HARP: Um novo protocolo para alta disponibilidade implementado em FPGA

Rômerson Deiny Oliveira, Daniel Gomes Mesquita, Pedro Frosi Rosa

Faculdade de Computação – Universidade Federal de Uberlândia (UFU)
Caixa Postal 593 – 38.408-100 – Uberlândia – MG – Brazil

romerson@mestrado.ufu.br, {mesquita,frosi}@facom.ufu.br

Abstract. *The network's downtimes has brought financial losses to Internet users and companies. This paper aims to show the developing of a new high availability protocol and how its validation was done. The existing protocols have two harmful situations, named No-Brain and Split-Brain conditions, that are algorithmic problems that attack the network availability. This paper will show HARP protocol and how it fixes its predecessors.*

Resumo. *Os períodos de indisponibilidade das redes de computadores têm causado grandes prejuízos às empresas e aos usuários da Internet. Este trabalho visa mostrar o desenvolvimento de um novo protocolo de alta disponibilidade e como ele foi validado. Os protocolos existentes apresentam situações conhecidas como Acéfalo e Cérebro Bipartido, que são problemas algorítmicos que afetam a disponibilidade das redes. Este trabalho apresentará o protocolo HARP e como ele corrige seus predecessores.*

1. Introdução

A Internet tornou-se um dos principais veículos para transações pessoais e empresariais nos últimos anos. Segundo a *International Telecommunications Union*, o número de usuários de Internet cresceu 389% entre 2001 e 2011 [ITU 2012], em números absolutos o total passou de 495 milhões para 2 bilhões e 421 milhões de usuários conectados. Acompanhando esta eclosão, cresce também o número de usuários com necessidade de suporte a algum tipo de disponibilidade.

Períodos de indisponibilidade podem causar prejuízos de alto valor econômico. A *CA Technologies* divulgou uma pesquisa dizendo que o tempo de inatividade custa às empresas norte-americanas coletivamente $ 26,5 bilhões em receita por ano [Technologies 2010]. Assim sendo, no intuito de deixar a Internet disponível o maior tempo possível, já há alguns anos diversas empresas e universidades abriram espaço para pesquisa e desenvolvimento sobre alta disponibilidade de rede.

Os mecanismos de alta disponibilidade são caracterizados por usarem soluções baseadas em redundância de hardware, softwares inteligentes e protocolos para identificação de falhas de sistemas [Kriens et al. 2006]. Eles funcionam por meio de elementos que aparecem para a rede como um único elemento abstrato chamado de elemento virtual [Hinden 2004]. Os elementos físicos operam na filosofia mestre/escravo, tendo sempre um nó como mestre (não mais e não menos que um) e os outros disponíveis como escravos. O protocolo de comunicação deve ser capaz de avisar periodicamente a existência de

um mestre na rede e deve também ser capaz de eleger um novo mestre no caso de falha do atual.

Entre os protocolos de alta disponibilidade existentes, o *Virtual Router Redundacy Protocol* (VRRP) [Hinden 2004] destaca-se por ser o padrão *de facto* para os equipamentos de alta disponibilidade. Em [Lopes Filho 2008], o autor mostra que situações de indisponibilidade foram percebidas em redes que operavam com o protocolo VRRP e [Hashimoto et al. 2010] aponta as condições de "acéfalo" e "cérebro partido" (seção 2) como as causas que afetavam diretamente seu funcionamento.

[Hashimoto et al. 2010] apresentou uma proposta de extensão do VRRP para contornar as condições de acéfalo e cérebro bipartido, modelada em Redes de Petri. A fim de implementar a proposta, devido ao seu nível de abstração elevado foi necessário refiná-la e definir os cinco elementos do protocolo [Holzmann 1991] para permitir a sua implementação. Neste cenário, este trabalho apresenta o protocolo *High Availability Router Protocol* (HARP), define seus elementos e mostra como ele corrige os seus predecessores. O processo de desenvolvimento do HARP e sua implementação em hardware reconfigurável estão descritas na seção 3.

O HARP é parte de um projeto maior que pesquisa sobre Internet do Futuro, liderado pelo grupo MEHAR [MEHAR 2012] e trata dos aspectos de requisitos de alta disponibilidade do projeto EDOBRA, que por sua vez visa ampliar a cobertura física da instalação experimental OFELIA no Brasil [EDOBRA 2012]. A versão do HARP tratada neste trabalho é a versão 1, desenvolvida para a arquitetura atual da internet. As versões compatíveis com IPv6 e abordagem *Clean Slate* para internet não são tratadas, mas são passíveis de desenvolvimento e prototipação, tendo em vista a plataforma reconfigurável adotada para implementar o HARP.

O artigo está organizado da seguinte forma: a Seção 2 trata do estado da arte e trabalhos relacionadas. Já a Seção 3 mostra o método de desenvolvimento do protocolo HARP e a sequência de passos para sua validação. A Seção 4 apresenta os cinco elementos do protocolo e a Seção 5 traz as considerações finais.

2. Fundamentação Teórica e Estado da Arte

Esta seção aborda fundamentos dos mecanismos de alta disponibilidade e as situações que afetam os protocolos. Em seguida, é feita uma revisão da literatura correlata a fim de situar como os trabalhos se relacionam a esta pesquisa.

2.1. Alta Disponibilidade

Alta disponibilidade refere-se capacidade de uma rede manter-se disponível próximo de 100% do tempo, evitando a perda de serviços por meio da redução ou gestão de falhas e minimizando o tempo de inatividade planejado para o sistema. É obtida através de um endereço virtual que é compartilhado entre dois ou mais equipamentos. Este endereço é definido como *gateway* padrão da rede para os nós internos. Um roteador virtual, por exemplo, é a abstração formada por um ou mais roteadores executando um protocolo de alta disponibilidade. Na ocasião de uma falha no equipamento principal (mestre), outro componente do grupo de alta disponibilidade (escravo) assume a tarefa de roteamento, utilizando o endereço IP virtual, correspondente a um MAC virtual. Desta forma, a fa-

lha fica imperceptível aos clientes locais, já que a comunicação permanece ininterrupta [Sonderegger et al. 2009].

A alta disponibilidade é um subconjunto de tolerância a falhas [Johnson 1988], dado que esta trata desde a redundância dos componentes até o gerenciamento da comunicação com um protocolo. Do ponto de vista do protocolo, são duas as principais causas que levam uma rede à indisponibilidade:

1. Condição de acéfalo é a condição em que o nó de uma infraestrutura de alta disponibilidade, correntemente no papel de mestre, se torna inoperante e nenhum outro nó (nós escravos) se habilita a assumir tal papel [Pereira Júnior 2010].
2. Condição de cérebro bipartido é a condição em que um ou mais nós, correntemente no papel de escravos, de uma infraestrutura de alta disponibilidade se alçam ao papel de nó mestre por julgarem que o atual nó mestre tornou-se inoperante [Pereira Júnior 2010].

Estas condições podem ser causadas por falha de interface ou por ataque de terceiros. Este trabalho trata da primeira causa.

2.2. Trabalhos Relacionados

A literatura apresenta várias publicações que se pode relacionar a esta. De um lado, tem-se o tangente a implementações em FPGA aplicadas a redes de computadores e aplicadas a tolerância a falhas no nível de hardware. De um outro, tem-se as pesquisas que envolvem os protocolos em níveis mais altos de abstração.

Em [Jiang and Prasanna 2012], são realizados testes com switches OpenFlow para gerenciamento de fluxo e encaminhamento de pacotes. Trata-se de um trabalho que explora o paralelismo abundante do FPGA [Brown and Rose 1996] para tratar a nova geração da classificação de pacotes e propõe melhorias neste quesito, no intuito de tornar o processo de classificação e encaminhamento de pacotes mais escalável e com menores perdas. Esta caracterísitica é esperada para as próximas versões do HARP.

Em [Casado et al. 2009], o autor mostra que a utilização de hardware especializado para o encaminhamento de pacotes é uma técnica eficiente. Ainda em [Casado et al. 2009], introduz-se a ideia do uso de processadores de rede mais flexíveis como uma forma de contornar a necessidade de refazer chips devido a mudanças nos protocolos ou por adicionar novas características a este hardware, já que o custo é uma fator limitante.

O trabalho desenvolvido por Straka et al [Straka and Kotasek 2009] apresenta uma metodologia de construção de sistemas tolerante a falhas baseada em FPGA. As arquiteturas baseiam-se tanto no sistema duplex como na técnica de redundância modular tripla para melhorar a detecção de falhas. Para este propósito, o uso de verificadores *on-line* é demonstrado. Também é mostrado como os parâmetros disponibilidade (por exemplo tempo médio entre falhas e taxa de recuperação) podem ser afetados pelo ambiente de funcionamento em que o sistema tolerante a falhas é implementado. O trabalho é focado em técnicas de replicação de hardware. O trabalho [Straka and Kotasek 2009] apresenta técnina para construção de elementos de hardware redundantes, enquanto o trabalho mostrado neste artigo cuida da comunicação entre os elementos.

O trabalho apresentado em [Lopes Filho 2008] investiga a suspeita de que o principal problema dos protocolos de alta disponibilidade seria a camada de transporte, devido ao uso de protocolos não orientados a conexão, o que levou à proposta de uma camada de transporte baseada no protocolo SCTP. No entanto, ele conclui que o problema não residia na camada de transporte e a hipótese passou a pairar sobre a camada de enlace, devido à transmissão de mensagens com falso positivo para as camadas superiores.

[Hashimoto 2009] atribui os erros não detectados pelos algoritmos de controle de erro da camada de enlace como causa para o problema da condição de cérebro bipartido e conclui pela necessidade de desenvolvimento de um novo protocolo de alta disponibilidade, ou a extensão de um já existente. Foi demonstrado que o autômato do VRRP é incompleto, pois ele não considera erros não detectáveis pela camada de enlace. O autor afirma que o problema encontrado no VRRP também se aplica aos protocolos CARP e HSRP. O trabalho apresenta uma modelagem em redes de Petri que especifica as perdas de mensagens de anúncio em função dos fenômenos da camada de Enlace.

[Pereira Júnior 2010] discute as condições concorrentes para as situações de acéfalo e cérebro bipartido e define uma especificação de serviço que compõe o projeto de um protocolo de alta disponibilidade. O trabalho apresenta suposições de um ambiente onde um serviço de alta disponibilidade deve operar e como os protocolos de alta disponibilidade podem resolver os desafios que surgem nesses ambientes.

Os trabalhos publicadas em [Lopes Filho 2008], [Hashimoto 2009] e [Pereira Júnior 2010] descrevem uma sequência de hipóteses e conclusões sobre os problemas que atacam os protocolos de alta disponinilidade. Os autores concluíram que os problemas residem nas condições de acéfalo e cérebro bipartido percebidas no protocolo VRRP. Este trabalho situa-se como a continuação das pesquisas desenvolvidas nos trabalhos citados.

3. Método de desenvolvimento

O processo de desenvolvimento do *High Availability Router Protocol* estabeleceu-se divido em três etapas. Partiu-se da caracterização e proposição de uma extensão do VRRP. Em seguida, houve o processo de especificação e desenvolvimento do HARP. Por fim, é mostrada a proposta de um sistema de avaliação para protocolos de alta disponibilidade. Os itens nesta seção explicam este processo evolutivo.

3.1. Caracterização do Problema

Situações de indisponibilidade foram percebidas no tráfego mantido por arquiteturas que operavam com o VRRP. Ao observar a situação, o fenômeno foi reproduzido e foram realizadas coletas e análises de tráfego de *Firewall* e *Intrusion Prevention Systems* operando em alta disponibilidade em ambientes de teste, devidamente estruturados no *Data Center* de uma empresa de telecomunicações [Lopes Filho 2008], [Hashimoto 2009]. As situações de acéfalo e cérebro bipartido ocorriam com a segunda sendo mais frequente. Nos dois casos as requisições de serviços não eram respondidas ou eram respondidas por mais de um equipamento, o que gerava uma sobrecarga impossível de ser gerenciada [Hashimoto et al. 2010].

O VRRP foi inicialmente lançado pela Cisco [Cisco 2012] e logo tornou-se padrão *de facto* para os equipamentos de alta disponibilidade. Em se tratando de alta

disponibilidade, há também a implementação *Common Address Redundancy Protocol* (CARP) [OpenBSD 2012], desenvolvida pela comunidade OpenBSD, e ainda os protocolos *Hot Standby Router Protocol* (HSRP) [Li et al. 1998] e *NetScreen Redundancy Protocol* (NSRP) [Kriens et al. 2006]. Segundo [Hashimoto 2009] o protocolo VRRP é mais utilizado que os outros e os problemas percebidos nele podem ser estendidos a todos os outros citados, em virtude de análise realizada sobre a máquina de estados de cada um deles.

Após estabelecer as causas da indisponibilidade e apontá-las como sendo problemas algorítmicos que atacam o protocolo, [Hashimoto et al. 2010] apresentou uma especificação abstrata, modelada em Redes de Petri e representada por um autômato para contornar as condições citadas. O autômato em [Hashimoto et al. 2010] não foi implementado e trata-se de uma descrição na mesma filosofia da especificação em Rede de Petri: a visão do comportamento do grupo de alta disponibilidade (elemento virtual).

Entretanto escapou-lhe a especificação do comportamento individual de cada um dos elementos físicos componentes do elemento virtual e, neste caso, fez-se necessária a especificação do autômato e dos outros quatro elementos de cada instância do protocolo [Holzmann 1991]. Foi necessário refinar a proposta de [Hashimoto et al. 2010] a fim de implementá-la e validar o funcionamento da extensão do VRRP.

Este trabalho apresenta as regras de comportamento para uma instância do HARP em um elemento físico, do ponto de vista da sua máquina de estados, o que não havia na proposta de [Hashimoto et al. 2010]. Elementos como parâmetros e condições para o processo de eleição a novo mestre, condições de entrada e saídas para transições entre estados de uma instância, especificação de serviço e análises dos casos de perdas de primitivas ainda eram necessários para uma implementação do protocolo. Esse processo de refinamento originou o HARP.

3.2. Desenvolvimento do *High Availability Router Protocol*

Nesta fase ocorreu efetivamente a especificação dos elementos do HARP, bem como a sua implementação em hardware. Os cinco elementos do HARP são apresentados detalhadamente na seção 4, enquanto as características de implementação são aqui tratadas.

A implementação em hardware permite a detecção de detalhes e correção de falhas que escapam ao projeto de protocolos em níveis elevados de abstração. No entanto, o custo de produção de um hardware específico é muito alto. Neste contexto, o uso de um dispositivo flexível faz-se necessário, sendo o FPGA o mais indicado para esta situação. O uso do FPGA permitiu que correções fossem feitas após cada prototipação do HARP, permitindo perceber erros do hardware e retornar à sua especificação para realizar as correções. Este processo de iterações pertimiu especificar o HARP e prototipar um hardware específico até que se chegasse à versão final. O uso de FPGA para implementar algoritmos de redes tem sido amplamente realizado por empresas e na academia, em virtude da sua adaptabilidade e flexibilidade, reduzindo o custo de produção e o *time-to-market* do projeto se comparados a um ASIC.

Uma plataforma foi montada para realizar a prova de conceito do HARP. Para tal, montou-se um esquema com três placas de prototipação DE2 simulando elementos de rede com conexões dedicadas entre si e em topologia estrela. Cada elemento de rede possui um FPGA Cyclone 2 [Altera 2006]. Os FPGAs foram conectados conforme a

figura 1. Cada FPGA recebe e envia dados a qualquer um dos outros através de conexão dedicada sobre um cabo *flat* de 36 vias conectado a um cabeçalho de expansão contido na placa de prototipagem. Cada solicitação de serviço é feita ao se pressionar um dos quatro botões de acionamento conectados ao FPGA. O meio comunicaçao foi divido em três canais (Figura 1) com doze vias cada um.

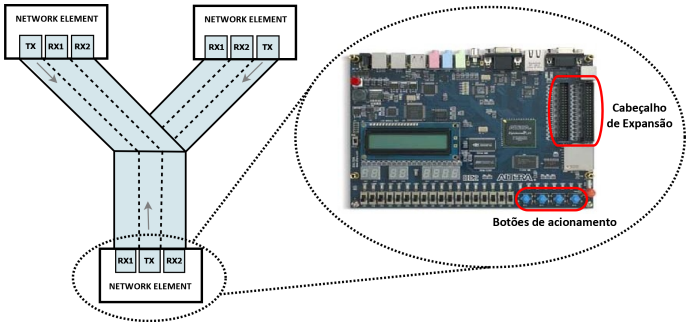

Figura 1. Esquema de conexão para a prova de conceito

Todos os serviços foram testados e, após uma sequência de reconfigurações do hardware, funcionaram corretamente. Dado que o HARP é baseado em *timeouts* (seção 4.4), como é o caso dos estados S2, S3, S4 e S6 da Figura 4, inserir atrasos entre estados e atender todos os possíveis casos de perdas de primitivas foram os principais benefícios da reconfigurabilidade do FPGA, que permitia revisar o HARP em cada implementação.

O HARP foi testado de forma que pudesse prever a perda de cada uma das primitivas reconhecidas no seu vocabulário, assim foi possível certificar de que o HARP conseguiria lidar com todas as situações de perdas. Esta análise foi feita paralelamente ao processo de implementação. Exemplos de situações de perdas são dadas na subseção 4.4.

Como dito anteriormente, a Figura 1 representa a prova de conceito dos elementos, que foi concluida com êxito. Entretanto, não trata do tempo de recuperação do sistema e nem traz dados sobre o tempo que cada primitiva leva para ser processada dentro de cada instância do HARP, isso será feito quando implementada a etapa de validação. A etapa de validação é independente da especificação do HARP e objetiva desenvolver um sistema capaz de testar qualquer protocolo de alta disponibilidade.

3.3. Sistema de Validação

A Figura 2 mostra o sistema de validação de protocolos de alta disponibilidade que desenvolvemos. A figura mostra um elemento virtual formada por três (ou mais) elementos de rede (FPGA) conectados entre si por uma via ethernet compartilhada. Esta mesma via está conectada a computadores pessoais responsáveis por gerar fluxo de informação.

O sistema é para testar protocolos de alta disponibilidade do ponto de vista das falhas de canal e interface de comunicação. Este é o principal caso gerador das condições de acéfalo e cérebro bipartido. Outro caso gerador é o ataque externo por invasores, o que não é tratado neste trabalho.

Cada um dos elementos de rede da Figura 2 é configurado na forma de um *System on Chip* (SoC) baseado no processador embarcado NIOS 2 [Altera 2012]. Um dos componentes do SoC é o hardware do protocolo de alta disponibilidade a ser testado. O

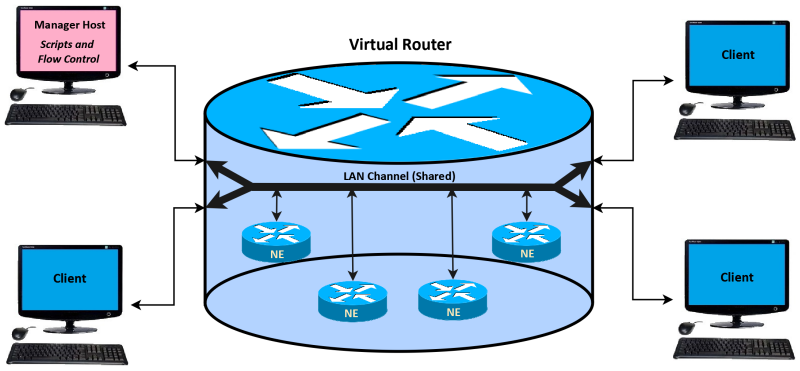

Figura 2. Sistema proposto para validação

sistema interage com uma aplicação em software que transmite dados via Ethernet ao FPGA através de um controlador operando na camada de enlace. Assim, o computador responsável por gerar o fluxo enviará mensagens aos outros computadores e o SoC as encaminhará.

Pode-se submeter qualquer protocolo de alta disponibilidade à validação, com implementação em hardware ou em software. Para implementações em hardware, inclui-se o protocolo como um módulo do SoC. Caso seja em software, a aplicação utliza o SoC apenas como plataforma. Uma camada de abstração de hardware foi criada para contornar os problemas de incompatibilidade do tamanho de primitivas na interface HW/SW entre diferentes protocolos.

4. HARP - Descrição dos Elementos

Os cinco elementos do protocolo são apresentados nesta seção, a saber: suposições sobre o ambiente, serviços, vocabulário, formatação e regras de procedimento. Por questões de espaço, os serviços e vocabulário serão apresentados em conjunto.

4.1. Suposições sobre ambiente

O HARP, como protocolo de alta disponibilidade, deve operar em elementos individuais de um grupo de equipamentos redundantes (Figura 2). Este grupo forma o elemento virtual e é visto pelo restante da rede como um único ponto de encaminhamento de pacotes. O HARP está projetado para trabalhar na camada três da arquitetura internet, juntamente com o protocolo IP.

4.2. Serviços e Vocabulário

Nesta subseção, tem-se a apresentação de todos os serviços oferecidos pelo HARP, de forma a contornar as condições de acéfalo e cérebro bipartido. As mensagens também são aqui introduzidas e é importante dizer que elas obedecem à taxonomia *Request/Indication* e *Response/Confirmation*. A Tabela 1 sumariza o conjunto dos serviços e mostra quais as mensagens são trocadas durante a execução de cada um deles. A seção 4.4 retoma esta discussão e detalha a execução de cada serviço, bem como a função de cada mensagem e seu devido uso.

Há ainda duas mensagens não citadas na Tabela 1, já que não são diretamente partes dos serviços, mas são apresentadas a seguir.

Tabela 1. Serviços do HARP

Nome	Mensagens	Descrição
Keep Alive (KA)	*KA Request (KA_REQ)*	Serviço não confirmado. Atua como *heartbeat* usado pelo mestre para avisar sua atividade. Monitorando estes *heartbeats*, os escravos do grupo de alta disponibilidade determinam quando um elemento mestre parou de funcionar.
Given Master (GM)	*GM Request (GM_REQ)* *GM Response (GM_RESP)* *GM Failed Request (GMFAIL_REQ)* *GM Ready Request (GMRDY_REQ)*	Serviço confirmado. Utilizado pelo mestre para indicar sua intenção de transferir o seu papel a um escravo específico. O endereço do escravo tem que constar na tabela de ativos do mestre.
Informe Node (INF)	*INF Request (INF_REQ)* *INF Response (INF_RESP)*	Serviço confirmado. Utilizado para indicar que um nó está se tornando membro do grupo de alta disponibilidade.
Remove Node (REM)	*REM Request (REM_REQ)* *REM Response (REM_RESP)*	Serviço confirmado. Utilizado por um escravo para indicar ao mestre do grupo de alta disponibilidade que ele está deixando a rede.
Check Brain (CB)	*CB Request (CB_REQ)* *CB Response Positive (CB_RESP+)* *CB Response Negative (CB_RESP-)*	Serviço confirmado. Utilizado por um nó escravo para se certificar de que não há mestre no grupo e evitar a situação do cérebro bipartido no processo de obtenção do novo mestre.

- *Active Slave Request* (ACTS_REQ): enviada pelo mestre, em *broadcast*, para atualizar a sua lista de escravos ativos;
- *Active Slave Response* (ACTS_REQ): enviada pelo escravo, ao mestre, para responder sua atividade.

4.3. Formato das Mensagens

Para atender às necessidades dos serviços apontados anteriormente e por operar na camada três da arquitetura Internet, a mensagem do HARP tem dez campos predefinidos para sua primeira versão, a operar com IPv4. As mensagens tem 128 bits de cabeçalho e não utilizam o campo de dados, que pode ser utilizado nas próximas versões e adaptações para Internet do futuro.

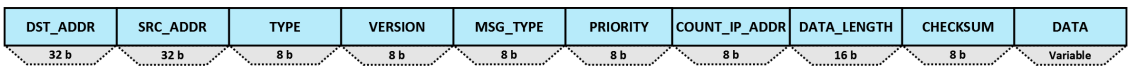

Figura 3. Formato da mensagem HARP

As mensagens HARP são essencialmente mensagens de controle, tendo por isso o cabeçalho maior que o campo de dados, em todos os casos. O formato das mensagens é ilustrado pela figura 3, que contém o identificador do campo e o seu comprimento em bits. O significado de cada campo pode ser conferido segundo a lista a seguir:

1. DEST_ADDR: IP de destino. Pode ser de um elemento ou *broadcast*;

2. SRC_ADDR: IP de origem. Especifica o elemento que enviou a mensagem;
3. TYPE: tipo do protocolo, i.e., número reservado à indentificação do HARP;
4. VERSION: versão do protocolo, para este caso será usada somente a vesão 1;
5. MSG_TYPE: indica qual o tipo de mensagem reconhecida pelo vocabulário do HARP que aquela primitiva está carregando;
6. PRIORITY: prioridade do nó emissor, atribuída pelo administrador de rede;
7. COUNT_IP_ADDR: em uma mensagem KA_REQ, indica o número de escravos ativos no grupo. Pode também indicar quantos endereços IP estão sendo carregados no campo DATA (em outras versões);
8. DATA_LENGTH: comprimento do campo de dados;
9. CHEKSUM: soma de verificação para detecção de erro;
10. DATA: campo de dados para parâmetros das mensagens, quando necessário.

Os endereços 0x0000 e 0xFFFF são reservados. O primeiro indica que não há endereço sendo transmitido, enquanto o segundo destina-se ao *broadast*. As próximas versões do HARP podem vir com mensagens de formatos diferentes para atender diferentes propostas de arquiteturas de redes.

4.4. Regras de Procedimentos

Cabe às regras de procedimentos explicar a dinâmica de troca de mensagens e o comportamento da máquina de estados (FSM) do HARP durante a execução dos procedimentos dos serviços. As condições de acéfalo e cérebro bipartido são evitadas ou contornadas segundo trocas de mensagens explicadas nesta seção. Para a condição de acéfalo, basicamente ela ocorrerá se o mestre atual sair de operação e é resolvida pelo processo Check Brain. Já a condição de cérebro bipartido pode ocorrer por diferendes situações de cadenciamento, que são exemplificadas nos itens subsequentes.

A Figura 4 representa a FSM do HARP e é a união de todos os autômatos parciais (por serviço), tanto para elementos emissores quanto para receptores. O restante desta seção explica as transições da FSM, ralacionando-as a cada serviço. O autômato é a descrição formal das transições explicadas e deve ser observado em cada item a seguir.

Keep Alive (KA): O HARP se inicia no estado *Idle* e baseado na prioridade cada nó irá de *Idle* a *Master* ou a *Slave*, tornando-se mestre ou escravo, respectivamente. Mestre tem prioridade zero, valores entre 1 e 255 são para escravos. Se um nó assume o papel de mestre ele inicia o serviço KA, enviando um KA_REQ a cada intervalo de tempo *t* para informar seu estado aos demais. O mestre utiliza o campo COUNT_IP_ADDR para informar o número de escravos ativos no grupo.

Inform Node (INF): Quando um nó escravo entra na rede, ele envia um INF_REQ ao grupo e aguarda um INF_CONF do mestre. O mestre deve incluí-lo em sua tabela de ativos. Um mestre não precisa enviar INF_REQ, a menos que ele transite ao estado de *Slave*.

Remove Node (REM): Um nó escravo envia ao mestre um REM_REQ para solicitar sua saída da rede. O mestre envia um REM_RESP ao solicitante autorizando a saídae a tualiza sua tabela. Recebendo confirmação, o nó pode deixar a rede.

Given Master (GM): Se houver necessidade de transferir a função de mestre, o processo GM é iniciado. Isso acontece quando é preciso realizar manutenção programada

de equipamentos, por exemplo. Para formalizar, o início do processo ocorre quando a *flag* GM_Start é setada no mestre, o que desencadeia o fluxo de mensagens do GM.

O mestre envia um GM_REQ a um escravo predeterminado e vai para o estado *Wait For Given Master Confirm* (WF_GM CONFIRM). Se ele recebe um GM_CONF, ele vai para o estado *Slave* e envia um GMRDY_REQ informando a conclusão do processo. Caso contrário, se ocorre o TO_1 [1] (intervalo *t*), ele envia o um GMFAIL_REQ informando erro no processo e volta ao estado *Master*.

Do lado do receptor, quando o escravo recebe o GM_IND, ele envia um GM_RESP e vai ao estado *Given Master Accepting* (GM ACCEPTING), esperando um intervalo para garantir que não há outro elemento de rede enviando mensagens *Keep Alive*. Se um GMRDY_IND é recebido, acontece a transição ao estado *Master*, mas se ocorre o TO_2, um KA_IND ou GMFAIL_IND, ele retorna ao estado *Slave*, evitando a condição de cérebro bipartido.

Dentro do contexto do GM, algumas situações são ilustradas para demonstrar a capacidade de recuperação do protocolo no caso de perdas de mensagens:

- Se a primitiva GM_REQ for perdida, então o escravo receptor não recebe o GM_IND e tampouco envia o GM_RESP, logo o mestre emissor deve voltar ao seu estado de origem antes que os outros escravos percebam a sua falta. Por isso, o TO_1 é menor que o TO_3 (*timeout* de não recebimento do KA_IND) para garantir que não haverá condição de acéfalo na rede.
- Já no caso de perda do GM_RESP, não haverá o envio da primitiva GMRDY_REQ, não permitindo que o receptor vá ao estado *Master*. O receptor, que estava no estado GM ACCEPTING, consequentemente voltará ao estado *Slave* pela ocorrência do TO_2[2], enquanto esperava o GMRDY_IND. Ele também deverá voltar ao estado *Slave* ao receber o GMFAIL_IND ou ao receber um KA_IND.
- Caso haja perda da primitiva GMFAIL_REQ não haverá problema, pois o TO_2 e o KA_IND contornam esta perda no estado *GM ACCEPTING*.
- Caso a primitiva GMRDY_REQ se perca no caminho, há a situação em que o mestre vai a *Slave* e o escravo continua em *Slave*, gerando a condição de acéfalo. Este é o pior caso quando se trata de perda de mensagens no processo GM, mas caso isso ocorra, o serviço *Check Brain* é iniciado automaticamente por algum escravo e um novo mestre será eleito.

Check Brain (CB): Se o escravo detecta a falta do mestre, o processo para eleger o novo mestre é iniciado. Este é um ponto crucial da especificação, pois garante a não ocorrência da condição de cérebro bipartido e resolve diretamente a condição de acéfalo. A *Check Brain Flag* (CB_Flag) é usada para inibir que o processo CB seja iniciado por algum escravo quando já tiver sido iniciado e não acabado por outro.

Um nó escravo percebe que há um intervalo de tempo TO_3[3] sem o recebimento

[1] TO_n representa *timeout* e é utilizado nos estados com tempo de permanência máximo pré-estabelecido

[2] O TO_2 e o TO_1 sao menores que o TO_3 e ambos valem *t*

[3] O TO_3 é o tempo necessário para entender que há falha do mestre e baseia-se no não recebimento de KA_IND. $TO_3 = (2 + a) \star t$, onde *a* é uma constante baseada na prioridade. Isso duminui a probabilidade de a falha do mestre ser percebida no mesmo momento por vários escravos. O TO_3 é maior que todos os outros TO's da FSM, garatindo que o Check Brain não seja iniciado durante a execução de um outro serviço.

do KA_IND e percebe ainda que a *flag* CB_flag está resetada, certificando-se de que o processo CB ainda não foi iniciado. O nó envia então um CB_REQ em *broadcast* para os outros escravos e vai ao estado *Wait For Check Brain Confirm* (WF_CB CONFIRM), para aguardar pela confirmação de outros escravos sobre a não existência do mestre na rede.

Caso o escravo *sender* receba uma confirmação positiva CB_CONF(+) ele volta ao estado *Slave*, resetando o contador do TO_3, que é naturalmente resetado a cada recebimento do KA_IND. O retorno ao estado *Slave* pode ocorrer também se houver o recebimento de um KA_IND (indicando que há mestre) ou caso nenhuma mensagem chegue e ocorra o TO_4[4].

Chegando um CB_CONF(-), entende-se que outro elemento escravo também não está recebendo mensagens do mestre. O emissor vai então ao estado *Master Election* para definitivamente certificar-se de que não há mestre na rede. Ele espera receber um total *b* de mensagens CB_CONF(-) proporcional ao número total de escravos no grupo. O valor *b* é dado pelo teto de $b = \frac{totaldeescravos}{2}$.

Se ele recebe as confirmações negativas esperadas, ele vai ao estado *Master* e envia KA_REQ aos outros escravos da rede. Ele deve então zerar o CB_flag e o contador *k* de CB_CONF (que espera atingir o valor *b*). Ao receber um KA_IND os nós receptores zeram todos os seus contadores e a CB_flag, continuando escravos e reconhecendo o novo mestre.

Do lado dos receptores, quando um CB_IND é recebido, ele armazena o endereço de origem e seta sua CB_flag, indicando que o processo CB foi iniciado e está em andamento. Então, ele vai para o estado *Search Master* para conferir quanto tempo decorreu desde o último KA_IND recebido. Caso haja no máximo *2t* ele envia resposta CB_RESP(+) e volta para o estado de *Slave*, pois para ele tudo está operando dentro da normalidade e certamente houve uma falha pontual do emissor.

Entretanto, se o receptor perceber no estado *Search Master* que não houve recebimento de KA_IND no último intervalo *2t*, ele envia um CB_RESP(-) atestando a falta do mestre e vai ao estado *Master Election* para atrasar sua volta e dar tempo para o emissor concluir o processo. Ele só deixa o estado *Master Election* para ir ao *Slave*, o que ocorre pelo recebimento de um KA_IND ou pelo TO_5, quando ele reseta seu CB_flag. O TO_5 é igual a *t* para emissor e *2t* para receptor.

Cabe destacar que apesar de o intervalo necessário para iniciar um CB seja TO_3, o intervalo *2t* já é suficiente para gerar respostas CB_RESP. Isso porque o escravo de menor TO_3 poderia perceber a falha e os outros ainda não terem atingido seus *timeouts* TO_3 (por causa da constante *a*) e deixariam o emissor sem resposta atualizada.

O uso de um TO_3 variável minimiza o risco de mais de um escravo iniciar o CB ao mesmo tempo. Ainda que isso ocorra, uma das mensagens atingirá o canal antes da outra, baseado no princípio de compartilhamento de canal (CSMA/CD). É a esta mensagem que os escravos responderão, pois eles ignorarão mensagens subsequentes por já terem setado a CB_flag. Este processo faz com que um dos escravos emissores volte ao estado *Slave*

[4]O TO_4 tem o valor de *t*. Quando ele ocorre o escravo volta ao seu estado inicial, resetando o TO_3 e a *flag* CB_flag, daí recomeça a esperar o KA_IND novamente e, depois disso, pode repetir todo o processo CB, caso necessário. Este procedimento garante que o escravo permanece escravo se, por exemplo, houver falha somente na interface dele, seja ela permanente ou temporária

quando ocorrer o TO_4 no estado WF_CB CONFIRM.

Para evitar mais uma vez a codição de cérebro bipartido, deve-se prever a situação em que o mestre fica indisponível um tempo suficientemente grande para a eleição de um novo mestre e depois disso volta a operar normalmente, deixando dois mestres no grupo. Para isso, ao receber um KA_IND os mestres envolvidos devem parar de enviar KA_REQ e irem ao estado de *Slave*. Isso evita o cérebro bipartido e, no máximo, gera a condição de acéfalo, que é resolvível com o processo *Check Brain*.

Ao se concluir um processo de *Given Master* ou *Check Brain*, o novo mestre envia um ACTS_REQ ao grupo para atualizar a sua tabela de ativos e, consequentemente, atualizar o grupo de quantos escravos estão inseridos. A tabela de ativos é sempre atualizada com o recebimento de um ACTS_CONF, INF_IND ou REM_IND.

Todos estes procedimentos descritos são executados por qualquer instância do protocolo e a qualquer tempo. Portanto, as regras são combinados e descritas formalmente em um único autômato capaz de gerenciar a comunicação.

4.5. Autômato Final

Combinando os autômatos dos serviços, é possível gerar uma FSM que expressa o comportamento de uma instância do HARP. A Figura 4 mostra todos os eventos e ações decorrentes disso. A descrição de todos os estados que compõem a FSM da figura já foram vistos na seção 4.4, bem como os eventos reconhecíveis para as transições entre eles. Esta FSM descreve o comportamento do HARP para qualquer um dos serviços oferecidos.

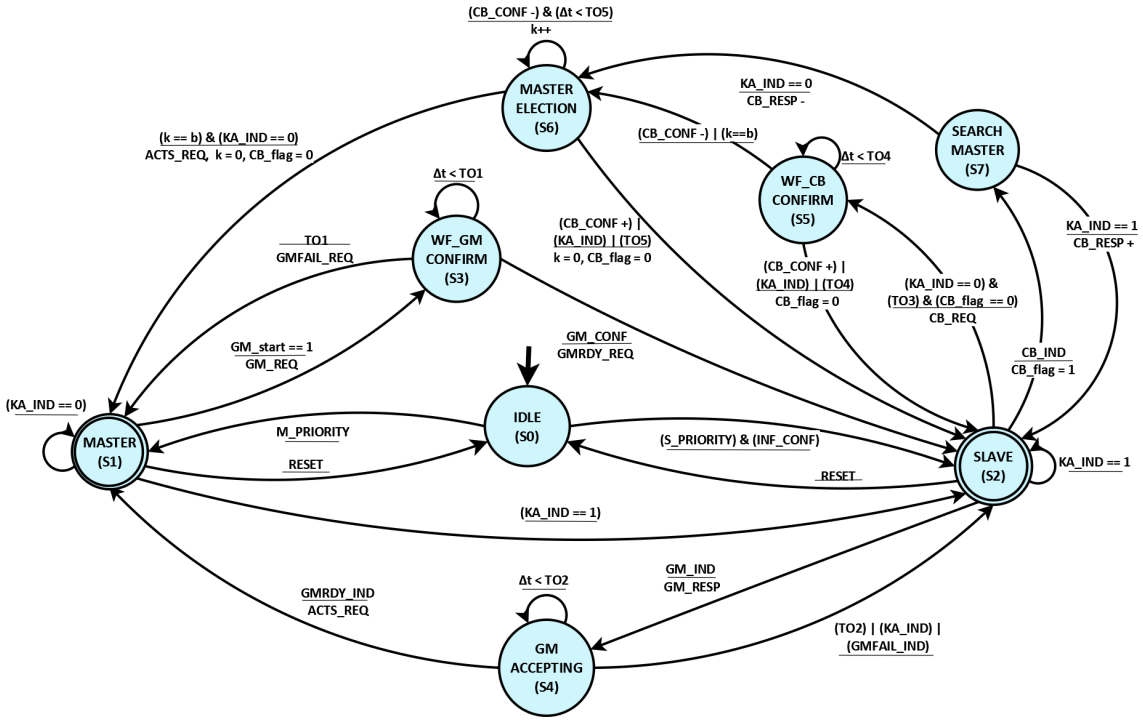

Figura 4. Autômato do HARP

Cabe salientar que o autômato apresentado na Figura 4 é bem formado e mantém as boas propriedades, a saber: é *k*-limitado, garantindo que a FSM é finita; não tem estados

sem sucessores, ou seja, livre de *deadlock*; é livre de *livelock* e há sempre uma sequência que leva ao estado inicial. Então, a modelagem foi feita no intuito de garantir a descrição completa dos cinco elementos e a demonstração das boas propriedades.

5. Considerações Finais

A principal contribuição deste artigo é apresentar o protocolo *High Availability Router Protocol* e como ele corrige os problemas algorítmicos que afetam os protocolos de alta disponibilidade existentes. Para a concepção do mesmo, partiu-se de uma proposta especificada em alto nível de abstração [Hashimoto et al. 2010], modelada em Rede de Petri e chegou-se a uma especificação completa e de mais baixo nível de abstração para o protocolo HARP, não deixando escapar questões necessárias à implemenção.

Os dois principais problemas que atacam a alta disponibilidade são as condições de acéfalo e cérebro bipartido. Estes problemas são resolvidos no HARP, considerando que o novo autômato provê estados e transições suficientes para cobrir estas possibilidades.

Um importante ponto mostrado ao longo do texto e principalmente ao se analisar o autômato do HARP é que as boas propriedades de uma FSM são mantidas: o HARP é k-limitado, livre de *livelock*, livre de *deadlock* e é reinicializável de qualquer estado. O autômato final tem oito estados e as transiçoes são controladas por um vocabulário bem definido.

A implementação em FPGA permitiu realizar atualizações com baixo custo, em virtude da sua reconfigurabilidade. Assim, foi possível realizar as mudanças necessárias após cada implementação até chegar a uma versão definitiva. Cabe lembrar que os parâmetros de tempo e desempenho não formam medidos nesta etapa, que foi destinada à demonstração de funcionamento do protocolo.

Este artigo apresenta resultados de um novo protocolo de rede. Tais resultados se concretizaram após um processo iterativo de proposições e testes em hardware reconfigurável. A partir de agora, podemos encaminhar o desenvolvimento de um módulo de hardware específico para tratar alta disponibilidade de rede e vislumbrar o engajamento deste módulo nas novas propostas de arquiteturas para Internet.

Referências

Altera (2006). De2 development and education board user manual. http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html.

Altera (2012). Nios 2 processor. http://www.altera.com/devices/processor/nios2/ni2-index.html.

Brown, S. and Rose, J. (1996). Fpga and cpld architectures: a tutorial. *Design Test of Computers, IEEE*, 13(2):42 –57.

Casado, M., Koponen, T., Moon, D., and Shenker, S. (2009). Rethinking packet forwarding hardware. In *ACM Workshops on Hot Topics in Networks*, volume VII, pages 1 –6.

Cisco, S. (2012). High availability. http://www.cisco.com.

EDOBRA (2012). Extending and deploying ofelia in brazil. http://www.mehar.facom.ufu.br/projects/ofelia-edobra.dot.

Hashimoto, G., Filho, E., Pereira, J., and Rosa, P. (2010). High avaliability: A long-term feature in network elements. In *Systems and Networks Communications (ICSNC), 2010 Fifth International Conference on*, pages 201 –206.

Hashimoto, G. T. (2009). Uma proposta de extensão para um protocolo para arquiteturas de alta disponibilidade. Dissertação de mestrado, Universidade Federal de Uberlândia.

Hinden, R. (2004). Virtual Router Redundancy Protocol (VRRP). RFC 3768 (Draft Standard).

Holzmann, G. J. (1991). *Design and validation of computer protocols*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

ITU (2012). Internet users. http://www.itu.int/ITU-D/ict/statistics/index.html.

Jiang, W. and Prasanna, V. (2012). Scalable packet classification on fpga. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(9):1668 –1680.

Johnson, B. W., editor (1988). *Design & analysis of fault tolerant digital systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Kriens, S., Cameron, R., Woodberg, B., and Madwachar, M. K. (2006). *Configuring Juniper Networks NetScreen & SSG Firewalls*. Syngress Publishing.

Li, T., Cole, B., Morton, P., and Li, D. (1998). Cisco Hot Standby Router Protocol (HSRP). RFC 2281 (Draft Standard).

Lopes Filho, E. (2008). Arquitetura de alta disponibilidade para firewall e ips baseada em sctp. Dissertação de mestrado, Universidade Federal de Uberlândia.

MEHAR (2012). Mondial entities horizontally addressed by requirements. http://www.mehar.facom.ufu.br/.

OpenBSD (2012). Pf: Firewall redundancy with carp and pfsync. http://www.openbsd.org/faq/pf/carp.html.

Pereira Júnior, J. E. (2010). Especificação de serviço e suposiçoes sobre o ambiente para um protcolo de alta disponibilidade. Dissertação de mestrado, Universidade Federal de Uberlândia.

Sonderegger, J., Blomberg, O., Milne, K., and Palislamovic, S. (2009). *Junos High Availability: Best Practices for High Network Uptime*. O'Reilly Media, Inc., 1st edition.

Straka, M. and Kotasek, Z. (2009). High availability fault tolerant architectures implemented into fpgas. In *Digital System Design, Architectures, Methods and Tools, 2009. DSD '09. 12th Euromicro Conference on*, pages 108 –115.

Technologies, C. (2010). North american businesses lose $26.5 billion annually from avoidable downtime, according to new ca technologies study. http://www.ca.com/us/news/Press-Releases/na/2010/North-American-Businesses-Lose-26-5-Billion- Annually.aspx.