GNU RADIO E INDEPENDÊNCIA DO HARDWARE EM SISTEMAS EMBARCADOS: CONSIDERAÇÕES SOBRE A APLICABILIDADE DE SCA COMO ALTERNATIVA EM BUSCA DE MAIOR FLEXIBILIDADE

MARIA SÍLVIA ITO
RAFAEL SCHENA

Engenharia Elétrica
Universidade de Brasília - UnB

http://www.ene.unb.br ***e-mails: maria.silvia.ito@gmail.com***
rafaelschena@gmail.com

Resumo – Um Rádio Definido por *Software* ideal é versátil, opera em vários modos e várias bandas. Para se alcançar essa versatilidade é necessária a criação de um conjunto de regras que padronize um sistema de rádio para que vários deles possam interoperar entre si. Uma das arquiteturas mais aceita atualmente como realização dessas regras é a SCA, que, por sua vez, utiliza CORBA como *middleware*. Isto possibilita o uso de várias plataformas no funcionamento de um rádio de forma que esse opere com outros rádios de plataformas diferentes. Esta proposta utiliza GNU *Radio* para fazer o processamento de sinais de modo flexível sobre um único processador. Utiliza-se também a placa USRP, como *front-end*. Por fim, é levantada a hipótese de se embarcar o sistema de RDS proposto em dispositivos reconfiguráveis, para assim possibilitar a portabilidade do sistema.

.

Abstract – An ideal *Software* Defined Radio is versatile and operates in multiple modes and bands. In order to achieve the desired versatility, the creation of a set of rules is necessary to standardize a radio system so that the radios could interoperate. One of the most accepted architectures as an accomplishment of this set of rules is SCA, wich uses CORBA as middleware. CORBA allows the use of several platforms in the functioning of a radio so that it operates with other radios using different platforms.This proposal uses GNU Radio,to process signals in a flexible way over a single processor, with the USRP board as front-end. Finally, the hypothesis of embedding the SDR system proposed in reconfigurable devices is raised, so that the portability of the system is allowed.

Keywords – SDR, Reconfigurability, SCA, CORBA, USRP, GNU Radio

1 Introdução

O projeto de um Rádio Definido por *Software* (RDS) almeja produzir um único transceptor reconfigurável capaz de atuar, por exemplo, como telefone sem fio, celular, GPS (*Global Position System*), e outras funções, e que seja operável em qualquer lugar do mundo. Daí é necessário um dispositivo multi-modo e multi-banda para que esse esteja apto a receber várias interfaces aéreas e bandas de transmissão.

Existe uma grande variedade de tecnologias, tanto de suporte ao *software*, como CORBA (*Common Object Request Broker Architecture*) e MVR (Máquina Virtual de Rádio), quanto de *hardware*, como DSP (Processador Digital de Sinais), FPGA (Matriz de Portas Lógicas Reconfiguráveis) e ASIC (Circuito Integrado de Aplicação Específica), existentes no mercado e que vêm melhor atender determinadas necessidades.

Das três tecnologias de *hardware* citadas, o DSP apresenta maior capacidade de reprogramação. Apresenta, também, um bom desempenho, porém com um consumo maior de potência, podendo inviabilizar ou dificultar aplicações com dispositivos portáteis. O ASIC, em oposição ao DSP, privilegia o desempenho com baixo consumo, porém apresenta baixa capacidade de reprogramação. Já o FPGA possui desempenho e reprogramação

intermediários se comparado ao DSP e ASIC.

Quanto às tecnologias de suporte ao *software*, a MVR representa uma maior flexibilidade e portabilidade do *software*, com baixo custo e integração dos mercados militar e comercial, com a conseqüente redução dos custos dos produtos militares. MVR, em conjunto com MVJ (Máquina Virtual Java), promove um *middleware* comum para desenvolvedores de *software* e *hardware*, simplificando e otimizando, assim, o desenvolvimento do RDS [1,2].

Com a profusão de tecnologias de *hardware* e configurações de interfaces aéreas, deve haver a padronização de procedimentos para a reconfiguração do rádio. Daí a necessidade de um *middleware* que possibilite o uso de várias plataformas *software-hardware*, na tentativa de integrar vários sistemas de rádio existentes.

O JTRS (*Joint Tactical Radio System* – Sistema de Rádio Tático para Operações Combinadas), um projeto militar americano com a finalidade de integrar rádios militares, estabeleceu, para a definição de elementos interoperáveis a Arquitetura de Comunicações de *Software* (SCA - Software *Communications Architecture*), uma arquitetura aberta que possibilita a gerência e interconexão de recursos de *software* em um ambiente computacional distribuído embarcado [3]. A SCA fornece um conjunto de regras focalizado nas especificações e nos padrões detalhados do desenvolvimento do rádio – incluindo a interface aérea e o *software* aplicativo – que detalha o que deve ser feito para tornar o sistema interoperável e para se ter equipamentos, *software* e outros componentes de rede permutáveis

Como *middleware* de sua estrutura central, o SCA utiliza CORBA, especificação desenvolvida pela OMG (*Object Management Group*). O CORBA é uma infra-estrutura aberta e independente de fabricante, que fornece interfaces e modelos independentes de plataforma para aplicações computacionais distribuídas e portáteis. Como o SCA e, conseqüentemente, o CORBA estão progressivamente sendo adotados como padrão pelo Fórum de Rádio Definido por *Software* (*Software Defined Radio* Forum– SDRF)[4], tornam-se candidatos naturais de um estudo mais aprofundado e estão sendo explorados como objeto de pesquisa do grupo de Rádio Definido por *Software* da Universidade de Brasília.

Por outro lado, o grupo de trabalho já utiliza a plataforma formada pela placa USRP (*Universal Software Radio Peripheral*) e o GNU *Radio* como base inicial de pesquisa na área. Nesta plataforma, a placa USRP é responsável pela conversão A/D e D/A e pela transmissão dos sinais digitalizados para o PC, enquanto o GNU *Radio* é uma biblioteca de blocos de processamento de sinais para a construção de RDS. Como esta plataforma USRP + GNU *Radio* apresenta limitações por não manter independência do *hardware* do rádio, uma idéia sendo explorada na pesquisa e alvo deste artigo é o estudo da adequabilidade da especificação SCA a esta plataforma. Em outras palavras, a arquitetura a ser proposta neste artigo pode ser usada na construção de um protótipo de RDS, independente de um PC. A camada de *hardware* é composta por uma antena e um dispositivo de processamento reconfigurável, enquanto a camada de *software* é composta basicamente pelo GNU *Radio* rodando sobre um kernel Linux. Existe ainda uma camada de *middleware* (CORBA) responsável pela independência do sistema em relação ao *hardware* utilizado.

Para explanar as bases dessa proposta, o artigo está organizado da seguinte maneira. A seção 2 trata sobre os principais aspectos de SCA. A seção 3 explana a estrutura da plataforma baseada em GNU Radio e a placa USRP e a seção 4 apresenta limitações dessa plataforma. A seção 5 aponta para as direções a serem tomadas no decorrer da pesquisa e a seção 6 conclui o artigo, apresentando as expectativas quanto ao desenvolvimento da pesquisa.

2 SCA e *middleware* CORBA

SCA foi desenvolvida na busca do dos futuros sistemas de comunicação, fazendo uso dos avanços das tecnologias já existentes, com a finalidade de enfatizar ainda mais a interoperabilidade de um sistema e reduzir custos de desenvolvimento e distribuição [5]. Sua estrutura pode ser observada na Figura 1. Como a padronização é a chave para a aceitação de uma tecnologia, o programa JTRS está cooperando com o Fórum de Rádio Definido por *Software* (*Software Defined Radio Forum* - SDRF) e a OMG. O SDRF

está envolvido no desenvolvimento da SCA, com a finalidade de garantir a conformidade com necessidades tanto comerciais como militares [3].

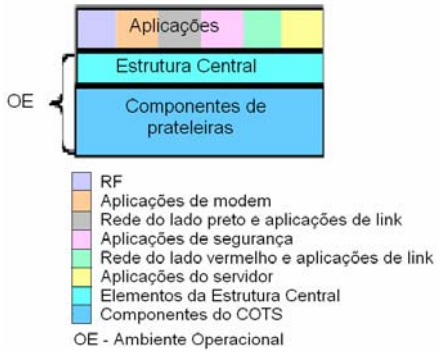

Figura 1 – estrutura da arquitetura da SCA

A versão explorada na pesquisa é a SCA 3.0, que descreve e explica detalhadamente as exigências dessa especificação. Entre as descrições encontra-se o ambiente operacional, arquitetura do *hardware* e do *software* e INFOSEC.

Uma das bases da SCA é a utilização de CORBA como *middleware*. O CORBA é uma infra-estrutura aberta e independente de fabricante que fornece interfaces e modelos independentes de plataforma para aplicações computacionais distribuídas e portáteis [1]. É recomendada para o desenvolvimento de novas aplicações e a integração destas a sistemas existentes.

O CORBA é usada no fornecimento de serviços de *middleware* de plataforma cruzada (*cross-platform middleware service*), que simplificam operações cliente-servidor padronizadas em um ambiente distribuído ao omitir os mecanismos de comunicação sob um barramento de *software* ORB (*Object Request Broker software bus*). Um objeto CORBA é uma entidade virtual capaz de ser localizada por um ORB. O objeto é virtual no sentido de que este não existe se não for concretizado pela sua implementação escrita em uma linguagem de programação. Um cliente é uma entidade que faz pedidos ao objeto CORBA.

O funcionamento do CORBA, seus clientes e objetos estão esquematizados na Figura 2. Cada objeto possui uma referência, que funciona como um endereço do mesmo. Um cliente faz um pedido a um objeto da seguinte maneira: o pedido do cliente passa pelo IDL *stub* para o ORB e depois pelo IDL *skeleton* para, finalmente, chegar ao objeto onde o pedido é executado. Tanto o *stub* quanto o *skeleton* esquematizados são interfaces bem definidas em IDL (Linguagem de Definição de Interface).

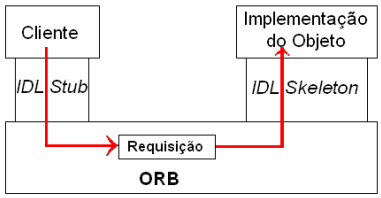

Figura 2 – Pedido realizado de cliente para objeto

Com a finalidade de requisitar um objeto remoto (Figura 3), o cliente precisa obter a referência do objeto, e com esta, o cliente faz o pedido ao objeto, como se este fosse um objeto local. O ORB do cliente localiza o ORB em que o objeto se encontra e o pedido é feito. Mas para que o pedido passe entre diferentes ORBs é necessário que ambos os ORBs operem em um protocolo comum, o IIOP (*Internet Inter ORB Protocol*) [6].

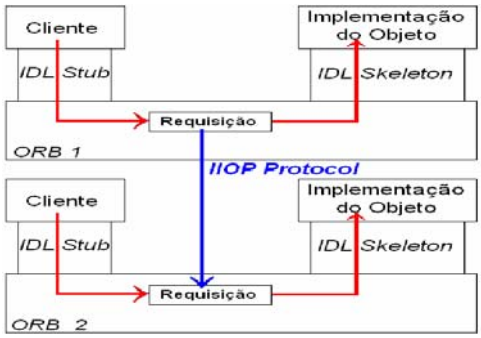

Figura 3 – A interoperabilidade usa comunicação de ORB para ORB

3 A plataforma de RDS formada pela placa USRP e o GNU Radio

A placa USRP [8,9], foi desenvolvida por Matt Ettus para ser uma alternativa de baixo custo à implementação de modelos de RDS baseados no GNU Radio. A placa USRP, mostrada na Figura 4, atua como *um front-end*, fazendo conversões AD e DA, multiplexações e decimações.

A placa USRP é composta basicamente de:

a) quatro conversores A/D 64 MS/s 12-bit, para digitalizar o sinal vindo da antena na recepção. Tais conversores podem digita-

lizar um sinal de freqüência máxima de até 200 MHz com banda de até 32 MHz.
b) quatro conversores D/A 128 MS/s 14-bit, para tornar analógico o sinal a ser transmitido.
c) um FPGA Altera EP1C12 Q240C8 "Cyclone" para reduzir as taxas de transmissão (decimação) de dados para taxas compatíveis com uma conexão via USB. Ou seja, a decimação é feita no FPGA, apesar de seus parâmetros serem definidos no *software*, através de variáveis criadas no script Python. Outra função executada pelo FPGA é a multiplexação, para que até quatro sinais possam ser processados simultaneamente (dois recebidos e dois transmitidos). Tal multiplexação é necessária pois todos os sinais devem passar pelo mesmo canal da placa USRP para o PC.
d) dois *slots* para placas-filhas de transmissão e dois *slots* para placas-filhas de recepção.
e) Uma interface USB 2.0.

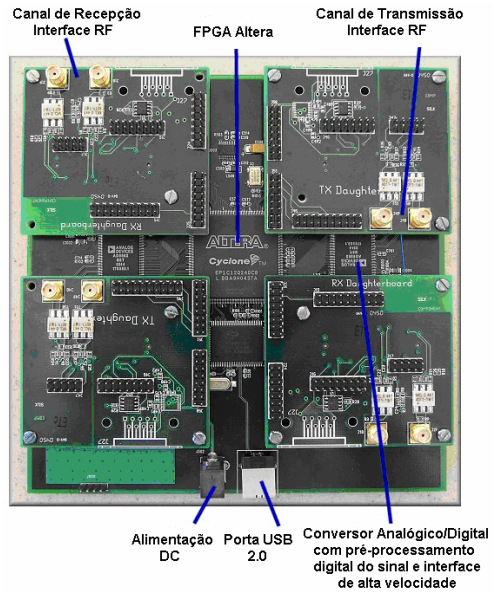

Figura 4 – Placa USRP com duas placas-filhas básicas de transmissão e duas de recepção.

A placa USRP juntamente com um PC e uma antena apropriada para recepção e transmissão dos sinais constituem o *hardware* necessário para a montagem desta plataforma. Já o *software* é composto pelo GNU Radio rodando sobre um *kernel* Linux com alguns programas auxiliares. O GNU Radio [10, 11] é uma biblioteca de blocos de processamento de sinais de código aberto, uma das diversas partes integrantes do projeto GNU de *software* livre, fornecendo um conjunto de ferramentas necessárias para a criação de um RDS. O GNU Radio fornece blocos de código em C++ para processamento de sinais e são conectados por *scripts* Python, uma linguagem interpretada e de alto nível, que acompanha muitas das distribuições Linux. Com isso, o programa não perde em desempenho, por ter suas funções críticas escritas em C++, e tem os fluxos de tratamento dos sinais montados facilmente, permitindo que qualquer usuário crie um RDS apenas criando um diagrama de blocos e implementando-o em Python.

A partir da plataforma apresentada até agora, pode-se construir um protótipo de RDS para a transmissão e recepção de sinais de áudio, com a estrutura do fluxo de dados para transmissão e recepção apresentada na Figura 5. Analisando a figura, verificam-se as principais etapas do seu funcionamento, tanto na transmissão quanto na recepção.

Na recepção, o sinal é recebido pela antena e então segue para a placa USRP, onde é digitalizado e decimado para reduzir a taxa de transmissão de dados. Segue então via USB para o PC, onde é, então, demodulado pelo GNU Radio e depois vai para a placa de som.

Na transmissão o processo é inverso: um sinal de áudio é captado por um microfone e segue para a placa de som, de onde é lido por um bloco do GNU Radio, e em seguida é modulado e segue para a placa USRP via porta USB. Lá chegando, o sinal é convertido pelo conversor D/A para então ser transmitido pela antena.

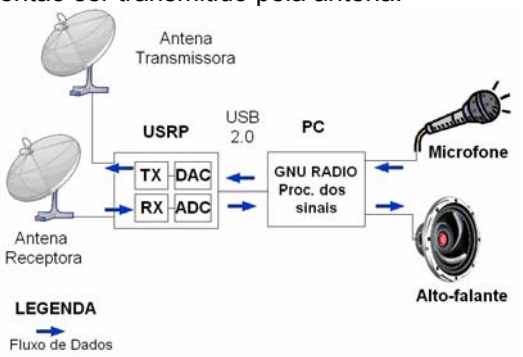

Figura 5 – Transmissão e recepção na plataforma USRP + GNU Radio.

4 Reconfigurabilidade oferecida pela plataforma USRP+GNU Radio e suas limitações

Dentro da filosofia de RDS, onde é imprescindível a capacidade de reconfiguração para que o sistema possa operar em outros modos, outras faixas de freqüência e

utilizando diferentes métodos de modulação e demodulação, a plataforma de RDS apresentada pode utilizar todos os módulos já presentes no núcleo do GNU Radio. Há também a possibilidade de se aproveitar o fato de o GNU Radio ser uma biblioteca de código aberto para modificar qualquer um dos seus módulos ou blocos de código para se adaptar a um problema específico, bem como criar soluções próprias. Na Figura 6, verifica-se um esquema mostrando apenas as etapas de aquisição e exportação dos dados via USB, a modulação e demodulação, e por fim, a aquisição e exportação dos dados da placa de som.

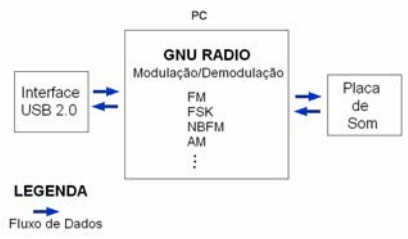

Figura 6 – Diversos métodos de modulação e demodulação já estão inclusos no GNU Radio

Apenas com a mudança de um *script*, pode-se substituir o método de modulação e demodulação e o tipo de sinal que se está sendo transmitido. É possível, por exemplo, receber um sinal modulado em amplitude de um ponto e depois receber um sinal com uma modulação FSK (*Frequency Shift Keying*), sem que, com isso, tenha-se a necessidade de se substituir o *hardware* utilizado.

Apesar de ser bastante didática e funcional, e da relativa versatilidade apresentada, essa plataforma apresenta limitações pelo fato de não guardar independência em relação ao *hardware* que a suporta. Por exemplo, poderia se pensar em um processador separado exclusivo para a realização de codificação e de decodificação convolucional. Neste caso, não há como, usando o GNU Radio, enviar o fluxo para a codificação ou para a decodificação nesse processador separado. Generalizando, isso resulta numa dificuldade em se obter soluções embarcadas em RDS baseadas no GNU Radio. Isso porque existe uma maior dificuldade em se explorar, da melhor maneira, a menor capacidade computacional e de armazenamento de dispositivos embarcados típicos, que podem oferecer diversidade de tecnologias, como FPGA e DSP, por exemplo.

5 Proposta de aplicação de arquitetura SCA e de *middleware* CORBA em um ambiente reconfigurável de RDS baseado no GNU Radio

A limitação da plataforma USRP + GNU Radio apresentada impõe restrições à utilização da grande variedade de ferramentas fornecidas pelo GNU Radio em uma plataforma independente do PC. Nesse sentido, é sugerida a aplicação da SCA à estrutura do GNU Radio. Assim, o protótipo de RDS poderá interoperar com outros rádios definidos pela SCA.

Na etapa seguinte do projeto, o CORBA será embarcado no sistema operacional Linux. Assim o GNU Radio poderá operar sobre diferentes plataformas. O uso de CORBA facilitará o processamento do sinal de forma que este, com o auxílio do *middleware* citado, poderá ser distribuído entre dois ou mais dispositivos de processamento, com a finalidade de não sobrecarregar o dispositivo de processamento reprogramável (ver Figura 7).

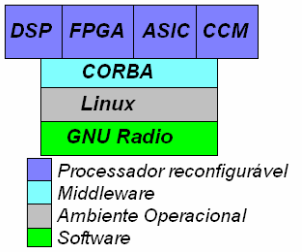

Figura 7 – Esquema de camadas a ser estudado

O uso de arquitetura SCA e do *middleware* CORBA permitirá, por exemplo, embarcar o GNU Radio em um FPGA. Neste caso o GNU Radio operará sobre um *kernel* Linux desenvolvido especificamente para este tipo de dispositivo de processamento, como é o caso do NIOS uClinux [12,13,14], uma distribuição Linux embarcada concebida especialmente para o processador NIOS, utilizado em FPGAs Altera.

6 Conclusão

A tecnologia de Rádio Definido por *Software* tem ainda um longo caminho a percorrer antes que possa estar presente na vida da maioria das pessoas. Contudo, essa realidade começa a ser vislumbrada com a reconfigurabilidade e a interope-

rabilidade sendo suas principais características.

Tentando mostrar este aspecto, o artigo faz uma breve incursão à SCA, que define regras para arquiteturas de RDS, e o CORBA, um *middleware* usado pela SCA para permitir uma maior flexibilidade no uso das diversas plataformas disponíveis. Por outro lado, tem-se o GNU *Radio*, que se utiliza de uma estrutura de *software* livre para permitir uma flexibilidade de soluções quanto ao processamento dos sinais em RDS. Juntamente com o GNU *Radio*, tem-se a placa USRP, formando uma plataforma para a construção RDS didática e funcional, porém limitada no sentido de ser dependente de um PC.

Assim, propõe-se a aplicação da SCA e do *middleware* CORBA, no que couber, ao GNU Radio, fazendo com que se possa prescindir de um dispositivo único de processamento, abrindo novas possibilidades ao uso do *software* livre. Isto possibilitará também a divisão do processamento entre dois ou mais dispositivos, uma vez que tais dispositivos reprogramáveis não possuem a mesma capacidade de processamento que os processadores utilizados em PCs atualmente.

7 Agradecimentos

Agradecimentos ao professor Eduardo Wolski pelo grande auxílio prestado durante a evolução dos estudos realizados pelo grupo em Rádio Definido por *Software*. Ficam também registrados os agradecimentos ao amigo e colega de projeto Francisco Augusto da Costa Garcia, que contribuiu muito nos estudos realizados pelo grupo desde a entrada deste no Projeto RDS – UnB.

8 Referências Bibliográficas

[1] Lima, André Gustavo, “Rádio Definido por Software: o próximo salto no mundo das telecomunicações e computação” (http://www.revdigonline.com/artigos_download/art_13.pdf)

[2] SILVA, Fernanda B. da, GARCIA, Francisco Augusto da C., TAKADA, Izumi Renata S., SASAKI, Marcello G, Princípio do RDS e Aplicação no DSP-10, DSPX e FPGA com Sistema Semi-embarcado, available on-line in: (http://www.revdigonline.com/artigos_download/art_2)

[3] Isomäki, Petri; Avessta, Nastooh, “An Overview of Software Defined Radio Technologies”(www.tucs.fi/publications/insight.php?id=tIsAv04a&table=techreport)

[4] “A Software Communications Architecture Compliant Software Defined Radio Implementation” (www.ece.neu.edu/info/architecture/publications/mbicer.pdf)

[5] JTRS SCA, disponível em (http://jtrs.army.mil/)

[6] OMG CORBA, disponível em www.omg.org

[7] Software Radio – a Modern Approach to Radio Engineering, Jeffrey H. Reed, Prentice Hall, 2002.

[8] UniversalSoftwareRadioPeripheral, (comsec.com/wiki?UniversalSoftwareRadioPeripheral)

[9] USRP User's Guide, (home.ettus.com/usrp/usrp_guide.html)

[10]GNU Radio - The GNU Software Radio, (http://www.gnu.org/software/gnuradio/)

[11]Dawei's GNU Radio Tutorials, (http://www.nd.edu/~dshen/GNU/)

[12]uClinux Embedded Linux - Microcontroller Project (www.uclinux.org)

[13] NIOS µClinux Project (http://www.enseirb.fr/~kadionik/embedded/uclinux/nios-uclinux.html)

[14]McCULLOUGH, David, “uClinux for Linux Programmers”, Linux Journal, disponível online em (www.linuxjournal.com/article/7221)

9 Biografias

Maria Silvia Ito cursa o sétimo semestre de Engenharia Elétrica na Universidade de Brasília – UnB, onde participa como voluntária do projeto de Rádio Definido por Software.

Rafael Schena é estudante de engenharia elétrica na Universidade de Brasília, onde cursa o sétimo semestre. Aluno do programa voluntário de Iniciação Científica – PIBIC – UnB/CNPq.