3
O modelo de programação proposto

O modelo de programação proposto utiliza como base um conjunto de funcionalidades básicas que podem ser combinadas por meio de um controle de fluxo baseado em máquinas de estados finitos (FSM). Conforme dito na seção 1.1, esse conjunto de funcionalidades representa padrões de interação típicos de aplicações para RSSF. Podemos dizer que os padrões de interação estão no nível da execução de uma linguagem de macroprogramação em RSSF citado na seção 1.3, enquanto que o conjunto de funcionalidades básicas equivalem ao nível de uma linguagem que é executada em cada mote.

Esse conjunto de funcionalidades básicas pode ser visto como uma biblioteca de ações e eventos. Com isso o controle de fluxo da máquina de estado pode comandar ações e reagir a eventos dessa biblioteca. Para completar o nosso modelo de programação, possibilitamos que o comportamento de algumas funcionalidades básicas possa ser configurado por meio de parâmetros.

Nas próximas seções vamos, primeiramente, apresentar o nosso conjunto de padrões de interação e a respectiva biblioteca de funções. Em seguida vamos descrever o funcionamento do controle de fluxo baseado em FSM para depois apresentar um pequeno exemplo de utilização do nosso modelo.

3.1
Padrões de interação para aplicações em RSSF

Entendemos que o requisito de facilidade de programação depende diretamente do conjunto de padrões de interação proposto. Esses padrões de interação devem representar funcionalidades genéricas e ao mesmo tempo devem ser suficientemente expressivos para poder atender diferentes tipos de aplicações em RSSF, i.e., deve ser possível construir diversas aplicações a partir da combinação desses padrões.

Optamos por consolidar a lista de padrões de interação a serem oferecidos a partir da análise de duas diferentes fontes de informação. A primeira fonte foi um conjunto de aplicações encontradas na literatura e a segunda fonte foi a demanda gerada pelos modelos de macroprogramação. Os modelos de macroprogramação considerados foram analisados em [15].

A seguir apresentamos cada fonte analisada e o resultado final com a lista consolidada. No apêndice A apresentamos, com mais detalhes, o procedimento utilizado e os padrões identificados para cada fonte.

3.1.1 Operações típicas das aplicações em RSSF

Para identificar as operações típicas das aplicações escolhemos três aplicações para RSSF com diferentes comportamentos funcionais. Essas aplicações foram inspiradas em alguns exemplos retirados da literatura sobre RSSF, principalmente de [2], [16] e [17]. A partir das operações necessárias para cada aplicação foi possível identificar alguns padrões de interação. A seguir apresentamos cada aplicação.

Aplicação 1 - Alarme de incêndio florestal

O objetivo da aplicação é detectar rapidamente focos de incêndio em florestas. Para isso os sensores são distribuídos em regiões da floresta de forma que se possa identificar a região do alarme.

Os nós ficam executando periodicamente uma leitura do sensor de temperatura. Caso um nó detecte uma leitura de temperatura maior que um valor predefinido, este deve verificar se as temperaturas dos nós vizinhos estão próximas da temperatura de alarme. A condição de alarme será confirmada se pelo menos dois nós vizinhos estiverem com o valor de temperatura maior que um valor percentual do valor de alarme. Nesse caso será enviada uma mensagem para a estação servidora. Essa mensagem deve conter a temperatura medida e o identificador do nó.

O nó ativado deve notificar os nós vizinhos que um alarme já foi enviado. Um nó só pode gerar um novo alarme para sua região caso já tenham se passado pelo menos 60 minutos da última notificação de alarme.

Aplicação 2 - Monitor de temperatura e Alarme de incêndio predial

Essa aplicação usa um conjunto de sensores com capacidade de leitura de temperatura com duplo objetivo: monitorar a temperatura média dos ambientes do prédio (controle do conforto ambiental) e gerar alarmes em caso de incêndio. Os nós são distribuídos pelo edifício e cada ambiente define um grupo de nós.

Para a monitoração, cada nó faz uma leitura periódica do sensor de temperatura e envia o valor para o coordenador do grupo. O coordenador do grupo calcula a média das temperaturas de todos os nós do grupo, e, em seguida, envia para a central de controle uma mensagem com o valor calculado.

Uma outra leitura periódica é usada para o alarme de incêndio. Se o valor estiver acima do valor de alarme predefinido, o nó deve consultar os nós vizinhos do mesmo grupo e verificar quantos nós estão com valores acima de 90% do valor de alarme. Independentemente dos valores respondidos, o nó ativo deve enviar uma mensagem de alarme com o quorum da consulta para a central de controle.

Sempre que um alarme é gerado, o nó ativado deve notificar os nós vizinhos que esse alarme já foi enviado. Um nó só pode gerar um novo alarme para seu ambiente caso já tenha passado pelo menos um minuto da última notificação de alarme.

Aplicação 3 - Estacionamento urbano e Monitoração de vagas

Essa aplicação auxilia motoristas na reserva de uma vaga de estacionamento. Também informa para uma central de controle a situação das vagas em cada região. Para isso usa um conjunto de sensores distribuídos pelas vagas de estacionamento da rua. Esses sensores indicam se a vaga está ocupada ou não. Os nós sensores são agrupados por região (CEP, bairro ou Rua).

Para a busca de vagas, o veículo, também equipado com um nó sem fio, envia uma mensagem para os nós vizinhos solicitando a reserva de uma vaga disponível. Todos os nós respondem com a situação da respectiva vaga e os nós com vaga não ocupada fazem uma pré-reserva com a identificação do nó solicitante. O nó solicitante seleciona um dos nós disponíveis e responde para os vizinhos a sua escolha, de forma a confirmar a vaga selecionada e liberar o restante das vagas. Então o nó selecionado responde a confirmação da seleção para o nó solicitante. Se ocorrer a situação de mais de uma solicitação durante o processo de negociação, os nós das vagas disponíveis devem responder de acordo com a ordem do recebimento.

Na monitoração, cada nó envia periodicamente para o nó coordenador do grupo a informação da sua vaga. O nó coordenador sumariza a situação da região e envia uma mensagem com os contadores de vagas para a central de controle.

3.1.2 Elementos utilizados nos modelos de macroprogramação

Os modelos de macroprogramação são uma fonte natural para identificação dos padrões de interação de interesse. No nosso caso analisamos os modelos de macroprogramação apresentados em [15]: Regiment[2, 18]; Pleiades[17]; Cosmos[3]; TinyDB[1]; WADL[16]; ATaG[19].

Os principais padrões encontrados foram relativos à comunicação. Identificamos padrões para diferentes formações de grupos de nós e a possibilidade de aplicar funções agregadoras nesses grupos. Também identificamos padrões para comunicação com a estação servidora.

3.1.3 Padrões de interação identificados

Na Tabela 3.1 apresentamos o resultado final da consolidação para os padrões de interação. No apêndice A apresentamos com mais detalhes a análise utilizada para a identificação e consolidação dos padrões.

Tabela 3.1: Padrões de interação identificados

Elemento	Descrição
Grupo: Comunicação	
Envia EB	Envia uma mensagem do nó em execução para a Estação Servidora.
Envia Nó	Envia uma mensagem do nó em execução para um nó específico dentro do grupo.
Envia Pai	Envia uma mensagem do nó em execução para o nó Pai na hierarquia topológica.
Envia Filhos	Envia uma mensagem do nó em execução para os nós Filhos na hierarquia topológica.
Grupo: Funções de agregação	
Média Grupo	Calcula o valor médio de valores distribuídos pelos nós do grupo.
Somatório Grupo	Calcula a soma de valores distribuídos pelos nós do grupo.
Máximo Grupo	Calcula o valor máximo dentre os valores distribuídos pelos nós do grupo.
Mínimo Grupo	Calcula o valor mínimo dentre os valores distribuídos pelos nós do grupo.
Sumário Grupo	Sumariza os estados dos nós distribuídos num grupo.
Reserva Recurso Grupo	Reserva de um determinado recurso dos nós do grupo. Baseado na negociação entre o nó ativo e o nós do grupo.
Agrega Grupo	Gera um vetor de dados com os respectivos valores dos nós de um grupo.
Grupo: Funções Locais	
Evento Timer	Configura um temporizador para execução de um procedimento.
Evento Condição de leitura sensor	Configura uma condição de leitura de sensor para execução de um procedimento.
Evento Condição de cálculo de agregação	Configura uma condição de um cálculo de agregação para execução de um procedimento.

Continua na próxima página...

Tabela 3.1: Padrões de interação – continuação

Elemento	**Descrição**
Evento Condição de mensagem recebida	Configura uma condição de um código de mensagem recebida para execução de um procedimento.
Grupo: Agrupamentos	
Vizinhos *1-hop*	Define um grupo de nós vizinhos do nó em execução ao alcance do sinal de rádio. Implementado com mensagem *broadcast*.
Vizinhos *n-hops*	Define um grupo de nós vizinhos do nó em execução ao alcance de até n saltos. Implementado com mensagem *broadcast* e subsequentes reencaminhamentos de mensagens pelos nós vizinhos.
Parâmetros Estáticos	Define um grupo de nós da rede que atendam uma determinada condição que não varia no tempo.
Parâmetros Dinâmicos	Define um grupo de nós da rede que atendam uma determinada condição que pode variar no tempo.
Filhos	Define um grupo de nós imediatamente abaixo na hierarquia topológica da rede.
Filhos n-camadas	Define um grupo de nós imediatamente abaixo n camadas na hierarquia topológica da rede.
Rede	Define o grupo de todos os nós da rede.

3.2 Conjunto das funcionalidades básicas

Nessa seção vamos identificar e detalhar o conjunto das funções oferecidas para disponibilizar os padrões de interação identificados.

Na próxima subseção (3.2.1), apresentamos a arquitetura proposta com a abordagem que estamos considerando para a implementação da nossa biblioteca. Na subseção 3.2.2, discutimos as operações que dependem da comunicação na rede. Em seguida, na subseção 3.2.3, propomos um conjunto de parâmetros que permitem a simplificação da quantidade de funções e ao mesmo tempo facilitam a implementação do procedimento de reconfiguração remota. Finalmente apresentamos a biblioteca de funções na seção 3.2.4.

3.2.1 Arquitetura da biblioteca de funções

Para uma melhor identificação das funções, utilizamos uma arquitetura em camadas. Essa arquitetura divide as funcionalidades básicas em diferentes funções, mas que colaboram entre si para atender aos requisitos dos padrões de interação.

Com essa arquitetura temos a maioria das operações disparadas pela aplicação local, mas também temos algumas operações disparadas por aplicações remotas via mensagens de comunicação. Isso é necessário para garantir o funcionamento de operações coordenadas a partir de um nó específico e que não dependam da aplicação local dos outros nós. A visão de camadas sugere que esse processamento de mensagens seja feito de forma hierárquica. Isto é, as camadas mais abaixo processam as suas mensagens e repassam as outras mensagens para as camadas acima.

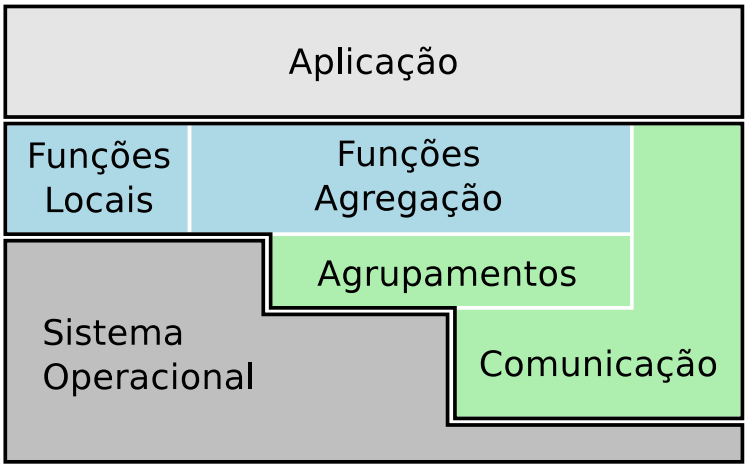

Figura 3.1: Camadas da arquitetura de execução

A seguir descrevemos cada camada da arquitetura proposta na figura 3.1:

Sistema Operacional - Essa camada provê as abstrações para acessar os dispositivos sensores e o rádio. Também gerencia a execução das diversas tarefas do próprio sistema operacional e das outras camadas. Nesse trabalho utilizamos o sistema operacional TinyOS[11].

Funções Locais - Essa camada disponibiliza abstrações para acesso a recursos locais, principalmente recursos de hardware do dispositivo.

Funções de agregação - Essa camada contém a execução dos cálculos em cima da agregação dos valores de um grupo de nós. Interage com a camada de Agrupamento para obter os valores distribuídos em outros nós e com a camada de funções locais para obter os valores locais.

Agrupamento - Essa camada é responsável pela definição e organização dos grupos de nós conforme a definição da aplicação. Interage com a camada de comunicação para a troca de mensagens entre os nós que compõem os grupos.

Comunicação - Essa camada é responsável pela comunicação entre os nós e com a estação base. Essa camada provê, para a camada de Agrupamento, algumas facilidades de comunicação para a formação de grupos de nós. Também fornece algumas funções para Camada da Aplicação.

Aplicação - Essa camada executa o programa responsável pelo comportamento da aplicação. É onde executamos o controle de fluxo que utiliza as funcionalidades básicas disponibilizadas pelas demais camadas.

3.2.2 Operações dependentes da comunicação

A maioria das funcionalidades levantadas dependem da comunicação na rede de sensores sem fio. Por isso, primeiro identificamos os protocolos básicos necessários para comunicação para, em seguida, definir o comportamento das funções biblioteca.

Analisando as características de comunicação para os padrões de interação levantados foi possível identificar a necessidade de três formas de comunicação descritas a seguir.

Roteamento - Formação de uma topologia hierárquica de comunicação baseada na intensidade do sinal de rádio entre os nós. Considerando a estação base como nó "raiz". Permite o roteamento de mensagens de qualquer nó da rede para a estação base e da estação base para qualquer nó da rede.

Disseminação - Propagação do conteúdo de uma estrutura de dados pelos nós. Deve-se manter um controle de versão e garantir que todos os nós recebam a última versão dos dados.

Formação de Grupos - Protocolo básico para formação de grupos. Uma mensagem, a partir de um nó origem, deve ser propagada em saltos pelos nós vizinhos. Cada nó poderá responder a essa mensagem, que deverá ser roteada no caminho inverso. O limite de saltos e os nós que devem responder são definidos nos parâmetros de formação do grupo.

Parâmetros na formação de grupos

Para garantir o reaproveitamento do protocolo de formação de grupos é necessário que a mensagem tenha parâmetros que permitam utilizar diferentes regras de formação de grupos. Esses mesmos parâmetros são utilizados diretamente pela aplicação como facilitadores na configuração da aplicação.

A ideia básica é que cada grupo de nós seja definido pela combinação de um conjunto de parâmetros e o alcance máximo em saltos na comunicação via rádio (hops). O nó que disparar a comunicação deve colocar na mensagem os seus parâmetros. Cada nó que receber uma mensagem deverá comparar os parâmetros da mensagem com os parâmetros locais para verificar a necessidade de responder a mensagem. A mensagem deverá ser propagada até alcançar o limite de hops definido.

Os parâmetros devem ser definidos de forma a atender todos os tipos de formação de grupos. Nos grupos hierárquicos, deve-se utilizar adicionalmente o parâmetro de nó pai definido na topologia do protocolo de roteamento.

Cálculo de agregação

O cálculo de agregação sempre será centralizado em um nó solicitante, podendo ser um nó coordenador de grupo (líder) ou um nó disparado por um evento interno como um alarme. O nó solicitante deverá utilizar o suporte de formação de grupos disponibilizado pela funcionalidade de agrupamento, o retorno da formação do grupo deve conter a informação requerida. A cada valor retornado, o nó solicitante deverá contabilizar parcialmente a função agregadora solicitada. Quando finalizar o tempo limite para efetuar a agregação, o nó solicitante deverá aplicar a finalização da função de agregação, e em seguida disparar um evento interno para indicar o final de agregação. A partir do evento com o resultado da agregação a aplicação pode, opcionalmente, efetuar uma operação de comparação do resultado com um valor constante pré-configurado.

Para o caso da função de Sumário a operação é executada em dois estágios. O primeiro estágio executa a agregação totalizando a quantidade de valores acima e abaixo de um limite parametrizado. O segundo estágio retorna o resultado de uma nova comparação em cima de um dos totais do primeiro estágio, por exemplo compara se o total acima é maior que um outro limite parametrizado.

3.2.3 Definição dos parâmetros para configuração

Com o objetivo de simplificar o desenvolvimento de novas aplicações e deixar as funções da nossa biblioteca mais genéricas, identificamos um conjunto de parâmetros que permitem configurar as diversas funcionalidades requeridas pelos padrões de interação. O modelo baseado em parâmetros também simplifica o processo de desenvolvimento e reconfiguração de aplicações.

Esses parâmetros influenciam a operação de algumas funções para que as mesmas atendam ao objetivo da aplicação final. A maioria dos parâmetros são definidos pelo desenvolvedor, com exceção dos identificadores automáticos para o mote coordenador do grupo e dos motes roteadores na hierarquia topológica.

Na Tabela 3.2 apresentamos os parâmetros utilizados atualmente no nosso sistema.

Tabela 3.2: Estrutura dos parâmetros de configuração

Parâmetro	**Valores**
Controle de tempo	
Timer Periódico	x milissegundos
Timers Customizado (x4)	y segundos
Coleta local	
Origem do valor	Id do Sensor
Valor de referência para comparação	Inteiro
Tipo de operação de comparação	$<, >, =$, etc..
Coleta Agregação - Estágio principal (Todas funções)	
Tipo de função	Avg, Sum, Max, etc..
Origem do valor	Id do Sensor
Valor de referência para comparação	Inteiro
Tipo de operação de comparação	$<, >, =$, etc..
Coleta Agregação - Estágio Secundário (Sumarização)	
Resultado do primeiro estágio	Valor para True ou False
Valor de referência para comparação	Inteiro
Tipo de operação de comparação	$<, >, =$, etc..
Formação de Grupos	
Valor A	Parâmetro Grupo
Valor B	Parâmetro Grupo
Valor C	Parâmetro Grupo
Max hops	Inteiro
Valores Considerados	A e/ou B e/ou C
Topologia do grupo	Hierárquica ou Livre
Nó Pai	Id Nó
Precisa de Coordenador	Verdadeiro ou Falso
Nó Coordenador	Id Nó
Período da Reeleição	z minutos

3.2.4 Biblioteca de Funções Parametrizáveis

Na Figura 3.2 apresentamos um diagrama da arquitetura com as principais funcionalidades oferecidas. Essa organização tem foco no comportamento das funcionalidades básicas e nas respectivas funções da nossa biblioteca.

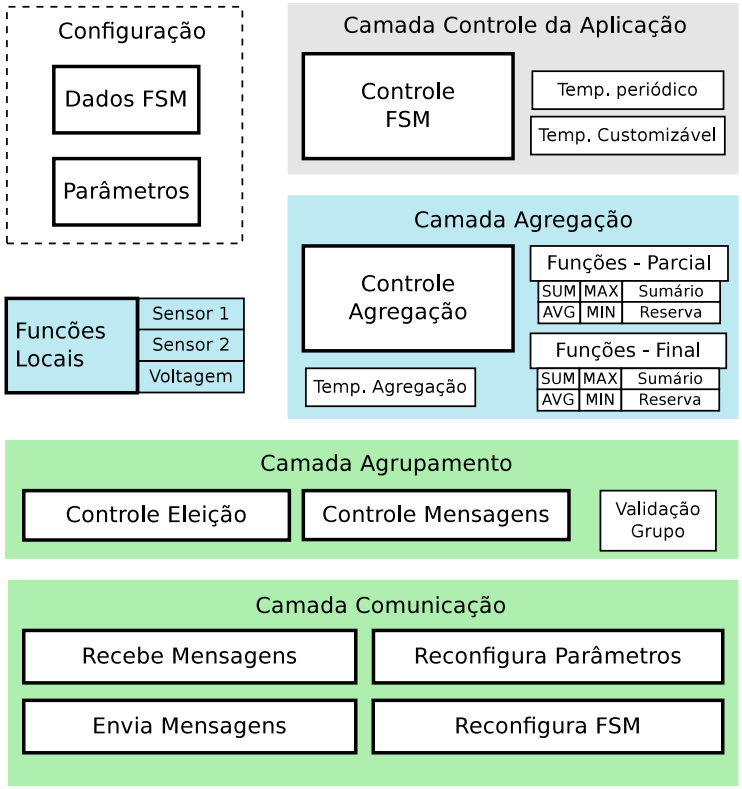

Figura 3.2: Arquitetura com as funcionalidades básicas

Na ótica do usuário do modelo, são disponibilizados diretamente somente algumas funções e eventos que por sua vez utilizam as demais funções. Por exemplo, a ação disponível para o usuário disparar uma agregação executa uma série de ações internas para se comunicar com os nós do mesmo grupo e calcular o valor da agregação solicitada.

Na Tabela 4.1 da seção 4.2 no capítulo Implementação (página 40), apresentamos um resumo das principais ações e eventos disponibilizados na versão atual da nossa implementação.

3.3 Controle de fluxo baseado em FSM

Os parâmetros permitem variações independentes no comportamento das diferentes funções da nossa biblioteca. Então é necessário, para construção de novas aplicações, um mecanismo de controle de fluxo que permita a combinação das funções da biblioteca conforme os requisitos da aplicação. Implementamos esse mecanismo de controle de fluxo baseado no modelo de máquina de estados finitos (FSM). O modelo de máquina de estados é facilmente adaptável ao modelo típico de eventos utilizado em RSSF, principalmente em relação ao comportamento *split-phase* e o modelo de programação orientado a eventos.

Nossa definição considera que a FSM será controlada por uma tabela de transições. Cada transição é descrita por um evento e um estado de entrada e uma ação e um estado de saída. As ações e eventos são disponibilizados pela biblioteca de funções. Na Figura 3.3 apresentamos um diagrama simplificado da arquitetura de controle da FSM. O módulo de Controle da FSM mantém os estados corrente para cada máquina de estado configurada, acessa a Tabela de Transições e dispara ações e recebe eventos do módulo de Funções/Eventos. O retorno da execução de uma ação não interfere na transição que efetuou o disparo da mesma, isto é, a transição sempre ocorre e, se for o caso, a ação gera um evento de erro para ser tratado normalmente pela FSM.

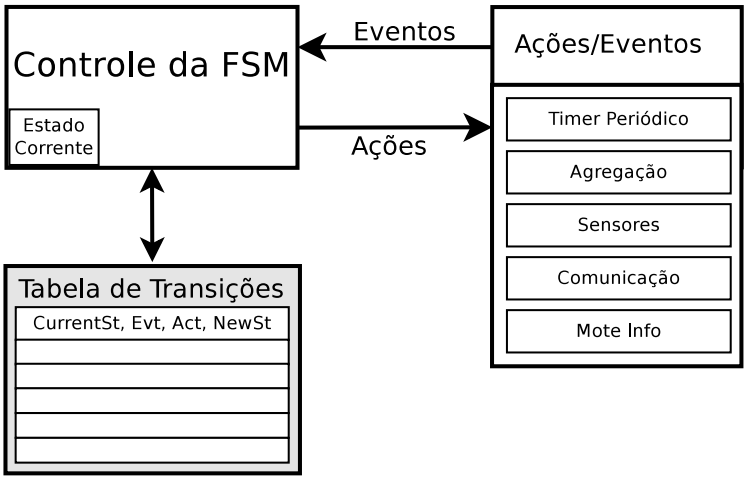

Figura 3.3: Arquitetura de controle da FSM

Na Figura 3.4 apresentamos a sintaxe utilizada para o diagrama FSM e também temos um exemplo de configuração simplificada da tabela de transições. O exemplo representa uma máquina com dois estados `A` e `B`. Ao receber o evento *Evt_1* com parâmetro *[success]* no estado `A`, o controle muda para o estado `B` e executa a ação *Act_X*. Ao receber o evento *Evt_2* com parâmetro *[success]* no estado `B`, o controle muda para o estado `A` e executa a ação *Act_Y*.

Na Tabela 3.3 apresentamos a definição dos campos da nossa tabela de transições. É possível a criação de até oito FSMs distintas em uma mesma aplicação. Na nossa implementação, o estado de entrada pode ser definido por um estado da máquina corrente e um segundo estado de uma máquina distinta, dessa forma possibilitando a interdependência entre máquinas. Como exemplo de aplicação desse tipo de recurso temos uma aplicação com duas máquinas. A primeira máquina X mantém o controle de coleta com os estados "ativo" ou "inativo". Uma segunda máquina Y, responsável efetivamente pela coleta, só reage ao disparo do evento de coleta se o estado máquina X for "ativo".

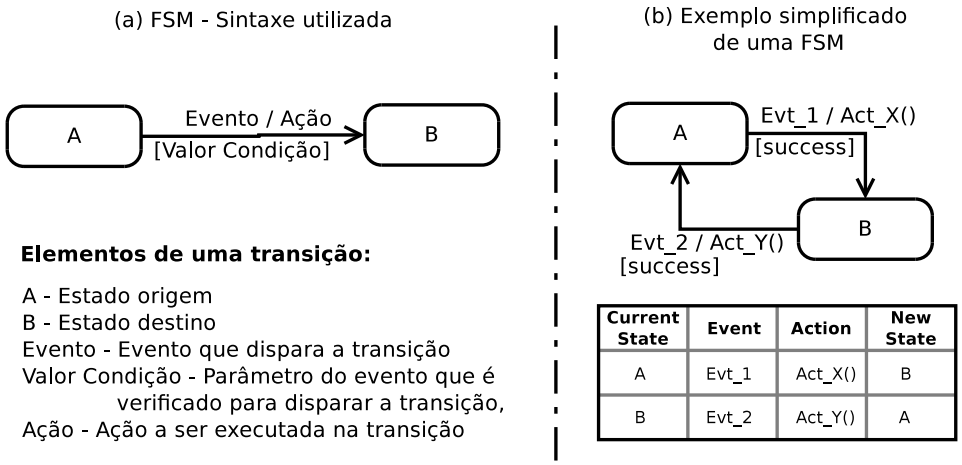

Figura 3.4: Sintaxe FSM e exemplo de tabela de transições

Tabela 3.3: Campos de um registro da tabela de transições

Campo	bits	Descrição
MachineID	3	Id da máquina
CurrState	5	Estado corrente
ParentMachine	3	Id da máquina "pai"
ParentState	5	Estado corrente da máquina "pai"
Event	6	Evento
Action	5	Ação
NewState	5	Novo estado

A definição dos estados é livre e de responsabilidade do desenvolvedor, mas as possíveis ações e eventos dependem dos componentes disponibilizados no sistema. A quantidade de estados depende da aplicação. Por exemplo, uma aplicação para coleta de temperatura de cada nó só precisa de três estados: aguardando disparo da coleta, aguardando leitura do sensor e aguardando envio de dados. Uma aplicação de alarme vai precisar de quatro estados: aguardando disparo da monitoração, aguardando leitura do sensor, aguardando resultado do teste de alarme e aguardando envio de dados.

3.4 Exemplo simplificado de aplicação do modelo de programação

Na Figura 3.5 apresentamos um exemplo simplificado do nosso modelo de programação para uma aplicação de coleta periódica de temperatura. Essa aplicação envia periodicamente para a estação base o valor do sensor de temperatura.

A solução utilizou o temporizador periódico, o sensor de temperatura e o serviço para envio da dados para a estação base. Por meio de parâmetros,

definimos o período de 10 minutos para o temporizador e indicamos a utilização do sensor de temperatura.

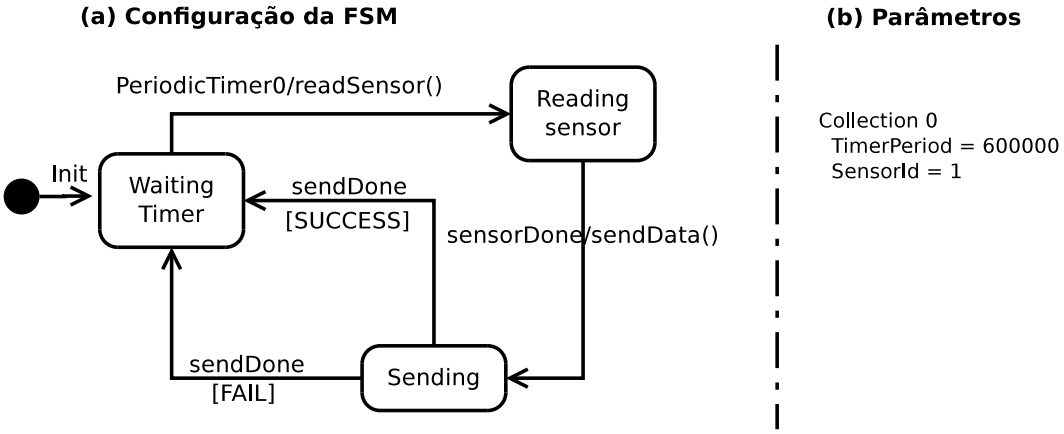

Figura 3.5: Aplicação exemplo - coleta periódica de temperatura

A máquina de estados contém três estados. O estado *Waiting Timer* que fica aguardando um evento do temporizador periódico -*PeriodicTimer0*. O estado *Reading sensor* que aguarda um evento de leitura do sensor - *sensorDone*. O estado *Sending* que aguarda um evento de envio finalizado - *sendDone*.

As ações consideradas são *readSensor* para leitura do sensor e *sendData* para envio da temperatura para a estação base. No nosso exemplo o sensor configurado é o de temperatura.

No capítulo 5 e no apêndice B apresentamos a utilização do nosso modelo para aplicações mais complexas.

3.5 Reconfiguração

O nosso modelo de programação trata a tabela de transições da FSM de forma equivalente a um interpretador restrito, permitindo, dessa forma, certo grau de reconfiguração dinâmica. Adicionalmente os parâmetros de configuração dos componentes também contribuem para a reconfiguração dinâmica.

A reconfiguração de uma aplicação é obtida com o envio de dois tipos de tabelas para cada dispositivo. O primeiro tipo acomoda as transições da FSM que é disseminada igualmente para toda rede. O segundo tipo acomoda os parâmetros de configuração dos componentes que são aplicados individualmente para cada nó.

A versão atual do protocolo de controle de reconfiguração suspende a execução da aplicação em cada nó durante uma reconfiguração. O nosso protocolo de controle utiliza um protocolo de disseminação que privilegia a convergência da disseminação, mas não garante o sincronismo da mesma. Essa versão do protocolo de controle não tem a intenção de prevenir possíveis inconsistências na execução concorrente entre nós com diferentes versões.