2
TinyOS e NesC

O framework de programação mais utilizado em redes de sensores sem fio é composto pelo sistema operacional TinyOS [11] e pela linguagem de programação NesC [12]. A linguagem NesC foi definida em resposta a alguns desafios encontrados na programação em ambientes de redes de sensores sem fio. Para isso seu modelo seguiu alguns princípios, um deles é suportar o desenho do TinyOS. O TinyOS foi todo escrito em NesC e apresenta um modelo de execução específico para dispositivos de baixa potência como os utilizados em redes de sensores sem fio. Apesar de NesC ser uma linguagem de programação e TinyOS um sistema operacional, alguns dos seus objetivos se misturam e outros se complementam. Do ponto de vista do usuário, a programação de uma aplicação é uma extensão do sistema operacional.

2.1
NesC

Seguem alguns dos principais desafios impostos à linguagem NesC:

- Ser orientado a interações com o ambiente - as aplicações devem reagir a mudanças do ambiente, como a leitura de um sensor, e estar preparadas para a concorrência de atividades, como a chegada de um evento durante o processamento de dados;
- Recursos limitados - Os motes têm recursos físicos limitados devido aos objetivos de tamanho pequeno, baixo custo e baixo consumo de energia;
- Confiabilidade - A aplicação deve funcionar por longos períodos sem apresentar defeitos, por exemplo uma aplicação de monitoração ambiental deve funcionar meses sem interação humana;
- Flexibilização do requisito de tempo real - Apesar de algumas atividades serem críticas, como o gerenciamento do rádio ou a amostragem dos sensores, a experiência mostra que é possível atingir essas restrições de tempo com um bom controle da aplicação e limitando o uso de recursos.

A concepção de NesC utilizou os seguintes princípios básicos:

– Ser uma extensão de C - C produz código eficiente para todos os microcontroladores normalmente utilizados em dispositivos para redes de sensores sem fio;

– Ser uma linguagem estática - Não tem alocação dinâmica e o grafo de chamadas é conhecido em tempo de compilação, facilitando a análise citada no item anterior;

– Suportar e refletir o desenho do TinyOS - Suporta os conceitos de TinyOS para componentes e concorrência.

Finalmente, o modelo de programação de NesC expõe ao usuário os seguintes itens:

– Execução orientada a eventos - As interações internas à aplicação utilizam comandos (*actions*) e funções de retorno (*signals*);

– Modelo flexível de concorrência - provê ferramentas para postar tarefas (tasks) e controlar seções atômicas;

– Desenho da aplicação orientado a componentes - utiliza um modelo de componentes onde os módulos (*modules*) implementam as interfaces dos componentes e as configurações (*configurations*) interligam os módulos.

Na prática um programa NesC é composto por um conjunto de componentes (*modules*) que implementam serviços definidos por interfaces (*interfaces*). Esses componentes podem ser interligados (*wired*), via interfaces, pelas definições dos arquivos de configurações (*configurations*). Esse modelo permite que se tenha diferentes implementações para a mesma interface, onde o arquivo de configuração é que vai indicar qual será a implementação utilizada. Tudo isso facilita a definição de aplicações para diferentes plataformas, necessitando somente criar novos componentes de acesso à nova plataforma e manter a definição da interface original.

2.2 TinyOS

TinyOS procura atender algumas características típicas das aplicações em rede de sensores sem fio que diferem das características dos sistemas convencionais. Essas aplicações normalmente operam de forma autônoma por longo tempo, sendo feitas para coletar dados, responder a um ambiente imprevisível e são sem fio.

Para endereçar essas características, TinyOS usa uma arquitetura baseada em componentes, um modelo de execução baseado em eventos e tarefas e permite operações em duas fases (*split-phase*).

TinyOS provê um ambiente de desenvolvimento composto por uma biblioteca de componentes e algumas ferramentas que simplificam o procedimento de construção de novas aplicações. Por se tratar de um sistema operacional para plataformas com recursos limitados, o TinyOS não funciona como um sistema operacional convencional, acima do qual se executam as aplicações do usuário, mas sim como uma biblioteca que deve ser ligada com essas aplicações para construir um único programa executável e que deve ser carregado em cada nó sensor.

Uma das principais características do TinyOS é facilitar a utilização de diversas plataformas de hardware. A utilização do modelo de componentes de NesC possibilita grande modularidade, permitindo que o TinyOS contenha componentes prontos para acessar diferentes tipos de hardwares. A seleção dos componentes adequados para cada hardware é feita durante o processo de compilação e de forma transparente para o usuário.

Uma outra característica importante que o TinyOS usa de NesC é o modelo orientado a eventos. Esse modelo segue o funcionamento natural das interfaces de hardware que normalmente integram esses dispositivos. Também possibilita uma operação de baixo consumo de energia, pois o processamento só ocorre quando solicitado. Por exemplo, uma leitura de sensor deve disparar o conversor A/D que, depois de finalizada, deve retornar o valor lido. O programa que implementada essa operação é quebrado em duas fases. Primeiro uma função é chamada para iniciar a conversão, essa função tem retorno imediato. Quando a conversão é finalizada, é disparada a execução de uma função de *callback* que então deverá processar o resultado da leitura. Esse tipo de operação, as vezes chamada de *split-phase*, é amplamente utilizada nos outros componentes como a interface de rádio ou o temporizador interno.

O modelo de execução do TinyOS gerencia as tarefas (*tasks*) NesC numa fila de execução para serem executadas oportunamente. As tarefas são executadas até o final e só podem ser interrompidas para execução de uma função de tratamento de interrupção de hardware. Isso exclui possíveis condições de corrida entre tarefas, mas não exclui a possibilidade de acontecer com uma função de tratamento de interrupção de hardware. Para os casos em que possam acontecer condições de corrida, sugere-se a quebra do código em tarefas, explicitando os pontos de possíveis interferências. Ainda assim, NesC permite marcar seções do código para execução atômica. Durante a execução de uma seção atômica, o sistema desliga as interrupções.

Resumimos os principais pontos fortes do TinyOS em uma grande biblioteca de componentes para vários tipos de hardwares e em um código robusto e seguro, consequência de cinco anos de amadurecimento do modelo,

e utilizado por milhares de desenvolvedores.

Como consequência do modelo adotado no TinyOS, podemos identificar algumas dificuldades encontradas pelo programador:

- Uma longa curva de aprendizado tanto para o modelo de programação orientado a eventos (*split-phase*) quanto para o modelo de interface/-componentes do NesC.
- A recomendação de manter as tarefas curtas também aumenta a dificuldade para escrever programas que requerem computação intensiva.
- A manutenção da aplicação após a implantação em campo pode ser inviabilizada pela dificuldade de recuperar os motes para uma carga via cabo ou pelo custo de energia para carregar, remotamente via rádio, todo o executável.

2.2.1 Tecnologia

O TinyOS pode ser executado em uma variedade de dispositivos que combinam diversos microcontroladores, chips de rádios e de armazenamento. A referência [13] apresenta uma lista com as características de vários motes comerciais e protótipos, incluindo os suportados pelo TinyOS. Nessa lista o TinyOS é o sistema operacional mais utilizado, sendo utilizado por 60% dos motes.

O processo de compilação no ambiente TinyOS é disparado pelo comando `ncc` que invoca o compilador NesC, que por sua vez invoca o compilador GCC. O comando `ncc` é específico do TinyOS e converte alguns parâmetros para os dois procedimentos de compilação seguintes, selecionando também os componentes da biblioteca do TinyOS adequados ao hardware indicado. O compilador NesC - nescc - gera, a partir do código NesC do usuário e do TinyOS, um código C que contém o sistema operacional e a parte da aplicação do usuário. O compilador C - gcc - gera o código executável. Nesse procedimento é indicado o microprocessador utilizado pelo hardware.

Além das ferramentas para compilação e carga, o TinyOS disponibiliza a ferramenta TOSSIM[14] que permite a simulação de uma rede de nós executando a aplicação do usuário. Para a execução da simulação deve-se criar um script em Python ou um programa em C++ que, por meio de funções da biblioteca do TOSSIM, ativam nós e definem quais nós podem se comunicar via rádio.

Principais funcionalidades

As principais APIs oferecidas pelo TinyOS são **Booting** - Controla a inicialização do componente; **Communication** - Implementa diversos protocolos como Single-hop, Multihop collection, Multihop dissemination e Binary reprogramming; **Time** - Suporte para criação de temporizadores; **Sensing** - Suporte para acesso aos sensores; **Storage** - Suporte para acesso a FlashMemory; **Data Structures** - Suporte para Filas e Vetor de Bits; **Utilities** - Suporte para números randômicos, acesso aos Leds e cálculo de CRC; **Low-Power** - Suporte para operação em baixa potência.