PROGRAMANDO APLICAÇÕES PARA REDES DE SENSORES SEM FIO USANDO UMA LINGUAGEM REATIVA

Aluno: Carlo Melo Caratori
Orientador: Noemi Rodriguez
Co-orientador: Francisco Sant'anna

Introdução

Redes de Sensores sem Fio (RSSF) consistem de sensores autônomos (motes) capazes de monitorar o ambiente no qual eles estão inseridos através de sensores de luz, temperatura, som, entre outros. Estes motes também são capazes de se comunicar, permitindo que este monitoramento abranja grandes áreas. Seu desenvolvimento inicial foi motivado por aplicações militares como vigilância em campos de batalha, mas a constante queda no custo dos motes tem incentivado outras áreas a pesquisar utilidades para os mesmos.

Objetivos

Softwares desenvolvidos para RSSF geralmente envolvem uma grande quantidade de medidas dos sensores, processamento, atuação e comunicação. Essas atividades são em grande parte reativas, tendo em vista que elas envolvem envio e recebimento de mensagens, sensoriamento, etc. O objetivo desta Iniciação Cientifica é avaliar a utilidade de Céu neste mundo de programação em sensores. Céu é uma linguagem de programação reativa que está sendo desenvolvida pelo grupo de RSSF da Puc-Rio. Esta avaliação deve ser executada durante o desenvolvimento da linguagem, dando suporte ao mesmo.

Metodologia e Desenvolvimento

Antes que se possa defender o uso de outra linguagem em RSSF, é preciso explicar o porquê de não estarmos interessados em usar as já existentes. Atualmente, a linguagem de programação mais difundida em aplicações com sensores é nesC [1], que nada mais é do que uma extensão de C desenvolvida com o intuito de englobar os conceitos estruturais e modelo de execução do TinyOS [2], o sistema operacional criado para os motes.

Contudo, apesar de ser a linguagem mais usada neste seguimento, nesC apresenta alguns contratempos. O mais notável diz respeito à dificuldade em se programar com ela. Por ser uma linguagem dirigida a eventos, nesC se baseia em uma cadeia de chamadas e retornos de funções, tornando a programação de problemas não triviais bastante difícil e propícia a erros.

Com isso chegamos a Céu, uma linguagem reativa baseada em um pequeno conjunto de primitivas com funcionalidades similares as de Esterel [3]. Céu é uma linguagem síncrona com sintaxe e semântica completamente diferentes de nesC, mas ao ser compilada gera códigos em C, levando a programas eficientes tanto em termos de memória quanto processamento. Céu também é baseada no uso de eventos internos e externos ao programa, que controlam seu fluxo de execução.

Para que pudéssemos comparar Céu a nesC e avaliar sua performance em aplicações de RSSF, decidimos que os testes seriam feitos baseados nos exemplos demonstrados pelo tutorial [4] do TinyOS e que alguns parâmetros deveriam ser levados em consideração. Estes testes são aplicações reais com diferentes níveis de dificuldade, além de serem muito bem documentados, facilitando a avaliação em si.

Para começar os testes foi escolhido um exemplo muito simples, que apenas pisca os LEDs dos motes um a um. Já neste teste foi percebida uma grande vantagem de Céu, chamada

por nós de expressividade. Com poucos comandos e linhas de código, Céu foi capaz de realizar o mesmo trabalho do código em nesC.

Com isso continuamos com os testes baseados nos exemplos do TinyOS, e novamente os códigos em Céu se mostraram muito mais simples e menores. Nestes testes foram implementados protocolos de comunicação simples, comunicação entre motes e computadores através de portas seriais, sensoriamento, etc. Vale a pena dizer que desde o início houve a preocupação de ser extremamente fiel aos exemplos abordados para que a comparação e avaliação posterior tivessem algum significado pertinente.

Por fim chegamos a um dos exemplos do tutorial que se mostra mais completo. Uma aplicação de detecção de furtos que leva em conta os valores lidos pelos sensores de luz e movimento dos motes para dizer se aquele mote foi roubado ou não. É uma aplicação completamente configurável, onde o usuário informa aos motes como eles detectam o furto e como devem agir ao detectá-lo, e que engloba as principais funções dos motes. O código relativo a esta aplicação é extremamente simples em Céu e foi concluído com certa facilidade devido às ferramentas extremamente uteis oferecidas pela linguagem como, por exemplo, a propagação de valores de variáveis (variáveis reativas) pelo programa.

Ferramenta de Testes

Outra grande funcionalidade de Céu é a possibilidade de testar o programa sem a necessidade de usar o hardware (mote) em si. Através da própria linguagem é possível criar o que chamamos de regiões assíncronas de código que permitem ao programador simular basicamente tudo desde envio e recebimento de mensagens, valores lidos pelos sensores e até condições de erro, dentre outros. Isso torna a programação em si muito menos propícia a erros, tendo em vista que os testes no programa podem ser feitos independentemente do hardware, isolando a ocorrência e facilitando a correção de eventuais erros.

Conclusões

A pesquisa desenvolvida durante esta Iniciação Cientifica foi extremamente válida. Foi uma experiência agregadora, onde tive a oportunidade de trabalhar conceitos vistos em sala de aula, como protocolos de comunicação, programação orientada a objetos, linguagens de programação, dentre outros.

Vale também salientar que o objetivo do programa de Iniciação Cientifica foi cumprido. Durante esses meses de testes usando Céu a linguagem evoluiu consideravelmente e em seu atual estagio oferece ao programador um ambiente simples e seguro de programar. Parte desta evolução se deve a este trabalho de pesquisa e testes desenvolvido por nós durante o último ano.

Referências

[1] Gay, D., Levis, P., Behren, R., Welsh, M., Brewer, E., and Culler, D. **The nesC Language: A Holistic Approach to Networked Embedded Systems**, In *Proceedings of Programming Language Design and Implementation* , June 2003.

[2] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. **System architecture directions for networked sensors**. SIGPLAN Not. 35 (November 2000), 93-104.

[3] Berry, G., and Gonthier, G. **The ESTEREL synchronous programming language: design, semantics, implementation**. Science of Computer Programming 19, 2 (1992), 87-152

[4] TinyOS Tutorials. Disponível em: http://docs.tinyos.net/tinywiki/index.php/TinyOS_Tutorials