reabilitação pós-operatória e pessoas com necessidades especiais [6, 40, 45].

Dentre as RSSF, aquelas voltadas para a área de saúde têm grandes desafios, como manutenibilidade, influência de ruídos do ambiente, determinação de áreas de sobreposição, definição do número de sensores necessários para acurácia dos dados coletados, agregação e roteamento de dados, que foram discutidos neste Capítulo.

Para aplicações na área de saúde, o padrão ZigBee foi escolhido para uso neste trabalho por ter as seguintes vantagens:

(a) Foi projetado para o mínimo consumo energético, estendendo a vida útil dos transdutores;
(b) Tecnologia de custo inferior ao de outras tecnologias, como o Bluetooth;
(c) Baixo tempo para saída do modo de inatividade (*sleep*), 30 milissegundos, em relação ao tempo aproximado de 3 segundos do Bluetooth, possibilitando uma maior permanência em estado de inatividade para maior longevidade da bateria, garantindo-se também a rápida saída deste estado para aplicações de monitoramento em tempo-real;
(d) Mecanismos básicos de segurança embutidos.

CAPÍTULO 3

Middlewares para Redes de Sensores Sem Fio

O objetivo deste Capítulo é apresentar o estado da arte acerca de middlewares para Redes de Sensores Sem Fio (RSSF). Para isso, são apresentados conceitos e tendências que estão em voga. Essas questões são importantes para o entendimento do restante do trabalho porque foram absorvidas pelo projeto de modelagem do middleware proposta.

Os frameworks são um meio de abstração para programação na qual se obtém um grau elevado de reuso de software. Neles são implementados componentes genéricos e definidas as suas interfaces e então, os detalhes específicos do subsistema da aplicação são implementados pela adição de componentes. Os frameworks raramente são aplicações propriamente ditas, mas são estruturas genéricas que podem ser ampliadas para formar subsistemas ou aplicações mais específicas. Os middlewares são uma classe de framework que auxiliam a conexão e troca de informações entre componentes ou aplicações, portanto em suma, são um componente mediador de comunicações.

O uso de frameworks e middlewares em soluções computacionais está facilitando o processo de obtenção de dados e a organização lógica de redes de sensores. Este fato tem se tornado cada vez mais atrativo e aumentado a qualidade dos produtos existentes. Com isso, vários trabalhos são desenvolvidos com estas soluções nos mais diversos níveis de abstração.

De acordo com [12], os middlewares para redes de sensores são comumente divididos em duas classes: (i) suporte a programação e (ii) abstrações de programação. A primeira classe é voltada ao provimento de serviços e mecanismos que sofrem alteração em tempo de execução, já a segunda provê somente abstrações para acesso à rede. Nos interessa neste trabalho o suporte a programação, no qual identificam-se cinco grandes paradigmas baseados nos seguintes aspectos:

- Máquina virtual;
- Programação modular;
- Banco de dados;
- Direcionamento à aplicação;
- Orientação a mensagens.

3.1 Middlewares baseados em máquinas virtuais

O conceito deste modelo está ligado ao conceito homônimo de máquinas virtuais de sistemas distribuídos, no qual é provido um mecanismo arquitetural que interpreta um conjunto de instruções, virtualizadamente, podendo diferir do conjunto de instruções da máquina real.

Este paradigma é bastante flexível por permitir que o programador escreva aplicações separadamente, em módulos, além de oferecer-lhe uma abstração genérica, isto é, as aplicações podem ser executadas em diferentes plataformas. Uma outra vantagem é o isolamento da aplicação com a máquina real pela camada de abstração da máquina virtual, garantindo que não haja acesso inadequado à máquina real, o que poderia comprometer o seu funcionamento.

Entretanto, sofrem alto *overhead* devido à exigência arquitetural da interpretação de instruções pela máquina virtual. Este alto *overhead* traz implicações para aplicações de monitoramento em tempo real, portanto não é ideal para RSSF em área de saúde. A dificuldade em explorar as particularidades da arquitetura trabalhada também se revela como outro problema [12]. Também, como disse Danny Hughes em seu trabalho "Middleware for WSN" [16], este modelo é orientado a nodos embarcados, assume que os sensores não têm capacidade de processamento (são passivos), fazem uma abstração de rede muito simples e fixa e os gastos energéticos são vistos como importantes, mas não críticos.

3.1.1 Maté

Maté [33] é um middleware usado para criação de interfaces de programação para o sistema operacional TinyOS. Seus programas, denominados capsulas ativas móveis, se autopropagam através da rede usando um protocolo de difusão e controle de versão [32]. Ele usa o modo de comunicação síncrono que torna a programação a nível de aplicação mais simples e menos propenso a erros de programação. A mobilidade é abordada ao usar vários protocolos de roteamento *ad-hoc* e atualizações de protocolos.

Os componentes-chave do Maté são o componente máquina virtual, o rede, o registrador de eventos (*logger*), o hardware e o gerenciador de componentes/escalonador.

Todavia, o Maté é indicado para aplicações nas quais os sensores fiquem maior parte do tempo ociosos. Para aplicações complexas, ele pode gastar muito os recursos, devido ao *overhead* causado pela interpretação da máquina virtual.

3.1.2 Magnet

O Magnet [3] foi desenhado com o objetivo de pôr um conjunto de nodos heterogêneos ad-hoc sob uma imagem simples e unificada de sistema. É formado de um

componente estático e de componentes dinâmicos. O componente estático é responsável por particionar aplicações entre os nodos, distribuindo-as transparentemente de forma a reduzir o consumo de energia, aumentando a longevidade do sistema, conectividade de rede e escalabilidade. Então, as instruções para criação de objetos são substituídas por chamadas ao componente estático, que é responsável por selecionar um nodo apropriado e criar uma nova instância do objeto naquele nodo. Os componentes dinâmicos ficam em execução nos nodos monitorando por alterações do sistema e podem migrar entre nodos [56].

3.2 Middlewares baseados em programação modular (agentes móveis)

O princípio desta metodologia é a construção de aplicações o mais modular possível, criando agentes móveis e facilitando a distribuição de componentes pela rede, evitando a transmissão de aplicações inteiras, facilitando a evolução destas. Isto implica o menor gasto de energia, pelo fato da propagação de componentes menores.

Contudo, a transmissão de meta-dados de controle causam alto *overhead*, além da impossibilidade da existência de hardware heterogêneo ou com recursos limitados na rede. Portanto, assim como no paradigma anterior, o alto *overhead* traz implicações para aplicações de monitoramento em tempo real, não sendo ideal para RSSF em área de saúde.

3.2.1 Impala

O Impala [34] propõe uma camada de abstração assíncrona e orientada a eventos, focando na adaptabilidade das aplicações. Por exemplo, um programador pode inserir novos protocolos a qualquer tempo e alternar o uso destes em suas aplicações como quiser. O Impala provê adaptação das aplicações em tempo de execução, o que garante que os erros de programação possam ser filtrados.

Ele é constituído de duas camadas. Na primeira camada, estão os três componentes agentes: o filtro de eventos, o adaptador e o atualizador. O filtro de eventos controla operações diversas e aciona alguns tipos de processamentos como temporizadores, envio e recepção de pacotes e eventos de dispositivos. O adaptador gerencia a adequação das aplicações aos diferentes cenários, não existe um único cenário como, por exemplo, o de eficiência energética. O atualizador gerencia as propagações de novas versões de aplicações, controladas através de números de versão. Na camada superior ficam as aplicações e protocolos, que utilizam uma abordagem de máquina de estados finitos, numa abordagem autonômica. As instruções são binárias e assíncronas.

3.2.2 Medical MoteCare

O Medical MoteCare [43] é baseado no protocolo Simple Network Management Protocol (SNMP), de modo que cada nodo é um agente SNMP (da suíte de software Harvard CodeBlue) gerenciado por uma ferramenta proprietária chamada JAguarSX. Foram definidos vários Object IDentifiers SNMP (OIDs) para os recursos da RSSF, portanto a RSSF pode ser acessado por qualquer ferramenta que comunique via SNMP. O middleware possui uma arquitetura de três camadas que possibilita o controle tanto de recursos compatíveis com SNMP, quanto de recursos não compatíveis, através de um proxy SNMP.

3.3 Middlewares baseados em bancos de dados

Toda a rede de sensores é vista como um banco de dados relacional virtual. Provê uma interface de uso fácil, mas com resultados aproximados, o que lhe faz não possuir suporte para aplicações de tempo real e desta forma não ser ideal para RSSF em área de saúde.

3.3.1 Cougar

O Cougar [64, 65] implementa operações de gerenciamento na forma de consultas similares às de SQL, de modo que o banco de dados de sensores contenha dados armazenados e dados de sensores. Seu uso é mais apropriado quando se tem uma grande quantidade de sensores, por ser fácil a agregação de dados, no entanto, tem um gasto considerável de recursos na transmissão deste montante de dados brutos dos nodos sensores ao servidor de banco de dados. Cada dado é datado, para facilitar as consultas, e então armazenado. Há também suporte ao conceito de *streaming*, no qual pode-se especificar por quanto tempo serão feitas leituras nos sensores e a taxa de atualização e para manter as visualizações persistentes de consultas em curso, os resultados destas são mostrados de forma incremental [15].

3.3.2 TinyDB

O TinyDB é um sistema processador de consultas em estilo SQL para extrair informações de dispositivos sensores. A despeito de outras soluções baseadas em TinyOS [31] (que é um sistema operacional e plataforma para RSSF), ele não exige que o usuário escreva código embarcado para os sensores, a única complexidade é a própria interface SQL. É também provida uma API Java para escrita de programas na estação de trabalho [37, 36].

Assim como no Cougar, as consultas podem fazer agrupamentos por meio de funções de agregação, o que produz ganhos consideráveis de banda e de energia, reduzindo o *overhead* de dados a ser transferidos. As consultas podem ser interrompidas ou terminadas por eventos gerados por outras consultas ou por alguma aplicação rodando em um nodo sensor. São construídas árvores semânticas para roteamento. O TinyDB envia as consultas para todos os nodos, o que demanda problemas de escalabilidade, já que a maioria das aplicações em redes de sensores tem comportamento localizado. Para RSSF no corpo humano (BSN), a escalabilidade do TinyDB não é um problema, no entanto se estendermos esta rede para toda o perímetro hospitalar, por exemplo para a identificação de movimentação de pacientes e profissionais e de equipamentos, a rede pode ter um número consideravelmente alto de elementos, já que os sensores deverão ser espalhados por toda a estrutura, assim como em todas as pessoas e equipamentos que ingressarem nesta estrutura hospitalar. Cada adição de aplicação requer modificação do processador de consultas de cada nodo sensor [12, 15].

3.3.3 SINA (System Information Networking Architecture)

O SINA [58] usa abstração de banco de dados em planilhas para monitoramento e consulta a sensores. Esta abstração é oferecida não somente através de consultas similares às SQL, mas também através de uma linguagem de *script* chamada SQTL (*Sensor Query and Tasking Language*). A rede é o conjunto de folhas da planilha e as células são atributos, na forma de valores simples como nível de bateria e localização ou valores múltiplos como histórico de mudanças de temperatura. Cada célula é única e cada sensor mantém toda a folha de dados. É desenvolvido um esquema de *cluster* hierarquizado, isto garante redução no consumo de energia, escalabilidade e uma mobilidade relativa [12, 15].

3.3.4 DsWare (Data Service Middleware)

O DsWare [66] é um middleware voltado para detecção de eventos e tem suporte à tomada de decisões em grupos de sensores. Provê às aplicações serviços como armazenamento e cache de dados, gerenciamento de grupos, detecção de eventos, subscrição de dados e escalonamento.

3.4 Middlewares direcionados a aplicação

Os middlewares direcionados à aplicação implementam uma arquitetura que permite a manipulação das pilhas de protocolos de rede, ou seja, permite ajustar a rede aos requisitos da aplicação. Isto, por um lado pode permitir o uso mais otimizado da rede

quanto, por outro lado, pode resultar em redes de sensores especializadas, excluindo-se a característica desejável de redes de propósito geral.

3.4.1 MiLAN (Middleware Linking Applications and Networks)

O MiLAN [14] é um middleware caracterizado especialmente por sua habilidade de controlar continuamente as funcionalidades de rede de acordo com as demandas das aplicações executadas. Ele monitora os componentes disponíveis e recursos como consumo energético e largura de banda. Através da API de alto nível desse middleware as aplicações podem declarar dinamicamente suas necessidades (requisitos de qualidade de serviço; a importância relativa de certas aplicações em relação a outras para o sistema ou para os usuários; áreas de alta atividade; etc) [41].

3.4.2 COSMOS (Common System for Middleware of Sensor Network)

O COSMOS [27, 28] é um middleware focado para execução de várias aplicações na RSSF. Ele permite a definição de políticas para gerenciamento e controle da rede, otimização de consultas, integração de dados entre vários sensores, manipulação de eventos, mineração de dados e processamento de informações de contexto. Foram definidas algumas interfaces para interação da aplicação com o middleware e foi sugerido a adoção dessas interfaces como padrão para redes de sensores ubíquas (USN), além de não somente para RSSF.

3.5 Middlewares orientados a mensagens

O modo de comunicação usado neste tipo de middleware é baseado no mecanismo de publicação-subscrição (*publish-subscribe*) de mensagens. Este paradigma garante de modo simples o suporte à comunicação assíncrona e à comunicação simplificada entre os nodos e é bastante adequado aos ambientes pervasivos, em especial àqueles de monitoramento, onde a maioria das aplicações são baseadas em eventos. Por outro lado, as mensagens transportadas têm alto overhead de metadados de controle, o que é um fator desfavorável a esta abordagem de middleware.

3.5.1 Mires

O componente central do Mires [57] é o mecanismo de publicação-subscrição, a comunicação é assíncrona e baseada no paradigma “Active Message” [7]. Este middleware foi modelado para executar sobre TinyOS [31] (que é um sistema operacional e

plataforma orientado a eventos para RSSF), exigindo que o usuário escreva código embarcado, em linguagem NesC, para os sensores. Além deste, ainda existe o componente de roteamento *multihop* e alguns serviços adicionais, como agregação de dados.

3.6 Avaliação geral

Após a breve avaliação de características de alguns middlewares para RSSF existentes, realizada neste Capítulo, elucidando vantagens e desvantagens de cada um deles, apresenta-se um comparativo na Tabela 3.1, baseada no modelo proposto em [12].

A facilidade de uso é definida pelo nível de abstração que o middleware proporciona, ou seja, o quanto se diminui da necessidade de interfaceamento com interfaces de baixo nível. A definição de middleware aberto define a capacidade de modificação e extensibilidade das funcionalidades do middleware. As outras características analisadas nesta avaliação foram discutidas anteriormente.

Nome do projeto	**Otimização energética**	**Aberto**	**Escalabilidade**	**Mobilidade**	**Heterogeneidade**	**Facilidade de uso**
Baseados em máquinas virtuais						
Maté [33], Universidade da Califórnia, Berkeley	Sim	Sim	Sim	Sim	Parcial	Pouca
Magnet [3], Universidade Cornell	Sim	Sim	Sim	Sim	Parcial	Sim
Baseados em bancos de dados						
Cougar [65], Universidade Cornell	Parcial	Pouco	Pouca	Pouca	Pouca	Sim
SINA [58], Universidade de Delaware	Sim	Pouco	Pouca	Pouca	Pouca	Sim
DsWare [66], Universidade da Virgínia	Sim	Parcial	Parcial	Pouca	Pouca	Sim
TinyDB [36], Universidade da Califórnia, Berkeley	Sim	Parcial	Parcial	Parcial	Parcial	Sim

Nome do projeto	Otimização energética	Aberto	Escalabilidade	Mobilidade	Heterogeneidade	Facilidade de uso
Baseado em programação modular / agentes móveis						
Impala [34], Universidade de Princeton	Sim	Sim	Sim	Sim	Pouca	Sim
Medical MoteCare [43], University of Technology Sydney	Parcial	Pouco	Sim	Sim	Pouca	Sim
Direcionado a aplicação						
Milan [14], Universidade de Rochester	Sim	Sim	Sim	Pouca	Pouca	Sim
COSMOS [28], Electronics and Telecommunications Research Institute, Korea	Sim	Parcial	Sim	Sim	Pouca	Sim
Orientado a mensagens						
Mires [57], Universidade de Pernambuco	Sim	Sim	Sim	Parcial	Parcial	Sim

Tabela 3.1: *Abordagens de middlewares para redes de sensores*

3.7 Considerações finais

Neste Capítulo foram apresentados middlewares para RSSF tanto da área de saúde quanto de propósito geral e com isto foi possível construir uma fundamentação para o desenvolvimento da proposta deste trabalho. A Figura 3.1 ilustra as categorias com os respectivos middlewares para RSSF comentados neste Capítulo e inclui o middleware proposto neste trabalho, denomidado **Kratos**, que mescla os conceitos de orientação a aplicação e mensagens, que permitem uma adaptação estrita e um alto nível de especialização do middleware com a aplicação, assim como o suporte a comunicação focada em eventos, que tem uma boa relação com o monitoramento de condições anormais de sinais fisiológicos. Outra característica bastante importante é a necessidade de padronização para interoperabilidade entre sensores e atuadores de diferentes fabricantes com o middleware, o que será discutido nos próximos capítulos.

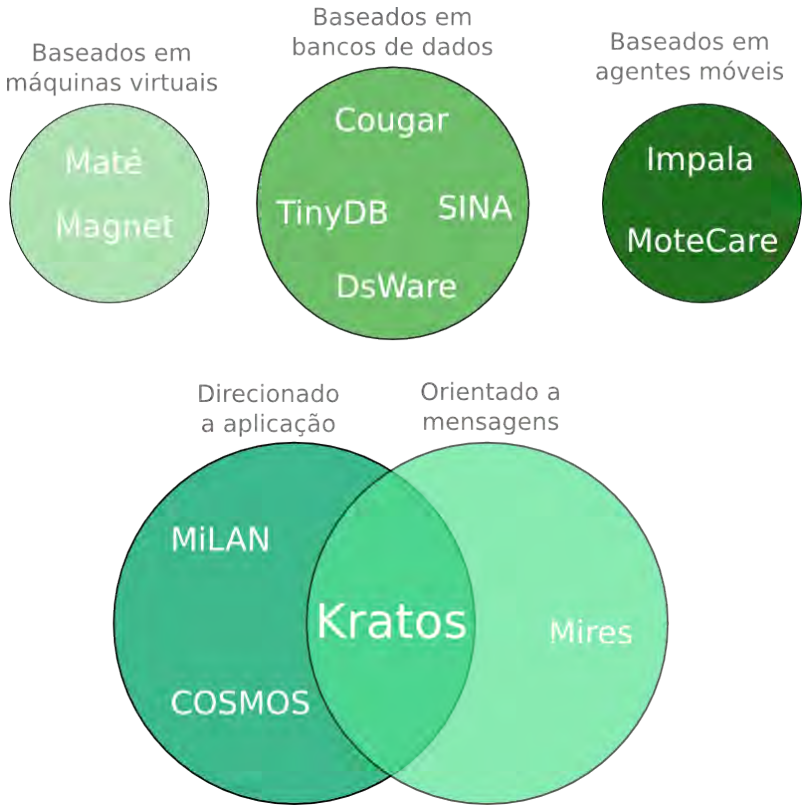

Figura 3.1: *O middleware Kratos em relação aos middlewares analisados neste Capítulo*

CAPÍTULO 4

Família de padrões IEEE 1451

Neste capítulo serão apresentados alguns dos padrões existentes, ligados a redes de sensores, que serão usados na proposta deste trabalho.

4.1 A família de padrões IEEE 1451

A IEEE 1451 é uma família de padrões de interfaceamento para transdutores (que podem ser sensores ou atuadores) que têm como objetivo facilitar a comunicação entre os transdutores e redes, sistemas e instrumentos. O propósito destes padrões é acabar com o problema de interoperabilidade entre transdutores de diferentes fabricantes que, do ponto de vista do software, cada um tem seu conjunto de interfaces proprietárias e na maioria das vezes incompatíveis [30] .

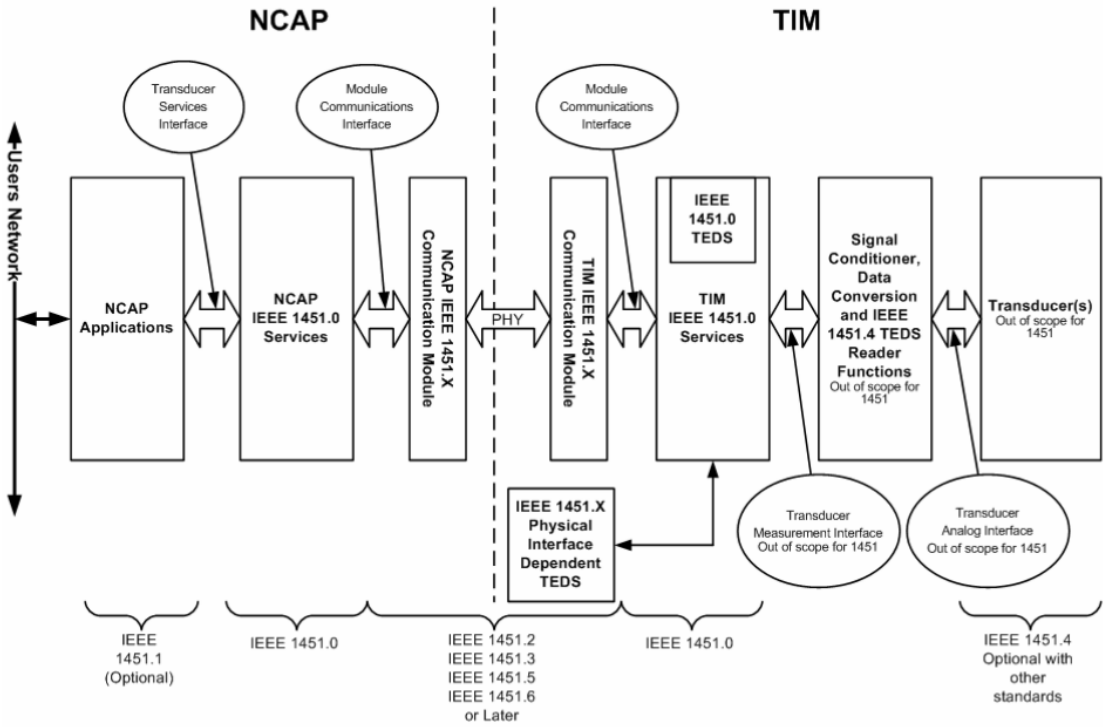

Figura 4.1: *Modelo de referência da família IEEE 1451 [17]*

Tabela 4.1: *Família de padrões IEEE1451*

Padrão	Descreve
IEEE 1451.0 [17]	Um conjunto de comandos e operações, além de definir TEDS, comuns para transdutores. O objetivo é fazer com que a comunicação entre um transdutor qualquer e um nodo de rede, denominado NCAP (*Network Capable Application Processor* - Processador de Aplicação com Interfaceamento em Rede), seja independente do meio físico. A IEEE 1451.0 tenta compatibilizar e servir de base para todas as outras normas da família.
IEEE 1451.1 [18]	Modelos de mensuração e amostragem e modelos de comunicação, como cliente-servidor e publicação-subscrição. O objetivo é fazer com que a comunicação entre os NCAP e os sistemas seja realizada de modo neutro.
IEEE 1451.2 [19]	Uma interface de comunicação entre transdutores e NCAP e TEDS para comunicação ponto a ponto.
IEEE 1451.3 [20]	Uma interface de comunicação entre transdutores e NCAP e TEDS para comunicação em arquitetura distribuída.
IEEE 1451.4 [21]	Uma interface para transdutores com modos de operação analógico e digital. Define também um modelo de TEDS em que o transdutor pode armazenar uma pequena quantidade de dados.
IEEE 1451.5 [22]	Uma interface de comunicação entre transdutores e NCAP e TEDS para comunicação sem fio usando protocolos de comunicação tais como 802.11 (WiFi), 802.15.1 (Bluetooth) e 802.15.4 (ZigBee).
IEEE 1451.7 [23]	Um esboço de interface de comunicação entre transdutores e NCAP e TEDS para comunicação com Radio Frequency Identification (RFID).

Uma breve descrição do conteúdo dos padrões da família IEEE 1451 encontra-se na Tabela 4.1, e a relação entre os padrões da norma IEEE 1451 pode ser encontrado na Figura 4.1.

A partir das descrições da Tabela 4.1, verifica-se que as normas IEEE 1451.0 e 1451.5 melhor se relacionam com o uso em redes de sensores. A norma IEEE 1451.0 tem como item mais relevante a definição de um dispositivo de memória embutido no transdutor, chamado *Transducer Electronic Data Sheets (TEDS)* (em português: Folha de Dados Eletrônica do Transdutor), que armazena dados de identificação, calibragem, correção de dados, faixas de medição, informações de fabricação dentre outras informações, além de definir um conjunto de interfaces de API usadas para acesso às características e funcionalidades do transdutor. Esta norma também define modelos de cliente-servidor e publicação-subscrição independentes de protocolos de rede. A norma IEEE 1451.5 define um protocolo para comunicação sem fio com transdutores. Pelo exposto, neste trabalho usaremos somente esses dois padrões que serão melhor detalhados nas seções seguintes. Por convenção, o termo “IEEE 1451.X” ou “1451.X” será utilizado para referenciar aos padrões IEEE 1451.2, 1451.3, 1451.5 e 1451.6, excluindo-se os padrões 1451.1 e 1451.4.

4.2 Padrão IEEE 1451.0

O padrão IEEE 1451.0 tem o título "*IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats*", em português, "Padrão IEEE para Interface de Transdutores Inteligentes para Sensores e Atuadores - Funções Comuns, Protocolos de Comunicação e Formatos de Folha de Dados Eletrônica do Transdutor (TEDS)". Este padrão tem como conceitos chave a definição de *Transducer Interface Module* (TIM) (em português: Módulo de Interface com Transdutores) e *Network Capable Application Processor* (NCAP) (em português: Processador de Aplicação com Interfaceamento em Rede) que são conectados por meio de uma mídia especificada por um dos padrões IEEE 1451.2, 1451.3, 1451.4, 1451.5, 1451.6 ou 1451.7. Um TIM contém as interfaces e os meios para acesso ao transdutor e este transdutor pode ser um componente do próprio TIM, sendo que ele pode variar em complexidade interfaceando desde um único transdutor a vários. Um NCAP intermedeia a comunicação entre um TIM e uma rede usuária ou um host processador. Os Conceitos de TIM e NCAP serão melhor abordados nas seções seguintes.

Um transdutor é considerado inteligente pelo fato de ser descrito por um TEDS que é interpretável por máquina, o controle e dados associados ao transdutor são digitais, e gatilhos, status e controle existem para prover o funcionamento do transdutor [17].

A seguir serão mostrados os conceitos mais relevantes do padrão IEEE 1451.0 para este trabalho.

4.2.1 Universal Unique Identification – UUID

O UUID é um campo único e universal de identificação associado ao TIM que contém 10 octetos e é definido pelo fabricante. A Tabela 4.2 mostra os campos do UUID na ordem que devem ser dispostos.

4.2.2 TransducerChannel

O **TransducerChannel** é considerado o conjunto do transdutor e todos os componentes de condicionamento e conversão de sinais.

São definidos dois níveis de endereçamento no padrão. Um nível associado à implementação de camada física, relacionado ao identificador de descoberta ("`destId`") associado durante a fase de descoberta dos TIMs e NCAPs e permite a comunicação entre um TIM e um NCAP ou entre TIMs. A fase de descoberta será melhor detalhada nas próximas seções. O outro nível é representado por um número de 16 bits chamado número de TransducerChannel que é usado para mensagens de comandos ou resposta e instrui ao

Tabela 4.2: *Estrutura dos UUID*

Campo	Descreve	Bits
1	**Localização**: Este campo pode representar a localização do fabricante do TIM ou algum outro valor dentro dos requisitos deste campo. O MSB indica posicionamento latitudinal norte ou sul (níveis alto e baixo, respectivamente) da localização. Os próximos 20 bits mais significativos são um número inteiro que representa a magnitude da latitudinal em arco-segundos. O próximo bit mais significativo indica longitude leste ou oeste (níveis alto e baixo, respectivamente). Os 20 bits restantes são um número inteiro que representa a magnitude longitudinal em arco-segundos.	42
2	**Fabricante**: Este campo é também definido pelo fabricante e deve ser negociado com outros fabricantes. A combinação dos campos Localização e Fabricante define de forma não ambígua um fabricante de TIMs.	4
3	**Ano**: A valor deste campo deve ser o ano da fabricação do dispositivo.	12
4	**Tempo**: Deverá conter o número de segundos desde o início do ano de fabricação do dispositivo. A representação é feita com um inteiro com intervalos de 10s. Deverá ser considerado como início do ano 1 de janeiro, 00:00:00, TAI.	22

TIM como a mensagem deve ser direcionada ao NCAP ou a origem da mensagem através do TIM. As regras de numeração deste segundo nível estão descritas na Tabela 4.3.

Tabela 4.3: *Regras para uso do número de TransducerChannel*

Classe		Valor	Descrição
Global		$0xFFFF$	Endereçamento especial para todos os TransducerChannels no TIM endereçado. Se o comando não estiver implementado, deverá ser ignorado.
AddressGroup	**Bit Mapped**	$A > 0x8000$ $A \leq 0xBFFF$	Usado quando forem necessários poucos AddressGroups. O bit mais significante vale um, o próximo bit mais significante vale zero e os outros bits identificam o(s) grupo(s). O endereço $0x8000$ não é funcional.
	Binário	$A \geq 0xC000$ $A \leq 0xFFFE$	Usado quando for necessário um grande número de AddressGroups. O dois bits mais significantes valem um e os demais bits identificam um grupo de endereços.
TransducerChannel		$A \geq 1$ $A \leq 0x7FFF$	Endereço com o bit mais significante igual a zero. Os outros bits identificam o TransducerChannel destinatário da mensagem.
TIM		0	Quando o endereço é zero indica que a mensagem é destinada ao TIM e não a um TransducerChannel.

4.2.3 Comandos

Os comandos estão divididos em duas categorias: os padrões e os definidos por fabricante. Eles são representados em dois octetos, sendo que o mais significante representa a classe do comando e o menos significante indica a função.

Estrutura de uma mensagem de comando

O cabeçalho de uma mensagem de comando possui seis octetos e está representada na Tabela 4.4. Os octetos dependentes do comando contêm informação relevante para o comando.

Tabela 4.4: *Estrutura de uma mensagem de comando*

7	6	5	4	3	2	1	0
Número do Transducer-Channel destinatário		Classe do comando	Função do comando	Tamanho da carga útil		Octetos dependentes do comando (carga útil)...	

Classes de comandos

Tabela 4.5: *Classes de comandos padrões*

cmdClassId	Nome do atributo	Categoria
0	`Reservado`	Reservado
1	`CommonCmd`	Comandos comuns aos TIM e aos Transducer-Channel
2	`XdcrIdle`	Transdutor em estado ocioso
3	`XdcrOperate`	Transdutor em modo operante
4	`XdcrEither`	Transdutor em estado ocioso ou operante
5	`TIMSleep`	Comandos de TIM em estado dormindo
6	`TIMActive`	Comandos de TIM em estado ativo
7	`AnyState`	Qualquer estado
8-127	`ReservedClass`	Reservado
128-255	`ClassN`	Aberto para fabricantes – N = número da classe

As classes de comandos definidas no padrão estão listadas na Tabela 4.5.

4.2.4 Resposta

Um TIM pode gerar uma resposta para um comando sob duas circunstâncias: quando o comando requer uma resposta ou quando o protocolo `status-event` estiver ativado.

Estrutura de uma mensagem de resposta

O cabeçalho de uma mensagem de resposta possui três octetos e está representada na Tabela 4.6. Os octetos dependentes da resposta contêm informação relevante para a resposta.

Tabela 4.6: *Estrutura de uma mensagem de resposta*

7	6	5	4	3	2	1	0
Flag de Sucesso/Falha	Tamanho da carga útil		Octetos dependentes da resposta (carga útil)...				

4.2.5 Triggers

Um trigger é um sinal aplicado a um TranducerChannel ou a um conjunto deles para acionar uma ação específica. Um comando de trigger deve ser endereçado a um TransducerChannel, a um TransducerChannel proxy, a um AddressGroup ou globalmente. Na Figura 4.2 encontra-se um diagrama de estados do comportamento de triggers em sensores e na Figura 4.3 encontra-se o diagrama respectivo para atuadores.

4.2.6 Estados de operação

Um TransducerChannel possui basicamente dois estados após sua inicialização, que são o estado Operando (*Transducer Operating*) e o estado Ocioso (*Transducer Idle*), como ilustrado na Figura 4.4. O TransducerChannel entra em modo Operando ao receber o comando `TransducerChannel Operate` e permanece neste estado até

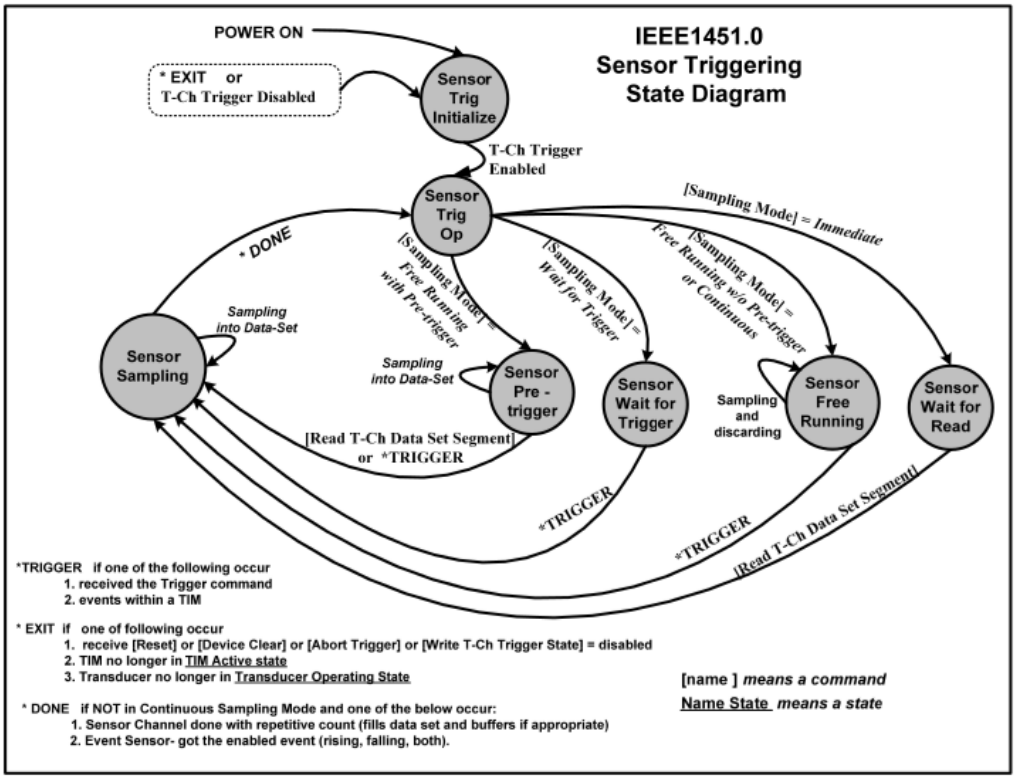

Figura 4.2: *Estados de triggers em sensores [17]*

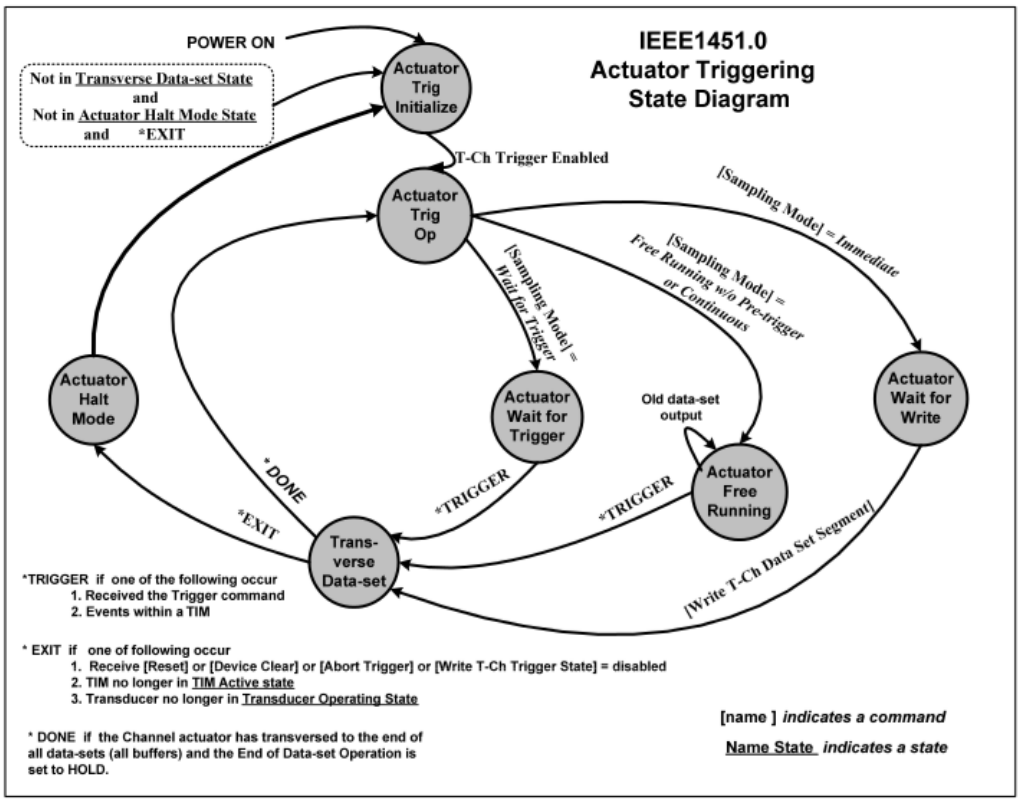

Figura 4.3: *Estados de triggers em atuadores [17]*

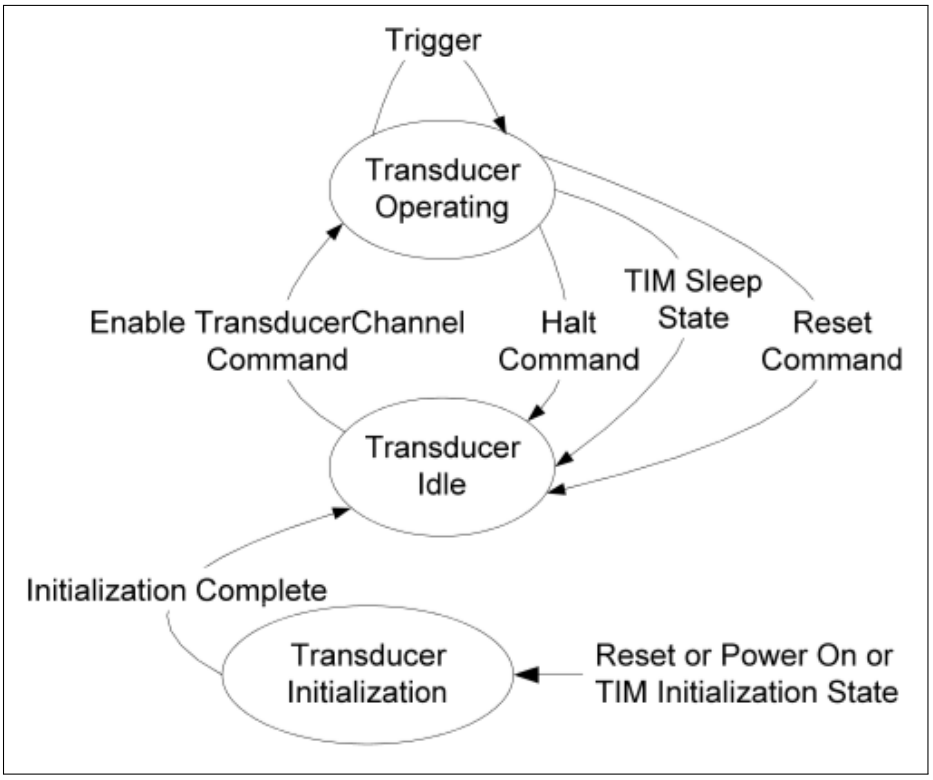

Figura 4.4: *Estados de operação de um TransducerChannel [17]*

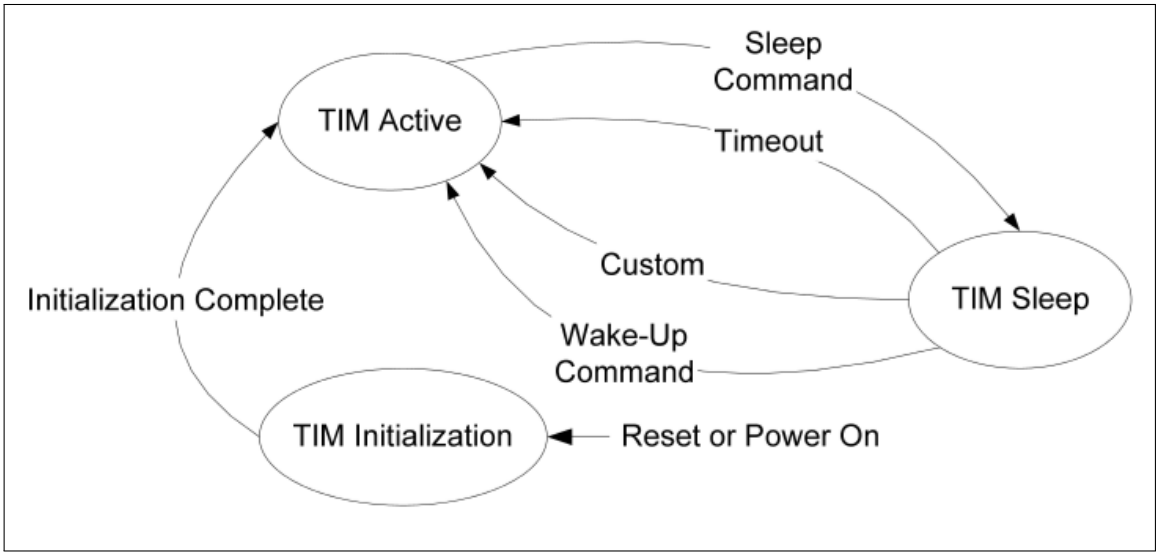

Figura 4.5: *Estados de operação de um TIM [17]*

receber o comando `TransducerChannel Idle`. O TransducerChannel permanece no modo Ocioso na maior parte do tempo.

Da mesma forma que um TransducerChannel, um TIM possui dois estados após a inicialização, que são o estado Ativo (`TIM Active`) e o estado Dormindo (`TIM Sleep`), como ilustra a Figura 4.5. O TIM entra em modo Ativo ao inicializar, ou ao receber o comando `Wake-up`, e entra em modo Dormindo pelo comando `TIM Sleep` A lista de comandos aceitos por um TIM em modo ativo encontra-se na Tabela 4.7 e a lista de

comandos aceitos por um TIM em qualquer estado encontra-se na Tabela 4.9. Os números de versão da norma IEEE 1451.0, necessários para o comando "`Ler versão da IEEE 1451.0`" encontram-se na Tabela 4.8.

Tabela 4.7: *Comandos para um TIM em estado Ativo*

cmdFunctionId	Comando	Endereçamento TIM	Endereçamento Global	Espera resposta	Obrigatório
0	Reservado	–	–	–	–
1	Ler versão do TIM	Sim	Não	Sim	Sim
2	Dormir (TIM Sleep)	Sim	Não	Não	Não
3	Armazenar configurações operacionais	Sim	Não	Não	Sim
4	Revocar configurações operacionais	Sim	Não	Não	Sim
5	Ler versão da IEEE 1451.0	Sim	Não	Sim	Sim
6-127	Reservado	–	–	–	–
128-255	Aberto para fabricantes	–	–	–	–

Tabela 4.8: *Números de versão da IEEE 1451.0*

Versão	Descrição
0	Reservado para protótipo ou outra versão não padronizada
1	Corresponde à versão original do padrão
2-127	Reservado para futuras versões do padrão
128-255	Aberto para fabricantes

Tabela 4.9: *Comandos para um TIM em qualquer estado*

cmdFunctionId	Comando	Endereçamento TIM	Endereçamento Global	Espera resposta	Obrigatório
0	Reservado	–	–	–	–
1	Reset	Sim	Não	Não	Não
2-127	Reservado	–	–	–	–
128-255	Aberto para fabricantes	–	–	–	–

4.2.7 Modos de amostragem

Os transdutores podem operar em cinco modos de amostragem, descritos a seguir. Os modos de amostragem que um `TransducerChannel` é capaz de operar devem estar descritos em seu TEDS. Os modos de operação `trigger-initiated` e `free-running` são mutuamente exclusivos e os outros modos são variações destes dois modos básicos.

Trigger-initiated

Neste modo de operação o sensor deverá adquirir dados ao receber um comando *trigger*. Um atuador deverá executar os dados assim que receber um comando *trigger*. O processamento da amostragem continua até que todas as amostras do conjunto de dados sejam processadas na taxa especificada pelo TIM.

Free-running without pre-trigger

No modo `free-running` um sensor obtém, converte e armazena contínua e autonomamente algum parâmetro físico, descartando leituras quando o conjunto de dados estiver completo. No momento do recebimento do comando *trigger*, o sensor envia os dados coletados e recomeça o processo contínuo de aquisição de dados. Um atuador neste modo deve aplicar os dados recebidos até o recebimento do trigger em concordância com o modo de operação `end-of-data-set` (4.2.9).

Free-running with pre-trigger

No modo `free-running` um sensor obtém, converte e armazena contínua e autonomamente algum parâmetro físico, até que a quantidade de amostras atinja o número de contagem de pre-*trigger* ou até o recebimento de um comando *trigger*. No momento do recebimento do comando *trigger*, o sensor envia os dados coletados e recomeça o processo contínuo de aquisição de dados. Quando o conjunto de dados está completo o sensor deve se comportar segundo os modos "`Free-running with pré-trigger without buffers enabled`" e "`Free-running with pre-triggers and buffers enabled`". Um atuador não deve operar neste modo.

Free-running with pre-trigger without buffers enabled

Quando um conjunto de dados estiver completo, o `TransducerChannel` começa a descartar amostras até que um comando *trigger* seja recebido, ou até que o conjunto de dados seja lido. Neste modo, os dados podem ser lidos uma única vez, as leituras subsequentes deverão retornar zero octetos.

Free-running with pre-trigger and buffers enabled

Quando um conjunto de dados estiver completo, o `TransducerChannel` passa para o próximo *buffer* vazio e continua coletando amostras até que não existam mais *buffers* vazios. Caso não haja *buffers* disponíveis, ele deve descartar as amostras coletadas até a disponibilidade de um *buffer*. Um *buffer* lido deve ser considerado vazio.

Continuous sampling mode

Neste modo o sensor deve começar a coletar e armazenar amostras em um de seus *buffers* quando receber um comando de *trigger* inicial. O modo de operação é similar ao "`free-running without pre-trigger`", exceto pelo fato de não parar no fim da aquisição de um conjunto de dados, mas continuar a coletar dados no próximo *buffer*. Quando todos os *buffers* estiverem cheios, o mais antigo é descartado, mesmo que ainda não tenha sido lido.

Immediate operation

No modo de operação imediato, o sensor deve coletar e transmitir uma conjunto de dados subsequentemente ao recebimento do comando de leitura. Um atuador deve aplicar imediatamente o conjunto de dados logo após a sua recepção.

4.2.8 Modos de transmissão

A Tabela 4.10 descreve os três modos de transmissão definidos neste padrão.

Tabela 4.10: *Modos de transmissão de dados*

Num.	Argumento	Descrição
0	`XmitMode.reserved[0]`	Reservado
1	`XmitMode.OnCommand`	A transmissão deve ocorrer somente em resposta a um comando de leitura de conjunto de dados.
2	`XmitMode.BufferFull`	Os dados serão transmitidos tão logo um *buffer* esteja cheio.
3	`XmitMode.Interval`	Os dados serão transmitidos em intervalos fixos, mesmo que o buffer ainda esteja incompleto. O número de amostras no conjunto de dados deve ser determinado pela taxa de amostragem e pelo intervalo de transmissão.
4-127	`XmitMode.reserved[N]` $4 \leq N \leq 127$	Reservado
128-255	`XmitMode.open[N]` $128 \leq N \leq 255$	Aberto para fabricantes

4.2.9 Modo de operação end-of-data-set

Este modo se aplica somente a atuadores e define o modo como o dispositivo operará após o recebimento de um dado.

Hold

O atuador deverá usar todas as amostras do conjunto de dados e continuar a usar a última amostra do conjunto de dados até o recebimento de um *trigger*.

Recirculate

O atuador deverá aplicar todas as amostras do conjunto de dados e então voltar para o começo do conjunto de dados, repetindo a operação até o recebimento de um *trigger*. Se o atuador receber outra *trigger*, ele deve mudar para o novo conjunto de dados somente após o término do conjunto corrente.

4.2.10 Operação em streaming

A combinação do modo amostragem "`continuous sampling`" com o modo de transmissão "`BufferFull`" ou com o modo "`Interval`" é chamada de operação em *streaming*. O sensor deve coletar e transmitir dados ao NCAP sem a necessidade de comandos de requisição individuais. Do mesmo modo, um atuador deve aplicar um conjunto de dados sem a necessidade de um comando *trigger* para cada conjunto de dados.

4.2.11 Especificações de TEDS

Os TEDS são blocos de informação que devem ser armazenados em memória não volátil do TIM; com exceção nos casos em que este armazenamento não seja praticável. Eles geralmente são escritos pelo fabricante e não são mais alterados, no entanto pode-se construir `TransducerChannels` que alteram os TEDS em tempo de execução. Os TEDS podem ser armazenados em localizações externas ao TIM, neste caso eles são referenciados como "TEDS virtuais".

São exigidos quatro tipos de TEDS para todos os TIMs:

- **Meta-TEDS**: contém informações para disponibilizar acesso a qualquer `TransducerChannel`, juntamente com informações comuns a todos os seus `TransducerChannels`;
- **TransducerChannel TEDS**: contém informações detalhadas de um transdutor específico de forma a operacionalizar meios adequados para acesso ao transdutor;

- **TEDS de nome do transdutor para o usuário**: provê um local para que o usuário armazene o nome pelo qual o sistema conhecerá o transdutor;
- **PHY TEDS**: TEDS dependente da mídia física de comunicação usada para conectar o TIM ao NCAP, definido nas IEEE 1451.X.

Os TEDS têm o formato especificado de três campos:

1. **TEDS Lenght**: O primeiro é um inteiro sem sinal de 4 octetos que é o total de octetos do bloco de dados do TEDS , mais 2 octetos do *checksum*.
2. **Bloco de dados**: Apresenta conteúdo informativo e possui estrutura que pode variar de tamanho e pode ser tanto textual quanto binária, dependendo da especificação do TEDS.
3. **Checksum**: Complemento da soma (módulo 2^{16}) de todos os octetos precedentes, excluindo-se somente o próprio campo de *checksum*. É um inteiro sem sinal de 2 octetos.

$$checksum = 0xFFFF - \sum_{i=1}^{nr.octetos-2} TEDSOctet(i) \tag{4-1}$$

Os blocos de dados são formados por estruturas de dados chamadas TLV (*Type/Length/Value*), que no caso de TEDS textuais, estas estruturas são usadas para prover um diretório que dê acesso às diferentes porções textuais do TEDS que utilizam conteúdo XML.

4.2.12 Definições de API

A definição de uma API na IEEE 1451.0 facilita a construção de módulos que se integrem e proveem diferentes funcionalidades e ainda tenha várias partes que integram entre si, simplificando a interação entre aplicações de coleta e controle em NCAPs e TIMs.

Os principais serviços desta API são:

- Descoberta de TIMs;
- Acesso a transdutores;
- Gerenciamento de transdutores;
- Gerenciamento de TEDS.

Os objetivos desta API são:

- Prover uma abstração de comunicação independente da tecnologia de conexão (IEEE 1451.X);
- Acomodar as tecnologias de comunicação descritas nas IEEE 1451.X, permitindo o uso de mecanismos mais apropriados;

- Acomodar a variedade de recursos de CPU e memória disponíveis, em todas as suas limitações, nos NCAPs e TIMs;
- Prover mecanismos que possibilitem a camada IEEE 1451.X interceptar chamadas de comunicação compatibilizando-as à rede;
- Prover mecanismos onde aplicações reconhecidas possam enviar comandos pelas camadas IEEE 1451.0 e esses serem reconhecidos em camadas IEEE 1451.X locais ou remotas;
- Prover mecanismos onde aplicações possam enviar comandos particulares para um TIM proprietário sem interpretação nos subsistemas IEEE 1451.0 ou IEEE 1451.X.

Os módulos de API da norma são:

- **TransducerServices**: API pública usada pelas aplicações de coleta e controle para interação com a camada IEEE 1451.0.
- **ModuleCommunications**: API para comunicação entre NCAPs e TIMs via IEEE 1451.X.
- **Args**: Pacote com os argumentos da IEEE 1451.0.
- **Util**: Pacote com classes utilitárias para conversão de `ArgumentArrays` de/para `OctetArrays`.

Módulo TransducerServices

As classes deste módulo, descritas na Tabela 4.11, proveem interfaces entre a aplicação executando no NCAP e as funções deste padrão.

Tabela 4.11: *Classes e interfaces da API TransducerServices*

Interface	Descrição
`TIMDiscovery`	Contém métodos para descoberta de módulos de comunicação, TIMs e TransducerChannels disponíveis.
`TransducerAccess`	Contém métodos para acesso a transdutores.
`TransducerManager`	Contém métodos para controle fino de acesso a TIMs.
`TEDSManager`	Controla leitura, escrita e gerenciamento de cache de TEDS.
`CommManager`	Manipula acesso a módulos de comunicação locais do dispositivo.
`AppCallback`	Interface para aplicações que necessitam características avançadas.

Módulo ModuleCommunication

As classes deste módulo, descritas na Tabela 4.12, proveem interfaces entre este padrão e os outros padrões da família IEEE 1451.

Tabela 4.12: *Classes e interfaces da API ModuleCommunications*

Interface	Descrição
`Comm`	Interface abstrata com mecanismos de controle de uma instância IEEE 1451.X.
`P2PComm`	Operações de comunicação ponto-a-ponto.
`NetComm`	Operações de comunicação em rede.
`Registration`	Operações de registro de um módulo IEEE 1451.X numa camada 1451.0.
`P2PRegistration`	Métodos para registro de TIMs específicos numa camada IEEE 1451.0.
`NetRegistration`	Métodos para registro de TIMs específicos e grupos de TIMs numa camada IEEE 1451.0.
`Receive`	Sem métodos genéricos definidos. Para uso futuro.
`P2PReceive`	Métodos para anunciamento de mensagens ponto-a-ponto recebidas à camada IEEE 1451.0.
`NetReceive`	Métodos para anunciamento de mensagens de rede recebidas à camada IEEE 1451.0.

4.3 Padrão IEEE 1451.5

O padrão IEEE 1451.5 tem o título "*IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats*", em português, "Padrão IEEE para Interface de Transdutores Inteligentes para Sensores e Atuadores - Protocolos de Comunicação em Redes sem Fio e Formatos de Folha de Dados Eletrônica do Transdutor (TEDS)". O escopo deste padrão é fornecer métodos de comunicação padronizados para redes sem fio e formato de dados para transdutores. É introduzido o conceito de Módulo de Interface com o Transdutor sem Fio (*Wireless Transducer Interface Module* – WTIM), que é conectado sem fios via um rádio aprovado a um NCAP. Os módulos aprovados da IEEE 1451.5 (`Dot5AR`) são as tecnologias IEEE 802.11™, IEEE 802.15.4™, IEEE Bluetooth™e IEEE ZigBee™.

Conceitos deste padrão definem que (i) um NCAP pode rotear comandos e dados de uma rede externa de/para um transdutor conectado a um transdutor, (ii) um NCAP pode registrar vários WTIMs, (iii) um WTIM deve ser registrado por apenas um NCAP, (iv) um WTIM pode interfacear vários transdutores, (v) é permitida a comunicação direta entre WTIMs.

4.3.1 PHY TEDS

Como citado em 4.2.11, as PHY TEDS são dependentes da mídia de comunicação física, portanto serão melhor definidas neste padrão. Na Tabela 4.13 está a descrição

do PHY TEDS descrito neste padrão.

Tabela 4.13: *Bloco de dados do PHY TEDS*

Campo	Nome	Descrição	Tipo
–		Tamanho do TEDS	UInt32
3	TEDSID	Identificação do TEDS. Ver Tabela 4.14.	UInt32
4-9		Reservado	
10	Radio	Tipo de rádio. Ver Tabela 4.15.	UInt8
11	MaxBPS	Throughput máximo de dados.	UInt32
12	MaxCDev	Limite de conexão de dispositivos.	UInt16
13	MaxRDev	Limite de registro de dispositivos.	UInt16
14	Encrypt	Tipo de encriptação. Ver Tabela 4.16.	UInt16
15	Authent	Suporte a autenticação	Boolean
16	MinKeyL	Tamanho mínimo para chaves em funções de segurança.	UInt16
17	MaxKeyL	Tamanho máximo para chaves em funções de segurança.	UInt16
18	MaxSDU	Tamanho máximo de payload para transferência.	UInt16
19	MinALat	Latência de acesso mínima.	UInt32
20	MinTLat	Latência de transmissão mínima	UInt32
21	MaxXact	Número máximo de transações simultâneas.	UInt8
22	Battery	Dispositivo energizado por bateria.	UInt8
23	RadioVer	Versão do rádio.	UInt16
24	MaxRetry	Máximo de tentativas até desconectar.	UInt16
25-31		Reservado	
32-41		Específico para Bluetooth.	
42-47		Reservado	
48-54		Específico para rádios 802.11.	
55-63		Reservado	
64-80		Específico para ZigBee.	
81-95		Reservado	
96-103		Específico para 6LoWPAN.	
104-127		Reservado	
128-255		Aberto para fabricantes.	
–		Checksum	UInt16

4.4 Considerações finais

A IEEE 1451 permite à proposta deste trabalho tanto uma melhor comunicação com transdutores padronizados, quanto uma maior coesão entre os seus componentes internos, uma vez que a norma define vários conceitos, comandos, modos de comunicação e transmissão de dados que favorecem a estruturação da modelagem, eximindo a neces-

Tabela 4.14: *Estrutura do Identificador de TEDS (TEDSID)*

Campo	Conteúdo	Descrição
Tipo	03	Tipo do identificador de TEDS.
Tamanho	04	Tamanho do TEDS. Valor sempre igual a 4.
Família	05	Define que o TEDS provem deste padrão.
Classe	13	Identifica o TEDS como PHY TEDS.
Versão		Identifica a versão do padrão adotado. Equivalente ao usado na IEEE 1451.0 descrito na Tabela 4.8
Tamanho da tupla	01	Número de octetos de todas as tuplas nos TEDS, exceto esta. Para o PHY TEDS este valor é sempre 1, simbolizando que existem 255 ou menos octetos.

Tabela 4.15: *Enumeração do identificador de tipo de radio (PHYID – Radio)*

Valor	Tipo de rádio
0	IEEE 802.11
1	Bluetooth
2	ZigBee
3	LoWPAN
4-254	Reservado para expansão futura
255	Reservado para fabricantes

Tabela 4.16: *Enumeração do identificador de criptografia (Encrypt)*

Valor	Significado
0	Sem criptografia
1	AES
2	SAFER
3	3DES
4-254	Reservado para expansão futura
255	Reservado para fabricantes

sidade de sustentar de novos conceitos ou simplesmente a escolha dentre algum conceito necessário já existente.

Na modelagem do middleware **Kratos** foram considerados todos os itens das normas IEEE 1451.0 e IEEE 1451.5 citados neste Capítulo, pois são itens mínimos exigidos para conformidade em ambas as normas. Os demais itens dessas normas poderão ser implementados como componentes auxiliares pelo usuário, dependendo da necessidade de sua aplicação.

CAPÍTULO 5

Middleware Kratos

Vislumbrando o potencial de impacto de RSSF em muitos aspectos da área de saúde, assim como a real necessidade de interoperabilidade, confiabilidade, dentre outros requisitos, foi assim pensado o middleware proposto neste trabalho. Proveniente desses pensamentos fundamentais, uma visão geral do middleware foi construída neste Capítulo com seus objetivos, restrições e benefícios, assim como foi definido o escopo do sistema.

5.1 Visão geral

Existem várias ferramentas computacionais para RSSF, como frameworks, middlewares e bibliotecas tanto de propósito geral, com modelagem mais genérica que geralmente podem ser adaptadas a qualquer domínio de aplicação, quanto ferramentas voltadas para domínios específicos como a área de saúde, militar, aviação ou agrícola, dentre várias outros domínios. Apesar da quantidade de ferramentas, elas geralmente não são de domínio livre ou então são direcionadas a uma classe de transdutores ou domínios específicos.

O middleware proposto possui uma arquitetura modular para que se tenha uma boa manutenibilidade e independência entre partes de código. As interfaces que permitirão tanto a construção de módulos quanto a interoperabilidade são padronizadas e conformes à norma IEEE 1451. Este trabalho propõe a modelagem de um middleware especializado, denominado **Kratos**, para o gerenciamento de Redes de Sensores Sem Fio da área de saúde. Com isto, o desenvolvimento de aplicações para a área de saúde em RSSF simplifica-se por causa da camada de gerência exercida pelo middleware.

O **Kratos**, como é chamado o middleware, tem seu nome de origem grega ($\kappa\rho\alpha'\tau o\varsigma$) e significa regência, domínio. O nome foi dado em razão do poder de regência e controle que o middleware proporciona à RSSF, organizando-a de maneira que atenda aos objetivos das aplicações em execução da melhor maneira possível, fazendo com que os resultados obtidos sejam o mais satisfatórios possível.

Além de ter foco na área de saúde, o middleware será de domínio livre, podendo servir de objeto de estudo para outros pesquisadores, assim como alvo de interesse de

outros desenvolvedores que pretendam melhorá-lo a fim de torná-lo uma ferramenta ainda mais eficiente e com um maior número de funcionalidades para a área de saúde.

5.2 Objetivos e requisitos

Os requisitos de uma RSSF, incluindo as da área de saúde, geralmente são definidos pela aplicação e pelo ambiente de implantação específicos, no entanto, podemos identificar algumas características comuns para aplicações da área de saúde [40, 54]:

Plataformas reduzidas As aplicações médicas em RSSF geralmente exigem transdutores de tamanho reduzido para que possam ser facilmente "vestidos", no entanto, a capacidade dos recursos é diretamente proporcional ao tamanho desses equipamentos, ou seja, possuem uma capacidade bastante limitada. Atualmente, o principal problema com a redução de tamanho são as baterias, que são ligadas à duração do ciclo de vida do transdutor.

Confiabilidade Como se tratam de dados usados para detecção e diagnóstico de possíveis doenças, com informações que devem ser exatas, a coleta desses dados que são sinais fisiológicos deve ser confiável. Um dos meios de se conseguir confiabilidade é através do processamento de sinal no próprio transdutor, em vez de enviar o sinal puro do sensor, assim como extração de características e identificação de eventos, enviando dados somente sob a existência de situações anormais. Isto reduz a demanda do canal de comunicação que é o principal gasto energético em um transdutor e portanto aumentado o ciclo de vida da RSSF.

Segurança Obviamente a segurança de informações sensíveis é trivial para sistemas de saúde, de forma a garantir confidencialidade, integridade, disponibilidade e autenticidade aos dados dos sistemas de saúde. Este quesito é bastante desafiador em RSSF, no entanto, para área médica pode-se conseguir níveis razoáveis de segurança, haja vista a área bem delimitada dos ambientes monitorados (quando comparamos com outros ambientes, como por exemplo uma floresta) e assim conseguimos impor melhores métricas de segurança.

Interoperabilidade A interoperabilidade entre transdutores e sistemas facilita a adoção, manutenção e expansão da RSSF, já que a tecnologia de transdutores e sistemas não fica associada a parâmetros de fabricantes, mas a padrões.

5.2.1 Objetivos

Este middleware tem como objetivos (requisitos funcionais):

- Coletar dados de sensores espalhados em seu perímetro de abrangência, independente da aplicação, no entanto com a semântica correta para agrupamento de regiões de trabalho, por exemplo, o middleware não pode agregar dados de pacientes diferentes;
- Através dos dados coletados dos sensores, gerar ações como acionamento de dispositivos ou sistemas externos;
- Prover uma integração de tempo-real com o mundo real;
- Eliminar redundância de dados, minimizando o número de transmissões e maximizando a vida útil da rede, sem comprometer a acurácia dos dados;

Os requisitos não funcionais do middleware são os seguintes:

- **Interoperabilidade:** capacidade do middleware de interagir com vários sensores em conformidade com a norma IEEE 1451 e outros sistemas;
- **Segurança de acesso:** o middleware deve forçar requisitos de segurança como confidencialidade, integridade, disponibilidade e autenticidade;
- **Recuperabilidade e tolerância a falhas em transdutores:** o middleware poderá se restabelecer caso haja perda de desempenho e recuperar dados em casos de falhas na comunicação;
- **Economia de recursos:** capacidade do middleware de gerenciar os recursos energéticos com parcimônia para um maior tempo de vida dos transdutores;
- **Boa manutenibilidade:** capacidade de localizar falhas para facilitar correções;
- **Alta portabilidade:** capacidade de adaptação a outros ambientes computacionais em relação a hardware e software e possibilidade de reuso em diferentes aplicações;
- **Acurácia na análise dos dados coletados:** capacidade do middleware de analisar com exatidão os dados coletados de uma RSSF.

5.2.2 Peculiaridades

Dos middlewares citados no Capítulo 3, o **Kratos** assemelha-se mais ao middleware Mires, devido fundamentalmente às características citadas na seção anterior. No entanto, uma característica essencial que difere o **Kratos** do Mires é o suporte a heterogeneidade de trandutores que é provida por meio da conformidade com o padrão IEEE 1451, o que quer dizer que ele é capaz de gerenciar uma rede que possua transdutores de várias famílias e fabricantes, desde que também em conformidade com o padrão IEEE 1451. O middleware **Kratos** possui a capacidade de gerenciar uma rede com diferentes tipos de transdutores, independente da família, basta que sejam conformes à norma IEEE 1451.

Essa é uma vantagem importante, pois uma rede com sensores de fabricantes diferentes pode ser usada para uma quantidade maior de aplicações, atendendo a difer-

entes usuários. Isso porque os usuários podem pensar em uma rede para uma aplicação de seu interesse, podendo inclusive criar seus próprios transdutores.

5.3 Características

De acordo com os paradigmas discutidos no Capítulo 3 o **Kratos** é um middleware que mescla orientação a aplicação e orientação a mensagens. Ele integra aplicações distribuídas por meio do uso de mensagens, provendo as habilidades de criação, manipulação, armazenamento e comunicação de dados através do mecanismo de mensagens. Por exemplo: mensagens são utilizadas para a comunicação entre um sistema usuário e os sensores para solicitação de dados, reorganização da rede, desativação de algum sensor, dentre outras atividades.

O fato do middleware ser orientado a mensagens traz algumas vantagens como a entrega confiável sem duplicação de mensagens e o envio e recebimento de mensagens que são feitos de forma assíncrona, possibilitando a execução independente das aplicações e um alto grau de tolerância a falhas. Em um middleware orientado a mensagens a comunicação ocorre basicamente de duas formas: ponto-a-ponto (*point-to-point*), em que as mensagens são enviadas com base no método de comunicação um-para-um e publicação-subscrição (*publish-subscribe*), onde as mensagens são publicadas com base no método um-para-muitos, que é o caso deste middleware [2].

O **Kratos** é um middleware caixa branca. Portanto, para sua utilização, é necessário que o usuário forneça uma implementação específica para a aplicação. Sendo assim, diferentes tipos de implementações são possíveis, já que essas implementações são oriundas de fora do middleware, diferente de um middleware caixa preta, onde todas as possíveis implementações estão em seu interior, cabendo ao usuário apenas escolher entre elas. Além disso, pelo fato do middleware ser construído a partir de detalhes e exigências de ferramentas para a área de saúde, ele é considerado um middleware vertical.

Outras características importantes são:

- **Capacidade de agregação de novas funcionalidades**, fazendo com que o middleware seja capaz de atender a novas expectativas, tornando-o cada vez mais especialista;
- **Independência de tecnologias de rede**; essa característica faz com que o middleware possa se comunicar com os transdutores conectados em diferentes tipos de redes, como por exemplo, um sensor em uma rede sem fios pode enviar dados para o middleware assim como um sensor conectado por uma rede cabeada;
- **Descoberta automática de dispositivos e de capacidades**; o middleware é capaz de detectar imediatamente transdutores que estão em região tangível e seus recursos;

- **Qualidade de serviço**; o middleware prioriza a coleta de dados de sensores que enviam dados mais relevantes para a aplicação. Por exemplo, entre temperatura e frequência cardíaca num monitoramento de um paciente de Unidade de Terapia Intensiva, o middleware dá preferência aos dados coletados pelo sensor de frequência cardíaca;
- **Capacidade de localização e roteamento**; essa característica faz com que o middleware saiba a localização de um sensor em relação a uma referência central e em relação aos outros sensores e também a sua própria localização em relação a RSSF;
- **Escalabilidade**; o middleware é capaz de se adequar às mudanças da RSSF, como o aumento ou diminuição do número de transdutores.

As características citadas fazem do middleware uma ferramenta computacional com funcionalidades importantes para o gerenciamento de uma RSSF, podendo ser utilizado em diversas aplicações de maneira eficaz [9].

5.4 Diagrama de contexto

O diagrama da figura 5.1 mostra o middleware **Kratos** de uma maneira simplificada. O middleware terá três atores diretos:

- **Sistema usuário:** aplicação que usará os recursos da RSSF;
- **Atuador:** dispositivo que alterará as condições do meio no qual está inserido;
- **Sensor:** dispositivo que coletará dados do meio no qual está inserido.

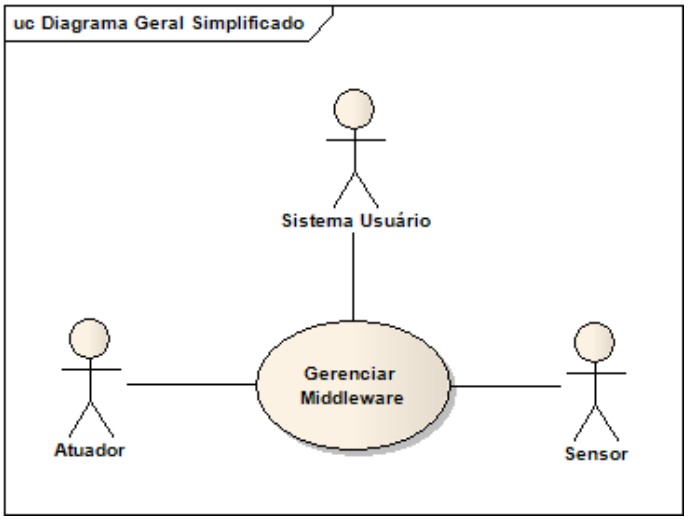

Figura 5.1: *Caso de uso Gerenciar middleware*

5.5 Introdução à arquitetura do middleware

O middleware **Kratos** é do tipo vertical, sendo assim, é dependente do domínio da aplicação da área de saúde, podendo ser usado em qualquer contexto desta área.

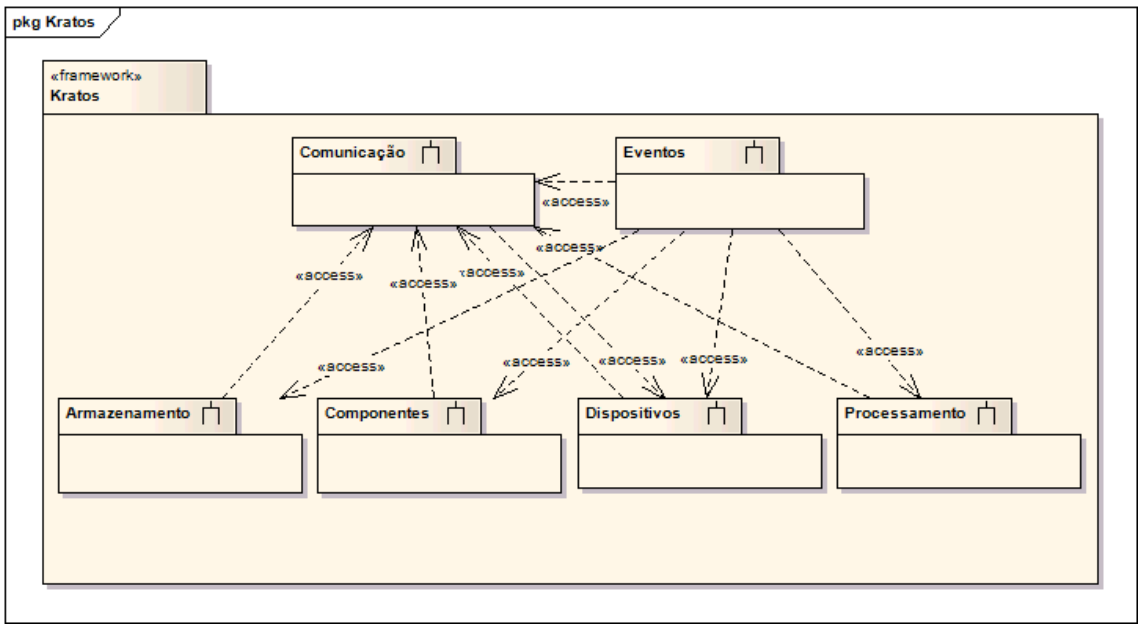

Figura 5.2: *Visão geral da arquitetura do middleware*

O diagrama de pacotes da Figura 5.2 ilustra os subsistemas do middleware e seus relacionamentos. A arquitetura foi desenhada de forma modular, de modo que cada subsistema possa executar o mais independentemente possível, gerando-se um modelo baseado em componentes sob a filosofia de reusabilidade. Há então a possibilidade de montar um sistema completamente centralizado ou completamente distribuído ou até mesmo um sistema misto; escolha que dependerá do tamanho da rede ou do *overhead* de comunicação tolerado. O middleware é composto de sete subsistemas principais em conformidade com o padrão IEEE 1451, introduzido no Capítulo 4, que se comunicam por meio do serviço de mensagens/comandos provido pelo subsistema `Comunicação`.

O diagrama da Figura 5.3 mostra uma visão geral dos componentes do middleware e suas principais interfaces. Verifica-se no diagrama a necessidade primordial do componente `GerenciarComunicacao`, que provê a interface `enviarMensagem`, responsável pela comunicação entre os componentes. Também é ilustrada a interação do componente usuário do middleware com o próprio middleware através dessa mesma interface `enviarMensagem`. Os serviços providos pelo componente usuário devem ser registrados, através da interface `obterServicos`, para que o middleware os publique e as outras aplicações tenham acesso a esses serviços. A listagem dos serviços providos pela RSSF pode ser obtida pela interface `listarServicos`. Estes componentes serão melhor explicados mais adiante.

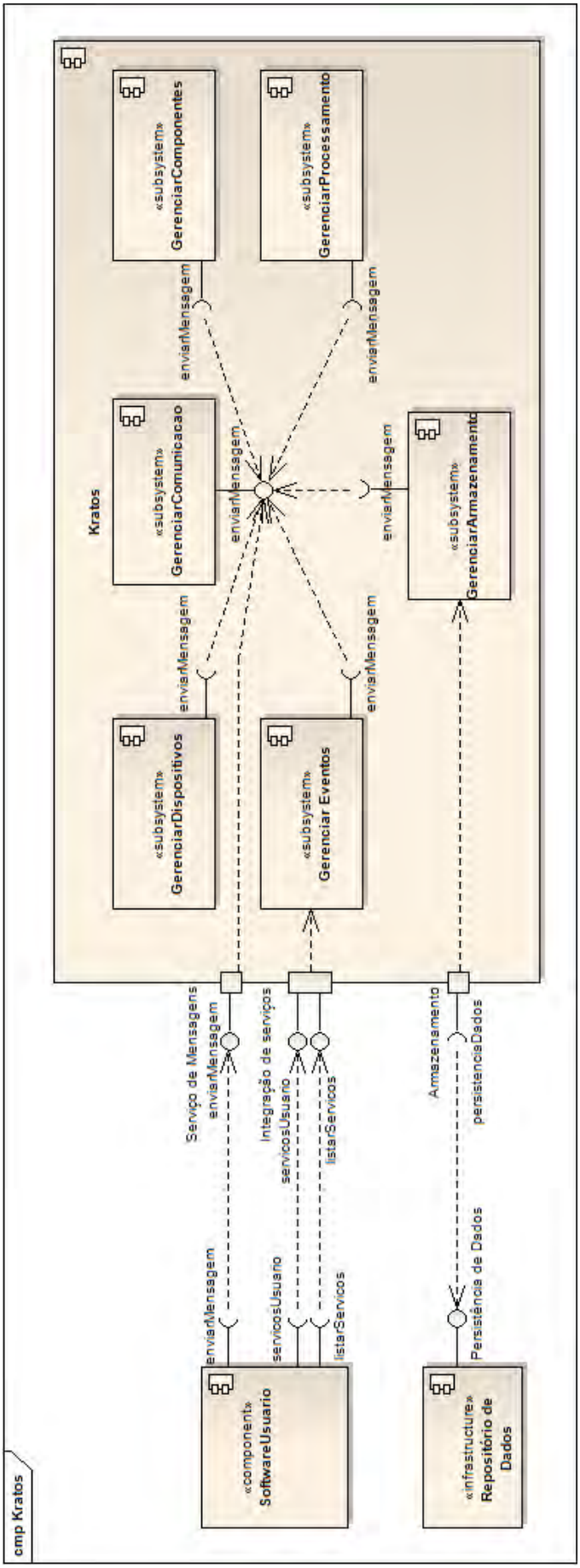

Figura 5.3: *Diagrama de componentes simplificado do middleware*

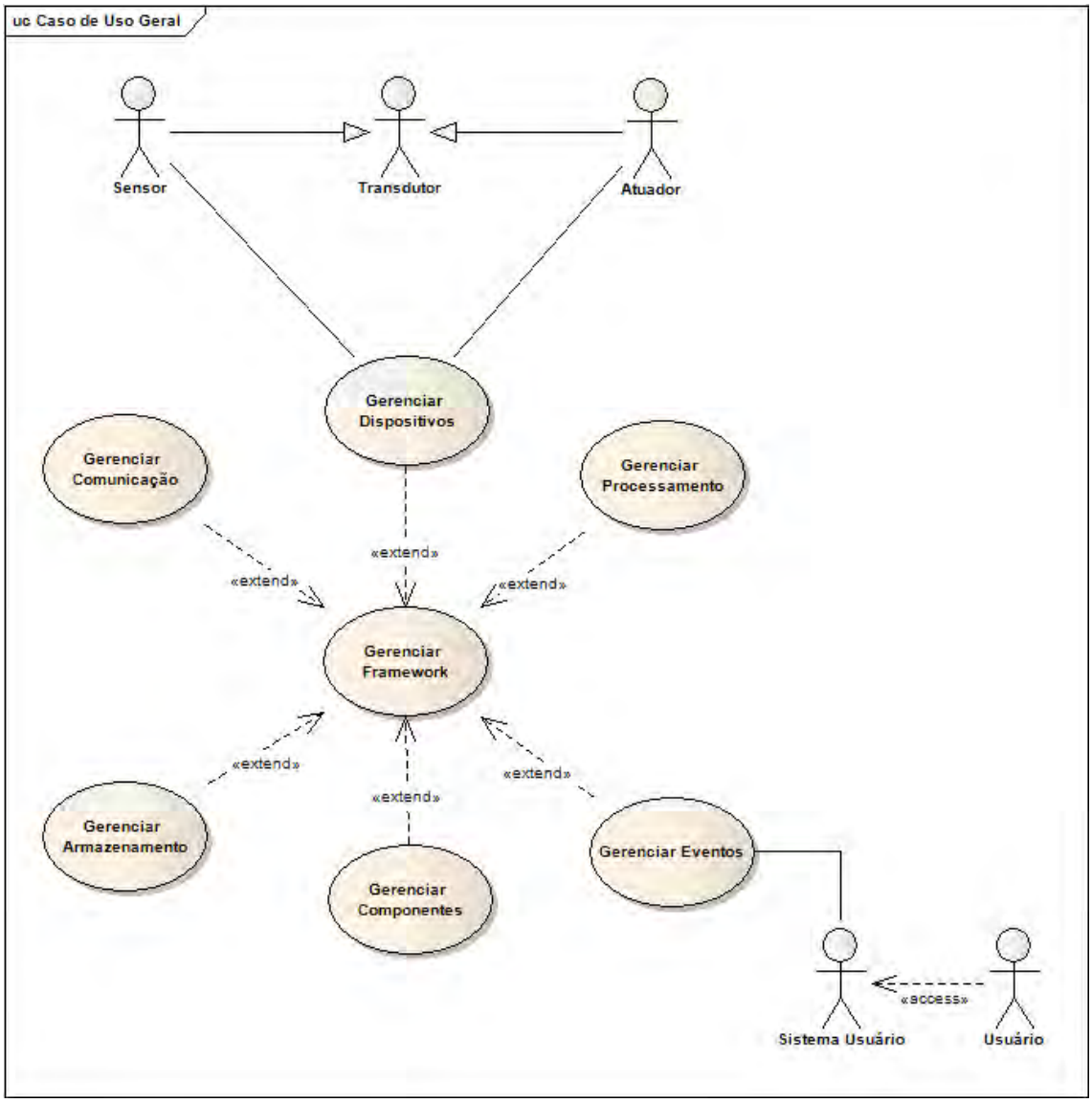

Figura 5.4: *Diagrama de casos de uso principal do middleware*

Como o middleware não interagirá diretamente com o usuário, mas sim as aplicações que rodam sobre ele, os três atores diretos são os sistemas usuários, os dispositivos sensores e atuadores e os atores indiretos são o objeto monitorado generalizado dos atores sensor e atuador e o usuário ligado ao sistema usuário que podem ser identificados no diagrama de casos de uso da Figura 5.4. Cada pacote e caso de uso será explicado nos itens seguintes.

5.6 Descrição dos subsistemas

5.6.1 Subsistema Comunicação

O middleware proposto é baseado no paradigma de publicação/subscrição de mensagens. Dada a natureza de seu modo de operação, é imprescindível a abordagem do subsistema `Comunicação` em primeiro lugar para melhor compreensão do funcionamento do middleware. A função deste subsistema é estabelecer um barramento de comunicação entre todos os componentes do middleware, unificando-os mesmo que não estejam fisicamente em um mesmo local.

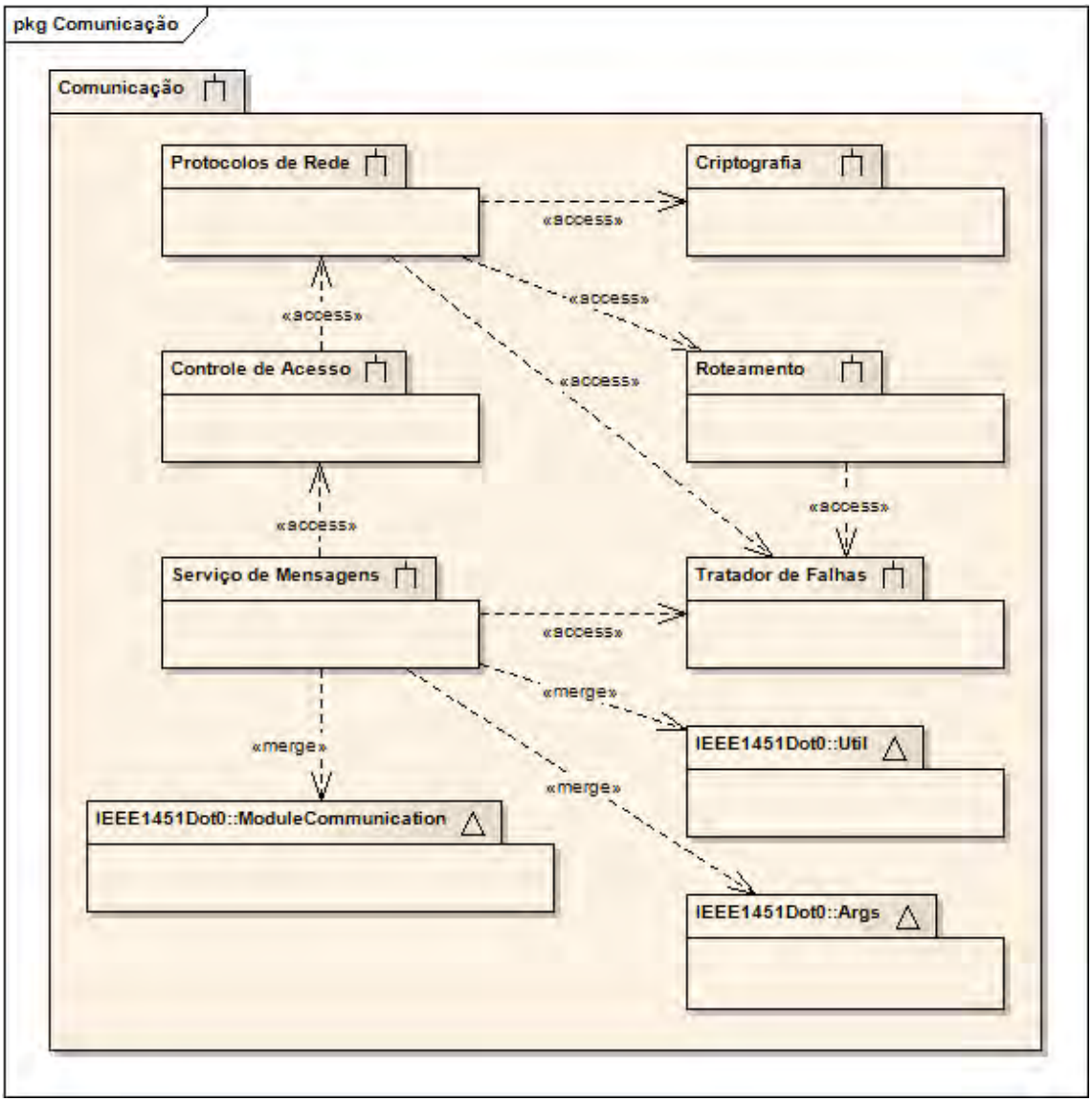

Figura 5.5: *Diagrama de pacotes do subsistema Comunicação*

A função básica deste subsistema é o envio de mensagens ou comandos, que é a unidade básica de comunicação entre os componentes do sistema. Essas mensagens são padronizadas pela IEEE 1451.0 como descrito na Seção 4.2.3, de modo a generalizar a

implementação do middleware, e caso necessário são encapsulados em outros protocolos ou traduzidas no subsistema `Dispositivos` de um NCAP para adequação ao protocolo próprio do TIM, garantindo a possibilidade de heterogeneidade dos dispositivos na rede. Este subsistema foi dividido em seis outros subsistemas, os quais serão descritos adiante.

No diagrama da Figura 5.6 é possível visualizar os casos de uso deste subsistema.

Na Figura 5.7 está o diagrama de componentes do subsistema `Comunicação`, cada subsistema será explicado adiante.

O subsistema `Controle de Acesso` tem como função elementar a verificação de origem/destino, não permitindo que mensagens sejam enviadas ou recebidas em componentes ou dispositivos aos quais não sejam destinadas ou necessárias, de acordo com as capacidades dos mesmos. Também há nele a possibilidade de criação de filtros de mensagens, na qual pode-se criar regras, como por exemplo de impedir que um certo componente ou dispositivo específico acesse à base de dados central ou use um determinado protocolo de rede.

Para garantir confidencialidade e integridade dos dados sensíveis transmitidos pela RSSF, protegendo de ataques e acesso descredenciado dados sensíveis transmitidos, além de proteger da inserção de dados manipulados por terceiros não autorizados, a criptografia é obrigatória e implementada através do subsistema `Criptografia`. Os protocolos aceitos estão descritos na Tabela 4.16. Desta forma, o dado será criptografado na origem e só poderá ser decifrado no componente destino requisitante ou em algum nodo agregador (*sink node*) [52].

O subsistema propriamente responsável pela transmissão de dados é o `Protocolos de Rede`; após a mensagem ser devidamente tratada, é repassada a este subsistema que faz a entrega da mensagem ao destinatário. Os `Protocolos de Rede` também são implementados modularmente, permitindo ao usuário a adição de novos protocolos como 802.11 (WiFi), 802.15.1 (Bluetooth), 802.15.4 (Zigbee), TCP/IP, etc.

Uma função importante para o middleware é a sua capacidade em definir os melhores caminhos ou rotas, fazendo o repasse de dados entre os nodos intermediários entre a origem e o destino da forma mais ótima possível. Usualmente, o meio mais apropriado de listar os vizinhos mais adequados para o repasse de dados para um determinado destino são as tabelas de roteamento e, por isto, sua construção e manutenção se faz de crucial importância para redes distribuídas de sensores [26]. No subsistema `Roteamento` será feita a fusão dos dados de localização determinados pelo subsistema `Dispositivos` que forem difundidos pelo subsistema `Componentes`. Devido à necessidade de ajuste ao cenário em que a rede será usada e aos serviços providos, se faz necessário o uso de diferentes protocolos de roteamento para melhor se adaptar aos seus recursos e melhor aproveitá-los, portanto, os protocolos de roteamento também podem ser implementados como componentes e agregados ao sistema.

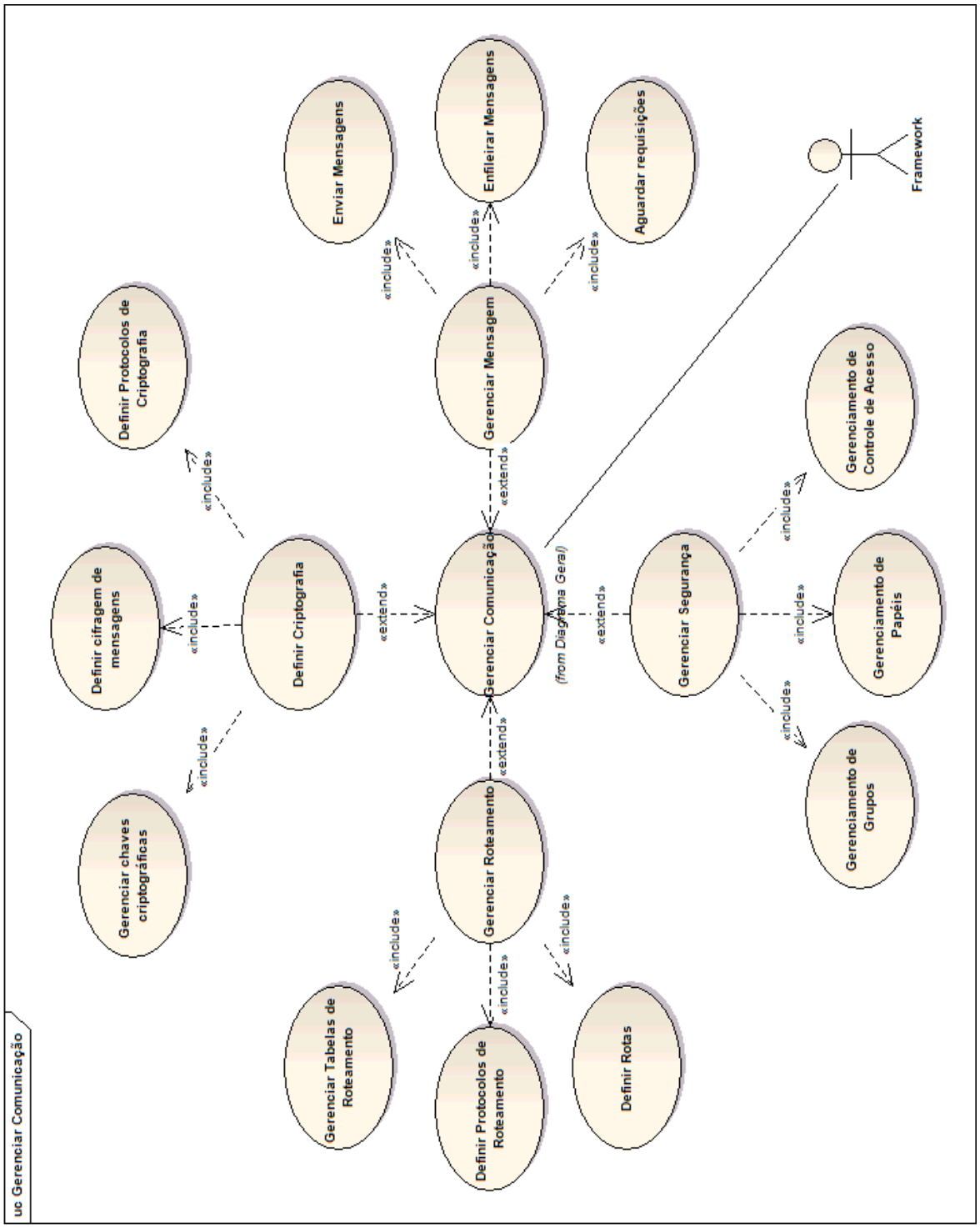

Figura 5.6: *Diagrama de casos de uso do subsistema Comunicação*

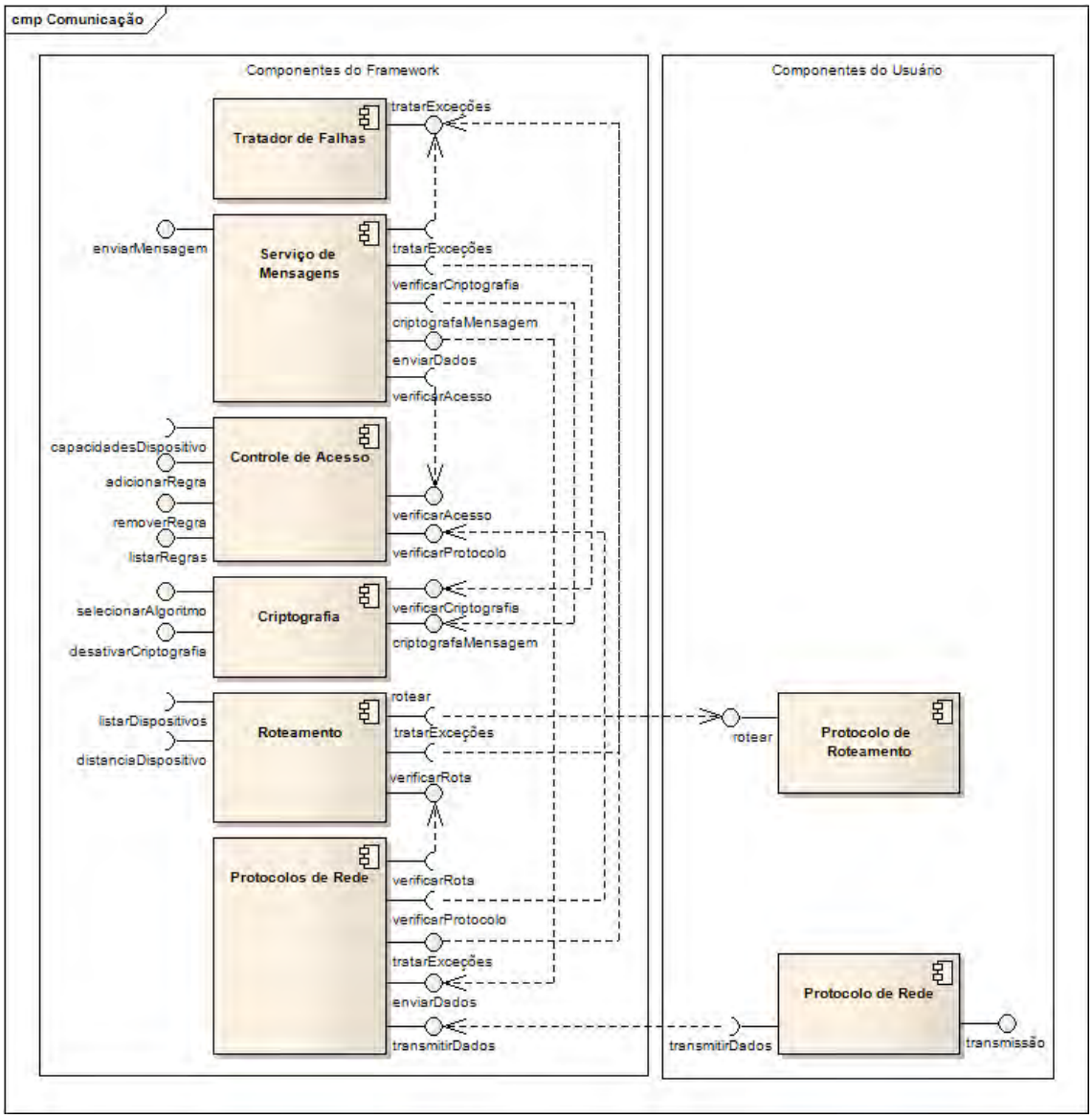

Figura 5.7: *Diagrama de componentes do subsistema Comunicação*

O componente integrador do subsistema `Comunicação` é o `Serviço de Mensagens`, é ele quem recebe as mensagens e as requisições, por subscrição às mesmas, e faz todos os trabalhos necessários para a postagem, como verificação de algoritmos criptográficos suportados pelo destino, verificação do controle de acesso do originador ao destino da mensagem e por fim requisição do envio do dado tratado. Este componente implementa os métodos do `IEEE1451Dot0::ModuleCommunication` (descrito em 4.2.12).

O `Tratador de Falhas` deste subsistema cuida de exceções como negação de acesso a componente/dispositivo e falhas de transmissão de dados, como intangibilidade do destino ao até mesmo sua negação em receber o dado, dentre outros tipos de falhas definidos pelo usuário.

O processo de envio de mensagens formalizado na Figura 5.8 é assíncrono para o emissor, portanto ele manda a requisição de envio e quando terminada a operação, o

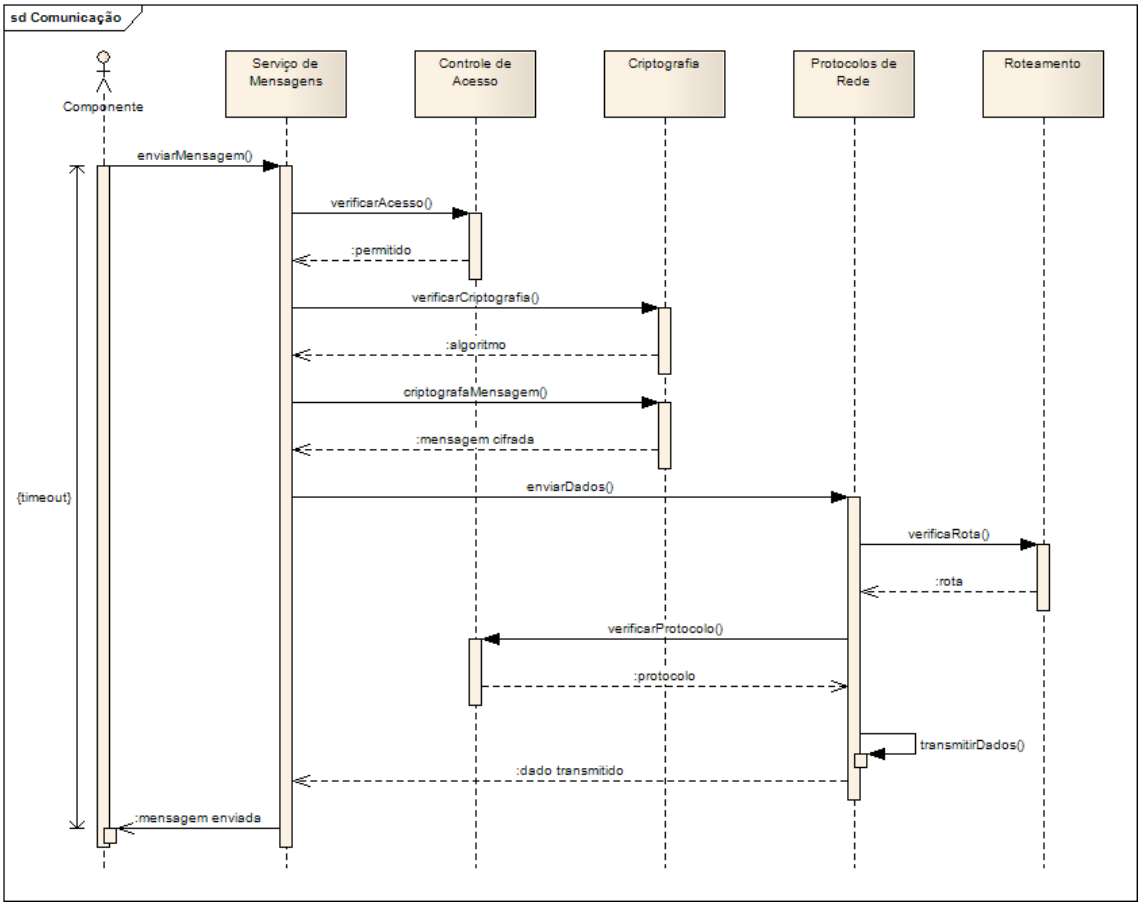

Figura 5.8: *Diagrama de sequência do processo de envio de mensagens*

subsistema pode enviar uma mensagem de confirmação de conclusão ao emissor.

Cada operação tem um tempo determinado para ser concluída (*timeout*), e caso não seja recebida uma mensagem de confirmação no final deste período, pode ser assumido que o envio de mensagem falhou e o processo pode então ser abortado. A origem deste tempo de espera é dependente da operação, podendo vir dos TEDS ou do próprio comando.

5.6.2 Subsistema Armazenamento

O subsistema `Armazenamento`, mostrado no diagrama da Figura 5.9, é a camada de abstração que tratará das interações com meios para persistência de dados. Nele estarão os componentes para persistência em EPROM, bancos de dados, arquivos, dentre outros possíveis modos de persistência de dados, além da geração de relatórios. Os outros subsistemas comunicarão sempre da mesma forma com este subsistema e este tratará o devido armazenamento e recuperação de dados transparentemente. Também é competência deste subsistema a replicação de dados e o tratamento e recuperação em caso de falhas de armazenamento, a fim de garantir-se a atomicidade e consistência dos dados sensoriados que devem ser ou que já estão armazenados; dado que falhas são eventos

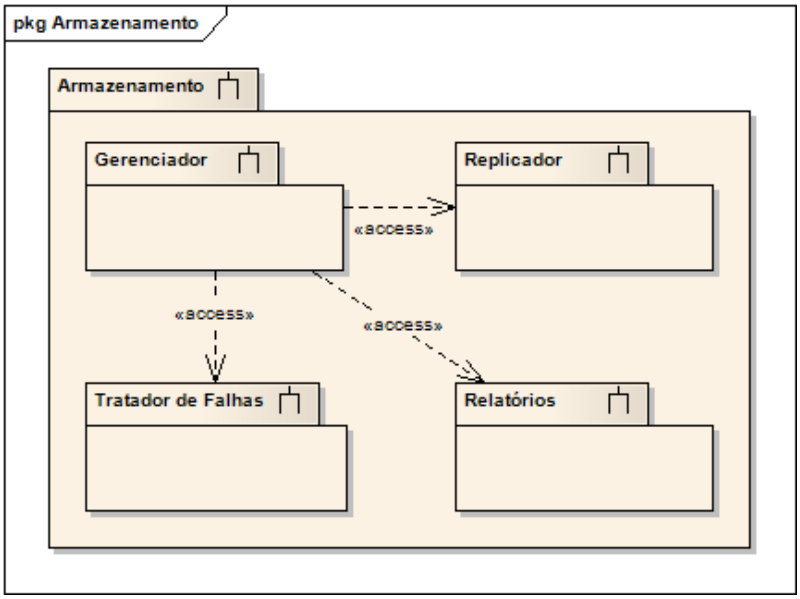

Figura 5.9: *Diagrama de pacotes do subsistema Armazenamento*

de natureza imprevisível e que não podem ser totalmente eliminados de sistemas, este componente é essencial na implementação deste subsistema.

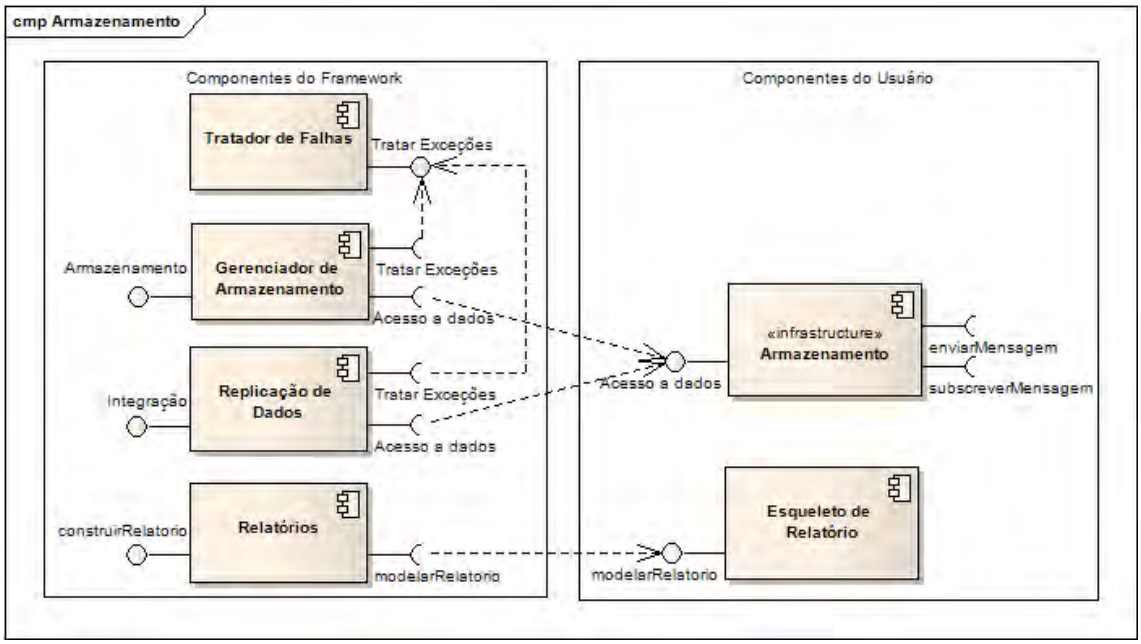

Figura 5.10: *Diagrama de componentes do subsistema Armazenamento*

Na Figura 5.10 são mostrados os relacionamentos entre os componentes internos de `Armazenamento`. Para uso em aplicações do middleware, são disponibilizadas duas interfaces: `Armazenamento` e `Integração`; a primeira ligada ao `Gerenciador de Armazenamento` é responsável pela recepção e envio direto de dados armazenados no componente de infraestrutura de armazenamento e a aplicação do usuário; a segunda é a interface de ligação entre um componente de armazenamento e outros de mesma função,

fazendo a sincronização de dados, garantindo a disponibilidade e integridade dos dados em todos os possíveis componentes de armazenamento de toda a estrutura de armazenamento do middleware. O `Tratador de falhas` é uma dependência dos outros componentes, dada a possibilidade e semelhança de falhas que podem ocorrer nestes componentes.

A partir dos dados em persistência é possível construir relatórios textuais, gráficos ou adaptados através de módulos de relatório desenvolvidos pelo usuário.

Na Figura 5.11 está apresentado o diagrama de casos de uso do subsistema `Armazenamento`.

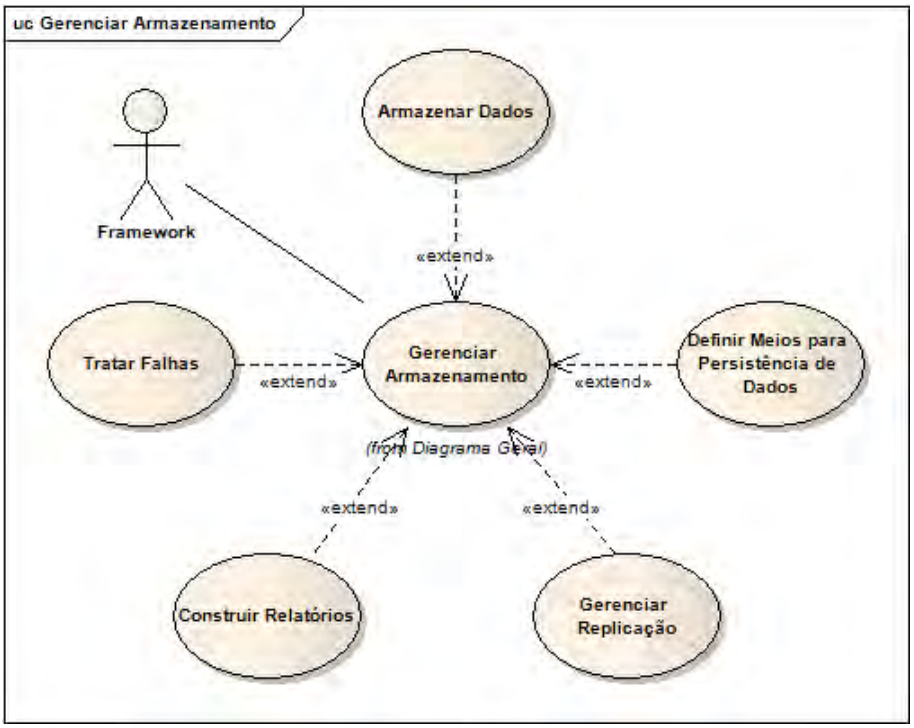

Figura 5.11: *Diagrama de casos de uso "Gerenciar Armazenamento"*

5.6.3 Subsistema Componentes

A abordagem do paradigma de componentes neste middleware é um meio para promover alguns requisitos não-funcionais, como (i) a reusabilidade, já que os componentes são especificados por meio de suas interfaces e não por meio de objetos individuais e detalhados; (ii) interoperabilidade, pelo mesmo motivo anterior, levando-se em conta que as interfaces são meios de acesso ao componente, independentes da implementação e; (iii) independência de software e hardware, para que o sistema seja facilmente adaptável a outros sistemas e dispositivos. A padronização dos componentes de usuário define interfaces que devem ser implementadas para que o componente funcione no middleware e esta implementação possa ser independente de linguagem

de programação. Por conseguinte, um componente programado em assembly poderá ser executado num microcontrolador que mantém comunicação com um outro componente programado em Java que é executado em um aparelho de telefone celular, por exemplo.

Os componentes do middleware são divididos em duas classes: os componentes organizacionais, que são definidos neste Capítulo e os componentes do usuário, que são aqueles criados para adicionar funcionalidades, adaptando o middleware às necessidades do novo ambiente de saúde, em particular. Generalizando esses componentes e também seus objetos dependentes, serão chamados a partir de agora de `elementos` da rede, sendo estes `elementos` nada mais que consumidores ou provedores de serviços na rede.

Definição 1 *Seja $V = v_1, v_2, ..., v_n$ uma RSSF com n nodos e seja $v_i = c_1, c_2, ..., c_m$ um nodo transdutor com m componentes de middleware. Um* `elemento` *do middleware é um componente c_j provedor ou consumidor de serviços na rede.*

A gerência de componentes instanciados é parte sensível do sistema, suscetível a trazer inconsistência se instalado de modo distribuído, isto é, caso hajam várias cópias em execução em uma mesma RSSF; a menos que se tenha um ambiente completamente segmentado, com vários domínios de execução paralela. Se o controle de objetos instanciados for inconsistente, as aplicações poderão fazer requisições a componentes que não existem em um determinado contexto, mas que existem em outro contexto, causando falhas. No entanto, este subsistema é replicado na RSSF de forma a garantir disponibilidade, sendo que as réplicas (secundários) servem apenas para consulta, não sofrendo alterações a menos que disparadas pelo subsistema `Componentes` principal.

A função principal do subsistema `Componentes` é instanciar os `elementos` do middleware no sistema e invocar suas rotinas de criação e destruição em tempo de execução, agregando as funções deste novo componente ao conjunto de funções do middleware. Outra função importante é a remoção imprevista de `elementos` extintos do sistema, como por exemplo por perca de conectividade ou falta de energização, sem causar falha ou degrado das características da rede, obviamente quando em presença de `elementos` redundantes. Portanto, este é um subsistema primordial na estrutura do middleware.

Cada componente e dispositivo deve estar registrado neste subsistema a fim de poder ser acessado por outros componentes, já que todos os gerenciadores de mensagens fazem acesso a esse subsistema para montagem de sua tabela de destinos. Os `elementos` da rede recebem um identificador único de segundo nível (descrito em 4.2.2) após a sua alocação e são organizados de forma hierárquica, de modo que no primeiro nível estão os componentes organizacionais e do segundo nível em diante os componentes do usuário e dispositivos.

Na estrutura descritora de `elemento`, além do identificador, também existem os campos `estado`, `componente agregado imediato` e `pai`, que são descritos a seguir [4, 59]:

- O campo «`estado`» contém um código para o estado corrente do componente; dentre estes estados estão: interruptível, ininterruptível e parado. O estado «`interruptível`» é aquele no qual o `elemento` se habilita a receber mensagens, enquanto que o «`ininterruptível`» não; este último estado pode ser utilizado para os `elementos` que deixaram de ser tangíveis imprevistamente ou para `elementos` que receberam tarefas que exigem exclusividade. O estado «`parado`» define os `elementos` que foram finalizados, mas que continuam alocados. Ao mudar de estado, este subsistema difunde a informação a todos os componentes de comunicação.
- O campo «`componente agregado imediato`» refere-se, caso seja um dispositivo, ao driver pelo qual são controlados e, em outros casos, ao componente do middleware mais próximo hierarquicamente. A intenção desse campo é criar agrupamentos de `elementos` correlacionados para facilitar a instrução de eventos.
- O campo «`pai`» pode muitas vezes possuir o mesmo valor do campo «`componente agregado imediato`», e refere-se ao `elemento` agregado imediato. Para os dispositivos pode indicar o nodo concentrador de dados (sink node) e é de suma importância para a comunicação com dispositivos passivos (explicados na Seção 5.6.4), dada a sua incapacidade de comunicação direta com o middleware.

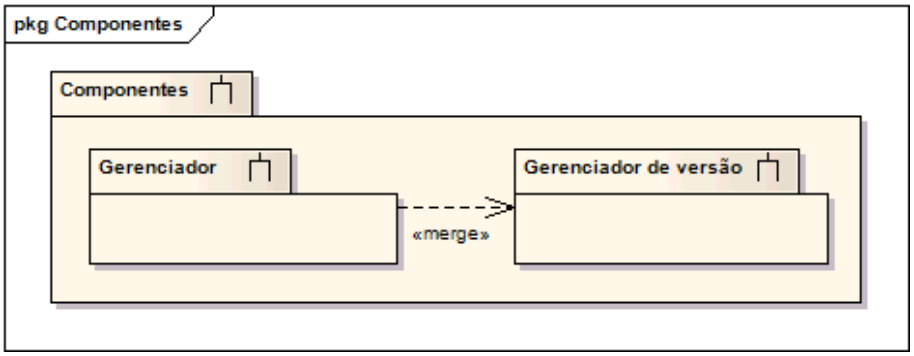

Figura 5.12: *Diagrama de pacotes do subsistema Componentes*

Na Figura 5.12 está ilustrado o diagrama de pacotes do subsistema Componentes.

Como ilustrado no diagrama da Figura 5.13, este subsistema possui seis interfaces para o middleware:

1. `carregarComponente`: responsável por buscar o conteúdo do componente em mídia de armazenamento e alocá-lo em local de destino (seja no mesmo dispositivo ou em outros), sendo também responsável por resolver dependências entre os componentes, executando assim qualquer atividade preliminar de ativação;

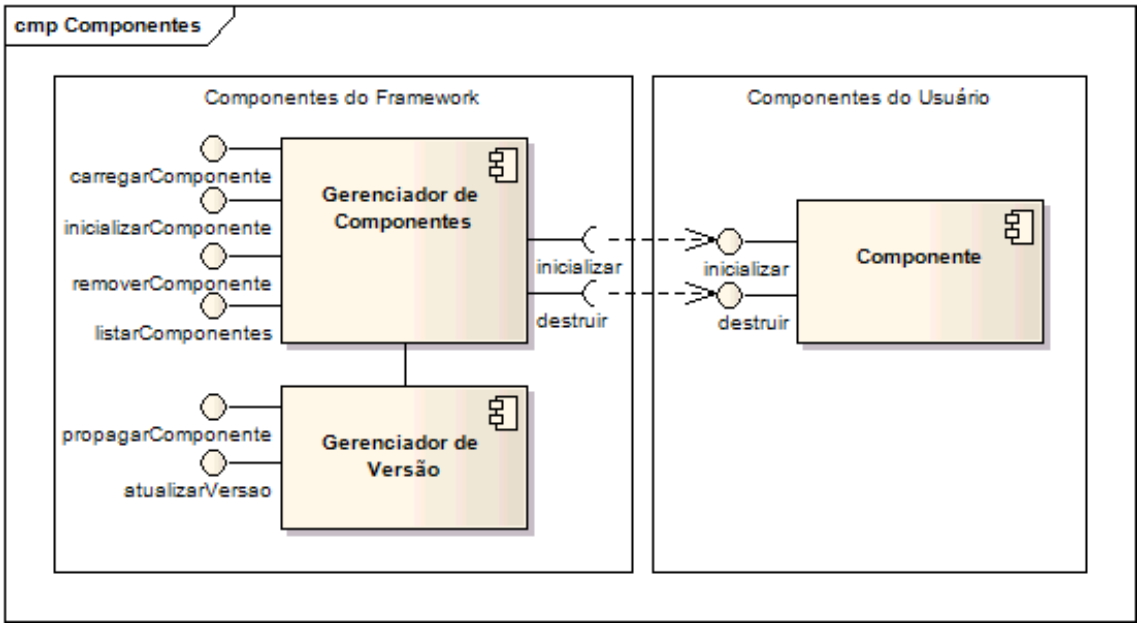

Figura 5.13: *Diagrama de componentes do subsistema Componentes*

2. `inicializarComponente`: invoca a interface inicializar do componente carregado e o põe na tabela de componentes ativos;
3. `removerComponente`: remove um componente do sistema simplesmente parando-o ou além de parar também o desaloca; estas ações são precedidas de verificação de uso dos componentes em questão, podendo ser enviadas mensagens aos componentes usuários, avisando-lhes o inicio da remoção e no final do processo, uma mensagem em difusão para todos os componentes gerenciadores de mensagens para que atualizem suas próprias tabelas de dispositivos; se existir algum componente dependente do que está sendo removido a transação não será efetuada;
4. `listarComponentes`: retorna a lista dos componentes ativos e inativos (inicializados ou não) do sistema, junto com informações notórias a seu respeito, como identificador de `elemento`, tipo de serviço provido, nível hierárquico, número de dependentes de serviços, versão de software, etc;
5. `propagarComponente`: distribui os componentes pela rede de forma manual ou executa distribuição automática, na qual a atualização é disponibilizada mas o componente tem autonomia de decidir o momento para atualização, não sendo uma opção disponível para todos os tipos de componentes;
6. `atualizarVersao`: força a atualização de versão de componentes.

Na Figura 5.14 está apresentado o diagrama de casos de uso do subsistema `Componentes`.

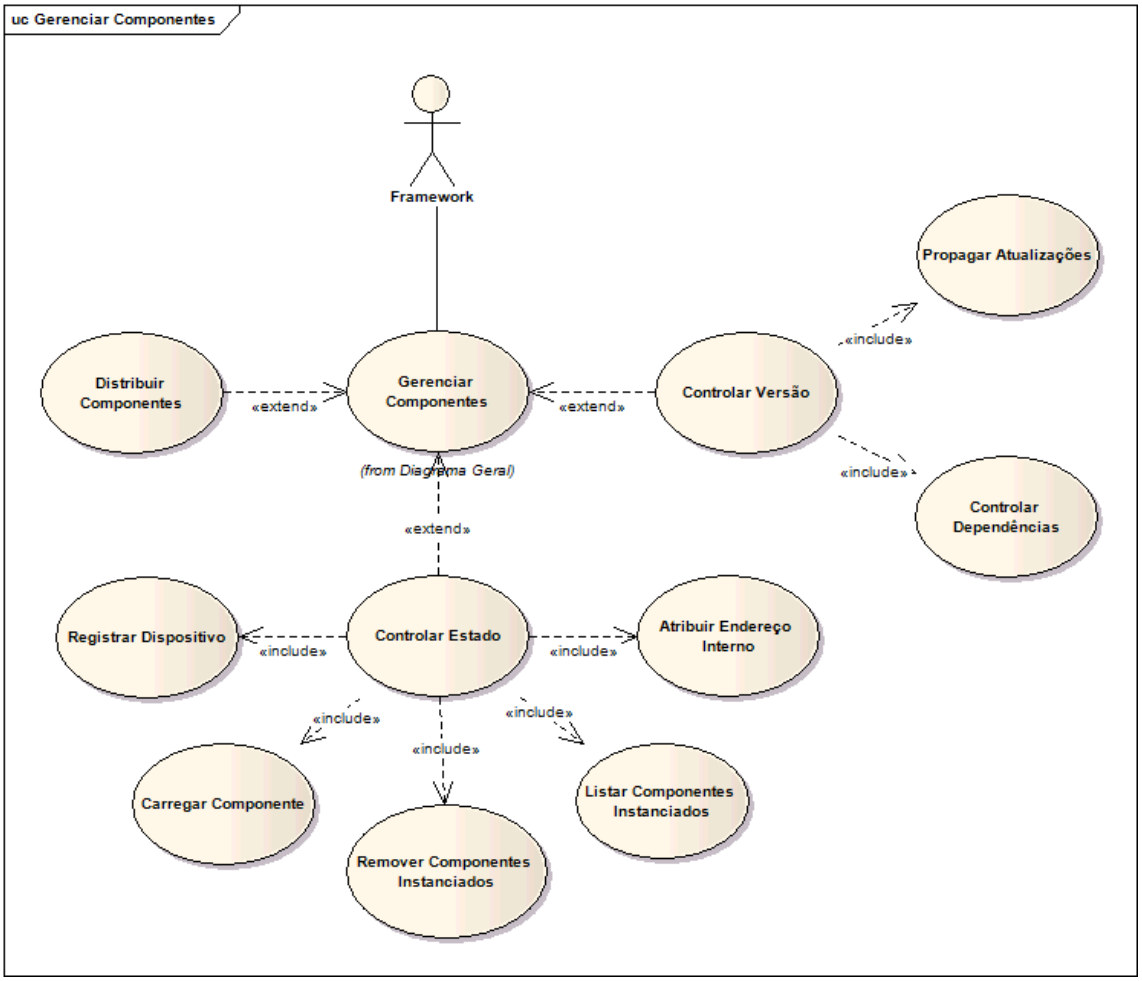

Figura 5.14: *Diagrama de casos de uso "Gerenciar Componentes"*

5.6.4 Subsistema Dispositivos

Os dispositivos em si, caracterizam uma rede de sensores por determinar as características funcionais, do ponto de vista do cliente, que ela pode executar. No modelo do middleware **Kratos** são considerados dois tipos de dispositivos, os ativos que são também componentes do sistema (abstraído na IEEE 1451.0 como NCAP), com todos os requisitos básicos para interagir com todo esse sistema, e os passivos que apenas recebem comandos (i.q. TIM).

O subsistema `Dispositivos`, mostrado no diagrama da Figura 5.15, é a camada de interação com os dispositivos agregados ao middleware. No `Gerenciador de Dispositivos` estarão implementados os componentes descritores de funcionamento de sensores e atuadores ("*drivers*"), ou seja, para cada modelo de dispositivo haverá um componente descritor para manipular-lhe e este componente poderá controlar vários dispositivos simultaneamente, não sendo necessário haver um descritor para cada dispositivo anexado ao sistema. Esta funcionalidade de descritores de dispositivo é importante por causa da abertura do padrão IEEE 1451 para funcionalidades definidas pelo fabri-

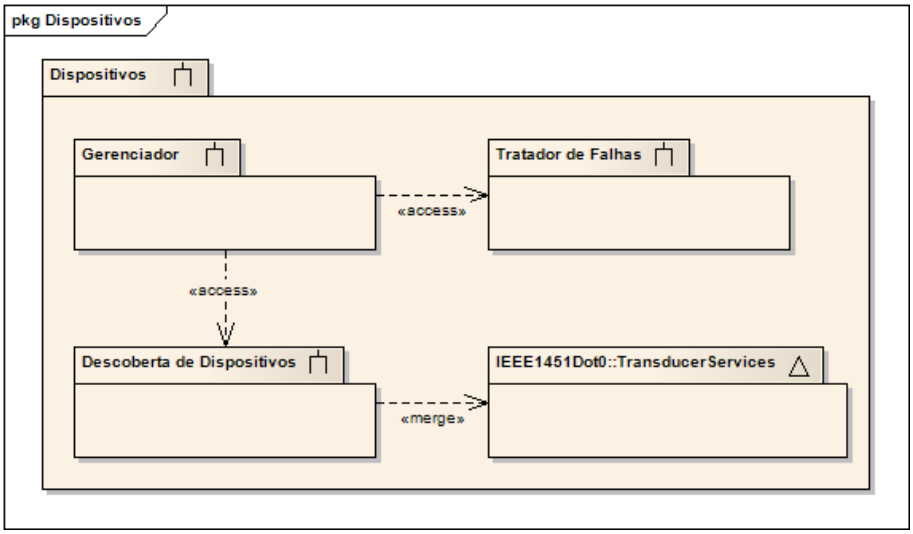

Figura 5.15: *Diagrama de pacotes do subsistema Dispositivos*

cante, mas isto também pode significar uma adaptabilidade a dispositivos não conformes àquele padrão, configurando a construção de adaptadores ("*wrappers*"). Estes descritores sempre residirão nos NCAP. O `Tratador de Falhas` deste subsistema trata exceções de acesso a dispositivos como, por exemplo, a ausência ou perda de carga de dispositivos, por exemplo. O pacote `Descoberta de Dispositivos` implementa os métodos do `IEEE1451Dot0::TransducerServices` (descrito em 4.2.12), portanto sua função principal é o interfaceamento entre as aplicações e as funções do middleware. Dentre as funções do subsistema `Descoberta de Dispositivos` estão a busca e rastreamento de transdutores e atualização do mapeamento da RSSF.

Antes de poder se comunicar com toda a rede de sensores, o dispositivo requisita um identificador (endereço) ao pacote `Descoberta de Dispositivos` que o aloca e o anuncia na RSSF após comunicar a existência de um novo dispositivo ao componente `Gerenciador de Componentes` da RSSF.

Conforme disposto no diagrama da Figura 5.16, o `Gerenciador de Dispositivos` possui sete interfaces para uso no middleware:

1. `adicionarModulo`: executa a adição de um descritor de dispositivo (*driver*) novo e com ajuda do subsistema componentes efetua seu registro na rede, esse descritor será um componente que manterá os dispositivos de mesmo modo de acionamento;
2. `removerModulo`: finaliza todas as transações correntes, pára todos os dispositivos submissos, enviando-lhes mensagens de término e por fim desaloca seus registros;
3. `configurarModulo`: envia parâmetros *on-the-fly* para componentes descritores, como por exemplo adição de novo dispositivo controlado;
4. `listarModulos`: lista os componentes descritores e sua possível hierarquia, trazendo também a listagem de dispositivos submissos;

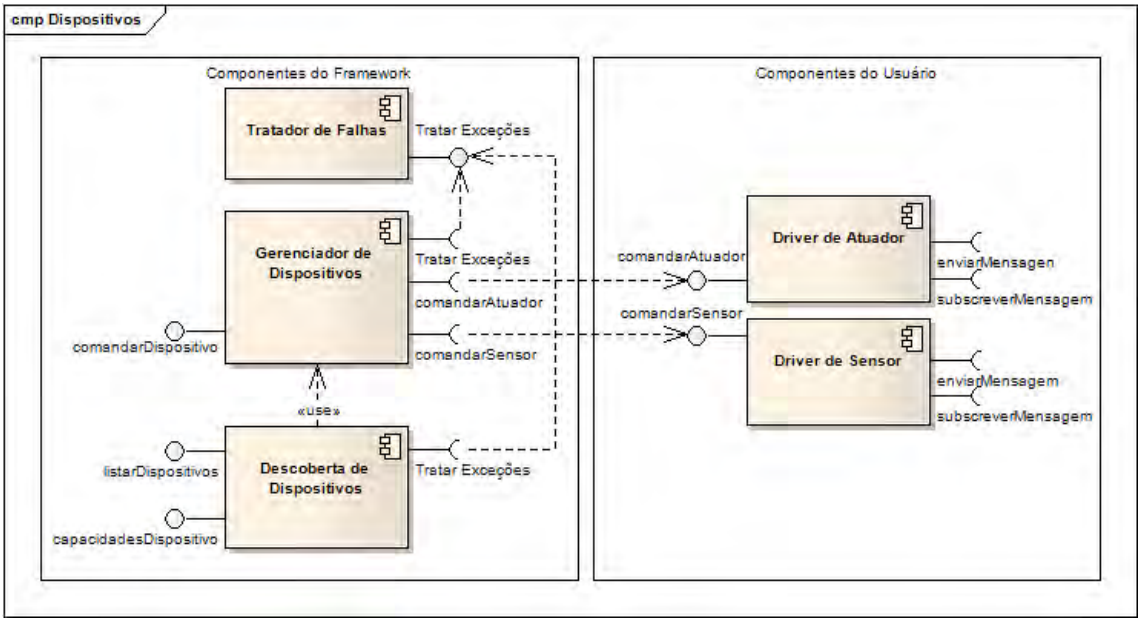

Figura 5.16: *Diagrama de componentes do subsistema Dispositivos*

5. `comandarDispositivo`: envia comando ao descritor de dispositivo para acionar o dispositivo;
6. `listarDispositivos`: compila uma lista de atributos de dispositivos;
7. `capacidadesDispositivos`: retorna as capacidades de sensoriamento ou atuação dos dispositivos.

Na Figura 5.17 está apresentado o diagrama de casos de uso do subsistema `Dispositivos`. Na Figura 5.18 está representada a sequência padrão de descoberta de dispositivos transdutores e requisição de dados.

5.6.5 Subsistema Processamento

O subsistema `Processamento` subsistema é responsável pelo processamento dos dados que são coletados através dos sensores de uma determinada rede. O diagrama da Figura 5.19 mostra os pacotes deste subsistema.

Dependendo das necessidades da aplicação, os dados de cada sensor podem ser enviados para nodos concentradores (*sink nodes*), de forma que chegue até o receptor uma quantidade reduzida de valores, economizando o uso de recursos, já que o custo de transmissão de dados é consideravelmente maior que qualquer computação complexa, o que produz eficiência energética e assim um tempo de sobrevida maior à rede. Os dados podem também ser enviados diretamente do sensor ao receptor que irá disponibilizá-los ao usuário. Este último procedimento é mais adequado para redes menores que visam monitorar regiões pequenas ou que o grau de importância dos dados de um sensor em

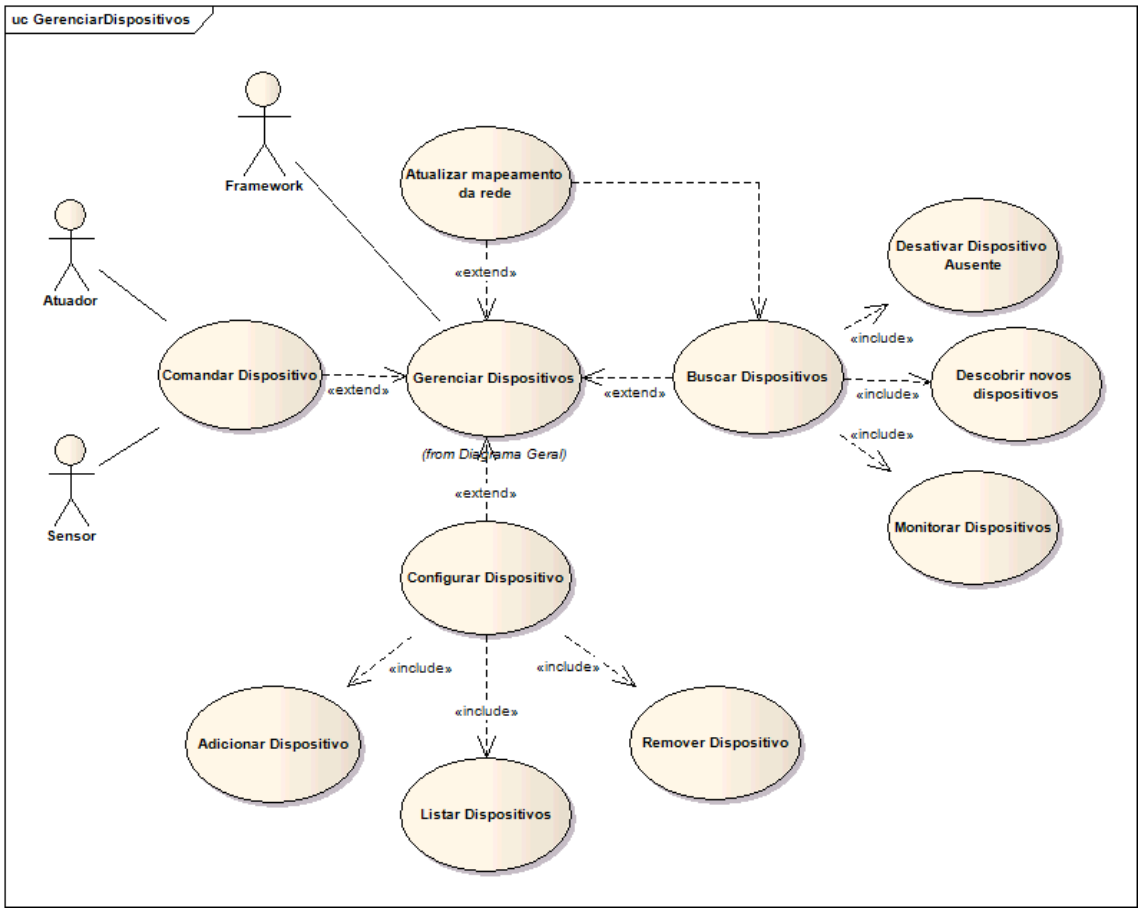

Figura 5.17: *Diagrama de casos de uso "Gerenciar Dispositivos"*

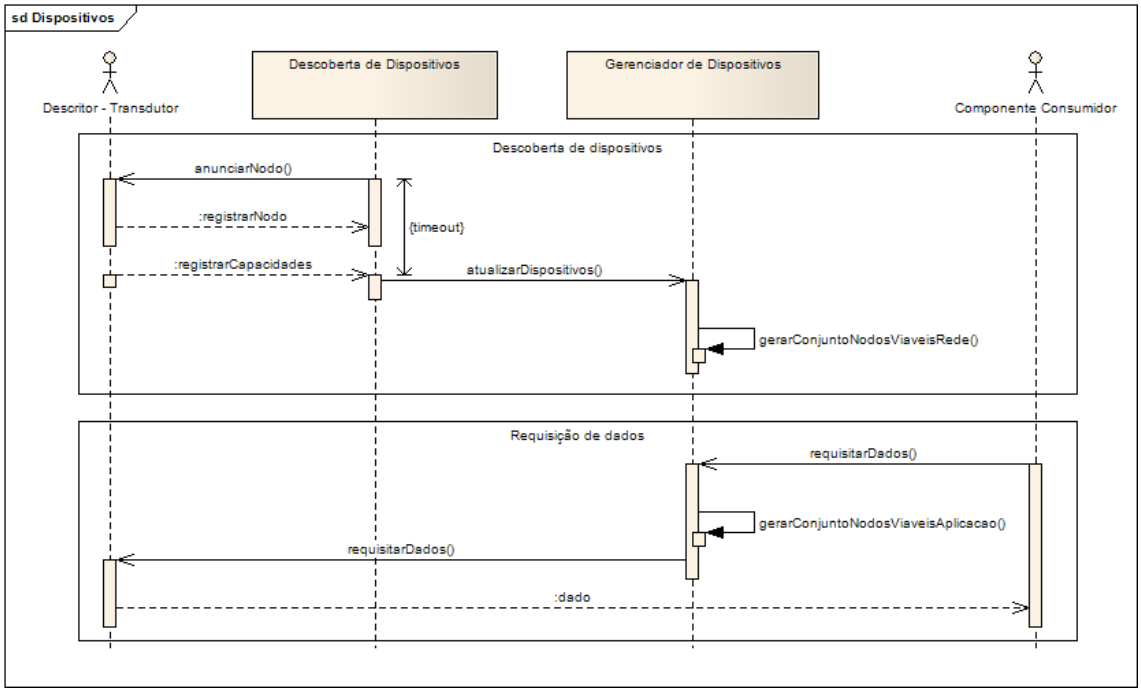

Figura 5.18: *Diagrama de sequência do processo de descoberta de dispositivos e requisição de dados.*

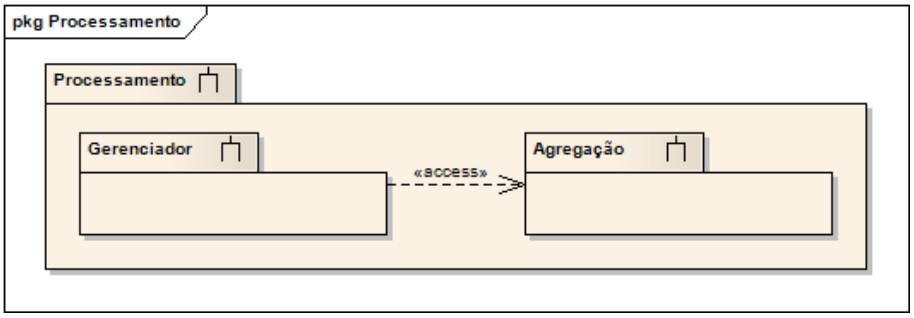

Figura 5.19: *Diagrama de pacotes do subsistema Processamento*

particular seja relevante. Como pode ser visto na Figura 5.20, essas características também podem ser definidas pelo usuário, adequando o middleware à aplicação [26].

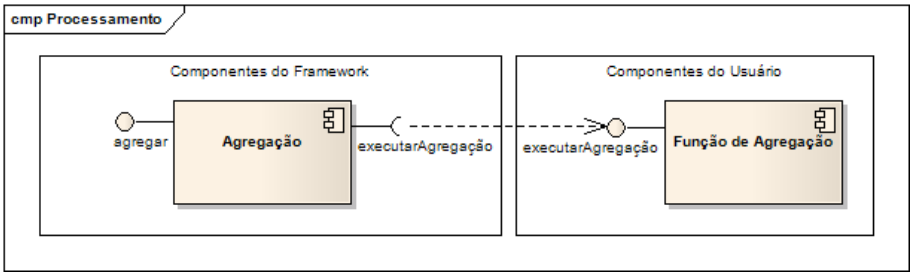

Figura 5.20: *Diagrama de componentes do subsistema Processamento*

O tratamento desses dados pode ser feito através de agregação de dados. No caso de fusão, os dados colhidos nos sensores são fundidos uns aos outros, de forma a chegar ao dispositivo requisitante todos os dados na íntegra. Já com agregação, os dados são passados a um nodo centralizador, que processa esses dados de forma a ter somente um dado a repassar para o próximo nodo centralizador ou para o dispositivo requisitante. Por exemplo, no monitoramento de temperatura do corpo humano, se três sensores estão colocados no braço de um paciente, com a fusão, três valores chegarão até a aplicação, enquanto que com agregação, pode ser feita uma média dos três sensores, sendo que o valor que chega até a aplicação é somente um. A Figura 5.21 ilustra o processamento de dados por agregação: à esquerda, as mensagens transmitidas são condensadas (fundidas) por nodos sorvedouros (*sink nodes*) e repassadas até o solicitante, e à direita, ao aplicar uma função de agregação, como por exemplo a média de valores, reduz-se o número de mensagens transmitidas.

Portanto, o processamento dos dados coletados numa rede de sensores depende diretamente da aplicação. Neste contexto, é um grande desafio determinar quando agregar resultados, levando-se em conta o risco de imprecisão ou perda de quantidade substancial de dados. A Figura 5.22, a seguir, mostra o diagrama de casos de uso do subsistema `Processamento`.

Os subsistemas implementados do padrão IEEE 1451.0 foram descritos na seção 4.2 e estão detalhados em [17].

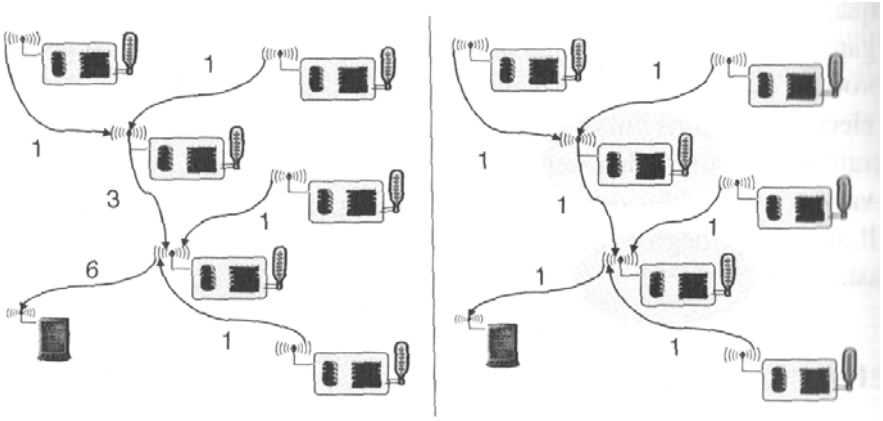

Figura 5.21: *Exemplos de processamento de dados [26]*

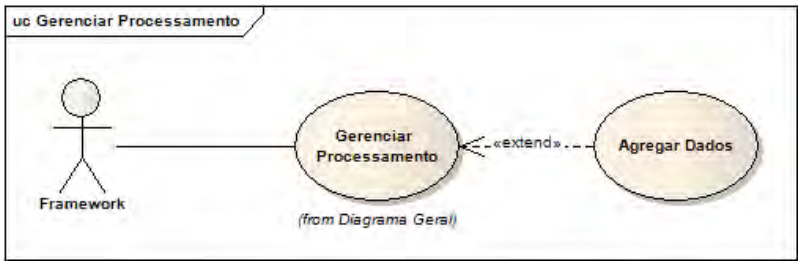

Figura 5.22: *Casos de uso do subsistema Processamento*

5.6.6 Subsistema Eventos

De modo a concretizar o modelo de programação escolhido para uso no middleware, este subsistema cuidará das questões relacionadas ao controle de eventos e o controle das atividades decorrentes destes eventos. Este subsistema é o meio de interação entre o software do usuário e o middleware, aqui ocorrem todos os processos ligados diretamente às ações executadas ou requisitadas pelo usuário. O subsistema `Eventos`, mostrado no diagrama da Figura 5.23, é dividido em três subsistemas, que serão explicados adiante.

No subsistema `Gerenciador de Eventos` é feito o controle dos sete tipos de amostragem definidos na IEEE 1451.0, descritos na seção 4.2.7. As requisições podem ser feitas tanto direcionadamente a um `elemento` (via identificador), quanto em modo *data-centric*, no qual requisitam-se dados que casem uma determinada expressão, por exemplo, pode-se pedir a temperatura torácica de uma pessoa monitorada, em vez de pedir a média das temperaturas dos sensores s_x, s_y e s_z.

Formas para generalização de coleta de informações não são fáceis de se criar, então foi definido o uso de expressões ternárias na forma «`atributo, valor, operador`», onde "`atributo`" corresponde à capacidade de sensoriamento desejada (e.g. temperatura), "`valor`" a um valor concreto como "`30°C`" ou um curinga (TODOS ou QUALQUER) e "`operador`" a um operador relacional (e.g. =, <, >=). Este formalismo se mostra bastante conveniente para os requisitos de acurácia ou suporte a medições per-

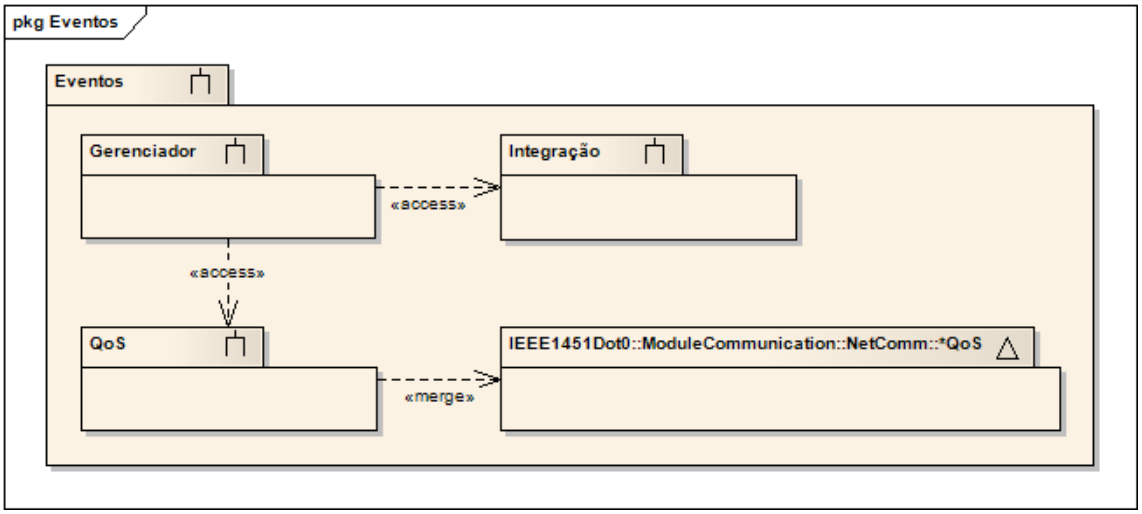

Figura 5.23: *Diagrama de pacotes do subsistema Eventos*

iódicas.

O subsistema `Integração` cuida do gerenciamento de sessão com outros sistemas, para tanto, registra a aplicação usuária no middleware, pois só depois deste registro que ela poderá usar o sistema. Neste registro estarão as informações sobre o contexto da aplicação, como o seu identificador único no sistema, eventos e atributos de QoS próprios.

O subsistema `QoS` foi desenvolvido a partir do módulo `IEEE1451Dot0::ModuleCommunication::NetComm`, com a adição de parâmetros mais específicos para área de saúde e cuida das questões de escalonamento e melhora da qualidade de serviço da rede de sensores. Ele cuida de definição de prioridades em baixo nível, que são aquelas definidas estaticamente e em alto nível que analisam alguns fatores dinâmicos como [26]:

- **Detecção de eventos/probabilidade de ocorrência** (e.g. qual seria a probabilidade de um evento ocorrido não ser detectado ou o interessado no evento não ser reportado do mesmo?).
- **Erro de classificação de evento**: quando um evento não deve ser apenas detectado mas também classificado, estes erros devem ser pequenos.
- **Demora de detecção de eventos**: quanto tempo se passa desde a detecção de um evento e o seu anúncio.
- **Eventos não reportados**: para aplicações que requeiram boletins periódicos, a probabilidade de não entrega de relatórios deve ser pequena.
- **Precisão em aproximações**: nas leituras com aproximação, qual pode ser a taxa erros média ou máxima absoluta ou relativa dessas aproximações. Do mesmo modo, para as aplicações de detecção de borda, qual deve ser a precisão destas descrições de borda.
- **Precisão de acompanhamento**: aplicações não podem perder um objeto sensoriado, seu posicionamento deve ser o mais próximo possível do real e os erros devem

ser mínimos. Leva-se em conta também os gaps de sensoriamento, dentre outros aspectos.

É também através deste subsistema que o middleware agrupa os sensores de uma RSSF em função de seu posicionamento ou por outras características que os tornem comuns, como tipo de sensores, por exemplo. Isso acontece quando uma sessão é iniciada para facilitar que os objetivos da aplicação sejam alcançados. Esse tipo de programação é denominada holística.

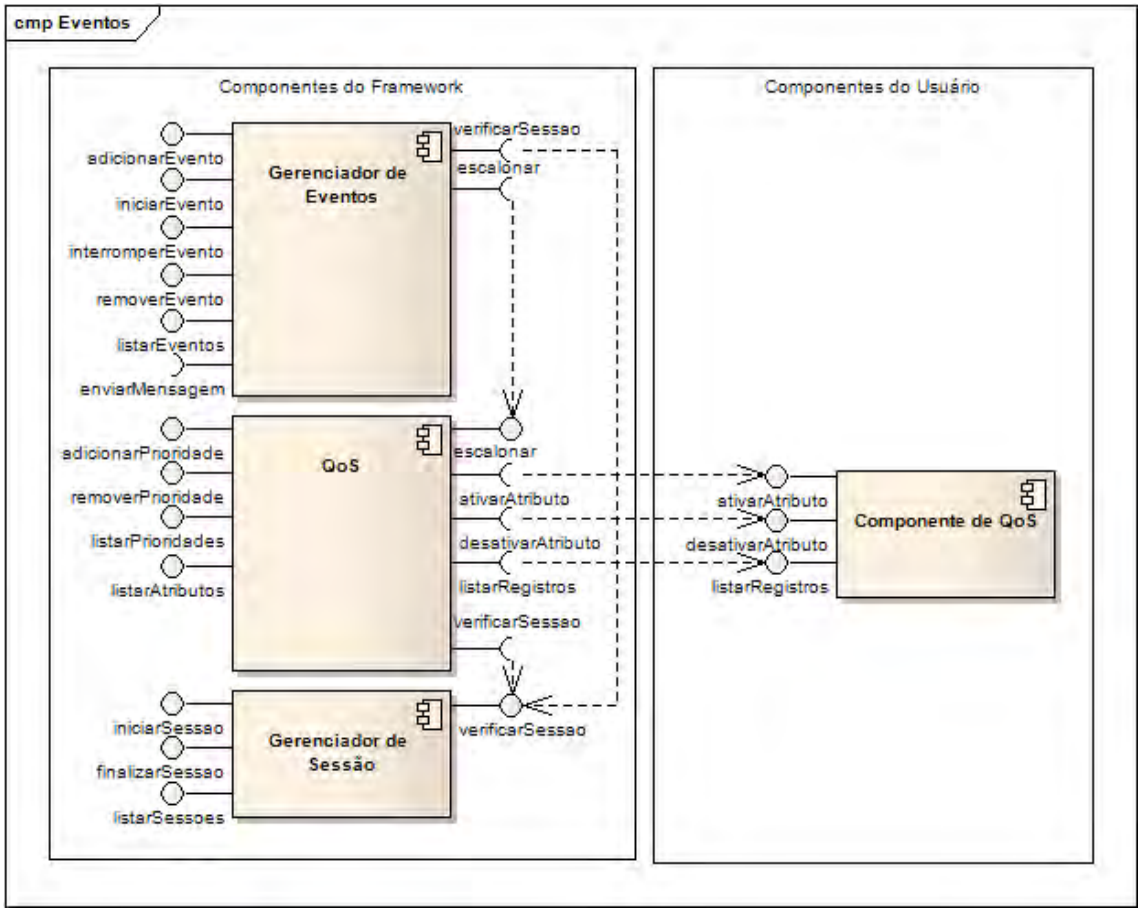

Figura 5.24: *Diagrama de componentes do subsistema Eventos*

Como ilustrado na Figura 5.24, o subsistema `Eventos` possui treze interfaces para o usuário:

- `adicionarEvento`: faz o trabalho de inclusão de requisições de leitura na rede de sensores, que podem ser imediatos, recorrentes ou agendados;
- `iniciarEvento`: inicia a execução de um evento pré-adicionado;
- `interromperEvento`: paralisa um evento corrente;
- `removerEvento`: exclui um evento da lista de execução
- `listarEventos`: lista os eventos registrados, tanto os que estão em execução, quanto os interrompidos ou esperando finalização de exclusão;

- `adicionarPrioridade`: adiciona, na categoria das prioridades estáticas, uma regra de priorização, como por exemplo, atribuir prioridade elevada à leitura de sensores de eletrocardiograma ou reduzir a prioridade dum sensor "s" específico;
- `removerPrioridade`: exclui uma regra de prioridade;
- `listarPrioridades`: mostra a listagem de prioridades atribuídas;
- `listarAtributos`: lista os atributos dinâmicos e correntes de QoS, adicionados pela inserção de componentes de QoS opcionais;
- `iniciarSessao`: inicia uma sessão para que o programa usuário possa começar a se comunicar na rede de sensores;
- `finalizarSessao`: termina uma sessão;
- `listarSessoes`: lista sessões ativas da aplicação;
- `requisicao`: processa uma requisição feita ao middleware por uma aplicação.

O componente de QoS tem três interfaces:

1. `ativarAtributo`: ativa um atributo determinado de QoS para um dado serviço de rede;
2. `desativarAtributo`: remove uma regra de ativação de atributo de QoS;
3. `listarRegistros`: lista os atributos ativos de QoS para o dado componente.

Na Figura 5.25 apresenta-se o diagrama de casos de uso do subsistema `Eventos`. Na Figura 5.26 está formalizada a sequência de eventos relacionados ao registro de aplicação no middleware, requisição de leituras disparadas por eventos, periódicas e unitárias.

5.7 Considerações finais

A arquitetura modular do **Kratos**, baseada em componentes, favorece uma boa manutenibilidade, portabilidade e possibilidade de utilização em plataformas reduzidas, já que não se faz necessário que todos os componentes do middleware, com todas as suas funcionalidades, estejam implementados em um `elemento` para o funcionamento da RSSF. Estes requisitos são alcançados através do subsistema `Componentes`, bastando respeitar as dependências de cada componente e funcionalidade necessários para as aplicações em execução.

O subsistema `Comunicação` implementa a maioria dos módulos da IEEE 1451 considerados neste trabalho e é um dos principais componentes do middleware, por proporcionar a integração entre os demais componentes e `elementos` do middleware, além de ser fundamental para o paradigma de publicação/subscrição. Este subsistema é o principal responsável pela segurança, por estabelecer um canal seguro entre os `elementos`

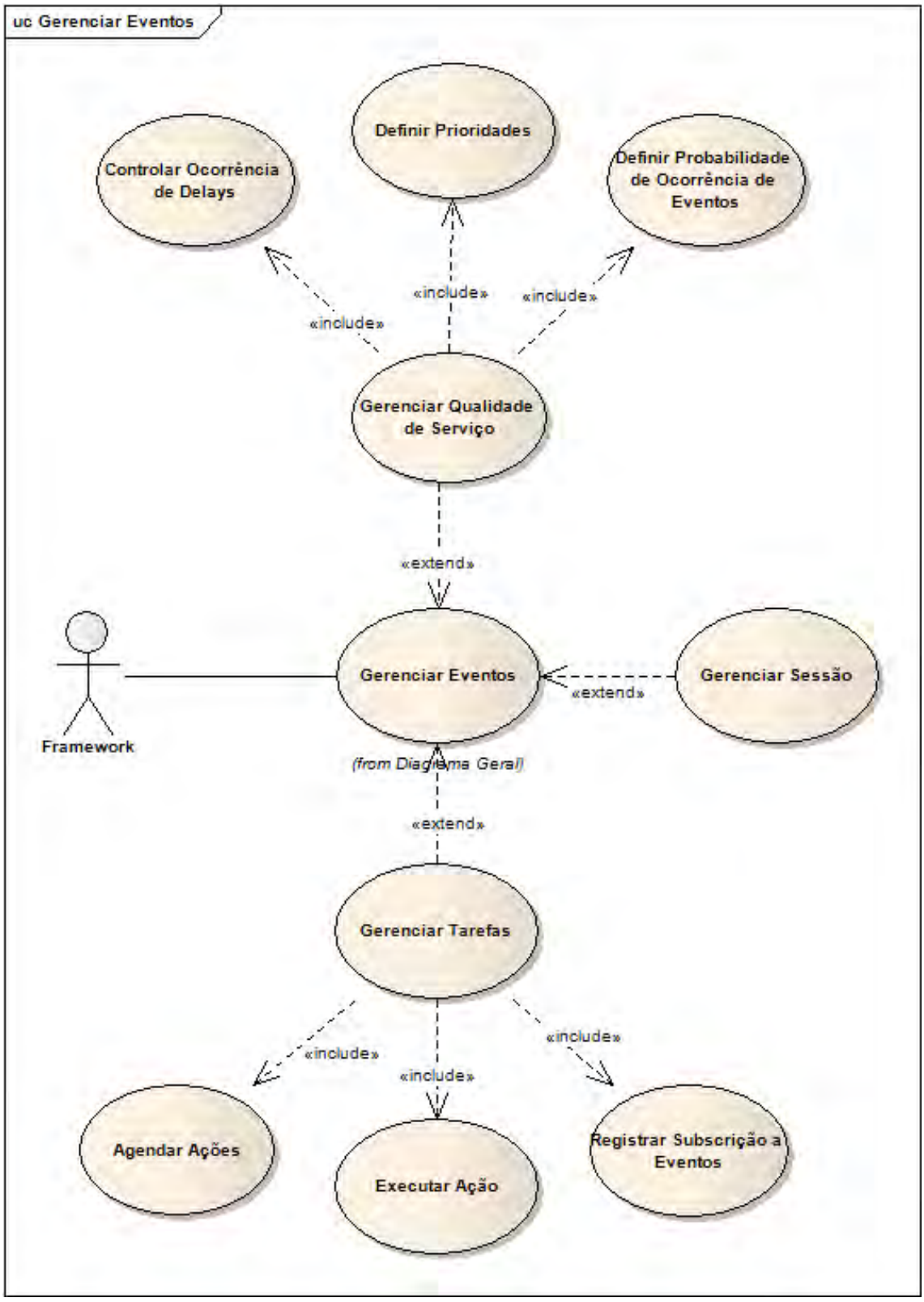

Figura 5.25: *Diagrama de casos de uso "Gerenciar Eventos"*

da RSSF, e também pela interoperabilidade, por ser o principal componente a implementar a IEEE 1451.

Os principais requisitos tratados pelo subsistema `Armazenamento` são a per-

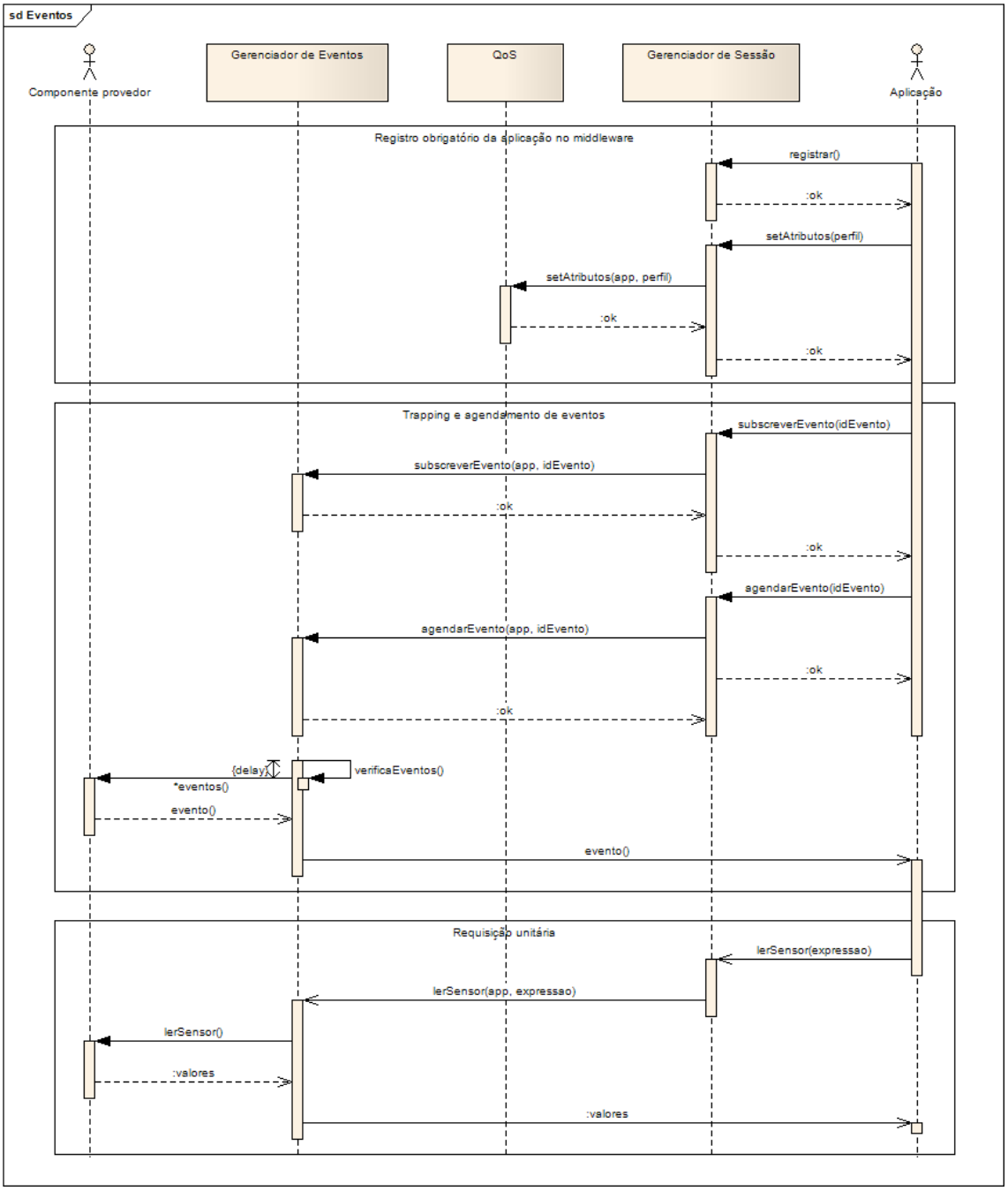

Figura 5.26: *Diagrama de sequência do processo de registro de aplicação e requisição de dados coletados.*

sistência e a recuperabilidade e segurança dos dados existentes na RSSF.

O subsistema `Dispositivos` realiza a integração entre o middleware e os dispositivos transdutores, além de garantir a acurácia e confiabilidade dos dados coletados por conhecer os parâmetros de calibragem de cada dispositivo.

O principal requisito alcançado pelo subsistema `Processamento` é uma análise semântica dos dados coletados para redução de transmissões desnecessárias, resultando na economia de recursos.

O subsistema `Eventos`, como camada entre a aplicação do usuário e o middleware, realiza a interoperabilidade e interação entre aplicações para diferentes finalidades em saúde e o middleware.

As seguintes limitações foram identificadas no middleware **Kratos**:

- O middleware **Kratos** é voltado para a área de saúde e é caixa branca. Portanto, as aplicações do usuário serão desenvolvidas como módulos específicos. Isto se dá por causa dos objetivos do middleware, podendo assim englobar uma quantidade maior de cenários na área de saúde;
- Como a comunicação será feita de forma assíncrona, usuários podem fazer solicitações mais rápido que o tempo de processamento dos transdutores, podendo sobrecarregar a RSSF.

CAPÍTULO 6

Conclusões e trabalhos futuros

Este trabalho tem como objetivo fornecer um middleware que é um componente de software para facilitar e aliviar os problemas do desenvolvimento de aplicações voltadas para o monitoramento da saúde de pacientes.

A fim de facilitar o uso de aplicações na área de saúde, o **Kratos** apresentou algumas características que foram introduzidas e mostradas no estudo de caso do sinal de ECG, para simplificar e garantir a transmissão das informações. Outro ponto importante foi proporcionar o uso confiável das caraterísticas propostas pelo middleware permitindo portabilidade e execução concorrente das aplicações por meio de interfaces padronizadas com uso de aplicações que fazem uso de dispositivos móveis, possibilitando assim o seu uso em RSSF aplicadas a saúde de forma transparente.

O processo de desenvolvimento proposto faz a adaptação de transdutores (nós sensores ou atuadores) já desenvolvidos menos complexo, pois elimina modificações desnecessárias, mantendo o foco na conformidade com os padrões da família IEEE 1451. Resultados preliminares destacam a possibilidade de interligar transdutores heterogêneos através de uma RSSF sem que haja impactos no funcionamento da mesma, sendo possível a construção de descritores de funcionamento de TIMs não padronizados à IEEE 1451 dentro dos NCAPs, possibilitando a sua integração ao **Kratos**. Espera-se então um aumento no grau de autonomia dessa aplicação, porque a rede poderá se adaptar automaticamente a mudanças de requisitos sem que haja necessidade de intervenção no sistema, graças à capacidade de autoconfiguração de cada transdutor. O interfaceamento de transdutores heterogêneos é possível graças à abstração da camada de rede utilizando os modelos distribuídos e modulares propostos e demonstrados do modelo do **Kratos**.

Com os resultados obtidos através da modelagem baseada nos padrões IEEE 1451, pode-se perceber a facilitação do entendimento e modelagem dos componentes, implicando também na facilidade de implementação de forma clara e consistente. Este middleware apresenta um diferencial em relação aos propostos na literatura encontrada até o desenvolvimento desta dissertação que é o desenvolvimento fortemente padronizado de seus componentes. Outro diferencial desta proposta é que ela está voltada para a área de saúde como um todo.

Para trabalhos futuros podemos focar na implementação e prova completa da proposta, assim como a implantação em um ambiente hospitalar, por exemplo, onde possa se aferir a ajustar melhor os componentes do middleware.

Referências Bibliográficas

[1] AKYILDIZ, I. F.; SU, W.; SANKARASUBRAMANIAM, Y.; CAYIRCI, E. **Wireless sensor networks: A survey**. *Computer Networks*, 38(4):393–422, 2002.

[2] ALVES, S. V. L.; ALVES, E. C.; BARROS, R. S. M. **XMOM - um middleware orientado a mensagens**. *Annals of I Jornada Científica da UNIBRATEC, Recife*, p. 515–527, 2006.

[3] BARR, R.; BICKET, J. C.; DANTAS, D. S.; DU, B.; KIM, T. W. D.; ZHOU, B.; SIRER, E. G. **On the need for system-level support for ad hoc and sensor networks**. *SIGOPS Oper. Syst. Rev.*, 36:1–5, April 2002.

[4] BECK, M.; BÖHME, H.; DZIADZKA, M.; KUNITZ, U.; MAGNUS, R.; VERWORNER, D. **Linux Kernel Internals**. Addison Wesley, 2 edition, 1998.

[5] BLUETOOTH SPECIAL INTEREST GROUP. **Specification of the Bluetooth system**. Bluetooth Special Interest Group (SIG), 2005.

[6] BONATO, P. **Wearable sensors and systems**. *Engineering in Medicine and Biology Magazine, IEEE*, 29(3):25 –36, may-june 2010.

[7] BUONADONNA, P.; HILL, J.; CULLER, D. **Active message communication for tiny networked sensors**, 2001.

[8] CALLAWAY, E.; GORDAY, P.; HESTER, L.; GUTIERREZ, J.; NAEVE, M.; HEILE, B.; BAHL, V. **Home networking with IEEE 802.15. 4: a developing standard for low-rate wireless personal area networks**. *Communications Magazine, IEEE*, 40(8):70–77, 2002.

[9] CARVALHO, H. S.; MURPHY, A. L.; HEINZELMAN, W. B.; JR., C. J. N. C. **Network-based distributed systems middleware.** In: *Middleware Workshops'03*, p. 13–20, 2003.

[10] CCSR. **Mobile communication / wireless sensors and actuator networks**. http://www.ee.surrey.ac.uk/CCSR/mobile/research/wsn [Online; acessado em 19-outubro-2009].

[11] ESTRIN, D.; GOVINDAN, R.; HEIDEMANN, J.; KUMAR, S. **Next century challenges: Scalable coordination in sensor networks**. p. 263–270, Seattle, Washington, EUA, 1999. ACM.

[12] HADIM, S.; MOHAMED, N. **Middleware challenges and approaches for wireless sensor networks**. *IEEE Distributed Systems Online*, 7(3), 2006.

[13] HAMBLEY, A. R.; MORUZZI, R. L.; FELDMAN, C. L. **The use of intrisic components in an ecg filter**. *IEEE Transactions on Biomedical Engineering*, BME-21(6):469–473, 11 1974.

[14] HEINZELMAN, W. B.; MURPHY, A. L.; CARVALHO, H. S.; PERILLO, M. A. **Middleware to support sensor network applications**. *IEEE Network*, 18(1):6–14, 2004.

[15] HENRICKSEN, K.; ROBINSON, R. **A survey of middleware for sensor networks: state-of-the-art and future directions**. In: *Proceedings of the international workshop on Middleware for sensor networks*, p. 60–65. ACM, National ICT Australia, 2006.

[16] HUGHES, D. **Middleware for wsn**, Aug. 2008. http://www.comp.lancs.ac.uk/computing/research/mpg/reflections/papers/WSNMiddleware.ppt [Online; acessado em 31-agosto-2010].

[17] IEEE STD 1451.0-2007. **IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats**. Institute of Electrical and Electronics Engineers, New York, 2007.

[18] IEEE STD 1451.1-1999. **IEEE Standard for a Smart Transducer Interface for Sensors and Actuators-Network Capable Application Processor (NCAP) Information Model**. Institute of Electrical and Electronics Engineers, New York, 2000.

[19] IEEE STD 1451.2-1997. **IEEE Standard for a Smart Transducer Interface for Sensors and Actuators-Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats**. IEEE Instrumentation and Measurement Society, New York, 1997.

[20] IEEE STD 1451.3-2003. **IEEE Standard for a Smart Transducer Interface for Sensors and Actuators-Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems**. Institute of Electrical and Electronics Engineers, New York, 2004.

[21] IEEE STD 1451.4-2004. **IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats**. Institute of Electrical and Electronics Engineers, New York, 2004.

[22] IEEE STD 1451.5-2007. **IEEE Standard for a Smart Transducer Interface for Sensors and Actuators Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats**. Institute of Electrical and Electronics Engineers, New York, 5 2007.

[23] IEEE STD 1451.7-2010. **IEEE Standard for Smart Transducer Interface for Sensors and Actuators–Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats**. Institute of Electrical and Electronics Engineers, New York, 2010.

[24] ISHIBASHI, K.; MORISHIMA, N.; KANBARA, M.; SUNAHARA, H.; IMANISHI, M. **Toward ubiquitous communication platform for emergency medical care.** *IEICE Transactions*, 92-B(4):1077–1085, 2009.

[25] KAL, S. **Basic Electronics Devices Circuits And It Fundamentals**. Prentice-Hall Of India Pvt. Ltd., 2004.

[26] KARL, H.; WILIG, A. **Protocols and Architectures for Wireless Sensor Networks**. John Wiley & Sons Ltd, England, 2007.

[27] KIM, M.; LEE, J.; LEE, Y.; RYOU, J. **Cosmos: A middleware for integrated data processing over heterogeneous sensor networks**. *ETRI journal*, 30(5):696–706, 2008.

[28] KIM, Y. B.; KIM, M.; LEE, Y. J. **Cosmos: a middleware platform for sensor networks and a u-healthcare service**. In: *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, p. 512–513, New York, NY, USA, 2008. ACM.

[29] KNAUP, P.; DICKHAUS, H. **Perspectives of medical informatics: Advancing health care requires interdisciplinarity and interoperability**. *Methods Inf Med*, 48:1 – 3, 2009.

[30] LEE, K. **IEEE 1451: A standard in support of smart transducer networking**. In: *Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE*, volume 2, p. 525–528. IEEE, 2000.

[31] LEVIS, P.; MADDEN, S.; POLASTRE, J.; SZEWCZYK, R.; WHITEHOUSE, K.; WOO, A.; GAY, D.; HILL, J.; WELSH, M.; BREWER, E.; OTHERS. **Tinyos: An operating system for sensor networks**. *Ambient intelligence*, 35, 2005.

[32] LEVIS, P. **Maté manual**, Aug. 2008. http://www.cs.berkeley.edu/~pal/mate-web/index.html [Online; acessado em 31-agosto-2010].

[33] LEVIS, P.; CULLER, D. **Maté: a tiny virtual machine for sensor networks**. *SIGPLAN Not.*, 37:85–95, October 2002.

[34] LIU, T.; MARTONOSI, M. **Impala: a middleware system for managing autonomic, parallel sensor systems**. *SIGPLAN Not.*, 38:107–118, June 2003.

[35] LOUREIRO, A. A. F.; NOGUEIRA, J. M. S.; RUIZ, L. B.; DE FREITAS MINI, R. A.; NAKAMURA, E. F.; FIGUEIREDO, C. M. S. **Redes de sensores sem fio**. *Annals of XXI Simpósio Brasileiro de Redes de Computadores (SBRC2003), Natal*, p. 179–226, May 2003.

[36] MADDEN, S. R. **Tinydb overview**, Aug. 2008. http://telegraph.cs.berkely.edu/tinydb/overview.html [Online; acessado em 31-setembro-2010].

[37] MADDEN, S. R.; FRANKLIN, M. J.; HELLERSTEIN, J. M.; HONG, W. **Tinydb: an acquisitional query processing system for sensor networks**. *ACM Trans. Database Syst.*, 30:122–173, March 2005.

[38] MAHER, D.; HARRIES, A.; ZACHARIAH, R.; ENARSON, D. **A global framework for action to improve the primary care response to chronic non-communicable diseases: a solution to a neglected problem**. *BMC Public Health*, 9(1):355, 2009.

[39] MAHMOUD, Q. H. **Wireless application programming with j2me and bluetooth**, 2 2003. http://developers.sun.com/mobility/midp/articles/bluetooth1/ [Online; acessado em 31-agosto-2010].

[40] MILENKOVIĆ, A.; OTTO, C.; JOVANOV, E. **Wireless sensor networks for personal health monitoring: Issues and an implementation**. *Computer Communications*, 2006.

[41] MURPHY, A. L.; HEINZELMAN, W. B. **Milan: Middleware linking applications and networks**. Technical report, University of Rochester, Rochester, NY, USA, 2002.

[42] NAKAMURA, E. T.; DE GEUS, P. L. **Segurança de redes em ambientes cooperativos**. Novatec, São Paulo, 1 edition, 2007.

[43] NAVARRO, K. F.; LAWRENCE, E.; LIM, B. **Medical MoteCare: A distributed personal healthcare monitoring system**. In: *Proceedings of the 2009 International Conference on eHealth, Telemedicine, and Social Medicine*, p. 25–30, Washington, DC, USA, 2009. IEEE Computer Society.

[44] OLZAK, T. **Secure your bluetooth wireless networks and protect your data**, 12 2006. http://www.techrepublic.com/article/secure-your-bluetooth-wireless-networks-and-protect-your-data/6139987 [Online; acessado em 1-dezembro-2009].

[45] PANTELOPOULOS, A.; BOURBAKIS, N. **A survey on wearable sensor-based systems for health monitoring and prognosis**. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(1):1 –12, 1 2010.

[46] PATEL, M.; WANG, J. **Applications, challenges, and prospective in emerging body area networking technologies**. *Wireless Commun.*, 17:80–88, February 2010.

[47] PATRICK, D.; FARDO, S. **Industrial electronics: devices and systems**. Fairmont Press, 2000.

[48] PINHEIRO, J. M. S. **As redes com zigbee**. *Projeto de Redes*, 27, 2004.

[49] PUCCINELLI, D.; HAENGGI, M. **Wireless sensor networks: applications and challenges of ubiquitous sensing**. *Circuits and Systems Magazine, IEEE*, 5(3):19–31, 2005.

[50] REN, Y.; PAZZI, R. W. N.; BOUKERCHE, A. **Monitoring patients via a secure and mobile healthcare system**. *Wireless Commun.*, 17:59–65, February 2010.

[51] RIGGS, T.; ISENSTEIN, B.; THOMAS, C. **Spectral analysis of the normal electrocardiogram in children and adults**. *Journal of Electrocardiology*, 12(4):377 – 379, 1979.

[52] SANG, Y.; SHEN, H.; INOGUCHI, Y.; TAN, Y.; XIONG, N. **Secure data aggregation in wireless sensor networks: A survey**. In: *Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT'06. Seventh International Conference on*, p. 315–320. Ieee, 2006.

[53] SENE, I. G.; DA ROCHA, A. F.; DE A. BARBOSA, T. M. G.; DE O. NASCIMENTO, F. A.; CARVALHO, H. S. **Energy efficient simulator for patient monitoring in body sensor networks**. In: *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, p. 1537 –1540, Aug. 2008.

[54] SHNAYDER, V.; CHEN, B.-R.; LORINCZ, K.; JONES, T. R. F. F.; WELSH, M. **Sensor networks for medical care**. In: *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, p. 314–314, New York, NY, USA, 2005. ACM.

[55] SINGH, M. **Introduction to Biomedical Instrumentation**. PHI Learning Pvt. Ltd.

[56] SIRER, E. G. **MagnetOS overview**, Aug. 2008. http://www.cs.cornell.edu/People/egs/magnetos/overview.html [Online; acessado em 31-agosto-2010].

[57] SOUTO, E.; GUIMARãES, G.; VASCONCELOS, G.; VIEIRA, M.; ROSA, N.; FERRAZ, C.; KELNER, J. **Mires: a publish/subscribe middleware for sensor networks**. *Personal Ubiquitous Comput.*, 10:37–44, December 2005.

[58] SRISATHAPORNPHAT, C.; JAIKAEO, C.; CHUNG SHEN, C. **Sensor information networking architecture and applications**. *IEEE Personal Communications*, 8:52–59, 2001.

[59] TANENBAUM, A. S.; RENESSE, R. V. **Modern operating systems**. Prentice Hall, 3 edition, 2008.

[60] TEXAS INSTRUMENTS. **INA118 Precision, Low Power INSTRUMENTATION AMPLIFIER**. Burr-Brown Corporation, 4 1998.

[61] TEXAS INSTRUMENTS. **TLV2470, TLV2471, TLV2472, TLV2473, TLV2474, TLV2475, TLV247xA FAMILY OF 600−μA/Ch 2.8−MHz RAIL−TO−RAIL INPUT/OUTPUT HIGH−DRIVE OPERATIONAL AMPLIFIERS WITH SHUTDOWN**, volume SLOS232C. Texas Instruments, 6 1999.

[62] WIKIPEDIA. **Typical instrumentation amplifier schematic**, 3 2011. http://en.wikipedia.org/wiki/File:Op-Amp_Instrumentation_Amplifier.svg [Online; acessado em 19-junho-2011].

[63] WORLD HEALTH ORGANIZATION. **World Health Statistics 2011**. WHO Press, Geneva, 2011.

[64] YAO, Y. **Cougar: The network is the database**, Aug. 2008. http://www.cs.cornell.edu/bigreddata/cougar/ [Online; acessado em 31-agosto-2010].

[65] YAO, Y.; GEHRKE, J. **The cougar approach to in-network query processing in sensor networks**. *SIGMOD Rec.*, 31:9–18, September 2002.

[66] YU, X.; NIYOGI, K.; MEHROTRA, S.; VENKATASUBRAMANIAN, N. **Adaptive middleware for distributed sensor environments**. *IEEE Distributed Systems Online*, 4:–, May 2003.

[67] ZIGBEE ALLIANCE. **ZigBee specification v1.0**. ZigBee Alliance, 2005.