

GILBERTO TADAYOSHI HASHIMOTO

**UMA PROPOSTA DE EXTENSÃO PARA UM
PROTOCOLO PARA ARQUITETURAS DE ALTA
DISPONIBILIDADE**

Dissertação apresentada ao programa de pós-graduação em ciência da computação da Universidade Federal de Uberlândia, para obtenção do grau de mestre em ciência da computação.

Área de concentração: redes de computadores.

Orientador: Prof. Dr. Pedro Frosi Rosa, da Universidade Federal de Uberlândia.

UBERLÂNDIA – MG

2009

Gilberto Tadayoshi Hashimoto

**UMA PROPOSTA DE EXTENSÃO PARA UM PROTOCOLO PARA ARQUITETURAS
DE ALTA DISPONIBILIDADE**

Dissertação apresentada ao programa de pós-graduação em ciência da computação da Universidade Federal de Uberlândia, para obtenção do grau de mestre em ciência da computação.

Área de concentração: redes de computadores.

Banca examinadora:

Uberlândia, 02 de setembro de 2009.

Prof. Dr. Pedro Frosi Rosa – Orientador – UFU

Prof. Dr. Sergio Takeo Kofuji – USP

Prof. Dr. Renan Gonçalves Cattelan – UFU

AGRADECIMENTOS

À Universidade Federal de Uberlândia, em especial, à Faculdade de Computação, pela oportunidade de realizar este curso.

À CTBC, empresa na qual trabalho, pela permissão que me foi concedida para que eu pudesse me dedicar a todas as atividades do curso.

Ao meu orientador, Prof. Dr. Pedro Frosi Rosa, pela constante orientação, pelo incentivo durante a elaboração desta dissertação e pela confiança e apoio sempre prestados.

Aos demais professores do curso, que também contribuíram para minha formação.

À minha esposa Fabíola e filhos: Bárbara e Bernardo, que sempre compreenderam a importância deste curso incentivando e apoiando-me nos momentos difíceis.

À minha mãe Yoshie que sempre me apoiou em todos os momentos e decisões da minha vida e que infelizmente partiu deste mundo antes de poder ver a finalização deste trabalho.

A todas as outras pessoas que, direta ou indiretamente, contribuíram para a realização deste sonho.

"O valor das coisas não está no tempo em que elas duram, mas na intensidade com que acontecem. Por isso existem momentos inesquecíveis, coisas inexplicáveis e pessoas incomparáveis."

(Autor desconhecido)

SUMÁRIO

RESUMO	i
ABSTRACT	ii
LISTA DE FIGURAS.....	iii
LISTA DE TABELAS.....	iv
LISTA DE ABREVIATURAS.....	v
1. Introdução	1
2. Estado da Arte para Alta Disponibilidade	3
2.1. Mecanismos de Alta Disponibilidade	4
2.1.1. Redundância de Hardware	5
2.1.2. Sistemas de Software inteligentes	6
2.1.3. Protocolos para Identificação de Falha	6
2.2. Aspectos Importantes do Protocolo SCTP.....	7
2.2.1. O Cabeçalho Comum do Protocolo SCTP	9
2.2.2. Mecanismo de <i>Multistreaming</i>	15
2.2.3. Mecanismo de <i>Multihoming</i>	16
2.2.4. <i>Multihoming</i> e HA	17
2.3. A Arquitetura de HA Baseada no Protocolo SCTP	18
2.4. Fluxo de Mensagens <i>Heartbeat</i> da Arquitetura.....	19
2.5. Visão Geral da Proposta de HA	24
2.5.1. Camada de Serviços HA (HSOL)	24
2.6. Conclusão	26
3. Definição do problema	28
3.1. Arquitetura Keepalived	28
3.2. Protocolo VRRP	30
3.3. Definição do problema de <i>Split Brain</i> e <i>No-Brain</i>	33
3.4. Apresentação dos problemas.....	34
4. Proposta de Extensão ao Protocolo VRRP	36
4.1. Formalização das Bases da Extensão	36
4.2. A arquitetura proposta.....	40
4.2.1. Autômato VRRP	42

4.2.2.	Renomeação de serviços existentes no VRRP.....	44
4.3.	Descrição Formal em Rede de Petri	49
4.4.	Descrição do Funcionamento da Rede de Petri	53
4.5.	Árvore de Cobertura	55
4.6.	Autômato do HA proposto.....	57
5.	Conclusão	62
6.	Referências Bibliográficas	65

RESUMO

Com o crescimento mundial no número de acesso banda larga e a ampla utilização de dispositivos móveis como telefones celulares, PDAs e outros tipos de acesso conectados à Internet, fica cada vez mais evidente que existam usuários com necessidade de suporte a algum nível de disponibilidade. Essa necessidade fez com que as arquiteturas de alta disponibilidade fossem utilizadas em larga escala e que na grande maioria foram migradas para os provedores de serviço que oferecem soluções e tempos de reparos arrojados. Esta dissertação aborda o tema “alta disponibilidade” que é um dos requisitos básicos dos sistemas que utilizam a internet, apresentando os principais conceitos e as possíveis soluções encontradas no mercado. Serão apresentados alguns protocolos que suportam estas soluções apontando as suas vantagens e fraquezas bem como um estudo com base em problemas apresentados em um ambiente de produção. A identificação dos fenômenos de cérebro bi-partido e ausência de cérebro foi determinante para a solução do problema. O trabalho propõe uma extensão para um protocolo para arquiteturas de alta disponibilidade visando resolver os problemas apresentados e a utilização de uma ferramenta de descrição formal para garantir o seu funcionamento.

Palavras chave: HA, SCTP, VRRP, Métodos Formais, Engenharia de Protocolos.

ABSTRACT

Behind the growth of the amount of broadband users connections in the world, the wide use of mobile devices as cell phones and PDAs and other access from anywhere throughout Internet, always require some level of availability to support customer needs. This necessity made which the high availability architectures were used on large scale and they were migrated to service providers that they offer solutions and ensure the short time to repair. This dissertation addresses “high availability”, one of the basic requirements of today systems on the Internet, presenting the main concepts and the possible solutions. Some protocols will be presented that support these solutions showing the advantages and disadvantages as well as the study on the basis of the problems presented in a production environment. The identification of split-brain and no-brain phenomena was determiner for the solution problem. This work proposes a protocol extension for high availability architectures regarding split-brain and no-brain issues found out on protocols which resides in that kind of architecture and the utilization of the formal language methods to ensure its functioning.

Keywords: HA, SCTP, VRRP, Formal Methods, Protocol Engineering.

LISTA DE FIGURAS

Figura 1: Arquitetura Clássica de Sistema de HA [LOPES FILHO 2008]	5
Figura 2: SCTP Four-way <i>Handshake</i> [LOPES FILHO 2008]	9
Figura 3: Cabeçalho SCTP comum	10
Figura 4: Fatia de Dados	13
Figura 5: Fatia Heartbeat Request	13
Figura 6: Fatia Heartbeat ACK	14
Figura 7: Associação SCTP [LOPES FILHO 2008]	15
Figura 8: Topologia de firewalls/IPS com duas interfaces HA [LOPES FILHO 2008]	18
Figura 9: Fluxo de fatias associado à proposição	19
Figura 10: Estados de transição, Mestre operando na rede e ausente	21
Figura 11: Estados de Transição, Falha do Mestre e Retorno a Operação	21
Figura 12: Arquitetura HA	24
Figura 13: Camada HSOL	25
Figura 14: Componentes Keepalived [KEEPALIVED 2003]	29
Figura 15: Máquina de Estados VRRP [HINDEN 2004]	32
Figura 16: Ponto de conexão entre <i>Master</i> e <i>Slave</i>	38
Figura 17: Visão Global da Arquitetura Proposta	41
Figura 18: Abstração da Arquitetura Proposta	42
Figura 19: Diagrama de Estados do Protocolo VRRP	43
Figura 20: <i>Adver_PRI_255_Request</i>	46
Figura 21: <i>Adver_PRI_0_Request</i> e <i>Adver_PRI_0_Indication</i>	47
Figura 22: <i>Adver_PRI_LocalPRI_Request</i>	47
Figura 23: <i>Adver_Preempt_False_Indication</i>	48
Figura 24: <i>Adver_PRI_0_Indication</i>	49
Figura 25: <i>Adver_PRI_LocalPRI_Request</i>	49
Figura 26: Especificação Rede de Petri para as instâncias <i>Master/Slave</i> e <i>Provider</i>	50
Figura 27: Especificação em Rede de Petri do Autômato Proposto	51
Figura 28: Árvore de Cobertura	56
Figura 29: Autômato HA Proposto	57

LISTA DE TABELAS

Tabela 1: Bits mais significativos	11
Tabela 2: Tipos de fatias (<i>chunks</i>)	12
Tabela 3: Confiabilidade de Serviços Connectionless	37
Tabela 4: Matriz de Renomeação de Estados	45
Tabela 5: Correlação Marcação (M) e Estados (S)	58
Tabela 6: Relação Transição/Evento	59

LISTA DE ABREVIATURAS

ARP – *Address Resolution Protocol*
ATM – *Asynchronous Transfer Mode*
BGP – *Border Gateway Protocol*
CARP – *Common Address Redundancy Protocol*
CIA – *Confidentiality, Integrity and Availability*
CL – *Connectionless*
CLNS – *ConnectionLess Network Service*
CLTS – *Connectionless Transport Services*
COTS – *Connection Oriented Transport Services*
CRC32c – *Cyclic Redundancy Check*
CSP – *Communicating Sequential Processes*
DARPA – *Defense Advanced Research Project Agency*
DDoS – *Distributed Denial of Service*
FSM – *Finite State Machine*
HA – *High Availability*
HSOL – *Hardware Service Oriented Layer*
HSRP – *Hot Standby Router Protocol*
HTTP – *HyperText Transfer Protocol*
IANA – *Internet Assigned Numbers Authority*
ICNS – *Fourth International Conference on Networking and Services*
IEEE – *Institute of Electrical and Electronics Engineers*
IETF – *Internet Engineering Task Force*
IGMP – *Internet Group Management Protocol*
IMS – *IP Multimedia Subsystem*
IP – *Internet Protocol*
IPS – *Intrusion Prevention Systems*
ISO – *International Standards Organization*
LAN – *Local Area Network*
LVS – *Linux Virtual Server*
MAC – *Media Access Control*
MAC – *Message Authentication Code*
MPLS – *Multiprotocol Label Switching*
MTTR – *Mean Time To Repair*
MTU – *Maximum Transmission Unit*
OSI – *Open Systems Interconnection*
PDA – *Personal Digital Assistants*
PDU – *Packet Data Unit*
PR-SCTP – *Partial Reliability Stream Control Transmission Protocol*
QoS – *Quality of Service*
RFC – *Request for Comments*
RTT – *Round Trip Time*
SCTP – *Stream Control Transmission Protocol*
SLA – *Service Level Agreement*

SNMP – *Simple Network Management Protocol*

SPOF – *Single Point of Failure*

SSL – *Secure Sockets Layer*

TCP – *Transmission Control Protocol*

TSN – *Transmission Sequence Number*

UDP – *User Datagram Protocol*

URL – *Uniform Resource Locator*

VPN – *Virtual Private Network*

VRID – *Virtual Router Identifier*

VRRP – *Virtual Routing Redundancy Protocol*

1. Introdução

Com a popularização da Internet através de acessos cada vez mais fáceis e rápidos e aplicações cada vez mais dependentes de disponibilidade, mostrou-se necessário soluções calcadas em alto desempenho, confiabilidade e segurança. Esses acessos viabilizaram que sistemas, ora somente utilizados em ambientes corporativos, pudessem ter sua acessibilidade além de suas fronteiras.

No mundo dos negócios que dependem de agilidade e informações, essas necessidades ficam bastante evidentes e os requisitos de disponibilidade dessas soluções dependem do nível de severidade do sistema utilizado. Por exemplo, a disponibilidade pode estar relacionada à localização geográfica onde os sistemas estão instalados, ou seja, existem casos onde há a necessidade de locais físicos distantes entre si e redundantes para garantir o funcionamento de ambientes com requisitos de alta disponibilidade (HA – *High Availability*).

Com isso, os provedores de serviços propõem soluções complexas para usuários exigentes, contemplando disponibilidades muito próximas de 100%, ou seja, com tempo médio de reparação (MTTR – *Mean Time to Repair*) na ordem de minutos por mês.

Isto nos mostra que sistemas complexos de alta disponibilidade são conjuntos de várias arquiteturas e protocolos desde a camada de enlace, rede e transporte, cuja função é dar suporte a esses ambientes, verificando a saúde dos elementos de rede e fazendo a automática recuperação em caso de falhas.

É possível observar que algumas falhas destes sistemas estão relacionadas a problemas de hardware ou mesmo de infra-estrutura, porém verifica-se que isto também pode acontecer por uma implementação errônea ou falha na especificação do protocolo usado.

Os problemas de hardware e infra-estrutura são na maioria das vezes, de fácil resolução, fazendo a substituição de componentes ou redimensionando o ambiente. É claro

que cada caso tem sua particularidade e quando se fala em substituição ou ampliação de componentes sempre haverá a necessidade de investimento. Os problemas relacionados à implementação também são passíveis de solução, porém, se a falha estiver na especificação do protocolo, toda a cadeia falhará.

Alguns fenômenos, tais como “cérebro bi-partido” e “acéfalo”, foram encontrados em ambientes de produção que utilizam uma solução de HA. A análise desta falha possibilitou a identificação da real causa e a motivação para a solução do problema.

Esta dissertação aborda o tema disponibilidade e propõe uma extensão ao protocolo VRRP [HINDEN 2004], uma vez que foram identificados pontos de melhoria da especificação. Na realidade, o protocolo VRRP serviu de ponto de partida uma vez que os fenômenos nele detectados também ocorrem em outros protocolos citados ao longo deste trabalho.

A utilização de uma linguagem de descrição formal para a especificação da extensão do protocolo foi de suma importância para garantir e atestar o total funcionamento da proposta.

O restante desta dissertação está dividido em quatro capítulos, da seguinte forma: o Capítulo 2 introduz o estado da arte para os mecanismos de alta disponibilidade, descrevendo o protocolo SCTP [STEWART 2000] em suas principais características e a proposta de uma arquitetura de Alta Disponibilidade apresentada por [LOPES FILHO 2008] baseada no protocolo SCTP, com seus fundamentos e conclusões; o Capítulo 3 define o problema e mapeia como ele acontece, descrevendo também a arquitetura Keepalived [KEEPALIVED 2003] e o protocolo VRRP; o Capítulo 4 propõe uma extensão ao protocolo VRRP e sua arquitetura, descrevendo suas bases teóricas e sua formalização através de Redes de Petri, árvore de cobertura e o conseqüente autômato finito; e o Capítulo 5 apresenta a conclusão e indicações de trabalhos futuros.

2. Estado da Arte para Alta Disponibilidade

O trabalho apresentado por [LOPES FILHO 2008] teve como principal objetivo diminuir os efeitos colaterais causados pelas tecnologias de alta disponibilidade existentes hoje. Essa proposta baseou-se na experiência prática adquirida em equipamentos de segurança e de alta disponibilidade como *firewall* e IPS (*Intrusion Prevention Systems*).

Como parte do problema das tecnologias de HA encontra-se nas camadas subjacentes (em particular na camada de transporte), então a utilização do protocolo SCTP, na qual o grupo de redes desta universidade é pioneiro no Brasil e possui ampla experiência desde 2003 [CANTELLI 2003] [PEREIRA 2004], surge como parte da solução.

Este trabalho propõe a utilização do protocolo SCTP (*Stream Control Transmission Protocol* [STEWART 2000]) como transporte de primitivas de verificação de saúde (*health-checkers*) do sistema. A proposta de [LOPES FILHO 2008] procurou atender aos critérios básicos de uma gestão de segurança, sendo eles: integridade, confidencialidade e disponibilidade, ou a “Tríade CIA (*Confidentiality, Integrity and Availability*)”.

Este capítulo tem o objetivo de fazer um posicionamento de contexto e estabelecer as bases para a presente apresentação. A seção 2.1 (Mecanismo de Alta Disponibilidade) fará uma breve descrição sobre mecanismos de HA, a seção 2.2 (Aspectos Importantes do Protocolo SCTP) descreverá de forma sucinta os aspectos do protocolo SCTP importantes ao presente trabalho, e as seções 2.3 (A Arquitetura de HA Baseada no Protocolo SCTP, 2.4 (Fluxo de mensagens *heartbeat* da arquitetura), 2.5 (Visão Geral da Proposta de HA) e 2.6 (Conclusão) apresentarão uma consolidação do estado da arte.

2.1. Mecanismos de Alta Disponibilidade

O principal objetivo de uma arquitetura de HA é eliminar os pontos únicos de falha (SPOF – *Single Point of Failure*) de um sistema. Um SPOF é um componente cuja falha causará uma indisponibilidade imediata em todo o sistema ou serviço [CAMERON 2007].

Geralmente, os mecanismos de alta disponibilidade são caracterizados por usarem soluções baseados em redundância de hardware, software inteligentes e protocolos para identificação de falhas de sistemas. Essas funcionalidades têm sido incluídas em equipamentos tais como *clusters* de servidores e produtos de segurança como *firewall* e IPS [CAMERON 2007].

Em geral, as arquiteturas de HA são baseadas nas seguintes abordagens:

- Ativo/passivo – um dos equipamentos atua como principal (mestre) e os demais atuam como secundário (*backup*), sendo que o mestre envia todas as informações de estado para o(s) *backup*(s) e, se o mestre falhar, um dos *backups* é promovido a mestre e assume as sessões previamente estabelecidas;
- Ativo/ativo – todos os equipamentos são configurados para o estado ativo, compartilhando as sessões entre eles através de balanceamento de carga e se um dos equipamentos falha, os demais assumem as sessões e os respectivos tráfegos.

2.1.1. Redundância de Hardware

Uma arquitetura de alta disponibilidade, muitas vezes é relacionada à redundância de hardware para minimizar os impactos de SPOF. A Figura 1 ilustra uma arquitetura clássica desse mecanismo.

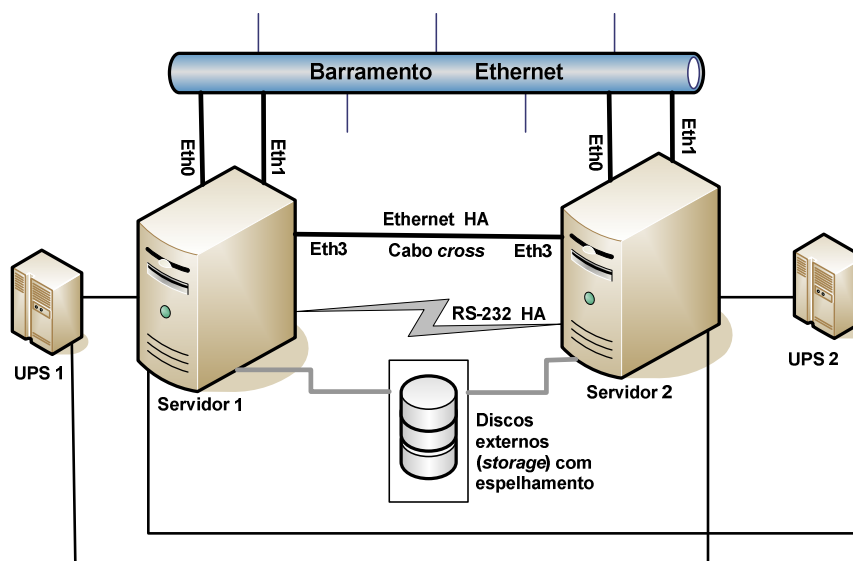


Figura 1: Arquitetura Clássica de Sistema de HA [LOPES FILHO 2008]

Pode-se observar que um sistema de alta disponibilidade deve ser resiliente a falha de software, hardware e energia, sempre objetivando manter os serviços disponíveis o máximo de tempo possível.

Redundância de hardware permite que vários equipamentos sejam arranjados, eventualmente em localizações geográficas distintas, de tal modo que os requisitos citados no parágrafo anterior sejam tratados e garantidos. Ao se oferecer as funcionalidades dos sistemas de múltiplos equipamentos, a falha de um equipamento não influi no funcionamento do sistema como um todo.

Contudo, a forma como a falha de um equipamento (um ponto) será reportada aos demais equipamentos pertencentes à arquitetura de HA é uma funcionalidade de camada superior e depende do autômato (software) que a implementa.

2.1.2. Sistemas de Software inteligentes

Os sistemas de software inteligentes residem em uma camada superior à camada de hardware e têm um papel fundamental na detecção de falhas e na reabilitação automática dos serviços oferecidos, sem a necessidade de intervenção humana e muitas vezes quase imperceptível ao usuário.

Normalmente, o único efeito colateral é percebido através de degradações no tempo de resposta, mas que podem ser minimizados a níveis imperceptíveis dependendo da quantidade de equipamentos que fazem parte da arquitetura de HA.

Essa inteligência nada mais é do que o monitoramento do sistema de HA através de *health-checkers* específicos. Um exemplo disto é a arquitetura Keepalived [KEEPAIVED 2003] apresentada na seção 3.1. Tal classe de software faz parte dos elementos de um protocolo, constituindo-se no autômato desses protocolos.

2.1.3. Protocolos para Identificação de Falha

De acordo com [HOLZMANN 1991], vários são os elementos que constituem um protocolo. Os sistemas de software inteligentes descritos na seção anterior necessitam dos protocolos que dão suporte ao sistema de HA e que são do ponto de vista conceitual, os demais elementos dos projetos utilizados para a identificação e notificação de falhas.

Esses protocolos, por razões óbvias, são associados a um sistema de software inteligente e têm como funções primordiais a detecção da falha e a tomada de ações para manter os serviços em funcionamento.

Dois destes protocolos serão descritos nesta dissertação e são base para o desenvolvimento deste trabalho: o protocolo SCTP (*Stream Control Transmission Protocol*) e o protocolo VRRP (*Virtual Router Redundancy Protocol*).

O protocolo SCTP será abordado na subseção 2.2 em função de ser um protocolo da camada de transporte e, para a arquitetura discutida neste trabalho, ser visto conceitualmente como uma camada *provider*. Posto de outra forma, a arquitetura de HA apresentada neste trabalho é usuária (faz uso do protocolo da camada de transporte SCTP) para as trocas de mensagens de alta disponibilidade.

O protocolo VRRP será abordado em profundidade no próximo capítulo, na seção 3.2, em função de ser um protocolo fundamental à proposição desta dissertação.

2.2. Aspectos Importantes do Protocolo SCTP

O protocolo SCTP (*Stream Control Transmission Protocol*) [STEWART 2000] é um protocolo de propósito geral para a camada de transporte, orientado a conexão (COTS – *Connection Oriented Transport Service*) [TANENBAUM, 1996], com controle de fluxo, entrega confiável, ordenada ou não.

Originalmente desenvolvido em 1998 pelo grupo de trabalho SIGTRAN/IETF (*Internet Engineering Task Force*) para suportar um mecanismo confiável para o transporte da sinalização de telecomunicações do controle de chamadas telefônicas sobre a Internet. A meta desse grupo era criar um complemento ao protocolo IP para a comutação de telefonia em redes SS7 (*Signaling System 7*) [ONG 1999].

SCTP é um protocolo de transporte *unicast* que suporta a troca de dados fim a fim. Ele pode identificar quando uma primitiva é perdida, ordenada, duplicada ou corrompida, retransmitindo-a quando necessário.

O protocolo SCTP provê um mecanismo de controle de congestionamento similar àquele oferecido pelo protocolo TCP (*Transmission Control Protocol*) [DARPA 1981].

Contudo, alguns problemas ou limitações identificados no protocolo TCP são resolvidos no protocolo SCTP, tais como:

- *Head-of-line blocking* – o protocolo SCTP minimiza os efeitos do bloqueio *head-of-line*, oferecendo múltiplos e independentes fluxos parcialmente ordenados ou desordenados, que ocorre quando múltiplas conexões de camadas superiores são multiplexadas sobre uma única conexão ordenada e as mensagens enviadas anteriormente são armazenadas e atrasadas em um *buffer* receptor até que a mensagem perdida anteriormente seja retransmitida e entregue;
- Ataques do tipo *SYN flood* [FERGUSON, 2000] – o protocolo SCTP utiliza o estabelecimento de conexão através do *four-way handshake*, mostrado na Figura 2, que usa *cookies* para a inicialização de uma nova associação de transporte.

Para o controle de sinalização, o bloqueio *head-of-line* é crítico, pois pode haver a expiração dos temporizadores do controle de chamadas e acontecer a indesejável queda das ligações. O fato é que o protocolo SCTP foi projetado para resolver esse problema em telecomunicação, um fenômeno também indesejável em praticamente todas as aplicações.

No caso de ataque do tipo *SYN Flood*, a relevância da utilização de *cookie* é que há a proteção por um código de autenticação segura (MAC – *Message Authentication Code*), que associa os elementos (pontos) envolvidos na associação. *Cookies* previnem, portanto, o ataque do tipo negação de serviço (DoS) que usa inundação de pacotes do tipo SYN.

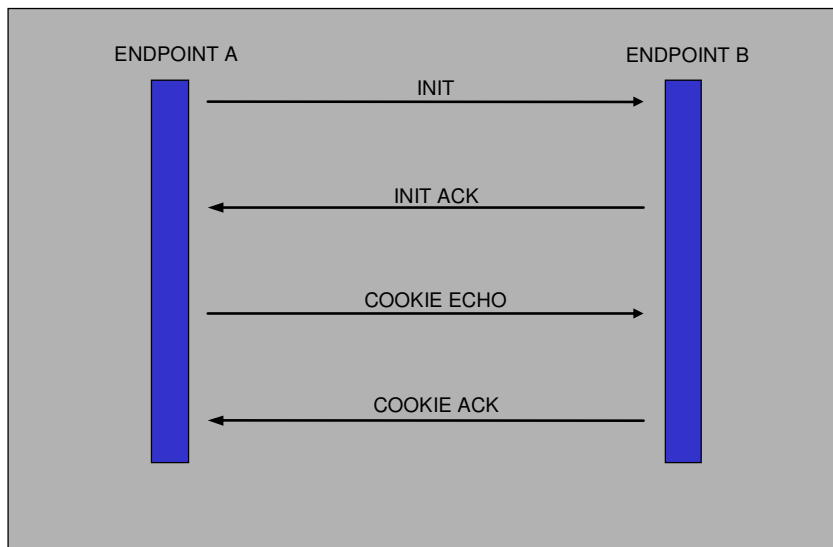


Figura 2: SCTP Four-way Handshake [LOPES FILHO 2008]

Embora seja um protocolo orientado a conexão, a configuração da associação pode ensejar a entrega desordenada de primitivas (mensagens). Contudo, o protocolo SCTP oferece entrega confiável mesmo para mensagens fora de ordem. Essa facilidade é muito interessante se comparada ao protocolo UDP (*User Datagram Protocol*) [POSTEL 2000] que provê um serviço desordenado e sem confiabilidade, isto é, uma mensagem pode simplesmente não ser entregue.

2.2.1. O Cabeçalho Comum do Protocolo SCTP

O protocolo SCTP apresenta um cabeçalho comum constituído de 12 octetos conforme é apresentado na Figura 3, utiliza o algoritmo CRC32c – *Cyclic Redundancy Check* (campo Soma e Controle) para detecção de erros de transmissão e integridade, onde cada primitiva é protegida por uma soma de controle (*checksum*) de 32 bits [STONE 2002]. Esse algoritmo é mais robusto que o utilizado pelos protocolos TCP e UDP que usam a soma de controle de 16 bits.

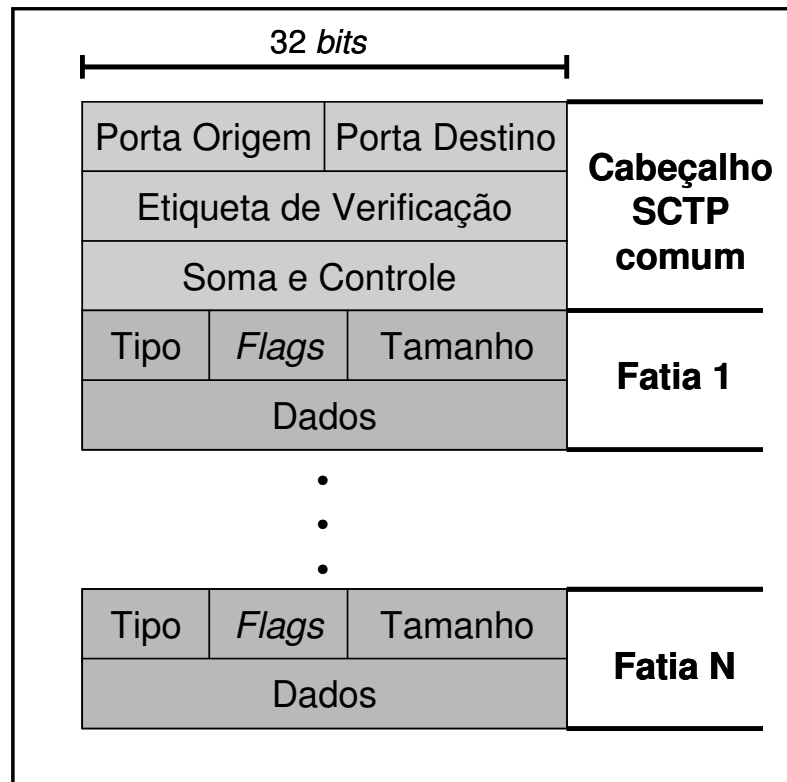


Figura 3: Cabeçalho SCTP comum

O protocolo SCTP cria a independência entre dados transmitidos e dados entregues. Cada fatia se inicia com um campo “Tipo” usado para distinguir as fatias dos tipos Dados ou Controle. Em particular, cada fatia (*chunk*) de dados usa dois conjuntos de seqüências, uma denominada TSN (*Transmission Sequence Number*) que controla a transmissão das mensagens e a detecção de perda, e o *Stream ID/Stream Sequence Number* que é usado para determinar a seqüência de entrega de dados recebidos.

O campo “Tipo” é um número de 8 bits onde os dois primeiros bits (bits mais significativos) definem a decisão da entidade destino, ilustrada pela Tabela 1 e os bits restantes referem-se propriamente ao tipo de fatia mostrada na Tabela 2.

Tabela 1: Bits mais significativos

Bits	Atitude do receptor
00	Pare o processamento deste pacote, descarte-o e não processe nenhuma outra fatia dentro deste pacote.
01	Mesma atitude dos “Bits 00” e informe que um parâmetro não reconhecido foi encontrado.
10	Ignore esta fatia e continue o processamento.
11	Mesma atitude dos “Bits 10” e informe que uma fatia não reconhecida foi encontrada.

Tabela 2: Tipos de fatias (*chunks*)

Número	Binário	Tipo da fatia (<i>chunk</i>)
0	00000000	Dados (<i>DATA</i>)
1	00000001	Início de conexão (<i>INIT</i>)
2	00000010	Reconhecimento de Identificação (<i>INIT ACK</i>) - O recebimento do <i>INIT ACK</i> estabelece a associação
3	00000011	Reconhecimento seletivo (<i>SACK</i>) - Reconhece o recebimento de fatias de dados (<i>DATA</i>).
4	00000100	Requisição de <i>Heartbeat</i> (<i>HEARTBEAT</i>)
5	00000101	Reconhecimento de <i>Heartbeat</i> (<i>HEARTBEAT ACK</i>)
6	00000110	Aviso de fim de conexão abrupta (<i>ABORT</i>)
7	00000111	Fim de conexão (<i>SHUTDOWN</i>)
8	00001000	Reconhecimento de fim de conexão (<i>SHUTDOWN ACK</i>)
9	00001001	Erro de operação (<i>ERROR</i>)
10	00001010	Situação <i>Cookie</i> (<i>COOKIE ECHO</i>)
11	00001011	Reconhecimento do <i>Cookie</i> (<i>COOKIE ACK</i>)
12	00001100	Reservado para notificação explícita de congestionamento (<i>ECNE</i>)
13	00001101	Reservado para redução da janela de congestionamento (<i>CWR</i>)
14	00001110	Fim de conexão completa (<i>SHUTDOWN COMPLETE</i>)
15 a 62		Reservado pelo IETF (<i>Internet Engineering Task Force</i>)
63	00111111	Definido pelo IETF para extensões de fatias
64 a 126		Reservado pelo IETF
127	01111111	Definido pelo IETF para extensões de fatias
128 a 190		Reservado pelo IETF
191	10111111	Definido pelo IETF para extensões de fatias
192 a 254		Reservado pelo IETF
255	11111111	Definido pelo IETF para extensões de fatias

Para o escopo deste trabalho, far-se-á uma descrição de fatias do tipo *DATA* e *HEARTBEAT*, sendo que os demais tipos de fatias são abordados com profundidade em [CANTELLI 2003] [LOPES FILHO 2008]. Para facilidade de compreensão do texto, ao invés de referenciar estes tipos de fatias, por exemplo “fatia do tipo *DATA*”, usar-se-á “fatia de dados” e “fatia de *heartbeat*”.

A fatia de dados ilustrada pela Figura 4 é utilizada para transportar os dados vindos da camada de aplicação.

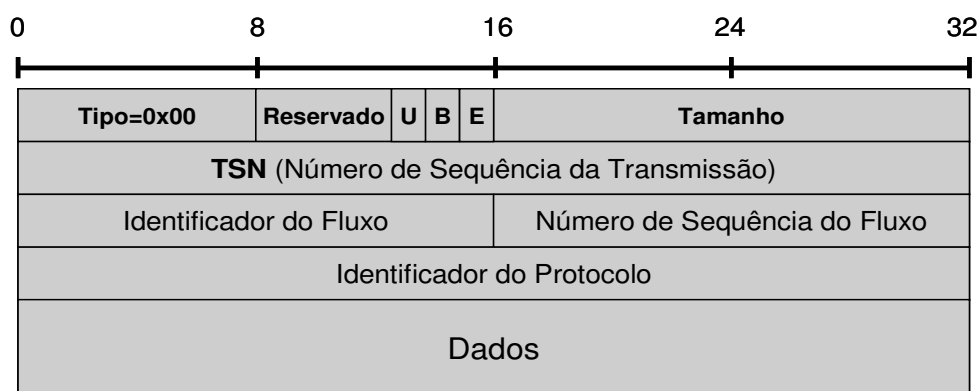


Figura 4: Fatia de Dados

A fatia de heartbeat (*HEARTBEAT*) é a responsável por verificar se a entidade remota está alcançável (reachability) através do IP definido na associação. A Figura 5 ilustra a estrutura da fatia Heartbeat Request.

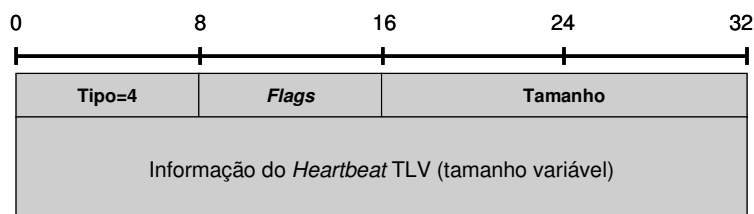


Figura 5: Fatia Heartbeat Request

Juntamente com a fatia *HEARTBEAT*, a entidade remota deve responder com a fatia *HEARTBEAT ACK*. Esta fatia é enviada ao endereço IP destino contido no pacote SCTP da fatia *Heartbeat Request*. A Figura 6 ilustra esta estrutura.

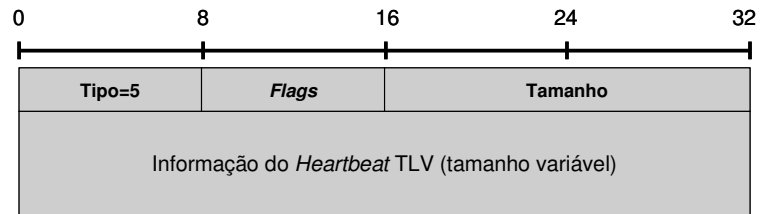


Figura 6: Fatia Heartbeat ACK

Para a detecção de uma falha, basicamente, o protocolo SCTP utiliza dois métodos: *heartbeat* – que é responsável por manter indicação de vivacidade da conexão e o limite (*thresholds*) de retransmissão de dados. Quando a entidade A envia uma fatia (*HEARTBEAT*) para a entidade B, ele espera a resposta da entidade B com o “*HEARTBEAT ACK*”. Se a entidade A não receber a primitiva “*HEARTBEAT ACK*”, a variável *Destination Error* é incrementada e, se exceder o limite (usualmente 5), a entidade B é declarada como inalcançável.

Uma extensão do protocolo SCTP é o PR-SCTP (*Partial Reliability Extension – SCTP*) [STEWART 2004] que permite estabelecer indicadores (*flags*) para o tratamento de mensagens fora de ordem (desordenados) de acordo com os requisitos de uma aplicação. A “disponibilidade programada” introduz a capacidade de se operar com uma disponibilidade parcial do serviço conforme especificado no protocolo PR-SCTP. Isto permite definir o tempo de vida de uma mensagem, sendo que quando este tempo expira a rede descarta as fatias de dados contendo a mensagem.

As próximas duas seções oferecem um breve resumo de aspectos do protocolo SCTP que o tornam vantajoso se comparado ao protocolo TCP, tais como múltiplos fluxos (*multistreaming*) e múltiplos endereços (*multihoming*), aspectos estes que ampliam as capacidades de disponibilidade.

2.2.2. Mecanismo de *Multistreaming*

O protocolo SCTP introduz o conceito de associação, que caracteriza a existência de relacionamento entre duas entidades usuárias da camada de transporte. Considerando-se a arquitetura Internet [TANENBAUM 1996], estas entidades usuárias são entidades da camada de aplicação. Este relacionamento, que para melhor entendimento pode ser visto como uma conexão enriquecida por várias facilidades de transporte suporta múltiplos e independentes fluxos (*streams*) de dados em uma mesma associação. Um fluxo é unidirecional e transporta mensagens ordenadas ou desordenadas, dependendo da configuração da associação. A Figura 7 ilustra uma associação com múltiplos fluxos (*streams*).

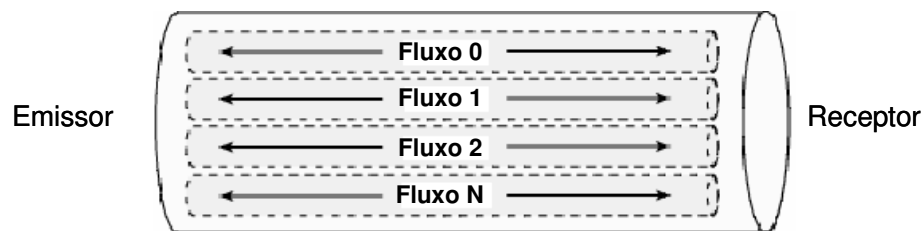


Figura 7: Associação SCTP [LOPES FILHO 2008]

O protocolo SCTP suporta múltiplos e independentes fluxos lógicos de mensagens dentro de uma associação. Cada mensagem enviada sobre uma associação é transferida a um fluxo particular. Esta característica permite que os dados sejam particionados dentro de múltiplos fluxos onde a propriedade de entrega é independente e seqüenciada, ainda que mensagens perdidas em um fluxo afetarão, em princípio, somente a entrega dentro deste fluxo e não nos outros fluxos.

Por outro lado, o protocolo TCP assume um único fluxo de dados e garante a entrega com a preservação da seqüência em nível de bytes. Isto pode ser desejável em grandes quantidades de dados a serem transportados, causando, entretanto, um adicional atraso quando a perda ou seqüência de erros ocorrer dentro da rede.

Quando isto acontece, o protocolo TCP atrasa a entrega de dados até que a seqüência correta seja restaurada, ou por receber mensagem fora da seqüência ou por retransmissão por

perda da mensagem. Para algumas aplicações, a característica de preservação da sequência não é realmente necessária.

A característica *Multistreaming* permite transportar estas mensagens parcialmente ordenadas ao invés de estritamente ordenado, resultando em uma melhoria na percepção do usuário. O protocolo SCTP transporta estes dados em uma única associação, com o mesmo mecanismo de congestionamento, reduzindo o *overhead* no nível de transporte.

2.2.3. Mecanismo de *Multihoming*

Outra importante funcionalidade do protocolo SCTP é o *multihoming*, ou seja, a habilidade de um simples elemento (ponto) SCTP pode ser endereçada por múltiplos endereços IP. O maior benefício do *multihoming* é a habilidade de manter uma sessão com um elemento *backup* na presença de uma falha de rede.

Em uma sessão convencional, a falha de uma rede local ethernet pode isolar o sistema, enquanto que as falhas no núcleo da rede (*core*) podem causar temporariamente a indisponibilidade no transporte até que os protocolos de roteamento IP possam convergir em torno da falha. Redes locais redundantes podem ser usadas para reforçar o acesso local, enquanto várias opções são possíveis no núcleo da rede, reduzindo a dependência de falhas para diferentes endereços. O uso de endereços de diferentes prefixos pode forçar o roteamento através de diferentes provedores.

Desta forma, o protocolo SCTP não faz balanceamento de carga, isto é, ele é usado somente para o propósito de redundância. Um simples endereço é escolhido como o endereço primário e é usado no destino de todas as fatias de dados de uma transmissão normal.

Dados retransmitidos usam o(s) endereço(s) alternativo(s) para alcançar o elemento remoto, enquanto a falha do endereço primário acontece, todas as fatias de dados serão

enviadas para o endereço alternativo até que o *heartbeat* possa restabelecer a alcançabilidade do primário. Para suportar o *multihoming*, os elementos SCTP trocam listas de endereços durante a iniciação da associação.

Um elemento com configurações *multihoming* apresenta múltiplas interfaces de rede e mais de um endereço IP. Quando uma falha acontece exatamente na interface escolhida para o tráfego de dados, a escolha de uma interface alternativa não depende da convergência de roteamento.

2.2.4. *Multihoming* e HA

As facilidades oferecidas pelo protocolo SCTP, tais como *multihoming* e *multistreaming*, apresentadas nas seções anteriores, aumentam significativamente a capacidade de disponibilidade. Isto mostra que esse protocolo está preparado para o desenvolvimento de aplicações e suporte a sistemas de alta disponibilidade.

É importante ressaltar que o uso de um protocolo de transporte que cuide dos aspectos descritos acima, diminui bastante o tempo de resposta em caso de falha, pois a falha é automaticamente contornada pela camada de transporte no núcleo do sistema operacional.

Do ponto de vista da aplicação, não há múltiplos envios de dados, mas simplesmente um envio. Cabe à camada de transporte enviar as múltiplas instâncias de dados aos diversos elementos da arquitetura de alta disponibilidade. Não se trata de detectar uma falha e fazer os eventuais re-envios, mas sim envios normais a diversos elementos da infra-estrutura.

2.3. A Arquitetura de HA Baseada no Protocolo SCTP

O trabalho proposto por [LOPES FILHO 2008] tem como objetivo melhorar os mecanismos de alta disponibilidade voltados para a necessidade do mercado, tais como disponibilidade geográfica, redundância de sites e menores custos. A Figura 8 ilustra a topologia base para o detalhamento da proposta.

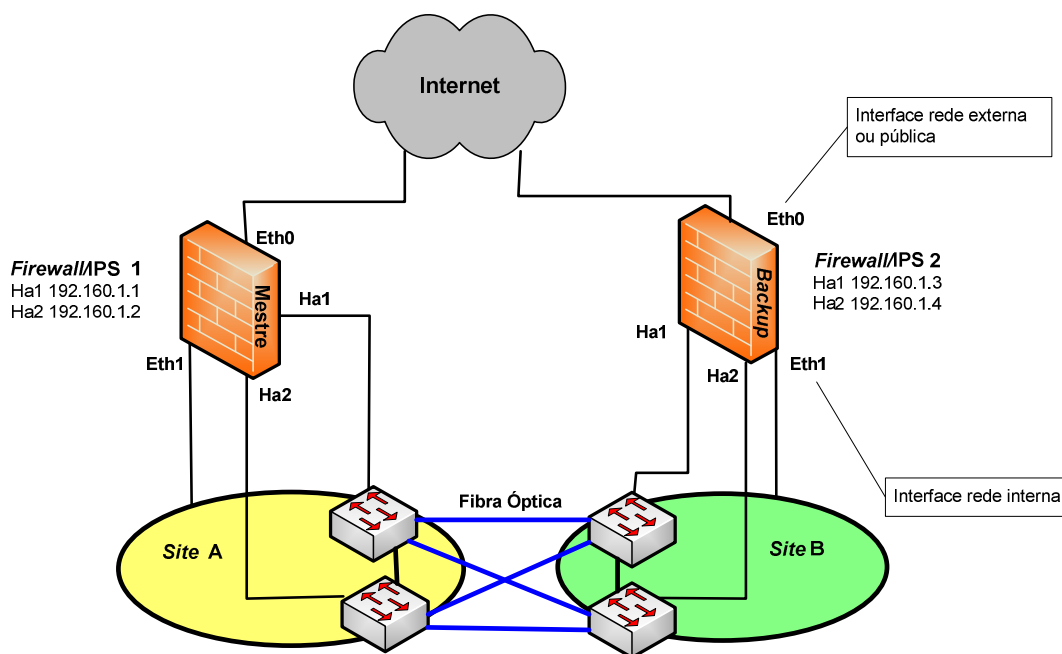


Figura 8: Topologia de firewalls/IPS com duas interfaces HA [LOPES FILHO 2008]

A utilização do protocolo SCTP deve-se às suas características naturais de confiabilidade necessárias aos mecanismos de HA descritos anteriormente. A proposta coloca o protocolo SCTP com a função de envio de mensagens de controle das interfaces HA utilizando-se de seu próprio mecanismo de verificação de saúde (*heartbeat*). O VRRP ficará com a função de controle de estado das interfaces HA e seus respectivos IP virtuais.

2.4. Fluxo de Mensagens *Heartbeat* da Arquitetura

Uma associação estabelecida entre os equipamentos ativo e passivo utiliza um fluxo para a fatia do tipo *DATA* – que transporta as informações do estado do equipamento Mestre para a máquina *Backup* – e para a fatia do tipo *HEARTBEAT* – que realiza a verificação da saúde das interfaces HA. Considerando a topologia da Figura 8, as fatias descritas na Figura 9 são transportadas em uma única associação com o seguinte formato:

- IP-Mestre-Ha1:Porta-Origem= {[IP-Backup-Ha1, IP-Backup-Ha2: Porta-HA]}

ou, por exemplo, utilizando os respectivos endereços IP

- 192.160.1.1:2008 = {[192.160.1.3, 192.160.1.4:20005]}

É possível a utilização de faixas de IP distintas, desde que existam mecanismos de roteamento e o emprego de QoS (*Quality of Service*) [ARMITAGE 2000] seja considerado.

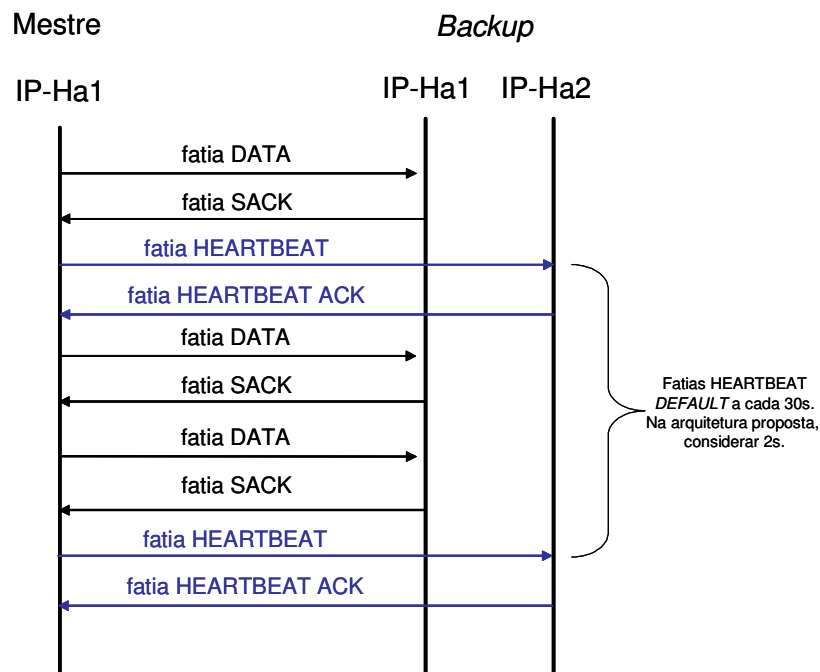


Figura 9: Fluxo de fatias associado à proposição

A identificação da falha é feita pela fatia *HEARTBEAT* e pela condição limite de retransmissão de dados. No processo de iniciação, um dos IP contidos na associação é eleito como primário e o mecanismo de monitoramento terá as seguintes características:

- O *heartbeat* é enviado para o endereço destino, a menos que ocorra um estouro do tempo (*timeout*);
- A contabilização de resposta atualiza o RTT (*Round Trip Time*), as fatias *DATA* e *HEARTBEAT*;
- O contador de tempo do *heartbeat* é re-iniciado toda vez que uma fatia de *DATA* ou *HEARTBEAT* é enviada;
- O receptor responde com uma fatia *HEARTBEAT-ACK*;
- Toda vez que uma fatia *HEARTBEAT* é enviada, a variável de registro de erro para o destino específico é incrementada;
- Toda vez que a fatia *HEARTBEAT-ACK* é recebida, o contador de erros é zerado;
- Toda vez que uma fatia de dados (*DATA*) enviada para o destinatário é confirmada (*SACK*), o contador de erros é zerado;
- Toda vez que o contador de erros exceder o limite preestabelecido (por padrão cinco), o destino é declarado como não acessível;
- Se o endereço de destino primário é marcado como não acessível e se existir endereço alternativo, este será escolhido e utilizado;
- Fatias *HEARTBEAT* continuarão a ser enviadas para os endereços não acessíveis, sendo que se houver resposta, o contador de erros é zerado e o destino é marcado como acessível;
- Para o último caso, se este destino (endereço não acessível) for o endereço IP eleito como primário no início da associação e não houve nenhuma intervenção do usuário, a comunicação é restaurada para este endereço.

A topologia de rede da Figura 8, mostrada na seção anterior permite que a funcionalidade de *multihoming* do protocolo SCTP seja utilizada, provendo caminhos alternativos para o sistema de HA.

A Figura 10 descreve os estados de transição e a troca de fatias entre o Mestre e *Backup*.

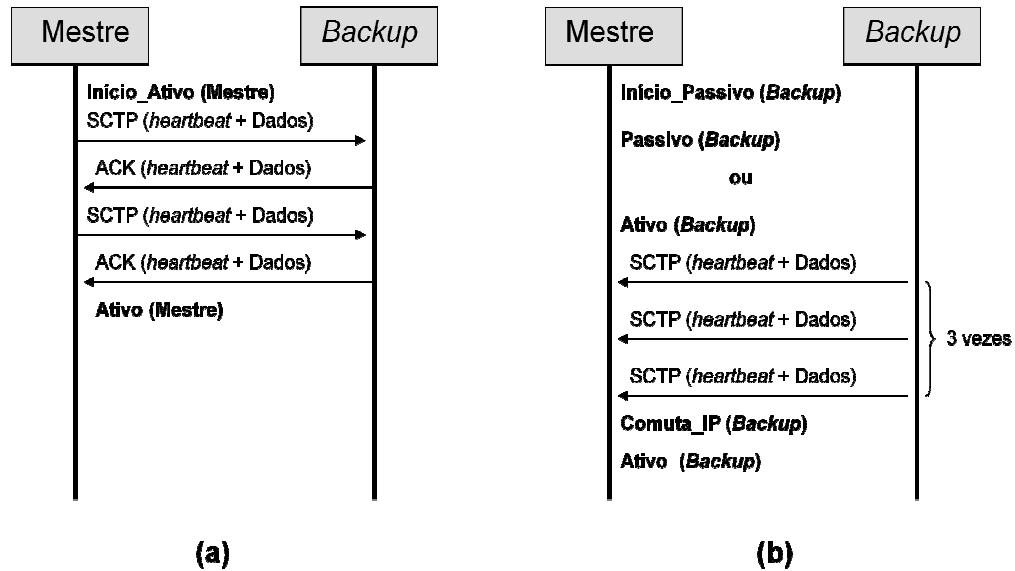


Figura 10: Estados de transição, Mestre operando na rede e ausente

A Figura 11 descreve os estados de transição e as mensagens quando ocorre a falha do Mestre e seu retorno ao estado operacional.

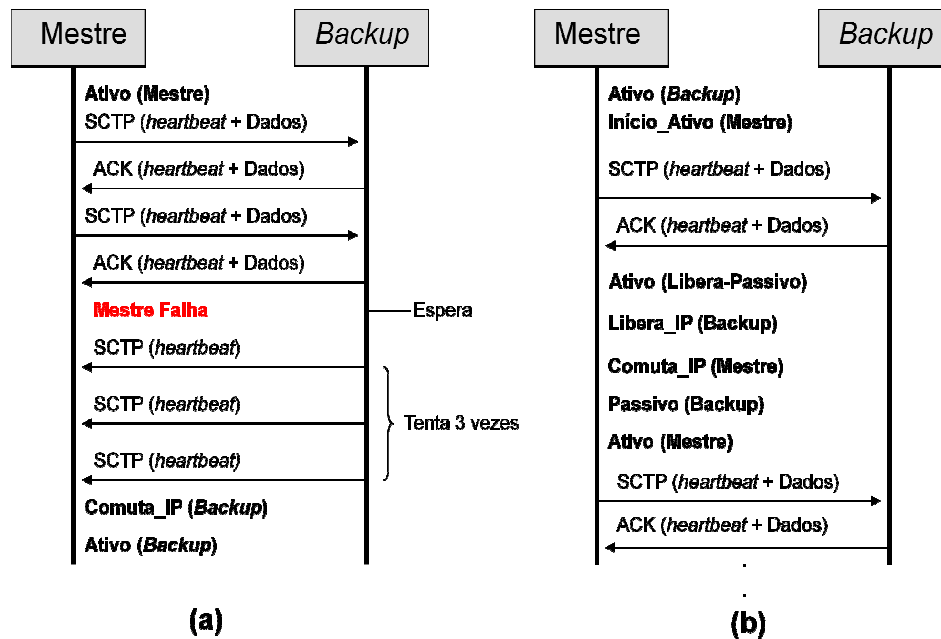


Figura 11: Estados de Transição, Falha do Mestre e Retorno a Operação

No estado de operação normal, o Mestre envia mensagens de dados e *heartbeat* para o *Backup*. Em caso de falha do Mestre, mostrado na Figura 11(a), o *Backup* verificará que não

mais está recebendo fatias de Dados, envia três mensagens de *heartbeat* e caso não haja resposta, assumirá a função de Mestre e passará a responder pelos IP das interfaces HA.

O *Backup* só retornará ao estado inicial se o Mestre voltar à rede, voltando a enviar mensagens de dados e *heartbeat*, como mostrado na Figura 11 (b).

Para ajudar na melhor compreensão do mecanismo, o modelo de funcionamento foi definido em uma metalinguagem em um alto nível de abstração:

Início_Ativo (Mestre)

```

    Se existe Ativo(passivo) Então      #O mestre está iniciando com backup
operando

    Ativo (Libera-Passivo)      # na rede como mestre

    Senão Ativo(Mestre)  # Início normal do mestre

Fim-Se

```

Início_Passivo ()

```

    Passivo ( )  # Passivo em operação, esperando

                # falha do mestre

```

Ativo ()

```

    Se Libera-Passivo Então #Backup operando como mestre

        Libera_IP (Backup) # controle retornará ao mestre

        Comuta_IP (Mestre)

        Passivo (Backup)

        Ativo (Mestre)

    Senão Se  Mestre Então #Mestre operando normalmente

        ARP-Gratuito (IP-Virtual)

    Envia pacotes de heartbeat e tabela de estado para o Backup (passivo)

        Senão

            Se Passivo Então #Backup assumirá como mestre

```

```

    Para I = 1 até 3 segundos Faça
        Envia mensagens de heartbeat para o mestre
        Se resposta = ok Então # O mestre retornou
            Passivo (Backup)
            Ativo (Mestre)
        Fim-Se
    Fim-Faça
    Comuta_IP (Backup) # O IP virtual é migrado para o
Backup
        Variável-Passivo = "Nó ativo como mestre é o
backup"
        Ativo (Backup)
        Senão # "Nó ativo como mestre é o Backup"
        # Não é necessário tomar o IP virtual novamente
        Armazena tabela de estado
        Ativo (Variável-Passivo)
        Fim-Se
    Fim-Se
Fim-Se

Passivo ( )
    Se existe Ativo (Mestre) Então
        Responda as mensagens de heartbeat e tabela de estado recebida
        Passivo (Backup)
    Senão # Não existe nenhum mestre operando
        Ativo (Backup)
    Fim-Se

```

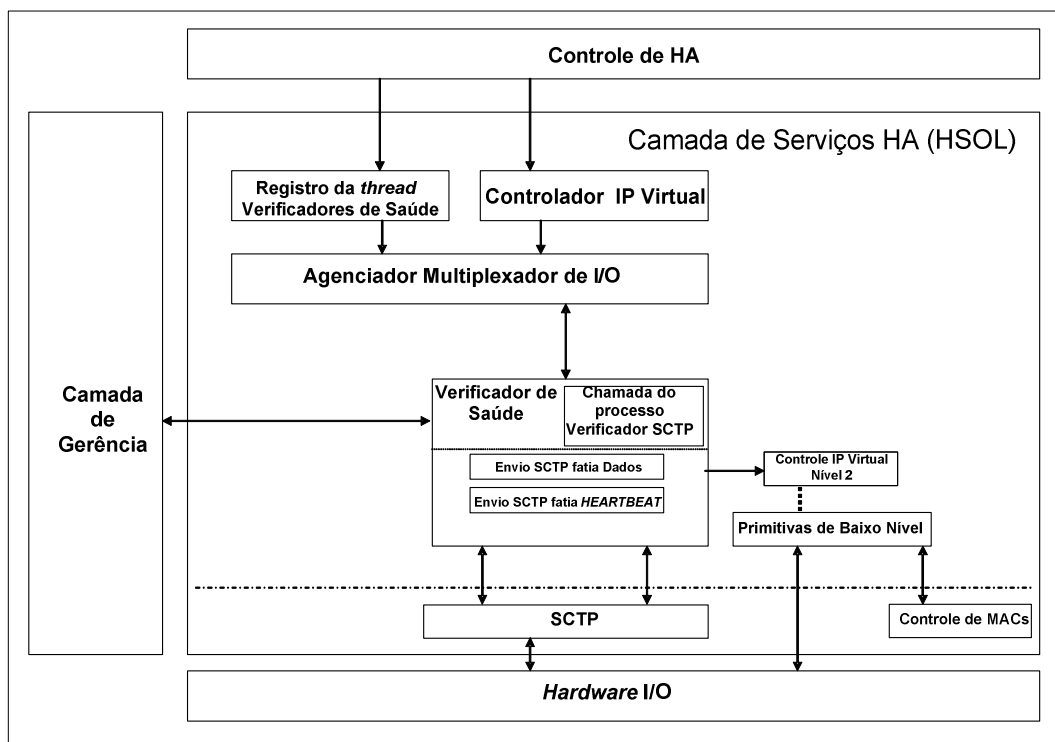



Figura 13: Camada HSOL

Nessa camada estão os mecanismos de controle de IP virtual definidos pelo VRRP, que controla o IP gateway das redes internas e externas.

Para a notificação de erros e mensagens informativas é utilizado o protocolo SNMP (*Simple Network Management Protocol*) [HARRINGTON 1999]. O mecanismo desta camada é responsável pelo envio de mensagens para o controle interno do *firewall/IPS* e/ou um servidor de log [LOPES FILHO 2008].

De fato, para propósito deste trabalho, os módulos “Verificador de Saúde” e o “Controlador de IP Virtual” são os mais importantes da estrutura e serão descritos a seguir:

- Verificador de saúde – é o responsável pela chamada interna dos processos associados ao protocolo SCTP, pelo controle de envio/recebimento de respostas das mensagens de Dados e *heartbeat* e comunica-se com a camada de Gerência de Rede e com o módulo “Controlador de IP Virtual”;
- Controlador de IP virtual – é o responsável pelo mecanismo de controle do IPs virtuais, tanto para as interfaces internas quanto para as interfaces externas,

sendo que este mecanismo é estruturado sobre o protocolo VRRP onde se faz necessária uma instância de controle de IP virtual para cada interface de rede não HA (note-se que o protocolo SCTP fará a função de verificação da saúde do sistema HA, enviando mensagens de *heartbeat* e dados para os equipamentos *Backups*);

- Controle de IP Virtual Nível 2 – este módulo funciona basicamente do mesmo modo que o módulo “Controlador de IP virtual”, porém está diretamente relacionado à verificação de *heartbeat* e alteração de IPs no caso de detecção de problemas;
- SCTP – esta camada recebe as fatias de Dados e *heartbeat* oriundos do “Verificador de Saúde” e os encaminham para a camada IP e vice-versa, sendo que o processo de *multihoming* é feito de forma transparente para as outras camadas.

A arquitetura proposta utiliza um combinado dos protocolos SCTP e protocolo VRRP para o controle de saúde das interfaces HA, internas e externas, respectivamente, ressaltando-se que a utilização do protocolo SCTP como transporte evita as vulnerabilidades associadas às implementações estruturadas sobre alguns protocolos *multicast* como IGMP (*Internet Group Management Protocol*) [CAIN 2002] e outros protocolos IP *multicast*.

2.6. Conclusão

As vantagens do protocolo SCTP sobre o protocolo TCP basicamente consideram o fato de que o protocolo SCTP entrega dados em fatias com fluxos independentes na mesma associação, eliminando problemas associados com o bloqueio *head-of-line*. Diferentemente do protocolo SCTP, o protocolo TCP entrega as mensagens através de fluxos de *bytes*, o que

aumenta a possibilidade de travamento (*stall*) devido ao espaço de numeração. A proposta definida por [LOPES FILHO 2008] utiliza fatias *DATA* e *HEARTBEAT* em uma única associação com fluxos independentes e ordenados.

Outra importante vantagem do protocolo SCTP para o transporte de mensagens de HA é o *multihoming*. Em uma associação SCTP, mais de um endereço IP pode ser utilizado para o estabelecimento desta associação. Assim, se o endereço primário falhar o fluxo é direcionado para o outro endereço.

Esta característica é particularmente importante devido ao fato que tira da aplicação a responsabilidade de enviar/receber mensagens/acks de diversos pontos. Ao fazer isso nativamente na camada de transporte, diminui-se o *overhead* em duas frentes: de processamento, uma vez que módulos no núcleo do sistema operacional tendem a fazer melhor uso de recursos do que no espaço do usuário; e de comunicação, uma vez que uma associação pode endereçar várias máquinas.

Para a questão de segurança, a arquitetura de HA proposta por [LOPES FILHO 2008] é imune a ataques DDoS do tipo *SYN Flood*. O mecanismo de autenticação baseado em *cookie* é mais eficiente do que o mecanismo utilizado pelo protocolo VRRP, evitando assim ataques contra VRID e a sobreposição de endereços *multicasting*.

Além disso, a característica de *multihoming* permite que mais de uma interface HA seja utilizada, e que o processo de comutação em caso de falha seja transparente para a camada que controla o estado operacional das interfaces HA. A utilização de mais de uma interface HA resolve de forma implícita e transparente o problema do Cérebro Bipartido ou “*Split-Brain*”, pois a verificação da saúde da rede será feita pelo protocolo SCTP (COTS) e não mais pelo protocolo *multicast*. A arquitetura proposta apresenta maior flexibilidade e melhor confiabilidade do que as demais arquiteturas.

3. Definição do problema

A arquitetura proposta por [LOPES FILHO 2008] para equipamentos *Firewall* e IPS, demonstrou-se factível pelos estudos detalhados de cada um dos problemas lá apresentados.

Contudo, alguns fenômenos restaram inexplicados, em particular o cérebro bi-partido (*split-brain*) e a ausência do Master (*no-brain*). Partindo-se deles, justifica-se a proposta deste trabalho que visa demonstrar formalmente a solução utilizando-se de métodos formais para verificar a confiabilidade da arquitetura de HA proposta.

Este capítulo apresenta uma breve descrição da arquitetura Keepalived [KEEPALIVED 2003] na seção 3.1, do protocolo VRRP [HINDEN 2004] na seção 3.2, bem como a definição do fenômeno “*Split-Brain*” na seção 3.3. A seção 3.4 apresentará resumidamente a causa dos problemas apresentados.

3.1. Arquitetura Keepalived

Keepalived [KEEPALIVED 2003] é uma arquitetura que adiciona robustez ao processo de verificação de saúde e implementa um protocolo *hot standby* para o processo de alta disponibilidade. Essas duas características ajudam o Linux Virtual Server (LVS) [LVS 2008] na manipulação de clusters de servidores Linux, adicionando-os ou removendo-os baseados na decisão dos verificadores de saúde.

O LVS é uma parte do Kernel do Linux que adiciona as facilidades de balanceamento de carga. O Keepalived é também usado para o suporte de *firewall* estruturado sobre Linux. A Figura 14 ilustra os componentes internos do Keepalived.

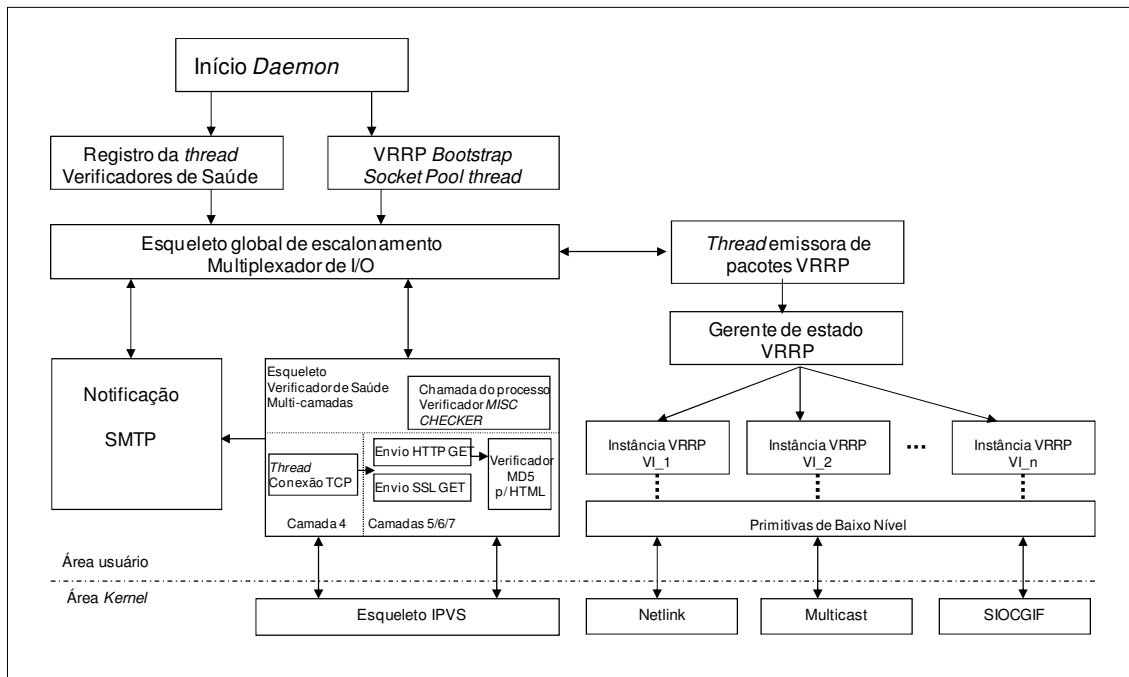


Figura 14: Componentes Keepalived [KEEPAIVED 2003]

O Keepalived usa uma abordagem *multithreaded* baseada em um multiplexador de E/S central e os principais componentes são o “Verificador de Saúde” e o “VRRP Packet Dispatcher”.

O “Verificador de Saúde” constitui-se de:

- Verificador TCP – trabalha na camada 4 da arquitetura Internet e garante a verificação através da verificação de conexões TCP *nonblocking/timed-out* e quando o servidor remoto não responde a uma requisição TCP após um determinado tempo (*time-out*), o resultado do teste é declarado como falho e o servidor é removido do conjunto;
- HTTP (*HyperText Transfer Protocol*) [FIELDING 1999] GET – trabalha na camada 5 da arquitetura Internet e checa através do envio de mensagens HTTP GET para uma URL (*Uniform Resource Locator*) pré definida, sendo que se o resultado não for o valor esperado, então o teste é considerado falho e o servidor é removido do conjunto;

- *SSL (Secure Sockets Layer) GET* – executa o mesmo teste da mensagem HTTP GET, porém utilizando o protocolo SSL [YLONEN 2006]; e
- *MISC CHECK* – esse verificador permite que um *script* de usuário seja ser utilizado como verificador, onde o resultado deve ser 0 (falha), que indica teste falho, ou 1 (positivo).

O *VRRP Packet Dispatcher* tem como principais funcionalidades:

- Administrar sincronismo das instâncias do protocolo VRRP;
- Fazer a passagem de atividades de um elemento falho para um elemento ativo (*failover*); e
- Fazer chamadas ao sistema através de programas ou *scripts*.

3.2. Protocolo VRRP

O protocolo VRRP (*Virtual Router Redundancy Protocol*) [HINDEN 2004] é um protocolo da arquitetura Internet (RFC 3768) que introduz a definição de roteador virtual, provendo maior confiabilidade aos ambientes de rede e resolvendo o problema do ponto de falha único (SPOF – *Single Point of Failure*) de uma LAN quando este usa somente um único *gateway* IP.

O protocolo VRRP é um protocolo de redundância de roteamento IP desenvolvido para permitir uma transparente comutação do *gateway* da rede. Esse método provê a redundância sem a intervenção manual do usuário ou mesmo sem configuração adicional nos elementos de rede.

Tal mecanismo especifica um protocolo de eleição que dinamicamente associa responsabilidades a um roteador virtual. O VRID (*Virtual Router Identifier*) identifica o roteador virtual e um conjunto de endereços IP. O roteador físico eleito controla os endereços

IP associado ao roteador virtual. O roteador virtual é chamado *Master* que envia periodicamente mensagens de anúncios VRRP aos demais roteadores chamados de *Backup*. O roteador *Backup* somente se torna *Master* se o *Master* se tornar indisponível ou se a prioridade do *Backup* for mudada para um valor maior do que a prioridade do *Master* corrente.

Existem vários benefícios na utilização do VRRP, tais como:

- Redundância – O VRRP habilita configurações de múltiplos roteadores com um único *gateway* IP, aumentando assim a disponibilidade do sistema;
- Múltiplos roteadores virtuais – suporta até 255 roteadores virtuais por interface física, habilita e implementa a redundância e o balanceamento do tráfego LAN.

O protocolo VRRP usa um endereço padrão *multicast* (224.0.0.18) definido pelo IANA (*Internet Assigned Numbers Authority*) para enviar os seus anúncios de mensagens. O roteador virtual é associado com o endereço MAC (00:00:5E:00:01:VRID) onde os três primeiros octetos são definidos também pelo IANA e os próximos dois octetos indica o endereço associado ao protocolo VRRP. O octeto VRID é o identificador do roteador virtual VRRP. Isto possibilita mapear até 255 roteadores no protocolo VRRP. Todos os roteadores virtuais terão os mesmos endereços IP e MAC (*Media Access Control*), mas somente o *Master* pode usá-lo.

O VRRP provê uma rápida transição do roteador *Backup* para o roteador *Master*, minimizando a interrupção de serviços e incorpora otimizações que reduzem a complexidade do protocolo enquanto garante o controle da transição do *Master* para os típicos cenários operacionais.

Estas otimizações resultam em: menores tempos de execução, mínima alteração no estado dos protocolos (processamento); tipos de mensagens simplificadas; um único roteador que envia as mensagens. O típico cenário operacional é definido por pelo menos dois

roteadores redundantes. O autômato do protocolo (processo) é definido como mostrado na Figura 15.

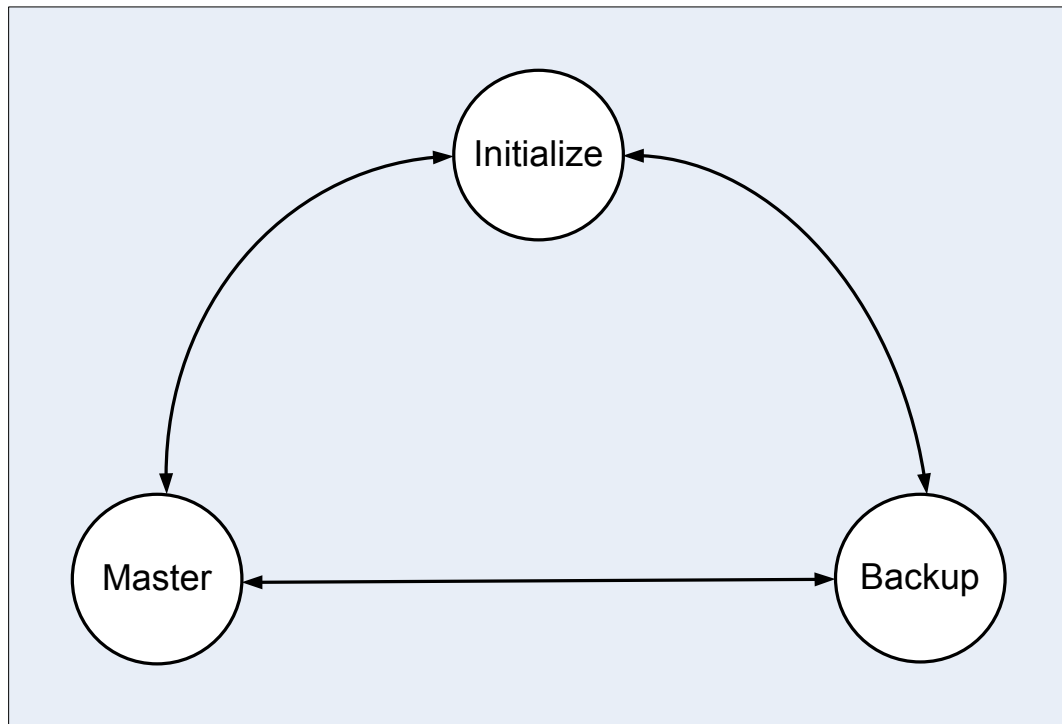


Figura 15: Máquina de Estados VRRP [HINDEN 2004]

As máquinas de estados dos autômatos residentes nos equipamentos começam pelo estado “*Initialize*” e, baseado em uma variável que estabelece a prioridade do elemento, transitam para o estado “*Master*” (elemento que tem a maior prioridade) ou para o estado “*Backup*” (elementos com prioridade inferior à prioridade do elemento *Master*).

Caso o elemento “*Master*” pare de enviar mensagens (indicando sua “saúde”), um dos demais elementos *Backups* será eleito como o novo *Master* da infra-estrutura. Em geral, o elemento de maior prioridade corrente assume o papel de *Master*.

3.3. Definição do problema de *Split Brain* e *No-Brain*

Observe-se que o comportamento descrito na seção anterior funciona a contento se não houver perda ou alteração das mensagens do *Master*. Entretanto, imagine-se que o elemento *Master* envia suas mensagens, mas por algum motivo essas mensagens não cheguem a alguns dos elementos *Backup*. Esses elementos (que não forem atingidos pelas mensagens do *Master*) farão uma eleição para escolha do novo *Master*. Então a infra-estrutura começa a contar com mais de um *Master*.

O problema “*Split-Brain*” ou Cérebro Bi-partido é uma condição onde os equipamentos de um sistema de HA (*Master* e *Backup*) perdem a comunicação entre si e começam a operar autonomamente, pois cada um acredita que o outro realmente se tornou inativo.

Com isso, ambos os equipamentos passam a responder como *Master*. Esse problema é o resultado de uma falha na ação do protocolo de HA utilizado e/ou uma informação incompleta do *heartbeat*. Essa é a principal falha na especificação dos protocolos, pressupor que não haverá perda ou corrupção de *frames* na LAN.

Existem algumas formas de prevenção para o problema *Split-Brain*, tais como:

- Utilização de interfaces físicas não dedicadas, por exemplo, conectadas através de equipamentos L2 (*switches*) com a checagem lógica da condição do link;
- Configuração de interfaces físicas e dedicadas entre os equipamentos para troca de informações de *heartbeat*;
- Utilização de interfaces não dedicadas como caminho secundário quando a interface dedicada falhar.

Ao contrário, o problema “*no-brain*” ou acéfalo é uma condição onde nenhum dos equipamentos assume o papel de *Master*. Isso pode ocorrer quando existem falhas múltiplas no ambiente HA, logo após a eleição de um novo *Master*, onde esses elementos estão conectados. Nessa situação, os equipamentos ficarão em estado inoperante.

3.4. Apresentação dos problemas

Os estudos realizados para este projeto identificaram algumas vulnerabilidades do protocolo VRRP, sendo que a proposição do protocolo parte do princípio que o meio (físico ou enlace) está disponível em todos os momentos e nunca falhará (perda ou corrupção de *frames*). Sabe-se que o meio tem uma taxa de erro que nem sempre é desprezível e isto pode levar o ambiente do VRRP a uma situação de “*Split-Brain*” ou “*No-brain*” .

Quando isso acontece, ocorre a eleição de mais de um *Master* ativo ao mesmo tempo, ocasionando assim uma inconsistência de *gateway* IP na rede e gerando assim a interrupção dos serviços do usuário. Ou, simplesmente, sem *gateway*.

Esse problema foi identificado em um ambiente de *Data Center* de uma renomada Empresa onde haviam equipamentos do tipo *Firewall* baseado em sistema operacional Linux e arquitetura de HA Keepalived *Daemon*. A partir desse evento surgiu a necessidade de um estudo detalhado de todo o ambiente composto por *switches* de camada 2 e os equipamentos de HA com arquitetura aberta ou não.

O resultado desse estudo identificou que o problema apresentado aparece em equipamentos que têm como base o protocolo *multicast*. Em [LOPES FILHO 2008] também foram feitas as análises das arquiteturas dos fabricantes de mercado onde foi identificado que cada um oferece soluções individuais tais como protocolos proprietários ou a obrigatoriedade de utilização de interfaces dedicadas para a garantia do processo de HA.

Devido ao fato do protocolo VRRP usar o protocolo IP (CLTS – *Connectionless Transport Services*) *multicast* para checar a saúde do *Master* ativo, o problema pode acontecer. Também pode ser identificado em outros protocolos como HSRP (*Hot Standby Router Protocol*) [LI, 1998] e CARP (*Common Address Redundancy Protocol*) [CARP, 2003].

Devido a esse tipo de falha, este trabalho propõe uma extensão ao protocolo VRRP a fim de dirimir ou amenizar os impactos ocorridos na arquitetura atual.

4. Proposta de Extensão ao Protocolo VRRP

O capítulo anterior teve o objetivo de mostrar as vulnerabilidades do ambiente de HA baseado na arquitetura Keepalived e no protocolo VRRP. A arquitetura Keepalived tem como uma das funções gerir a saúde da rede utilizando o protocolo VRRP e na proposta apresentada por [LOPES FILHO 2008], essa função foi substituída pelo protocolo SCTP.

O protocolo VRRP tem uma especificação que falha ao considerar que o ambiente de interconexão (meio físico/enlace) é isento de erro. Viu-se no capítulo anterior os fenômenos que esta consideração introduziu em ambientes baseados no protocolo VRRP.

É importante frisar que [LOPES FILHO 2008] pretendia transferir a responsabilidade de verificação da saúde da arquitetura Keepalived para a camada de transporte, supondo-se que isto minimizaria/eliminaria os problemas descritos. Viu-se, entretanto, que, apesar de oferecer uma solução mais elegante e performática, os problemas persistiram. Na realidade, aquela proposição focava mais na eliminação do protocolo VRRP.

4.1. Formalização das Bases da Extensão

Os mecanismos de HA introduzidos na seção 2.1 apresentam diferentes classes de erros residuais, sendo que esses erros, para as tecnologias de enlace atuais – em particular a LAN Ethernet – oferecem a seguinte tabela lógica, considerando os papéis de Originador (que envia um *frame*) e Destinatário (que recebe o *frame*).

Tabela 3: Confiabilidade de Serviços Connectionless

Originador	Receptor
X	X
X	X'
X	-

Observe-se então que ao enviar uma primitiva de enlace (X), esta primitiva pode: chegar ao destino (X); chegar com distorção em bits e não ser detectado pelo controle de erro (X'); e pode não chegar em função de descarte pela camada de enlace do Receptor (-).

A distância de Hamming [TANENBAUM 1996] do controle de erro das tecnologias de LAN introduzidas pelo Comitê IEEE 802.3 [TANENBAUM 1996] garante que até 2 bits de distorção há 100% de chance de detectar o erro, mas acima de 2 bits de erro não há garantia de detecção do mesmo. Considerando a filosofia introduzida pelo Modelo de Referência OSI [TANENBAUM 1996], e seguida por várias arquiteturas – inclusive a arquitetura Internet, isto implica que uma primitiva pode chegar à camada de aplicação e não ser interpretada corretamente. Tudo depende da região atingida pela distorção. O proposto pelo protocolo VRRP, oferece a alta disponibilidade sem levar em consideração as propriedades do meio, tais como distorção e perdas e incluindo também a falha de software ou hardware.

[LOPES FILHO 2008] ilustra bem as vulnerabilidades existentes em alguns protocolos que se propõem a operar diretamente sobre uma LAN. O protocolo VRRP discutido no capítulo 3 é um caso exemplar, pois fora desenvolvido para operar sobre redes locais e, portanto, não estão preparados para suportar por completo os mecanismos de HA devido ao exposto no início deste capítulo. O mesmo aplica-se ao protocolo HSRP e CARP.

Na tentativa de abstrair o meio de comunicação (*provider*), o que facilita a modelagem dos autômatos, as proposições existentes de HA consideram que os módulos da arquitetura (*Master/Slave*) conversam diretamente entre si, isto é, trocam interações através de um canal abstrato que na maioria das vezes são implementadas através de LANs ou comunicações seriais (distância de Hamming igual a 2 e taxa de erro da ordem de 10^{-5}). A Figura 16 a seguir representa o ponto de partida para a modelagem desses mecanismos.

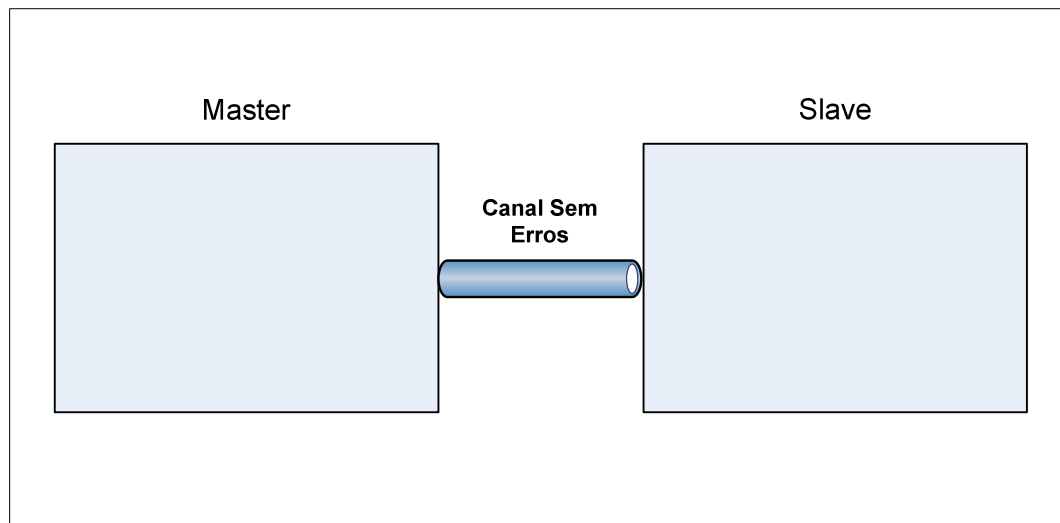


Figura 16: Ponto de conexão entre *Master* e *Slave*

Nesta figura, o módulo *Slave* abstrai as instâncias de equipamentos no estado Backup, uma vez que as comunicações são feitas através de protocolos *multicast*.

Entretanto, sabe-se que as propostas de HA existentes usam a arquitetura de rede, incluindo desde as camadas de transporte até a camada física para estas comunicações. Os estudos desenvolvidos mostraram que havia a necessidade de mudanças no protocolo VRRP e que a utilização de outro protocolo seria necessária para garantir o envio de mensagens de checagem da saúde do sistema (*health-checker*).

Em particular, o uso do protocolo UDP (*User Datagram Protocol*), que é um protocolo não orientado a conexão (CLTS – *Connectionless Transport Services*), justificado pela necessidade de comunicações *multicast*, implica em estender os efeitos das LANs explicitados na Tabela 3 até a camada de transporte. Com um agravante, pois as tecnologias

de LANs (excetuando-se a rede ATM (*Asynchronous Transfer Mode*)) [HÄNDEL 1998] oferecem mecanismos de detecção de erro, o que não acontece com o protocolo UDP.

Isto significa que há o envio de *frames multicast*, mas não há nenhuma garantia de entrega (própria dos serviços CL) e também não há como se assegurar que houve as respectivas recepções. Nossos estudos mostram que essa é a origem dos problemas existentes nas arquiteturas de HA atuais (ver seções 3.3 e 3.4).

Esta proposta introduz uma camada denominada de HA *Provider* que tem a função de modelar o transporte das primitivas de serviços trocadas entre o *Master* e os *Slaves*, para permitir a modelagem formal dos eventos existentes no canal de comunicação.

Isto tudo é justificado pelas necessidades de disponibilidade do processo de HA requerida pelos usuários tais como, sites redundantes com qualquer distância geográfica e menores custos. Estas necessidades foram viabilizadas por soluções baseadas em acesso via Internet ou soluções oferecidas por provedores de serviços que provêm VPN L2/L3 (*Virtual Private Networks*) com tecnologias baseada em redes BGP/MPLS (*Border Gateway Protocol/Multiprotocol Label Switching*) [KOMPELLA 2009] [ROSEN 2006].

É compreensível a utilização de um protocolo com serviços do tipo CL (*connectionless*), uma vez que há a necessidade de comunicação *multicast* (um *Master* e diversos *Slaves*), mesmo que se trate de um protocolo cuja entrega não é garantida, pois o uso de um protocolo orientado à conexão – como é o caso do TCP – com comunicações essencialmente *unicast*, aumentaria significativamente o *overhead* de transmissão e de processamento nas estações.

Este capítulo detalha a arquitetura proposta (seção 4.2), bem como apresenta a especificação formal (seção 4.3) em Rede de Petri, e a descrição do funcionamento da Rede de Petri gerada (seção 4.4). A seção 4.5 mostra a árvore de cobertura e por fim o autômato gerado (seção 4.6).

4.2. A arquitetura proposta

Em alto nível de abstração podemos afirmar que a arquitetura HA proposta é composta de duas camadas:

- A camada de Aplicação: que é constituída de duas categorias de entidades – *Master* e *Slave*;
- A camada *HA Provider*: que modela o canal de comunicação entre as entidades da camada de Aplicação (*Master* e *Slaves*).

É importante ressaltar que o modelo de HA atual considera uma infra-estrutura de comunicação que consiste em uma Entidade *Master* e diversas (N) Entidades *Slaves*.

Esta proposta apresenta os componentes *Master* e *N-Slaves* que são baseados na subcamada HSOL representada pela Figura 12. A camada HSOL é uma subcamada da Camada de Aplicação introduzida na seção 2.3. A Figura 13 apresenta o relacionamento entre as camadas, subcamadas e, particularmente, a camada *HA Provider*.

A camada *HA Provider* foi evidenciada a partir de dados reais coletados em equipamentos que utilizam o protocolo VRRP em seu mecanismo de HA. Esta coleta possibilitou a visualização dos pontos vulneráveis ignorados ou considerados parcialmente pelo protocolo. A Figura 17 apresenta uma visão geral do exposto até o momento.

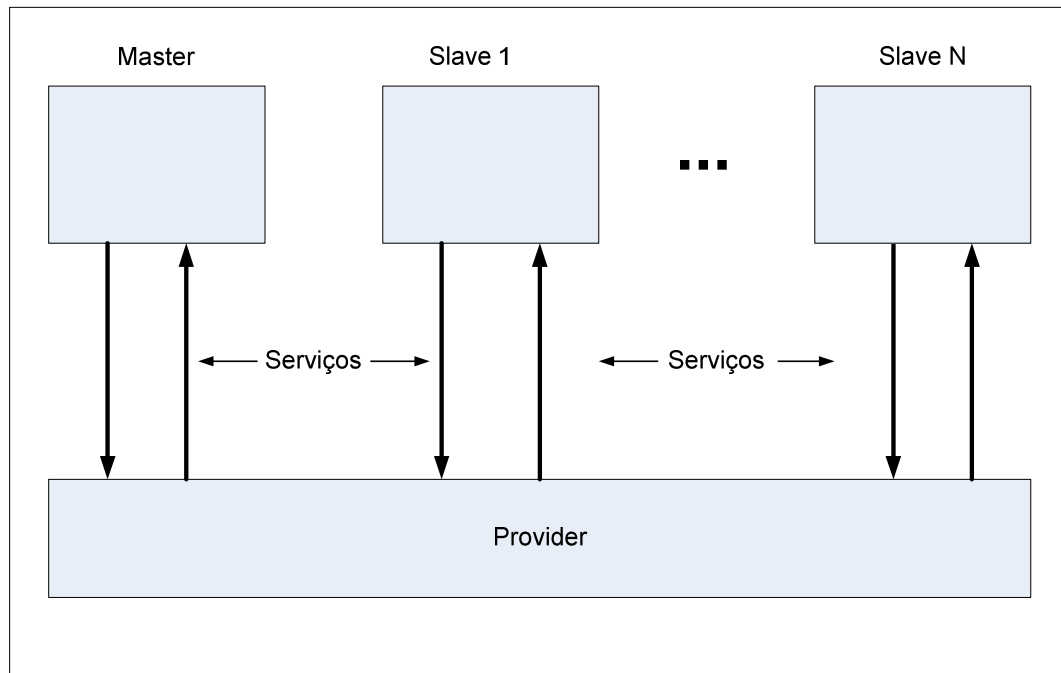


Figura 17: Visão Global da Arquitetura Proposta

Formalmente falando, *Slave* é um estado que o autômato pode alcançar, então os equipamentos ditos *Standby* são aqueles para os quais o autômato que ele hospeda está no estado *Slave*. Desse modo, os equipamentos *Standby* podem ser visto como instância de um equipamento modelo denominado aqui simplesmente de *Standby*.

Portanto, como todas as instâncias do autômato no estado *Slave* apresentam o mesmo comportamento, para fins da especificação formal, a representação da Figura 17 pode ser modelada como contendo dois equipamentos sendo um no estado *Master* e o outro no estado *Slave*, isto é, um apresentando o comportamento de um equipamento ativo e o outro apresentando o comportamento de um equipamento *Standby*. A Figura 18 representa esta abstração.

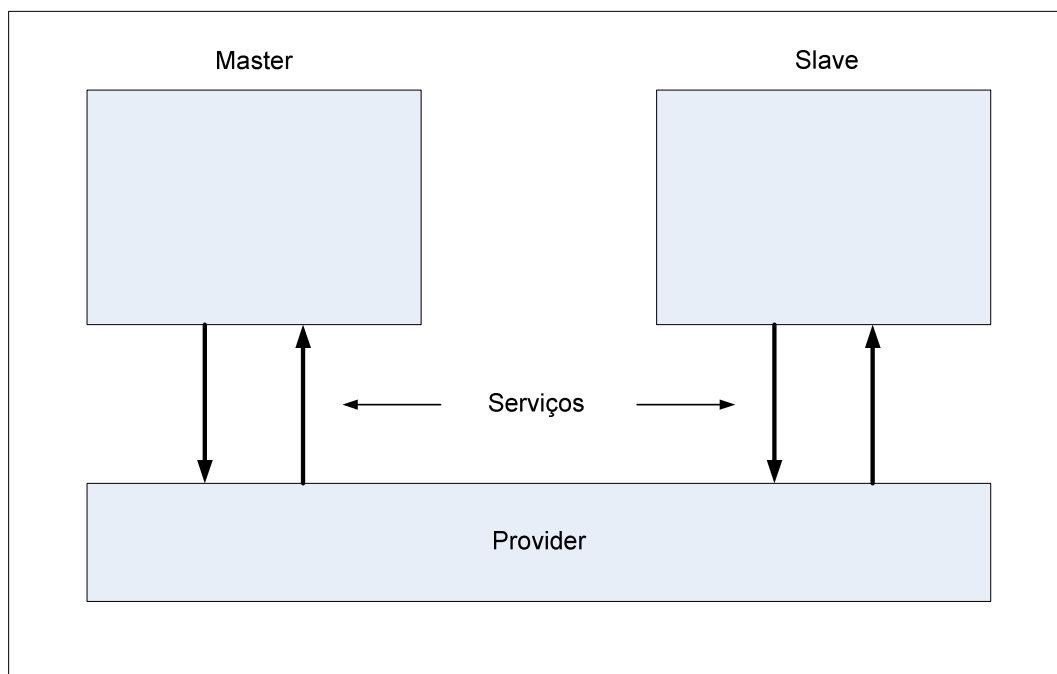


Figura 18: Abstração da Arquitetura Proposta

É importante ressaltar que a eleição do *Master* é feita durante a iniciação da operação e, portanto, a atuação como *Master* ou como *Slave* não é definida previamente. Há um elenco de comunicação na iniciação para definir esses papéis, dado que todos os equipamentos devem estar aptos a exercê-los.

4.2.1. Autômato VRRP

Nos estudos da especificação do VRRP, foi identificado que o diagrama de transições original não trazia os detalhes necessários para a visualização de todos os estados que compõem o autômato (comportamento) deste protocolo. A partir disto, especificamos uma FSM (*Finite State Machine*) na qual foi possível especificar mais detalhadamente seu comportamento.

Para não comprometer a legibilidade da figura, as ações associadas aos eventos foram relacionadas de forma sucinta, restando completa a relação de estados e eventos.

A Figura 19 apresenta o diagrama de estados do protocolo VRRP. Nesse diagrama foi utilizada a notação de CSP (*Communicating Sequential Processes*) [HOARE 1985], sendo que o símbolo “?” indica a recepção de uma primitiva e o símbolo “!” indica o envio de uma primitiva.

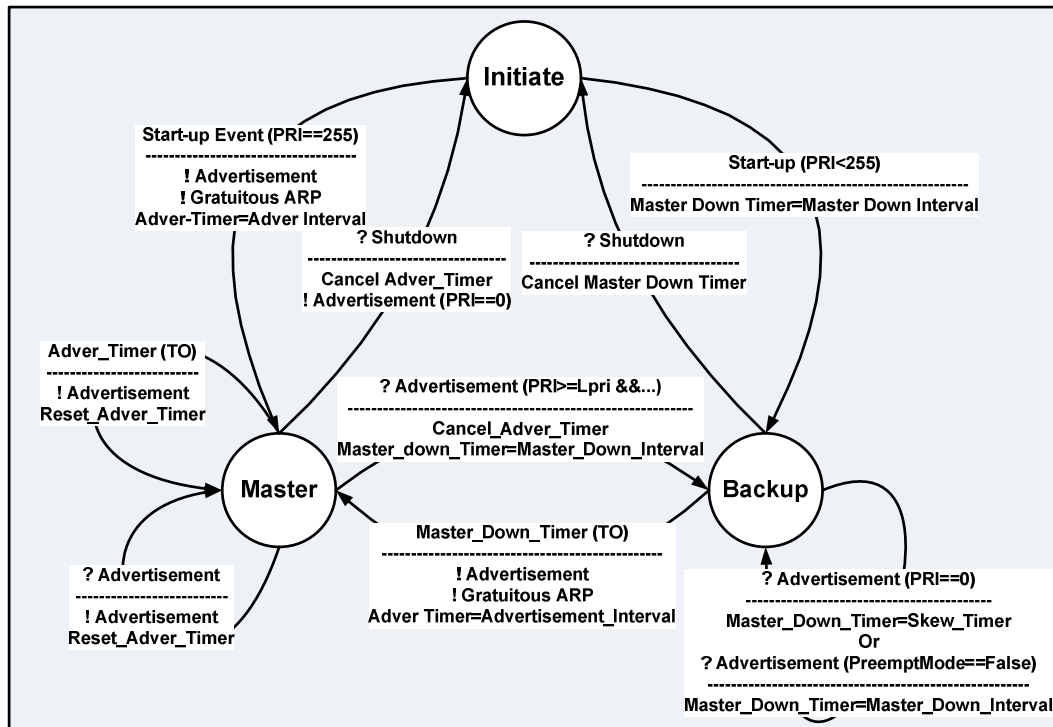


Figura 19: Diagrama de Estados do Protocolo VRRP

Uma análise introdutória deste diagrama mostra que a manutenção dos estados se dá através de uma primitiva básica, que mantém a comunicação entre os equipamentos (Ativo e *Standby*), denominada de “*Advertisement*”. O equipamento Ativo (estado *Master*) envia periodicamente esta primitiva em *multicast*, que ao ser recebida pelos equipamentos *Standby* (estado *Slave*) reinicializam seus temporizadores e permanecem no estado *Slave*.

Entretanto, é interessante notar que a transição do estado *Slave* para o estado *Master* se dá quando há o estouro de um temporizador (*Master_down_Timer*) por não ter recebido a primitiva “*Advertisement*”. Uma visita à Tabela 3 mostra que uma propriedade marcante do tipo de serviço não orientado a conexão (caso dos protocolos das LANs, do protocolo IP e do protocolo UDP) é a possibilidade de não entrega de uma primitiva.

Observe-se que mesmo o *Master* tomando a iniciativa de enviar a primitiva “*Advertisement*” no tempo devido, não significa que ela chegue ao destino e basta que um equipamento *Standby* não receba esta primitiva para apresentar o efeito denominado de Cérebro Bi-partido (seção 3.3). De fato, de acordo com a norma, serão necessários 3 *Advertisements (TimeOut)*, para que o *Slave* tente assumir o papel de *Master*, contudo, essa é uma análise superficial e tem o objetivo momentâneo de mostrar a fragilidade dessa especificação (protocolo).

A compreensão desse diagrama é dificultada pela falta de padronização na nomeação das primitivas. Na seção 4.2.2 será feita uma renomeação das primitivas de acordo com o Modelo de Referência ISO [ITU-T X.200 1994].

4.2.2. Renomeação de serviços existentes no VRRP

De acordo com o Modelo de Referência OSI, as primitivas de serviços podem possuir quatro fases sendo:

- *Request*: a Entidade Usuária Requisitante (*Requester User Entity*) requisita um serviço à camada subjacente (*Provider*);
- *Indication*: a camada subjacente entrega para a Entidade Usuária Respondedora (*Responder User Entity*) da camada superior uma requisição vinda do meio de comunicação;
- *Response*: é uma primitiva de resposta a uma indicação de serviço recebida da camada subjacente que a Entidade Usuária Respondedora envia à camada subjacente;
- *Confirmation*: é o nome que a primitiva de *response* recebe ao chegar à Entidade Usuária Requisitante.

Essa padronização permite que ao analisar um diagrama se perceba facilmente se uma primitiva veio/vai da/para a camada subjacente.

Outro aspecto a ser considerado no modelo de referência OSI são os tipos de serviços:

- Serviços confirmados – Nos serviços confirmados é preciso haver um acordo entre a entidade requisitante e respondedora e neste caso, a respondedora pode aceitar ou rejeitar a requisição do serviço (esse serviço possui as quatro fases);
- Serviços não confirmados – Nos serviços não confirmados ou sem confirmação, não há a necessidade de haver um acordo entre a entidade requisitante e a respondedora (esse serviço possui somente as primitivas *request* e *indication*).

No diagrama de estado do processo VRRP há um agravante. É possível verificar que existem serviços com mesmo nome no diagrama de estados, mas com diferentes funções a serem executadas. Além do aspecto filosófico, esse foi um dos motivos para fazer uma diferenciação das transições vinculando-as ao estado de onde ela é originada e ao sentido da primitiva.

Tabela 4: Matriz de Renomeação de Estados

De/Para	<i>Initiate</i>	<i>Master</i>	<i>Backup</i>
<i>Initiate</i>	-----	<i>Start-up</i> (PRI=255)	<i>Start-up</i> (PRI<255)
<i>Master</i>	<i>Shutdown</i>	<i>Advertisement</i>	<i>Advertisement</i> (PRI>Lpri)
		<i>TimeOut</i> (TO)	<i>Advertisement</i> (PRI=Lpri) && (IP_Address > LocalIP_Address)
<i>Backup</i>	<i>Shutdown</i>	<i>TimeOut</i> (TO) <i>Master_Down_Timer</i>	<i>Advertisement</i> (PRI =0)
			<i>Advertisement</i> (PreemptMode=False)

O processo de renomeação será apresentado a seguir:

- *Start-up* (PRI=255) → *Adver_PRI_255_Request*
 - Neste serviço, o *Master* é iniciado imediatamente e envia *request* de *Advertisement* e *Gratuitous ARP* para a rede, a Figura 20 ilustra a mensagem;

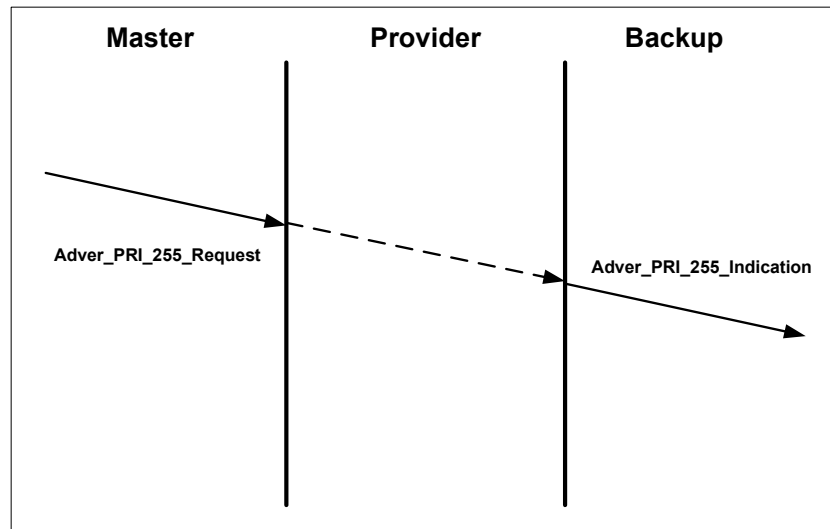


Figura 20: *Adver_PRI_255_Request*

- *Start-up* (PRI<255) → *Startup_Backup*
 - Serviço que inicializará todos os elementos no processo backup;
- *Shutdown* → *Adver_PRI_0_Request*
 - Neste serviço o *Master* avisa aos outros elementos que o mesmo estará inoperante, enviando um *request* com o campo *priority* igual a zero;
- ? *Advertisement* (Priority = 0) → *Adver_PRI_0_Indication*
 - Neste serviço, o processo *Backup* recebe o *indication* vindo do processo *Master* (vide seção anterior), anunciando que o mesmo ficará inoperante, conforme mostrado na Figura 21;

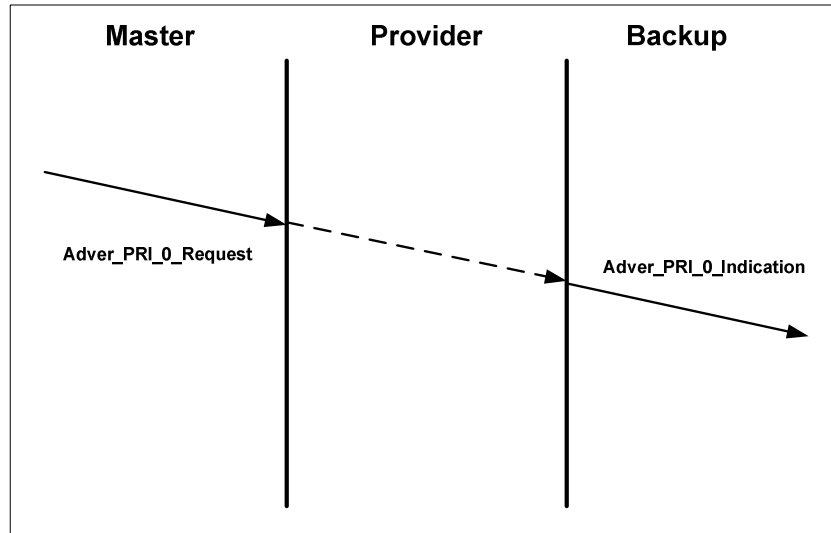


Figura 21: Adver_PRI_0_Request e Adver_PRI_0_Indication

- ! *Advertisement (Priority = Local_Priority)* → Adver_PRI_LocalPRI_Request
 - Neste serviço, o Backup envia o valor do campo *priority* e faz a transição para *Master*,
- ? *Advertisement (Priority > Local_Priority)* ou ? *Advertisement (Priority = Local_Priority e IP_Address > Local_IP_Address)* → Adver_PRI_LocalPRI_Indication
 - Este serviço faz a verificação do valor do campo *priority* recebido pelo *Master* e se a condição for satisfeita, haverá a transição do *Backup* para *Master*, conforme mostrado na Figura 22;

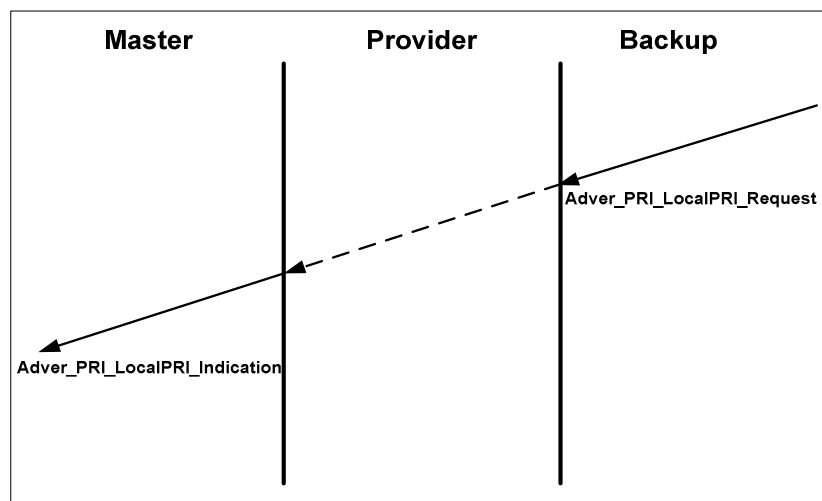


Figura 22: Adver_PRI_LocalPRI_Request

- ? *Advertisement (PreemptMode == False) → Adver_Preempt_False_Indication*
 - Esse serviço verifica se a *priority* recebida é maior que o *priority* local e se for verdadeiro e o campo *PreemptMode* for verdadeiro será permitido a transição de *Master* para *Backup*, caso contrário, o processo de transição do *Backup* para *Master* será proibido, conforme ilustrado na Figura 23;

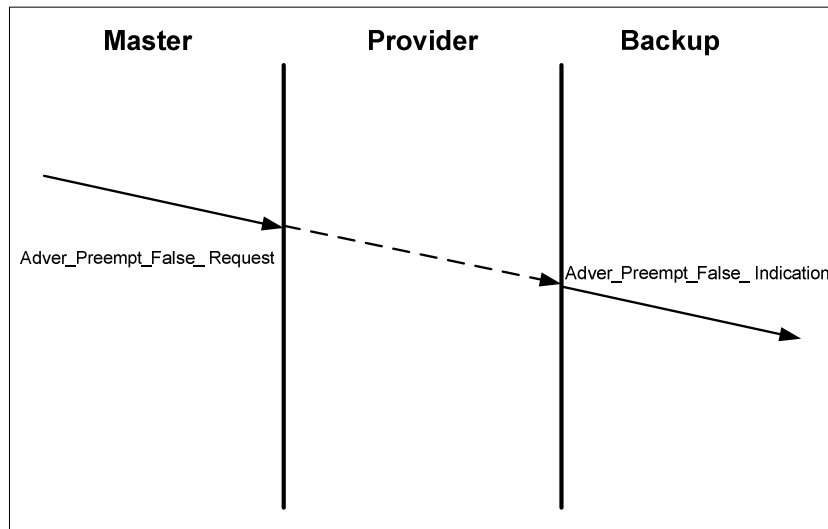


Figura 23: *Adver_Preempt_False_Indication*

- ? *Advertisement (Priority == 0) → Adver_PRI_0_Indication*
 - Neste serviço o *Master* recebe um *advertisement* com o campo *priority* igual a zero, o processo envia um *request advertisement* (*Adver_PRI_LocalPRI_Request*) e reinicia o contador *Adver_Timer*, conforme ilustrado na Figura 24 e Figura 25;

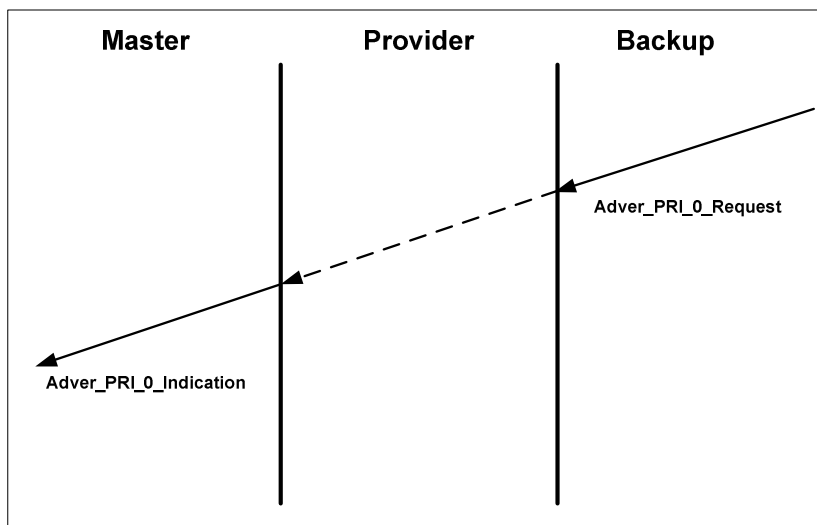


Figura 24: *Adver_PRI_0_Indication*

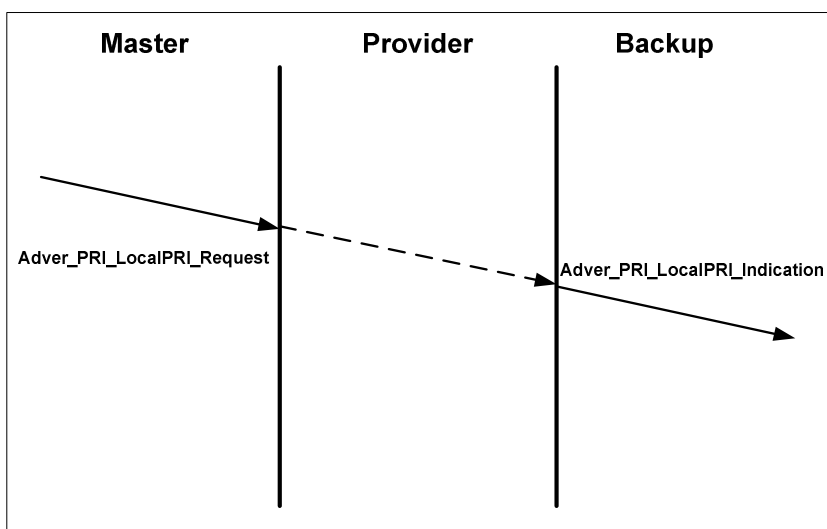


Figura 25: *Adver_PRI_LocalPRI_Request*

4.3. Descrição Formal em Rede de Petri

De acordo com a Figura 18, as comunicações entre entidades *Master* e *Slave* se dão através da camada HA *Provider*. A Figura 26 representa a abstração da camada *Provider* em Rede de Petri.

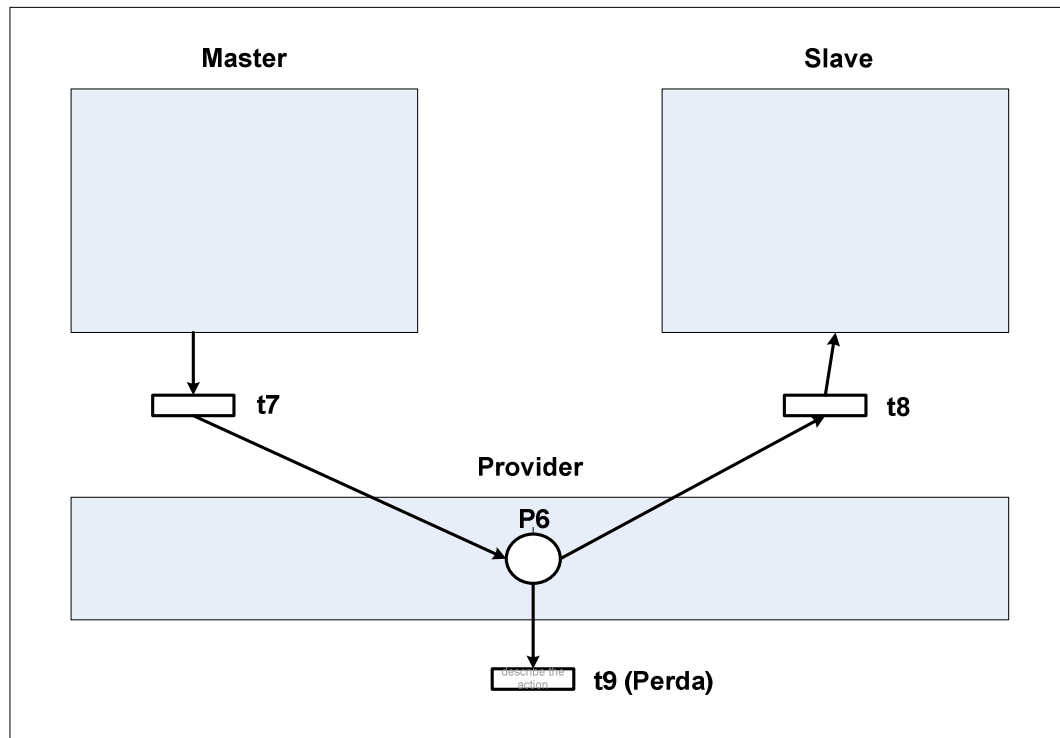


Figura 26: Especificação Rede de Petri para as instâncias *Master/Slave* e *Provider*

Nesta abstração podemos verificar a existência de uma transição “t9” que especifica a perda de uma primitiva e cujo disparo pode significar a falha dos equipamentos que compõem a estrutura do *Provider* ou por problemas no meio, conforme apresentado na seção 4.1.

A proposta deste trabalho visa resolver o problema apresentado, modificando o autômato original do protocolo VRRP.

A eleição do *Master* é feita durante a iniciação da operação, mas, entretanto, a configuração inicial pode ser determinante nesta escolha. Observe-se que o valor do parâmetro Prioridade (*Priority*) determina quem assumirá o papel de Ativo, sendo que os demais assumirão o papel de *Standby*.

De um ponto de vista abstrato, se houver “N” equipamentos no pool de alta disponibilidade, um equipamento será eleito como Ativo (estado *Master*) e os demais “N-1” assumirão o papel de *Standby* (estado *Slave*).

Após esta definição de papéis no início da operação seguir-se-á uma seqüência de comunicações *multicast*, originadas pelo *Master*, destinadas aos equipamentos *Standby* cujas

recepções significarão que eles devem permanecer no estado *Slave* e que tomem a ação de ressetar seus respectivos temporizadores.

O não recebimento dentro do intervalo de tempo configurado de uma primitiva “*Advertisement*” indicando que o *Master* continua ativo, estourando o tempo, significará uma nova eleição de *Master*, com a conseqüente transição de estado.

Observe que a descrição do comportamento em linguagem natural introduz uma série de ambigüidades que podem levar aos efeitos descritos no capítulo 3.

De acordo com o objetivo deste trabalho, a Figura 27 é uma especificação abstrata em Redes de Petri [CARDOSO 1997] da máquina de estados proposta.

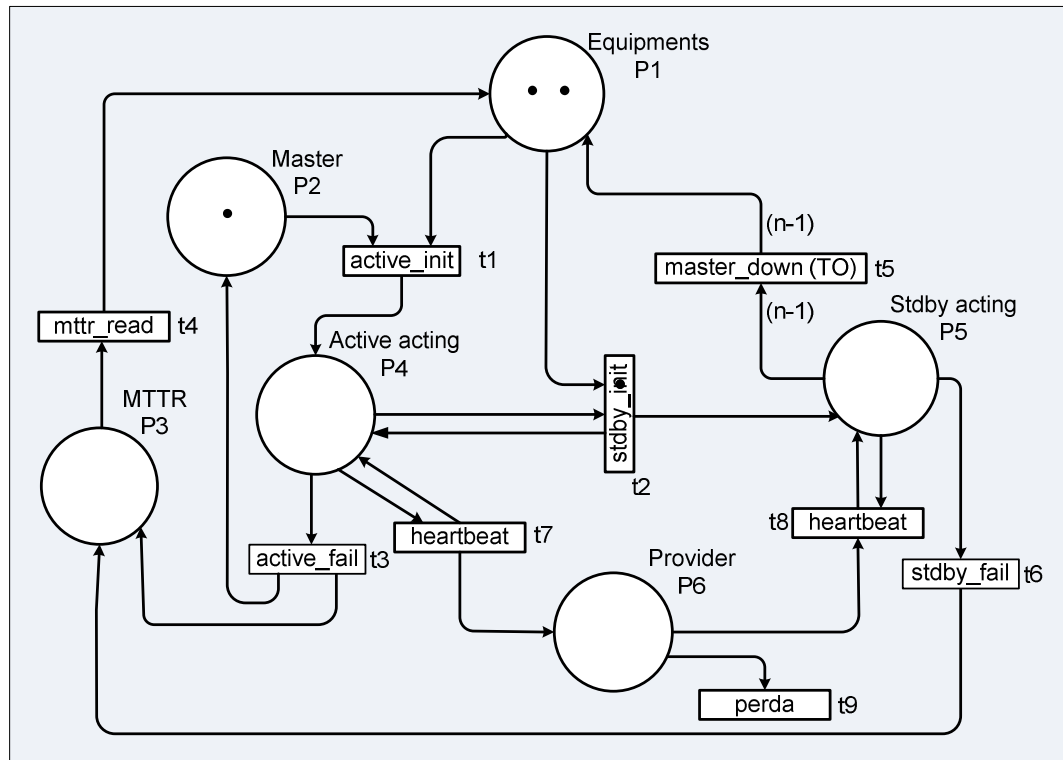


Figura 27: Especificação em Rede de Petri do Autômato Proposto

Esta rede de Petri é definida pelos seguintes lugares (P) e transições (t):

- P1: é um lugar que significa o conjunto de máquinas que fazem parte da HA e cuja marcação significa o número de equipamentos disponíveis para tal, sendo que para o cenário desta dissertação considerar-se-á duas fichas;

- P2: é um lugar que significa que NÃO há um *Master* designado, sendo que as marcações possíveis para ele são zero (*Master* atribuído) ou um (HA sem *Master*);
- P3: é um lugar que significa que os equipamentos estão em processo de manutenção, por isto o uso da sigla MTTR (*Mean Time to Recovery*);
- P4: é o lugar onde a marcação mostra que existe um equipamento em estado *Master*;
- P5: é o lugar onde a marcação mostra a existência de um equipamento em estado *Slave*;
- P6: é o lugar onde a marcação mostra que o *provider* recebeu uma primitiva *heartbeat* do equipamento no estado *Master*;
- t1: é a transição onde a marcação indica que o equipamento passará para o estado *Master*;
- t2: é a transição onde a marcação indica que o equipamento passará para o estado *Slave*;
- t3: é a transição onde a marcação indica que houve uma falha no equipamento *Master* e o mesmo passará do estado *Master* para o estado de manutenção;
- t4: é a transição onde a marcação indica o retorno de um equipamento ao estado disponível;
- t5: é a transição onde a marcação indica que o equipamento *Master* está indisponível devido a *time-out* e o equipamento *Backup* passará para o estado disponível;
- t6: é a transição onde a marcação indica que houve uma falha no equipamento *Backup* e o mesmo passará para o estado de manutenção;

- t7: é a transição onde a marcação indica o envio de *heartbeat* do equipamento *Master* para o *Provider*;
- t8: é a transição onde a marcação indica o envio de *heartbeat* do lugar *Provider* para o equipamento *Backup*;
- t9: é a transição que mostra uma perda de primitiva (*heartbeat*).

A próxima seção introduzirá uma breve descrição para o entendimento da árvore de cobertura (*Covering Tree*), que é o passo intermediário para o levantamento do diagrama de estado autômato.

4.4. Descrição do Funcionamento da Rede de Petri

As definições dos lugares, da evolução das marcações e das transições podem resultar em um diagrama muito interessante que enseja toda a cobertura dos estados do autômato.

O local P1 representa um conjunto de equipamentos que participa da arquitetura HA. Em consonância com a Figura 18, a Rede de Petri apresentada na Figura 27 especifica duas fichas (token), mas é interessante notar que poderiam ser especificadas mais fichas (leia-se equipamentos) que o comportamento seria o mesmo. Isto mostra que a consideração feita sobre a instância de equipamentos Standby elaborada no início deste capítulo e que resultou na representação da Figura 18 estava correta.

O lugar P2 representa a necessidade de se eleger um equipamento *Master* e se houver uma ficha (no máximo uma, dado que soluções de HA especificam apenas uma *Master*) indica que não há equipamento com o estado de *Master* para aquela marcação.

No início da operação apenas a transição “t1” (*active_init*) está sensibilizada e, portanto será a única a ser disparada. O disparo dessa transição significa que um equipamento

foi eleito como *Master*. Como as fichas no lugar P1 significam equipamentos, em princípio, quaisquer dos equipamentos podem ser eleitos como *Master*.

Após o disparo de “t1” há duas transições sensibilizadas (“t2 – *stdy_init*” e “t3 – *active_fail*”) sendo que em regime permanente, a possibilidade de disparo é muito maior para “t2”. Observe-se que esta transição tem a possibilidade de disparar tantas vezes quantas forem as fichas (equipamentos) em P1. Para o cenário desta dissertação, sempre consideraremos 2 fichas em P1 (no estado normal).

Dependendo do contexto (que é um detalhe de implementação), para o cenário deste trabalho, um equipamento é iniciado como *Master* (*Active_init*) ou iniciado como *Slave* (*Stdby_init*).

Após a iniciação, as transições “t3” (*active_fail*), “t5” (*TimeOut*) e “t6” (*stdby_fail*) estão sensibilizadas. Note-se que estas transições somente serão disparadas em caso de falhas de equipamentos ou em caso de falha de comunicação (“t5”). As transições “t7” (*heartbeat*) e “t8” (*heartbeat*) também serão sensibilizadas e serão disparadas de tempo em tempo para checagem da saúde do *Master*.

Se houver o disparo de “t3”, indicando que o equipamento *Master* falhou, então a HA está momentaneamente sem *Master*. Note-se que o disparo “t3” acarretará o estouro no temporizador com o conseqüente disparo de “t5”.

A única opção para que não haja o disparo de “t5” é que o equipamento que falhou volte (disparo de “t4”) da manutenção (MTTR) antes de estourar o *time-out* do equipamento *Standby*. Considerando que houve falha do equipamento no estado *Master*, o mais natural é que um equipamento *Standby* se torna *Master* através de uma sequência de transições (t3->t5) de *Stdby_acting* (*Slave*) para *Active_acting*.

Assim como no caso de falha do equipamento *Master*, o equipamento *Standby* também pode falhar (disparo de “t6”). Nesse caso, a sequência mais provável de disparo será $t6 \rightarrow t4 \rightarrow t2$, o que colocará o equipamento de volta no papel de *Standby*.

O disparo de “t7” acontece quando o equipamento *Master* envia primitivas de *heartbeat* para o lugar P6 (*Provider*). O lugar P6 pode sensibilizar a transição “t8” que envia a primitiva de *heartbeat* para *Standby* ou sensibilizar a transição “t9” que é a perda da primitiva.

Esta descrição pode ser longa e de difícil compreensão, sendo então oferecida uma propriedade da Rede de Petri que é a árvore de cobertura, que será apresentada na próxima seção.

4.5. Árvore de Cobertura

A simulação de um jogador para a rede de Petri, especificada na seção 4.3 e detalhada na seção 4.4, é o passo natural para a validação dos requisitos do projeto. Em particular, duas características eram ambicionadas. Observe-se que o comportamento da marcação no lugar P4 (que significa presença do *Master*) é o ponto fulcral deste trabalho. A presença de mais de um *token* neste lugar significa a ocorrência de “cérebro bi-partido”. A simulação da rede mostrou cabalmente que este lugar é 1-limitado, o que demonstra que foi eliminada a possibilidade de cérebro bi-partido.

Ausência de *token* neste lugar significa que a infra-estrutura de HA está “acéfalo”. Este último caso merece uma consideração adicional. Quando se fala em ausência de cérebro, há de se considerar que pequenos intervalos de tempos são aceitáveis. Desse modo, a ausência de cérebro é vinculada à taxa de chegada de requisições.

Não se pode classificar como ausência de cérebro intervalos de tempos pequenos nos quais não há um equipamento no papel de *Master*, mas sim a ausência de *Master* por longos períodos de tempo.

A árvore de cobertura apresentada na Figura 28 mostra cabalmente que estes fenômenos não mais ocorrem.

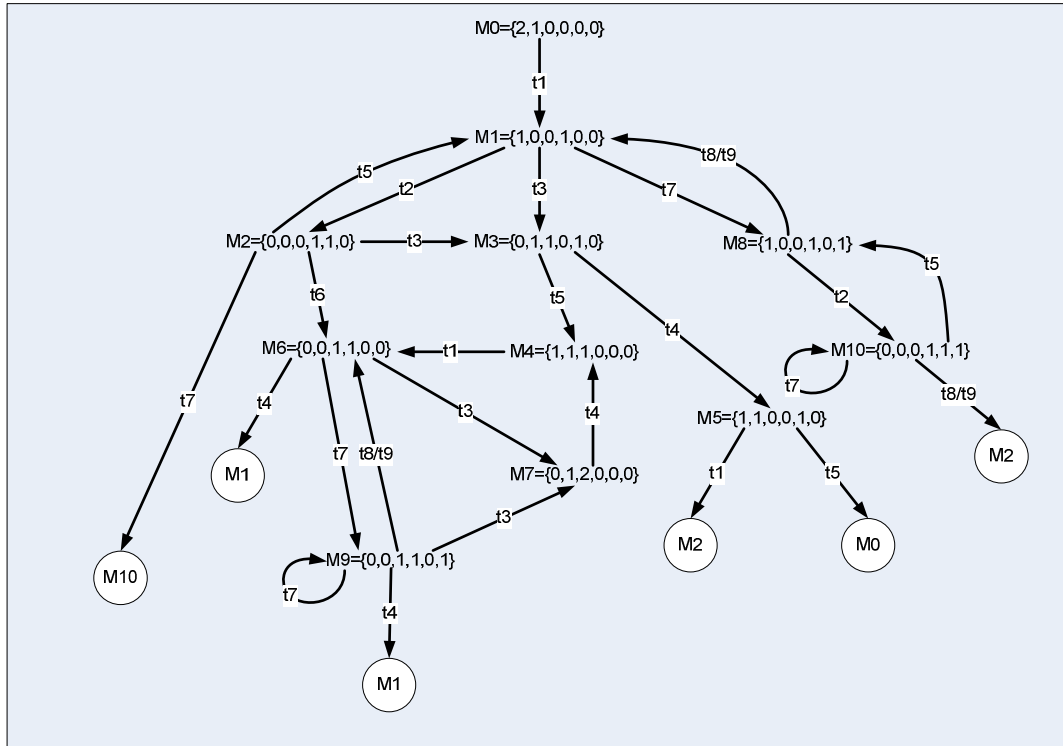


Figura 28: Árvore de Cobertura

Correndo o risco de ser redundante, mas como um último olhar sobre a árvore de cobertura, é possível notar que o fenômeno da “ausência de cérebro” somente aparecerá se não houver equipamentos disponíveis para assumir o papel de *Master*, caso ele venha a falhar.

A análise das marcações para o lugar P4 é um dos aspectos importantes mencionados no início desta seção. O segundo aspecto, e que será abordado com maior profundidade na seção 4.6, é que não há ramo da árvore que não conduza à marcação inicial. Este aspecto mostra que se trata de uma especificação que sempre poderá atingir seu estado inicial.

4.6. Autômato do HA proposto

A partir da Rede de Petri e da árvore de cobertura, considerando que as marcações são estados da rede de Petri, temos condições de gerar o autômato proposto que é mostrado pela Figura 29.

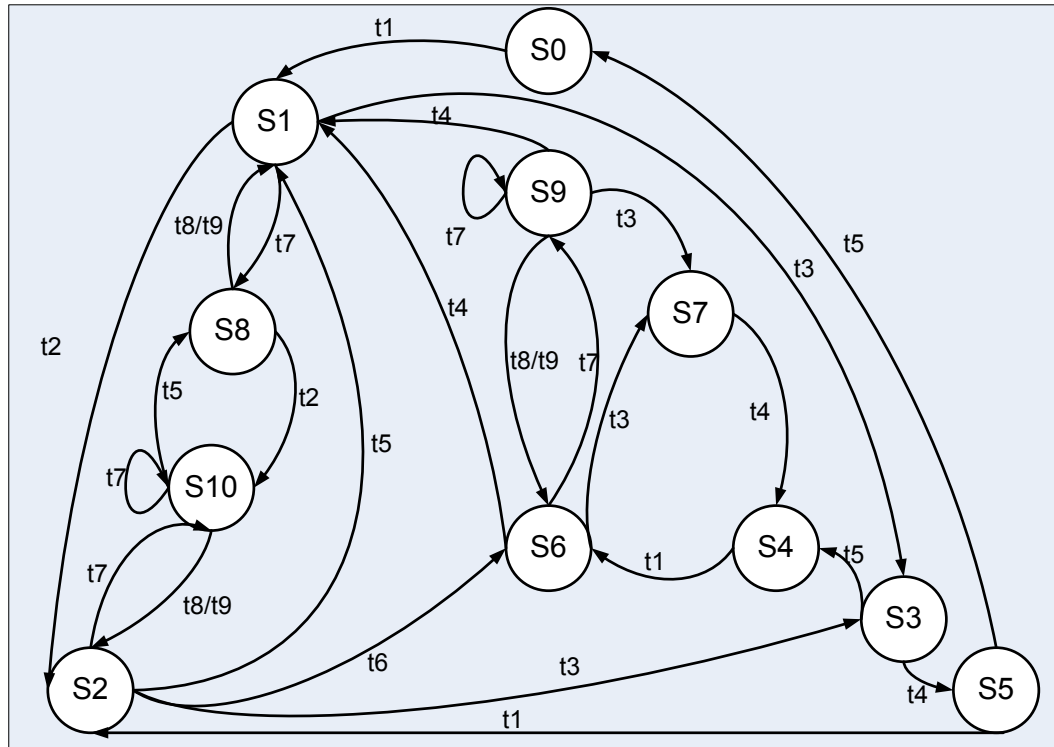


Figura 29: Autômato HA Proposto

A correlação entre as marcações (M) da árvore de cobertura e os estados (S) são descritos na Tabela 5.

Tabela 5: Correlação Marcação (M) e Estados (S)

Marcação	Estados	Significado
M0	S0	estado_inicial
M1	S1	estado_atribuição_stdby
M2	S2	estado_master_stdby
M3	S3	<i>master_manutenção</i>
M4	S4	<i>master_down</i>
M5	S5	stdby
M6	S6	stdby_manutenção
M7	S7	todos_manutenção
M8	S8	estado_atribuição_stdby
M9	S9	stdby_manutenção
M10	S10	<i>Master_health_check</i>

As transições e seus respectivos eventos são mostrados na Tabela 6.

Tabela 6: Relação Transição/Evento

Transição	Evento
t1	<i>active_init</i>
t2	<i>stdby_init</i>
t3	<i>active_fail</i>
t4	<i>mttr_read</i>
t5	<i>master_down</i>
t6	<i>stdby_fail</i>
t7	<i>Heartbeat</i>
t8	<i>Heartbeat</i>
t9	Perda

O estado S0 representa o estado inicial do autômato, onde podem existir equipamentos esperando para se tornar *Master*.

No estado S1, temos a situação do *Master* já designado com um elemento esperando para ser *Standby*.

O estado S2 representa o estado que existe um *Master* designado e outro equipamento designado como *Standby*. A partir desse estado podemos ter a situação onde exista a falha do *Master* (estado S3) ou a falha do *Standby* (estado S6). Nesse estado também ocorre a troca de mensagens *heartbeat* através do estado S10.

O estado S10 representa o estado onde o *Master* envia mensagem *heartbeat* para o *Standby* utilizando-se do *Provider*. A mensagem pode ser entregue ao *Standby* ou pode se perder. Se houver a perda da mensagem *heartbeat*, o *Standby* voltará para o estado de

equipamento disponível (estado S8) disparado pela transição “t5” de *time-out* (TO), passando para o estado S1 e conseqüentemente S3 com a falha do *Master*. Isto garantirá que mesmo existindo a perda de mensagens *heartbeat*, não haverá a ocorrência de dois *Master*, resolvendo assim o fenômeno cérebro bi-partido.

No estado S6 temos um equipamento em manutenção e um *Master* designado, onde pode ocorrer o retorno do equipamento *Backup* como disponível (estado S1) ou haver uma falha do *Master* (estado S7). Nesse caso, podemos ter o envio de mensagens *heartbeat* para o *Provider* (estado S9), mesmo que não exista *Backup*.

A ocorrência do estado S7 é uma situação onde todos os equipamentos estão em manutenção, ou seja, não há *Master* disponível no sistema, ocorrendo o fenômeno acéfalo. Esse é um momento de total catástrofe que pode ser provocado por fatores como: energia, falhas de hardware ou erros de implementação.

A proposta foi modelada de forma a estar isenta do problema apresentado pelo protocolo VRRP. Isto é possível verificar que o autômato resultante que tem uma máquina de estados finita e dispõe de boas propriedades, como:

- Reiniciável, pois, independentemente do estado, sempre há um conjunto de transições que levam ao estado inicial (marcação S0);
- K-Limitada, pois as marcações em todos os ramos nunca ultrapassam um valor máximo, para esta especificação K=2, significando que não apresenta efeitos colaterais como vazamento de memória (*memory leak*), criação infinita de processos ou *threads*, etc.;
- Ausência de *Deadlock*, pois não apresenta nenhum ramo a partir do qual não seja possível evoluir;
- Ausência de *Livelock*, pois não apresenta nenhuma seção da árvore que enseje um laço sem saída.

Observe-se que, de um ponto de vista abstrato, a única região preocupante com relação a este último item (*Livelock*) é a sequência de transições t1, t3 e t4, que formam um laço, mas que quando analisada tendo em vista o seu significado real, trata-se de uma situação muito pouco provável, isto é, é uma sequência de atribuição de *Master*, seguida de uma transição de falha e seguida pela volta do equipamento (do mesmo equipamento que deu causa à sequência) da manutenção e, portanto ao estado operacional. Não se configura essencialmente um *livelock*. Então, se isto ocorrer, embora isto demonstre uma falha, ele remete a uma boa sequência.

A sequência de transições t7 e t8/t9 mostra o envio ou perda da mensagem *heartbeat* do equipamento *Master* para o equipamento *Backup*.

O autômato proposto neste trabalho é uma extensão que endereça o fenômeno de cérebro partido (*split-brain*), pois nunca há mais de um equipamento no papel de *Master* e acéfalo (*no-brain*), pois sempre há pelo menos um equipamento no papel de *Master*, detectado no processo do protocolo VRRP original.

Ao comparar-se o autômato apresentado nesta seção (10 estados) àquele apresentado na seção 4.2.1 (3 estados), é visível a diferença entre os seus estados e comportamentos. É importante destacar a importância do uso de formalismos nestas validações, pois, do ponto de vista da descrição daquele autômato, não era possível perceber o fenômenos que foram o ponto fulcral para este trabalho.

Somente após a experiência adquirida em [LOPES FILHO 2008] foi possível perceber que, sem a aplicação de técnicas de descrição formais, não seria possível chegar-se às conclusões que este trabalho enseja.

5. Conclusão

O avanço tecnológico de hardware e software proporcionou que sistemas corporativos antes somente utilizados em ambientes internos, pudessem ser acessíveis também via Internet. Esse avanço permitiu que houvesse um crescimento explosivo de usuários com acesso à banda larga, via dispositivos móveis como telefones celulares e dispositivos pessoais com acesso a redes 2G e 3G.

Além disso, a competitividade entre provedores de serviços fez com que os usuários ficassem mais exigentes. Sendo assim, a questão de disponibilidade desses sistemas tornou-se um ponto crítico para os negócios. As soluções de alta disponibilidade existentes hoje contemplam disponibilidades próximas de 100%, porém, ainda temos o risco de que algum desses elementos que compõem a solução falhe.

Este trabalho focou nos protocolos da camada de enlace e de transporte que suportam alguns desses sistemas de HA, pois havia evidências de que considerações incompletas para esses níveis acarretariam na indisponibilidade da solução. O estudo realizado com os protocolos VRRP, *multicast* e arquitetura de HA como Keepalived nos permitiu identificar as respectivas vulnerabilidades se utilizada uma LAN Ethernet como meio para suas primitivas.

O protocolo SCTP foi utilizado por [LOPES FILHO 2008] como solução para os problemas apresentados nas soluções de *firewall* e IPS. Porém, verificou-se que os problemas apresentados foram parcialmente endereçados. Naquela ocasião acreditava-se que os problemas eram devidos ao uso de protocolo de transporte não orientado a conexão (UDP), que suportava endereçamento *multicast*, mas não garantia a entrega das primitivas às entidades de transporte parceiras. Isto era parte do problema e o trabalho mostrou-se importante para fundamentar as análises.

Com isso, as pesquisas e os estudos aprofundaram-se para a identificação do problema que quedava parcialmente resolvido. O estudo do protocolo VRRP foi decisivo na definição desta proposição e verificou-se que o principal equívoco da especificação era a suposição de que o meio se comportava como um canal sem erros e que todas as primitivas enviadas pelo equipamento no estado *Master* chegavam aos equipamentos no estado *Backup*.

Sendo assim, identificou-se que seria necessário introduzir uma nova camada chamada de “HA *Provider*”, cuja camada modelasse um canal de comunicação real (com perdas e distorções). Foi utilizado um método formal para a modelagem de protocolos, capaz de especificar e validar seu funcionamento. Esta nova arquitetura foi modelada em Redes de Petri e, a árvore de cobertura e, a máquina de estados dela derivada, incluindo a extensão objeto do presente trabalho, mostrou que os problemas de cérebro bi-partido e acéfalo detectados no autômato original do protocolo VRRP foram resolvidos.

Até onde a pesquisa pode sondar neste período, embora fossem conhecidos os problemas do protocolo VRRP, não houve até o presente momento trabalho similar que diagnosticasse e classificasse os fenômenos que assolam as arquiteturas de HA, como feito neste trabalho.

Certamente, o trabalho deixa claro que a utilização de métodos formais para modelagem de protocolos é de suma importância, pois através dela é possível provar que a má formação do autômato original fora resolvido. Há uma crença no grupo que o problema de cérebro bipartido é muito mais corriqueiro do que se imagina. Em geral, módulos de uma aplicação distribuída são projetados ou desenvolvidos por equipes distintas e a falta de formalismo nas especificações podem ser a chave para a ocorrência de problemas de implementação.

Como fruto das pesquisas realizadas, foram publicados dois artigos “An IMS Control Layer PR-SCTP Based Network”, aceito e publicado no ICNS 2008 (*Fouth International*

Conference on Networking and Services), pp 61-66, realizado em Gosier, Guadalupe no período de 16 a 21 de março de 2008 e o artigo “*A High Availability Firewall Model Based on SCTP Protocol*”, aceito e publicado no ICSNC 2008 (*Third International Conference on Systems and Networks Communications*), pp 202-207, realizado em Sliema, Malta no período de 26 a 31 de outubro de 2008.

O projeto e o desenvolvimento das extensões propostas neste trabalho são objeto de estudo de trabalhos futuros, sendo que a maturidade adquirida leva o grupo a vislumbrar a possibilidade de se desenvolver uma camada de sessão que seja um super conjunto daquela definida pelo Modelo de Referência OSI.

Uma proposta de implementação sobre ambientes de HA baseado em *cluster* de servidores *Linux* é vislumbrado pelo grupo, bem como os protocolos apresentados serão base para novas propostas de trabalho.

Vê-se ainda a possibilidade, ou até necessidade, de imbricar o corrente tópico por uma necessidade corrente nos sistemas ubíquos que é a escalabilidade de sistemas. Considerando que para se alcançar alta disponibilidade é necessária uma infra-estrutura constituída de vários equipamentos, então disponibilizar todas as instâncias para uso concomitante passa a ser algo natural oferecendo escalabilidade.

6. Referências Bibliográficas

- ARMITAGE, G. **Quality of Service in IP Networks: Foundations for a Multi-Service Internet**. 1. ed. Macmillan Technical Publishing/SAMS, 2000. 336 p.
- CAIN, B.; DEERING, S.; KOUVELAS, I.; FENNER, B.; THYAGARAJAN, A. Internet Group Management Protocol, Version 3. **RFC 3376**, 2002.
- CAMERON, R.; WOODBERG, B.; MADWACHAR, K. M.; SWARM, M.; WYLER, N. R.; ALBERS, M.; BONNELL, R. **Configuring Juniper Networks NetScreen & SSG Firewalls**. 1. ed. Syngress Publishing, Inc. 2007. 512 p.
- CANTELLI, L. H. **Estudo, Implantação e Análise Qualitativa de um novo Protocolo *Multi-homing* e *Multi-stream* da Camada de Transporte**. 2003. 93 f. Dissertação (Mestrado em Ciência da Computação)-Universidade Federal de Uberlândia, Uberlândia, 2003.
- CARDOSO, J.; VALETTE, R. **Redes de Petri**. 1. ed. Florianópolis: Editora da UFSC, 1997. 212 p.
- CARP. **Common Address Redundancy Protocol**. Disponível em <http://www.openbsd.org/faq/faq6.html#CARP>. Acesso em: 20 jul. 2009.
- DARPA Internet Protocol – IP. **RFC 791**, 1981.
- DARPA Transmission Control Protocol DARPA Internet Program Protocol Specification. **RFC 793**, 1981.
- FERGUSON, P.; SENIE, D. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. **RFC 2827**, 2000.
- FIELDING, J.; GETTYS, J.; MOGUL, J.; FRYSTYK, H.; MASINTER, L.; LEACH, P.; BERNERS-LEE, T. Hypertext Transfer Protocol HTTP/1.1. **RFC 2616**, 1999.

- HÄNDEL, R.; HUBER, M.; N., SCHRÖDER, S. **ATM Networks Concepts, Protocols, Applications**. 3. ed. Addison-Wesley. 1998. 327 p.
- HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. An Architecture for Describing SNMP Management Frameworks. **RFC 2571**, 1999.
- HINDEN, R. Virtual Router Redundancy Protocol (VRRP). **RFC 3768**, 2004.
- HOARE, C. A. R. **Communicating Sequential Processes**. 1. ed. Prentice Hall International. 1985. 238 p.
- HOLZMANN, G., J. **Design And Validation Of Computer Protocols**. 1. ed. Prentice Hall. 1991. 500 p.
- ITU-T Recommendation. Open Systems Interconnection – Basic Reference Model: The Basic Model. **ITU-T X.200**, 1994.
- KEEPALIVED **HealthChecking for LVS & High Availability**. Disponível em <http://www.keepalived.org/>. Acesso em: 20 de jan. 2008.
- KOMPELLA, K.; KOTHARI, B.; CHERUKURI, R. Layer 2 Virtual Private Networks Using BGP for Auto-discovery and Signaling. **Draft-kompella-l2vpn-l2vpn-03**, 2009.
- LI, T.; COLE, C.; MORTON, P.; LI, D. Cisco Hot Standby Router Protocol (HSRP). **RFC 2281**, 1998.
- LOPES FILHO, E. **Arquitetura de Alta Disponibilidade para Firewall e IPS Baseada em SCTP**. 2008. 135 f. Dissertação (Mestrado em Ciência da Computação)-Universidade Federal de Uberlândia, Uberlândia, 2008.
- LVS LVS - **Linux Virtual Server**. Disponível em <http://www.linuxvirtualserver.org/>. Acesso em: 11 de ago. 2009.
- ONG, L. et al. Framework Architecture for Signaling Transport. **RFC 2719**, 1999.

- Pereira, J. H. de S. **Proposta de Arquitetura de um Media Gateway em uma Rede de Próxima Geração**. 2004, Dissertação (Mestrado em Ciência da Computação)- Universidade Federal de Uberlândia, Uberlândia, 2004.
- POSTEL, J. User Datagram Protocol – UDP. **RFC 768**, 1980.
- ROSEN, E.; REKTER, Y. BGP/MPLS IP Virtual Private Networks (VPNs). **RFC 4364**, 2006.
- STEWART, R.; RAMALHO, M.; XIE, Q.; TUOXEN, M.; CONRAD, P. SCTP Partial Reliability Extension. **RFC 3758**, 2004.
- STEWART, R.; XIE, Q.; MORNEAULT, K.; SHARP, C.; SCHWARZBAUER, H.; TAYLOR, T.; RYTINA, I.; KALLA, M.; ZHANG, L.; PAXSON, V. SCTP: Stream Control Transmission Protocol. **RFC 2960**, 2000.
- TANENBAUM, A. S. **Computer Networks**. Third. ed. Prentice Hall. 1996. 814 p.
- YLONEN, T.; LONVICK, C. The Secure Shell (SSH) Protocol Architecture. **RFC 4251**, 2006.