UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
EM INFORMÁTICA

DISSERTAÇÃO DE MESTRADO

Especificação de Componentes para Modelagem de Redes Locais de Computadores Sem Fio Padrão IEEE 802.11

Geovane Vitor Vasconcelos

Campina Grande-PB, Agosto de 2002

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Dissertação de Mestrado

ESPECIFICAÇÃO DE COMPONENTES PARA MODELAGEM DE REDES LOCAIS DE COMPUTADORES SEM FIO PADRÃO IEEE 802.11

Geovane Vitor Vasconcelos

Joberto Sergio Barbosa Martins
Orientador

Maria Izabel Cavalcanti Cabral
Orientadora

Campina Grande, Paraíba, Brasil
Agosto de 2002

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ESPECIFICAÇÃO DE COMPONENTES PARA A MODELAGEM DE REDES LOCAIS DE COMPUTADORES SEM FIO PADRÃO IEEE 802.11

Dissertação submetida à Coordenação do Curso de Pós-graduação em Informática da Universidade Federal de Campina Grande, como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de concentração: Ciência da Computação.

Linha de pesquisa: Redes de Computadores e Sistemas Distribuídos.

Geovane Vitor Vasconcelos

Joberto Sergio Barbosa Martins
Orientador

Maria Izabel Cavalcanti Cabral
Orientadora

Campina Grande, Paraíba, Brasil
Agosto de 2002

Ficha Catalográfica

VASCONCELOS, Geovane Vitor

V33EM

Especificação de Componentes para Modelagem de Redes Locais de Computadores Sem Fio Padrão IEEE 802.11.

Dissertação (Mestrado), Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande - Pb Agosto de 2002.

119 p. Il.

Orientadores: Joberto Sérgio Barbosa Martins

Maria Izabel Cavalcanti Cabral

Palavras-Chave:

1. Simulação Digital
2. Redes de Computadores Sem Fio
3. Componentes de Software

CDU - 519.876.5

"ESPECIFICAÇÃO DE COMPONENTES PARA MODELAGEM DE REDES LOCAIS DE COMPUTADORES SEM FIO PADRÃO IEEE 802.11"

GEOVANE VITOR VASCONCELOS

DISSERTAÇÃO APROVADA EM 28.08.2002

PROFª MARIA IZABEL CAVALCANTI CABRAL, D.Sc
Orientadora

PROF. JOBERTO SÉRGIO BARBOSA MARTINS, Dr.
Orientador

PROF. ELMAR UWE KURT MELCHER, Dr.
Examinador

PROF. ÁLVARO FRANCISCO DE C. MEDEIROS, D.Sc
Examinador

CAMPINA GRANDE – PB

"Apenas uma palavra: Deus."

Agradecimentos

A Deus, sem dúvidas, por tudo. E pensar que ainda tem gente que não acredita Nele.

A minha mãe, que mesmo não estando mais aqui conosco, foi e sempre será a pessoa mais importante da minha vida.

A minha esposa Patrícia que me acompanha nas batalhas da vida na função de ser o ar que me faz respirar.

Aos meus filhos Vitor e Vitória, que são a principal a razão de minha vida, e é por eles que tento fazer com que as coisas boas aconteçam na minha vida.

Ao meu pai Genival, meus irmãos Geane, Girleide e George, que sempre torcem por mim e me incentivam com aquele união e amor que sempre tivemos um pelo outro.

Aos meus demais familiares, em especial ao meu tio Geraldo, a pessoa mais respeitada e querida de nossa família.

Ao meu grande amigo Jeová e sua digníssima família (Jaidete, Kiury e Kaila). Jeová, que sempre me incentivou e faz parte da minha vida, o qual adotei como irmão, mesmo não girando muito bem da cabeça (risos).

Aos meus irmãos adotados de mestrado Edna Dias e Dean James do Pará, que fazem parte da minha vida de uma forma muito especial.

Aos meus orientadores professor Joberto Sergio Barbosa Martins e professora Maria Izabel Cavalcanti Cabral, que contribuíram de forma grandiosa para que eu conseguisse essa conquista. São verdadeiros profissionais aos quais devo muito. A atenção, a dedicação, o compromisso, a empolgação, o incentivo, a qualidade, enfim, a forma como eles me orientaram, me deixou ainda mais confiante e certo do sucesso alcançado. Obrigado a vocês dois de coração.

Aos demais colegas de mestrado, em especial a Eulanda, Carlinhos, Flávio, Lidiana, Fernando e Elizabeth.

Ao professor Antônio Berto Machado, que me deu uma nova visão da vida quando da realização da disciplina Metodologia do Ensino Superior.

Aos demais professores da COPIN, que contribuíram com suas experiências e conhecimentos nas aulas.

Aos demais funcionários da COPIN, em especial a Aninha, Zeneide e Vera. O que seria de nós mestrandos sem elas? Da mesma forma dona Inês, que com sua simpatia e carisma, conquista o carinho e respeito de todos.

Aos meus professores de graduação Severino Paiva, Luiz Maurício, Kamiensk, Josemar Viana, Nilton, Cosmo, Fechine, Micheline, Cândido, Álvaro e Adriano, que me deram um alicerce sólido e consistente para que eu tivesse sucesso na presente conquista. Aos dois primeiro professores, um carinho especial por terem acreditado em mim não apenas como aluno, mas também como profissional.

Resumo

As redes de computadores sem fio têm evoluído consideravelmente nos últimos anos tendo assumido lugares de destaque nas soluções de comunicação para ambientes locais. Apostando nessa tecnologia, o IEEE (*Institute of Electrical and Electronics Engineers*) constituiu um grupo de pesquisa para investir na evolução das redes locais sem fio, através de um projeto denominado padrão IEEE 802.11. Redes de computadores, a exemplo de redes sem fio, podem exigir a construção de ferramentas de simulação específicas que permitam estudar e analisar os seus desempenhos face à contínua evolução de suas tecnologias. Esta Dissertação de Mestrado apresenta uma especificação de componentes de software para ser utilizada na construção de ferramentas de simulação específicas para a modelagem e avaliação de desempenho de sistemas de redes locais de computadores sem fio padrão IEEE 802.11. Essa especificação apresenta recursos das fases de análise e de projeto do processo de desenvolvimento proposto por Craig Larman [Larman98], que adota UML (*Unified Modeling Language*) como recurso de modelagem. O projeto detalhado apresentado volta-se para a tecnologia de componentes JavaBeans, orientada a eventos. Os componentes propostos representam os elementos essenciais de modelos de redes sem fio *ad hoc* e multi-célula, quais sejam: Fonte de Tráfego, Estação Origem, Camada de Acesso ao Meio e seu Protocolo, Enlace, Ponto de Acesso Origem, Sistema de Distribuição, Ponto de Acesso Destino, Estação Destino e Sorvedouro. A especificação de componentes de software apresentada pode ser reutilizada na construção de novas ferramentas de simulação como também na extensão de ferramentas já existentes.

Abstract

The wireless computer network technologies have considerably evolved in the last decade and, as such, do represent an important technical alternative for local area networking. The IEEE (Institute of Electrical and Electronics Engineers), through the development of the IEEE 802.11 standard project, is the main institution promoting the use and boosting this new technology for networking in the global scenario. Development of computer network systems typically requires the use of tools like simulators. These tools are mainly used to study and analyze system performance in relation either to the available technological alternatives or with respect to the expected system behavior. This Master Degree Dissertation presents the software components specification that could be used in the construction of simulation tools for computer networks based on the IEEE 802.11 standard. The specification presents the resources for the analysis and project phases according with the development process, as proposed by Craig Larman. UML (Unified Modeling Language) is used as modeling resource. The detailed project presented is based on event oriented JavaBeans components. The proposed components represent the essential elements for modeling both wireless ad hoc and multi-cell networks. The proposed components are: traffic source, source station, link, access protocol, source access point, distribution system, destination access point, destination station and sink. The software components specification presented may be easily reused for constructing new simulation tools or, alternatively, for extending available functionalities in existing simulation tools.

Sumário

Capítulo 1 – Introdução

Capítulo 2 – Simulação digital

Capítulo 3 – Redes sem fio IEEE 802.11

Capítulo 4 – Fase de análise

Capítulo 5 – Fase de projeto

Lista de tabelas e figuras

Capítulo 01

Introdução

1.1 Introdução

Este capítulo contextualiza toda esta Dissertação de Mestrado, destacando sua finalidade, motivação, objetivo, relevância e a organização dos demais capítulos.

1.2 Motivações

Várias são as motivações que sustentam a elaboração deste trabalho. O investimento em três importantes assuntos da área de computação promove naturalmente o estímulo para a produção desta Dissertação de Mestrado. Simulação digital, redes locais de computadores sem fio e desenvolvimento de componentes de software são os assuntos explorados. Vejamos informações relevantes sobre cada uma deles e o que oferecem como motivação.

1.2.1 Simulação digital

A simulação digital é uma técnica numérica que tem sido usada para estudar sistemas do mundo real. Ela permite prever o comportamento de diversos tipos de sistemas, fornecendo resultados que são usados em tomada de decisões. Do ponto de vista prático, a simulação constitui-se no projeto e construção de modelos de sistemas reais ou propostos, visando compreender seus comportamentos em um conjunto de situações específicas [Kelton98].

As ferramentas de simulação tornaram-se cada vez mais populares devido ao surgimento de novos instrumentos de software e do avanço da tecnologia de hardware, que fez com que os computadores ficassem mais eficientes e baratos e, em conseqüência, mais acessíveis às pessoas e empresas que trabalham na construção e no aprimoramento de sistemas computacionais.

A simulação digital consiste basicamente na criação de modelos – modelo aqui é definido como sendo uma representação matemática do sistema real, que destaca aspectos relevantes para análise – que abstraem as características mais importantes e essenciais do funcionamento dos sistemas reais [Kelton98].

A técnica da simulação digital permite obter informações importantes para o estudo de um sistema, tais como [Souto93]:

- Identificação de problemas.
- Possibilidade de comparação de desempenho entre sistemas similares.
- Avaliação da capacidade dos recursos existentes em um sistema.
- Detalhamento de processos que compõem um sistema.
- Compreensão profunda do funcionamento de um sistema.
- Documentação precisa dos recursos que fazem parte de um sistema.

A simulação digital é freqüentemente usada na modelagem e avaliação de desempenho de sistemas de redes de computadores. Esses sistemas necessitam de estudos de desempenho, principalmente, devido à contínua evolução de suas tecnologias.

A construção de ferramentas de simulação voltadas para a modelagem e avaliação de desempenho de sistemas de redes de computadores pode ser complexa, devido, em geral, a quantidade de recursos a serem considerados e ao relacionamento dinâmico que esses recursos apresentam [Rocha02].

1.2.2 Redes de computadores sem fio

Entre as tecnologias de redes de computadores que têm recebido especial atenção nos últimos anos encontram-se as redes sem fio *(Wireless)*, com mais destaque as redes locais sem fio *(Wireless Local Area Network - WLAN)*. "Uma rede sem fio é um sistema que interliga vários equipamentos fixos ou móveis através da transmissão sem fio" [IEEE802.11a].

Uma rede sem fio proporciona flexibilidade e mobilidade, além de dispensar a construção de uma infra-estrutura de cabos, podendo ainda ser interligada a outras redes com fio, sendo também considerada a única solução para ambientes onde o lançamento de cabos é inviável.

Apostando nessa tecnologia, o IEEE (*Institute of Electrical and Electronics Engineers*) constituiu um grupo de pesquisa para criar padrões abertos para tornar a tecnologia das redes locais sem fio cada vez mais uma realidade. Esse projeto, denominado de Padrão IEEE 802.11, ou simplesmente Padrão 802.11, nasceu em 1990, mas ficou inerte por aproximadamente sete anos devido a fatores que não permitiam que a tecnologia sem fio evoluísse significativamente. Um desses fatores (o principal) era a baixa taxa de transferência de dados que inicialmente a tecnologia oferecia, que girava em torno de alguns milhares de bits por segundo (Kbps) [Soares95].

De acordo com a elevação da taxa de transferência de dados, que passou a atingir vazões de milhões de bits por segundo (Mbps), a rede sem fio começou a ser vista como uma tecnologia promissora e a receber investimentos para a construção de equipamentos que possibilitassem a comunicação sem fio entre computadores. Esse fato revela o avanço da microeletrônica, que tem tornando possível à construção de equipamentos para transmissão sem fio com eficiência e a preços cada vez mais acessíveis.

Apesar dessa significante elevação da taxa de transferência de dados que subiu de poucas dezenas de Kbps para 2Mbps, as redes de computadores sem fio não atendiam satisfatoriamente à necessidade de largura de banda desejada pelas empresas. Com isso, o IEEE investiu no melhoramento do Padrão 802.11, que passou a oferecer taxas de transferência de dados entre 5 e 11Mbps [IEEE802.11c], impulsionando a tecnologia sem fio e estimulando as comunidades científica e industrial a padronizarem, projetarem e a produzirem produtos para essas redes.

Atualmente, o foco das redes de computadores sem fio encontra-se no contexto das redes locais de computadores (*WLANs*), representadas pelo Padrão IEEE 802.11. Inicialmente, foram colocados em prática alguns padrões proprietários, através de empresas como IBM, CISCO, Telecom e 3COM [Zanetti99]. Hoje, essas e outras empresas baseiam seus produtos no Padrão 802.11, devido às inúmeras e já conhecidas vantagens que um padrão aberto oferece: interoperabilidade, baixo custo, demanda de mercado, confiabilidade de projeto, entre outras [3Com00].

1.2.3 Especificação de componentes de software

Na construção de ferramentas de simulação, **componentes** de software reutilizáveis são recursos que podem ser explorados para facilitar a construção dessas ferramentas. "Componentes são elementos de software auto-suficientes e reutilizáveis que podem interagir entre si, seguindo um grupo de regras pré-estabelecidas" [Englander97].

A Engenharia de Software propõe a utilização de um processo de desenvolvimento que usa um conjunto de recursos sistemáticos para a construção de sistemas de software de qualidade [Sommerville00]. Os processos de desenvolvimento têm comumente adotado uma abordagem de orientação a objetos com a linguagem de modelagem UML *(Unified Modeling Language)* [UML97].

A experiência comprova que durante o processo de desenvolvimento de software, as fases de análise (descrição do problema) e de projeto (descrição da solução) ocupam maior parte do tempo [Larman98]. Com isso, os processos de desenvolvimento atuais investem no sucesso das etapas que antecedem a implementação, ou seja, investem numa especificação que seja capaz de representar o sistema de forma completa, através documentos, diagramas, modelos, etc.

Dois processos de desenvolvimento bastante conhecidos são o processo unificado [Jacobson99] e o processo de Larman [Larman98]. O processo unificado sugere um conjunto de atividades capaz de transformar os requisitos do usuário em um sistema de software, usando UML para representar os artefatos do sistema. O processo unificado é direcionado a casos de uso (use cases), é centrado na arquitetura e usa uma filosofia de desenvolvimento iterativa e incremental [Jacobson99].

O processo de Larman sugere uma fase de planejamento e elaboração, onde são levantados os requisitos do usuário e, a partir disso, são realizadas as fases de análise, projeto e codificação para a construção do sistema. O processo de Larman também usa uma filosofia de desenvolvimento iterativa e incremental, e UML como recurso de modelagem [Larman98]. Esse processo decompõe o problema numa perspectiva de objetos. Ambos os processos facilitam a construção de sistemas de software orientados a objetos ou a componentes [Jacobson99] [Larman98].

Esta Dissertação adota o processo de Larman, que tem sido bastante usado no meio acadêmico, e oferece recursos de análise e projeto que podem facilmente representar componentes para simulação de redes sem fio. O processo de Larman sustenta naturalmente a especificação proposta nesta Dissertação, uma vez que a construção de componentes segue princípios e fundamentos semelhantes à abordagem orientada a objetos [Jones00].

1.3 Objetivos da Dissertação

1.3.1 Objetivo geral

Especificar componentes reutilizáveis de software que possam ser utilizados na construção de ferramentas de simulação específicas para a modelagem e avaliação de desempenho de sistemas de redes locais de computadores sem fio padrão 802.11. A especificação apresenta recursos das fases de análise e projeto do processo de desenvolvimento proposto em [Larman98], que usa UML (*Unified Modeling Language*) como recurso de modelagem. O projeto detalhado apresentado volta-se para a tecnologia de componentes JavaBeans, orientada a eventos.

Os componentes propostos, doravante denominados "Componentes 802.11", representam os elementos essenciais de modelos de redes sem fio *ad hoc* e multi-célula, quais sejam: Fonte de Tráfego, Estação Origem, Camada de Acesso ao Meio e seu Protocolo, Enlace, Ponto de Acesso Origem, Sistema de Distribuição, Ponto de Acesso Destino, Estação Destino e Sorvedouro.

1.3.2 Objetivos específicos

- Estudar a tecnologia sem fio no contexto das redes de computadores.
- Estudar o funcionamento das redes locais de computadores sem fio padrão 802.11.
- Estudar os recursos utilizados pela Engenharia de Software para o desenvolvimento de software (componentes), tais como processo de desenvolvimento de sistema de software, orientação a objetos, orientação a componentes, recursos de análise e projeto, UML e modelo de componentes (modelo JavaBeans).
- Especificar componentes reutilizáveis de software que representem os principais elementos dos modelos de redes *ad hoc* e multi-célula padrão 802.11, usando recursos de análise e projeto do processo de desenvolvimento proposto em [Larman98].

1.4 Relevância

No que diz respeito aos fatores relevantes inerentes à tecnologia sem fio podemos facilmente identificar a tendência atual de se implantar cada vez mais as redes sem fio ao invés de redes com fio. Essa propensão é motivada pelos seguintes aspectos:

- Inviabilidade da instalação de redes com fio em alguns lugares;

- Barateamento dos equipamentos para as redes sem fio;
- Interoperabilidade da tecnologia devido à evolução do Padrão 802.11;
- Velocidade satisfatória que a tecnologia das redes sem fio atualmente já oferece, que atualmente gira em torno de 11Mbps, com projeção para 54Mbps [IEEE802.11b].

Outros fatores relacionam-se com as facilidades de mobilidade e flexibilidade que as comunicações sem fio possibilitam, estimulando esforços das comunidades científica e industrial para padronizar e projetar produtos que permitam a comunicação sem fio, tanto no contexto de redes de computadores como em outros contextos, como por exemplo, na comunicação de dispositivos baseada na tecnologia *Bluetooth* (*Bluetooth Special Interest Group* [Zanetti99]), que se propõe a interligar dispositivos próximos através da transmissão sem fio, tais como teclados, mouse, monitor.

> **Mais de vinte fabricantes competem atualmente em um mercado impulsionado pela necessidade de substitutos para sistemas com fio em instalações de redes. Vários produtos têm sido desenvolvidos e colocados a disposição no mercado [UFRGS00].**

Devido à complexidade inerente às tecnologias de redes de computadores e ao fato de que algumas, entre elas, as redes sem fio, demandarem estudos mais detalhados, existe a necessidade de se avaliar esses sistemas de comunicação, visando dar suporte a decisões. Para isso, pode-se usar a técnica de simulação digital, que se apresenta como uma importante alternativa para viabilizar esses estudos.

Ferramentas de simulação voltadas para a modelagem e a avaliação de desempenho de redes sem fio são escassas. O Network Simulator (NS) [NS98] é um exemplo de uma ferramenta bem conhecida. O NS apresenta código aberto e pode modelar redes em geral. No entanto, ele não é específico, por exemplo, para a tecnologia das redes sem fio, embora já possua um módulo para a simulação dessas redes. Ambientes de mais alto nível como o BONeS Designer [AltaGroup96] por sua vez, procuram facilitar o processo de modelagem apresentando recursos como animação e interpretação dos dados gerados, dentre outros. No entanto, esses ambientes são produtos comercializados a preços altos.

Uma outra alternativa para modelar e avaliar redes de computadores é investir no desenvolvimento de ferramentas de simulação específicas voltadas para a tecnologia de interesse, usando linguagens de programação de propósito geral, como C, C++ e Java. Essas ferramentas devem apresentar flexibilidade de forma que novas funcionalidades decorrentes da evolução natural das tecnologias possam ser a elas agregadas com facilidade. Um aspecto considerado relevante é o fato de que essas ferramentas podem ter seus processos de desenvolvimento simplificados através da reutilização de software.

A presente Dissertação investe na especificação de componentes que representam elementos de uma rede sem fio padrão 802.11. Essa especificação pode ser reutilizada na construção de novas ferramentas de simulação como também na extensão de ferramentas já existentes, facilitando a agregação de novas funcionalidades.

Finalmente, também destacamos como fator relevante, a multidisciplinalidade desta Dissertação, em que são envolvidas três importantes áreas da computação: Simulação Digital, Redes de Computadores e Engenharia de Software (especificação de componentes de software).

1.5 Organização da Dissertação

Esta Dissertação de Mestrado está organizada em seis capítulos (incluindo esta Introdução). No capítulo 2, abordamos a técnica de simulação digital, mostrando como ela se aplica à modelagem e à avaliação de desempenho de sistemas de redes de computadores.

No capítulo 3, apresentamos um estudo sobre a tecnologia das redes locais sem fio padrão 802.11, identificando os elementos funcionais dessas redes que são especificados neste trabalho. Em seguida, nos capítulos 4 e 5, apresentamos a especificação dos componentes 802.11. No capítulo 4, apresentamos os recursos de análise e no capítulo 5 os recursos de projeto. No capítulo 6, apresentamos a conclusão desta Dissertação, revelando sua contribuição e indicando sugestões para trabalhos futuros.

Por fim, apresentamos um apêndice (Apêndice A) contextualizando os recursos de análise e projeto do processo de desenvolvimento proposto por Larman [Larman98], usado na especificação dos componentes 802.11. Recomendamos a leitura desse apêndice antes da leitura dos capítulos 4 e 5, que apresentam a especificação proposta.

Capítulo 02

Simulação Digital

2.1 Introdução

Neste capítulo abordamos a técnica da simulação digital, destacando seu uso na avaliação de desempenho de sistemas discretos, especialmente nos sistemas de redes de computadores.

A experiência tem mostrado a necessidade de se avaliar o comportamento de diversos sistemas do mundo real, com a finalidade de reunir informações que podem ser usadas na tomada de decisões sobre esses sistemas. Sistemas emergentes, a exemplo de redes de computadores sem fio, necessitam ser continuamente avaliados, devido à necessidade de se conhecer seus funcionamentos.

Para analisar sistemas de redes de computadores é necessário definir um conjunto de medidas de desempenho relevantes de interesse, para que se possa observar o comportamento dos sistemas, permitindo conhecer suas habilidades em suportar a demanda de sua utilização [Wagner00]. Apresentamos a seguir algumas dessas medidas de desempenho no contexto de um sistema de redes de computadores:

- **Atraso médio fim-a-fim:** atraso médio de transmissão de um quadro entre duas estações.
- **Vazão média:** taxa média de quadros transmitidos por unidade de tempo (geralmente quadros/segundo).
- **Fator de utilização do meio de comunicação:** fração da capacidade total utilizada pelo canal de comunicação.
- **Descarte de quadros:** quantidade de quadros descartados pelos elementos da rede.

Para avaliar o desempenho de redes de computadores usamos basicamente duas formas [Celestino90]:

- Campanhas de medição em sistemas reais e operacionais; e
- Construção de um modelo para um sistema proposto ou existente.

Cada uma dessas duas formas de avaliação de sistemas discretos possui características próprias que definem sua aplicabilidade. A primeira alternativa avalia o desempenho das redes de computadores através de campanhas de medições (*benchmarks*). A segunda alternativa, a de construir um modelo do sistema, pode ser mais interessante, devido à possibilidade de se construir modelos que projetem as diversas configurações e situações do sistema, de forma relativamente simples e econômica [Souto93].

Um modelo representa as características fundamentais do comportamento de um sistema, e pode ser avaliado a partir de duas alternativas [Damoun79]:

- Através da técnica de simulação digital; e
- Através de estudos analíticos, baseado na Teoria das Filas.

A alternativa de estudos analíticos se apresenta como a mais econômica e eficiente [Dias92]. Entretanto, a aplicação da solução analítica em sistemas complexos pode se tornar inviável, uma vez que essa alternativa geralmente impõe limitações ao modelo. Com isso, para viabilizar o estudo de sistemas complexos, muitas vezes, a simulação digital se apresenta como a melhor solução [Cabral98].

2.2 Simulação digital

A simulação digital é baseada na construção de modelos que mostram o comportamento dos sistemas reais ou fictícios. Esses modelos permitem realizar experimentos automatizados sobre sistemas complexos, que foram implementados ou não. E ainda, a simulação digital permite realizar experimentos de um sistema diversas vezes, até se obter a melhor condição de funcionamento desse sistema [Dias92].

Um modelo representa uma abstração do sistema real, reunindo um conjunto de características relevantes definidas para a modelagem [Damoun79]. O escopo do modelo depende do que se pretende modelar, ou seja, depende do problema que se pretende dimensionar.

Para construir um modelo é necessário definir o nível de abstração desejado. Vários níveis de abstração podem ser usados, desde um nível mais alto até um nível mais baixo. Quanto mais alto for o nível, menos detalhado será o modelo [Wagner00]. A filosofia adotada é a de se construir o modelo a partir do nível mais abstrato, realizando os refinamentos incrementais até chegar no nível mais baixo, possibilitando a contemplação completa do modelo, revelando os detalhes de funcionamento do sistema em questão.

Quanto mais baixo for o nível de abstração atingido, maior será o tempo utilizado na criação do modelo e, em conseqüência, maior será o detalhamento alcançado. Com isso, podemos observar que o sucesso da simulação depende muito do esforço de elaboração do modelo. Além disso, é necessário um conhecimento elevado na utilização de ferramentas de simulação, requisito fundamental para otimização do tempo total usado no processo de simulação.

É também importante ressaltar que a qualidade do modelo e a escolha dos dados que são aplicados ao referido modelo, vão refletir diretamente nos resultados obtidos, ou seja, um ótimo modelo e dados adequados levam a resultados satisfatórios [Kelton98].

2.3 Processo de simulação

Para a construção do modelo de simulação é preciso adotar um processo que abranja todas as atividades necessárias. Conforme [Kelton98], essas atividades são as seguintes:

- Formulação do problema.
- Especificação funcional do sistema e da simulação.
- Formulação e construção do modelo.
- Verificação.
- Validação.
- Análise.
- Apresentação dos resultados.

O processo de simulação sugere que essas atividades sejam realizadas de forma incremental e adaptadas de acordo com a particularidade do sistema sobre o qual será construído o modelo. Sendo assim, pode-se afirmar que o processo de simulação é realizado através de vários ciclos, onde cada um desses ciclos representa um refinamento do modelo [Kelton98].

A Tabela 2.1 apresenta um resumo da definição de cada atividade que faz parte do processo de simulação [Kelton98].

Atividade	**Comentário**
Formulação do Problema	Investe na definição do domínio do problema, identificando o sistema que será modelado, os limites desse sistema, as métricas de desempenho adotadas, os valores que são considerados para essas métricas, enfim, todos os objetivos do estudo que devem ser atingidos e contemplados com o modelo de simulação.

Especificação Funcional do Sistema e da Simulação.	Uma vez definido o problema e os objetivos que se desejam alcançar, busca-se nesta etapa entender o funcionamento detalhado do sistema, e definir os parâmetros necessários a sua abstração. Basicamente, esses parâmetros representam os dados de entrada e saída da simulação.
Formulação e Construção do Modelo.	Nessa etapa é feita a construção propriamente dita do modelo no ambiente computacional, em que é definida a lógica de controle, a especificação dos dados que devem ser usados e o nível de detalhamento do modelo.
Verificação	Aqui, verifica-se se a lógica aplicada ao modelo está de acordo com o projeto de modelagem.
Validação	Esta atividade consiste em verificar se o modelo está representando corretamente o sistema real, particularmente com relação ao comportamento do sistema e aos resultados obtidos na simulação.
Análise	Considerando a validação do modelo, nesta etapa, identificam-se os problemas encontrados, a intensidade desses problemas e a viabilidade da solução.
Apresentação dos Resultados	Significa apresentar os resultados alcançados, de forma simples e natural, uma vez que a apreciação desses resultados é feita não apenas pelo desenvolvedor do modelo como também por outras pessoas não envolvidas na simulação.

Tabela 2.1 – Atividades do processo de simulação.

2.4 Simuladores

Simuladores são ferramentas de software usadas para construir e simular os modelos dos sistemas [Soares90]. Cada ferramenta oferece um conjunto de recursos necessários para que os sistemas sejam representados, processados e analisados.

A adoção da abordagem orientada a objetos na construção de simuladores revela a possibilidade de adicionar novos elementos/funcionalidades, permitindo simular um conjunto maior de sistemas.

Um exemplo de um ambiente de simulação de propósito geral (voltado para a modelagem e avaliação de desempenho de qualquer sistema) é o Arena [Kelton98]. Este ambiente foi projetado com abordagem orientada a objetos, usando a linguagem visual C++. Tendo sido

projetado especificamente para plataforma Windows, o Arena utiliza a tecnologia *ActiveX*, o que lhe permite uma fácil integração com programas que também utilizam essa tecnologia, como o editor de texto Winword e a planilha de cálculo Excel [Wagner00], ambos produtos da empresa Microsoft Corporation.

Um outro simulador que merece destaque é o Ptolemy [PtolemyII02]. O Ptolemy foi construído usando a linguagem de programação Java, e tem como principio de funcionamento a simulação dirigida por eventos. Sua proposta é a de modelar um conjunto grande e complexo de sistemas, tais como sistemas digitais, sistemas de comunicação e dispositivos mecânicos. Esse simulador oferece uma interface gráfica amigável e um conjunto significativo de componentes usados para a construção dos modelos.

Outros simuladores se voltam para sistemas específicos, a exemplo de redes de computadores. Seguem informações sucintas de três conhecidos simuladores que também adotaram a abordagem orientada a objetos nos seus desenvolvimentos:

- **Network Simulator (NS)** [NS98]**:** trata-se de uma ferramenta de código aberto construída em C++. O NS oferece módulos de simulação para diversos tipos de sistemas, com ênfase em redes TCP-IP. Atualmente o NS já modela redes sem fio. Aos poucos o NS vem ganhando interfaces gráficas interessantes. Seu uso é bastante popular, principalmente no meio acadêmico.
- **OPNET Modeler** [OPNET02]**:** é uma ferramenta comercial que oferece vários módulos para simulação. Possui excelentes recursos gráficos, possibilitando a criação gráfica de redes, nodes, links, sub-redes, protocolos, equipamentos, serviços e a ilustração geral dos modelos. Lançado em 1987, foi um dos primeiros simuladores de redes. Atualmente, o OPNET Modeler trabalha com modelagem orientada a objetos.
- **SimATM** [SimATM02]**:** simulador desenvolvido pela UNICAMP possui código aberto e foi construído em C++. É específico para os estudos de redes ATM. Seu princípio de simulação é o de orientação a eventos.

Esta Dissertação foca a especificação de componentes para a construção de ferramentas de simulação de redes de computadores sem fio, utilizando um processo de desenvolvimento que adota UML e observando a tecnologia JavaBeans. Essa especificação pode ser reutilizada na construção de novas ferramentas de simulação como também na extensão de ferramentas já existentes.

Capítulo 03

Redes sem fio IEEE 802.11

3.1 Introdução

A especificação proposta nesta Dissertação sugere a utilização de um processo de desenvolvimento. Esse processo decompõe o problema para saber *o que* é o sistema e *como* construí-lo. Para isso, é necessário conhecer o sistema que se deseja construir. Portanto, é preciso conhecer o funcionamento das redes locais de computadores sem fio padrão IEEE 802.11, identificando seus elementos e características relevantes.

Esta Dissertação visa especificar componentes de software que possibilitem a modelagem do funcionamento das redes 802.11. Com isso, apresentamos neste capítulo a fundamentação necessária sobre esse tipo de rede de computadores, para a especificação dos componentes que vão representar seu funcionamento. Essa especificação é apresentada nos capítulos 4 e 5.

3.2 Redes IEEE 802.11

Uma rede sem fio (*Wireless*) é tipicamente uma extensão de uma rede local (*Local Area Network* - LAN) convencional com fio, criando-se o conceito de rede local de computadores sem fio (*Wireless Local Area Network* – WLAN). Uma WLAN converte pacotes de dados em onda de rádio ou infravermelho e os envia para outros dispositivos sem fio ou para um ponto especial que serve como uma conexão para uma LAN cabeada [IEEE802.11a].

O IEEE (*Institute of Electrical and Electronics Engeneers*) constituiu um grupo chamado de *Wireless Local-Area Networks Standard Working Group*, com a finalidade de criar padrões para redes sem fio, definindo um nível físico para redes onde as transmissões são realizadas na freqüência de rádio ou infravermelho, e um protocolo de controle de acesso ao meio. Esse

padrão é conhecido como Padrão IEEE 802.11, ou simplesmente Padrão 802.11, e tem, entre outras, as seguintes premissas [IEEE802.11a]:

- Suportar diversos canais;
- Sobrepor diversas redes na mesma área de canal;
- Apresentar robustez com relação a interferência;
- Possuir mecanismos para evitar estações perdidas (*hidden node*);
- Oferecer privacidade e controle de acesso ao meio.

A maioria das redes sem fio é baseada no Padrão 802.11, que tem evoluído consideravelmente [IEEE802.11b]. A evolução se deu basicamente no aumento da taxa de transferência de dados, que girava entre 1 a 2Mbps, para 5 a 11Mbps. O Padrão IEEE especifica uma arquitetura comum, métodos de transmissão e outros aspectos de transferência de dados sem fio, permitindo a interoperabilidade entre os produtos e dispositivos sem fio [Soares95].

Apesar de que só nos últimos anos as redes locais de computadores sem fio começaram a fazer parte do contexto mundial dos sistemas de comunicação, essa tecnologia não é tão recente assim. As WLANs já são discutidas desde 1990, mas seus produtos caros inviabilizaram seu uso. Atualmente, temos comunicação sem fio sendo usada pela medicina móvel, no atendimento aos pacientes, e em outras aplicações, tais como controle de inventário, restaurantes, transações comerciais e bancárias e sistemas de ensino [3Com00].

A evolução do Padrão 802.11 e o conseqüente barateamento dos equipamentos para as redes sem fio, tornaram as WLAN uma realidade. Com isso, as redes sem fio ficaram mais acessíveis para algumas empresas, aumentando consideravelmente a comercialização de produtos para computadores móveis, como o cartão PCMCIA para Notebook, e o cartão ISA/PCI para PCs [Nogee01].

A evolução do padrão não pára em 11Mbps. Taxas de até 54Mbps já são trabalhadas pelo Padrão 802.11 [IEEE802.11b], embora essa recente evolução da tecnologia oferece produtos sem fio muito onerosos e, em conseqüência, ainda inacessíveis à maioria das empresas.

3.2.1 Vantagens e desvantagens das redes sem fio

Apesar da considerável diferença em relação à velocidade, quando comparadas as redes tradicionais com fio, as redes sem fio também oferecem vantagens que as tornam singulares em relação a outros sistemas similares. Duas dessas vantagens merecem destaque: flexibilidade e mobilidade.

A flexibilidade é reconhecida por não existir estações fixas como acontece com as redes com fio, já que estações sem fio (computadores móveis e até mesmo de mesa), podem facilmente ser transportadas para outros lugares e continuar fazendo parte da rede, desde que continuem dentro da área de abrangência do sinal. A flexibilidade acaba revelando uma outra vantagem: a diversidade de topologias que uma rede sem fio pode assumir.

Mas o maior destaque se dá a mobilidade, em que estações móveis podem se mover dentro da área de cobertura sem perder a comunicação, de forma ininterrupta. Mobilidade já é algo oferecido pelos sistemas de telefonia celular, GPS (*Global Position System*), radiofonia, transmissão de TV, etc, sistemas que possuem os principais princípios seguidos pelas redes sem fio [Alencar98].

Outras vantagens que as redes sem fio têm em relação às redes cabeadas que merecem destaque, são a dispensa de lançamento de cabos, a facilidade de instalação e a robustez associada a continuar existindo mesmo ocorrendo desastres naturais. Como desvantagens podemos citar a limitação da velocidade, a curta distância entre as estações e a vulnerabilidade na segurança dos dados que trafegam na rede.

O custo elevado do projeto e instalação de uma rede sem fio também ainda é considerado uma desvantagem, pois mesmo dispensando o lançamento do sistema de cabeamento, os equipamentos e as placas de comunicação sem fio ainda são itens mais onerosos do que seus respectivos similares de uma rede cabeada. Por exemplo, uma placa para uma LAN cabeada custa tipicamente de 3 a 6 vezes menos do que uma placa para uma WLAN. Essa diferença fica ainda mais significativa quando consideramos os equipamentos [Nogee01].

3.3 Arquitetura IEEE 802.11

O Padrão 802.11 define uma arquitetura para as redes sem fio baseada na divisão da área coberta pela rede em células. Essas células são denominadas de BSA *(Basic Service Area)*. O tamanho da célula (BSA) depende das características do ambiente e da potência dos transmissores/receptores usados nas estações. Outros elementos fazem parte do conceito da arquitetura de rede local sem fio, quais sejam [IEEE802.11a]:

- **BSS *(Basic Service Set)*** – representa um grupo de estações comunicando-se por radiodifusão ou infravermelho em uma célula. Seguindo este conceito, podemos observar que cada célula é representada por um BSS. Duas estações sem fio se comunicando numa mesma célula já caracterizam um BSS.
- **Ponto de acesso *(Access Point – AP)*** – são estações especiais responsáveis pela captura das transmissões realizadas pelas estações de sua célula, destinadas às estações localizadas em outras células, retransmitindo-as através de um sistema de distribuição.
- **Sistema de distribuição (*Distribution System – DS*)** – representa uma infra-estrutura de comunicação que interliga múltiplas células para permitir a construção de redes cobrindo áreas maiores que uma célula. Esse sistema de distribuição pode ser com fio.
- **ESA *(Extend Service Area)*** – representa a interligação de várias células pelo sistema de distribuição através dos APs.
- **ESS *(Extend Service Set)*** – representa um conjunto de estações formado pela união de vários BSSs conectados por um sistema de distribuição.

As estações mencionados na definição dos elementos da arquitetura, são todos os equipamentos transmissores e receptores de dados. Dois tipos de estações sem fio merecem destaque:

- **Estações fixas sem fio** – são computadores fixos ligados à rede através de conexões sem fio.
- **Estações móveis sem fio** – são equipamentos, tipicamente notebooks e PDAs (*Personal Digital Assistant*), que são ligados à rede através de conexões sem fio. A mobilidade ocorre com esses equipamentos, utilizando-se do mesmo princípio do sistema de telefonia móvel celular.

3.3.1 Redes *ad hoc* e multi-célula

Existem basicamente dois tipos de redes sem fio, definidas na arquitetura IEEE 802.11: redes "ad hoc" e redes "multi-célula". Numa rede *ad hoc*, as estações se comunicam dentro de uma mesma célula. Esse tipo de rede é caracterizada quando o ESS é formado por um único BSS (BSS independente – *Independent Basic Service Set – IBSS*), permitindo, dessa forma, que estações próximas e que estejam na mesma célula possam se comunicar. Nesse tipo de rede, não é necessário a utilização de nenhuma estação especial para controlar as comunicações e nenhum tipo de infra-estrutura que possa expandir a rede [IEEE802.11a].

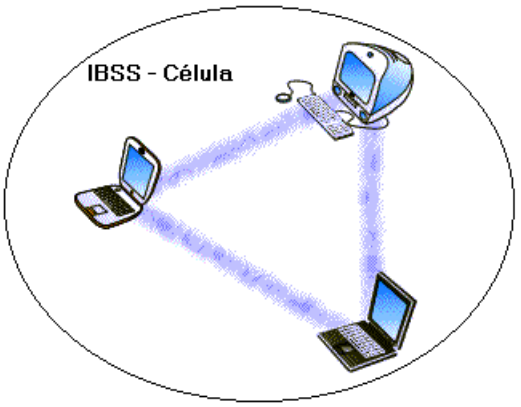

Figura 3.1 – Rede IEEE 802.11 *ad hoc*.

O termo *ad hoc* significa funcionamento independente. No contexto das redes 802.11, *ad hoc* significa uma simplificação atribuída ao sistema de comunicação sem fio, em que se configura a comunicação independente de estações que fazem parte de uma mesma célula.

Numa rede sem fio multi-célula (com infra-estrutura), é necessária a interconexão de múltiplas células. Nesse caso, uma infra-estrutura de interligação de células é necessária. Essa infra-estrutura é representada pelos APs e pelo sistema de distribuição que interliga esses APs. O sistema de distribuição, além de interligar os vários pontos de acesso, fornece os recursos necessários para interligar a rede sem fio a outras redes [IEEE802.11a].

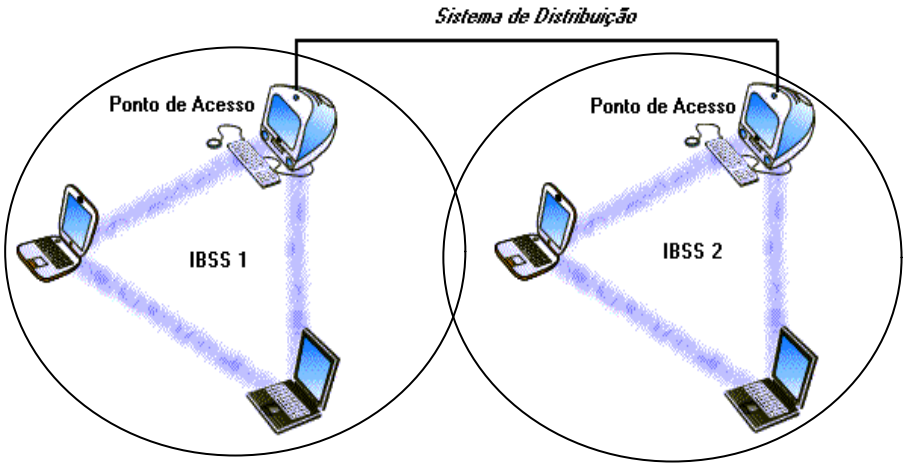

Figura 3.2 – Rede IEEE 802.11 multi-célula.

3.3.2 Sistema de distribuição (DS) e ponto de acesso (AP)

O sistema de distribuição geralmente é representado por um sistema de comunicação com fio (cobre ou fibra), e pode ser baseado em vários tipos de tecnologias como, por exemplo, na tecnologia Ethernet [IEEE802.11a].

O Padrão 802.11 não especifica o sistema de distribuição (DS). Portanto, o DS pode ser baseado em qualquer tecnologia. Um exemplo de DS seria um meio físico ponto-a-ponto ligando duas LANs em prédios distintos. Para uma rede sem fio usar o DS, o Padrão 802.11 especifica o que o DS deve suportar, mas não como isso é feito. Essa integração é feita através de serviços de comunicação, abordados na seção 3.5.

O Padrão 802.11 difere o meio das redes sem fio do meio usado no sistema de distribuição. Cada um desses meios é usado diferentemente pelos elementos que fazem parte de uma rede multi-célula: especificamente as estações e os pontos de acesso. No caso dos pontos de acesso, existem, no mínimo, duas interfaces nessas estações especiais, uma para a comunicação sem fio e a outra para a comunicação com o sistema de distribuição.

Com isso, qualquer que seja o meio usado na infra-estrutura, será possível ligar várias células, construindo um sistema de comunicação sem fio cada vez maior [IEEE802.11a]. Esse fato revela dois aspectos importantes: a flexibilidade do padrão, já que no mínimo duas mídias podem ser usadas, e a independência do padrão, uma vez que a arquitetura 802.11 é especificada independentemente das características do meio físico usado na interligação de células.

Além disso, a infra-estrutura não integra apenas redes sem fio. É possível e perfeitamente desejável ligar redes sem fio com redes cabeadas. Isso é feito através de um *portal*. Um portal representa a integração lógica desses dois tipos de redes, funcionando como uma ponte entre uma rede 802.11 e 802.X. A função de portal pode ser executada distintamente por uma estação especial ou por um ponto de acesso [IEEE802.11a].

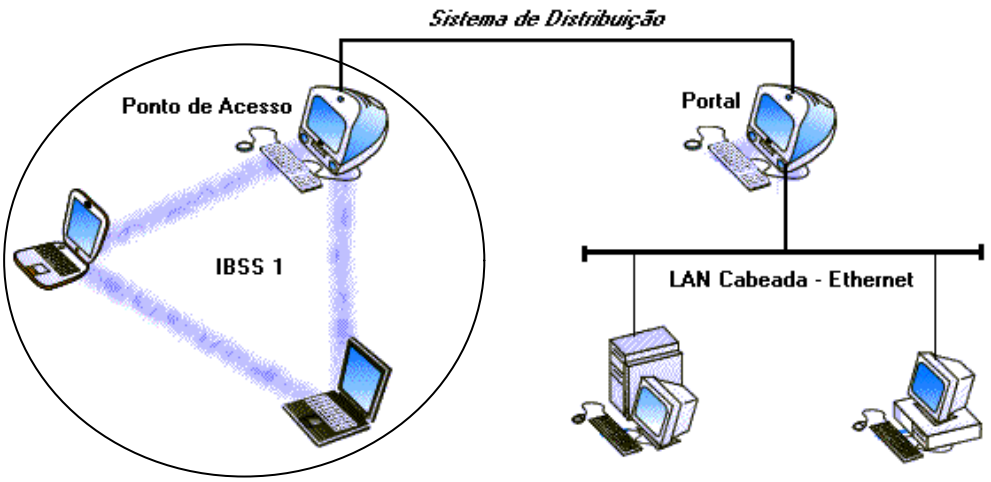

Figura 3.3 – Rede IEEE 802.11 com Rede 802.X.

Notamos que um ponto de acesso possui diversas funções, inclusive poder funcionar como um portal. Vejamos outras importantes funções desempenhadas por essa estação especial [Soares98]:

- **Autenticação, associação e reassociação** – permite que uma estação móvel mesmo saindo de sua célula de origem continue conectada à infra-estrutura e não perca a comunicação;

  A função que permite manter a continuidade da comunicação quando um usuário passa de uma célula para outra, é conhecida como *Roaming* [Alencar98].

- **Gerenciamento de potência** – permite que as estações operem economizando energia, através de um modo chamado *power save* (*standby*);
- **Sincronização** – garante que as estações associadas a um AP estejam sincronizadas por um relógio comum.

3.3.3 Modelo de referência 802.11

A arquitetura de uma rede sem fio padrão 802.11 abrange os níveis de comunicação *físico* e de *enlace*. O IEEE define esse padrão contextualizando esses dois níveis.

O IEEE define o funcionamento da maioria das redes locais que possuem arquitetura aberta, sempre especificando os níveis físico e de enlace de dados dessas redes. A Figura 3.4 mostra uma visão gráfica do Padrão IEEE 802.11, contextualizado-o com o modelo de referência usado para interconexão de sistemas abertos, o RM OSI da ISO (*Reference Model for Open Systems Interconnection*).

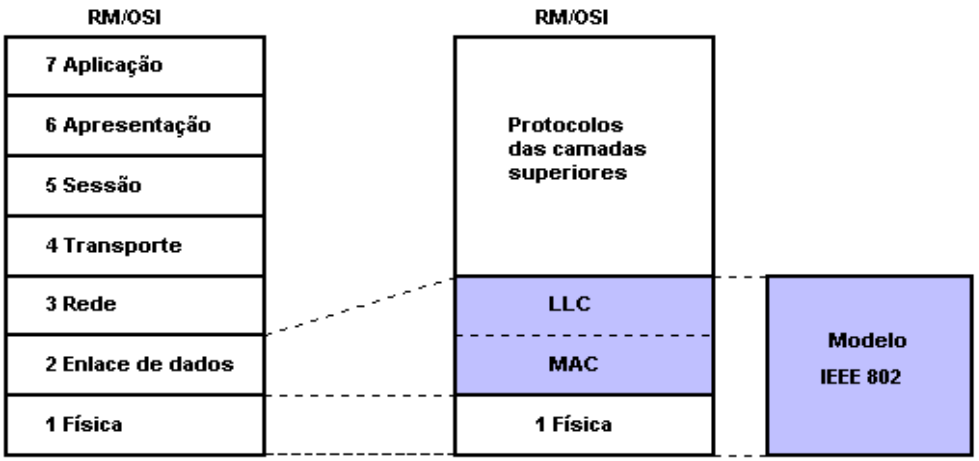

Figura 3.4 – Ilustração do modelo IEEE 802 com o RM OSI.

Especificamente com relação ao Padrão IEEE 802.11, o nível físico é especificado com transmissões em radiofreqüência ou infravermelho, e o nível de enlace com o protocolo de acesso ao meio (protocolo MAC 802.11). A Figura 3.5 ilustra com mais detalhe o padrão para as redes sem fio, mostrando as subcamadas da arquitetura, os pontos de acesso a serviços (*Services Access Point – SAP*) e as entidades de gerenciamento.

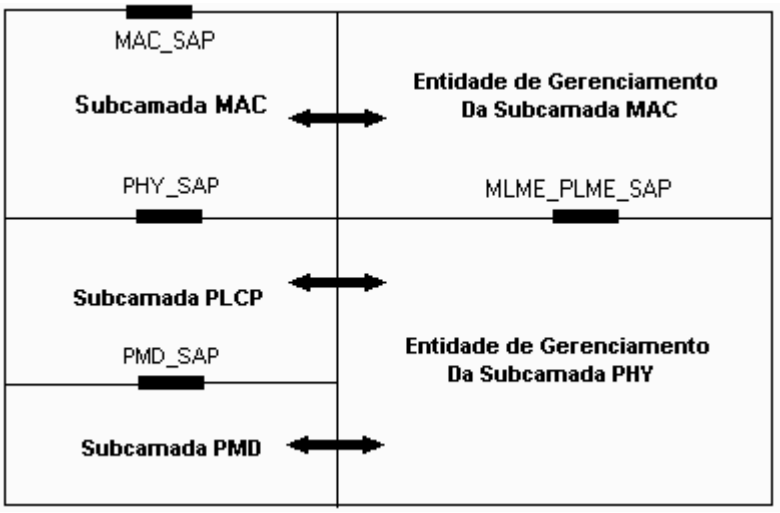

Figura 3.5 – Visão detalhada da arquitetura 802.11 (IEEE802.11, pág 28).

A seção seguinte aborda os conceitos relacionados ao nível físico. A fundamentação referente ao nível de enlace de dados é mostrada na seção 3.8.

3.4 Transmissão física 802.11

A transmissão de informações feita sem fio é algo que vem sendo usada há várias décadas. Transmissão de TV, radiofonia, telefonia celular e rádio amador, são alguns exemplos bastante conhecidos de uso da transmissão sem fio. Isso é possível representando a

informação através de ondas eletromagnéticas que se propagam no ar (inclusive no vácuo) [Tanenbaum96].

Essas ondas diferem por várias características, sendo a principal delas a sua freqüência. É de acordo com a freqüência que as formas de transmissão sem fio são classificadas em radiofreqüência, microondas, infravermelho, ultravioleta, raios X e raios gamas [Tanenbaum96].

Recentemente a transmissão sem fio tem sido usada em outros contextos, como em redes de computadores, por exemplo. Teoricamente qualquer tipo de transmissão sem fio poderia ser usada na comunicação entre computadores. Na prática isso não acontece, uma vez que cada forma de transmissão reúne um conjunto de características, vantagens e desvantagens, que define sua aplicabilidade. O Padrão 802.11, por exemplo, define suas transmissões usando as alternativas radiofreqüência e infravermelho.

3.4.1 Técnicas de transmissão FHSS e DSSS

A grande maioria das redes sem fio que seguem o padrão 802.11 usa transmissões com radiofreqüência, através de duas técnicas [Torres01]:

- FHSS (*Frequency Hopping Spread Spectrum – Espalhamento Saltante de Freqüência*);
- DSSS (*Direct Sequence Spread Spectrum – Espalhamento Direto de Freqüência*).

A idéia é a de usar várias freqüências na transmissão de dados, e não uma única freqüência fixa. Com isso, diminuem-se problemas de confiabilidade, uma vez que o sinal não será representado apenas por uma freqüência, e aumentam-se as taxas de transmissão, já que várias freqüências podem ser usadas numa mesma transmissão de informações.

Tanto a técnica FHSS como DSSS utilizam freqüências diferentes na transmissão de dados. A Figura 3.6 ilustra essas técnicas.

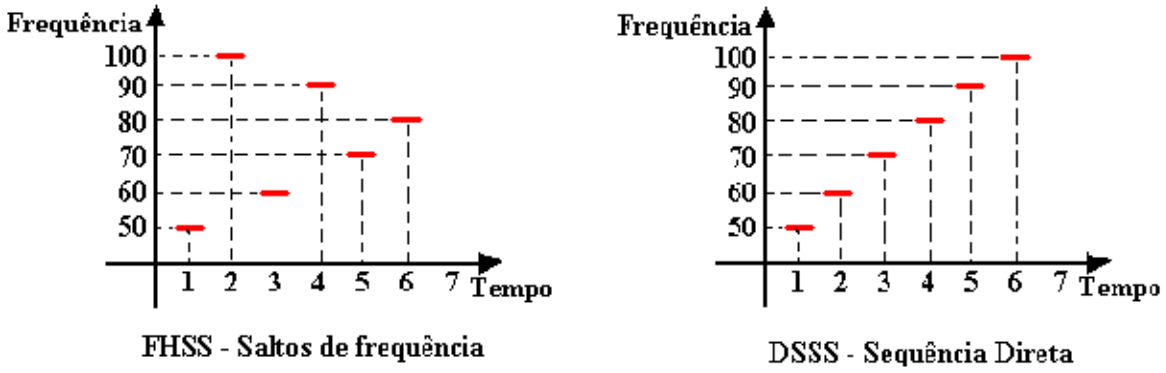

Figura 3.6 – Técnicas de transmissão FHSS e DSSS.

A técnica de transmissão DSSS funciona de maneira similar à técnica FHSS. A única diferença encontra-se na forma de como a técnica DSSS muda de canal. Na técnica DSSS, a troca de canal é feita de forma seqüencial, enquanto que na técnica FHSS é feita de forma aleatória, conforme ilustrado na Figura 3.6. Esse é um dos fatores que faz com que a técnica DSSS consiga atingir taxas de transmissão de 5,5Mbps e 11Mbps, além das taxas de 1 ou 2Mbps, que já são oferecidas pela técnica FHSS [Torres01].

Com relação a área de abrangência do sinal, em termos aproximados, para ambientes fechados a área da rede sem fio pode chegar a 120 metros, com FHSS ou DSSS. Para ambientes abertos, essa área pode chegar a 300 metros, também com ambas as técnicas [Torres01]. Essa característica é comum para ambas às técnicas, e a área pode variar tanto para maior como para menor, não pelo o uso de FHSS ou DSSS, e sim pelas características do ambiente.

Tanto a técnica DSSS como a FHSS são especificadas pelo Padrão 802.11. A técnica DSSS se apresenta com a mais interessante, uma vez que oferece maior largura de banda. Portanto, a técnica DSSS é a mais usada nas redes 802.11 [IEEE802.11a].

Outros tipos de transmissão podem ser usados em redes locais de computadores sem fio, tais com infravermelho e laser. Esses tipos de transmissão também utilizam suas técnicas específicas. O próprio Padrão 802.11 especifica transmissão com infravermelho através de uma técnica de transmissão conhecida como *difusa*.

3.4.2 Faixa de freqüência

O Padrão 802.11 usa faixas de freqüências conhecidas como ISM (*Industrial - Scientific - Medical*), que assumem freqüências de 900MHz, 2.4GHz e 5GHz. Quanto maior a freqüência maior é a quantidade de informação que um dispositivo pode enviar num canal [Tanenbaum96].

As primeiras redes locais sem fio operavam na freqüência de 900MHz, atingindo uma taxa de 256Kbps. O Padrão 802.11 aumentou a taxa de transmissão para 1Mbps, usando a técnica FHSS, e posteriormente para 2Mbps, usando a técnica DSSS, trabalhando na freqüência de 2.4GHz.

Conforme já mencionado, a técnica DSSS oferece taxas maiores por trocar de canal de forma seqüencial. Mas não é só por isso. Para conseguir taxas maiores, a técnica DSSS também utiliza freqüências mais altas do que a técnica FHSS. Por isso que a maioria das empresas que fabricam equipamentos sem fio optou pela técnica DSSS.

Além de permitir taxas de transferência satisfatórias para o padrão, as faixas ISM são de domínio público, não sendo necessário, portanto, solicitar qualquer tipo de concessão ou autorização de algum órgão do governo para o uso dessas faixas (aqui no Brasil esse órgão é a Anatel – Agência Nacional de Telecomunicações). Com isso, os fabricantes de dispositivos de

comunicação para redes locais sem fio usam essa facilidade e investem ainda mais na tecnologia.

3.5 Serviços de comunicação 802.11

Para que haja a integração das redes sem fio entre células e com outras redes, criando-se um sistema de comunicação maior, o Padrão 802.11 não especifica explicitamente o sistema de distribuição. Ao invés disso, o padrão especifica serviços de acesso. Esses serviços são executados pelos componentes da rede sem fio. Duas categorias de serviços são definidas: o serviço estação (*Station Service – SS*) e o serviço sistema de distribuição (*Distribution System Service – DSS*). Ambos serviços são especificados na subcamada de acesso ao meio, e são os seguintes [IEEE802.11a]:

- Autenticação;
- Associação;
- Desautenticação;
- Desassociação;
- Distribuição;
- Integração;
- Privacidade;
- Reassociação;
- Entrega de dados.

Alguns desses serviços (os SS) são executados pelas estações. Outros (os DSS) são executados exclusivamente pelos pontos de acesso, dentro do contexto do sistema de distribuição. Os SS são os seguintes [IEEE802.11a]:

- Autenticação;
- Desautenticação;
- Privacidade;
- Entrega de dados.

Os SS são especificados para todas as estações, inclusive para os pontos de acesso, quando estas estações especiais desempenham funções de estações comuns. Os DSS são realizados pelos pontos de acesso para a integração de células. Esses serviços são os seguintes [IEEE802.11a]:

- Associação;
- Desassociação;
- Distribuição;
- Integração;
- Reassociação.

Esses serviços são implementados quando a comunicação envolve, por exemplo, uma rede com infra-estrutura (que contém pontos de acesso e sistema de distribuição), ou seja, uma rede multi-célula.

Os DSS "associação, reassociação e desassociação", estão relacionados com a mobilidade das estações, e esses serviços ocorrem dinamicamente, como resultado do movimento das estações na rede. Com isso, é possível que estações de uma célula possam se comunicar com estações de outras células, e que estações móveis possam visitar outras células sem que percam a comunicação.

Ambos os serviços – os SS e DSS – são especificados na subcamada de acesso ao meio (MAC) do Padrão 802.11.

3.5.1 Descrição dos serviços de comunicação 802.11

A seguir descrevemos os serviços de comunicação 802.11, com a finalidade de entendermos como ocorre a comunicação nas redes 802.11. A disposição seqüencial em que os serviços são definidos visa facilitar o entendimento do funcionamento desses serviços. Todos os nove serviços são especificados em [IEEE802.11a].

Serviço DISTRIBUIÇÃO

O serviço "distribuição" é classificado na categoria DSS e representa um serviço básico usado pelas estações da rede, quando a comunicação se dá através de um sistema de distribuição. Vejamos na Figura 3.7 uma mensagem de dados sendo enviada da *estação1* (origem) para *estação4* (destino), que ficam localizadas em BSSs distintos que são ligados por um sistema de distribuição (DS).

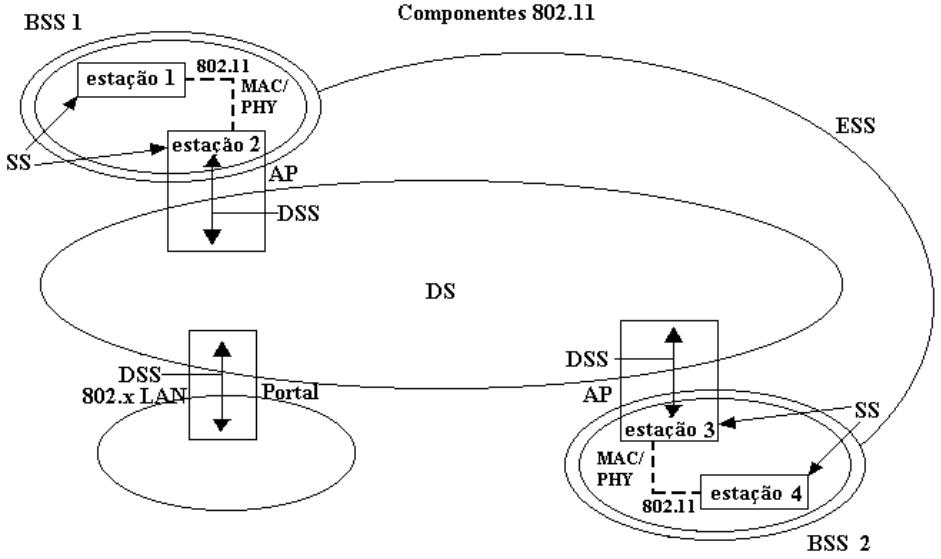

Figura 3.7 – Componentes da Rede IEEE 802.11 (IEEE802.11a, pág 16).

A mensagem é enviada da *estação1* para o AP do BSS1 (ponto de acesso origem da transmissão – que é a estação2), que executa o *serviço distribuição* no DS, com a finalidade de enviar os dados para o AP do BSS2 (ponto de acesso destino da transmissão – que é a estação3), que entregará esses dados à *estação4*. Localizar o AP do BSS2 e fazer a mensagem de dados chegar até ele é função do serviço de distribuição.

Um caso especial nesse serviço é quando o AP origem e o AP destino são os mesmos, ou seja, quando as estações origem e destino fazem parte de um mesmo BSS. Nesse caso, o serviço de distribuição utilizará ou não o DS, sendo essa decisão baseada na configuração da rede.

Basicamente o serviço de distribuição é executado pelos pontos de acesso. Para isso, essas estações especiais executam os serviços de associação, reassociação e desassociação, abordados mais a frente.

Serviço INTEGRAÇÃO

O serviço "integração" também é um DSS. Esse serviço faz com que as mensagens sejam entregues da mídia do DS para a mídia da LAN que está sendo integrada pelo portal. O serviço de integração é executado antes do serviço de distribuição, com intuito de integrar as mídias das LANs com a do DS. Só depois dessa integração, o serviço de distribuição é executado.

O serviço de integração inicia quando se verifica que o AP destino é um portal, ou seja, quando este integra uma rede 802.11 a uma rede 802.X, a qual pertence a estação destino envolvida na comunicação.

Serviço ASSOCIAÇÃO

O propósito básico do protocolo MAC é o de transferir dados entre as entidades da subcamada MAC de estações distintas. Para isso, é preciso que o serviço de distribuição seja executado. Este serviço necessita de informações sobre o estabelecimento da comunicação entre as estações envolvidas na comunicação. Essas informações são fornecidas pelo serviço de "associação".

O conceito de associação, mais reassociação e desassociação, suporta o conceito de mobilidade. Existem basicamente três tipos de mobilidade previstos pelo Padrão 802.11:

- **Sem transição** – comporta duas alternativas de mobilidade numa mesma área de serviço básico (BSA – célula):
    - **Estática:** não há mobilidade.
    - **Mobilidade local:** quando uma estação se move unicamente dentro de seu BSA.
- **Mobilidade BSS** – quando uma estação se move de um BSS para outro BSS de um mesmo ESS.

- **Mobilidade ESS** – quando uma estação se move de um BSS para outro BSS de diferentes ESSs.

O serviço de associação suporta os diferentes tipos de mobilidades supracitados. Para entregar uma mensagem dentro de um DS, o serviço de distribuição precisa saber qual AP acessar para uma determinada estação destino. Essa informação é provida ao DS pelo serviço de associação. Este tipo de serviço é classificado como DSS.

Vejamos como uma associação é feita. Quando uma estação necessita enviar uma mensagem de dados por um AP, primeiro ela deve se associar a este AP. Essa associação é realizada pelo serviço de associação, que estabelece uma ligação entre a estação, o AP e o DS. Com base nas informações que representam esta ligação, o DS usa o serviço de distribuição para o envio da mensagem.

Uma estação só deve se associar a um único AP. Isso garante que o DS saiba qual AP está servindo aquela estação, para que as comunicações com a mesma sejam estabelecidas e mantidas. A solicitação de associação é sempre iniciada pela estação, e um AP pode associar várias estações.

Serviço REASSOCIAÇÃO

O serviço de "reassociação" também é um serviço DSS. O serviço reassociação é necessário quando a associação é estendida (movida) de um AP para outro AP que fazem parte de BSSs distintos, mas de um mesmo ESS, possibilitando, dessa forma, a mobilidade entre BSS. Com isso, uma estação consegue alterar sua ligação de um AP para outro.

Serviço DESASSOCIAÇÃO

O serviço "desassociação" representa o último serviço da categoria DSS. O *serviço desassociação* é executado sempre quando for necessário desfazer uma associação. Esse serviço, diferentemente dos quatro anteriormente definidos, pode ser iniciado não apenas pela estação, mas também pelo AP, quando algum desses elementos sentir a necessidade de desfazer a comunicação na qual eles estejam envolvidos. A desassociação pode ocorrer, entre outras, pelas seguintes razões: necessidade de remover algum AP da rede, e quando uma estação deixa uma rede.

Os quatro próximos serviços que veremos fazem parte da categoria SS. São eles autenticação, desautenticação, privacidade e entrega de dados.

Serviço AUTENTICAÇÃO

A segurança das informações é uma questão importante nas redes sem fio. Em redes com fio, a segurança das informações pode ser parcialmente garantida pela segurança física dos fios, tornando esses sistemas menos vulneráveis à interceptação de dados por pessoas não autorizadas, quando comparados aos sistemas sem fio.

Para abordar esse problema, o Padrão 802.11 define, dentre outros, o serviço "autenticação", usado por todas as estações da rede, com a finalidade de que cada estação tenha sua identificação autenticada. Duas estações só podem se comunicar uma com a outra, se elas se autenticarem reciprocamente. Caso o serviço de autenticação não tenha sucesso, o serviço de associação não será estabelecido, e, conseqüentemente, a comunicação não ocorrerá.

O serviço de autenticação é executado para identificar a estação na rede. Numa rede *ad hoc* (IBSS), essa autenticação pode ser feita entre duas estações quaisquer da rede. Numa rede multi-célula, essa autenticação é feita pelo respectivo AP do BSS ao qual pertence a estação.

Existem dois tipos de autenticação:

- **Autenticação aberta *(Open System Authentication)*** – qualquer estação pode receber autenticação, bastando para isso solicitá-la.
- **Autenticação de chave *(Shared Key Authentication)*** – só é fornecida autenticação para as estações que tenham uma chave secreta específica. Esse mecanismo é implementado por um algoritmo de privacidade chamado WEP (*Wired Equivalent Privacy* – Privacidade Equivalente a Rede Cabeada). Esse mecanismo tem o propósito de tornar a privacidade de uma rede sem fio equivalente a de uma rede com fio.

Ressaltamos que o serviço de autenticação aqui mencionado atua no nível de enlace, e que outros tipos de autenticação podem ser implementadas em níveis superiores da arquitetura.

Serviço DESAUTENTICAÇÃO

O serviço de "desautenticação" é invocado quando for necessário desfazer uma autenticação. Na comunicação de BSSs distintos de um mesmo ESS, onde a associação é o elemento fundamental para estabelecer a comunicação, a desautenticação implicará na finalização também da associação (realizado pelo serviço desassociação).

O serviço de desautenticação pode ser solicitado tanto por uma estação como por um AP.

Serviço PRIVACIDADE

Numa rede cabeada, apenas as estações conectadas ao cabeamento podem receber o tráfego da rede. Nas redes sem fio, onde o meio é completamente compartilhado por todas as estações (autorizadas ou não autorizadas, que fazem parte da rede, que não fazem parte da rede) que estejam dentro da área de cobertura da rede (célula), esse fato acaba se tornando um sério problema conhecido como *violação de privacidade*. Uma simples ligação de uma estação sem fio com uma rede cabeada, já torna vulnerável a privacidade de toda a rede.

Para atacar esse problema, o Padrão 802.11 define o serviço "privacidade", que codifica o conteúdo das mensagens que trafegam na rede. Esse serviço é um mecanismo opcional que tem como finalidade tornar as comunicações sem fio ao menos tão seguras quanto as com fio.

Serviço ENTREGA DE DADOS

O serviço "entrega de dados" entrega dados entre estações após as fases de autenticação e associação, requisitos fundamentais para que ocorra uma comunicação entre estações da rede.

3.6 Mensagens 802.11

Cada serviço de comunicação é suportado por tipos de mensagens 802.11. O quadro 802.11 indica o tipo de mensagem que ele representa. Essa informação pode ser vista na seção seguinte, que trata sobre o formato do quadro MAC 802.11. Vejamos as principais informações contidas em cada tipo de mensagem, que são especificadas em [IEEE802.11a] e são fundamentais para a comunicação segundo o padrão IEEE 802.11:

Mensagens de DADOS

- Indicação de que o tipo da mensagem é de dados.
- Indicação de que o subtipo da mensagem é de dados.
- Endereço de origem da mensagem (endereço IEEE 802.11 – 48 bits).
- Endereço de destino da mensagem (endereço IEEE 802.11 – 48 bits).
- Identificação do BSS ao qual pertence a estação que enviou a mensagem.
- Direção da mensagem: estação – estação.

Mensagens de ASSOCIAÇÃO (pedido)

- Indicação de que o tipo da mensagem é de gerenciamento.
- Indicação de que o subtipo da mensagem é de pedido de associação.
- Endereço IEEE 802.11 da estação que iniciou o pedido de associação.
- Endereço IEEE 802.11 do AP que recebeu o pedido de associação.
- Identificação do ESS.
- Direção da mensagem: estação – AP.

Mensagens de ASSOCIAÇÃO (resposta)

- Indicação de que o tipo da mensagem é de gerenciamento.
- Indicação de que o subtipo da mensagem é de resposta de associação.
- Resposta do pedido de associação:
    - Associação sem êxito.
    - Associação com êxito – resposta inclui a identificação da associação (*Association-ID*).
- Direção da mensagem: AP – estação.

Mensagens de REASSOCIAÇÃO (pedido)

- Indicação de que o tipo da mensagem é de gerenciamento.
- Indicação de que o subtipo da mensagem é de pedido de reassociação.
- Endereço IEEE 802.11 da estação que iniciou o pedido de reassociação.
- Endereço IEEE 802.11 do AP com o qual a estação solicita a reassociação.
- Indicação do AP que a estação está associada no momento.
- Identificação do ESS.
- Direção da mensagem: estação – AP.

Mensagens de REASSOCIAÇÃO (resposta)

- Indicação de que o tipo da mensagem é de gerenciamento.
- Indicação de que o subtipo da mensagem é de resposta de reassociação.
- Resposta do pedido de reassociação:
    - Reassociação sem êxito.
    - Reassociação com êxito – resposta inclui a identificação da associação (*Association-ID*).
- Direção da mensagem: AP – estação.

Mensagens de DESASSOCIAÇÃO

- Indicação de que o tipo da mensagem é de gerenciamento.
- Indicação de que o subtipo da mensagem é de desassociação.
- Endereço IEEE 802.11 da estação que está sendo desassociada. O AP usa uma mensagem *broadcast* quando precisa desassociar todas as estações a ele associadas.
- Endereço IEEE 802.11 do AP com o qual a estação está associada no momento.
- Direção da mensagem: estação – estação, AP – estação, estação – AP.

Para executar uma autenticação, o serviço de autenticação usa um ou mais quadros de gerenciamento que devem ser trocados para contemplar tal tarefa. A seqüência exata e o conteúdo desses quadros vão depender do esquema de autenticação adotado. Isso porque vários tipos de autenticação podem ser implementados, o que permite opções quanto ao grau de segurança desejado.

A identificação de qual esquema (algoritmo) de autenticação está sendo usado na rede é feita no quadro de gerenciamento que representa a mensagem de autenticação. Vejamos detalhes dessas mensagens.

Mensagens de AUTENTICAÇÃO (primeiro quadro de uma seqüência de quadros)

- Indicação de que o tipo da mensagem é de gerenciamento.
- Indicação de que o subtipo da mensagem é de autenticação.
- Identificação de qual algoritmo de autenticação deve ser usado.
- Identificação da estação que está sendo autenticada.
- Número de seqüência na operação de autenticação.
- Informações dependentes do algoritmo de autenticação utilizado.
- Direção da mensagem: estação – estação. Lembramos que em redes *ad hoc* a autenticação ocorre entre duas estações quaisquer, e que numa rede multi-célula, a autenticação deve ser atendida por um AP.

Mensagens de DESAUTENTICAÇÃO

- Indicação de que o tipo da mensagem é de gerenciamento.
- Indicação de que o subtipo da mensagem é de desautenticação.
- Endereço IEEE 802.11 da estação que está sendo desautenticada.
- Endereço IEEE 802.11 da estação com a qual a estação sendo desautenticada está autenticada (é usado endereço *broadcast* para desautenticar todas as estações num dado momento autenticadas).
- De qual estação para qual estação ocorreu o pedido de desautenticação.

3.7 Formato do quadro MAC 802.11

Todas as estações de uma rede 802.11 podem gerar, enviar e receber quadros MAC. O quadro MAC 802.11 possui três elementos básicos: cabeçalho, corpo e uma seqüência de verificação do quadro. A estrutura do quadro MAC 802.11 é mostrada na Figura 3.8 [IEEE802.11a].

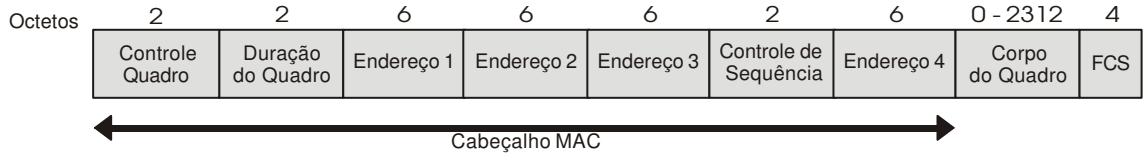

Figura 3.8 – Formato do quadro MAC 802.11.

- **Controle do quadro (*frame control*)** – ocupa 2 octetos e é composto por 11 subcampos, que sevem para definir a versão do protocolo de acesso ao meio, o tipo e subtipo do quadro, o destino do quadro, informações de fragmentação do quadro, informações de retransmissão do quadro, informações sobre a ativação ou não do mecanismo de privacidade, e controle de ordem do quadro.

- **Duração do quadro (*duration-ID*)** – ocupa 2 octetos e armazena o tempo de duração de um quadro, definido para cada tipo de quadro.
- **Endereço1 (*address1*)** – ocupa 6 octetos e é usado por todas as estações da rede numa comunicação *broadcast*.
- **Endereço2 (*address2*)** – ocupa 6 octetos e é usado para identificar a estação transmissora do quadro. Com esse campo, por exemplo, a estação destino sabe para qual estação enviar o *ack* de confirmação de recebimento de um quadro.
- **Endereço3 (*address3*)** – ocupa 6 octetos e é usado apenas quando o quadro está sendo transmitido através de um DS, indicando de onde e para onde vai o referido quadro.
- **Controle de seqüência (*sequence control*)** – ocupa 2 octetos e é composto por dois subcampos:
    - **Número do fragmento (*fragment number*)** – ocupa 4 bits e indica o número do fragmento da mensagem.
    - **Número de seqüência (*sequence number*)** – ocupa 12 bits e indica o número de seqüência da mensagem.
- **Endereço4 (*address4*)** – ocupa 6 octetos e é basicamente usado para indicar que o quadro veio de uma rede LAN cabeada.
- **Corpo do quadro (*frame body*)** – campo de tamanho variado que pode assumir valores que vai de 0 a 2.132 bytes. Este campo é usado para transportar os dados úteis transmitidos pela rede.
- **Seqüência de verificação do quadro** (*Frame Check Sequence - FCS*) – ocupa 4 octetos e é usado para implementar o mecanismo de verificação de erros.

3.8 Nível de enlace 802.11

3.8.1 Introdução

O nível de enlace é responsável por algumas das funções mais importantes de comunicação numa rede local de computadores. Seu papel é oferecer serviços de comunicação para os níveis superiores da arquitetura (aplicações), transformando as informações em unidades chamadas quadros (*frames*), e fazendo o acesso ao meio físico para a efetiva transmissão desses quadros [Tanenbaum96].

Para isso, o nível de enlace é subdividido em duas subcamadas: subcamada LLC (*Link Logical Control – Controle de Enlace Lógico*), que é definido pelo Padrão 802.2 para todas as tecnologias da família IEEE 802, e a subcamada MAC (*Medium Access Control – Controle de Acesso ao Meio*), que corresponde ao Padrão 802.11, no caso das redes locais de computadores sem fio [IEEE802.11a]. A subcamada MAC possui o protocolo de acesso ao meio, usado para a transmissão dos dados.

3.8.2 Protocolo MAC 802.11

Além de definir um mecanismo para transmissão física usando radiofreqüência ou infravermelho, o IEEE definiu um protocolo de acesso ao meio (subcamada MAC do nível de enlace de dados), denominado genericamente de DFWMAC (*Distributed Foundation Wireless Medium Access Control – Função de Distribuição MAC para Redes Sem Fio*), que suporta dois métodos de acesso [IEEE802.11a]:

- Método distribuído básico, que é obrigatório.
- Método centralizado, que é opcional.

O método de acesso distribuído forma à base sobre a qual é construído o método centralizado. Os dois métodos podem também ser chamados de funções de coordenação *(Coordination Functions)*. Uma "função de coordenação" é usada para decidir quando uma estação tem permissão para transmitir. Na função de coordenação distribuída *(Distributed Coordination Functions - DCF)*, essa decisão é realizada individualmente pelas estações da rede, podendo, dessa forma, ocorrer colisões. Na função de coordenação centralizada, também chamada de função pontual *(Point Coordination Function - PCF)*, a decisão de quando transmitir é centralizada em uma estação especial (geralmente um AP), que determina qual estação deve transmitir e em que momento, evitando teoricamente a ocorrência de colisões [Soares95].

3.8.2.1 Função de coordenação distribuída – DCF

A função de coordenação distribuída *(Distributed Coordination Function – DFC)* representa o método de acesso básico do protocolo de acesso ao meio MAC IEEE 802.11. O algoritmo utilizado é o CSMA/CA *(Carrier Sense Multiple Access / Collision Avoidance)* com reconhecimento. O CSMA/CA tem diferenças com relação ao protocolo CSMA/CD do Ethernet. Por exemplo, o CSMA/CD controla (detecta e trata de imediato) as colisões, enquanto que o protocolo CSMA/CA tenta evitar as colisões. A utilização dessa função distribuída é obrigatória para todas as estações e pontos de acesso, nas configurações *ad hoc* e multi-célula. O algoritmo CSMA/CA trabalha da seguinte maneira, quando uma estação deseja transmitir [IEEE802.11a]:

- A estação sente o meio para determinar se outra estação já está transmitindo.
- Se o meio estiver livre, a seqüência de transmissão do quadro pode prosseguir, caso contrário, a estação aguarda o final da transmissão que está em andamento.
- Após cada transmissão com ou sem colisão, a rede fica em um modo onde as estações só podem começar a transmitir em intervalos de tempo a elas pré-alocados.
- Ao findar uma transmissão, as estações alocadas ao primeiro intervalo têm o direito de transmitir. Se não o fazem, o direito passa as estações alocadas ao segundo intervalo, e assim sucessivamente, até que ocorra uma transmissão, quando todo o processo reinicia.

- Se todos os intervalos não são utilizados, a rede entra então no estado onde o CSMA sem pré-alocação de intervalos é usado para acesso, podendo, dessa forma, ocorrer colisões.

No método CSMA/CA pode ocorrer colisões e esse método não garante a entrega correta dos quadros. Com isso, uma estação após transmitir um quadro, necessita de um aviso de recebimento que deve ser enviado pela estação receptora (um quadro *ack*). Para isso, a estação que enviou o quadro aguarda um tempo *(timeout)* pelo aviso de recebimento do quadro por parte da estação destino. Caso esse aviso não chegue no tempo considerado, a estação origem realiza novamente a transmissão do quadro.

Para melhorar a transmissão de dados, o protocolo CSMA/CA acrescenta ao algoritmo básico com reconhecimento, um mecanismo opcional que envolve a troca de quadros de controle RTS *(Request To Send)* e CTS *(Clear To Send)* antes da transmissão de quadros de dados. Esse mecanismo funciona da seguinte forma [Soares95]:

- Uma estação antes de efetivamente transmitir o quadro de dados, transmite um quadro de controle RTS, que carrega uma estimativa da duração do tempo da futura transmissão do quadro de dados.
- A estação destino em reposta ao quadro de controle RTS, envia um quadro de controle CTS avisando que está pronta para receber o quadro de dados.
- Depois disso, a estação origem envia o quadro de dados, que deve ser respondido com um reconhecimento *(ack)* enviado pela estação destino.

O quadro RTS basicamente possui as seguintes funcionalidades:

- Reservar o meio para a transmissão do quadro de dados;
- Verificar se a estação destino está pronta para receber o quadro de dados.

Esta última funcionalidade é devido à possibilidade da estação destino estar operando no modo de economia de energia (modo *power save*). A Figura 3.9 apresenta a troca de dados para a transmissão de informações, usando o mecanismo opcional com RTS e CTS.

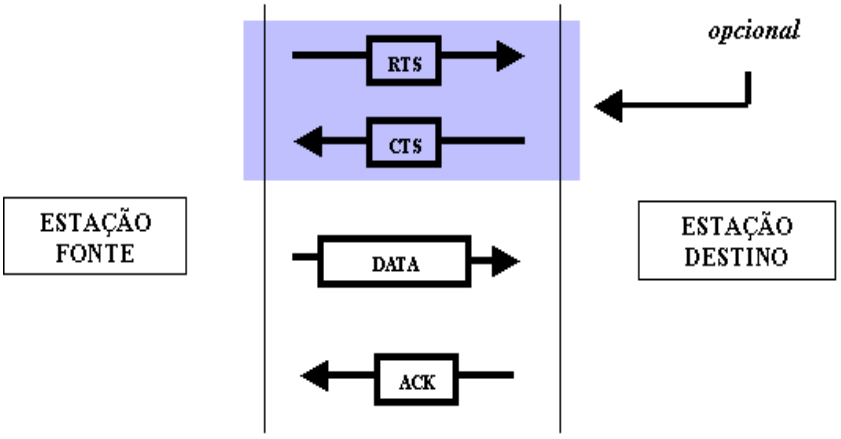

Figura 3.9 – Troca de dados de controle RTS/CTS (Soares, pág. 275).

A utilização da troca de quadros de controle RTS/CTS é interessante para as redes sem fio, uma vez que uma estação qualquer pode não escutar o quadro RTS enviado por um transmissor, mas pode escutar o quadro CTS enviado por um receptor. Considerando que os quadros RTS/CTS carregam uma indicação da duração da transmissão do quadro de dados, e que o Padrão 802.11 define que as estações trabalham sincronizadas, todas as estações ao receberem um quadro RTS ou CTS anotam a duração da futura transmissão do quadro de dados e, por conseqüência, adiam suas tentativas de transmissão para depois de passado o tempo anotado [Soares95].

Esse gerenciamento é feito com base numa estrutura de dados mantida nas estações denominada NAV (*Net Allocation Vector – Vetor de Alocação de Informações da Rede*). Uma vez que o mecanismo RTS/CTS é usado basicamente para proteger a transmissão de quadros longos, e para que as estações anotem a futura transmissão de quadros de dados de outras estações, e a partir daí tomarem decisões sobre suas transmissões, o uso do RTS/CTS é controlado por um parâmetro que varia de acordo com a estação, e define quais os quadros (baseado no tamanho) devem ser transmitidos usando ou não o mecanismo RTS/CTS.

3.8.2.2 Função de coordenação pontual

A função de coordenação pontual *(Point Coordination Function – PCF)* é uma função opcional que pode ser inserida no protocolo MAC IEEE 802.11, sendo construída sobre uma função de coordenação distribuída (DCF). Essa função só pode ser usada em redes com infra-estrutura e onde não haja intersecção entre BSSs que operam na mesma faixa de freqüência. A PCF é implementada através de um mecanismo de acesso ordenado ao meio, que suporta a transmissão de tráfego com retardo limitado ou tráfego assíncrono [Soares95].

Para a integração dessas duas funções – pontual e distribuída – é utilizado o conceito de superquadro, que permite que o protocolo possa trabalhar de uma forma em que a função pontual assuma o controle da transmissão, para evitar a ocorrência de colisões. Para isso, o protocolo MAC IEEE 802.11 divide o tempo em períodos denominados superquadros, que consiste em dois intervalos de tempo consecutivos, que são usados da seguinte maneira:

- No primeiro intervalo, controlado pela PCF, o acesso é ordenado, o que evita a ocorrência de colisões.
- No segundo intervalo, controlado pela DCF, o acesso baseia-se na disputa pela posse do meio, podendo ocorrer colisões.

A Figura 3.10 ilustra o conceito de superquadro.

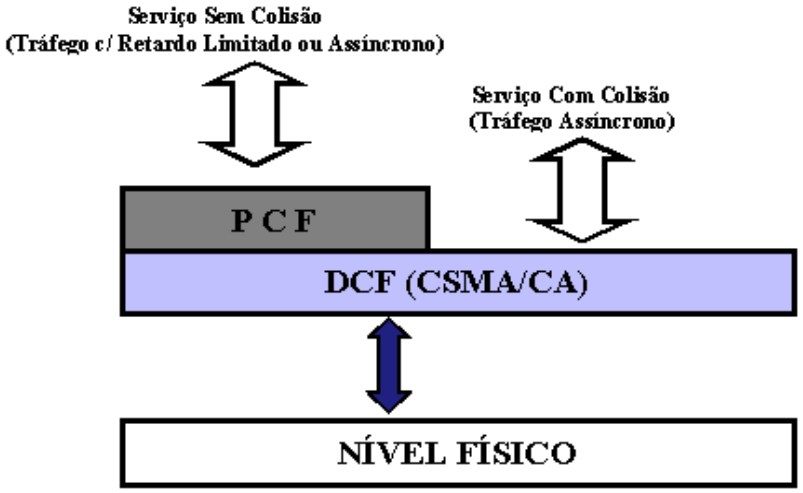

Figura 3.10 – Ilustração do conceito de superquadro (Soares, pág 277).

Capítulo 04

Fase de Análise

4.1 Introdução

A especificação dos componentes para modelagem das redes 802.11 proposta nesta Dissertação de Mestrado é apresentada neste capítulo e no capítulo seguinte. Neste capítulo focamos a *análise* da especificação, utilizando recursos do processo de desenvolvimento proposto por [Larman98] que fazem parte dessa fase. No próximo capítulo é focada a fase de *projeto.*

Os componentes especificados representam o funcionamento essencial das redes 802.11 e são descobertos nessa fase de análise, quais sejam: Fonte de Tráfego, Estação Origem, Camada de Acesso ao Meio e seu Protocolo, Enlace, Ponto de Acesso Origem, Sistema de Distribuição, Ponto de Acesso Destino, Estação Destino e Sorvedouro.

Outros componentes que fazem parte do contexto de simulação são ilustrados nesta Dissertação, com a finalidade de contemplar o entendimento conceitual de um ambiente de simulação, mas não são especificados, uma vez que não fazem parte do escopo deste trabalho. Esses componentes são especificados em [Lula01] e [Rocha02].

4.2 Requisitos do sistema

O objetivo desta Dissertação de Mestrado é especificar um conjunto de componentes que podem ser usados na construção de ferramentas de simulação para a modelagem e avaliação de desempenho de redes locais de computadores sem fio padrão 802.11. Para atingir esse

objetivo é preciso conhecer o contexto que define o escopo do problema (o domínio do problema).

Para isso, é preciso definir as necessidades do sistema, investindo na revelação de suas funcionalidades. Essas funcionalidades são identificadas através dos *requisitos funcionais* do sistema. Para levantarmos esses requisitos é preciso, essencialmente, descrever o sistema, definir seus usuários e estabelecer seus objetivos. Vejamos a definição de cada um desses itens.

- **O sistema:** componentes de software reutilizáveis para a construção ou aprimoramento de ferramentas de simulação de alto nível, com a finalidade de modelar redes sem fio padrão 802.11.
- **O usuário do sistema:** desenvolvedores de software de simulação digital.
- **O objetivo:** facilitar o desenvolvimento de novos ambientes de simulação ou mesmo estender as funcionalidades de ambientes já existentes, para tratar, especificamente, as redes sem fio 802.11.

Listamos a seguir os requisitos funcionais necessários para a construção dos componentes 802.11, baseados na fundamentação sobre as redes 802.11 apresentada no capítulo anterior. Inicialmente, listamos os requisitos funcionais. Em seguida, listamos os requisitos não funcionais do sistema.

Para os requisitos funcionais, consideramos todos eles do tipo *evidente*. A classificação desses requisitos em evidentes, ocultos e opcionais, pode ser vista no apêndice A.

RF	Descrição dos Requisitos Funcionais (RF)
RF01	Os componentes especificados devem representar os elementos essenciais para o funcionamento de uma rede sem fio 802.11.
RF02	Os componentes devem possibilitar a simulação de modelos de redes *ad hoc*.
RF03	Os componentes devem possibilitar a simulação de modelos de redes multi-célula.
RF04	Nos modelos de redes *ad hoc* e multi-célula, uma ou mais fontes de tráfego podem ser consideradas.
RF05	Nos modelos de redes *ad hoc* e multi-célula, uma ou mais estações emissoras podem ser consideradas na simulação.
RF06	Nos modelos de redes *ad hoc* e multi-célula, a transmissão deve ser feita através de um canal compartilhado.
RF07	Nos modelos de redes *ad hoc* e multi-célula, o acesso das estações ao enlace deve ser feito baseado no funcionamento básico do protocolo de acesso ao meio CSMA/CA.

RF08	Nos modelos de redes multi-célula, cada célula deve possuir um único ponto de acesso.
RF09	Nos modelos de redes multi-célula, as células são interligadas através de seus respectivos pontos de acesso por um sistema de distribuição.
RF10	Nos modelos de redes multi-célula, o sistema de distribuição deve ser exclusivo entre dois pontos de acesso.
RF11	Os componentes devem coletar, durante a simulação, os seguintes dados para o cálculo de medidas de desempenho de interesse: • Número de quadros transmitidos com sucesso. • Número de quadros descartados. • Número de quadros colididos. • Número de quadros recebidos. • Tempo de utilização do canal de comunicação (enlace e sistema de distribuição).
RF12	Os componentes devem oferecer propriedades que podem ser configuradas pelo usuário (analista de modelagem) quando da simulação dos modelos de redes *ad hoc* e multi-célula.

Tabela 4.1 – Requisitos funcionais para os componentes 802.11.

Vejamos na Tabela 4.2 os requisitos não funcionais relevantes para a construção dos componentes 802.11.

RNF	Descrição dos Requisitos Não Funcionais (RNF)
RNF01	Os componentes devem possibilitar o desenvolvimento de novas ferramentas de simulação.
RNF02	Os componentes devem possibilitar a extensão de funcionalidade de ferramentas de simulação existentes, para modelar redes sem fio padrão 802.11.
RNF03	A documentação da especificação dos componentes deve ser elaborada com detalhes para o entendimento completo do desenvolvedor de ferramentas de simulação.
RNF04	A especificação dos componentes deve adotar uma linguagem padrão de modelagem.
RNF05	A especificação dos componentes deve adotar um processo de desenvolvimento conhecido.

Tabela 4.2 – Requisitos não funcionais dos componentes 802.11.

4.3 Diagrama de caso de uso

Usamos a técnica de "caso de uso" (use case) para dois contextos: o primeiro, focando o ambiente de simulação de forma ampla e, o segundo, focando o funcionamento das redes 802.11.

4.3.1 Caso de uso Ambiente de Simulação

Nessa primeira visão, fazemos a ilustração do usuário interagindo com as funcionalidades do ambiente de simulação. Para um melhor entendimento de um sistema de simulação, o cenário apresentado possui como ator o analista de modelagem (usuário da ferramenta de simulação), elemento responsável pela configuração e execução do modelo [Lula01].

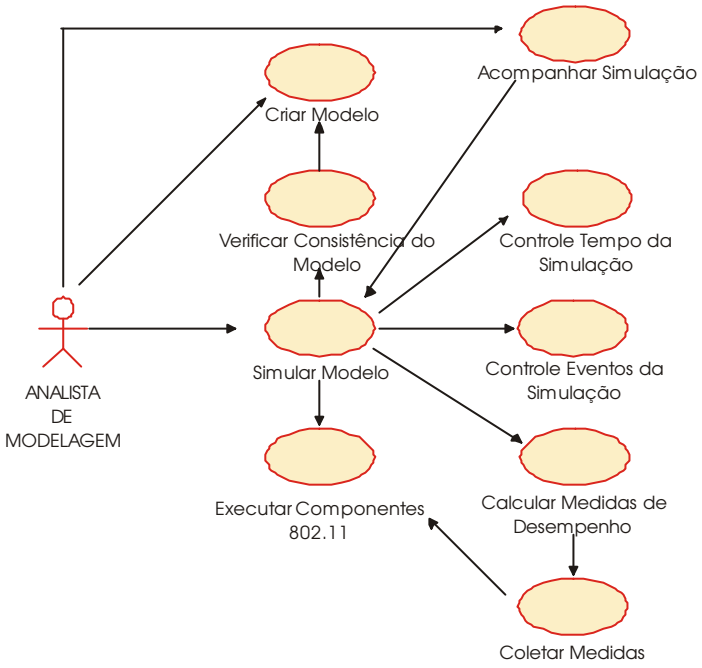

Figura 4.1 – Diagrama de caso de uso Ambiente de Simulação.

Nesta Dissertação, do modelo apresentado na Figura 4.1, apenas os casos de uso "Criar Modelo" e "Simular Modelo" são considerados. O caso de uso Simular Modelo promove o caso de uso "Executar Componentes 802.11", que é explorado com detalhes na subseção seguinte. O detalhamento dos casos de uso que tratam do ambiente de simulação pode ser visto em [Lula01] e [Rocha02].

4.3.2 Caso de uso Criar Modelo / Simular Modelo

A Figura 4.2 ilustra os casos de uso referentes à configuração do modelo e à execução dos componentes 802.11 necessários para representar o funcionamento das redes 802.11. Essa ilustração também tem como ator o analista de modelagem, usuário que irá criar o modelo de rede 802.11 que será simulado.

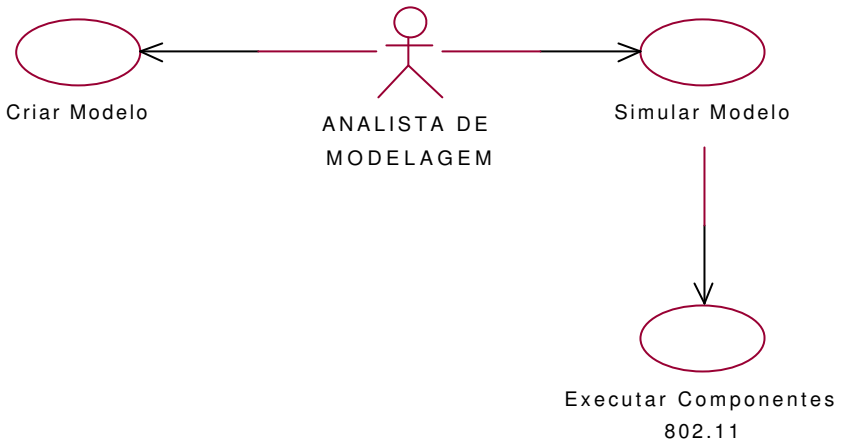

Figura 4.2 – Diagrama de caso de uso Criar Modelo / Simular Modelo.

Segue o detalhamento dos casos de uso ilustrados na Figura 4.2.

Caso de Uso 01	**CRIAR MODELO**
Ator	Analista de modelagem.
Tipo	Primário.
Descrição	O analista de modelagem configura o modelo de rede 802.11 que ele pretende simular. Para isso, o analista escolhe e configura os componentes 802.11 que são executados na simulação.

Os dois seguintes casos de uso apresentados correspondem aos casos de uso expandidos do caso de uso "Criar Modelo".

Caso de Uso 02	**CRIAR MODELO AD HOC**
Ator	Analista de modelagem.
Tipo	Primário.
Descrição	O analista de modelagem configura o modelo de rede 802.11 *ad hoc*. Neste tipo de rede os elementos essenciais necessários para a simulação são fonte de tráfego, estação origem, camada de acesso ao meio e seu protocolo, enlace, estação destino e sorvedouro.

Caso de Uso 03	**CRIAR MODELO MULTI-CÉLULA**
Ator	Analista de modelagem.
Tipo	Primário.
Descrição	O analista de modelagem configura o modelo de rede 802.11 multi-célula. Neste tipo de rede os elementos essenciais necessários para a simulação são fonte de tráfego, estação origem, camada de acesso ao meio e seu protocolo, enlace, ponto de acesso origem, sistema de distribuição, ponto de acesso destino, estação destino e sorvedouro.

Basicamente o entendimento representado pelo caso de uso Criar Modelo e seus casos de uso expandidos (casos de uso de 01 a 03), dispensa a elaboração do caso de uso "Simular Modelo" e sua conseqüente expansão, já que fica evidente que a simulação considera o modelo configurado pelo analista de modelagem, baseado no tipo de rede 802.11 que se pretende simular. O caso de uso Simular Modelo não faz parte do escopo desta Dissertação.

Porém, o detalhamento do caso de uso "Executar Componentes 802.11" se faz necessário, uma vez que processos e requisitos relevantes são identificados nesse caso de uso. Esse detalhamento irá representar o papel de cada elemento na simulação das redes 802.11, e possibilitar a identificação de outros elementos pertinentes, de maneira que os componentes 802.11 possam ser identificados.

A execução dos componentes para a simulação é feita pelo simulador. Com isso, não fica caracterizada a interação externa de um ator com esse caso de uso, conforme pode ser visto na Figura 4.2. Dessa forma, a interação ocorre internamente entre casos de uso, partindo do caso de uso Simular Modelo, que é estimulado pelo analista de modelagem.

Caso de Uso 04	**EXECUTAR COMPONENTES 802.11**
Ator	Interação interna.
Tipo	Primário.
Descrição	Os elementos são executados pelo simulador de acordo com o modelo de rede 802.11 configurado. Cada modelo necessita de um conjunto de elementos. Alguns elementos são obrigatórios em qualquer que seja o modelo: fonte de tráfego, estação origem, camada de acesso ao meio e seu protocolo, enlace, estação destino e sorvedouro.

Caso de Uso 05	**EXECUTAR COMPONENTES 802.11 P/ REDE AD HOC**
Ator	Interação interna.
Tipo	Primário.
Descrição	Os elementos são executados pelo simulador para contemplar a simulação de rede ***ad hoc***: fonte de tráfego, estação origem, camada de acesso ao meio e seu protocolo, enlace, estação destino e sorvedouro. A fonte de tráfego gera o tráfego de informações geradas pelas aplicações que rodam nas estações da rede. Esse tráfego é oriundo da subcamada LLC e é repassado para a camada MAC, para estabelecer a comunicação na rede. Depois disso, a camada MAC executa o protocolo de acesso ao meio (CSMA/CA), que converte as informações em quadros e executa o acesso ao meio. Para a efetiva transmissão desses quadros, o protocolo MAC usa o enlace, que representa o canal de comunicação entre dois elementos na rede. Nesse caso de rede *ad hoc*, os enlaces fazem parte de uma mesma célula, que delimita a área de cobertura da rede. Em seguida, a estação destino recebe o quadro e o repassa para o elemento sorvedouro, que representa o recebimento com sucesso do tráfego da rede.

Caso de Uso 06	**EXECUTAR COMPONENTES 802.11 P/ REDE MULTI-CÉLULA**
Ator	Interação interna.
Tipo	Primário.
Descrição	Os elementos são executados pelo simulador para contemplar a simulação de rede **multi-célula**: fonte de tráfego, estação origem, camada de acesso ao meio e seu protocolo, enlace, ponto de acesso origem, sistema de distribuição, ponto de acesso destino, estação destino e sorvedouro. As mesmas narrações feitas no caso de uso anterior servem para o uso dos elementos ali citados. Porém, nesse tipo de rede multi-célula, considerações adicionais se fazem necessárias: uma vez que a comunicação ocorre entre estações de células distintas, o sistema de distribuição é necessário para possibilitar a interligação das células através dos pontos de acesso. Nesse caso, existe um enlace entre a estação origem e o ponto de acesso origem, entre este e o ponto de acesso destino (que é representado pelo sistema de distribuição), e entre este ponto de acesso e a estação destino, que entregará os quadros ao sorvedouro.

Em quaisquer tipos de simulação de rede 802.11, devem ser coletadas medidas de desempenho nos elementos da rede. Essas medidas visam contemplar o requisito funcional RF11.

Modelos de redes 802.11 onde se configurem várias fontes de tráfego é perfeitamente desejável, e vai depender da configuração imposta pelo analista de modelagem.

4.4 Modelo conceitual

Para elaboração do modelo conceitual, documento de alto nível que ilustra os conceitos significantes do funcionamento do sistema, usamos as seguintes etapas: (1) levantamento dos conceitos; (2) definição dos principais atributos dos conceitos; (3) relacionamentos (associações) entre os conceitos; (4) cardinalidade das relações entre os conceitos e (5) elaboração gráfica do modelo conceitual [Larman98].

O modelo conceitual é o documento mais importante da fase de análise [Larman98], devido às informações que ele oferece e da facilidade do seu entendimento, uma vez que é a partir desse entendimento que o sistema (aqui sendo os componentes 802.11) começa a ter uma representação consistente.

A seguir são apresentados dois modelos conceituais. O primeiro, para representar os conceitos que fazem parte do ambiente de simulação. O segundo, para representar os conceitos do funcionamento das redes 802.11.

4.4.1 Modelo conceitual Ambiente de Simulação

No modelo conceitual da Figura 4.3 ilustramos os principais conceitos necessários para promover uma simulação de rede. Esse modelo é apresentado em [Rocha02] e é colocado aqui para facilitar o entendimento do funcionamento de um ambiente de simulação. Nesse modelo, posicionamos os componentes 802.11 nesse ambiente.

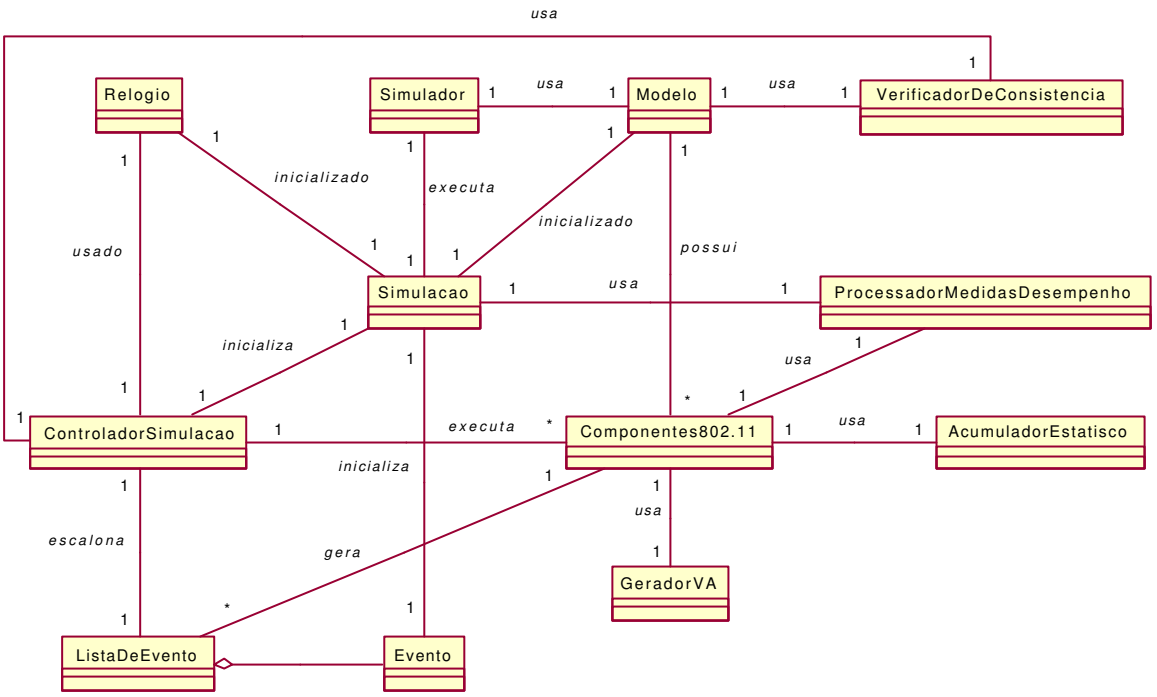

Figura 4.3 – Modelo conceitual Ambiente de Simulação.

Podemos observar que no modelo conceitual da Figura 4.3, os componentes 802.11 se comunicam com os seguintes elementos do ambiente de simulação: Controlador da Simulação, Gerador de Variáveis Aleatórias (GeradorVA), Acumulador Estatístico, Processador de Medidas de Desempenho, Modelo e Lista de Eventos. Esse fato ajuda a entender a solução (projeto) dos componentes 802.11, apresentada no capítulo seguinte.

Vejamos na subseção seguinte o modelo conceitual referente aos conceitos candidatos a componentes 802.11.

4.4.2 Modelo conceitual Candidatos a Componentes 802.11

Adotando as técnicas de levantamento de conceitos propostas em [Larman98] e [Sauvé00], levantamos um conjunto significativo de conceitos, embora nem todos tenham de ser considerados na especificação dos componentes 802.11 (conceitos candidatos).

Um importante reforço pode ser visto nas narrativas dos casos de uso abordadas na seção anterior, que são recursos relevantes para o levantamento de conceitos.

Levantamento dos conceitos

Vejamos a lista de conceitos candidatos a elementos de software (classe, objeto, interface, componentes). Essa lista se apresenta reduzida, apresentando apenas os elementos necessários para representar o funcionamento das redes 802.11.

Comunicação.	Canal de comunicação.	Ad hoc.
Transmissão.	Colisão.	Multi-célula.
Quadro.	Meio de transmissão.	Estação origem.
Aplicação.	Acesso ao meio.	Estação destino.
Tráfego.	Camada de acesso ao meio.	Ponto de acesso origem.
Fonte de tráfego.	Protocolo de acesso ao meio.	Sistema de distribuição.
Célula.	Interligação.	Ponto de acesso destino.
Enlace.	Modelo de rede.	Sorvedouro.

Ilustração do modelo conceitual

De acordo com os conceitos levantados, construímos o modelo conceitual com o intuito de identificar os conceitos candidatos a componentes 802.11, através da ilustração do funcionamento das redes 802.11. Esse modelo conceitual mostra alguns dos atributos dos conceitos, os nomes dos conceitos, as relações entre eles e as multiplicidades dessas relações.

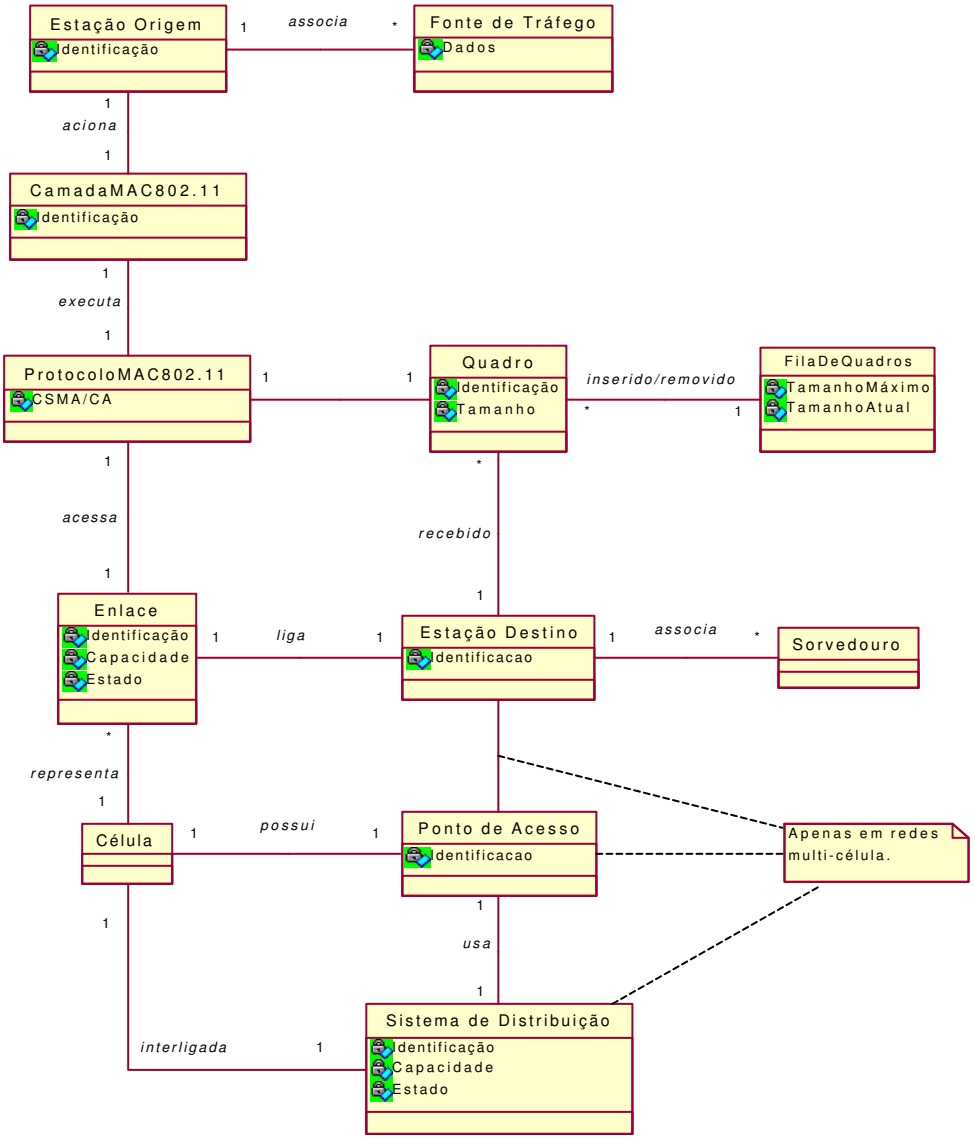

Figura 4.4 – Modelo conceitual Candidatos a Componentes 802.11.

Vejamos uma descrição do modelo conceitual apresentado na Figura 4.4. Algumas descrições já são apresentadas nas narrativas dos casos de uso (seção 4.3). Esse reforço enriquece mais a especificação, uma vez que melhor explicita a documentação aqui apresentada.

- Uma estação origem pode associar várias fontes de tráfego para receber o tráfego de informações originadas das aplicações.
- A estação origem aciona a sua camada de acesso ao meio, para que as informações (quadros) sejam enviadas à estação destino.
- A camada de acesso ao meio envia os quadros pelo enlace (canal de comunicação), executando, para isso, o seu protocolo de acesso ao meio (ProtocoloMAC802.11).
- A transmissão com sucesso de um quadro no enlace, indica que a estação destino recebeu o referido quadro.
- Em seguida ao recebimento de um quadro, a estação destino repassa esse quadro para o elemento sorvedouro, que representa a entrega de quadros na estação destino à aplicação correspondente.

Basicamente a explicação supracitada atende as redes 802.11 *ad hoc*. Para contemplar o tráfego de informações numa rede multi-célula, outros passos são necessários, que indicam a continuação da transferência a partir da transmissão do quadro no enlace:

- A transmissão com sucesso de um quadro no enlace, indica que o ponto de acesso origem recebeu o referido quadro.
- Em seguida, o ponto de acesso origem envia o quadro pelo sistema de distribuição.
- Em seguida, o ponto de acesso destino recebe o quadro.
- O ponto de acesso destino concorre ao acesso ao meio na sua célula, da mesma forma que uma estação origem. Portanto, o ponto de acesso destino aciona a sua camada de acesso ao meio, para que os quadros sejam enviados à estação destino.
- O restante da transferência é semelhante à entrega do quadro na estação destino de uma rede *ad hoc*.

Ressaltamos que essa fase de análise revela questões sobre o entendimento do funcionamento do sistema que se deseja construir. Portanto, os conceitos ilustrados no modelo conceitual da Figura 4.4 são fortes candidatos a componentes de simulação das redes 802.11.

Um conceito é um candidato a ser uma entidade de software (classe, interface, componente). Uma ou mais classes podem constituir um componente. Isso depende da tradução da fase de análise feita pela fase de projeto.

Os modelos conceituais mostrados nas Figuras 4.3 e 4.4 viabilizam a elaboração dos diagramas de seqüência e dos contratos, que são apresentados mais a frente, nas seções 4.6 e 4.7, respectivamente. Esses modelos também auxiliam a fase de projeto, na elaboração dos diagramas de colaboração e do projeto detalhado, conforme já mencionado.

4.5 Identificação dos componentes 802.11

Baseado nos artefatos de análise criados até aqui é possível identificarmos os componentes 802.11 que são projetados no capítulo seguinte. Quais sejam: fonte de tráfego, estação origem, camada de acesso ao meio e seu protocolo, enlace, estação destino, ponto de acesso origem, sistema de distribuição, ponto de acesso destino e sorvedouro.

Vejamos nas Figuras 4.4 e 4.5 a ilustração dos componentes 802.11 comparando com os tipos de redes que eles vão representar: respectivamente redes *ad hoc* e multi-célula.

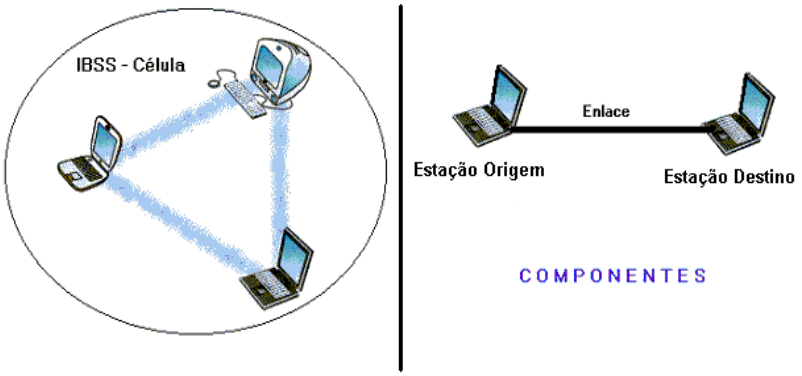

Figura 4.5 – Rede *Ad Hoc* (E) / Componentes *Ad Hoc* (D).

Os componentes fonte de tráfego, camada de acesso ao meio e seu protocolo e sorvedouro, não aparecem nessa visão gráfica, mas são identificados nas narrativas dos casos de uso (seção 4.3) e na elaboração do modelo conceitual (seção 4.4).

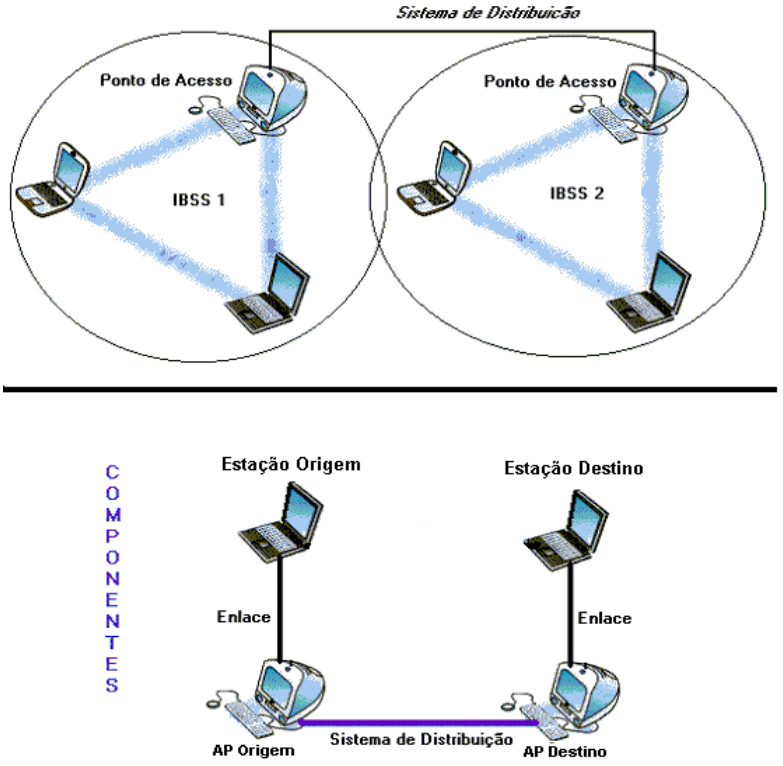

Figura 4.6 – Rede Multi-célula / Componentes Multi-célula.
(parte superior) ***(parte inferior)***

Na rede 802.11 multi-célula, novos componentes aparecem: ponto de acesso origem, sistema de distribuição e ponto de acesso destino, além dos componentes já mencionados na ilustração da rede *ad hoc*.

O conceito "célula", mesmo sendo um elemento fundamental no contexto das redes sem fio, não aparece como componente. Conforme visto no capítulo 3, o que determina a célula é a potência do sinal das estações envolvidas, sendo célula um conceito abstrato que define a área de cobertura da rede sem fio. Na representação de rede através de componentes, o componente célula não fica explícito, uma vez que é contemplado no modelo de rede *ad hoc* através do componente enlace e, em rede multi-célula, através dos componentes enlace, ponto de acesso origem, sistema de distribuição e ponto de acesso destino.

Denominaremos os componentes 802.11 identificados da seguinte forma: FonteTrafego, EstacaoOrigem, CamadaMAC802.11, Enlace, EstacaoDestino, PontoAcessoOrigem, SistemaDistribuicao, PontoAcessoDestino e Sorvedouro. Os artefatos de análise apresentados nas seções seguintes (diagramas de seqüência e contratos) já tratam esses elementos com essa nomenclatura.

4.6 Diagrama de seqüência

4.6.1 Introdução

Uma vez que o modelo conceitual não é suficiente para entendermos o funcionamento completo do sistema, principalmente com relação a sua dinâmica, é necessária a utilização de outros recursos da fase de análise. O diagrama de seqüência se apresenta como um elemento que auxilia nessa compreensão, detalhando as funcionalidades e revelando a dinâmica do sistema [Larman98].

Em sistemas grandes e complexos, muitas vezes é necessária a elaboração de um diagrama de seqüência para cada funcionalidade do sistema. No caso dos componentes 802.11, elaboramos os diagramas de seqüência para identificar os eventos dentro do sistema, ou seja, os eventos entre os elementos 802.11, quando da execução de uma simulação.

Conforme visto no modelo conceitual da Figura 4.3 (modelo conceitual Ambiente de Simulação), os componentes 802.11 interagem freqüentemente com os componentes do ambiente de simulação, quando da execução da simulação. Ignoramos esse fato nos diagramas de seqüência que se seguem, para que o foco seja exclusivo nos elementos das redes 802.11.

O diagrama completo com todos os eventos e componentes (inclusive os componentes 802.11) de um ambiente de simulação é apresentado no capítulo seguinte.

4.6.2 Diagrama de seqüência Rede *Ad Hoc* Sem Colisão

O diagrama de seqüência da Figura 4.7 ilustra os eventos na simulação de uma rede *ad hoc* sem a ocorrência de colisão.

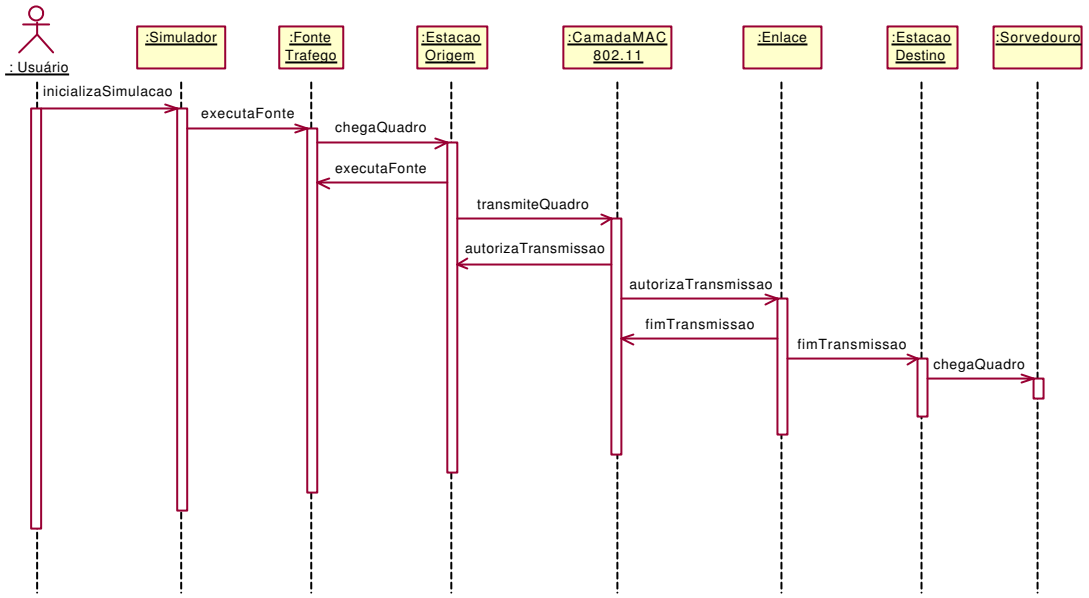

Figura 4.7 – Diagrama de seqüência Rede *Ad Hoc* Sem Colisão.

Segue a explicação do diagrama de seqüência da Figura 4.7:

- O primeiro evento aciona a FonteTrafego para a geração do tráfego na rede.
- A FonteTrafego aciona a EstacaoOrigem para o recebimento do tráfego, através do evento *chegaQuadro*.
- O recebimento do evento *chegaQuadro* já possibilita a EstacaoOrigem acionar novamente a FonteTrafego, para promover a auto-geração de quadros.
- Em seguida, a EstacaoOrigem gera o evento *transmiteQuadro*, indicando a sua CamadaMAC802.11 que tem quadro para transmitir.
- A CamadaMAC802.11 gera o evento *autorizaTransmissao* que é observado simultaneamente pela EstacaoOrigem, para transmitir o quadro, e pelo Enlace, para ser liberado, caso haja uma outra transmissão a ser feita.
- Em seguida, o Enlace gera o evento *fimTransmissao*, que é observado simultaneamente pela CamadaMAC802.11, para iniciar outra transmissão, e pela EstacaoDestino, para receber o quadro.
- Finalmente, a EstacaoDestino gera o evento *chegaQuadro*, que aciona o Sorvedouro para a conclusão do recebimento do quadro por parte da aplicação correspondente.

4.6.3 Diagrama de seqüência Rede *Ad Hoc* Com Colisão

O diagrama de seqüência da Figura 4.8 ilustra os eventos na simulação de uma rede *ad hoc* na ocorrência de colisão.

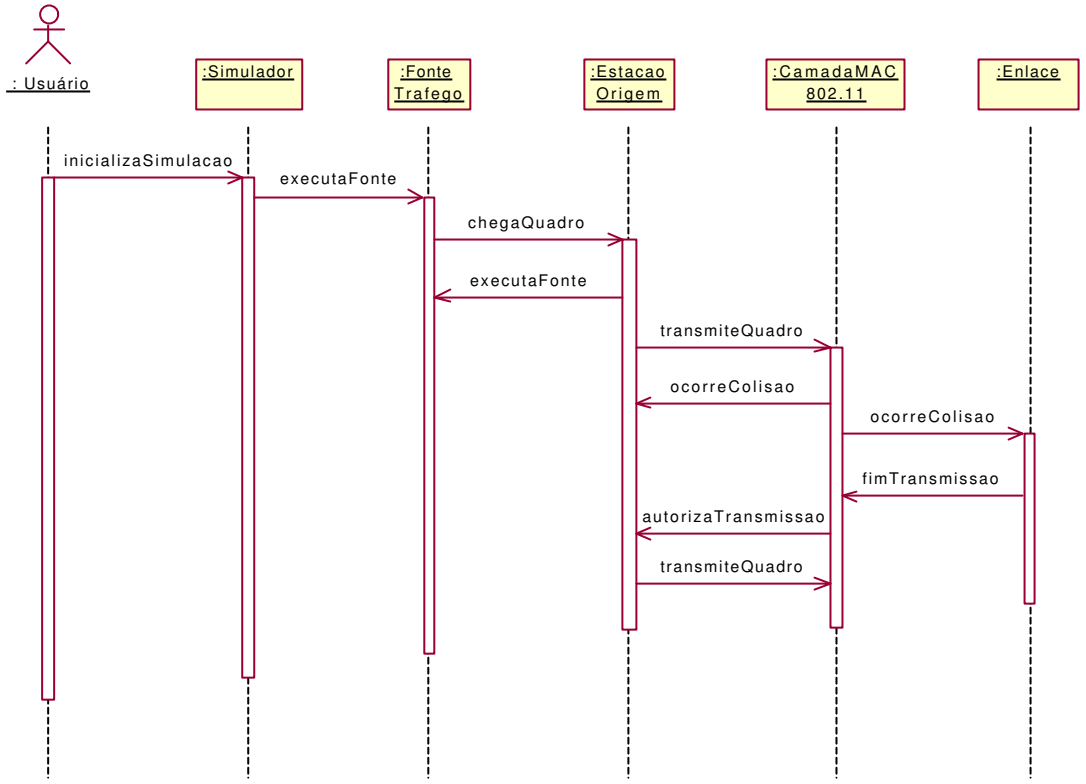

Figura 4.8 – Diagrama de seqüência Rede *Ad Hoc* Com Colisão.

Até a ocorrência de colisão, a simulação ocorre de forma semelhante ao diagrama da Figura 4.7. Na ocorrência de colisão, a dinâmica da simulação apresenta as seguintes mudanças:

- A EstacaoOrigem gera o evento *transmiteQuadro*, indicando a CamadaMAC802.11 que tem quadro para transmitir. Esta camada verifica a seguir se outra estação nesse mesmo instante também gerou o evento *transmiteQuadro*.
- Caso duas EstacaoOrigem tenham gerado o evento *transmiteQuadro* no mesmo instante, a CamadaMAC802.11 gera o evento *ocorreColisao*, que é observado pelas estações que lhe enviaram os quadros, para que os quadros sejam retransmitidos, e pelo Enlace, para indicar a ocorrência de colisão, modelando a ocupação do canal de comunicação.
- Em seguida, a CamadaMAC802.11 gera o evento *autorizaTransmissao*, e a simulação prossegue conforme diagrama de seqüência da Figura 4.7.

4.6.4 Diagrama de seqüência Rede Multi-Célula

O diagrama de seqüência da Figura 4.9 ilustra os eventos na simulação de uma rede multi-célula.

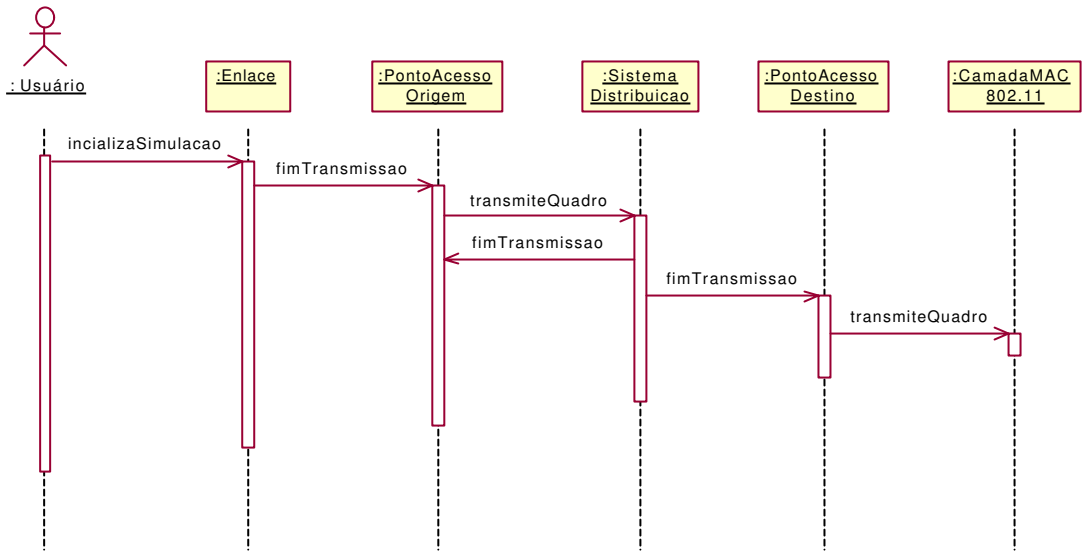

Figura 4.9 – Diagrama de seqüência Rede Multi-Célula.

Até que o Enlace seja acessado, a simulação de uma rede multi-célula ocorre de forma semelhante à de uma rede *ad hoc*, conforme diagrama de seqüência da Figura 4.7. Em seguida, a dinâmica da simulação apresenta as seguintes mudanças:

- O Enlace gera o evento *fimTransmissao,* que aciona o PontoAcessoOrigem.
- Em seguida, o PontoAcessoOrigem gera o evento *transmiteQuadro,* que aciona o SistemaDistribuicao.
- O SistemaDistribuicao gera o evento *fimTransmissao*, que é observado pelo PontoAcessoOrigem, para iniciar outra transmissão, e pelo PontoAcessoDestino, para receber o quadro.
- O PontoAcessoDestino gera o evento *transmiteQuadro,* que aciona a CamadaMAC802.11. Em seguida, a simulação prossegue conforme mostrado no diagrama da Figura 4.7, quando este evento ocorre.

Na dinâmica de simulação, envolvendo os componentes 802.11, ilustrada nos diagramas de seqüência das Figuras 4.7, 4.8 e 49, apenas os eventos existentes entre os componentes é que são revelados. Para completar o entendimento do sistema numa visão de análise, ou seja, para entender *o que* o sistema deve fazer, é preciso identificar também as operações que vão contemplar os eventos levantados nos diagramas de seqüência.

Os contratos das operações do sistema auxiliam nessa compreensão, recurso apresentado na próxima seção.

4.7 Contratos das operações

Baseado nos diagramas de seqüência apresentados na seção anterior, apresentamos os contratos das operações do sistema, com a finalidade de identificar as respostas que o sistema (componentes 802.11) vai fornecer na ocorrência dos eventos que promovem a simulação das redes 802.11.

Segundo [Larman98] um contrato oferece um conjunto de itens que são usados de acordo com o sistema. Existem itens que são obrigatórios para todos os sistemas. Para atender o funcionamento das redes 802.11, usamos os itens que se seguem. A definição desses itens pode ser vista no apêndice A.

- Nome.
- Responsabilidade.
- Pré-condições.
- Pós-condições.

4.7.1 Contratos dos eventos dos componentes 802.11

Apresentamos a seguir os contratos dos componentes 802.11. Mostramos cada evento num único contrato, uma vez que um mesmo evento pode acionar vários componentes em diversos modelos de rede 802.11.

Os contratos além de registrar as operações que são realizadas na ocorrência de um evento, eles reforçam a explicação sobre a dinâmica da simulação envolvendo os componentes 802.11, apresentada na seção anterior.

Contrato do evento *executaFonte*

Item	Descrição
Nome	executaFonte.
Responsabilidade	Gerar o tráfego na rede.
Pré-condições	A FonteTrafego deve estar inicializada.
Pós-condições	• A FonteTrafego é associada a uma EstacaoOrigem. • A FonteTrafego processa o quadro e gera o evento *chegaQuadro* para a EstacaoOrigem associada.

Contrato do evento *chegaQuadro*

Item	Descrição
Nome	chegaQuadro.
Responsabilidade	Indicar a algum elemento da rede que ele tem um quadro para receber.
Pré-condições	Esse evento deve ser escutado pela EstacaoOrigem e pelo Sorvedouro.
Pós-condições	• A EstacaoOrigem processa o quadro e gera o evento *transmiteQuadro* para CamadaMAC802.11. • O Sorvedouro recebe o quadro.

Contrato do evento *transmiteQuadro*

Item	Descrição
Nome	transmiteQuadro.
Responsabilidade	Indicar a algum elemento da rede que ele tem um quadro a transmitir.
Pré-condições	Esse evento deve ser escutado pela CamadaMAC802.11 e pelo SistemaDistribuicao.
Pós-condições	• A CamadaMAC802.11 processa o quadro e gera o evento *autorizaTransmissao* ou *ocorreColisao*, dependendo do estado do Enlace e do critério para ocorrência de colisão. • O SistemaDistribuicao processa o quadro e gera o evento *fimTransmissao* para a EstacaoOrigem e para o PontoAcessoDestino.

Contrato do evento *ocorreColisao*

Item	Descrição
Nome	ocorreColisao.
Responsabilidade	Indicar as EstacaoOrigem que houve colisão de seus quadros.
Pré-condições	Duas ou mais EstacaoOrigem devem ter gerado ao mesmo tempo o evento *transmiteQuadro*.
Pós-condições	A EstacaoOrigem processa o quadro e gera o evento *fimTransmissao* e envia ao Enlace para liberação do mesmo.

Contrato do evento *autorizaTransmissao*

Item	Descrição
Nome	autorizaTransmissao.
Responsabilidade	Indicar a algum elemento da rede que seu quadro pode ser transmitido.
Pré-condições	Esse evento deve ser escutado pela EstacaoOrigem e pelo Enlace.
Pós-condições	A EstacaoOrigem e o Enlace processam o quadro e geram o evento *fimTransmissao*.

Contrato do evento *fimTransmissao*

Item	Descrição
Nome	fimTransmissao.
Responsabilidade	Indicar aos elementos da rede que houve a transmissão de um quadro.
Pré-condições	Esse evento deve ser escutado pela CamadaMAC802.11, pela EstacaoDestino, pelo PontoAcessoOrigem e pelo PontoAcessoDestino.
Pós-condições	• A CamadaMAC802.11 processa o quadro e gera o evento *autorizaTransmissao* para EstacaoOrigem e para o Enlace. • A EstacaoDestino processa o quadro e gera o evento *chegaQuadro* para o Sorvedouro. • O PontoAcessoOrigem processa o quadro e gera o evento *transmiteQuadro* para o SistemaDistribuicao. • O PontoAcessoDestino processa o quadro e gera o evento *transmiteQuadro* para a CamadaMAC802.11.

Os contratos vistos nesta seção tratam exclusivamente dos eventos internos envolvendo os componentes 802.11. Nesse caso, as operações especificadas nos contratos revelam o que os componentes devem fazer para executar a simulação de redes 802.11. Essas operações são consideradas na elaboração dos diagramas de colaboração, apresentados no próximo capítulo.

4.8 Resumo da fase de análise da especificação

A fase de análise aborda questões referentes ao que deve ser construído, ou seja, qual sistema construir, levantando os elementos que fazem parte do domínio do problema. Este capítulo investiu numa análise capaz de revelar as informações fundamentais para a solução (projeto) da especificação dos componentes 802.11.

Inicialmente identificamos o sistema, seu usuário e os requisitos funcionais e não funcionais considerados. Depois, utilizamos a técnica de caso de uso para levantar e ilustrar as funcionalidades do sistema. Neste momento, o destaque foi a descrição dos casos de uso que dizem respeito ao funcionamento das redes 802.11.

Em seguida, abordamos o modelo conceitual, onde identificamos os conceitos candidatos a componentes 802.11, e apresentamos uma visão geral do funcionamento das redes 802.11 e do ambiente de simulação.

Com os artefatos de análise criados até o modelo conceitual, foi possível identificar os componentes 802.11, que receberam a seguinte nomenclatura: FonteTrafego, EstacaoOrigem, CamadaMAC802.11, Enlace, EstacaoDestino, PontoAcessoOrigem, SistemaDistribuicao, PontoAcessoDestino e Sorvedouro.

Para ilustrar a dinâmica do sistema, elaboramos os diagramas de seqüência, revelando os eventos e elementos (agora vistos como componentes) necessários para promover a simulação das redes 802.11. Finalizando, elaboramos os contratos referentes aos eventos mostrados nos diagramas de seqüência.

No capítulo seguinte apresentamos a solução (projeto) para a especificação dos componentes 802.11 aqui identificados.

Capítulo 05

Fase de Projeto

5.1 Introdução

Neste capítulo apresentamos a especificação dos componentes para a modelagem das redes locais de computadores sem fio padrão 802.11, focando a fase de projeto (descrição da solução). Nessa fase, recursos do processo de desenvolvimento propostos em [Larman98] são usados.

Inicialmente mostramos, através de um diagrama de seqüência, a dinâmica do ambiente de simulação no qual se inserem os componentes 802.11 especificados. Em seguida, apresentamos o modelo de componente adotado: modelo JavaBeans. Em seguida, apresentamos diagramas de colaboração focando as interações entre os componentes 802.11 e os componentes do ambiente de simulação. Por fim, apresentamos o projeto arquitetural de alto nível e o projeto detalhado dos componentes 802.11.

5.2 Diagrama de seqüência Ambiente de Simulação

Vimos na fase de análise que os modelos conceituais elaborados abordam o ambiente de simulação e os componentes 802.11. Para o entendimento completo desses componentes é preciso conhecer a dinâmica de um ambiente de simulação no qual os componentes 802.11 estão inseridos. Essa dinâmica pode ser ilustrada através de um diagrama de seqüência, conforme mostrado na Figura 5.1.

O diagrama de seqüência é considerado um artefato de análise, mas é colocado aqui para facilitar o entendimento dos demais artefatos de projeto usados na especificação dos componentes 802.11, uma vez que esse diagrama também mostra que os componentes 802.11 interagem com diversos componentes de um ambiente de simulação, através de eventos. Os componentes de um ambiente de simulação são especificados em [Lula01].

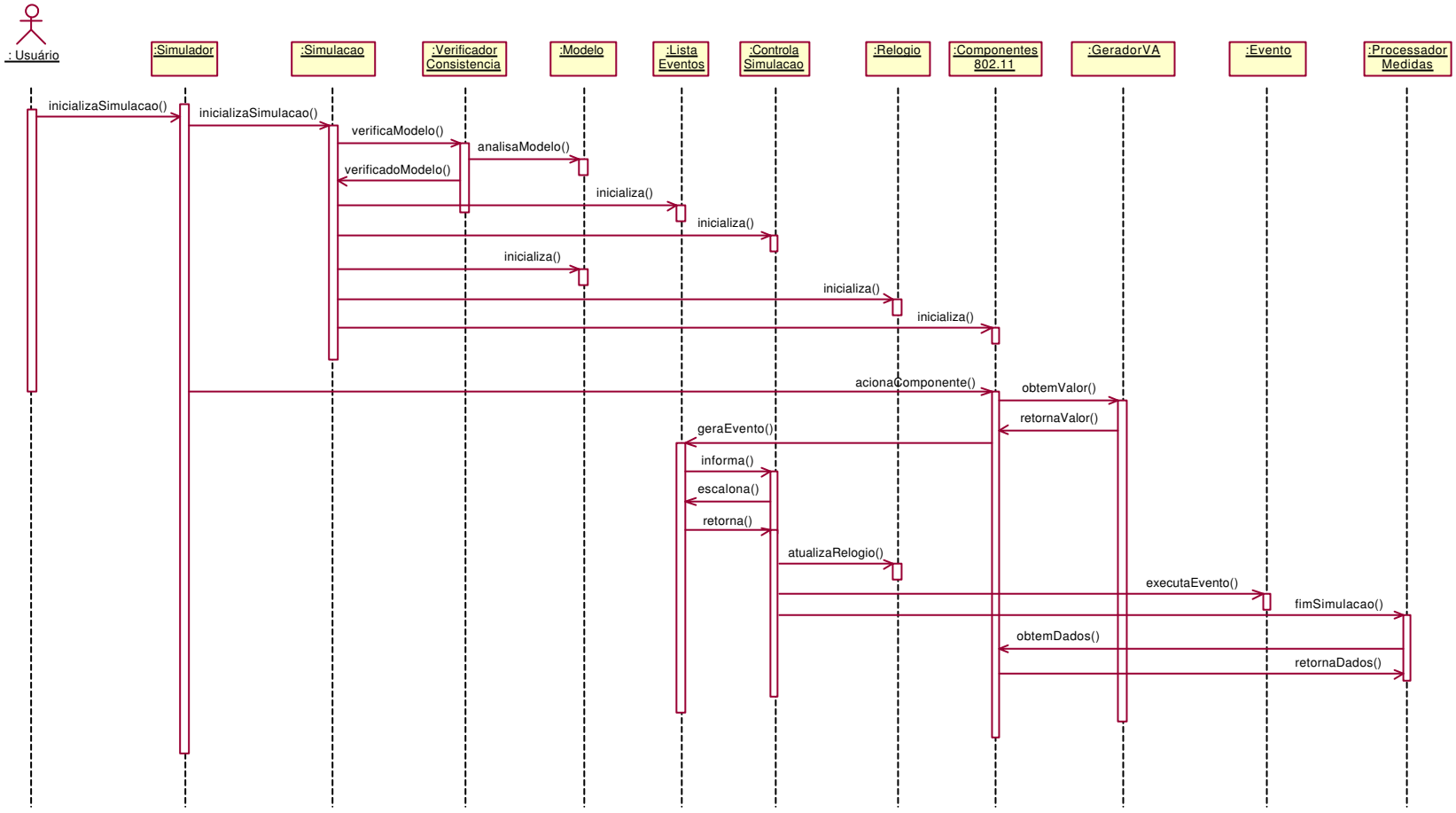

Figura 5.1 – Digrama de seqüência Ambiente de Simulação [Rocha02] e [Lula01].

Segue a explicação do diagrama de seqüência da Figura 5.1:

- O simulador inicializa a simulação através do evento *inicializaSimulacao* enviado para o componente Simulacao.
- O componente Simulacao aciona o componente VerificadorConsistencia através do evento *verificaModelo*. O componente VerificadorConsistencia analisa o modelo gerando o evento *analisaModelo*. Em seguida, esse componente gera o evento *verificadoModelo* para informar ao componente Simulacao da validação ou não do modelo.
- Em seguida, o componente Simulacao gera o evento *inicializa,* que inicializa os seguintes componentes:
  - ✓ ListaEventos – para ativar a lista de eventos usada na simulação.
  - ✓ ControlaSimulacao – para escalonar os eventos e acionar componentes durante a simulação.
  - ✓ Modelo – para carregar no ambiente de simulação o modelo de rede que será simulado.
  - ✓ Relogio – para iniciar o tempo da simulação.
  - ✓ Componentes802.11 – para carregar no ambiente os componentes de rede que são usados na simulação, ou seja, que constituem o modelo de rede que é simulado.
- Em seguida a inicialização dos componentes supracitados, o simulador aciona os componentes 802.11, através de eventos pertinentes aos modelos de rede 802.11 (*executaFonteEvent, chegaQuadroEvent, fimTransmissaoEvent*).
- Sendo acionado por um evento, o componente 802.11 interage com o componente GeradorVA, para obter uma amostra necessária para o cálculo do tempo do seu processamento.
- Depois de interagir com o componente GeradorVA, o componente 802.11 gera um evento que é adicionado ao componente ListaEventos.
- O componente ListaEventos interage com o componente ControlaSimulacao para informar o evento que foi escalonado.
- O componente ControlaSimulacao interage com o componente Relogio para a atualização do tempo da simulação.

- O componente ControlaSimulacao também é responsável pelo acionamento da simulação. Para isso, este componente interage diretamente com o componente Evento.
- O evento final considerado é *fimSimulacao* que, quando escalonado, aciona o componente ProcessadorMedidasDesempenho.
- Por fim, o componente ProcessadorMedidasDesempenho, obtém dos componentes 802.11 os dados coletados durante a simulação. Esses dados permitem o cálculo das medidas de desempenho de interesse.

5.3 Modelo de componentes

A especificação proposta nesta Dissertação investe na especificação de componentes reutilizáveis de software. "Componente é um pacote coerente de artefato de software que pode ser desenvolvido e disponibilizado de forma independente e autônoma, e que pode facilmente ser acoplado a outros componentes para construir algo maior" [D'Souza98].

A definição de [D'Souza98] sobre componentes revela conceitos da Engenharia de Software sobre flexibilidade, reusabilidade, modularidade e portabilidade. Com isso, podemos identificar componentes como uma solução eficiente na indústria de software.

Numa visão mais técnica, um componente é uma classe de software ou um conjunto de classes, que é acessado através de uma interface construída com regras específicas [Booch98]. Essa definição confirma que um processo de desenvolvimento orientado a objetos pode ser usado na construção de componentes, desde que a abordagem seja direcionada a componentes.

Um outro aspecto que deve ser considerado é o modelo de componente a ser adotado. Muitos modelos são disponibilizados, mas ainda não há uma padronização entre eles [Kobryn00]. Três modelos bastante conhecidos são EJV (*Enterprise JavaBeans*) da Sun Microsystems, CORBA (*Common Object Request Broker*) da Object Management Group, e COM (*Component Object Model*) da Microsoft.

O modelo adotado nesta Dissertação é o JavaBeans. O modelo JavaBeans é composto de uma arquitetura e de uma API (*Application Programing Interface*) [Englander97], e

apresenta um conjunto de regras e diretrizes que deve ser seguido pelos seus componentes, doravante denominados "Beans". O modelo JavaBeans fornece uma estrutura na qual os componentes Beans podem ser combinados para criar uma aplicação.

O modelo JavaBeans possui as seguintes características [Lula01]:

- **Descoberta e registro** – processo que possibilita que um componente possa ser conhecido no sistema (aplicação).
- **Criação e tratamento de eventos** – geração de eventos para que as ações sejam executadas no sistema.
- **Persistência** – mecanismo de armazenamento das informações dos componentes.
- **Representação visual** – define a forma como os componentes são exibidos no sistema, geralmente baseada nas suas propriedades.

A linguagem Java levou o conceito de componente ao mais alto grau de simplicidade possível. Um componente JavaBeans é simplesmente uma classe que deve apenas modificar o modo de nomear seus métodos [Eckel00]. É o nome do método que define se uma operação deve ser realizada por um Bean ou se é um método comum, instanciando uma classe de software, por exemplo.

O modelo de componentes JavaBeans suporta a comunicação de componentes baseada em eventos [Neto01], ou seja, os Beans se comunicam através da troca de eventos. Um evento é uma perturbação que ocorre num dado instante para que uma ação seja executada por um elemento (componente) do sistema [Neto01].

Nesse modelo, os conceitos de produtor e consumidor são considerados. Componentes produtores (*source*) geram eventos que são escutados por componente consumidores (*listener*). Esse modelo segue o padrão de projeto *observer* [GoF95], que faz com que um componente consumidor se cadastre a um componente produtor. Com isso, quando um componente muda de estado, todos os componentes cadastrados a ele são notificados e atualizados automaticamente, ou seja, quando um evento ocorre, os interessados nesse evento são notificados instantaneamente.

Em JavaBeans, usamos os seguintes termos no início do nome do evento, que são considerados quando da elaboração do projeto detalhado, apresentado na subseção 5.6.2:

- **add** – para cadastrar no componente A (consumidor – listener) eventos do componente B (produtor – source).
- **remove** – para remover no componente A (consumidor – listener) eventos do componente B (produtor – source).
- **fire** – para indicar que é um método Bean e não um método comum de classes. Os eventos iniciados com *fire* ativam os métodos de um Bean.

Os diagramas de colaboração que seguem revelam os eventos necessários para a simulação dos modelos de rede 802.11. No modelo JavaBeans, um evento é identificado pela terminação *Event*. Interações que não possuem essa terminação são apenas mensagens entre classes de software.

5.4 Diagramas de colaboração

Os *diagramas de interações* ilustram as interações entre os elementos de software. Uma vez que os componentes de alto nível já são definidos, é importante conhecer a comunicação entre eles, através de eventos que promovem essa comunicação.

O funcionamento das redes 802.11 mostrado no modelo conceitual, as interações reveladas nos diagramas de seqüências e os itens *responsabilidade* e *pós-condições* mostrados nos contratos, ajudam na elaboração dos digramas de interações, tornando mais clara a forma de comunicação dos componentes de software especificados [Larman98].

Diagramas de seqüência e diagramas de colaboração são exemplos de diagramas de interações [Larman98]. O diagrama de seqüência, já usado na fase de análise, foi mostrado na seção 5.2 para ilustrar a dinâmica de funcionamento do ambiente de simulação (Figura 5.1). Nesta fase de projeto (solução do sistema), usamos diagramas de colaboração, definindo a comunicação entre componentes na modelagem das redes 802.11. Esses diagramas ilustram os componentes, os eventos e a lógica de funcionamento para permitir a simulação de modelos de rede 802.11.

Como em UML nenhum artefato é definido para representar um componente (geralmente é usado o artefato *package* para isso), usamos a mesma representação de classe, adotando que os elementos representados pelos retângulos são componentes e não classes ou objetos. Da mesma forma, consideramos eventos as mensagens que acionam os componentes.

Nos diagramas de colaboração mostrados adiante fica explicitado que os componentes 802.11 interagem diretamente com diversos componentes do ambiente de simulação: GeradorVA, Relogio, ControlaSimulacao, ListaEventos e ProcessadorMedidasDesempenho.

Os componentes do ambiente de simulação ControlaSimulacao e ListaEventos sempre iniciam esses diagramas. O componente ControlaSimulacao gera o evento *escalonaListaEventosEvent*, o qual aciona o componente ListaEventos que escalona o evento iminente (aquele que apresenta menor tempo de ocorrência).

O evento iminente é informado ao componente ControlaSimulacao (geração do evento *retornaEventoEvent*). O componente ControlaSimulacao aciona o evento iminente (nesta Dissertação, doravante denominado evento corrente). Os diagramas de colaboração apresentados a seguir dependem do evento corrente focado e do componente que ele aciona.

Inicialmente, apresentamos os diagramas de colaboração de uma rede *ad hoc*, com os elementos mínimos necessários para simulação desse tipo de rede (diagramas de colaboração das Figuras 5.2 a 5.11) e, indicamos, quando oportuno, os eventos e componentes que fazem parte das redes multi-célula, a título de citação e esclarecimento de que alguns eventos e componentes também fazem parte da simulação dessas redes. Em seguida, apresentamos os diagramas de colaboração específicos das redes multi-célula (diagramas de colaboração das Figuras 5.12 a 5.15).

Os eventos primários que promovem a simulação das redes 802.11 são os seguintes:

- *executaFonteEvent.*
- *chegaQuadroEvent.*
- *fimTransmissaoEvent.*

A ocorrência de um desses eventos pode desencadear a geração de outros eventos, sejam primários ou secundários. Os eventos secundários dependem da ocorrência de um evento primário. Os eventos secundários considerados neste projeto são os seguintes:

- *transmiteQuadroEvent.*
- *autorizaTransmissaoEvent.*
- *ocorreColisaoEvent.*

5.4.1 Diagrama de colaboração *ExecutaFonteEvent* (FonteTrafego)

O diagrama de colaboração da Figura 5.2 ilustra os componentes de simulação e eventos gerados quando da ocorrência do evento *ExecutaFonteEvent,* que aciona o componente FonteTrafego.

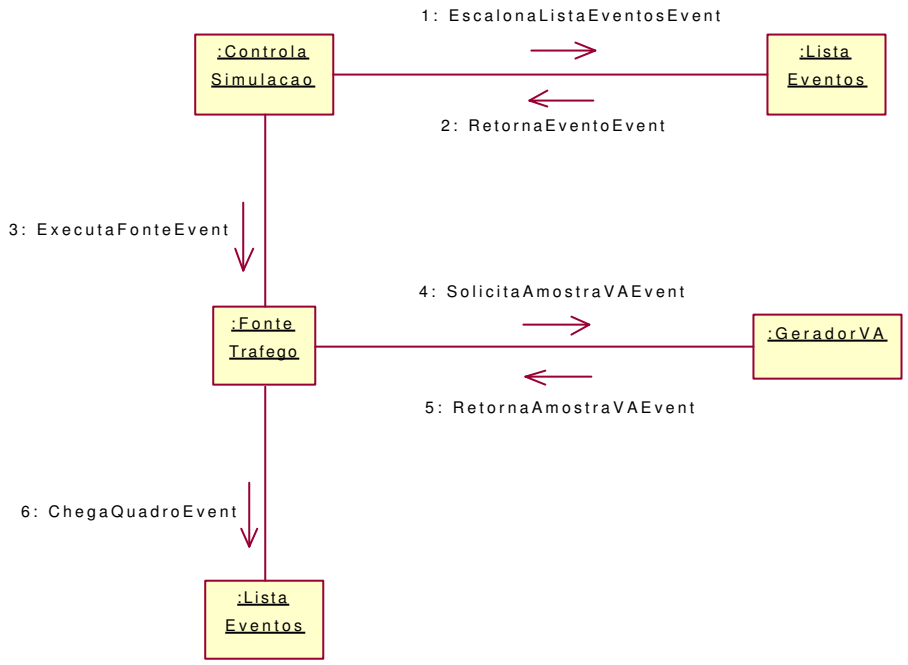

Figura 5.2 – Diagrama de colaboração *ExecutaFonteEvent* (FonteTrafego).

No diagrama de colaboração da Figura 5.2 a ocorrência do evento *ExecutaFonteEvent* inicia o processo de geração de quadros numa fonte de tráfego do modelo construído. Este evento também pode ser gerado pelo componente EstacaoOrigem, que aciona o processo de auto-geração de quadros, abordado na próxima subseção.

O componente FonteTrafego interage com o componente GeradorVA para obter o comprimento do quadro a ser gerado, caso este comprimento não seja constante.

O componente FonteTrafego interage novamente com o componente GeradorVA para obter uma amostra do tempo de interchegada do quadro a ser gerado, caso este tempo não seja constante. Essa amostra de tempo é somada ao tempo de ocorrência do evento corrente (*ExecutaFonteEvent*), para encontrar o tempo de ocorrência do evento *ChegaQuadroEvent*. Este evento é adicionado a lista de eventos (componente ListaEventos).

5.4.2 Diagrama de colaboração *ChegaQuadroEvent* (EstacaoOrigem)

No diagrama de colaboração da Figura 5.3 focamos o evento *ChegaQuadroEvent* que aciona o componente EstacaoOrigem.

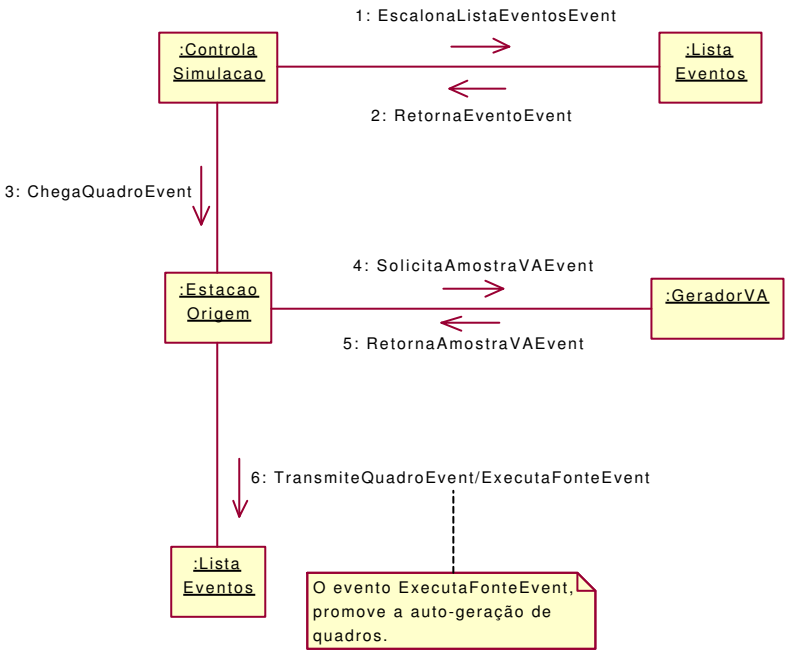

Figura 5.3 – Diagrama de colaboração *ChegaQuadroEvent* (EstacaoOrigem).

Além do componente EstacaoOrigem, o evento *ChegaQuadroEvent* pode acionar o componente Sorvedouro (subseção 5.4.10).

Quando o componente EstacaoOrigem é acionado pelo evento *ChegaQuadroEvent*, este componente inicialmente gera um evento *ExecutaFonteEvent*, modelando assim o processo de auto-geração de quadros. O tempo de ocorrência do evento *ExecutaFonteEvent* é o mesmo do evento corrente (*ChegaQuadroEvent*).

O componente EstacaoOrigem armazena os quadros recebidos numa fila. Seguem considerações sobre a modelagem dessa fila, conforme as condições de funcionamento de uma rede 802.11, abordadas no capítulo 3, quais sejam:

- Na fila ficam os quadros que chegam e que esperam pela liberação do enlace.
- A disciplina de atendimento da fila é FIFO (*First In First Out*), ou seja, o primeiro quadro que chega na fila é o primeiro a ser transmitido.
- A fila só é criada, devido o acúmulo de mais de um quadro recebido na estação.
- Quando a fila tem a apenas um quadro, este quadro está sendo transmitido (enlace ocupado).

O componente EstacaoOrigem interage com o componente GeradorVA para encontrar uma amostra do tempo de processamento do quadro nesta estação. Em seguida, o comprimento de fila é verificado, de acordo com as seguintes considerações:

- Se o comprimento de fila for maior que zero (enlace ocupado), o quadro é colocado em fila, e o comprimento de fila é acrescido de 1 (um).
- Se o comprimento de fila for 0 (zero), significa que o quadro que chega pode ser imediatamente transmitido, se o enlace estiver livre. Neste caso, o comprimento de fila é igual a 1 (um).

Na situação em que a fila de quadros é maior ou igual a 1 (um), um evento secundário chamado *TransmiteQuadroEvent* é gerado e enviado ao ambiente de simulação (para o componente ListaEventos). O tempo de ocorrência do evento *TransmiteQuadroEvent* é calculado somando o tempo do evento corrente (*ChegaQuadroEvent*) com a amostra do tempo de processamento do quadro na estação origem.

Quando o evento *ChegaQuadroEvent* ocorre e o comprimento da fila está no valor máximo, o quadro que chega é descartado. O componente EstacaoOrigem incrementa um contador, para guardar o número total de quadros descartados.

5.4.3 Diagrama de colaboração *TransmiteQuadroEvent* (CamadaMAC802.11)

A Figura 5.4 foca o evento *TransmiteQuadroEvent* que aciona o componente CamadaMAC802.11.

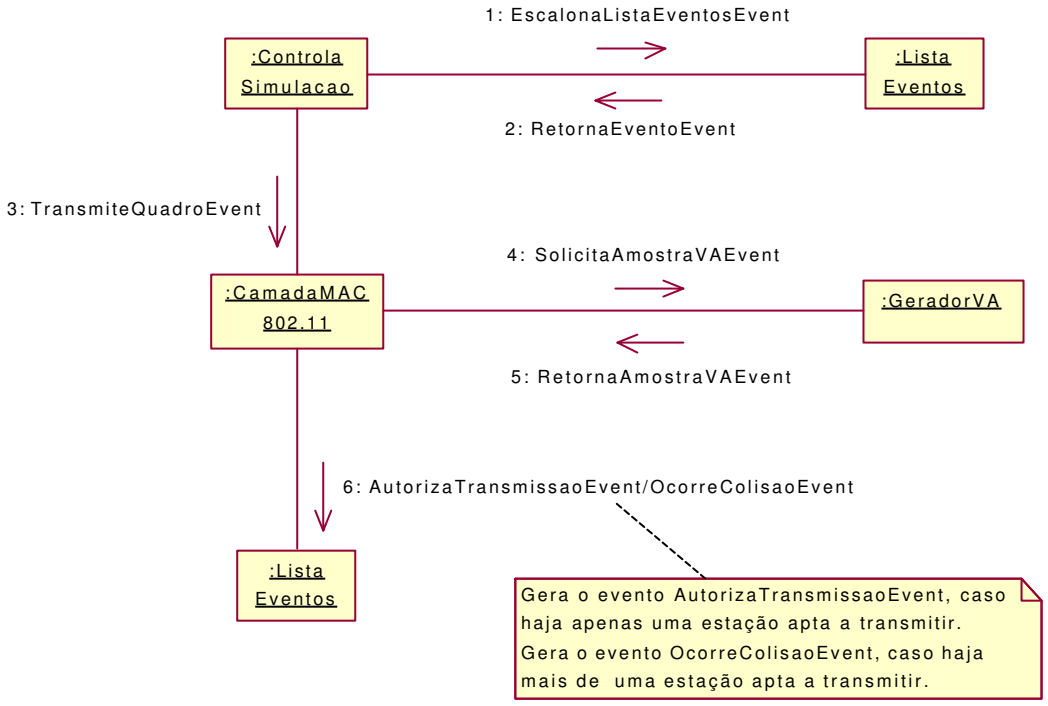

Figura 5.4 – Diagrama de colaboração *TransmiteQuadroEvent* (CamadaMAC802.11).

O componente CamadaMAC802.11 interage com o componente GeradorVA para encontrar uma amostra do tempo de processamento local – decisão de qual estação origem deve transmitir, em caso de não haver colisão, ou quais estações devem transmitir, caso contrário.

Inicialmente, consideramos uma transmissão em que não há colisão. O componente CamadaMAC802.11 gera o evento *AutorizaTransmissaoEvent* e o envia para o componente ListaEventos. Este evento informa sobre a estação origem que está autorizada a enviar o quadro. Neste caso, só existe apenas uma estação apta a transmitir.

O evento *AutorizaTransmissaoEvent* aciona os componentes EstacaoOrigem e Enlace (e o componente PontoAcessoDestino, em caso de redes multi-célula). Somente o componente que corresponde à estação origem autorizada a transmitir é acionado.

Quando mais de uma estação tem condições de transmitir, caracteriza a ocorrência de colisão. Nesse caso, o componente CamadaMAC802.11 gera o evento *OcorreColisaoEvent* que aciona os componentes EstacaoOrigem que colidiram e o componente Enlace. Os diagramas de colaboração focando esse evento são apresentados a seguir.

5.4.4 Diagrama de colaboração *OcorreColisaoEvent* (EstacaoOrigem)

O diagrama de colaboração da Figura 55 foca o evento *OcorreColisaoEvent* que aciona o componente EstacaoOrigem.

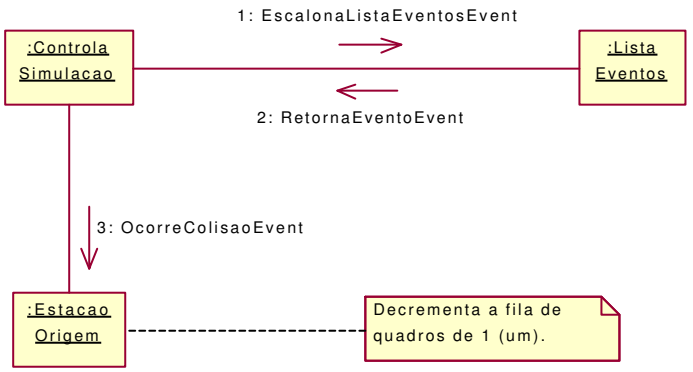

Figura 5.5 – Diagrama de colaboração *OcorreColisaoEvent* (EstacaoOrigem).

Na ocorrência desse evento, os componentes EstacaoOrigem decrementam o comprimento de fila de 1 (um), ou seja, retiram de suas filas o quadro transmitido com colisão.

5.4.5 Diagrama de colaboração *OcorreColisaoEvent* (Enlace)

O evento *OcorreColisaoEvent* também aciona o componente Enlace que modela à transmissão dos quadros colididos, conforme mostrado no diagrama de colaboração da Figura 5.6.

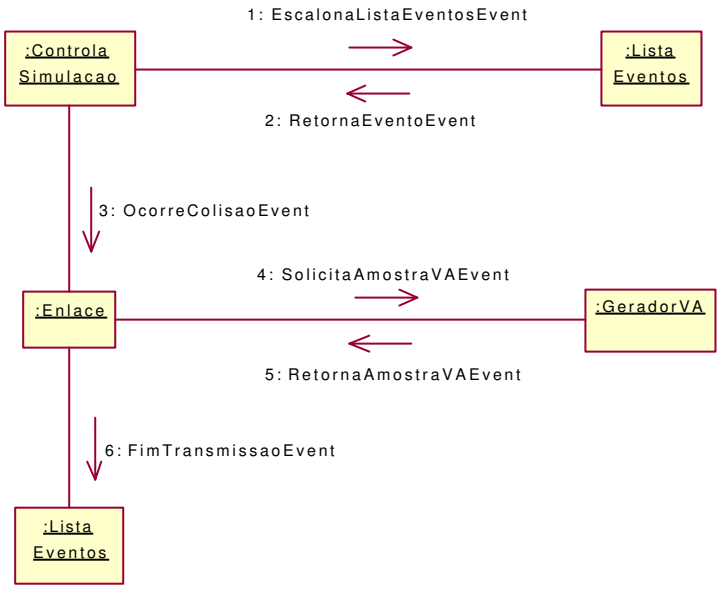

Figura 5.6 – Diagrama de colaboração *OcorreColisaoEvent* (Enlace).

O componente Enlace interage com o componente GeradorVA para obter uma amostra do tempo de transmissão dos quadros colididos. Essa amostra de tempo é somada ao tempo do evento corrente (*OcorreColisaoEvent*), para obter o tempo de ocorrência do evento *FimTransmissaoEvent*. Dessa forma, o componente Enlace gera o evento *FimTransmissaoEvent* que é enviado ao componente ListaEventos.

5.4.6 Diagrama de colaboração *AutorizaTransmissaoEvent* (EstacaoOrigem)

Quando não há colisão, a estação origem escolhida para transmitir é acionada pelo evento *AutorizaTransmissaoEvent*. O componente EstacaoOrigem decrementa a fila de quadros de 1 (um). Esta situação é ilustrada no diagrama de colaboração da Figura 5.7.

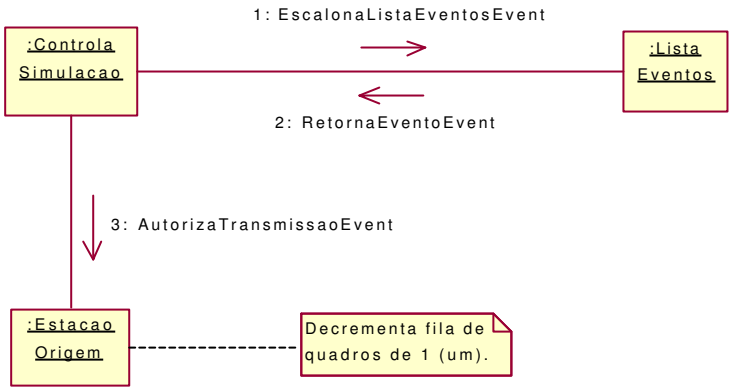

Figura 5.7 – Diagrama de colaboração AutorizaTransmissaoEvent (EstacaoOrigem).

5.4.7 Diagrama de colaboração *AutorizaTransmissaoEvent* (Enlace)

O diagrama de colaboração da Figura 5.8 foca o evento *AutorizaTransmissaoEvent* que aciona o componente Enlace.

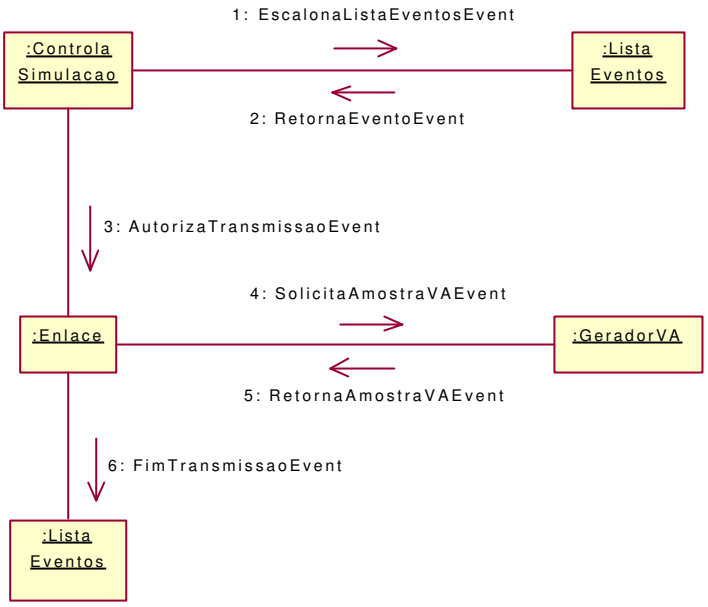

Figura 5.8 – Diagrama de colaboração *AutorizaTransmissaoEvent* (Enlace).

O componente Enlace recebe do ambiente de simulação (componente ControlaSimulacao) o evento *AutorizaTransmissaoEvent*. Este componente interage com o componente GeradorVA para obter uma amostra do tempo de transmissão do quadro. O evento *FimTransmissaoEvent* é gerado pelo componente Enlace e adicionado ao componente ListaEventos.

O tempo de ocorrência do evento *FimTransmissaoEvent* é obtido somando a amostra do tempo de transmissão do quadro com o tempo de ocorrência do evento corrente (*AutorizaTransmissaoEvent*). O evento *FimTransmissaoEvent* é observado pelos componentes CamadaMAC802.11 e EstacaoDestino (e pelo componente PontoAcessoOrigem, em caso de redes multi-célula).

5.4.8 Diagrama de colaboração *FimTransmissaoEvent* (CamadaMAC802.11)

O evento *FimTransmissaoEvent* também aciona o componente CamadaMAC802.11, com a finalidade de informar que o enlace está livre, viabilizando, dessa forma, outra transmissão de quadro, conforme ilustrado no diagrama de colaboração da Figura 5.9. Esse evento

também deve ser gerado no início da simulação, para informar aos demais componentes que o observam que o enlace está livre.

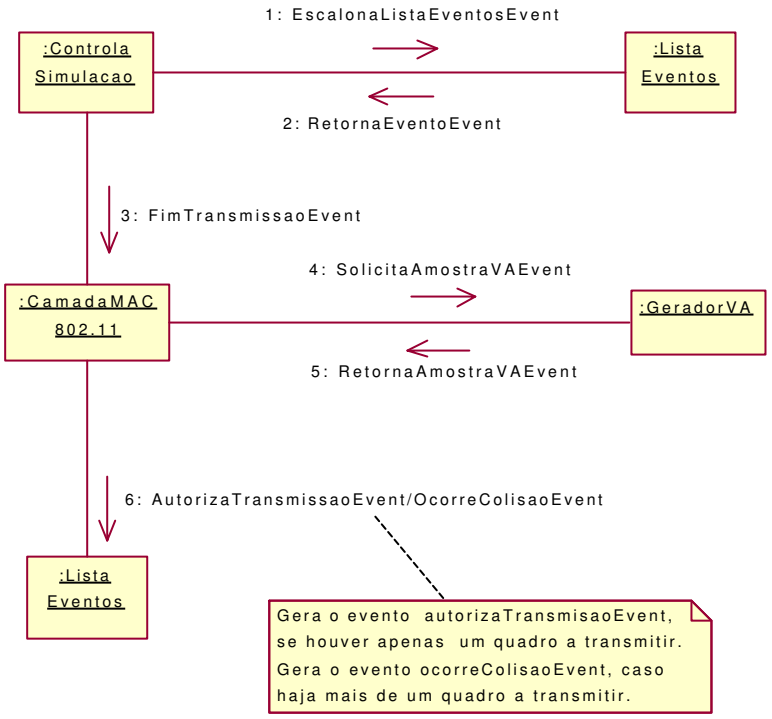

Figura 5.9 – Diagrama de colaboração *FimTransmissaoEvent* (CamadaMAC802.11).

O componente CamadaMAC802.11 verifica se existem estações aptas a transmitir. Se não houver, este componente nada faz. Havendo uma única estação apta a transmitir, ele gera o evento *AutorizaTransmissaoEvent* para o componente ListaEventos, para acionar o componente EstacaoOrigem.

Se mais de uma estação estiver apta a transmitir, caracteriza uma colisão. Nesse caso, o componente CamadaMAC802.11 gera o evento *OcorreColisaoEvent*. Este evento aciona os componentes EstacaoOrigem que têm quadros a transmitir, para retirarem o quadro de suas filas de transmissão, e aciona o componente Enlace, para modelar a ocupação do canal de comunicação.

O diagrama da Figura 5.9 também é considerado em redes multi-célula.

5.4.9 Diagrama de colaboração *FimTransmissaoEvent* (EstacaoDestino)

O digrama de colaboração da Figura 5.10 foca o evento *FimTransmissaoEvent* que aciona o componente EstacaoDestino.

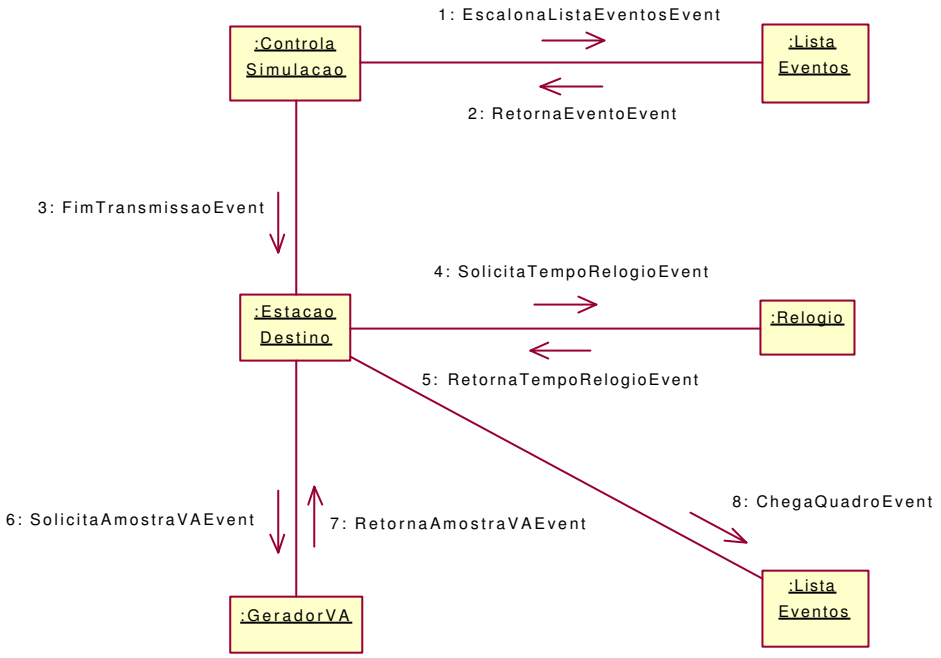

Figura 5.10 – Diagrama de colaboração *FimTransmissaoEvent* (EstacaoDestino).

Quando o evento *FimTransmissaoEvent* aciona o componente EstacaoDestino, este componente interage com o componente GeradorVA para encontrar uma amostra do tempo de processamento do quadro nesta estação. Em seguida, o evento *ChegaQuadroEvent* é gerado e enviado ao ambiente de simulação (componente ListaEventos).

O tempo de ocorrência do evento *ChegaQuadroEvent* é obtido somando o tempo de ocorrência do evento corrente (*FimTransmissaoEvent*) com a amostra do tempo de processamento do quadro.

5.4.10 Diagrama de colaboração *ChegaQuadroEvent* (Sorvedouro)

O diagrama de colaboração da Figura 5.11 foca o evento *ChegaQuadroEvent* que aciona o componente Sorvedouro.

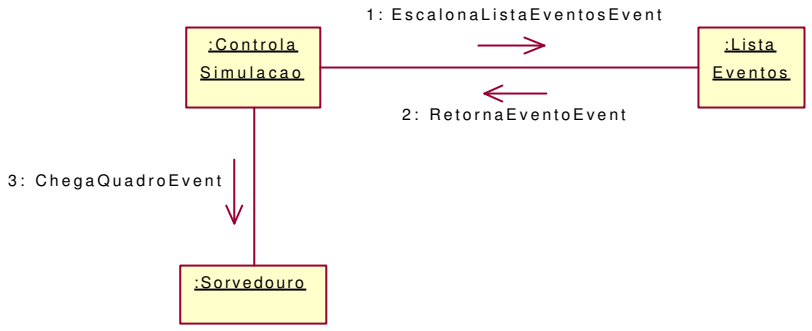

Figura 5.11 – Diagrama de colaboração *ChegaQuadroEvent* (Sorvedouro).

A ocorrência do evento *ChegaQuadroEvent* faz o componente Sorvedouro eliminar o quadro recebido e coletar estatísticas de recebimento de quadro, caracterizando a entrega do quadro no destino.

Os diagramas de colaboração seguintes abordam a dinâmica de simulação de uma rede multi-célula.

Diagramas de Colaboração – REDES MULTI-CÉLULA

Em um modelo de rede multi-célula, uma estação origem recebe um quadro de uma fonte de tráfego, armazena-o e aciona a camada de acesso ao meio para transmitir o quadro através do Enlace. Transmitindo o quadro, a estação origem decrementa de 1 (um) sua fila de quadros.

Depois disso, o enlace transmite o quadro que é recebido pelo ponto de acesso origem, que coloca o quadro na sua fila para ser transmitido. O ponto de acesso origem é interligado ao

ponto de acesso destino através do sistema de distribuição. Quando o sistema de distribuição fica livre, o ponto de acesso origem, tendo quadro a transmitir, faz a transmissão de um quadro e decrementa de 1 (um) sua fila de quadros.

O quadro transmitido pelo sistema de distribuição chega ao ponto de acesso destino, que coloca o quadro em sua fila (aumenta de 1 sua fila de quadros). No modelo, considera-se que o ponto de acesso destino comporta-se de forma semelhante a uma estação origem. Dessa forma, a camada de acesso ao meio deve controlar a utilização do enlace, observando o ponto de acesso destino e as estações origens que tenham quadros a transmitir.

No contexto da célula destino, o recebimento de um quadro pelo ponto de acesso destino, é semelhante ao modelo de uma rede *ad hoc*, quando uma estação origem recebe um quadro de uma fonte de tráfego. Assim, todos os diagramas de colaboração de um modelo de rede *ad hoc* se aplicam a um modelo de rede multi-célula.

Além desses diagramas, outros específicos das redes multi-célula se fazem necessários, para focar os eventos que acionam os componentes PontoAcessoOrigem, SistemaDistribuicao e PontoAcessoDestino. Vejamos nas subseções seguintes esses diagramas.

5.4.11 Diagrama de colaboração *FimTransmissaoEvent* (PontoAcessoOrigem)

A Figura 5.12 foca o evento *FimTransmissaoEvent* gerado pelo componente Enlace da célula origem. Esse evento aciona o componente PontoAcessoOrigem. Neste componente os quadros que chegam ficam armazenados em fila, aguardando a vez de serem transmitidos. Essa fila de quadros é semelhante àquela do componente EstacaoOrigem do modelo de rede *ad hoc* (subseção 5.4.2).

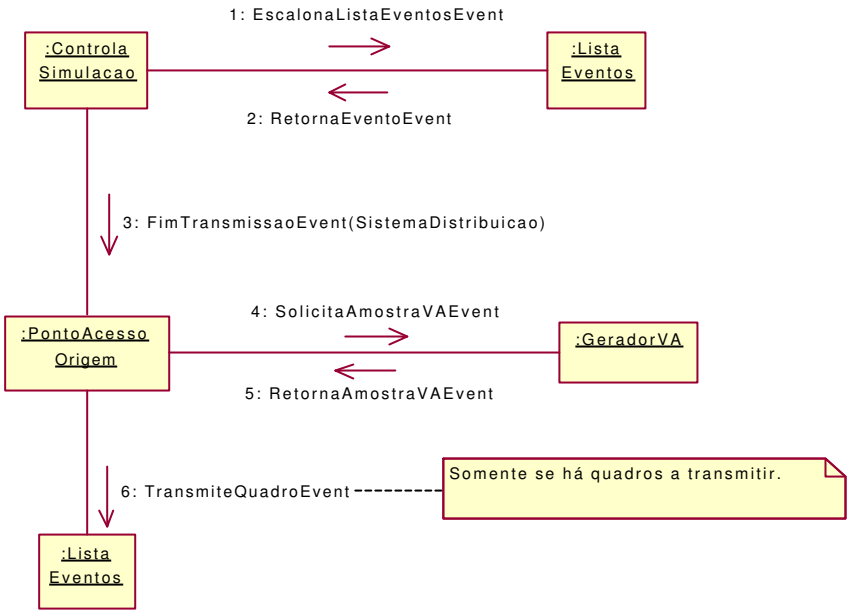

Figura 5.12 – Diagrama de colaboração *FimTransmissaoEvent* (PontoAcessoOrigem).

Quando o componente PontoAcessoOrigem é acionado, ele interage com o componente GeradorVA para obter uma amostra do tempo de processamento do quadro. O comprimento de fila de quadros é aumentado de 1 (um) e o evento *TransmiteQuadroEvent* é gerado com tempo de ocorrência igual a soma do tempo de ocorrência do evento corrente (*FimTransmissaoEvent*) com a amostra do tempo de processamento obtida.

5.4.12 Diagrama de colaboração *TransmiteQuadroEvent* (SistemaDistribuicao)

A Figura 5.13 foca o evento *TransmiteQuadroEvent* que aciona o componente SistemaDistribuicao. Este componente modela o meio de comunicação entre o componente PontoAcessoOrigem e PontoAcessoDestino.

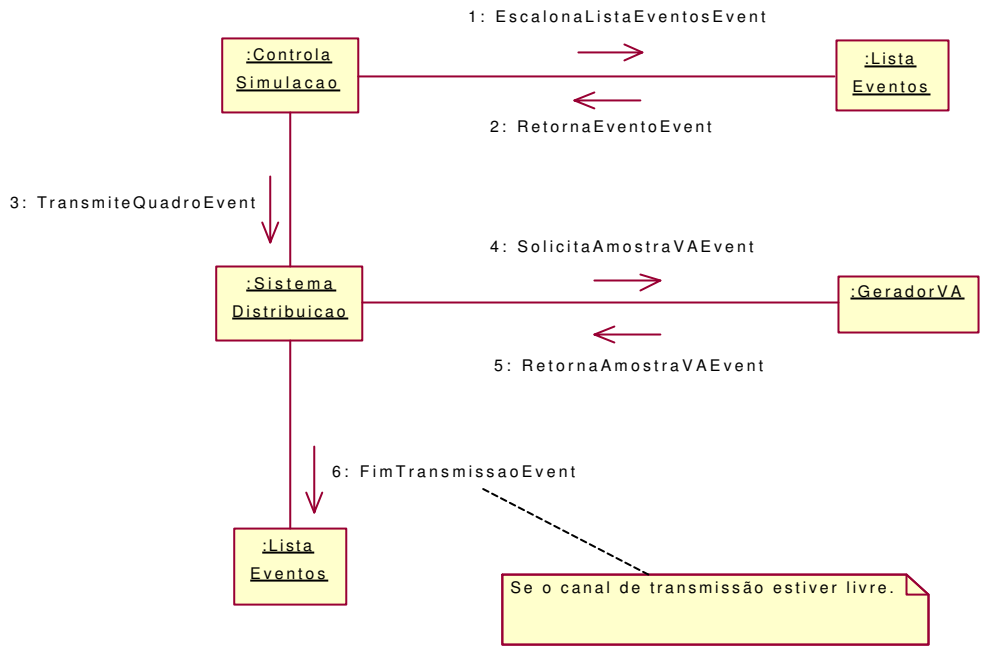

Figura 5.13 – Diagrama de colaboração *TransmiteQuadroEvent* (SistemaDistribuicao).

Quando o componente SistemaDistribuicao é acionado, ele verifica se o meio de transmissão está livre. Se o meio não estiver livre, esse componente nada faz. Se sim, ele interage com o componente GeradorVA para obter uma amostra do tempo de transmissão do quadro, e gera o evento *FimTransmissaoEvent* no componente SistemaDistribuicao, com tempo de ocorrência igual a soma do tempo de ocorrência do evento corrente (*TransmiteQuadroEvent*) com a amostra do tempo de transmissão do quadro.

5.4.13 Diagrama de colaboração *FimTransmissaoEvent* (SistemaDistribuicao – PontoAcessoOrigem)

A Figura 5.14 foca o evento *FimTransmissaoEvent* gerado pelo componente SistemaDistribuicao que aciona o componente PontoAcessoOrigem.

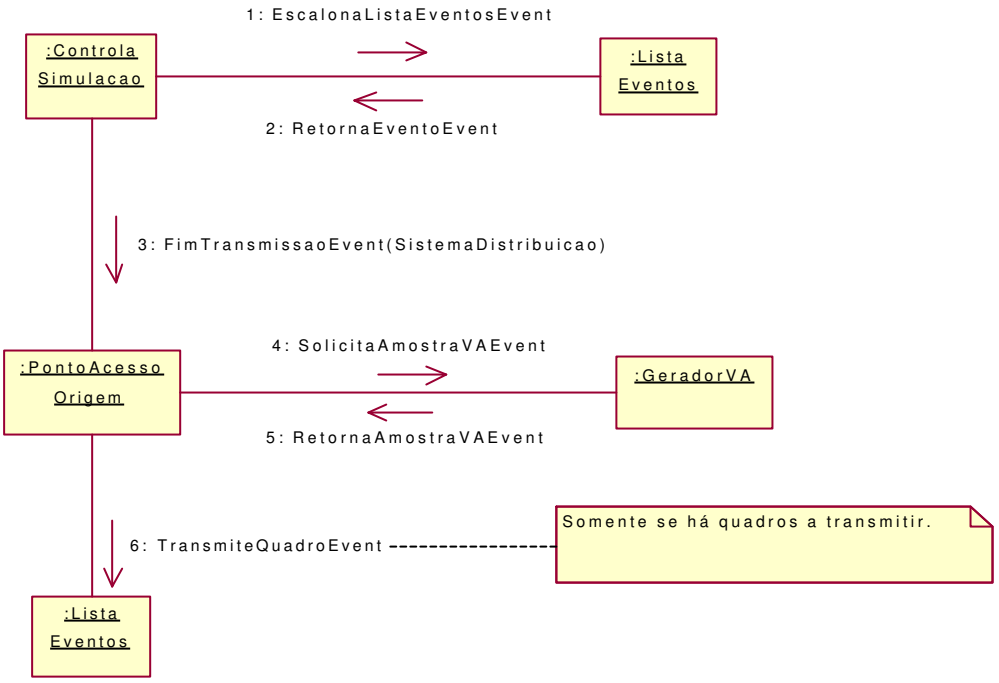

Figura 5.14 – Diagrama de colaboração *FimTransmissaoEvent* (SistemaDistribuicao–PontoAcessoOrigem).

Quando o componente PontoAcessoOrigem é acionado pelo evento *FimTransmissaoEvent*, ele verifica se há quadro esperando em fila para ser transmitido. Se não há quadro, esse componente nada faz. Se sim, ele interage com o componente GeradorVA para obter uma amostra de tempo para o processamento do quadro e, em seguida, gera o evento *TransmiteQuadroEvent*.

5.4.14 Diagrama de colaboração *FimTransmissaoEvent* (SistemaDistribuicao – PontoAcessoDestino)

O diagrama de colaboração da Figura 5.15 foca o evento *FimTransmissaoEvent* gerado pelo componente SistemaDistribuicao que aciona o componente PontoAcessoDestino.

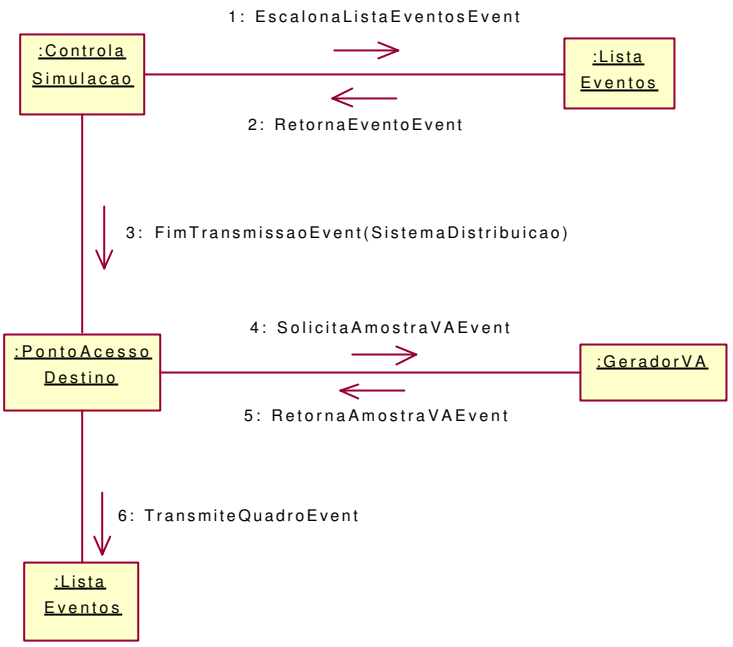

Figura 5.15 – Diagrama de colaboração *FimTransmissaoEvent* (SistemaDistribuicao – PontoAcessoDestino).

O componente PontoAcessoDestino é modelado da mesma forma que o componente EstacaoOrigem da célula origem, quando este componente recebe um quadro (subseção 5.4.2), diferindo apenas do fato daqui não ser necessária a auto-geração de quadros. Sendo assim, o componente PontoAcessoDestino depois de interagir com o componente GeradorVA, aciona o evento *TransmiteQuadroEvent,* que é enviado para o ambiente de simulação (componente ListaEventos).

O diagrama de colaboração *TransmiteQuadro* é o mesmo apresentado para o modelo de rede *ad hoc* (esse evento aciona o componente CamadaMAC802.11 na célula destino). Conforme já mencionado, os diagramas de colaboração do modelo de rede *ad hoc* se aplicam ao modelo de rede multi-célula aqui modelado.

5.5 Componentes 802.11 e eventos

Baseado nos diagramas de colaboração dos tipos de redes 802.11, ilustramos a seguir os eventos de simulação que cada componente 802.11 pode receber e gerar. Aqui usamos o artefato de UML *package* para representar os componentes.

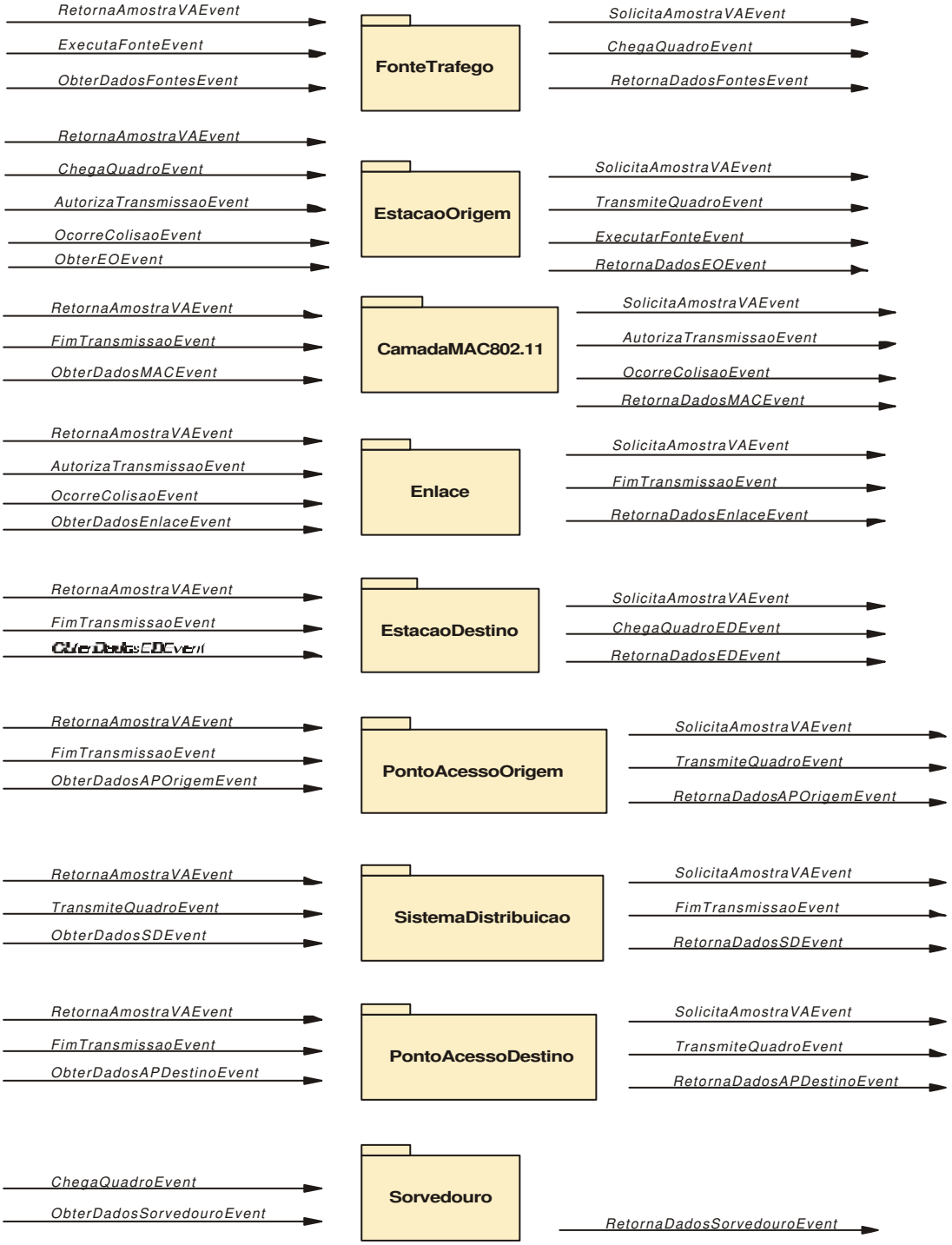

Figura 5.16 – Componentes 802.11 e Eventos.

5.6 Projeto arquitetural

5.6.1 Projeto de alto nível

O projeto arquitetural ilustra a arquitetura do sistema. A arquitetura adotada é a de três camadas. As três camadas desse modelo são: *apresentação*, *lógica da aplicação* e *armazenamento de dados* [Larman98]:

- **Apresentação:** representa a interface a qual o analista de modelagem configura os componentes para a simulação.
- **Lógica da aplicação:** representa os elementos necessários para a modelagem das redes 802.11, considerados a partir do mais alto nível de abstração possível.
- **Armazenamento de dados:** representa as estruturas de dados usadas para armazenar as informações persistentes de configurações dos componentes 802.11 e de filas.

Vejamos na Figura 5.17 o projeto arquitetural de alto nível para os componentes 802.11.

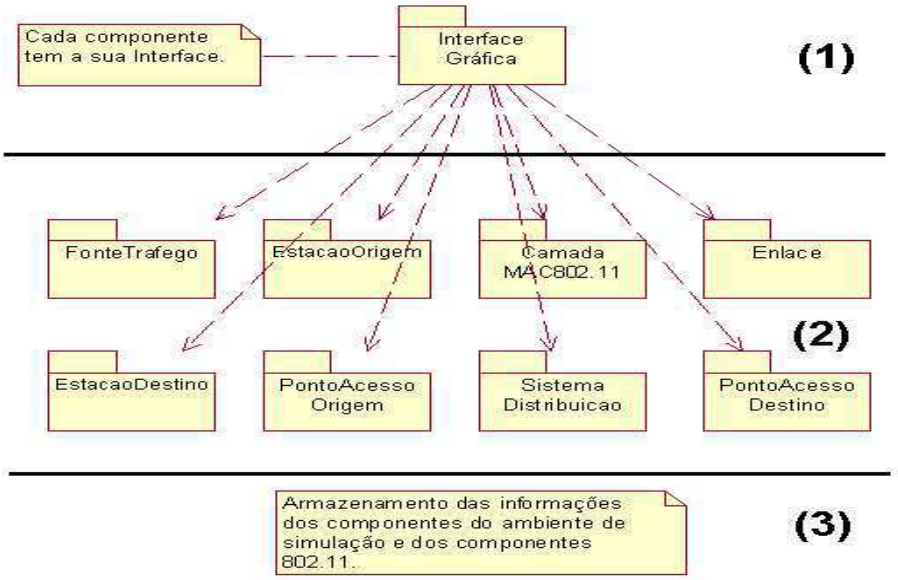

Figura 5.17 – Visão arquitetural (3 camadas) dos componentes 802.11.

O primeiro nível da Figura 5.17 (1), representa as interfaces que cada componente terá para receber suas respectivas configurações. O seguinte nível (2), representa os componentes 802.11, tidos como a lógica da aplicação. O terceiro nível (3), representa o armazenamento das informações referentes aos componentes 802.11 e aos demais componentes de um ambiente de simulação.

Uma vez que esta Dissertação investe na especificação de componentes que representam o funcionamento essencial das redes 802.11, apenas os elementos da **lógica de aplicação** (nível 2) são detalhados, já que os elementos dos níveis (1) e (2) da Figura 5.17 não fazem parte do escopo deste trabalho.

Dessa forma, faremos a especificação mais detalhada de cada componente definindo suas interfaces, classes, atributos, operações e interações internas. No detalhamento desses componentes é adotado o artefato de UML "classe", que apresenta informações referentes às propriedades e serviços (métodos) de cada elemento que compõe um componente 802.11.

Iniciamos a especificação dos componentes a partir do componente FonteTrafego, que gera o tráfego na rede. A ordem da especificação segue o funcionamento das redes 802.11 ilustrada no modelo conceitual, apresentado no capítulo anterior.

5.6.2 Projeto detalhado

5.6.2.1 Componente FonteTrafego

A Figura 5.18 ilustra o componente FonteTrafego.

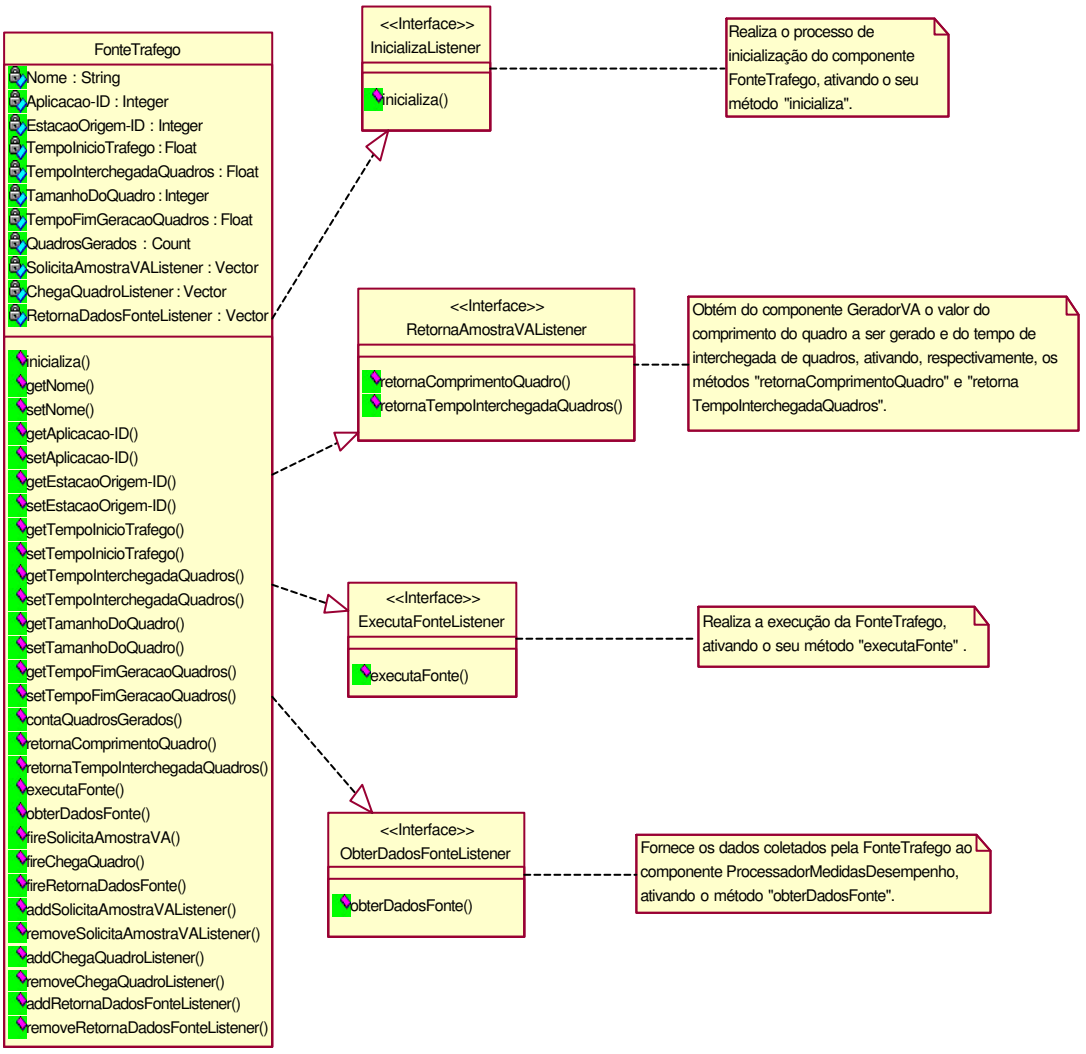

Figura 5.18 – Componente FonteTrafego.

Vejamos a seguir a descrição dos atributos e dos principais métodos do componente FonteTrafego.

Atributos do componente ***FonteTrafego***

- **Nome:** representa a identificação do componente FonteTrafego.
- **Aplicacao-ID:** representa a identificação da aplicação que gerou o tráfego.
- **EstacaoOrigem-ID:** representa a identificação do componente EstacaoOrigem o qual o componente FonteTrafego está associado.
- **TempoInicioTrafego:** representa o tempo de início da geração de quadros pelo componente FonteTrafego.
- **TempoInterchegadaQuadros:** representa o valor médio de uma função de distribuição de probabilidade (ou uma constante) usada para definir o tempo entre a geração de quadros.
- **TamanhoDoQuadro:** representa o valor médio de uma função de distribuição de probabilidade (ou uma constante) usada para definir o tamanho das unidades de informações (quadros) que são geradas na rede.
- **TempoFimGeracaoQuadros:** armazena o tempo em que a geração de quadros deve ser interrompida.
- **QuadrosGerados:** armazena o número total de quadros gerados pelo componente FonteTrafego.
- **SolicitaAmostraVAListener:** armazena os listerners do evento *SolicitaAmostraVAEvent*.
- **ChegaQuadroListener:** armazena os listeners do evento *ChegaQuadroEvent*.
- **RetornaDadosFonteListener:** armazena os listeners do evento *RetornaDadosFonteEvent*.

Métodos do componente ***FonteTrafego***

- **inicializa():** método da *Interface* "InicializaListener" que realiza o processo de inicialização do componente FonteTrafego.
- **contaQuadrosGerados():** método usado para calcular o número total de quadros gerados pelo componente FonteTrafego.

- **retornaComprimentoQuadro():** método da *Interface* "RetornaAmostraVAListener" que obtém do componente GeradorVA, o valor médio do comprimento do quadro a ser gerado pelo componente FonteTrafego, caso esse comprimento não seja pré-definido (constante).
- **retornaTempoInterchegadaQuadro():** método da *Interface* "RetornaAmostraVAListener" que obtém do componente GeradorVA, o valor médio do tempo de interchegada dos quadros gerados pelo componente FonteTrafego.
- **executaFonte():** método da *Interface* "ExecutaFonteListener" que executa o componente FonteTrafego.
- **obterDadosFonte():** método da *Interface* "ObterDadosFonteListener" usado para fornecer ao componente ProcessadorMedidasDesempenho, os dados coletados pelo componente FonteTrafego.
- **fireSolicitaAmostraVA():** método usado para disparar o evento *SolicitaAmostraVAEvent*.
- **fireChegaQuadro():** método usado para disparar o evento *ChegaQuadroEvent*.
- **fireRetornaDadosFonte():** método usado para disparar o evento *RetornaDadosFonteEvent*.
- **addSolicitaAmostraVAListener():** método que fornece o cadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **removeSolicitaAmostraVAListener():** método que fornece o descadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **addChegaQuadroListener():** método que fornece o cadastro de listener para o evento *ChegaQuadroEvent*.
- **removeChegaQuadroListener():** método que fornece o descadastro de listener para o evento *ChegaQuadroEvent*.
- **addRetornaDadosListener():** método que fornece o cadastro de listener para o evento *RetornaDadosEvent*.
- **removeRetornaDadosFonteListener():** método que fornece o descadastro de listener para o evento *RetornaDadosFonteEvent*.

5.6.2.2 Componente EstacaoOrigem

A Figura 5.19 ilustra o componente EstacaoOrigem.

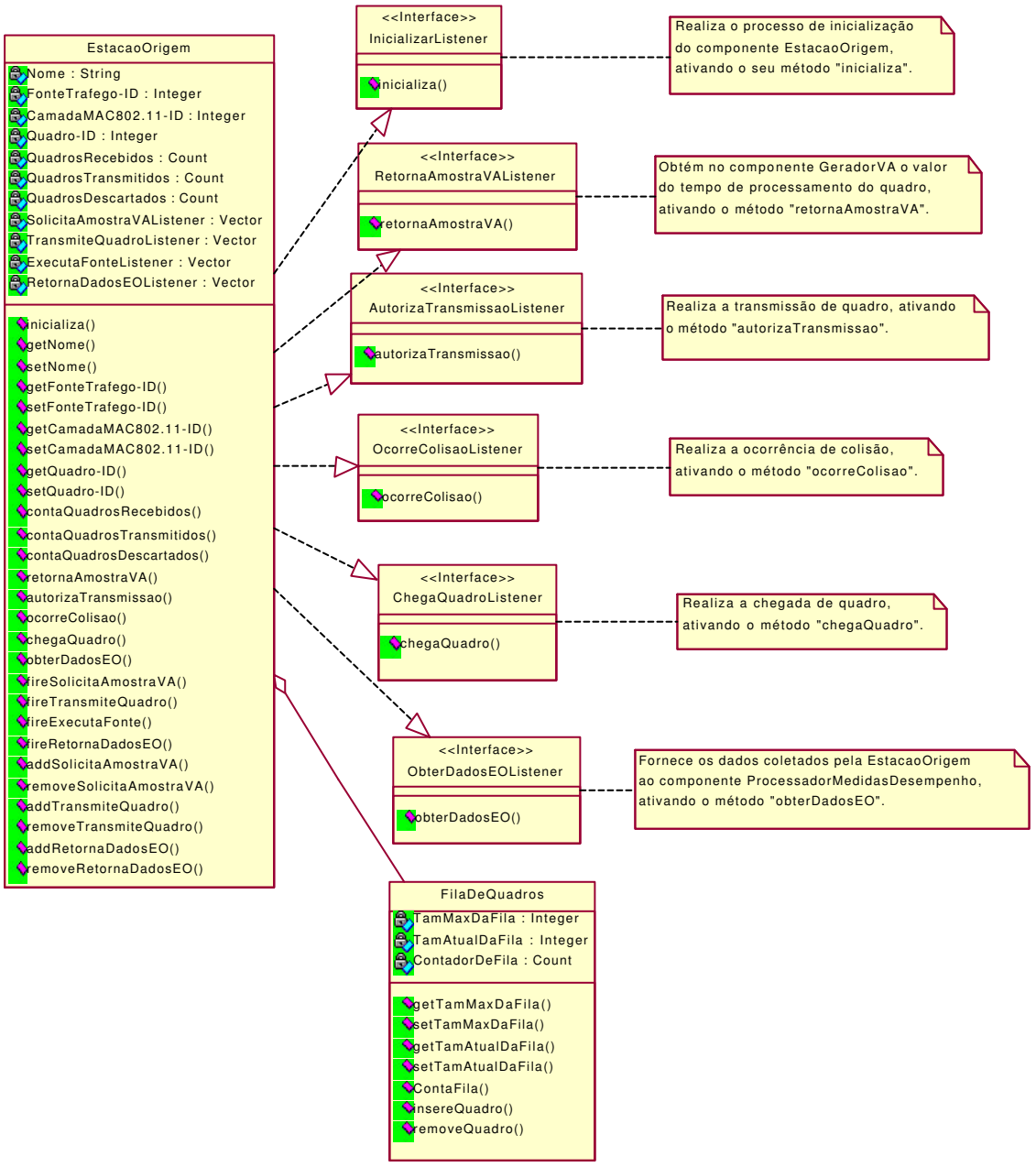

Figura 5.19 – Componente EstacaoOrigem.

Vejamos a seguir a descrição dos atributos e dos principais métodos do componente EstacaoOrigem e de sua classe "FilaDeQuadros".

Atributos do componente ***EstacaoOrigem***

- **Nome:** representa a identificação do componente EstacaoOrigem.
- **CamadaMAC802.11-ID:** representa a identificação do componente CamadaMAC802.11 o qual o componente EstacaoOrigem está associado.
- **QuadrosRecebidos:** armazena o número total de quadros recebidos pelo componente EstacaoOrigem oriundos do componente FonteTrafego.
- **QuadrosTransmitidos:** armazena o número total de quadros transmitidos pelo componente EstacaoOrigem.
- **QuadrosDescartados:** armazena o número total de quadros descartados pelo componente EstacaoOrigem.
- **Quadro-ID:** representa a identificação do quadro.
- **SolicitaAmostraVAListener:** armazena os listerners do evento *SolicitaAmostraVAEvent*.
- **TransmiteQuadroListener:** armazena os listeners do evento *TransmiteQuadroEvent*.
- **ExecutaFonteListener:** armazena os listeners do evento *ExecutaFonteEvent*.
- **RetornaDadoEOListener:** armazena os listeners do evento *RetornaDadoEOEvent*.

Métodos do componente ***EstacaoOrigem***

- **inicializa():** método da *Interface* "InicializaListener" que realiza o processo de inicialização do componente EstacaoOrigem.
- **contaQuadrosRecebidos():** armazena o número total de quadros recebidos pelo componente EstacaoOrigem.
- **contaQuadrosTransmitidos():** armazena o número total de quadros transmitidos pelo componente EstacaoOrigem.
- **contaQuadrosDescartados():** armazena o número total de quadros descartados pelo componente EstacaoOrigem.

- **retornaAmostraVA():** método da *Interface* "RetornaAmostraVAListener" que obtém do componente GeradorVA, o valor médio do tempo de processamento do quadro no componente EstacaoOrigem.
- **autorizaTransmissao():** método da *Interface* "AutorizaTransmissaoListener" que executa a transmissão de um quadro pelo componente EstacaoOrigem.
- **ocorreColisao():** método da *Interface* "OcorreColisaoListener" que informa ao componente EstacaoOrigem que seu quadro colidiu.
- **chegaQuadro():** método da *Interface* "ChegaQuadroListener" que executa a chegada de quadro no componente EstacaoOrigem.
- **obterDadosEO():** método da *Interface* "ObterDadosEOListener" usado para fornecer ao componente ProcessadorMedidasDesempenho, os dados coletados pelo componente EstacaoOrigem.
- **fireSolicitaAmostraVA():** método usado para disparar o evento *SolicitaAmostraVAEvent*.
- **fireTransmiteQuadro():** método usado para disparar o evento *TransmiteQuadroEvent*.
- **fireExecutaFonte():** método usado para disparar o evento *ExecutaFonteEvent*.
- **fireRetornaDadosEO():** método usado para disparar o evento *RetornaDadosEOEvent*.
- **addSolicitaAmostraVAListener():** método que fornece o cadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **removeSolicitaAmostraVAListener():** método que fornece o descadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **addTransmiteQuadroListener():** método que fornece o cadastro de listener para o evento *TransmiteQuadroEvent* .
- **removeTransmiteQuadroListener():** método que fornece o descadastro de listener para o evento *TransmiteQuadroEvent*.
- **addExecutaFonteListener():** método que fornece o cadastro de listener para o evento *ExecutaFonteEvent*.
- **removeExecutaFonteListener():** método que fornece o descadastro de listener para o evento *ExecutaFonteEvent*.

- **addRetornaDadosEOListener():** método que fornece o cadastro de listener para o evento *RetornaDadosEOEvent* .
- **removeRetornaDadosEOListener():** método que fornece o descadastro de listener para o evento *RetornaDadosEOEvent*.

Seguem descrições da classe "FilaDeQuadros" do componente EstacaoOrigem. Essa classe também é considerada nos componentes PontoAcessoOrigem e PontoAcessoDestino. Portanto, seus atributos e métodos só são descritos aqui.

- **TamMaxDaFila:** representa o número máximo de quadros que a Fila pode armazenar.
- **TamAtualDaFila:** representa o número de quadros na Fila num dado momento.
- **ContadorDeFila:** representa o número total de quadros que passaram pela Fila.
- **contaFila():** método usado para calcular o número total de quadros que passaram pela Fila.
- **insereQuadro():** método que insere um quadro na Fila.
- **removeQuadro():** método que remove um quadro da Fila.

5.6.2.3 Componente CamadaMAC802.11

A Figura 5.20 ilustra o componente CamadaMAC802.11.

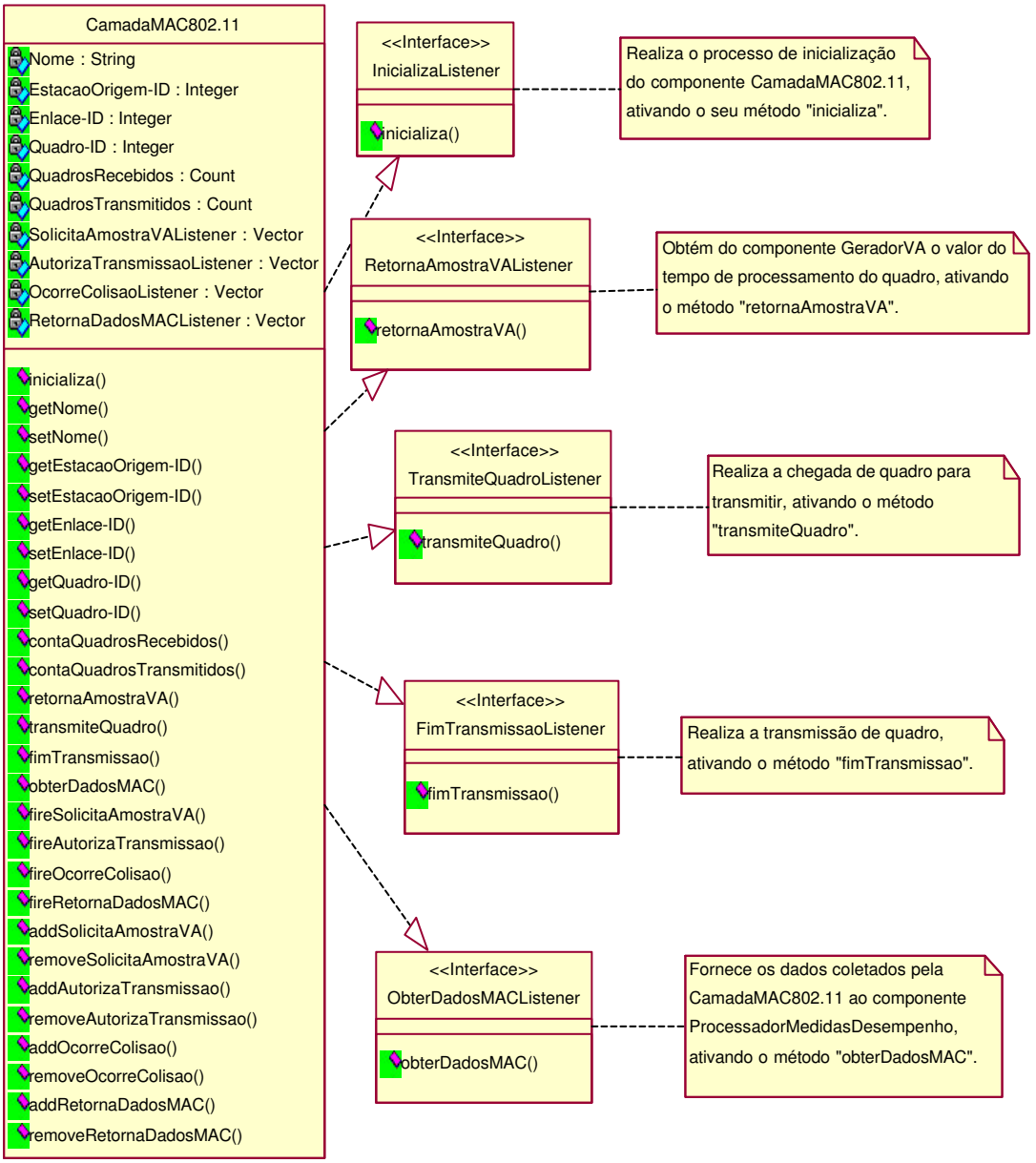

Figura 5.20 – Componente CamadaMAC802.11.

Vejamos a seguir a descrição dos atributos e dos principais métodos do componente CamadaMAC802.11.

Atributos do componente ***CamadaMAC802.11***

- **Nome:** representa a identificação do componente CamadaMAC802.11.
- **EstacaoOrigem-ID:** representa a identificação do componente EstacaoOrigem o qual o componente CamadaMAC802.11 está associado.
- **Enlace-ID:** representa a identificação do componente Enlace o qual o componente CamadaMAC802.11 está associado.
- **Quadro-ID:** representa a identificação do quadro.
- **QuadrosRecebidos:** armazena o número total de quadros recebidos pelo componente CamadaMAC802.11.
- **QuadrosTransmitidos:** armazena o número total de quadros transmitidos pelo componente CamadaMAC802.11.
- **SolicitaAmostraVAListener:** armazena os listerners do evento *SolicitaAmostraVAEvent*.
- **AutorizaTransmissaoListener:** armazena os listeners do evento *AutorizaTransmissaoEvent*.
- **OcorreColisaoListener:** armazena os listeners do evento *OcorreColisaoEvent*.
- **RetornaDadosMACListener:** armazena os listeners do evento *RetornaDadosMACEvent* .

Métodos do componente ***CamadaMAC802.11***

- **inicializa():** método da *Interface* "InicializaListener" que realiza o processo de inicialização do componente CamadaMAC802.11.
- **contaQuadrosRecebidos():** método usado para calcular o número total de quadros recebidos pelo componente CamadaMAC802.11.
- **contaQuadrosTransmitidos():** método usado para calcular o número total de quadros transmitidos pelo componente CamadaMAC802.11.

- **retornaAmostraVA():** método da *Interface* "RetornaAmostraVAListener" que obtém do componente GeradorVA, o valor médio do tempo de processamento do quadro no componente CamadaMAC802.11.
- **transmiteQuadro():** método da *Interface* "TransmiteQuadroListener" que executa a chegada de quadro para transmitir no componente CamadaMAC802.11.
- **fimTransmissao():** método da *Interface* "FimTransmissaoListener" que executa a transmissão de quadro pelo componente CamadaMAC802.11.
- **obterDadosMAC():** método da *Interface* "ObterDadosMACListener" usado para fornecer ao componente ProcessadorMedidasDesempenho, os dados coletados pelo componente CamadaMAC802.11.
- **fireSolicitaAmostraVA():** método usado para disparar o evento *SolicitaAmostraVAEvent*.
- **fireAutorizaTransmissao():** método usado para disparar o evento *AutorizaTransmissaoEvent*.
- **fireOcorreColisao():** método usado para disparar o evento *OcorreColisaoEvent*.
- **fireRetornaDadosMAC():** método usado para disparar o evento *RetornaDadosMACEvent*.
- **addSolicitaAmostraVAListener():** método que fornece o cadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **removeSolicitaAmostraVAListener():** método que fornece o descadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **addAutorizaTransmissaoListener():** método que fornece o cadastro de listener para o evento *AutorizaTransmissaoEvent*.
- **removeAutorizaTransmissaoListener():** método que fornece o descadastro de listener para o evento *AutorizaTransmissaoEvent*.
- **addOcorreColisaoListener():** método que fornece o cadastro de listener para o evento *OcorreColisaoEvent*.
- **removeOcorreColisaoListener():** método que fornece o descadastro de listener para o evento *OcorreColisaoEvent*.
- **addRetornaDadosMACListener():** método que fornece o cadastro de listener para o evento *RetornaDadosMACEvent*.

- **removeRetornaDadosMACListener():** método que fornece o cadastro de listener para o evento *RetornaDadosMACEvent*.

5.6.2.4 Componente Enlace

A Figura 5.21 ilustra o componente Enlace.

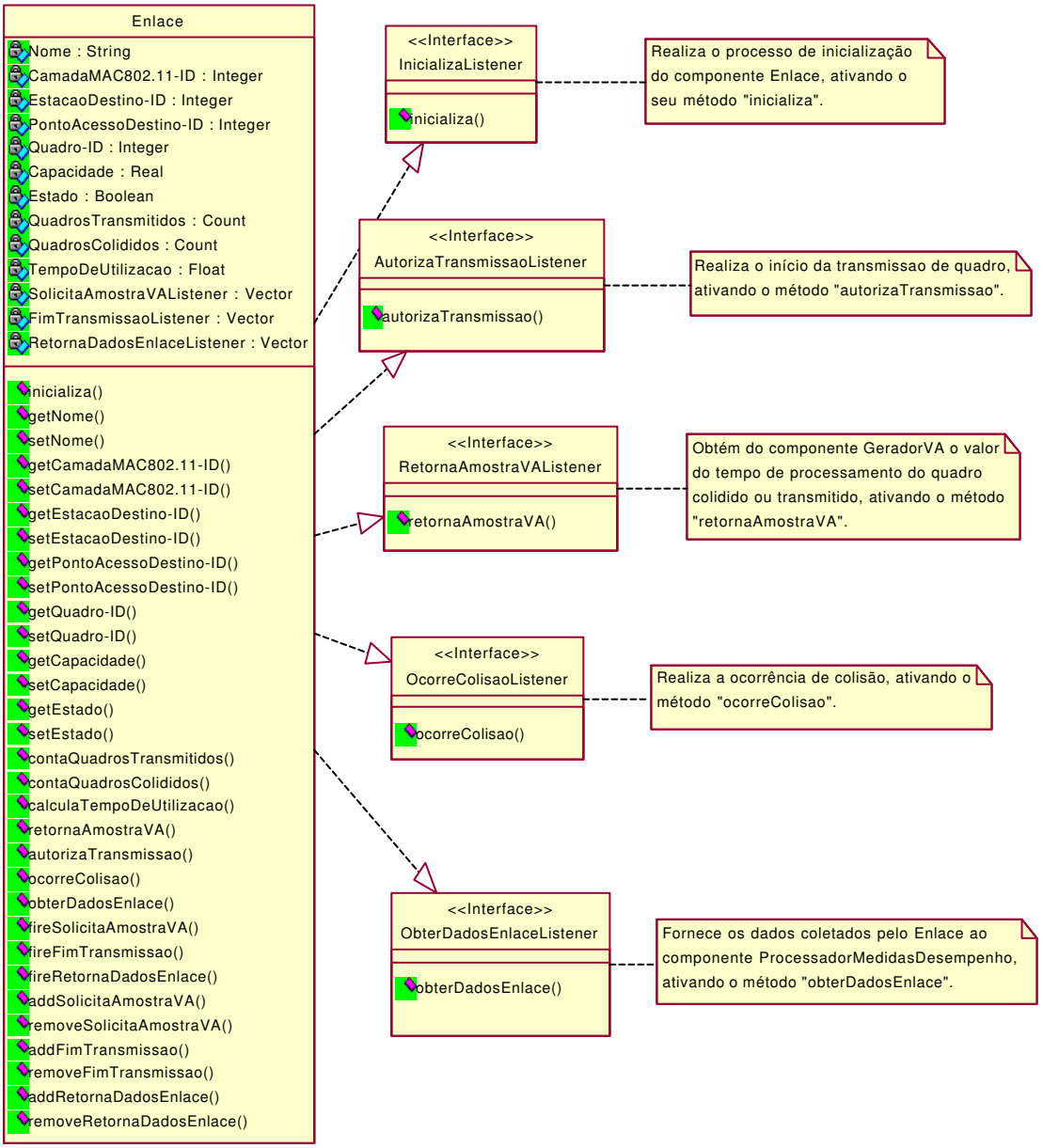

Figura 5.21 – Componente Enlace.

Vejamos a seguir a descrição dos atributos e dos principais métodos do componente Enlace.

Atributos do componente Enlace

- **Nome:** representa a identificação do componente Enlace.
- **CamadaMAC802.11-ID:** representa a identificação do componente CamadaMAC802.11 o qual o componente Enlace está associado.
- **EstacaoDestino-ID:** representa a identificação do componente EstacaoDestino o qual o componente Enlace está associado.
- **PontoAcessoOrigem-ID:** representa a identificação do componente PontoAcessoOrigem o qual o componente Enlace está associado, em caso de modelos de redes multi-célula.
- **Quadro-ID:** representa a identificação do quadro.
- **Capacidade:** capacidade de transmissão do componente Enlace.
- **Estado:** estado do componente Enlace, se ocupado ou livre.
- **QuadrosTransmitidos:** armazena o número total de quadros transmitidos no componente Enlace.
- **QuadrosColididos:** armazena o número total de quadros colididos no componente Enlace.
- **TempoDeUtilizacao:** armazena o tempo total de utilização do componente Enlace.
- **SolicitaAmostraVAListener:** armazena os listerners do evento *SolicitaAmostraVAEvent*.
- **FimTransmissaoListener:** armazena os listeners do evento *FimTransmissaoEvent*.
- **RetornaDadosEnlaceListener:** armazena os listeners do evento *RetornaDadosEnlaceEvent*.

Métodos do componente Enlace

- **inicializa():** método da *Interface* "InicializaListener" que realiza o processo de inicialização do componente Enlace.
- **contaQuadrosTransmitidos():** método usado para calcular o número total de quadros transmitidos no componente Enlace.

- **contaQuadrosColididos():** método usado para calcular o número total de quadros colididos no componente Enlace.
- **calculaTempoDeUtilizacao():** método usado para calcular o tempo de utilização do componente Enlace.
- **retornaAmostraVA():** método da *Interface* "RetornaAmostraVAListener" que obtém do componente GeradorVA, o valor médio do tempo de transmissão/colisão do quadro no componente Enlace.
- **autorizaTransmisao():** método da *Interface* "AutorizaTransmissaoListener" que executa a transmissão de quadro no componente Enlace.
- **ocorreColisao():** método da *Interface* "OcorreColisaoListener" que executa a ocorrência de colisão de quadros no componente Enlace.
- **obterDadosEnlace():** método da *Interface* "ObterDadosEnlaceListener" usado para fornecer ao componente ProcessadorMedidasDesempenho, os dados coletados pelo componente Enlace.
- **fireSolicitaAmostraVA():** método usado para disparar o evento *SolicitaAmostraVAEvent*.
- **fireFimTransmissao():** método usado para disparar o evento *FimTransmissaoEvent*.
- **fireRetornaDadosEnlace():** método usado para disparar o evento *RetornaDadosEnlaceEvent*.
- **addSolicitaAmostraVAListener():** método que fornece o cadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **removeSolicitaAmostraVAListener():** método que fornece o descadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **addFimTransmissaoListener():** método que fornece o cadastro de listener para o evento *FimTransmissaoEvent*.
- **removeFimTransmissaoListener():** método que fornece o descadastro de listener para o evento *FimTransmissaoEvent*.
- **addRetornaDadosEnlaceListener():** método que fornece o cadastro de listener para o evento *RetornaDadosEnlaceEvent*.
- **removeRetornaDadosEnlaceListener():** método que fornece o cadastro de listener para o evento *RetornaDadosEnlaceEvent*.

5.6.2.5 Componente EstacaoDestino

A Figura 5.22 ilustra o componente EstacaoDestino.

Figura 5.22 – Componente EstacaoDestino.

Vejamos a seguir a descrição dos atributos e dos principais métodos do componente EstacaoDestino.

*Atributos do componente **EstacaoDestino***

- **Nome:** representa a identificação do componente EstacaoDestino.
- **Enlace-ID:** representa a identificação do componente Enlace o qual o componente EstacaoDestino está associado.
- **Sorvedouro-ID:** representa a identificação do componente Sorvedouro o qual o componente EstacaoDestino está associado.
- **Quadro-ID:** representa a identificação do quadro.
- **QuadrosRecebidos:** armazena o número total de quadros recebidos pelo componente EstacaoDestino.
- **SolicitaAmostraVAListener:** armazena os listerners do evento *SolicitaAmostraVAEvent*.
- **ChegaQuadroListener:** armazena os listeners do evento *ChegaQuadroEvent*.
- **RetornaDadosEDListener:** armazena os listeners do evento *RetornaDadosEDEvent*.

*Métodos do componente **EstacaoDestino***

- **inicializa():** método da *Interface* "InicializaListener" que realiza o processo de inicialização do componente EstacaoDestino.
- **contaQuadrosRecebidos():** método usado para calcular o número total de quadros recebidos pelo componente EstacaoDestino.
- **retornaAmostraVA():** método da *Interface* "RetornaAmostraVAListener" que obtém do componente GeradorVA, o valor médio do tempo de recebimento de quadro pelo componente EstacaoDestino.
- **fimTransmissao():** método da *Interface* "FimTransmissaoListener" que executa o recebimento de quadro pelo componente EstacaoDestino.
- **obterDadosED():** método da *Interface* "ObterDadosEDListener" usado para fornecer ao componente ProcessadorMedidasDesempenho, os dados coletados pelo componente EstacaoDestino.

- **fireSolicitaAmostraVA():** método usado para disparar o evento *SolicitaAmostraVAEvent* .
- **fireChegaQuadro():** método usado para disparar o evento *ChegaQuadroEvent*.
- **fireRetornaDadosED():** método usado para disparar o evento *RetornaDadosEDEvent*.
- **addSolicitaAmostraVAListener():** método que fornece o cadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **removeSolicitaAmostraVAListener():** método que fornece o descadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **addChegaQuadroListener():** método que fornece o cadastro de listener para o evento *ChegaQuadroEvent*.
- **removeChegaQuadroListener():** método que fornece o descadastro de listener para o evento *ChegaQuadroEvent*.
- **addRetornaDadosEDListener():** método que fornece o cadastro de listener para o evento *RetornaDadosEDEvent* .
- **removeRetornaDadosEDListener():** método que fornece o descadastro de listener para o evento *RetornaDadosEDEvent*.

5.6.2.6 Componente PontoAcessoOrigem

A Figura 5.23 ilustra o componente PontoAcessoOrigem.

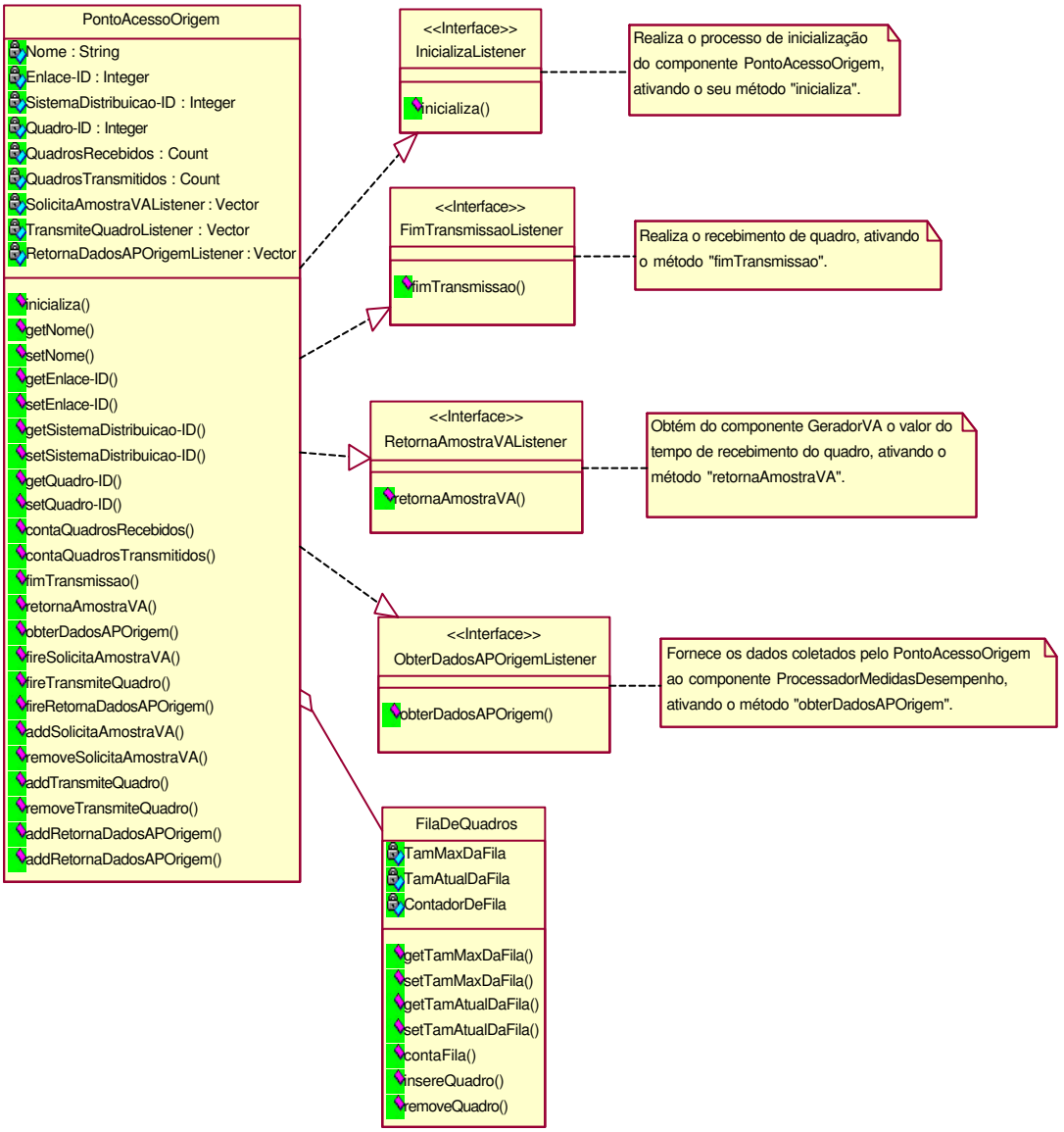

Figura 5.23 – Componente PontoAcessoOrigem.

Vejamos a seguir a descrição dos atributos e dos principais métodos do componente PontoAcessoOrigem.

Atributos do componente ***PontoAcessoOrigem***

- **Nome:** representa a identificação do componente PontoAcessoOrigem.
- **Enlace-ID:** representa a identificação do componente Enlace o qual o componente PontoAcessoOrigem está associado.
- **SistemaDistribuicao-ID:** representa a identificação do componente SistemaDistribuicao o qual o componente PontoAcessoOrigem está associado.
- **Quadro-ID:** representa a identificação do quadro.
- **QuadrosRecebidos:** armazena o número total de quadros recebidos pelo componente PontoAcessoOrigem.
- **QuadrosTransmitidos:** armazena o número total de quadros transmitidos pelo componente PontoAcessoOrigem.
- **SolicitaAmostraVAListener:** armazena os listerners do evento *SolicitaAmostraVAEvent*.
- **TransmiteQuadroListener:** armazena os listeners do evento *TransmiteQuadroEvent*.
- **RetornaDadosAPOrigemListener:** armazena os listeners do evento *RetornaDadosAPOrigemEvent*.

Métodos do componente ***PontoAcessoOrigem***

- **inicializa():** método da *Interface* "InicializaListener" que realiza o processo de inicialização do componente PontoAcessoOrigem.
- **contaQuadrosRecebidos():** método usado para calcular o número total de quadros recebidos pelo componente PontoAcessoOrigem.
- **contaQuadrosTransmitidos():** método usado para calcular o número total de quadros transmitidos pelo componente PontoAcessoOrigem.
- **fimTransmissao():** método da *Interface* "FimTransmissaoListener" que executa o recebimento de quadro pelo componente PontoAcessoOrigem.

- **retornaAmostraVA():** método da *Interface* "RetornaAmostraVAListener" que obtém do componente GeradorVA, o valor médio do tempo de recebimento de quadro pelo componente PontoAcessoOrigem.
- **obterDadosAPOrigem():** método da *Interface* "ObterDadosAPOrigemListener" usado para fornecer ao componente ProcessadorMedidasDesempenho, os dados coletados pelo componente PontoAcessoOrigem.
- **fireSolicitaAmostraVA():** método usado para disparar o evento *SolicitaAmostraVAEvent*.
- **fireTransmiteQuadro():** método usado para disparar o evento *TransmiteQuadroEvent*.
- **fireRetornaDadosAPOrigem():** método usado para disparar o evento *RetornaDadosAPOrigemEvent*.
- **addSolicitaAmostraVAListener():** método que fornece o cadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **removeSolicitaAmostraVAListener():** método que fornece o descadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **addTransmiteQuadroListener():** método que fornece o cadastro de listener para o evento *TransmiteQuadroEvent* .
- **removeTransmiteQuadroListener():** método que fornece o descadastro de listener para o evento *TransmiteQuadroEvent*.
- **addRetornaDadosAPOrigemListener():** método que fornece o cadastro de listener para o evento *RetornaDadosAPOrigemEvent*.
- **removeRetornaDadosAPOrigemListener():** método que fornece o descadastro de listener para o evento *RetornaDadosAPOrigemEvent*.

5.6.2.7 Componente SistemaDistribuicao

A Figura 5.24 ilustra o componente SistemaDistribuicao.

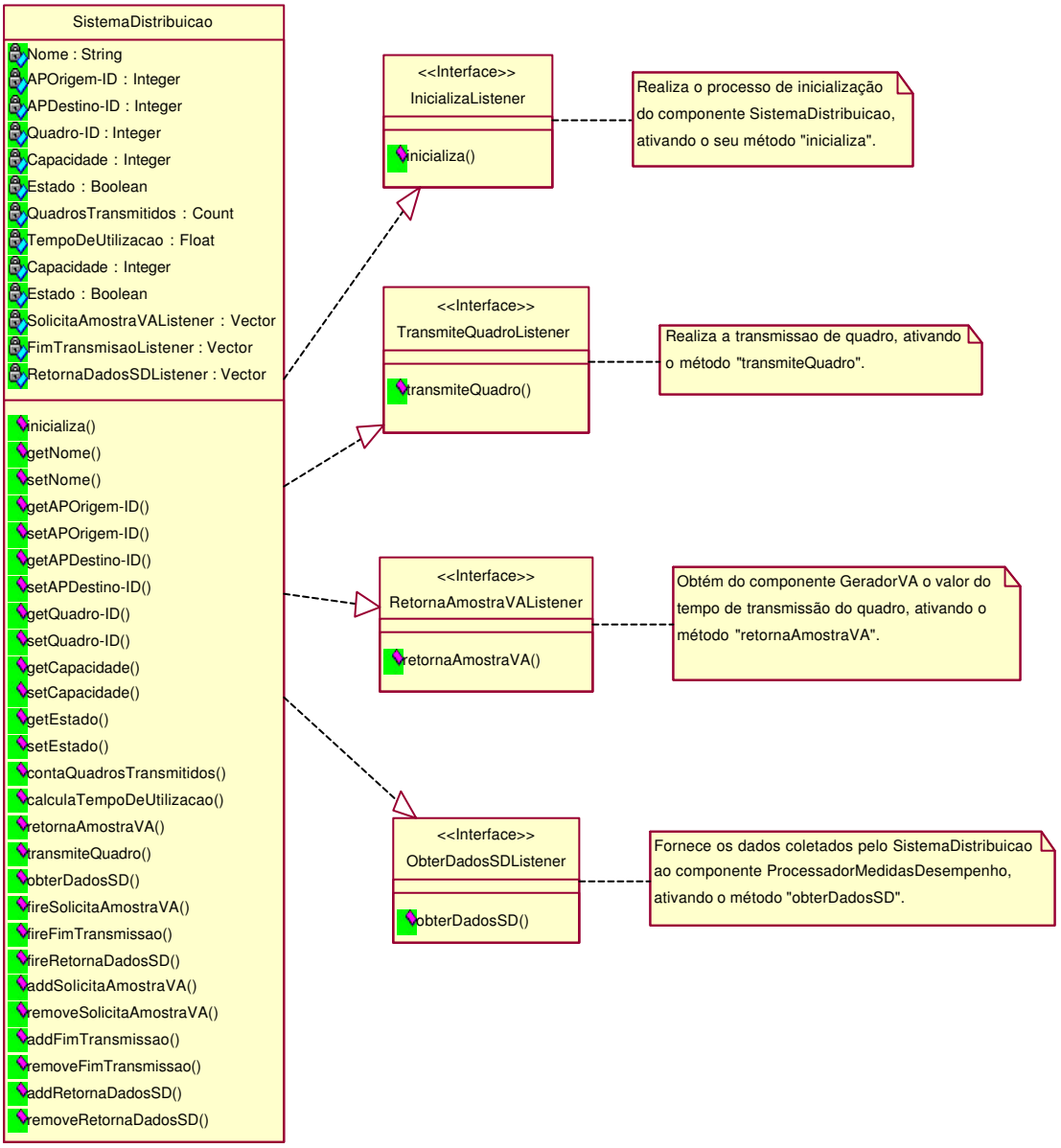

Figura 5.24 – Componente SistemaDistribuicao.

Vejamos a seguir a descrição dos atributos e dos principais métodos do componente SistemaDistribuicao.

Atributos do componente ***SistemaDistribuicao***

- **Nome:** representa a identificação do componente SistemaDistribuicao.
- **APOrigem-ID:** representa a identificação do componente PontoAcessoOrigem o qual o componente SistemaDistribuicao está associado.
- **APDestino-ID:** representa a identificação do componente PontoAcessoDestino o qual o componente SistemaDistribuicao está associado.
- **Quadro-ID:** representa a identificação do quadro.
- **TempoDeUtilizacao:** armazena o tempo total de utilização do componente SistemaDistribuicao.
- **Capacidade:** capacidade de transmissão do componente SistemaDistribuicao.
- **Estado:** estado do componente SistemaDistribuicao, se ocupado ou livre.
- **QuadrosTransmitidos:** armazena o número total de quadros transmitidos no componente SistemaDistribuicao.
- **SolicitaAmostraVAListener:** armazena os listerners do evento *SolicitaAmostraVAEvent*.
- **FimTransmissaoListener:** armazena os listeners do evento *FimTransmissaoEvent*.
- **RetornaDadosSDListener:** armazena os listeners do evento *RetornaDadosSDEvent*.

Métodos do componente ***SistemaDistribuicao***

- **inicializa():** método da *Interface* "InicializaListener", que realiza o processo de inicialização do componente SistemaDistribuicao.
- **contaQuadrosTransmitidos():** método usado para calcular o número total de quadros transmitidos no componente SistemaDistribuicao.
- **calculaTempoDeUtilizacao():** método usado para calcular o tempo total de utilização do componente SistemaDistribuicao.

- **retornaAmostraVA():** método da *Interface* "RetornaAmostraVAListener" que obtém do componente GeradorVA, o valor médio do tempo de transmissão de quadro no componente SistemaDistribuicao.
- **transmiteQuadro():** método da *Interface* "TransmiteQuadroListener" que executa a chegada de quadro para transmitir no componente SistemaDistribuicao.
- **obterDadosSD():** método da *Interface* "ObterDadosSDListener" usado para fornecer ao componente ProcessadorMedidasDesempenho, os dados coletados pelo componente SistemaDistribuicao.
- **fireSolicitaAmostraVA():** método usado para disparar o evento *SolicitaAmostraVAEvent*.
- **fireFimTransmissao():** método usado para disparar o evento *FimTransmissaoEvent*.
- **fireRetornaDadosSD():** método usado para disparar o evento *RetornaDadosSDEvent*.
- **addSolicitaAmostraVAListener():** método que fornece o cadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **removeSolicitaAmostraVAListener():** método que fornece o descadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **addFimTransmissaoListener():** método que fornece o cadastro de listener para o evento *FimTransmissaoEvent*.
- **removeFimTransmissaoListener():** método que fornece o descadastro de listener para o evento *FimTransmissaoEvent*.
- **addRetornaDadosSDListener():** método que fornece o cadastro de listener para o evento *RetornaDadosSDEvent*.
- **removeRetornaDadosSDListener():** método que fornece o descadastro de listener para o evento *RetornaDadosSDEvent*.

5.6.2.8 Componente PontoAcessoDestino

A Figura 5.25 ilustra o componente PontoAcessoDestino.

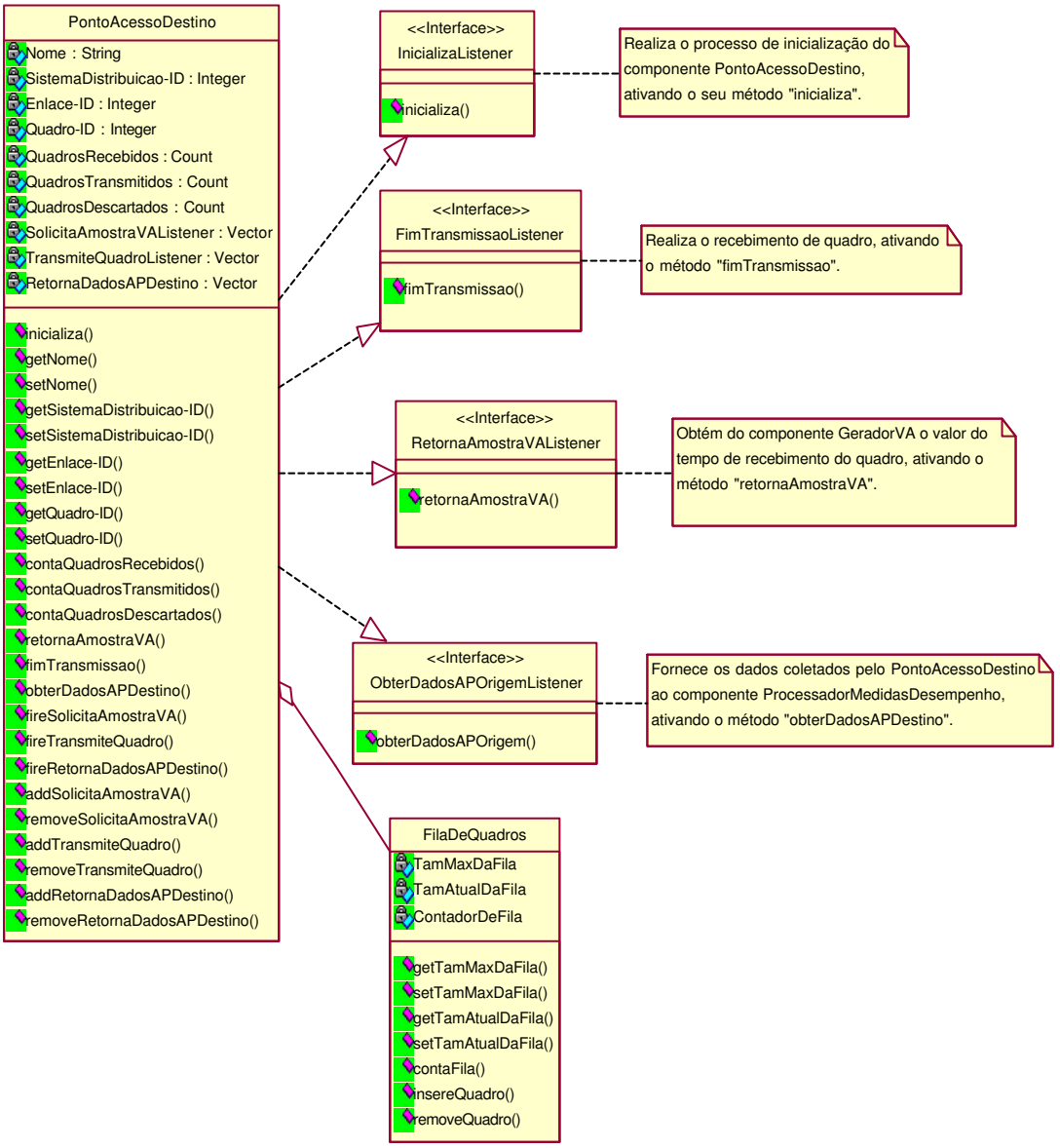

Figura 5.25 – Componente PontoAcessoDestino.

Vejamos a seguir a descrição dos atributos e dos principais métodos do componente PontoAcessoDestino.

Atributos do componente ***PontoAcessoDestino***

- **Nome:** representa a identificação do componente PontoAcessoDestino.
- **SistemaDistribuicao-ID:** representa a identificação do componente SistemaDistribuicao o qual o componente PontoAcessoDestino está associado.
- **Enlace-ID:** representa a identificação do componente Enlace o qual o componente PontoAcessoDestino está associado.
- **Quadro-ID:** representa a identificação do quadro.
- **QuadrosRecebidos:** armazena o número total de quadros recebidos pelo PontoAcessoDestino.
- **QuadrosTransmitidos:** armazena o número total de quadros transmitidos pelo PontoAcessoDestino.
- **QuadrosDescartados:** armazena o número total de quadros descartados pelo PontoAcessoDestino.
- **SolicitaAmostraVAListener:** armazena os listerners do evento *SolicitaAmostraVAEvent*.
- **TransmiteQuadroListener:** armazena os listeners do evento *TransmiteQuadroEvent*.
- **RetornaDadosAPDestinoListener:** armazena os listeners do evento *RetornaDadosAPDestinoEvent*.

Métodos do componente ***PontoAcessoDestino***

- **inicializa():** método da *Interface* "InicializaListener" que realiza o processo de inicialização do componente PontoAcessoDestino.
- **contaQuadrosRecebidos():** método usado para calcular o número total de quadros recebidos pelo componente PontoAcessoDestino.
- **contaQuadrosTransmitidos():** método usado para calcular o número total de quadros transmitidos pelo componente PontoAcessoDestino.

- **contaQuadrosDescartados():** método usado para calcular o número total de quadros descartados pelo componente PontoAcessoDestino.
- **fimTransmissao():** método da *Interface* "FimTransmissaoListener" que executa o recebimento de quadro pelo componente PontoAcessoDestino.
- **retornaAmostraVA():** método da *Interface* "RetornaAmostraVAListener" que obtém do componente GeradorVA, o valor médio do tempo de recebimento de quadro pelo componente PontoAcessoDestino.
- **ObterDadosAPDestino():** método da *Interface* "ObterDadosEDListener" usado para fornecer ao componente ProcessadorMedidasDesempenho, os dados coletados pelo componente PontoAcessoDestino.
- **fireSolicitaAmostraVA():** método usado para disparar o evento *SolicitaAmostraVAEvent*.
- **fireTransmiteQuadro():** método usado para disparar o evento *TransmiteQuadroEvent*.
- **fireRetornaDadosAPDestino():** método usado para disparar o evento *RetornaDadosAPDestinoEvent*.
- **addSolicitaAmostraVAListener():** método que fornece o cadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **removeSolicitaAmostraVAListener():** método que fornece o descadastro de listener para o evento *SolicitaAmostraVAEvent*.
- **addTransmiteQuadroListener():** método que fornece o cadastro de listener para o evento *TransmiteQuadroEvent*.
- **removeTransmiteQuadroListener():** método que fornece o descadastro de listener para o evento *TransmiteQuadroEvent*.
- **addRetornaDadosAPDestinoListener():** método que fornece o cadastro de listener para o evento *RetornaDadosAPDestinoEvent*.
- **removeRetornaDadosAPDestinoListener():** método que fornece o descadastro de listener para o evento *RetornaDadosAPDestinoEvent*.

5.6.2.9 Componente Sorvedouro

A Figura 5.26 ilustra o componente Sorvedouro.

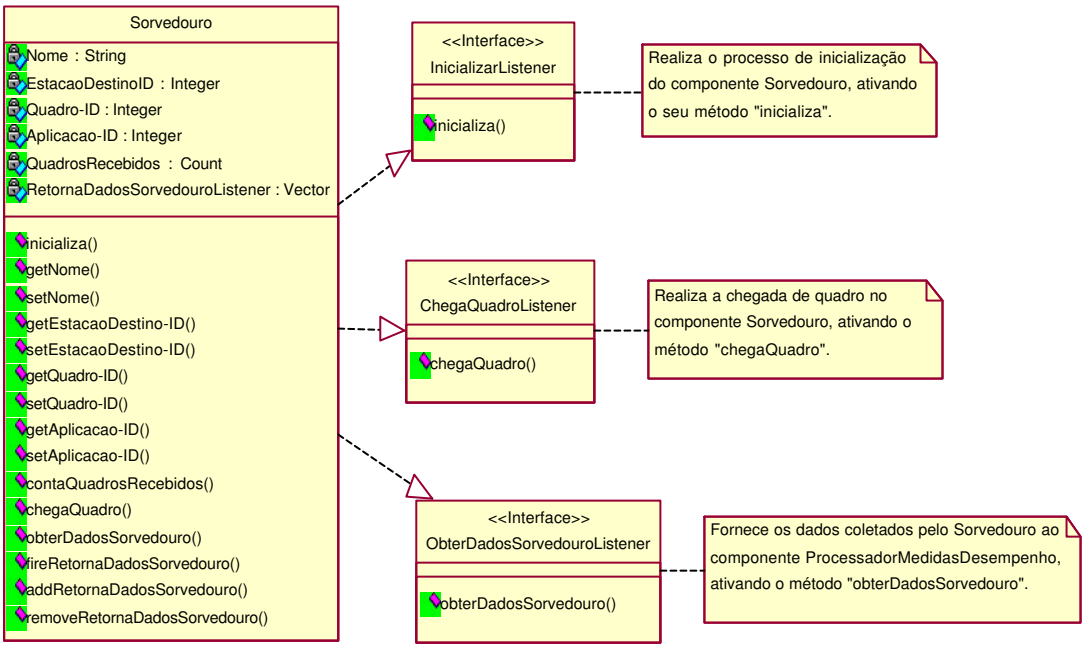

Figura 5.26 – Componente Sorvedouro.

Vejamos a seguir a descrição dos atributos e dos principais métodos do componente Sorvedouro.

Atributos do componente ***Sorvedouro***

- **Nome:** representa a identificação do componente Sorvedouro.
- **EstacaoDestino-ID:** representa a identificação do componente EstacaoDestino o qual o componente Sorvedouro está associado.
- **Quadro-ID:** representa a identificação do quadro.
- **Aplicacao-ID:** representa a identificação da aplicação a qual o quadro recebido se destina.

- **QuadrosRecebidos:** armazena o número total de quadros recebidos pelo Sorvedouro.
- **RetornaDadosSorvedouroListener:** armazena os listeners do evento *RetornaDadosSorvedouroEvent*.

Métodos do componente Sorvedouro

- **inicializa():** método da *Interface* "InicializaListener" que realiza o processo de inicialização do componente Sorvedouro.
- **contaQuadrosRecebidos():** método usado para calcular o número total de quadros recebidos pelo componente Sorvedouro.
- **chegaQuadro():** método da *Interface* "ChegaQuadroListener" que executa o recebimento de quadro pelo componente Sorvedouro.
- **obterDadosSorvedouro():** método da *Interface* "ObterDadosSorvedouroListener" usado para fornecer ao componente ProcessadorMedidasDesempenho, os dados coletados pelo componente Sorvedouro.
- **fireRetornaDadosSorvedouro():** método usado para disparar o evento *RetornaDadosSorvedouroEvent*.
- **addRetornaDadosSorvedouroListener():** método que fornece o cadastro de listener para o evento *RetornaDadosSorvedouroEvent*.
- **removeRetornaDadosSorvedouroListener():** método que fornece o descadastro de listener para o evento *RetornaDadosSorvedouroEvent*.

5.7 Contemplação de requisitos

Finalizamos a especificação dos componentes 802.11 verificando a contemplação dos requisitos funcionais e não funcionais considerados neste projeto. Essa contemplação se apresenta de maneira informal, uma vez que a validação de uma especificação reúne um conjunto extenso de outras atividades que fogem do escopo desta Dissertação, basicamente relacionadas à implementação.

RF	Contemplação
RF01	Os componentes 802.11 especificados representam os elementos essenciais para o funcionamento de uma rede sem fio 802.11, uma vez que a identificação dos componentes, a representação da colaboração entre eles através de eventos, os métodos Bean que cada um pode executar e disparar, os atributos que armazenam informações da simulação e o comportamento interno de cada componente, são projetados baseados no funcionamento das redes 802.11 apresentado no capítulo 3.
RF02	Os componentes FonteTrafego, EstacaoOrigem, CamadaMAC802.11, Enlace e EstacaoDestino, possibilitam a simulação de modelos de rede *ad hoc*.
RF03	Os componentes FonteTrafego, EstacaoOrigem, CamadaMAC802.11, Enlace, EstacaoDestino, PontoAcessoOrigem, SistemaDistribuicao, PontoAcessoDestino e Sorvedouro, possibilitam a simulação de modelos de rede multi-célula.
RF04	A possibilidade de múltiplas instâncias do componente FonteTrafego num mesmo modelo, identificado no modelo conceitual da subseção 4.4.2 e no diagrama de colaboração da subseção 5.4.2, viabilizam a simulação contendo várias FonteTrafego associadas a uma mesma EstacaoOrigem.
RF05	A possibilidade de múltiplas instâncias do componente EstacaoOrigem num mesmo modelo, identificado no modelo conceitual da subseção 4.4.2 e no diagrama de colaboração da subseção 5.4.3, viabilizam a simulação contendo várias EstacaoOrigem associadas a uma mesma CamadaMAC802.11.
RF06	O diagrama de colaboração da subseção 5.4.4 modela a ocorrência de colisão, quando duas ou mais EstacaoOrigem solicitam ao mesmo tempo a CamadaMAC802.11 a transmissão de quadros, representando, dessa forma, o compartilhamento do Enlace.
RF07	O acesso ao Enlace por parte da EstacaoOrigem e do PontoAcessoDestino, e do SistemaDistribuicao por parte do PontoAcessoOrigem, segue o princípio básico de acesso ao meio do protocolo CSMA/CA, apresentado na subseção 3.8.2, que impõe o compartilhamento do canal de comunicação.
RF08	Os diagramas de colaboração das Subseções 5.4.11 e 5.4.13, revelam a instância única para os componentes PontoAcessoOrigem e PontoAcessoDestino em cada modelo de rede multi-célula.

RF09 e RF10	Os diagramas de colaboração das Subseções 5.4.11, 5.4.12 e 5.4.13, revelam a existência de um único SistemaDistribuicao interligando os componentes PontoAcessoOrigem e PontoAcessoDestino em cada modelo de rede multi-célula.
RF11	Os componentes 802.11 armazenam durante a simulação dados e os fornecem ao componente ProcessadorMedidasDesempenho (componente do ambiente de simulação), para o cálculo das medidas de desempenho de interesse. Todos os diagramas de colaboração e os projetos detalhados dos componentes revelam isso.
RF12	A configuração da função de probabilidade usada pela componente GeradorVA (componente do ambiente de simulação), para fornecer aos componentes 802.11 as amostras necessárias para a simulação, pode ser feita pelo usuário de modelagem, bem como a definição de constantes para representar esses valores. Os projetos detalhados dos componentes 802.11 revelam isso.

Tabela 5.1 – Contemplação dos requisitos funcionais.

Segue a contemplação dos requisitos não funcionais.

RNF	Contemplação
RNF01	A utilização dos componentes 802.11 permite o desenvolvimento rápido de ferramentas de simulação, uma vez que eles foram especificados observando as características de um ambiente de simulação e a tecnologia JavaBeans, que oferece portabilidade e fácil integração.
RNF02	Da mesma forma com relação à contemplação do requisito RNF01, a utilização dos componentes 802.11 pode estender ferramentas já existentes.
RNF03	A especificação é feita seguindo um processo de desenvolvimento orientado a objetos conhecido, que usa UML como recurso de modelagem. Portanto, a documentação apresentada é naturalmente entendível pelo usuário do sistema, o desenvolvedor de ferramentas de simulação.
RNF04	A especificação adota UML, oferecida pelo processo de desenvolvimento. Todos artefatos são elaborados baseados na UML.
RNF05	A especificação adota o processo de desenvolvimento proposto por Craig Larman [Larman98], que possui uma abordagem orientada e objetos e usa UML como recurso de modelagem.

Tabela 5.2 – Contemplação dos requisitos não funcionais.

Os requisitos não funcionais RNF03, RNF04 e RNF05 se confundem um pouco, mas possuem uma sutil diferença. O RNF03 exige uma documentação que seja naturalmente entendida pela desenvolvedor de ferramentas de simulação. Os RNF04 e RNF05 exigem que recursos da Engenharia de Software sejam usados para garantir o RNF03, através do uso de UML e de um processo de desenvolvimento, respectivamente.

Capítulo 06

Conclusão

6.1 Introdução

Este capítulo apresenta a conclusão desta Dissertação, revelando sua contribuição e apresentado sugestões para trabalhos futuros.

6.2 Conclusões/Contribuições

Esta Dissertação de Mestrado apresenta uma especificação de componentes de software para ser utilizada na construção de ferramentas de simulação para a modelagem e avaliação de desempenho de sistemas de redes locais de computadores sem fio padrão 802.11.

Essa especificação apresenta recursos das fases de análise e de projeto, conforme o processo de desenvolvimento proposto em [Larman98], que adota UML (*Unified Modeling Language*) como recurso de modelagem. Com os recursos de análise (fase de análise), identificamos os componentes 802.11. Com os recursos de projeto (fase de projeto), apresentamos a solução para a construção desses componentes.

Na fase de projeto, a especificação mostra a colaboração entre os componentes 802.11 e, destes, com os componentes do ambiente de simulação, e as propriedades e métodos dos componentes 802.11, representados em forma de classes e interfaces de software. É a partir disso que cada componente pode ser codificado, mapeando seu conteúdo para atributos e operações numa linguagem de programação orientada a objetos, seguindo um modelo de componente. O projeto detalhado apresentado baseia-se no modelo de componentes JavaBeans.

A tecnologia de componentes JavaBeans direciona a futura implementação dos componentes 802.11 para a linguagem de programação Java. Os componentes, quando implementados, poderão ser manipulados através de uma ferramenta de composição visual de aplicações para formar a ferramenta de simulação pretendida.

Os componentes propostos representam os elementos essenciais de modelos de redes sem fio *ad hoc* e multi-célula, quais sejam: Fonte de Tráfego, Estação Origem, Camada de Acesso ao Meio e seu Protocolo, Enlace, Estação Destino, Ponto de Acesso Origem, Sistema de Distribuição, Ponto de Acesso Destino e Sorvedouro.

A especificação de componentes de software apresentada nesta Dissertação permite a construção de ferramentas de simulação explorando a reusabilidade de software. Dessa forma, o desenvolvedor de ferramentas de simulação pode utilizar essa especificação e implementar esses componentes.

Para viabilizar a especificação dos componentes 802.11 foi necessário estudar a técnica da simulação digital e as tecnologias de redes sem fio. Entendimento de um processo de simulação e conhecimentos de ferramentas de simulação também foram necessários. Na fundamentação das redes 802.11, foram contextualizados os principais elementos e conceitos que fazem parte de uma rede sem fio padrão 802.11, tais como elementos da arquitetura (sistema de distribuição, pontos de acesso, células, etc), e tipos de redes sem fio (redes *ad hoc* e multi-célula).

Esses dois contextos, simulação digital e redes 802.11, foram de extrema importância na especificação dos componentes 802.11. Todavia, um outro assunto também recebeu investimento considerável: o processo de desenvolvimento de software adotado. Com esse processo, técnicas eficientes de Engenharia de Software puderam ser usadas, através do uso de UML, dos recursos de análise e projeto oferecidos pelo processo de desenvolvimento e das vantagens da tecnologia de orientação a objetos. Adicionalmente, para viabilizar a fase de projeto, estudos sobre a tecnologia JavaBeans também foram necessários.

A estrutura e conseqüente documentação da especificação também recebeu cuidados de planejamento e elaboração. A estrutura segue um raciocínio sistemático capaz de representar (documentar) a especificação proposta. Nessa documentação, o contexto sobre a simulação digital é explorado e logo em seguida a fundamentação 802.11 é apresentada. Depois, seguem as fases de análise e de projeto, que representam a especificação dos componentes 802.11.

Na prática, as narrativas, diagramas, tabelas, modelos, figuras, foram sistematicamente sendo elaborados para de fato representar a especificação proposta de forma a ser entendida numa ótica natural (através dos recursos de análise) e numa ótica mais técnica (através dos recursos de projeto). Portanto, investimos na arte de documentar, "princípio fundamental para elaboração de qualquer projeto de software" [D'Souza98].

Finalmente, a especificação apresentada pode ser usada na construção de novas ferramentas de simulação como também na extensão de ferramentas já existentes.

6.3 Sugestões para trabalhos futuros

Trabalhos futuros podem investir na implementação dos componentes ora especificados, visando a construção de um ambiente de simulação para modelos de redes sem fio *ad hoc* e multi-célula padrão 802.11.

O refinamento da especificação realizada nesta Dissertação, para atender outras funcionalidades dos sistemas de comunicação sem fio 802.11, pode ser feito sem demandar grandes esforços, uma vez que as funcionalidades essenciais já são aqui apresentadas. Funções de mobilidade, tratamento de estações perdidas (*hidden node – HN*) e a modelagem do protocolo de acesso ao meio CSMA/CA com maior detalhamento, são exemplos de extensões dessa especificação que podem ser contempladas.

Referências bibliográficas

[3Com00] 3Com. "IEEE 802.11b Wireless LANs". Papers, www.3com.com, April 25, 2000.

[Alencar98] Alencar, Marcelo Sampaio de Alencar. "Telefonia Digital". São Paulo, Érica, 1998.

[AltaGroup96] Alta Group, "BONES Designer User's Guide", Alta Group of Cadence Design System, Inc., 1996.

[Alves00] Alves, Carlos Alessander Costa. "Template para Modelagem de Redes ATM no Ambiente Arena", Proposta de Dissertação de Mestrado, UFPB, 2000.

[Booch98] Booch, Grady. "The Unified Modeling Language User Guide", Addison Wesley Reading, MA, 1998.

[Cabral95] Cabral, Maria Izabel. "SAVAD – Um Ambiente de Simulação Inteligente para Modelar Sistemas de Redes de Fila / Maria Izabel Cabral, Stanley R. M. Oliveira, Edílson Ferneda, Marcos A. G. Brasileiro", Artigo, Brasil, 1995.

[Cabral98] Cabral, Maria Izabel Cavalcanti. "Métodos de Avaliação de Redes de Computadores", Material de Aula da Disciplina Avaliação de Desempenho de Redes de Computadores, UFPB, CCT, Campina Grande, 1998.

[Camara00] Câmara, Daniel Câmara. "Proposta para Cobertura de Área de Sombra em Redes Wireless", UFMG, 2000.

[Celestino90] Celestino, J. "Avaliação de Desempenho de Redes Locais Brasileiras", Dissertação de Mestrado, CCT, UFPB, Campina Grande, 1992.

[D'Souza98] D'Souza Desmond, F.; Wills, Alan C.; "Objects Components and Framework with UML: The Catalysis Approach"; Addison-Wesley, 1998.

[Damoun79] Damoun, F., Kleinrock L. "Stochastic Perfomance Evaluation of Hierarchical Routing for Large Networks", Computer Network, vol. 3, 1979.

[Dayem00] Dayem, Rifaat A. "Mobile Data Wireless LAN Technologies". U.S.A, PTRPH, 2000.

[Dias92] Dias, Maria Madalena. "SIMILE – Um Simulador Reutilizável para Avaliação de Desempenho de Redes Locais", Dissertação de Mestrado, CCT, UFPB, Campina Grande, 1992.

[Donaduzzi00] Donaduzzi, Daniel Angelo. "Simuladores de Redes", Tópicos Avançados em Redes e Internet, UNISINOS, 2000.

[Eckel00] Eckel, Bruce, "Thinking in Java", 2a edição, Revision 12, New Jersey, Prentice-Hall, 2000.

[Englander97] Englander, Robert. "Developing Java Beans", O' Reilloy & Associates, Inc., 1997.

[Eriksson98] Eriksson, H. E. "UML Toolkit". New York, John Wiley, 1998.

[Freire00] Freire, Raissa Dantas. "Especificação de um Framework Baseado em Componentes de Software Reutilizáveis para Aplicações de Gerência de Falhas em Redes de Computadores", Dissertação de Mestrado, UFPB, 2000.

[GoF95] GoF, Gang of Four. "Design Patterns: Elements of Reusable Object-Oriented Software", Gamma, Helm, Johnson e Vlissides. Addison-Wesley, 1995.

[IEEE802.11a] IEEE Doc. IEEE P802.11-96/49C. "802.11 Tutorial – 802.11 MAC Entity: MAC Basic Access Mechanism Privacy and Access Control". U.S.A., 1996.

[IEEE802.11b] IEEE Standard 802.11. "The IEEE 802.11 Standard". U.S.A., 1997.

[IEEE802.11c] IEEE Standard 802.11. "The IEEE 802.11b Standard". U.S.A., 1998.

[Jacobson99] Jacobson, et al. "The Unified Software Development Process". Addison-Wesley, 1999.

[Jones00] Jones, Meilir Page. "Fundamentals of Object-Oriented Design in UML", Addison-Wesley, 2000.

[Kamiensk96] Kamienski, Carlos Alberto. "Introdução ao Paradigma de Orientação a Objetos". Artigo, UFPB, 1996.

[Kelton98] Kelton, W. David. "Simulation With Arena". WCB/McGraw-Hill, 1998.

[Kleironck75] Kleironck, Leonard. "Queueing System – Volume I: Theory", Los Angeles, John Wiley & Sons, 1975.

[Kleironck76] Kleironck, Leonard. "Queueing System – Volume II: Computer Aplication", Los Angeles, John Wiley & Sons, 1976.

[Kobryn00] Kobryn, Cris. "Modeling Components and Frameworks with UML", Communication of the ACM, 43-10, 2000.

[Larman98] Larman, Craig; "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design"; Pretence-Hall, 1998.

[Lula00] Lula, Juliana Camboim Lopes Andrade. "Desenvolvimento de Componentes para Simuladores de Redes de Computadores", Artigo, UFPB, 2000.

[Lula01] Lula, Juliana Camboim Lopes Andrade. "Especificação de Componentes para um Ambiente de Simulação de Rede TCP/IP" Dissertação de Mestrado, UFPB, 2001.

[Lula01b] Lula, Juliana Camboim Lopes Andrade. "Especificação de Componentes para Construção de Ferramentas de Simulação". Jornada Chilena de Computação, Punta Arenas, Paper, 2001.

[Moura84] Moura, José Antão Beltrão. "Técnica de Simulação Digital / J. F. Sauvé, W. F. Giozza e J. F. M. Araújo", DSC, UFPB, 1984.

[Neto99] Neto, Gino Olivato. "Redes de Computadores Sem Fio", Artigo, 1999.

[NS01] UCB/LBNL/VINT "Network Simulator – NS" (version 2): http://www.isi.edu/nsnam/ns/, dez 20001.

[NS98] NS, "Network Simulator", Projeto VINT, E.U.A, 1998.

[Oliveira95] Oliveira, S. R de M. "ALLOS – Uma Ferramenta para Solucionar Modelos de Redes de Filas Usando Cadeias de Markov", Dissertação de Mestrado, CCT, UFPB, 1995.

[OPNET02] OPNET, Modeler. "Network Simulator", http://www.opnet.com/products/modeler/home.html, Ago 2002.

[Paragon99] Paragon "Arena – Introdução a Simulação com Arena", São Paulo, 1999.

[Pedgen97] Pedgen, C. Dennis. "Introduction to Simulation Using SIMAN", 1997.

[Pressman95] Pressman, Roger S. "Engenharia de Software". São Paulo, Makron Books, 1995.

[PtolemyII02] "Ptolemy II – Heterogeneous Concurrent Modeling and Design in Java", University of California at Berkeley, http://ptolemy.eecs.berkeley.edu, 2002.

[Rocha02] Rocha, Flávio Gonçalves. "Componentes de Software para a Construção de Ferramentas de Simulação / Juliana Camboim Lula, Maria Izabel Cavalcanti Cabral". Artigo, UFPB.

[Sauvé00] Sauvé, Jacques P. "Análise e Projeto de Sistema Orientado a Objetos", Material de Aula da Disciplina de Mestrado APOO, UFPB, COPIN, Campina Grande, 2000.

[Schneider00] Schneider, "Applying Use Cases", Schneider e Winters, Addison-Wesley, 1998.

[Silva00] Silva, Ricardo Pereira. "Suporte ao Desenvolvimento e Uso de Framework e Componentes"; Tese de Doutorado, UFRGS/II/PPGC, Porto Alegre-RS, 2000.

[SimATM02] SimATM, "Projeto SimATM", UNICAMP, Campinas, http://sheratan.mc21.fee.unicamp.br/simatm, Ago 2002.

[Soares90] Soares, Luiz Fernando Guimarães. "Modelagem e Simulação Discreta de Sistemas", IME-USP, 1990.

[Soares95] Soares, Luiz Fernando G. "Redes de Computadores: das LANs, MANs às redes ATM / Luiz Fernando Gomes Soares, Guido Lemes, Sérgio Colcher". Rio de Janeiro, Campus, 1995.

[Sommerville00] Sommerville, Ian. "Engineering Software", 6th Edition, Addison Wesley, 2000.

[Souto93] Souto, Francisco de Assis Coutinho. "SAVAD – Sistema de Avaliação de Desempenho de Modelos Redes de Filas", UFPB, 1993.

[Sun02] Sun Microsystems Inc, "Java Beans Specification", 2002. Disponível em http://www.sun.com/beans.

[Takus97] Takus, David A. "Arena Software Tutorial", Pensylvania, System Modeling Corporation, 1997.

[Tanenbaum96] Tenenbaum, Andrew S. "Computer Networks", 3rd. Ed., Prentice-Hall, 1996.

[Torres01] Torres, Gabriel. "Redes de Computadores Curso Completo", Axcel Books, 2001.

[UFRGS00] UFRGS, Grupo de Redes, "Transmissão de Dados Sem Fio", UFRGS, 2000.

[UML97] UML. "The UML Logo and Objectory are Trademarks of Rational Software Corp", Rational Software, USA, 1997. www.rational.com .

[Vasconcelos02] Vasconcelos, Geovane Vitor. "Components Specification for Modeling Wireless IEEE 802.11 Networks / Maria Izabel Cavalcanti Cabral, Joberto Sergio Barbosa Martins". Paper, ICCON-WSEAS, 2002.

[Wagner00] Wagner, Marcus Vinícius da Silva. "Especificação de Componentes para a Simulação de Redes TCP/IP", Dissertação de Mestrado, UFPB, 2000.

[Zacker00] Zacker, Craig. "Redes de Computadores: Configuração, Manutenção e Expansão / Craig Zacker, Paul Doyle", São Paulo, Makron, 2000.

[Zanetti99] Zanetti, Alberto René. "Redes Locais Sem Fio / Alberto René Zenetti, Leandro de Carvalho Gonçalves". Pesquisa de Mestrado, UFSCar, 1999.

Apêndice A

Processo de Desenvolvimento

1. Introdução

Uma especificação de sistema de software deve seguir um processo de desenvolvimento, usando com critérios os recursos oferecidos por esse processo [Pressman95]. Este apêndice apresenta os recursos de análise e projeto do processo de desenvolvimento proposto por [Larman98], adotado nesta Dissertação para a especificação dos componentes 802.11.

O processo de [Larman98] possui uma abordagem orientada a objetos e é dividido basicamente em três fases: análise, projeto e codificação. A fase de codificação não é abordada aqui, uma vez que esta Dissertação não aborda implementação.

Da mesma forma que outros processos de desenvolvimento de sistemas, os recursos do processo de [Larman98] atendem uma quantidade variada e complexa de sistemas do mundo real que podem ser transformados em programas de computador. Com isso, escolhemos sistematicamente os recursos do processo, definindo a seqüência típica, a forma e o grau de profundidade em que eles são usados, para que, dessa forma, esses recursos possam representar a especificação dos componentes 802.11.

A perspectiva de objetos que é oferecida pelos processos de desenvolvimento atuais, a exemplo do processo de [Larman98], facilita naturalmente a construção dos componentes 802.11, uma vez que a especificação de sistemas baseados em componentes segue os mesmos princípios de uma especificação de sistemas orientados a objetos. Essa característica revela uma forte semelhança entre esses dois paradigmas, e considera a tecnologia de componentes uma evolução natural da tecnologia de orientação a objetos [Jones00].

Além de apresentar os recursos do processo de desenvolvimento adotado, este apêndice aborda a linguagem UML e a programação orientada a objetos. Conceitos sobre componentes de software são apresentados no capítulo 5, quando da elaboração da fase de projeto da especificação.

2. Fase de análise

A fase de análise representa uma atividade de investigação que se propõe levantar os requisitos do sistema. Nessa fase, a questão se concentra em definir *o que* deve ser feito para se construir o sistema [Larman98], a partir de um escopo bem definido do problema que está sendo analisado (domínio do problema).

Para que as funcionalidades do sistema sejam satisfeitas, é preciso identificar claramente os elementos que as executam. Esses elementos são denominados de conceitos do domínio do problema, e são descobertos numa perspectiva de objetos.

A especificação dos componentes 802.11 investe bastante na utilização desta fase de análise, pois o sucesso da especificação depende basicamente do levantamento e apresentação dos requisitos necessários à modelagem das redes sem fio, o que conseqüentemente reflete no projeto. Para isso, vários recursos da fase de análise são usados. A seguir apresentamos esses recursos.

2.1 Levantamento de requisitos

O levantamento de requisitos relaciona todos as funcionalidades definidas para o sistema. Trata-se de um documento a partir do qual são definidos o que se pretende construir. Sucesso nesse levantamento garante um ótimo projeto de software [Sauvé00]. O processo de desenvolvimento adotado sugere uma fase preliminar que envolve planejamento e elaboração, onde são levantados esses requisitos.

Basicamente os requisitos considerados são "funcionais", ou seja, dizem respeito ao funcionamento do sistema. Além disso, um outro tipo de requisito também pode ser considerado, o requisito "não funcional". Este tipo de requisito revela questões que não fazem parte diretamente das operações do sistema, e sim de questões que envolvem o contexto, como por exemplo, para qual plataforma de software ou hardware o sistema será construído.

Uma outra classificação é atribuída apenas aos requisitos funcionais: *evidentes*, *ocultos* e *opcionais*. Nos requisitos evidentes, o usuário tem a certeza de que a funcionalidade irá ser executada pelo sistema. Nos requisitos ocultos, a funcionalidade ocorre, mas não é percebida pelo usuário. Por fim, nos requisitos opcionais, as funcionalidades não são relevantes ao sistema [Larman98].

Vejamos na Tabela 4.1 um modelo para o relacionamento dos requisitos do sistema.

Requisito	Descrição	Tipo
Numeração dos requisitos (R01, R02, ..., R*n*)	Descrição individual do requisito, expressando uma funcionalidade que se deseja do sistema.	Evidente, oculto ou opcional.

Tabela 2.1 –Levantamento de requisitos.

Para que uma funcionalidade do sistema seja contemplada, é necessário que objetos (objetos na fase de análise são chamados de conceitos [Larman98]) sejam manipulados internamente no sistema. A identificação desses conceitos aqui na fase de análise é fundamental. Para isso, várias técnicas podem ser usadas.

Uma técnica muita usada e que tem contribuído significativamente na descoberta desses conceitos é a técnica de caso de uso (*use case*), abordada na próxima subseção.

2.2 Técnica de caso de uso (use case)

A técnica de "caso de uso" (use case) ajuda no levantamento dos requisitos funcionais, e pode ser usada para identificar os conceitos que fazem parte do sistema. Sua elaboração é relativamente simples: são descrições de casos de uso a cerca do sistema que se pretende construir [Schneider98].

Para cada suposta funcionalidade do sistema, criamos um caso de uso que irá explorar como essa funcionalidade será executada pelo sistema. Um caso de uso pode apenas conseguir apresentar uma funcionalidade de forma genérica. Nesse caso, o caso de uso é denominado de caso de uso de "alto nível". Para detalhar mais cada funcionalidade e obter uma compreensão mais profunda, usamos um outro tipo de caso de uso, o caso de uso "expandido".

A Tabela 4.2 sugere como um caso de uso deve ser elaborado, modelo que serve tanto para os casos de uso de alto nível como para os casos de uso expandidos. Um caso de uso expandido é criado a partir de um caso de uso de alto nível. Não são todos os casos de uso de alto nível que têm seus casos de uso expandidos, apenas aqueles mais relevantes ou que mereçam um detalhamento mais acentuado [Larman98].

Caso de Uso	Nome do caso de uso, que representa a funcionalidade que se deseja descrever, geralmente iniciada por um verbo que revela uma ação.
Atores	São os usuários ou sistemas externos que fazem parte da funcionalidade, ou seja, que interagem com o sistema para executar a referida funcionalidade.
Tipo	Representa uma classificação sistemática do caso de uso em primário, secundário e opcional. O primeiro, indica uma funcionalidade essencial do sistema. O segundo, indica uma funcionalidade que dificilmente ocorre no sistema, mas que é importante. O terceiro, indica uma funcionalidade que pode acontecer, mas que não é considerada relevante para o sistema.
Descrição	Esse item é o mais importante da técnica, pois é aqui que é descrita como as funcionalidades são executadas. É a narração feita pelo usuário do sistema de como é ou será a funcionalidade.

Tabela 2.2 – Descrição de caso de uso.

Uma outra ilustração também importante que o caso de uso apresenta é uma visão gráfica da interação dos usuários com o sistema. Essa interação é mostrada através de um diagrama conhecido como diagrama de caso de uso (Figura 2.1).

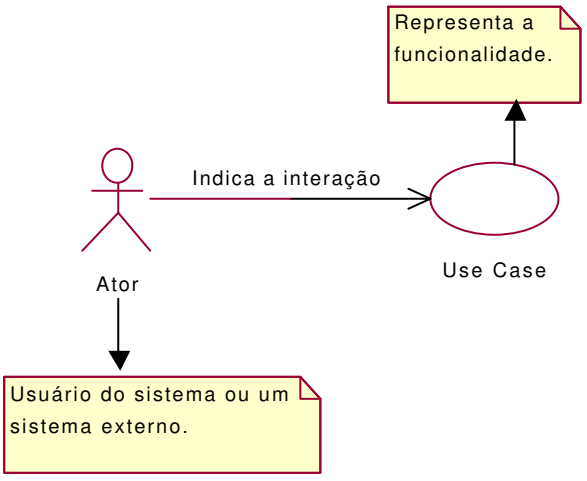

Figura 2.1 – Diagrama de caso de uso.

A identificação dos conceitos através da técnica de caso de uso é feita isolando e analisando os substantivos existentes nas narrações dos casos de uso feitos no item *descrição* de cada caso de uso. Além disso, outras formas de encontrar conceitos podem ser usadas, através de uma lista de idéias ou de coisas do mundo real que podem ser conceitos [Sauvé00]:

• Objetos físicos ou tangíveis. • Especificação ou descrição de coisas. • Lugares. • Transações. • Detalhe de uma transação. • Papéis de uma pessoa. • Coleções de coisas • Coisas dentro de coleções.	• Outros sistemas externos ao sistema. • Conceitos abstratos. • Organizações. • Eventos. • Diretivas. • Catálogos. • Manuais e livros. • Idéias.

Apesar do uso dessa lista contribuir no levantamento de quase todo tipo de conceito que um sistema pode ter, a identificação de conceitos feitos com uso dos casos de uso é a forma mais explorada, uma vez que substantivos freqüentemente aparecem nas narrações dos casos de uso [Sauvé00]. Esses conceitos viabilizam a elaboração do modelo conceitual, abordado a seguir.

2.3 Modelo conceitual

O modelo conceitual ilustra os conceitos do domínio do problema e é considerado o artefato (documento) mais importante da fase de análise [Larman98]. Esse modelo pode ser construído numa versão sumária e ser refinado até atingir uma versão que possa ilustrar todos os conceitos existentes, suas associações e principais atributos.

Na fase de análise, os conceitos não são pensados como elementos de software (objeto, classe, interface ou componente), mas como elementos levantados através do uso da linguagem natural do mundo real, não mencionando quaisquer terminologias ou tecnologias da informática.

Para a elaboração do modelo conceitual temos as seguintes etapas [Larman98]:

- Levantamento dos conceitos do sistema.
- Identificação dos principais atributos dos conceitos.
- Identificação dos relacionamentos (associações) entre os conceitos.
- Identificação das multiplicidades entre os conceitos.

A identificação das operações dos conceitos pode ser explorada. Porém, não é recomendada na fase de análise a realização dessa tarefa, evitando conflito de idéias, uma vez que a identificação dessas operações é mais eficiente quando da elaboração da fase de projeto.

Depois da realização das etapas supracitadas, o modelo conceitual é graficamente representado da forma mostrada na Figura 2.2.

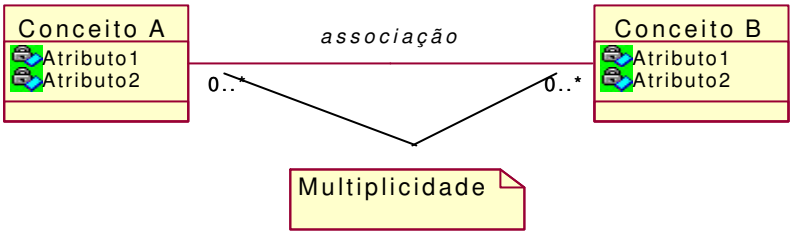

Figura 2.2 – Modelo conceitual.

A multiplicidade representa o número de instâncias possíveis numa relação. Um conceito pode promover nenhuma instância (zero), exatamente algumas instâncias (1, 2, 5, 10), uma quantidade cronológica crescente de instâncias (de 1 a 5), ou um número indefinido de instâncias (*).

A relação indica o relacionamento com significado entre conceitos [Larman98]. A leitura da relação é feita sempre de cima para baixo e da esquerda para a direita.

O mais importante do modelo conceitual é que a partir dele podemos entender o funcionamento geral do sistema, mesmo que estaticamente. Com isso, podemos observar a contemplação das funcionalidades exigidas pelo sistema, numa visão de alto nível, onde são envolvidos todos os elementos que fazem parte do sistema, com suas principais características.

Mas não é só isso. Na fase de projeto, o modelo conceitual viabiliza a elaboração do diagrama de colaboração, que vai representar a colaboração entre os elementos do sistema. Essa informação revela a importância do modelo conceitual, e indica a necessidade de uma atenção especial na realização das etapas para abstração e elaboração do modelo conceitual.

2.4 Diagrama de seqüência

Apesar do conjunto rico e importante de informações que são apresentadas no modelo conceitual, ele é um documento estático. Com isso, necessitamos mostrar a dinâmica do sistema. Para isso, usamos os diagramas de seqüência.

Os diagramas de seqüência são documentos criados a partir dos diagramas de casos de uso, ambos elaborados na fase de análise. A necessidade de usar os casos de uso para construir os diagramas de seqüência é devido às ilustrações de interações que os uses cases apresentam [Larman98].

Essas interações são detalhadas nos diagramas de seqüência através de uma seqüência de eventos que ocorrem no tempo, necessários para a realização de uma ação. Com isso, conseguimos visualizar a dinâmica no sistema.

Os eventos revelados no diagrama de seqüência são os estímulos (ações de entrada) feitos pelos atores externos do sistema, para realizar uma ação. O importante no diagrama de seqüência é identificar os eventos e respeitar a ordem em que eles são realizados [Larman98].

Com isso, conseguimos abstrair o que deve ser exigido do sistema, mesmo que não saibamos ainda como o sistema irá responder a esses estímulos, uma vez que o sistema é visto, neste momento do processo de desenvolvimento, como uma caixa preta [Sauvé00]. A Figura 4.3 apresenta um exemplo do diagrama de seqüência.

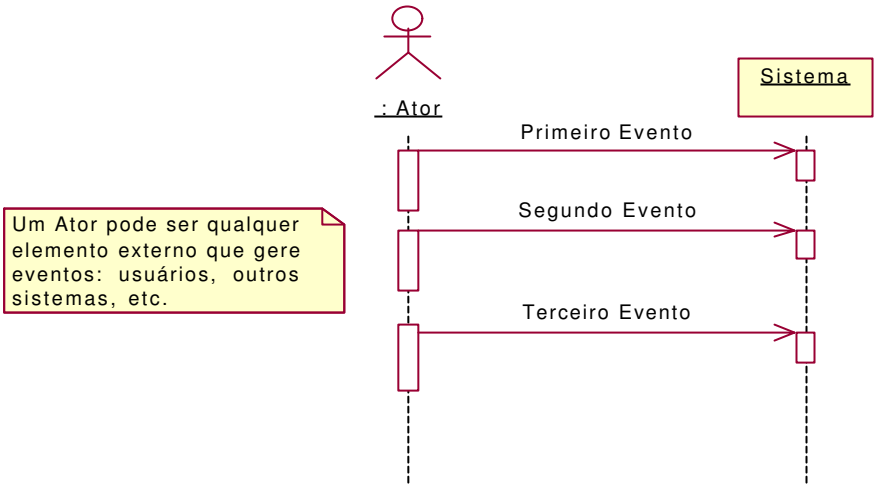

Figura 2.3 – Exemplo do diagrama de seqüência.

2.5 Contratos

Reconhecemos que os diagramas de seqüência apresentam os eventos no sistema. Teoricamente, para cada evento, o sistema irá realizar alguma operação interna. Essas operações são naturalmente descobertas e descritas em documentos que chamamos de contratos.

Um contrato é o registro do que uma operação promete cumprir [Larman98]. Olhamos ainda para o sistema como uma caixa preta, já que essas operações expressam funções internas do sistema como respostas para os eventos dos atores externos, e não como essas funções são executadas.

Da mesma forma como o modelo conceitual, os contratos são muitos importantes na fase de projeto, na elaboração dos diagramas de colaboração, que são diagramas que vão permitir a identificação das mensagens (eventos) de interação entre os elementos (componentes). Um modelo padrão para elaboração dos contratos é apresentado na Tabela 2.3 [Larman98].

Item	Finalidade/Descrição
Nome	Nome do evento que originou o contrato.
Responsabilidade	O que o sistema deve fazer para responder ao evento.
Referencias Cruzadas	Números das referências que o contrato atende, mesmo que parcialmente. Essas referências dizem respeito aos requisitos funcionais do sistema.
Notas	Indicação de uma informação importante para o projeto, particularmente relacionada a alguma lógica do sistema.
Exceções	São situações excepcionais que devem ser tratadas caso algum evento não possa ser atendido.
Saída	São mensagens ou registros enviados para fora do sistema, ou seja, para o usuário.
Pré-condições	Representam as condições que o sistema se encontra antes da execução da operação.
Pós-condições	Representam as condições do sistema após a execução da operação.

Tabela 2.3 – Modelo de um contrato.

Nem todos os itens de um contrato são necessários, embora sejam fortemente recomendados os itens *responsabilidade* e *pós-condições* [Larman98]. O primeiro, pelo fato de ser a essência do contrato, ou seja, de representar a promessa da execução das operações que vão atender aos respectivos eventos. O segundo, por representar uma mudança de estado do sistema, condição fundamental para se analisar o comportamento de um sistema [Larman98].

2.6 Resumo da fase de análise

A fase de análise oferece recursos que vão identificar as funcionalidades do sistema e os elementos que fazem parte dessas funcionalidades, numa visão conceitual e numa linguagem natural, sem a preocupação de tratar de questões relacionados com software.

Para isso, usamos os seguintes recursos: levantamento de requisitos do sistema, técnica de caso de uso, diagrama de caso de uso, modelo conceitual, diagrama de seqüência e contratos. Vemos a seguir os recursos escolhidos da fase de projeto do processo adotado, usados na especificação dos componentes 802.11.

3. Fase de projeto

A fase de projeto traduz os conceitos levantados na fase de análise e os representam como elementos de software. Nessa fase são definidas a arquitetura que o sistema terá e as operações internas que cada elemento irá executar. A questão nessa fase é *o como* fazer para se construir o sistema [Larman98].

A fase de projeto é dependente da fase de análise, o que significa dizer que uma análise bem feita implicará num projeto que contemple eficientemente todos os requisitos funcionais do sistema.

Da mesma forma como na análise, vários recursos da fase de projeto são usados na especificação dos componentes 802.11. Vejamos a seguir os recursos de projeto escolhidos do processo de [Larman98] para essa especificação.

3.1 Diagramas de colaboração

Os diagramas de casos de uso e de seqüência, documentos feitos na fase de análise, revelam interações com o sistema. O primeiro apresenta uma visão macro de cada interação que um ator externo (usuário ou sistema externo) faz com o sistema, visando executar uma funcionalidade. O segundo, o diagrama de seqüência, detalha essas interações através de uma seqüência de eventos que ocorrem no tempo.

Já sabemos que esses eventos geram operações no sistema, e que essas operações são descritas nos contratos, documento que também faz parte da fase de análise. Os contratos são alicerçados por pós-condições, que amarram o estado do sistema após a execução de uma operação. Mas os contratos não mostram uma solução de como os elementos de software procedem para satisfazer essas pós-condições [Larman98].

Portanto, precisamos ilustrar o comportamento do sistema, através da representação da colaboração entre os objetos, para atender as funcionalidades do sistema. Para isso, usamos um outro diagrama de interação, agora na fase de projeto, chamado de "diagrama de colaboração". O diagrama de colaboração ilustra como os elementos de software interagem através de mensagens para cumprir tarefas" [Larman98].

Para construirmos os diagramas de colaboração precisamos basicamente de três documentos de análise: o modelo conceitual, os contratos e os casos de uso. No modelo conceitual, definimos os elementos de software que vão participar das interações ilustradas nos diagramas de colaboração. Nos contratos, identificamos as responsabilidades e as pós-condições que os diagramas de colaboração devem satisfazer. E nos casos de uso, coletamos informações sobre quais tarefas os diagramas de colaborações vão ilustrar, mesmo reconhecendo que essas tarefas também já são identificadas nos contratos (um reforço necessário).

O diagrama de colaboração ilustra as interações com mensagens entre instâncias e classes, com a finalidade de satisfazer as pós-condições dos contratos. O diagrama de seqüência também ilustra essas interações. Ambos, os diagramas de colaboração e de seqüência, executam basicamente as mesmas funções. Mas o destaque se faz ao diagrama de colaboração, devido seu formato simples e intuitivo de ilustrar essas interações, se apresentando como um recurso mais interessante para a solução (projeto) do sistema do que o diagrama de seqüência [Larman98].

A Figura 3.1 mostra um exemplo de um diagrama de colaboração, onde se configuram mensagens (eventos) envolvendo dois elementos de software (componentes).

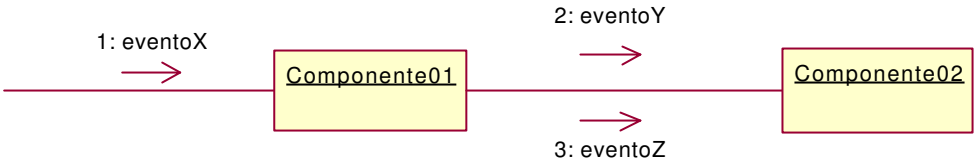

Figura 3.1 – Exemplo do diagrama de colaboração.

3.2 Projeto arquitetural

O projeto arquitetural ilustra a arquitetura do sistema, ou seja, os níveis de comunicação que o sistema terá (camadas) e os diversos módulos existentes nesses níveis (partições) [Sauvé00]. As camadas e partições podem ser vistas como subsistemas, conforme mostrado na Figura 3.2.

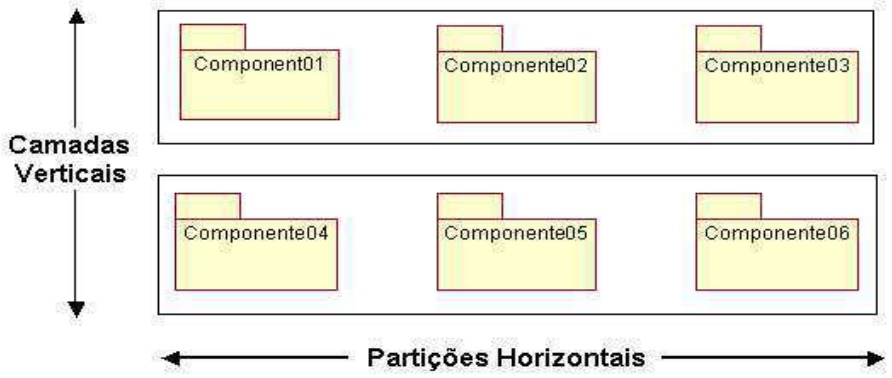

Figura 3.2 – Visão de uma arquitetura com partições e camadas.

O desejado é que dentro do subsistema haja um forte acoplamento entre os elementos que o compõem, e que entre os subsistemas haja um fraco acoplamento, permitindo, dessa forma, uma coesão bem definida [Sauvé00].

Na fase de projeto são tomadas decisões essenciais para a construção do sistema. São decisões de alto nível e estratégicas que têm fortes conseqüências na solução do sistema. O projeto arquitetural ajuda na tomada dessas decisões. Vejamos as principais delas [Sauvé00]:

- **Modularização** – divisão do projeto em subsistemas.
- **Estrutura de comunicação e controle** – escolha do mecanismo de comunicação entre os subsistemas.
- **Interfaces entre subsistemas** – definição das interfaces para interação entre os subsistemas.
- **Persistência** – definição do mecanismo de armazenamento físico dos dados do sistema.
- **Paradigma DBMS** – definição do paradigma de gerenciamento de banco de dados adotado.
- **Reuso** – decisões sistemáticas e oportunas para a reutilização de código.

Estudos consideráveis sobre arquiteturas de sistemas de software têm sugerido a definição de um número coerente de camadas para a construção dos sistemas. Um número grande de camadas tornaria o sistema bastante complexo. Em contrapartida, uma arquitetura com poucas camadas, ofereceria uma simplicidade de projeto, mas implicaria num sistema inflexível e de difícil manutenção.

O modelo inicialmente usado adotava duas camadas, que de acordo com a constatação de suas desvantagens, foi cedendo espaço para o modelo com três camadas, que tem sido muito usado atualmente. Hoje se fala num modelo com *n* camadas, onde a definição dos níveis da arquitetura dependerá da complexidade do sistema. Adotamos para a especificação dos componentes 802.11 o modelo com três camadas, exemplificado na Figura 3.3.

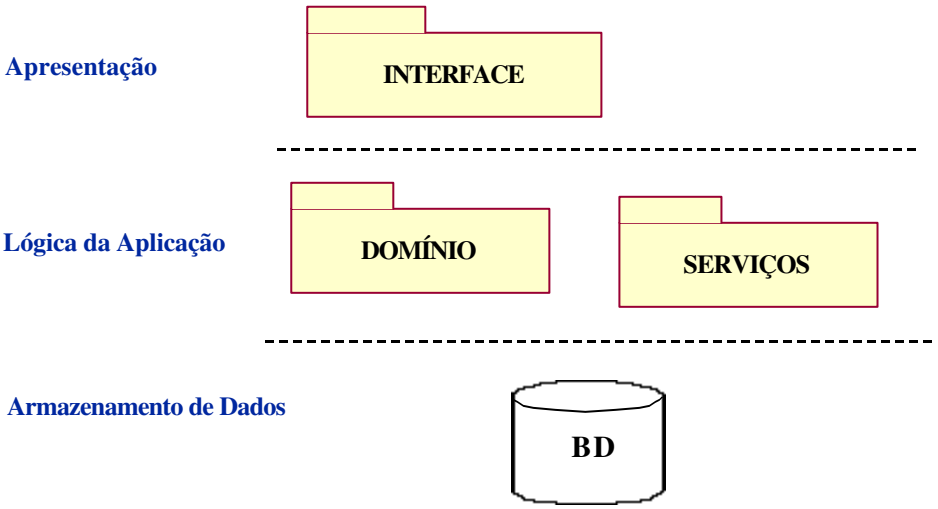

Figura 3.3 – Modelo de arquitetura com 3 camadas.

Esse modelo apresenta três níveis de comunicação (camadas). A camada de "apresentação" é representada por janelas, relatórios e outros elementos que são usados na interface ou serviço com usuário. A "lógica da aplicação" é representada pelas tarefas e regras que governam o sistema. A camada de "armazenamento de dados" trata de questões de persistência.

O projeto arquitetural trata de questões macros, relativas ao sistema, sendo considerado um projeto de alto nível. Mas a fase de projeto exige a elaboração de um outro projeto: o projeto arquitetural de "baixo nível" (também chamado de "projeto detalhado"), que lida com objetos individuais e suas interações, procurando resolver problemas de atribuição de responsabilidade às classes [Sauvé00].

O projeto detalhado sugere a utilização de uma tecnologia relativamente recente para definir a responsabilidade entre os elementos de software: "padrões de projeto" [GoF95]. Essa tecnologia oferece formas interessantes de solucionar problemas que tratam da responsabilidade de cada elemento na solução do sistema.

Padrões de projeto revelam quem faz o que, como e com quem. Por exemplo, o que um componente deve fazer, com qual componente ele deverá se comunicar, e qual evento ele deve enviar para esse outro componente, tudo isso para contemplar uma certa funcionalidade do sistema [GoF95].

Padrões de projeto representam um conjunto de experiências feitas ao longo dos anos, de soluções de sistemas que deram certo no passado. Essa tecnologia oferece padrões para instanciar, estruturar e definir comportamento de elementos de software [GoF95]. Trata-se de uma tecnologia recente e promissora, que tem sido considerada um diferencial na construção de sistemas, oferecendo flexibilidade e reusabilidade ao software.

Podemos encontrar um conjunto com 23 padrões de projeto que foram definidos no livro Design Patterns [GoF95]. Na especificação dos componentes 802.11, apenas o padrão de projeto "observer" é usado. Detalhes deste padrão pode ser visto em [GoF95].

4. Fase de codificação

A fase de codificação representa a implementação do sistema, envolvendo o uso de uma linguagem de programação orientada a objetos (Java, C++, etc), testes, prototipação, entre outras tarefas. Nessa fase acontece a tradução dos documentos feitos nas fases de análise e projeto para código. Essa tradução (passagem) ocorre da seguinte maneira:

- A fase de análise se propõe a conhecer o que deve ser feito, ou seja, qual sistema deve ser construído.
- Na fase de projeto é definido como deve ser feito, expressando a solução para construir o sistema.
- Na fase de codificação acontece a construção propriamente dita do sistema, em termos de código, através de uma linguagem de programação.

A fase de codificação é relativamente simples e pode ser considerada mecânica, uma vez que os esforços e investimentos feitos para desvendar e contemplar os requisitos exigidos pelo sistema, já foram indicados nas fases anteriores (análise e projeto). Sendo assim, destacamos novamente a importância de uma especificação que seja capaz de identificar conscientemente o problema, e apresentar a solução de forma inteligente, confiável e eficiente. Essa é a proposta desta Dissertação, que investe na especificação de componentes para a modelagem das redes 802.11.

Devido a esse escopo, não entramos em detalhes nos recursos que fazem parte da fase de codificação, apenas apresentamos mais adiante as vantagens da programação orientada a objetos, com intuito de comprovar que a especificação proposta nesta Dissertação pode ser naturalmente validada pelo uso do paradigma de orientação a objetos.

Vejamos a seguir informações sobre a linguagem de modelagem unificada (UML), adotada na especificação dos componentes 802.11.

5. Desenvolvimento com UML

Para representar os diagramas e modelos que fazem parte particularmente das fases de análise e projeto, o processo de [Larman98] utiliza a UML (*Unified Modeling Language – Linguagem de Modelagem Unificada*).

A proposta da UML é unificar as notações usadas para representar os elementos que fazem parte da especificação de diversos tipos de sistemas. Até recentemente existiam, e ainda existem, diversas linguagens de modelagem, cada uma com características próprias. Portanto, a UML tem a intenção de padronizar essas linguagens de modelagem, facilitando o processo de desenvolvimento [UML97].

A principal função da UML é auxiliar o desenvolvedor na especificação de sistemas. Para isso, ela dispõe de um conjunto de recursos que abrange as fases de análise e projeto de um processo de desenvolvimento. Na especificação dos componentes 802.11, os seguintes documentos UML são usados:

Documentos de análise

- Diagrama de caso de uso.
- Modelo conceitual.
- Diagrama de seqüência.

Documentos de projeto

- Diagrama de colaboração.
- Projeto arquitetural de alto nível.
- Projeto detalhado.

A UML especifica, visualiza, constrói e documenta artefatos de sistemas de software, ajudando a construir sistemas grandes e complexos, através de modelos prontos e expressivos, independente da linguagem de programação empregada. Com a UML o desenvolvedor consegue representar e entender o sistema que está sendo construído, tanto numa visão natural (artefatos de análise), quanto numa visão técnica (artefatos de projeto) [UML97].

Atualmente existem muitas ferramentas UML que podem ser usadas na construção de sistemas. Empresas como a Microsoft, Oracle, Rational Software, HP, MCI Systemhouse, Unisys, entre outras, já disponibilizam e comercializam ferramentas UML [UML97]. Para especificação dos componentes propostos nesta Dissertação usamos a ferramenta Rational Rose 2000, da empresa Rational Software, por se apresentar como uma ferramenta comprovadamente eficiente.

6. Programação orientada a objetos

A especificação proposta nesta Dissertação pode ser usada na construção de ferramentas de simulação voltadas para a modelagem e avaliação de desempenho de redes 802.11. A grande maioria dessas ferramentas é construída através de linguagens de programação orientadas a objetos, como a linguagem C++, por exemplo.

A construção de novas ferramentas de simulação ou de componentes que podem ser agregados a ambientes já existentes que atendam a simulação de sistemas específicos, é naturalmente possível devido às vantagens que a linguagem de programação orientada a objetos oferece.

Conscientemente, entendemos o paradigma de orientação a objetos como uma evolução comprovada de paradigmas anteriores, pois além da aplicação de uma perspectiva de objetos, ele oferece inúmeras vantagens que o torna mais interessante do que outros paradigmas de programação. Vejamos algumas dessas vantagens [Jones00] [Kamienski96]:

- **Perspectiva de objetos:** toda a especificação leva à decomposição do problema a unidades conceituais intuitivas conhecidas como objetos. Esses objetos representam a tradução de coisas do mundo real que irá fazer parte do escopo que o sistema irá atender. O termo orientação a objetos indica que todas as funcionalidades do sistema são atendidas através da comunicação e colaboração desses objetos.

- **Abstração de dados:** consiste de um conjunto de atributos e operações que caracterizam o comportamento de objetos. Tipos abstratos de dados possuem características como *encapsulamento* e *modularidade*.
- **Reuso de código:** significa que elementos podem herdar ou ser criados a partir de elementos já existentes, sem a necessidade daqueles serem construídos por inteiro. Existem basicamente duas formas de reuso de código: *herança* e *composição*.
- **Herança:** esse mecanismo oferece a possibilidade de criar elementos (classes de software) a partir de outros elementos, através de uma hierarquia sistemática que oferece clareza e reuso de código na construção de sistemas.
- **Composição:** o mesmo que herança, diferindo pelo fato de que na composição um objeto é composto a partir de várias classes, com uma forte independência nessa composição.
- **Encapsulamento:** representa o fato de um objeto ser inviolável, caracterizando integridade e flexibilidade nos sistemas. Composição representa melhor o encapsulamento do que herança, já que quando uma classe herda de outra, basicamente cria-se um elo permanente entre elas, o que não ocorre na composição, uma vez que neste caso os objetos são instanciados sem a criação de elos entre o objeto composto e as classes que o criou.
- **Polimorfismo:** Objetos recebem mensagens que ativam seus métodos (operações) para atender as diversas funcionalidades do sistema. Para tornar os sistemas mais claros e dinâmicos, objetos podem responder a uma mesma mensagem, cada um ao seu próprio modo. Com isso, não precisamos escrever (codificar) mensagens personalizadas para objetos que vão executar operações comuns. A essa possibilidade chamamos de polimorfismo.
- **Flexibilidade:** alterações podem ser facilmente feitas no sistema sem causar problemas. Devido ao encapsulamento, quando se muda um objeto, essa mudança apenas reflete nesse objeto, e não nos outros objetos que o utilizam.
- **Escalabilidade:** representa a capacidade de uma aplicação crescer sem aumentar sua complexidade e sem comprometer seu desempenho.

De um modo geral, a programação orientada a objetos combina recursos modernos de codificação com as facilidades de abstração e especificação dos elementos do mundo real que devem ser representados nos sistemas de software: os objetos.