DoRiS: Um Novo Protocolo de Comunicação de Tempo Real sobre Ethernet e sua Implementação em Linux/Xenomai

Paul D. E. Regnier [1] **Antônio M. Carianha** [1] **George Lima** [1]

{**pregnier, gmlima, amcarianha**}**@ufba.br** *

[1] Laboratório de Sistemas Distribuídos (LaSiD) - Depart. de Ciência da Computação
Programa de Pós-graduação em Mecatrônica - Universidade Federal da Bahia
Campus de Ondina, 40170-110, Salvador-BA, Brazil

Abstract. *A new Ethernet-based real-time communication protocol, named* DoRiS*, is described.* DoRiS *has been implemented in Linux/Xenomai to make it suitable for supporting hybrid real-time systems, which are composed of hard and soft real-time services. By defining a distributed medium access control, the protocol provides temporal predictability, flexibility and fault tolerance. Also, an efficient network bandwidth utilization mechanism is provided, which is important for supporting services which demand high communication throughput. The proposed protocol has recently been formally specified and verified via model checking. This paper addresses its implementation. Experimental results indicate its good performance when compared to the real-time communication support currently provided in Linux/Xenomai.*

Resumo. *Um novo protocolo de comunicação para sistemas de tempo real sobre Ethernet, chamado* DoRiS*, é descrito. Sua implementação foi realizada no sistema operacional Linux/Xenomai, a fim de torná-lo adequado aos sistemas de tempo real híbridos, compostos de serviços de tempo real críticos e não-críticos. Baseado num controle de acesso distribuído, o protocolo fornece previsibilidade temporal, confiabilidade e tolerância a falhas. Além disso,* DoRiS *provê uso eficiente da largura de banda, característica importante aos serviços que precisam de altas taxas de transmissão. Recentemente, o protocolo foi especificado formalmente e sua correção foi atestada através do uso de verificadores de modelos. O presente artigo aborda sua implementação. Resultados experimentais indicam seu bom desempenho quando comparado ao suporte atualmente oferecido pelo sistema Linux/Xenomai.*

1. Introdução

Ainda recentemente, considerava-se que sistemas de tempo real eram compostos apenas de tarefas de tempo real críticas, simples e de curta duração, típicas das malhas de controle. Atualmente, os sistemas de tempo real envolvem novas aplicações ligadas às áreas de telecomunicação, multimídia, indústria, transporte, medicina, etc. Portanto, além das habituais tarefas periódicas críticas que requerem previsibilidade dos tempos de respostas, os sistemas atuais tendem a incluir tarefas não-críticas para dar suporte a eventos esporádicos ou aperiódicos [Liu 2000]. Outra característica da evolução dos sistemas

*Este trabalho recebeu apoio financeiro da FAPESB, projetos número 7320/2007 e 7630/2006.

modernos é a modificação das suas estruturas, que passaram de arquiteturas centralizadas para arquiteturas distribuídas. Sistemas distribuídos, compostos de aplicações com requisitos temporais variados, críticos e não-críticos, são chamados aqui de sistemas híbridos.

Para dar suporte a tais sistemas híbridos, a rede e os protocolos de comunicação devem ser adaptados para poder lidar com vários padrões de comunicação e oferecer qualidade de serviço adequada, tanto para as tarefas com requisitos temporais críticos quanto para as tarefas não-críticas ou aperiódicas que requerem elevadas taxas de transmissão. Em conseqüência, as redes industriais tradicionais tais como Profibus, ControlNet ou CAN [Lian et al. 2001, Decotignie 2005], por exemplo, conhecidas por serem deterministas e confiáveis, podem não dispor de largura de banda suficiente para atender a estas novas exigências. Uma alternativa às redes industriais tradicionais é o padrão Ethernet 802.3, devido a sua alta velocidade de transmissão e à disponibilidade de componentes de prateleira de baixo custo. No entanto, apesar da tecnologia Ethernet ser popular nas arquiteturas de redes a cabo e eficiente para a implementação de protocolos de comunicação um-para-muitos (*multicast*) [Dolejs et al. 2004], tanto a modalidade compartilhada (*half-duplex*) quanto a modalidade comutada (*full-duplex*) apresentam fontes de imprevisibilidade que dificultam o seu uso direto para aplicações com requisitos temporais críticos [Decotignie 2005].

Em face destas constatações, este trabalho apresenta a implementação de *DoRiS*, um novo protocolo de comunicação determinista e confiável, baseado em software e desenvolvido no Sistema Operacional de Tempo Real (SOTR) Linux/Xenomai. Tal protocolo, descrito na Seção 3, possui as seguintes características: (i) mantém a compatibilidade com interfaces Ethernet comuns, que respeitam a norma IEEE 802.3; (ii) oferece os modos de comunicação um-para-todos e um-para-muitos, desejáveis para sistemas de tempo real, sem perdas significativas de desempenho; (iii) provê mecanismos de tolerância a falhas de omissão e de parada; (iv) provê determinismo para a entrega de mensagens com requisitos temporais críticos; (v) provê um mecanismo inovador de alocação dinâmica da largura de banda para a comunicação com requisitos temporais críticos; (vi) otimiza o uso da largura de banda, que é compartilhada entre as comunicações críticas e não-críticas; (vii) é disponibilizado sob licença GNU/GPL na plataforma Linux/Xenomai.

O desenvolvimento de *DoRiS* foi ancorado em uso extensivo de métodos formais [Regnier et al. 2008a] e teve as suas primeiras idéias discutidas em [Regnier and Lima 2006]. O presente trabalho foca a descrição da implementação de *DoRiS* na plataforma operacional de tempo real Linux/Xenomai, cuja escolha foi motivada pelos resultados de um estudo extenso dos sistemas operacionais de tempo real (SOTR) baseados em Linux, apresentados em [Regnier et al. 2008b]. Atualmente, este SOTR oferece suporte restrito à comunicação de tempo real em termos de flexibilidade, não contemplando características dos sistemas híbridos. Desta forma, *DoRiS* provê uma significativa contribuição dado o crescente uso de Linux/Xenomai. Resultados experimentais indicam o bom desempenho de *DoRiS* quando comparado com o suporte atual oferecido por este SOTR.

Este artigo prossegue com a Seção 2, na qual discutimos trabalhos relacionados. Em seguida, a Seção 3 descreve detalhes do protocolo *DoRiS* importantes para entender a Seção 4, a qual apresenta a plataforma Linux/Xenomai e a implementação de *DoRiS*. A Seção 5 apresenta resultados experimentais e a Seção 6 conclui este texto.

2. Trabalhos relacionados

As propostas para aumentar a previsibilidade das redes Ethernet e oferecer garantias temporais às aplicações podem ser encontradas em grande número na literatura [Decotignie 2005]. Focamos aqui nas soluções baseadas em software, as quais consistem em estender a camada MAC (*Medium Access Control*) do protocolo CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*), utilizando a subcamada de controle lógico (LLC) como filtro para evitar ou reduzir a probabilidade de colisões.

Um dos mecanismos para oferecer garantias temporais é a divisão do meio físico em ciclos temporais com atribuição de *slots* temporais de emissão para cada estação. O exemplo mais conhecido desta idéia é o TDMA (*Time Division Multiple Access*), utilizado, por exemplo, para suporte a microcontroladores sobre Ethernet [Brito et al. 2004]. Apesar de prover determinismo, TDMA apresenta um problema de desempenho em situação de baixa carga da rede, pois, quando uma estação não tem nada para transmitir, nenhuma outra pode aproveitar o seu *slot* disponível. Uma outra posibilidade é a utilização do modelo Mestre-Escravo no qual uma estação gerencia a rede e distribui as autorizações de emissão por meio de mensagens para as demais estações. Baseado neste modelo, o protocolo FTT-Ethernet [Pedreiras et al. 2002] permite a coexistência de comunicação crítica e não-crítica. Porém, nesta arquitetura centralizada, o mestre constitui um ponto único de falha, que deve ser replicado a fim de prover tolerância a falhas.

Vários protocolos, tais como 802.5, FDDI, RTnet [Hanssen and Jansen 2003] e RETHER [Venkatrami and Chiueh 1994] utilizam a abordagem do bastão circulante (*token*) explícito para estabelecer as reservas e os direitos de acesso ao meio entre as estações organizadas em anel lógico. Apesar de conseguir oferecer garantias temporais, estas soluções apresentam limitações. Primeiro, o tempo de transmissão do bastão gera uma sobrecarga que pode ser significativa quando a maioria das mensagens são de tamanho mínimo. Segundo, no caso da perda do bastão, o tempo de recuperação causa um indeterminismo potencial na entrega das mensagens. Para resolver estas limitações, novas abordagens foram desenvolvidas usando mecanismos implícitos de passagem do bastão circulante [Carreiro et al. 2003]. Tais abordagens utilizam temporizadores para definir bifurcações no fluxo de execução do protocolo. Observando a comunicação, cada estação determina o instante no qual ela tem direito de transmitir em função da sua posição no anel lógico e do valor do seu temporizador. No entanto, desconhecemos a aplicação desta abordagem para contemplar as especificidades dos sistemas de tempo real híbridos.

Apesar da pesquisa sobre comunicação de tempo real em Ethernet ter produzido resultados relevantes citados, a implementação de novos protocolos para sistemas híbridos e baseados em Ethernet não tem sido observada em plataformas operacionais de tempo real de código aberto. Por exemplo, vários dos SOTR seguindo o padrão POSIX limitam-se a oferecer serviços de comunicação orientados à dispositivos, tais como sensores e atuadores [Kohn et al. 2004, Barbalace et al. 2008]. A implementação de DoRiS em Linux/Xenomai vem assim contribuir para dar maior flexibilidade a este sistema, que atualmente oferece apenas TDMA como suporte a comunicação de tempo real.

3. O protocolo *DoRiS*

Localizado nas camadas *LLC* e MAC do modelo OSI, o protocolo *DoRiS*, cujo significado em inglês é *An Ethernet Double Ring Service for Real -Time Systems*, atua como

um filtro lógico, tanto evitando as colisões inerentes ao compartilhamento do barramento Ethernet CSMA/CD quanto os atrasos não-determinísticos típicos de arquiteturas de redes baseadas em Ethernet comutada. *DoRiS* é concebido para dar suporte a sistemas híbridos nos quais a velocidade de processamento e as características da comunicação das aplicações com requisitos de tempo real críticos e não-críticos podem ser bastante diferentes. De fato, a maioria dos dispositivos industriais de prateleira tem capacidade de processamento pequena em relação à taxa de transmissão de Ethernet. Por exemplo, Carreiro et al [Carreiro et al. 2003] utilizaram microcontroladores que podem gastar até $111\,\mu s$ para processar uma mensagem de 64B. Como o tempo de transmissão de tal mensagem numa rede 100Mbps é $\delta = 5,76\,\mu s$, isso permite somente cerca de $5,2\%$ de utilização do barramento para a comunicação com requisitos temporais críticos. Por outro lado, aplicações não-críticas têm geralmente capacidades de processamento maiores e podem, portanto, utilizar taxas de transmissão mais elevadas. Chamaremos "estações lentas" os dispositivos industriais (microcontroladores, sensores, etc.) com velocidade de processamento relativamente baixa em comparação com a dos microcomputadores, os quais serão chamados de "estações rápidas".

O conjunto de estações (lentas e rápidas) interligadas, através de um barramento compartilhado ou de um comutador, constitui um segmento *DoRiS*. Cada estação do segmento executa um servidor *DoRiS*, o qual é responsável pela transmissão das mensagens críticas e não-críticas. Num servidor, distinguem-se duas tarefas de gerenciamento das duas filas de mensagens críticos e não-críticos, respectivamente chamadas de servidor critico e não-crítico. As estações lentas mantêm apenas o servidor crítico enquanto que as estações rápidas mantêm os dois servidores. As aplicações com requisitos temporais críticos são chamadas tarefas e são atendidas pelo servidor crítico, enquanto as aplicações com requisitos temporais não-críticos são chamadas processos e utilizam o servidor não-crítico. Num segmento *DoRiS*, o conjunto dos servidores críticos forma o anel crítico, enquanto o conjunto dos servidores não-críticos forma o anel não-crítico. Nestes dois anéis lógicos, um único bastão circula para autorizar o acesso ao meio das estações.

As mensagens enviadas pelas estações lentas são curtas, usualmente periódicas, e têm requisitos de tempo real críticos. Considera-se que tais mensagens, chamadas de "mensagens críticas", têm um tamanho de 64 bytes e que suas transmissões no barramento levam um tempo δ. Denota-se π o tempo de processamento, no pior caso, de tal mensagem pela estação mais lenta do segmento. Assumimos que: (i) $\delta \ll \pi$, – pois isto reflete a existência de estações lentas no segmento; (ii) a recepção e o processamento das mensagens são duas operações independentes que podem ser realizadas simultaneamente, – pois os dispositivos de hardware modernos são dotados de memórias locais e de capacidade de DMA (*Direct Memory Access*); e (iii) o sistema distribuído é síncrono, – pois o esquema de divisão temporal de *DoRiS*, tem pontos de sincronização regulares e previsíveis, que ocorrem dentro de uma janela de tempo curta comparada com o desvio dos relógios locais. A suposição (ii) implica, notadamente, que duas ou mais mensagens críticas podem ser enviadas seguidamente. Assumimos também que as estações podem falhar por *crash-recovery*, ou seja, uma estação pode parar e voltar a funcionar depois de um tempo arbitrário. Mensagens enviadas podem ser perdidas, porém, estações rápidas devem imperativamente perceber a interrupção associada à recepção de uma mensagem.

A comunicação num segmento *DoRiS* é temporalmente dividida em uma al-

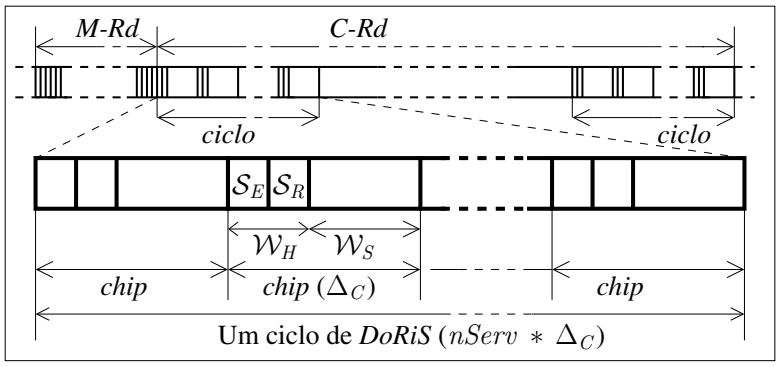

Figura 1. O esquema de divisão temporal de *DoRiS*

ternância de fases (*rounds*) de comunicação (*C-Rd*) e fases de configuração dos membros (*M-Rd*), assim como ilustrado na Figura 1. Durante a fase de configuração, o algoritmo de controle da composição do segmento *DoRiS* é responsável por estabelecer uma visão única do grupo de servidores, a qual é compartilhada por todos os membros do segmento. O problema do estabelecimento de tal visão, também conhecido como o problema de composição de grupos, é assunto de vários trabalhos [Cristian 1988]. Consideramos aqui, que o conjunto dos servidores é definido em tempo de projeto, ou seja, o número de servidores do segmento, denotado $nServ$, é suposto constante.

Usando TDMA, cada fase de comunicação (*C-Rd*) define um número arbitrário, porém fixo, de ciclos periódicos, os quais são subdivididos em exatamente $nServ$ *chips*, assim como pode ser observado na Figura 1. Um mapeamento dos naturais sobre o conjunto dos *chips* associa um inteiro positivo módulo $nServ$ a um *chip*. Cada *chip*, de tamanho Δ_C , é, por sua vez, dividido em duas janelas, crítica e não-crítica, denotadas $\mathcal{W}_H$ e $\mathcal{W}_S$, associadas, respectivamente, às comunicações de tempo real críticas e não-críticas. Os servidores críticos transmitem durante as janelas $\mathcal{W}_H$, enquanto os servidores não-críticos transmitem durante $\mathcal{W}_S$. Para permitir certa flexibilidade e a definição de políticas de escalonamento das mensagens, a janela $\mathcal{W}_H$ é subdividida em dois *slots*: o *slot* elementar ($\mathcal{S}_E$) e o *slot* de reserva ($\mathcal{S}_R$). As mensagens críticas transmitidas nos *slots* $\mathcal{S}_E$ e $\mathcal{S}_R$ são chamadas, respectivamente, de mensagem elementar e de reserva. Uma vez por ciclo, cada servidor envia uma mensagem elementar em $\mathcal{S}_E$, enquanto $\mathcal{S}_R$ é utilizada para oferecer um mecanismo de reserva de largura de banda. É importante ressaltar que cada servidor crítico deve imperativamente enviar uma única mensagem elementar por ciclo, no seu respectivo *chip*. Esta regra é necessária para permitir a tolerância a falhas, aumentar a confiabilidade do protocolo e garantir a sincronização dos relógios locais.

O controle de acesso ao meio de *DoRiS* é organizado por um bastão virtual, que circula nos anéis crítico e não-crítico, de acordo com regras temporais e lógicas baseadas na observação da comunicação no barramento e na sincronização dos relógios locais. A Seção 4.4 apresenta a implementação destes procedimentos. O isolamento dos dois anéis de *DoRiS* é garantido pelo uso do mecanismo TDMA. Em relação ao anel não-crítico, o bastão circula de uma estação para a próxima a cada nova janela $\mathcal{W}_S$. A otimização do uso da largura de banda neste mecanismo TDMA é garantida pelo gerenciamento da composição dinâmica do grupo de servidores não-críticos. Em caso de perda do bastão, na ocorrência de falhas, um mecanismo baseado em contador e na detecção das mensagens

é utilizado para criar um novo bastão.

Para aumentar a sua taxa de transmissão de mensagens críticas, um servidor crítico pode utilizar o seguinte mecanismo de reserva dinâmica de *slots*. Definimos o horizonte H_i de um servidor crítico SC_i como o conjunto dos $nServ$ *chips* seguindo o *slot* $\mathcal{S}_E$ no qual SC_i envia sua mensagem elementar. Cada mensagem elementar enviada por SC_i carrega uma lista de inteiros (módulo $nServ$), que especifica os identificadores dos *slots* de H_i que SC_i pretende usar para enviar mensagens adicionais. SC_i só pode reservar um *slot* que ainda não foi reservado por uma outro servidor crítico. Para tanto, cada servidor mantém localmente uma tupla de reservas.

4. A implementação de *DoRiS*

4.1. A plataforma operacional de tempo real e sua infra-estrutura de comunicação

A plataforma Linux/Xenomai – Para a implementação de *DoRiS*, escolhemos o sistema operacional de propósito geral Linux, [Bovet and Cesati 2005] versão 2.6.19.7, dotado do *patch* de tempo real Xenomai, versão 2.4-rc5 e do *nanokernel* Adeos correspondente [Gerum P. et al. 2009]. Num trabalho anterior [Regnier et al. 2008b], mostramos que este sistema operacional de tempo real (SOTR) oferece garantias temporais da ordem de dezenas de microsegundos, variando com o hardware adotado, o que é suficiente para a implementação de *DoRiS*. Para oferecer tais garantias temporais, Xenomai utiliza uma camada de abstração do hardware (HAL), localizada entre o *kernel* e os dispositivos de hardware, a qual disponibiliza uma interface de programação para serviços de tempo real. Para realizar comunicação de rede com garantias temporais, Linux/Xenomai disponibiliza a infra-estrutura de comunicação de tempo real RTnet [Kiszka et al. 2005] baseada na interface RTDM (*Real Time Driver Model*) [Kiszka 2005], descritas a seguir.

A infra-estrutura RTnet – RTnet provê controladores de placas de rede portados para Linux/Xenomai e uma nova implementação dos protocolos UDP e IP, onde causas de não-determinismos são evitadas, o que permite alcançar garantias temporais na comunicação. Os principais componentes do RTnet estão localizados entre a camada de rede e acima dos controladores de dispositivos e da camada HAL. Eles constituem a camada RTmac e o núcleo RTnet ilustrados na Figura 2. Resumidamente, destacamos três mecanismos essenciais da infra-estrutura RTnet: (i) a alocação estática de memória para pacotes (*rtskb*) em tempo de configuração evita atrasos imprevisíveis decorrentes de possíveis faltas de páginas; (ii) a atribuição da maior prioridade no sistema à tarefa de recepção de pacote, chamada "gerente da pilha", garante que um pacote que chega para uma tarefa crítica seja entregue com o menor *jitter* possível; (iii) a disponibilização de uma interface virtual, chamada VNIC (*Virtual NIC*), permite a integração da comunicação não-crítica de processos do Linux com as tarefas de tempo real que usam os serviços RTnet.

A utilização dos serviços do RTnet se concretiza pela definição de uma disciplina de acesso ao meio, cujo encaixe na infra-estrutura RTnet é realizado pela implementação das funções definidas pela API da camada RTmac. Na sua versão atual, RTnet disponibiliza duas disciplinas: TDMA e NoMAC. A primeira é baseada numa arquitetura centralizada do tipo mestre/escravo e oferece um serviço de comunicação com garantias temporais. Em tempo de configuração, os diferentes clientes se registram no mestre, que pode ser replicado por motivos de tolerância a falhas. Cada escravo reserva uma ou várias janelas de tempo, de acordo com as suas necessidades de largura de banda. A

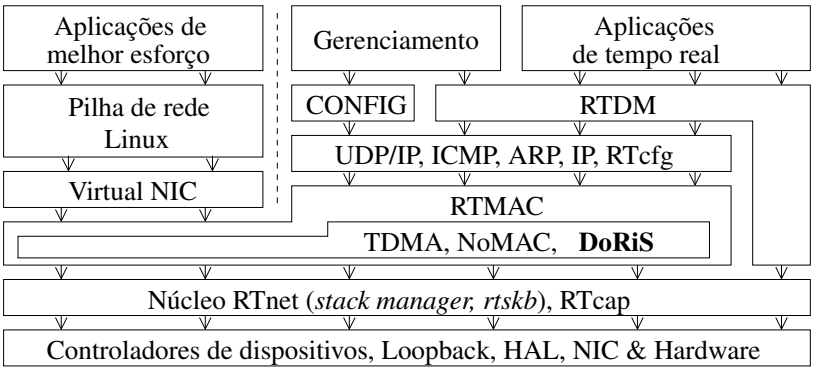

Figura 2. Estrutura em camada do RTnet / RTDM.

verificação da capacidade da rede em atender às diferentes aplicações requisitando largura de banda deve ser efetuada em tempo de projeto pelos desenvolvedores do sistema. A segunda, a disciplina NoMAC, simplesmente disponibiliza os serviços da pilha RTnet sem definir nenhuma política específica de acesso ao meio. No entanto, este esqueleto de implementação é disponibilizado para facilitar o desenvolvimento de novas disciplinas de acesso ao meio. Como pode ser visto na Figura 2, o protocolo *DoRiS* foi incorporado à infra-estrutura RTnet como uma disciplina adicional.

Serviços oferecidos por RTDM – Através desta interface de programação, as aplicações de tempo real requisitam serviços de comunicação, interagindo com a interface de alto nível de RTDM, a qual segue o padrão POSIX de *socket* e *ioctl*, enquanto a infra-estrutura RTnet interage com a interface de baixo nível de RTDM, a qual oferece uma compacta abstração dos serviços do SOTR subjacente. Dentre os vários serviços oferecidos pela interface de baixo nível para a implementação de uma nova disciplina na camada RTmac, existem notadamente: (i) Serviços de relógio – Permitem retornar o valor do relógio expresso em nanosegundos; (ii) Serviços de tarefas – Este grupo de serviços permite criar, destruir e suspender tarefas ou modificar suas características, como prioridade e periodicidade; (iii) Serviços de sincronismo – *Mutex* e semáforos são oferecidos, entretanto, variáveis do tipo evento são muito utilizadas, nas disciplinas TDMA e *DoRiS*, para realizar sincronismo entre funções de recepção e transmissão de pacotes. (iv) Serviços utilitários – Permitem utilizar e liberar memória ou realizar depuração utilizando interface texto em contexto de tempo real.

4.2. *DoRiS* como uma nova disciplina do RTnet

A implementação de *DoRiS* exigiu aproximadamente 1260 linhas de código, além de algumas modificações em 3 arquivos do núcleo RTnet. No entanto, estas modificações não interferem no funcionamento das demais disciplinas do RTnet. As funções fundamentais de *DoRiS* são implementadas através da definição dos campos de uma estrutura de dados padronizada pela camada RTmac. Esta estrutura permite definir as funções acionadas nos momentos de encaixe e desencaixe da disciplina, responsáveis pela inicialização das variáveis internas e liberação de recursos, respectivamente. Outras duas funções relacionadas com o fluxo de pacotes devem ser definidas nesta estrutura. Uma destas funções recebe os pacotes da comunicação crítica realizada pelas aplicações de tempo real, as quais utilizam a interface de alto nível de RTDM. A outra recebe os pacotes da comunicação não-crítica das aplicações do Linux, provenientes do VNIC. No caso da disciplina *DoRiS*,

estas funções colocam os pacotes recebidos nas respectivas filas crítica e não-crítica. Assim, cada servidor *DoRiS* utiliza três tarefas de tempo real para organizar o acesso ao meio Ethernet. A primeira tarefa, chamada *gerente da pilha* em referência ao RTnet, cuida da recepção assíncrona das mensagens. As segunda e terceira tarefas correspondem aos servidores crítico e não-crítico, respectivamente (ver Seção 3). Estes servidores administram as correspondentes filas de pacotes críticos e não-críticos.

Recepção – *DoRiS* utiliza o modelo de comunicação *publish-subscribe* [Dolejs et al. 2004], de acordo com o qual, quando uma aplicação quer enviar uma mensagem, ela utiliza o endereço Ethernet de comunicação um-para-todos padrão (`FF:FF:FF:FF:FF:FF`). Quando um servidor *DoRiS* recebe uma mensagem, ele determina, de acordo com a identidade do seu emissor, se há alguma aplicação interessada naquela mensagem. Para diferenciar os $nServ$ servidores do segmento *DoRiS*, o último byte do seus números IP, configurado em tempo de projeto, serve como identificador único. Esta escolha de implementação permite aproveitar os códigos baseados em IP do RTnet, deixando a possibilidade de se ter até 254 participantes num segmento *DoRiS*.

Para fins de eficiência, o gerenciamento das informações vinculadas ao protocolo, carregadas pelas mensagens, é realizado no tratamento da interrupção de recepção dos pacotes. Esta escolha de implementação permite descartar os pacotes que não são destinados à estação antes de acordar o gerente da pilha. Num evento de recepção de uma mensagem elementar, as principais operações de gerenciamento realizadas por um servidor crítico consistem em: incrementar os seus contadores de *chip* e de mensagens elementares; atualizar a sua tupla de reservas e a composição do anel não-crítico; atualizar o valor do contador $token$, o qual define a próxima estação do anel não-crítico que deverá transmitir; atualizar o valor do instante de início do próximo $\mathcal{S}_E$. Nesta última atualização, o instante de início do próximo *slot* elementar $\mathcal{S}_E$ de cada servidor é definido com a melhor precisão possível, o que aumenta o sincronismo dos relógios locais.

Após estas operações de gerenciamento do protocolo *DoRiS*, executadas pelo tratador de interrupção associado à chegada de um pacote, o servidor pode descartar a mensagem se ela não for destinada a alguma aplicação que ele hospeda. Caso contrário, o servidor coloca o pacote na fila de recepção do gerente da pilha, antes de acordá-lo. Assim que ele acorda, o gerente determina a aplicação que está esperando pelo pacote. Uma mensagem não-crítica é entregue para a pilha usual de Linux, enquanto que uma mensagem crítica é imediatamente encaminhada por *DoRiS* para o seu respectivo *socket* de tempo real, previamente aberto por alguma aplicação de tempo real com a interface RTDM. Percebe-se que o gerente da pilha deve ter a maior prioridade do sistema para garantir que uma mensagem de tempo real seja entregue com a menor latência possível.

Emissão de mensagens críticas – Para controlar as operações de emissão, os servidores utilizam temporizadores e condições lógicas, como foi visto na Seção 3. Para melhorar a precisão de sincronismo da operação de emissão, a tarefa de emissão de um servidor crítico SC_i é acordada pouco antes do instante t_i de início do seu próximo $\mathcal{S}_E$. Em outras palavras, esta tarefa acorda $t_i - d$, onde $d = \frac{\Delta_C}{2}$ é escolhido de tal forma a garantir que a estação acorda depois do último $\mathcal{S}_E$ que precede o seu próprio $\mathcal{S}_E$. Ao acordar, como SC_i dispõe da atualização mais recente de t_i, ele pode agendar o início do seu $\mathcal{S}_E$ com a melhor precisão possível. Quando a fila de um servidor crítico fica vazia, uma mensagem elementar é criada e enviada, conforme a especificação de *DoRiS*.

Emissão de mensagens não-críticas – As emissões de mensagens não-críticas são regidas pela circulação do bastão e por uma condição temporal que garante que estas mensagens sejam enviadas durante uma janela $\mathcal{W}_S$. O contador $token$ é atualizado a cada recepção de uma mensagem elementar. Para poder transmitir uma mensagem não-crítica m durante uma janela $\mathcal{W}_S$, um servidor não-crítico SNC_i deve verificar que as seguintes condições são verdadeiras: (i) ele está em posse do bastão, ou seja, $token = i$; (ii) o tempo ainda disponível em $\mathcal{W}_S$ é maior que o tempo necessário para a transmissão da m; (iii) SNC_i recebeu todas as mensagens elementares dos $nServ$ *chips* anteriores. A violação da condição (iii) indica a perda do bastão. Neste caso, um mecanismo de recuperação entra em ação. Por sua vez, se a condição (ii) não se verificar, SNC_i apenas adiará sua transmissão. Observa-se que o tamanho da janela $\mathcal{W}_S$ deve imperativamente ser maior que o tempo de transmissão de uma mensagem de tamanho máximo (1518 bytes). Assim, cada SNC pode enviar pelo menos uma mensagem a cada vez que ele detém o bastão.

5. Resultados experimentais

Usamos aqui cenários de comunicação simples para ilustrar o desempenho de *DoRiS*. Como será visto, *DoRiS* é capaz de entregar mensagens oferecendo garantias temporais da ordem de alguns microsegundos sem prejudicar o desempenho relativo à comunicação não-crítica. Foram utilizados três computadores Pentium IV, com processadores de 2.4 GHz, 512MB de memória e placa de rede Ethernet modelo *RealTek 8139*. Estas estações, denotadas E_1, E_2 e E_3, foram interligadas através de um comutador Ethernet 100Mbps dedicado. Os cenários de comunicação utilizados entre as três estações consistem em: (i) um fluxo de mensagens críticos (CC) entre duas tarefas T_1 e T_2, hospedadas, respectivamente, em E_1 e E_2; e (ii) um fluxo não-crítico (CNC) entre um processo P_1, hospedado em E_1, e um processo P_3, hospedado em E_3. Nos experimentos, considerou-se que apenas a tarefa T_1 e o processo P_1 enviam mensagens. No entanto, o sistema foi configurado de forma que qualquer estação poderia enviar e receber mensagens. Os cenários foram considerados usando as disciplinas NoMAC (Seção 5.1), TDMA e *DoRiS* (Seção 5.2).

5.1. Não-determinismo de NoMAC

Para ilustrar o não-determinismo de comunicação proveniente da ausência de controle de acesso ao meio, consideramos um cenário no qual T_1 envia mensagens de tamanho mínimo (64 bytes) com período $650\mu s$ e monitoramos a variabilidade na recepção das mensagens por E_2. Como esperado, na ausência de CNC, a comunicação crítica não sofreu atrasos observáveis. Os intervalos de tempo entre as chegadas de mensagens em E_2 apresentaram uma variabilidade máxima menor que $3\mu s$, abaixo de 0.4% em valores relativos. No entanto, adicionando o fluxo CNC, de período $1400\mu s$[1], observamos desvios da ordem de $200\mu s$ no pior caso (acima de 30% em valor relativo), o que é inaceitável em sistemas críticos. A Figura 3 ilustra tal fenômeno. O padrão observado nos valores dos picos é causado pelas relações de fase temporal entre os eventos de CC e CNC.

5.2. Estudo comparativo

O tempo total de transmissão de uma mensagem na rede é influenciado pelos seguintes fatores: (i) latência de interrupção da plataforma Linux/Xenomai; (ii) tempo de

[1]Apesar de ter observado comportamentos semelhantes com outros períodos, o valor $1\,400us$ foi escolhido por evitar interferências entre os eventos do fluxo CNC.

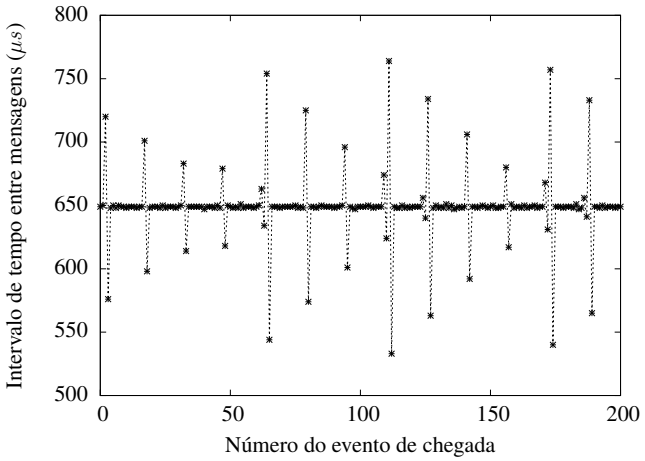

Figura 3. Disciplina NoMAC. Intervalos de tempo entre eventos de chegada de mensagens críticas consecutivas – períodos $CC = 650\mu s$ e $CNC = 1\,400\mu s$

cópia do pacote no hardware da placa de rede; (iii) transmissão da mensagem; (iv) atraso de comutação. A influência dos fatores (i) e (iv) é de aproximadamente $10\mu s$ [Regnier et al. 2008b] e causa pouca variabilidade na comunicação. Através de experimentos, aferimos que a transmissão de uma mensagem de tamanho mínimo (64 bytes) leva $26\mu s \pm 3\mu s$. Destes, o fator (ii) contribui com aproximadamente $10\mu s$. As mensagem de tamanho máximo (1518 bytes) são transmitidas em $230\mu s \pm 3\mu s$, com uma contribuição do fator (ii) de cerca de $100\mu s$. Deve ser observado que, no caso do envio de várias mensagens consecutivas pela mesma estação, o fator (ii) contribui apenas uma vez, pois a cópia de mensagens para o hardware da placa de rede é otimizado pelo efeito *pipeline*: enquanto uma mensagem está sendo transmitida a próxima já pode ser copiada para o hardware da placa de rede.

Uma vez determinada a influência dos fatores (i)-(iv), definimos $\delta = 30\mu s$ e configuramos *DoRiS* da seguinte maneira. Os *chips* de *DoRiS* foram configurados para as três estações, contendo portanto três janelas $\mathcal{W}_H$ de $60\mu s$. Consideramos um fluxo CC com periodicidade $1\,950\mu s$. Notar que neste experimento, o fluxo CC usa apenas o *slot* elementar $\mathcal{S}_E$ alocado a T_1, o que equivale a $1/6$ do total disponível dado que há dois *slots* para mensagens críticas por estação, $\mathcal{S}_E$ e $\mathcal{S}_R$. O *slot* de reserva $\mathcal{S}_R$ não foi usado neste experimento, pois este tem o objetivo de avaliar a capacidade de vazão do protocolo relativo ao fluxo CNC. Além disso, definimos $\Delta_C = 650\mu s$. É importante enfatizar que este valor de Δ_C permite a transmissão de três mensagens não-críticas de tamanho máximo em cada janela $\mathcal{W}_S$. De fato, como mencionado previamente, em torno de $230\mu s$ são gastos para a transmissão da primeira mensagem enquanto as demais levam um pouco menos de $130\mu s$ para serem transmitidas. Considerando os tempos relativos a $\mathcal{W}_H$ ($60\mu s$) e $\mathcal{W}_S$ ($230 + 2 \times 130\mu s$), reservamos $100\mu s$ para isolar os fluxos CC e CNC.

O protocolo TDMA foi configurado de maneira a oferecer a mesma alocação de largura de banda para os serviços críticos e não-críticos atendidos por *DoRiS*. O período TDMA foi definido em $650\mu s$ dentre os quais $200\mu s$ foram alocados para o quadro de sincronização TDMA enviado pela estação mestre. Em seguida, estão dois *slots* de $30\mu s$ para comunicação crítica, correspondendo aos *slots* $\mathcal{S}_E$ e $\mathcal{S}_R$ de *DoRiS*. Em seguida, três

slots de $130\mu s$ são definidos para a comunicação não-crítica. É preciso mencionar dois aspectos relativos à implementação de TDMA do RTnet. Primeiro, notamos que TDMA parece ter sido otimizado para escalonar as mensagens de forma a reduzir o efeito referente à cópia das mensagens na placa de rede, que mostrou-se mais acentuado em *DoRiS*. Isso é possível com TDMA, pois cada estação tem o conhecimento prévio do instante no qual ela irá transmitir suas mensagens não-críticas. Segundo, o tempo necessário para o quadro de sincronização não pôde ser reduzido. Após consultas ao grupo de desenvolvimento de RTnet, obtivemos informações de que tal tempo é função do hardware utilizado.

Como esperado, TDMA e *DoRiS* oferecem garantias temporais para o fluxo crítico, com variabilidade de tempo de recepção desprezível. No entanto, *DoRiS* mostrou-se superior a TDMA quanto ao fluxo CNC. De fato, $10\,000$ mensagens não-críticas de tamanho máximo foram transmitidas com TDMA em $6,6s$ enquanto *DoRiS* foi capaz de fazê-lo em $2,2s$. Esta diferença de desempenho reflete o fato de TDMA não poder modificar a alocação dos *slots* definida em tempo de projeto, enquanto *DoRiS* disponibiliza dinamicamente todas as janelas $\mathcal{W}_S$ para a única estação querendo transmitir. Portanto, com TDMA, uma mensagem não-crítica é enviada em cada *chip*, enquanto, com *DoRiS*, 3 são enviadas em cada *chip*.

Além disso, é importante observar que, com *DoRiS*, a alocação de largura de banda para os servidores críticos pode ser feita dinamicamente através dos *slots* de reserva, o que não ocorre com TDMA. Este mecanismo de *DoRiS* foi implementado sem impacto significativo no desempenho do protocolo.

6. Conclusão

Descrevemos o protocolo *DoRiS* e sua implementação no sistema Linux/Xenomai. O protocolo proposto foi incorporado a este sistema operacional como uma nova disciplina de comunicação de tempo real baseada em Ethernet. *DoRiS* vem atender às necessidades dos sistemas híbridos, para os quais tolerância a falhas, previsibilidade temporal e desempenho devem ser considerados conjuntamente. Como o uso de sistemas de tempo real baseados em Linux tem se tornado expressivo recentemente e a complexidade dos sistemas de tempo real tem exigido suporte adequado aos sistemas híbridos, a presente proposta amplia o espectro de aplicações contempladas por Linux/Xenomai. Resultados de experimentos mostraram que *DoRiS* consegue garantir determinismo temporal para serviços de tempo real críticos sem comprometer significativamente o uso da largura de banda de comunicação necessária aos serviços não-críticos.

Possíveis trabalhos futuros, baseados nas contribuições aqui apresentadas, podem ser considerados. *DoRiS* deverá ser disponibilizado sob licença de software livre como política adicional do Linux/Xenomai. Para tanto, alguns aspectos de documentação e padronização estão sendo considerados. Uma possível extensão de *DoRiS* é sua adaptação para redes sem fio. Acreditamos que sua política de controle de acesso ao meio compartilhado e sua arquitetura distribuída podem ser adaptados com ajustes para tal finalidade.

Referências

Barbalace, A., Luchetta, A., Manduchi, G., Moro, M., Soppelsa, A., and Taliercio, C. (2008). "Performance Comparison of VxWorks, Linux, RTAI, and Xenomai in a Hard Real-Time Application". *IEEE Trans. on Nuclear Science*, 55(1):435–439.

Bovet, D. P. and Cesati, M. (2005). *Understanding the Linux Kernel (3rd ed.)*. O'Reilly.

Brito, A., Brasileiro, F., Leite, C. E., and Buriti, A. C. (2004). "Comunicação Ethernet em Tempo-real para uma Rede de Microcontroladores". In *XV Congresso Brasileiro de Automática*.

Carreiro, F. B., Fonseca, J. A., and Pedreiras, P. (2003). "Virtual Token-Passing Ethernet - VTPE". In *5th Int. Conf. on Fieldbus Systems and their Applications*.

Cristian, F. (1988). "Agreeing on Who Is Present and Who Is Absent in a Synchronous Distributed System". In *18th IEEE Int. Conf. on Fault-Tolerant Computing*.

Decotignie, J.-D. (2005). "Ethernet-Based Real-Time and Industrial Communications". *IEEE (Special Issue on Industrial Communication Systems)*, 93(6):1102–1117.

Dolejs, O., Smolik, P., and Hanzalek, Z. (2004). "On the Ethernet Use for Real-time Publish-subscribe Based Applications". In *5th IEEE Int. Workshop on Factory Communication Systems*, pages 39–44.

Gerum P. et al. (2009). "Xenomai". http://www.xenomai.org.

Hanssen, F. T. Y. and Jansen, P. G. (2003). "Real-Time Communication Protocols: an Overview". Technical Report TR-CTIT-03-49, University of Twente, The Netherlands.

Kiszka, J. (2005). "The Real-Time Driver Model and First Applications". In *7th Real-Time Linux Workshop*.

Kiszka, J., Wagner, B., Zhang, Y., and Broenink, J. (2005). "RTnet - A Flexible Hard Real-Time Networking Framework". In *10th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, pages 449–456.

Kohn, N., Varchmin, J.-U., Steiner, J., and Goltz, U. (2004). "Universal Communication Architecture For High-Dynamic Robot Systems Using QNX". In *8th Int. Control, Automation, Robotics and Vision Conf.*, pages 205–210.

Lian, F.-L., Moyne, J. R., and Tilbury, D. M. (2001). "Performance Evaluation of Control Networks: Ethernet, ControlNet and DeviceNet". *IEEE Control Systems Magazine*, 21(1):66–83.

Liu, J. W. S. (2000). *Real-Time Systems*. Prentice-Hall.

Pedreiras, P., Almeida, L., and Gai, P. (2002). "The FTT-Ethernet Protocol: Merging Flexibility, Timeliness and Efficiency". In *14th Euromicro Conf. on Real-Time Systems*, pages 134–142.

Regnier, P. and Lima, G. (2006). "Deterministic Integration of Hard and Soft Real-Time Communication over Shared-Ethernet". In *8th Workshop on Real Time Systems*.

Regnier, P., Lima, G., and Andrade, A. (2008a). "A TLA+ Formal Specification and Verification of a New Real-time Communication Protocol". In *Brazilian Symposium on Formal Methods*, pages 209–224.

Regnier, P., Lima, G., and Barreto, L. (2008b). "Evaluation of Interrupt Handling Timeliness in Real-time Linux Operating Systems". *ACM SIGOPS Operating Systems Review*, 42(6):52–63.

Venkatrami, C. and Chiueh, T. (1994). "Supporting Real-Time Traffic on Ethernet". In *15th IEEE Int. Real-Time Systems Symposium*, pages 282–286.