Otimizando o Desempenho de Rádios Definidos por Software Através do Desacoplamento de Canais

Roberto de Matos[1,2], Antônio Augusto Fröhlich[2], Leandro Buss Becker[1]

[1]Departamento de Automação e Sistemas – DAS
Universidade Federal de Santa Catarina – UFSC

[2]Laboratório de Integração de Software e Hardware – LISHA
Universidade Federal de Santa Catarina – UFSC

{rmatos,lbecker}@das.ufsc.br, guto@lisha.ufsc.br

Abstract. *The demand for embedded systems with wireless communication has created specific standards to carry out different requirements of range, security and power consumption. This increases diversity and is an obstacle to create a single standard. Software Defined Radios emerged as an alternative to this scenario as it possesses a physical layer which is completely adaptable, however with an overhead much greater than desired. This paper presents an architecture of channels for SDRs based on GNU Radio and USRP2, which demonstrated a significant decrease in the processing cost of the Host, without limiting the flexibility of radio implementations.*

Resumo. *A demanda de sistemas embarcados com comunicação sem fio tem criado padrões específicos para o cumprimento de diferentes requisitos de alcance, segurança e consumo de energia, aumentando a diversidade e impossibilitando a convergência para um único padrão. Os Rádios Definidos por Software surgem como alternativa a esse cenário por possuírem a camada física completamente adaptável, entretanto com o sobrecusto muito maior que o desejado. Este trabalho apresenta uma arquitetura de canais para SDRs baseados no GNU Radio e USRP2, que demonstrou uma diminuição significativa no custo de processamento no Host, sem limitar a flexibilidade das implementações dos rádios, possibilitando assim, uma transição mais suave para o mundo dos sistemas embarcados.*

1. Introdução

Um dos aspectos mais importantes e estudados na área de sistemas embarcados é a forma de comunicação e interação dos vários equipamentos existentes. Atualmente, essa interação tem caminhado cada vez mais para o mundo sem fio, onde é necessário levar em conta o domínio da aplicação, o qual delimita os requisitos com relação a capacidade de dados a serem transmitidos, imunidade a fatores externos, alcance, segurança e consumo de energia. Consequentemente, implicando na criação de diversos protocolos e estruturas de comunicação coexistentes e muitas vezes incompatíveis entre si.

Considerando impossível a convergência de todas as redes sem fio para um único protocolo, a solução tem sido equipamentos capazes de se comunicarem com vários padrões de redes, o que tradicionalmente tem sido alcançado integrando diversos componentes, cada um com compatibilidade para um tipo de rede. Exemplos desses equipamentos

são os novos televisores que se conectam em redes locais sem fio (802.11x) para download de conteúdos especiais da internet e possibilitam a integração com redes pessoais (ex.: Bluetooth) para transmissão de som para fones de ouvidos ou transmissão de dados de câmeras fotográficas e celulares. Ou ainda os gateways residenciais que se comunicam com a rede 3G de telefonia celular e distribuem internet para uma rede local sem fio (ex.: 801.11x).

As desvantagens do modelo tradicional são o aumento do espaço físico e do consumo de energia na adição de cada novo componente, além da inflexibilidade para adição ou adaptação de novos padrões de comunicação sem fio. Assim uma alternativa a esse cenário tem sido o uso de Rádios Definidos por Software (SDR - *Software Defined Radio*) [Mitola 1995], os quais possuem sua camada física completamente reconfigurável, permitindo flexibilidade em vários parâmetros de comunicação como faixa de freqüência, tipo de modulação, protocolo e potência de transmissão [Reed 2002]. Além disso, como os componentes são desenvolvidos em software, eles podem ser reaproveitados em outras plataformas, testados e modificados facilmente [Blossom 2009].

Atualmente, existem várias propostas [Blossom 2009, Ettus 2009, Balister et al. 2007, WARP 2009, KUAR 2009, Ackland et al. 2005] para implementação de SDRs, as quais tipicamente utilizam modelos de alto nível para definição do funcionamento do rádio e o conceito de distribuição de processamento por várias unidades heterogêneas, como FPGAs, DSPs e GPPs (*General Purpose Processors*) nas plataformas de uso específico ou no *host*. As diferenças são percebidas nas regras de criação dos blocos de processamento digital de sinais, conexão desses blocos e a maneira como o sinal é processado pela estrutura que forma o rádio. Um dos projetos mais populares da área é o GNU Radio [Blossom 2009], que juntamente com uma plataforma de baixo custo, chamada de USRP [Ettus 2009] (*Universal Software Radio Peripheral*), possibilita a criação de rádios funcionais a partir de modelos de alto nível utilizando computadores pessoais.

Entretanto, apesar da flexibilidade total da camada física e do provimento de modelos de alto nível para descrição dos rádios, o sobrecusto para implementação de SDRs ainda é muito maior que o desejado para sistemas embarcados. Mesmo computadores pessoais modernos não conseguem processar mais de quatro canais do protocolo IEEE 802.15.4, utilizado em redes de sensores sem fio [Choong 2009]. Assim um dos desafios atualmente é diminuir o custo da execução dos blocos em software utilizando arquiteturas que distribuam de maneira eficiente as tarefas entre as unidades de processamento heterogêneas (FPGA, DSP e GPP) sem a perda de flexibilidade dos blocos de software. Isto possibilitaria, por exemplo, a sua utilização em sistemas embarcados de alto desempenho [Mccarthy et al. 2008] ou de recursos mais limitados [Balister et al. 2007].

Este trabalho apresenta a concepção de uma arquitetura para SDRs que propõem o desacoplamento do conceito de canal da camada física. Isto possibilita a implementação do oneroso estágio de separação de canais (translação de freqüência e decimação/interpolação) de forma mais eficiente em hardware reconfigurável (FPGA), sem perder a flexibilidade de funcionamento do rádio definido por software. O uso da arquitetura proposta mostrou uma melhora significativa no desempenho global do sistema e uma simplificação na interface com a camada física, uma vez que não há necessidade de configurar as variáveis relacionadas com os ajustes do próprio hardware.

O restante do artigo está organizado da seguinte forma: Na próxima seção é apresentado uma visão geral sobre SDR, GNU Radio e a USRP. A seção 3 apresenta a arquitetura de canais proposta. A implementação e os testes comparativos de desempenho são apresentados na Seção 4. Finalmente as conclusões e os trabalhos futuros são apresentados na Seção 5.

2. Software Defined Radio

Rádio Definido por Software (*Software Defined Radio* – SDR) é uma tecnologia emergente que tem sido pesquisada por mais de uma década e vem modificando a forma como os novos rádios de comunicação estão sendo projetados. O termo SDR foi adotado pela primeira fez por Joseph Mitola [Mitola 1995] para designar rádios com a capacidade de funcionamento em várias faixas de freqüência, tipo de modulação, protocolo e potência de transmissão, onde pelo menos 80% das funcionalidades são providas por software, podendo ser reconfiguráveis ou adaptáveis.

O objetivo principal dessa tecnologia é conceber um rádio que virtualmente possa se comunicar com qualquer nova tecnologia apenas atualizando o software, ou seja, o esforço é colocar o software mais próximo da antena, e usá-lo para filtrar, modular, demodular e executar outros estágios da transmissão e recepção. O SDR ideal elimina quase todo hardware, utilizando somente um ADC (*Analog to Digital Converter*) para amostrar o sinal vindo da antena e enviar ao software. No entanto, nem os ADCs mais rápidos são o suficiente para amostrar toda a fatia do espectro de freqüência desejada, sendo necessário mais hardware para converter a faixa interesse para banda base.

O uso de software para implementar rádios garante um ciclo de desenvolvimento muito mais rápido e atualizações em campo com modificações completas das funções, entretanto a extrema flexibilidade e a reconfiguração *on-the-fly* ocasionam um maior consumo de energia se comparados com os ASICs (*Application-Specific Integrated Circuit*), assim, se a flexibilidade da camada física não é importante, este custo seria desnecessário.

Atualmente várias arquiteturas e *frameworks* estão disponíveis para implementar SDRs. Um dos mais utilizados é o GNU Radio, sendo escolhido para desenvolver este trabalho por fazer parte de uma comunidade altamente ativa de software livre, o que garantiu fácil acesso ao *core* para execução das modificações. Além disso, o GNU Radio possui suporte a várias plataformas de hardware, incluindo a USRP (*Universal Software Radio Peripheral*), que é uma placa de baixo custo e serve como interface com o mundo RF. As próximas duas subseções darão uma visão geral sobre o GNU Radio e a USRP, respectivamente.

2.1. GNU Radio

GNU Radio é uma ferramenta de software livre para o desenvolvimento de Software Defined Radios [Blossom 2009]. Foi iniciado com o financiamento de John Gilmore e tem sido desenvolvido desde 2001, com uma reformulação completa em 2004. A facilidade de uso e o alto desempenho para processamento digital de sinais são alcançados pela sua natureza híbrida, que usa C++ para o núcleo dos componentes de processamento de sinais (*Processing Blocks*) e Python para as conexões entre esses componentes na forma de grafos de fluxo (*flowgraphs*) possibilitando a criação de filtros, moduladores, demoduladores e outras estruturas que compõem os rádios.

Sendo executados em plataformas de uso geral, o GNU Radio faz parte de uma vertente iniciada pelo projeto SpectrumWare [Welborn et al. 2009], que desenvolveu uma abordagem que separa temporalmente os fluxos de amostras dos módulos de software, relaxando as restrições temporais sobre os algoritmos de processamento e a sua execução [Tennenhouse and Bose 1996]. O uso em computadores pessoais ordinários permite desenvolvimento facilitado, o que criou uma grande comunidade de desenvolvedores, propiciando um fortíssimo suporte para novas aplicações. Atualmente, o GNU Radio pode ser executado nos sistemas operacionais Linux, Windows e MAC OSX.

O GNU Radio é formado por três principais entidades, são elas: os blocos de processamento (*Processing Blocks*), *Flowgraph* e escalonador. Os blocos de processamento são os componentes que realmente atuam no fluxo de dados e, normalmente, são desenvolvidos em C++ para garantir o desempenho exigido. O *Flowgraph* é responsável por abstrair o fluxo de dados, ou seja, a seqüência dos blocos de processamento de sinais e as conexões entre eles. Finalmente, o Escalonador movimenta os dados (amostras) pelo *flowgraph*. Passando repetidas vezes por cada bloco, ele verifica a relação entre os buffers de entrada e saída para determinar se o bloco deve executar ou não.

2.2. USRP

A USRP (*Universal Software Radio Peripheral*) é uma plataforma aberta, de baixo custo e extremamente flexível desenvolvida sob medida por Matt Ettus [Ettus 2009] para o projeto GNU Radio, o que a fez ganhar rapidamente uma grande comunidade de desenvolvedores em todo o mundo. Sendo basicamente composta por ADCs, DACs, uma FPGA, slots para placas filhas e uma interface de comunicação, a qual conecta o GNU Radio ao mundo RF. O principal objetivo da USRP é permitir aos desenvolvedores a criação de SDRs de baixo orçamento e mínimo esforço inicial. Existem diversas placas filhas, que tem a função de *front-end RF*, as quais são responsáveis por converter a faixa de interesse do espectro de freqüência para uma freqüência intermediária na recepção do sinal e ao contrário para transmissão.

Atualmente, existem duas versões da placa USRP que possuem basicamente as diferenças apresentadas na Tabela 1. O principal problema da USRP1 é a interface de conexão com o PC, o padrão USB2 tem uma taxa máxima teórica de 60MB/s, limitando na prática transferências a 32MB/s. Como cada amostra complexa é composta por quatro bytes, 2 bytes do canal I e 2 bytes do canal Q, a USB2 limita a transferência de oito milhões de amostras complexas por segundo (8MS/s), o que permite uma janela de amostragem de 8MHz. Enquanto o ADC pode amostrar uma janela de aproximadamente 32MHz e o DAC pode gerar sinais dentro de uma janela de 60MHz. Isso caracteriza uma grande limitação na flexibilidade dos SDRs implementados com a USRP.

As melhorias com relação a interface na USRP2 foram alcançadas substituindo a interface USB por uma interface Gigabit Ethernet, com uma capacidade de transferência de 125MB/s, permitindo uma janela de amostragem de 25MHz. Outras melhorias foram o aumento das taxas de amostragens do ADC e do DAC e uma FPGA mais rápida e maior, permitindo a implementação de mais funções no hardware, o que combina a flexibilidade da reconfiguração e a performance requerida por algumas aplicações. Por esse motivo, essa foi a versão escolhida para desenvolvimento do trabalho.

	USRP1	**USRP2**
Interface	USB 2.0	Gigabit Ethernet
FPGA	Altera EP1C12	Xilinx Spartan3 2K
RF Bandwidth to/from host	8MHz @ 16bits	25MHz @ 16bits
Cost	700	1400
ADC Samples	12-bit,64 MS/s	14-bit,100 MS/s
DAC Samples Daughterboard	14-bit, 128 MS/s	16-bit, 400 MS/s
capacity	2 TX, 2 RX	1 TX, 1 RX
SRAM	None	1 Megabyte
Power	6V, 3A	6V, 3A

Tabela 1. Diferenças entre as Versões da placa USRP [Blossom 2009]

3. Arquitetura Proposta

A relação de um para um da interface física (componentes analógicos, ADCs/DACs) para camada física também é comum nas implementações atuais para SDRs, mais por uma questão da herança de implementação dos rádios tradicionais do que por necessidade (Figure 1). Por exemplo, a USRP exporta somente um canal por interface, o qual deve ser configurado para possuir as características esperadas pela implementação da camada física, o que inutiliza o resto do espectro capturado pela interface para outras camada física (Figura 2).

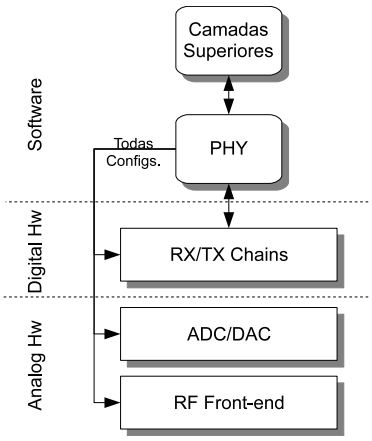

Figura 1. Implementação tradicional das camadas físicas.

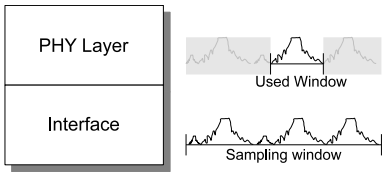

Figura 2. Desperdício do espectro capturado pela interface física.

Outra possibilidade é configurar a USRP para envio de todo o espectro de interesse e fazer a separação de cada canal em software (Figura 4), o que é um processo altamente oneroso para o processamento do *host* e deve ser feito sob-medida para cada conjunto de camadas físicas. Como os parâmetros de configuração do hardware são inferidos a partir das características dos canais, cada mudança exige um esforço grande de reconfiguração que pode variar com o tipo de hardware ou com o novo conjunto de canais alocados.

Com uma análise mais cuidadosa percebe-se que todas as camadas físicas utilizam o conceito de canal, o qual possui apenas dois parâmetros: freqüência central e largura de banda. Padrões para comunicação digital (ex.: 802.11, 802.15.4) constantemente fixam a largura do canal e utilizam identificadores numéricos para referenciá-los, os quais podem ser facilmente traduzidos utilizando a tabela de padronização. Ou seja, o desacoplamento

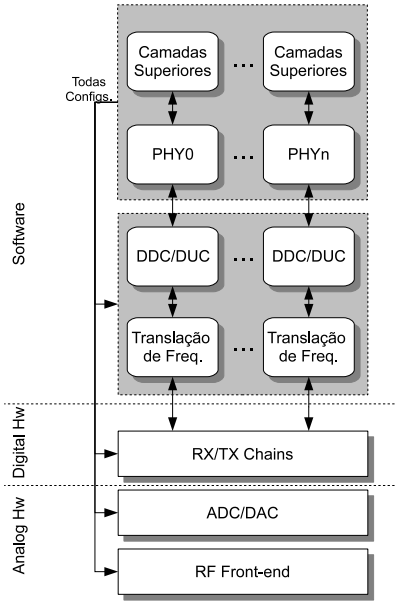

Figura 3. Abordagem em software altamente onerosa para separação de canais.

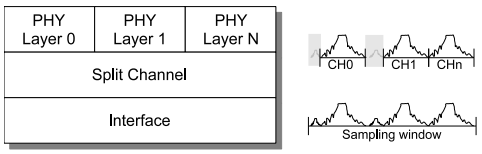

Figura 4. Múltiplas camadas utilizando uma única interface física.

do conceito de canal da implementação da camada física é simples e adiciona flexibilidade no desenvolvimento de aplicações com múltiplas camadas físicas.

O conceito de canal simplifica a interface com a camada física, uma vez que não há necessidade de configurar as variáveis relacionadas com os ajustes do próprio hardware, como é exigido na interface que lida diretamente com o espectro de freqüência. Essas variáveis podem ser inferidas a partir dos canais solicitados pelo usuário, bem como as configurações dos blocos relacionados com a translação de freqüência e decimação/interpolação. Outra vantagem é que a dependência da camada física passa a ser um canal ou um grupo de canais e não mais a interface física completa. Isso permite uma estrutura que extraia vários canais de uma mesma interface e divida-os entre as camadas físicas que estão funcionando em paralelo.

Assim, uma visão geral da arquitetura de canais proposta nesse trabalho é apresentada na Figura 5, onde o principal conceito é uma interface simples e única para alocação de canais independentes, os quais podem ser reconfigurados a qualquer momento no processo de comunicação, respeitando os, aqui classificados, fatores dinâmicos e estáticos.

3.1. Controle

Fatores dinâmicos e estáticos influenciam no projeto e funcionamento da camada física de uma rede sem fio. O fatores dinâmicos estão relacionados com diferentes requisitos das aplicações, ambiente RF e a taxa de consumo de recursos finitos do sistema. Enquanto que os fatores estáticos têm relação com os limites do hardware e regulamentação. No caso da arquitetura proposta, os parâmetros de controle de cada canal são: Freqüência central e largura de banda. A limitação dos recursos do sistema aparece como fator estático e a alocação de canais concorrentes que consomem esses recursos é o fator dinâmico.

O fator estático, recursos do sistema, que limita a configuração da freqüência cen-

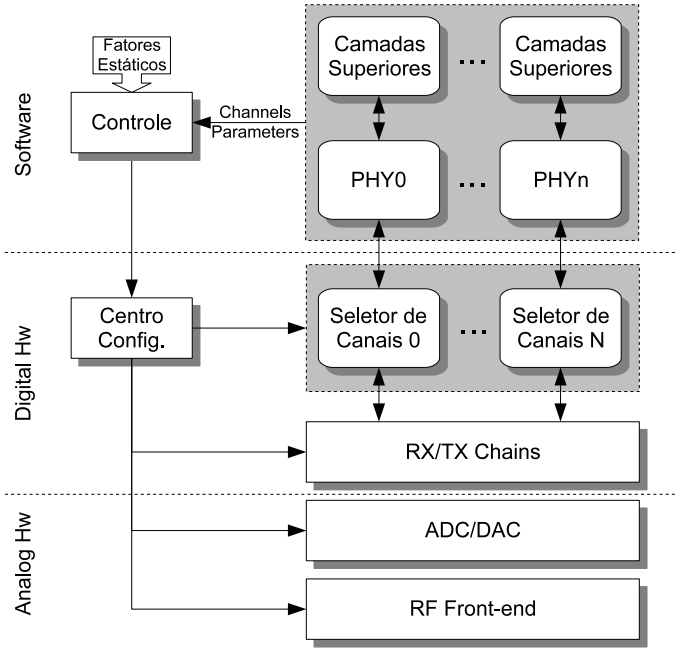

Figura 5. Arquitetura de múltiplos canais.

tral e largura de banda dos canais são:

Capacidade do *RF Front-End*: A função do RF Front-End é deslocar uma janela de interesse para uma freqüência intermediária (IF), onde o sinal é amostrado por ADCs e o inverso para a transmissão. Cada RF Front-End possui uma faixa de freqüência de trabalho com limites máximo e mínimo.

Janela de amostragem: Após o deslocamento para IF uma faixa do espectro é amostrado pelo ADC, esse por sua vez tem limitações ditadas pelo teorema de *Nyquist-Shannon*, onde a freqüência de amostragem tem que ser o dobro da freqüência de interesse. Ou seja, para uma janela de 100MHz o ADC deve trabalhar amostrando a 200MHz.

Interface de comunicação: O hardware transmite os canais selecionados para uma unidade de processamento onde será executado o software que define o comportamento da camada física. Se o processador estiver ligado via interfaces de alta velocidade, esse limite será maior que a freqüência de trabalho do ADC, tendo pouca influência no sistema. Entretanto, não é raro a utilização de interfaces USB ou Giga Ethernet, limitando a transferência de espectros com janelas de 8MHz ou 25MHz, respectivamente.

As equações (1), (2) e (3), apresentam a relação entre a freqüência central (f_{cx}) e a largura de banda (L_{cx}) de um canal qualquer com os limites impostos pela capacidade do *RF Front-End* (f_{FEmin} e f_{FEmax}), pela largura da janela de amostragem do ADC (L_{max}) e pela capacidade da interface de comunicação (LI_{max}). A Tabela 2 resume os parâmetros utilizados nas equações.

$$f_{FEmin} \leq f_{cx} \leq f_{FEmax} \quad (1)$$

$$L_{cx} \leq L_{max} \tag{2}$$

$$L_{cx} \leq LI_{max} \tag{3}$$

Parâmetro	Variável
Freqüência central de um canal *x*	f_{cx}
Largura de banda de um canal *x*	L_{cx}
Limites mínimo e máximo das freqüências de trabalho do *RF Front-End*	f_{FEmin} e f_{FEmax}
Largura máxima da janela de amostragem do ADC	L_{max}
Largura máxima da janela transmitida pela interface de comunicação	LI_{max}

Tabela 2. Características do canal e limitantes do recurso do sistema.

Além dos fatores estáticos existem a concorrência dos recursos pelos canais, ou seja, dinamicamente os canais concorrem por fatias do espectro e a soma dos recursos alocados não podem ultrapassar os limites estabelecidos do sistema. Assim a alocação de todos os canais gera uma janela que possui uma freqüência inferior (B_i) e um freqüência superior (B_s) que tem relação com os canais de menor (f_{cmenor}, L_{cmenor}) e maior freqüência (f_{cmaior}, L_{cmaior}), respectivamente. As equações (4) e (5) apresentam essa relação.

$$B_i = f_{cmenor} - \frac{L_{cmenor}}{2} \tag{4}$$

$$B_s = f_{cmaior} + \frac{L_{cmaior}}{2} \tag{5}$$

Após calcular as freqüências inferior e superior da janela formada pela alocação dos canais, os limites impostos pela capacidade do *RF Front-End* (f_{FEmin} e f_{FEmax}) e pela largura da janela de amostragem do ADC (L_{max}) devem ser verificados. As equações (6), (7) e (8) apresentam essa relação.

$$B_i \geq f_{FEmin} \tag{6}$$

$$B_s \leq f_{FEmax} \tag{7}$$

$$B_s - B_i \leq L_{max} \tag{8}$$

Finalmente, após amostrados, cada canal gera uma quantidade de dados proporcional a sua largura de banda L_c. Assim, a somatória da largura de banda de todos os canais alocados deve ser menor ou igual a capacidade de transmissão da interface de comunicação (equação (9)).

$$\sum_{i=0}^{n} L_{ci} \leq LI_{max} \tag{9}$$

O Algoritmo 1 apresenta o pseudocódigo para uma das possíveis implementações da verificação dos recursos do sistema e alocação de uma lista de canais seguindo as equações descritas acima.

Algorithm 1 Verificação dos recursos do sistema para alocação de uma lista de canais.

```
l_ca = 0
for i := 1 to n_channels do
  if f_FEmin ≤ f_c[i] ≤ f_FEmax then
    l_ca = l_ca + l_c[i]
    v[ ].insere_ordenado(f_c − L_c/2)
    v[ ].insere_ordenado(f_c + L_c/2)
    if (v[ ].max() − v[ ].min() > L_max) or (l_ca > LI_max) then
      error("Impossivel alocar todos os canais")
    end if
  end if
end for
```

4. Implementação e Cenários de Aplicação

A arquitetura proposta foi implementada utilizando a placa USRP2 e o GNU Radio. A USRP2 sofreu modificações no hardware reconfigurável, para adição dos blocos responsáveis pela separação dos canais e implementação das estruturas de suporte. Além disso, a lógica de controle foi modificada para fazer a verificação e roteamento apropriado do fluxo de amostras para cada canal adicionado. O número de canais é limitado pelos recursos da FPGA da plataforma. Já no GNU Radio foi criado um novo par de blocos *Source* e *Sink*, os quais implementam as interfaces de configuração e recepção/transmissão, onde os canais podem ser alocados dinamicamente seguindo as restrições apresentadas na Seção 3.

Para os testes e análise do desempenho foram utilizadas uma implementação do demodulador multi-canal IEEE 802.15.4 da UCLA [Choong 2009] seguindo os componentes tradicionais do GNU Radio e uma implementação equivalente utilizando a arquitetura de canais proposta. As duas implementações foram executadas no mesmo PC com processador Intel QuadCore de 2.83 GHz, com 4GB de RAM e rodando Ubuntu 9.10 com o kernel 2.6.31-19. A placa de rede gigabit utilizada para interface com a USRP2 foi a Broadcom BCM5755 integrada.

O ambiente de experimentações montado é composto por 4 motes MicaZ, uma USRP2 e um *host*, como mostrado na Figura 6. Os motes MicaZ utilizam o rádio CC2420 com a implementação do padrão 802.15.4, cada um transmitindo em um canal, que por limitações da implementação padrão, devem ser vizinhos. Cada mote envia mensagens contendo apenas seu identificador, que servem apenas para *debug*, onde o intervalo e o sincronismo não fazem diferença, pois a ocupação do canal não influência no desempenho

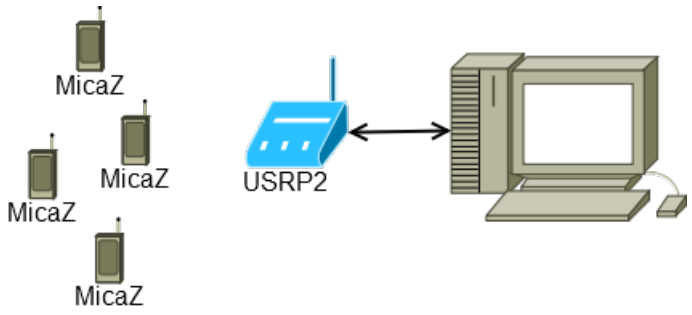

Figura 6. Ambiente de testes montados.

dos testes. Cada implementação foi executada com prioridade "tempo real" e testada durante 300 segundos com dados de performance sendo lidos em intervalos de 2 segundos.

O primeiro teste foi executado utilizando a implementação tradicional, que possui a estrutura mostrada na Figura 7. A predominância dos blocos em software e do paralelismo das funções como "Translação e DDC" em altas taxas de amostras por segundo são a causa principal do baixo desempenho da implementação. Tais resultados são mostrados na Figura 8, que apresenta três medidas principais: *IDLE*, *USR* e *SYS*.

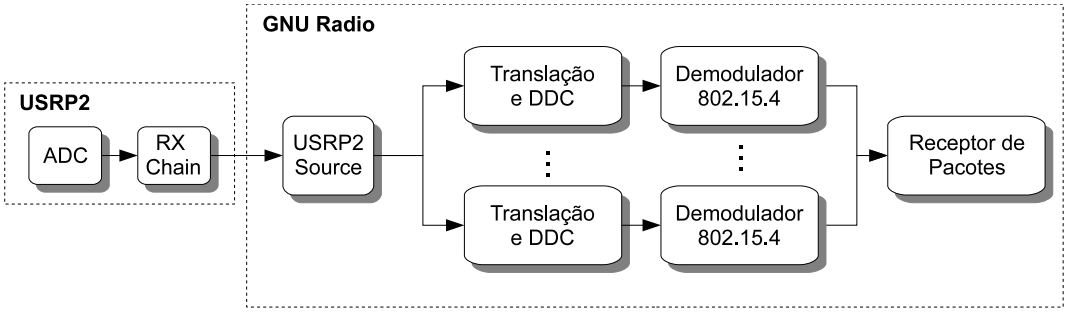

Figura 7. Diagrama da implementação tradicional.

A curva *USR* mostra a porcentagem de utilização das aplicações executados em espaço de usuário. Considerando o mesmo ambiente de teste e a prioridade "tempo real" da tarefa que executa a aplicação do rádio, consegue-se comparar o impacto no sistema das diferentes implementações. A curva *SYS* representa a porcentagem de utilização da *CPU* por aplicações/serviços em espaço de sistema. Finalmente, a curva *IDLE* apresenta a porcentagem da capacidade de processamento livre da *CPU* durante os testes.

A estrutura definida para o segundo teste, desenvolvida sobre a arquitetura de canais proposta, pode ser vista na Figura 9. O deslocamento do processamento dos canais para o hardware reconfigurável permite que a janela do espectro de freqüência capturada pelo *RX chain* seja "fatiada" em canais com freqüências centrais (FC) e larguras distintas. Cada um desses parâmetros pode ser configurado pelo *host*, não diminuindo em nada a flexibilidade alcançada pelos blocos de software utilizados no primeiro teste, conseguindo assim, o mesmo nível de configuração em tempo de execução. Utilizando essa estrutura, cada camada física pode livremente escolher a sua freqüência de trabalho, com a possibilidade do uso de um ou mais canais. As medições de desempenho podem ser vistas na Figura 10.

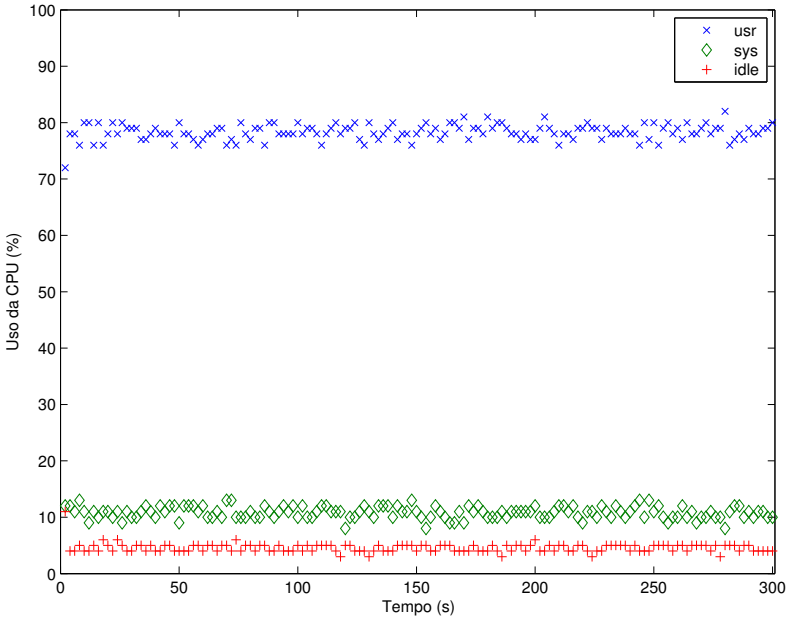

Figura 8. Resultados dos testes de desempenho da implementação tradicional.

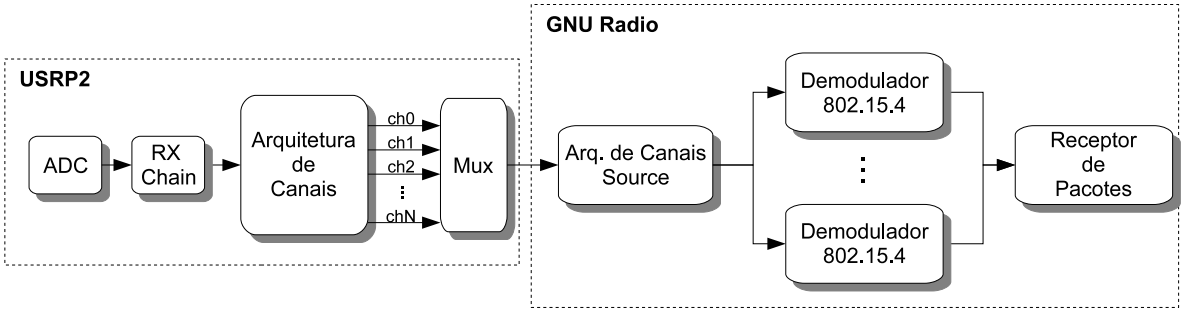

Figura 9. Diagrama da implementação com a arquitetura de canais.

A Figura 11 apresenta as médias de ocupação da CPU. A diminuição de 64,3% de ocupação da CPU quando a arquitetura de canais é utilizada mostra um ganho de performance significativo para execução da mesma tarefa. Além disso, a diminuição do fluxo de dados entre a USRP2 e o *host* traz diversos benefícios, como por exemplo, a diminuição do número de interrupções e consequentemente o *overhead* do sistema para o tratamento das mesmas, o que pode ser visto com a diminuição de 49,3% na ocupação da CPU.

Uma das limitações mais drásticas no modelo original é largura máxima da janela transmitida pela interface de comunicação que interliga a USRP2 e o *host* ($LI_{max} = 25MHz$), sendo um dos fatores estáticos discutidos na Seção 3. Assim, outra vantagem da capacidade do gerenciamento de canais diretamente no hardware é o melhor aproveitamento dos recursos do sistema. Com a possibilidade de separar somente as fatias do espectro que contém informações relevantes nos estágios iniciais, a propagação de dados sem informação útil gera uma economia na largura de banda das interfaces de comunicação.

Por exemplo, o padrão 802.15.4 prevê o uso de 16 canais em uma janela de

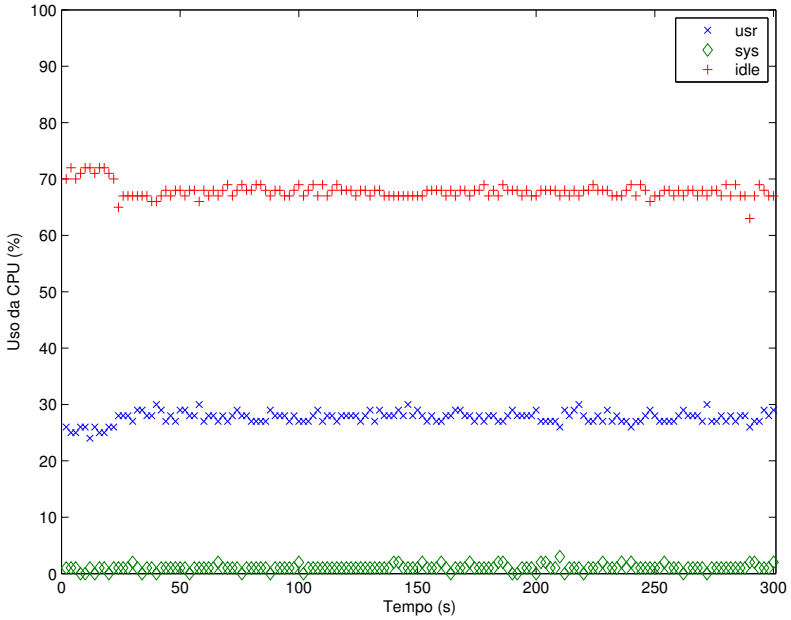

Figura 10. Resultados dos testes de desempenho da implementação com a arquitetura de canais.

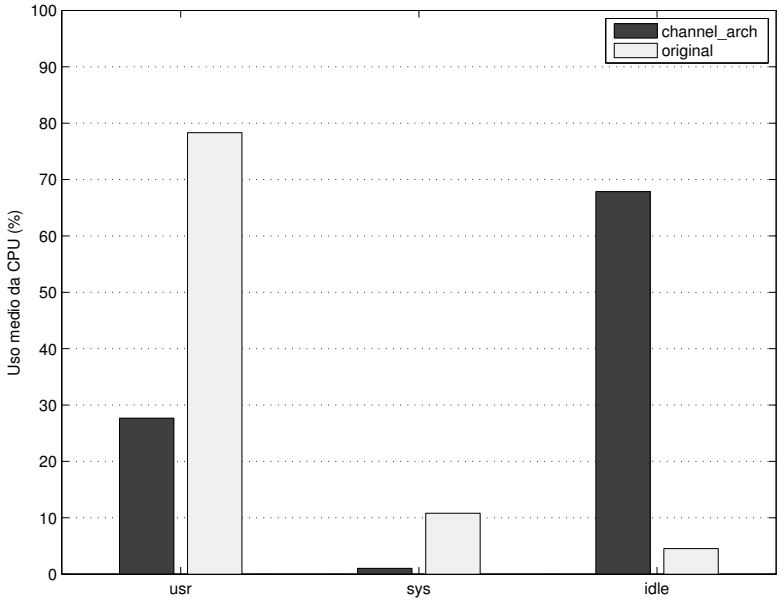

Figura 11. Ocupação média da CPU.

freqüência de 2400MHz até 2483.5MHz. Cada canal tem uma largura de 2MHz e um espaço entre canais de 3MHz. Com a implementação original, uma janela de 25MHz é enviada ao *host* sem processamento de canais prévio, ou seja, as amostras referentes aos espaços do espectro sem informação útil ocupam banda do canal de comunicação. Assim, podemos considerar que cada canal na verdade ocupa 5MHz e pela Equação (9) chegamos a um número máximo de apenas 5 canais e ainda com a limitação de serem consecutivos, como abordado anteriormente. O aproveitamento da janela de 25MHz é de apenas 10MHz de informações úteis (40%). Com a arquitetura proposta, podemos pro-

gramar cada canal separadamente enviando apenas 2MHz de informação por canal para o *host*. Permitindo assim, até 12 canais sem a necessidade de serem consecutivos, com um aumento do aproveitamento da janela de 25MHz de 96%.

5. Conclusões

Esse artigo apresentou uma nova arquitetura para SDRs, que propôs o conceito de desacoplamento de canais da camada física, possibilitando o deslocamento das fases com alto consumo de processamento, devido as altas taxas de amostras por segundo, para o hardware reconfigurável. Isso permitiu uma ganho de performance significante sem qualquer perda de flexibilidade. A arquitetura foi implementada utilizando o GNU Radio e a USRP2, e os testes comparativos entre a implementação original e a arquitetura proposta demonstraram ganhos significativos no aproveitamento dos recursos do sistema, relacionados mais especificamente ao desempenho e as interfaces de comunicação. Com a exigência de um desempenho menor para processar os blocos de software, a possibilidade do uso dessa tecnologia em sistemas embarcados torna-se factível.

Como trabalhos futuros, o porte da arquitetura de canais para plataformas embarcadas será finalizado, permitindo análises dos benefícios da utilização da proposta em um ambiente embarcado real. Além disso, problemas relacionados com a latência de comunicação entre o momento em que o sinal é capitado pela antena e as amostras chegam r nos blocos de software e a melhor integração entre os projetos da camada física e o MAC despontam como grandes desafios para a área.

Referências

Ackland, B., Raychaudhuri, D., Bushnell, M., Rose, C., and Seskar, I. (2005). High performance cognitive radio platform with integrated physical and network layer capabilities. Technical report, http://www.winlab.rutgers.edu/pub/docs/NeTS-ProWiN1.pdf.

Balister, P., Tsou, T., and Reed, J. H. (2007). Software defined radio on small form factor systems.

Blossom, E. (2009). Gnu radio. http://www.gnu.org/software/gnuradio.

Choong, L. (2009). Multi-channel ieee 802.15.4 packet capture using software defined radio. Technical report.

Ettus, M. (2009). Universal software radio peripheral. http://www.ettus.com/.

KUAR (2009). Kansas university agile radio. https://agileradio.ittc.ku.edu/.

Mccarthy, N., Blossom, E., Goergen, N., Oshea, T., and Clancy, C. (2008). High-performance sdr: Gnu radio and the ibm cell broadband engine. In *Virginia Tech Wireless Personal Communications Symposium*.

Mitola, J. (1995). The software radio architecture. *Communications Magazine, IEEE*, 33(5):26–38.

Reed, J. (2002). *Software radio: a modern approach to radio engineering*. Prentice Hall Press, Upper Saddle River, NJ, USA.

Tennenhouse, D. L. and Bose, V. G. (1996). The spectrumware approach to wireless signal processing. *Wireless Network Journal*, 2.

WARP (2009). Rice university wireless open-access research platform (warp). http://warp.rice.edu.

Welborn, M., Rao, S., Nathuji, R., Tuchinda, R., Ankcorn, J., Garland, S., and Guttag, J. (2009). Spectrumware project. http://nms.csail.mit.edu/projects/spectrumware/.