UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ESPECIFICAÇÃO DE SERVIÇO E SUPOSIÇÕES SOBRE O AMBIENTE PARA UM PROTOCOLO DE ALTA DISPONIBILIDADE

JOÃO EURÍPEDES PEREIRA JÚNIOR

Uberlândia - Minas Gerais
2010

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO EURÍPEDES PEREIRA JÚNIOR

ESPECIFICAÇÃO DE SERVIÇO E SUPOSIÇÕES SOBRE O AMBIENTE PARA UM PROTOCOLO DE ALTA DISPONIBILIDADE

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Redes de Computadores.

Orientador:
Prof. Dr. Pedro Frosi Rosa

Co-orientador:
Prof. Dr. Renan Gonçalves Cattelan

Uberlândia, Minas Gerais
2010

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Especificação de Serviço e Suposições sobre o Ambiente para um protocolo de Alta Disponibilidade**" por **João Eurípedes Pereira Júnior** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 29 de Setembro de 2010

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

Co-orientador: _____
Prof. Dr. Renan Gonçalves Cattelan
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Prof. Dr. Sérgio Takeo Kofuji
Universidade de São Paulo

_____
Prof. Dr. Luís Fernando Faina
Universidade Federal de Uberlândia

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: Setembro de 2010

Autor:	**João Eurípedes Pereira Júnior**
Título:	**Especificação de Serviço e Suposições sobre o Ambiente para um protocolo de Alta Disponibilidade**
Faculdade:	**Faculdade de Computação**
Grau:	**Mestrado**

_____

Autor

Dedicatória

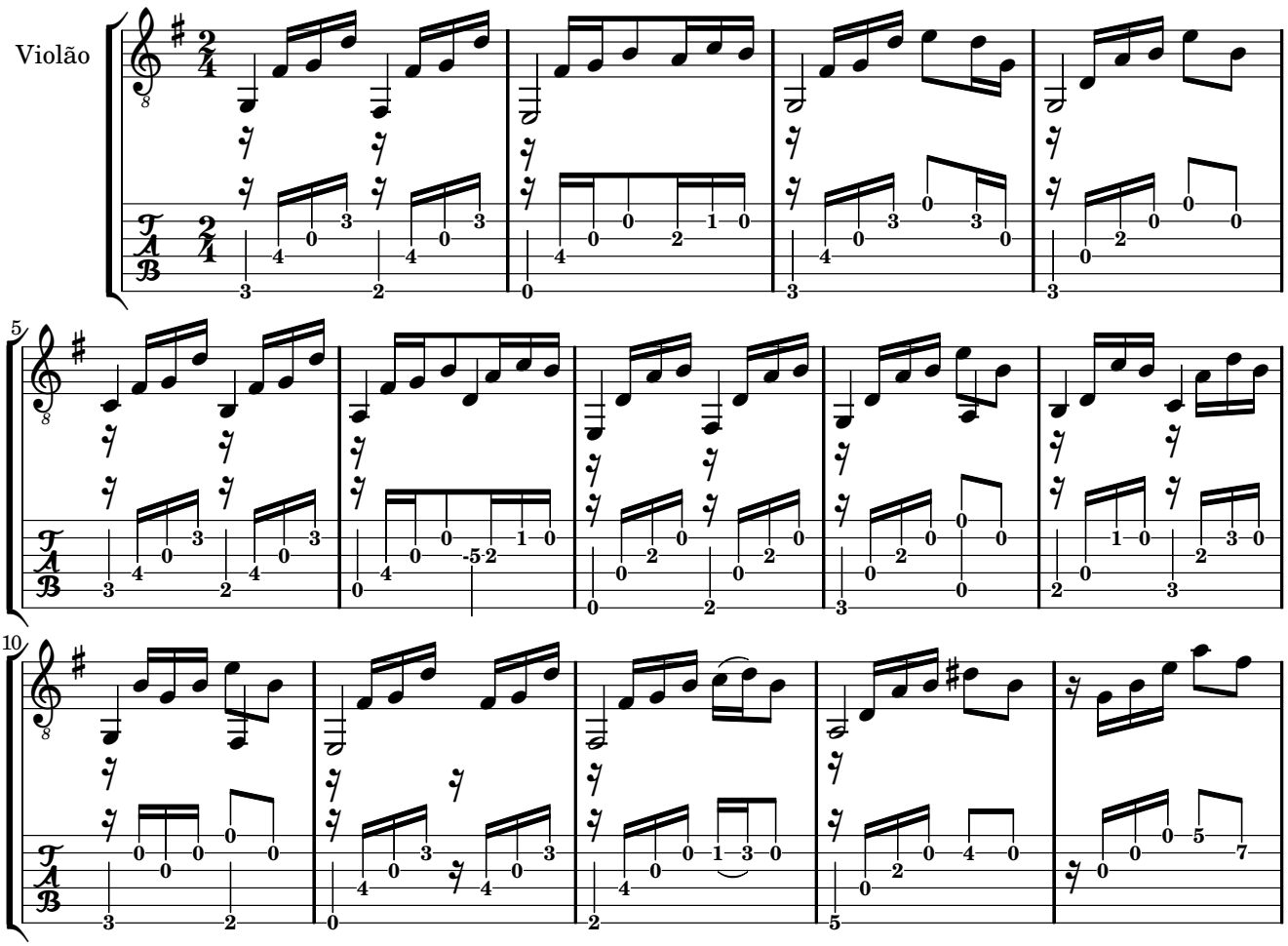
Aos meus pais João e Maria José, que emprestam a esta obra a visão de futuro e o desejo de progresso. À minha namorada Lorenna, pela compreensão e afeto.

Agradecimentos

Agradeço a Deus por me permitir co-criar nesta singela obra e por poder fazer tantos outros agradecimentos. Pelo amparo e apoio da minha família, sem os quais seria impossível iniciar esse trabalho. Pelo incentivo dos meus amigos, sem o qual seria improvável terminá-lo, e pela sinceridade deles, à qual devo o bom senso que há nesta obra. Pelo esforço — e principalmente pela paciência — do meu orientador, Prof. Pedro Frosi Rosa PhD, pela presteza e o desvelo do meu co-orientador, Prof. Renan Gonçalves Cattelan PhD, e pela didática dos meus professores, que me guiaram e moldaram durante essa jornada. Pelas oportunidades e confiança que a instituição me depositou, aliviando o peso de minhas passadas rumo ao conhecimento. A todos que direta ou indiretamente me ajudaram neste trabalho. Muito obrigado.

O Mar e o Lago

Gilberto Gil, 1997



Resumo

Esta dissertação descreve uma arquitetura de alta disponibilidade baseada no protocolo VRRP e as condições de acéfalo e cérebro partido, explicando porque essas condições são consideradas problema. Este documento cita propostas de melhoria em protocolos de alta disponibilidade que visam resolver esses problemas e apresenta um estudo comparativo entre os protocolos VRRP, CARP e Paxos, mostrando que o caminho mais curto para uma solução é um novo protocolo de alta disponibilidade. Esta dissertação tem entre seus objetivos servir como ponto de partida para a criação de uma base de conhecimento para o projeto de novos protocolos na área de alta disponibilidade, e apresenta suposições a respeito do ambiente onde um serviço de alta disponibilidade deve operar e como os protocolos de alta disponibilidade podem resolver os desafios que surgem nesses ambientes. Estas suposições levam ao objetivo principal desta dissertação, uma especificação de serviço de alta disponibilidade em alto nível de abstração, que é apresentada na forma de um modelo, primitivas, parâmetros, valores e pontos de acesso. Esta especificação de serviço deve ser combinada com um autômato, vocabulário e formato das mensagens, que culminará no novo protocolo de alta disponibilidade.

Palavras chave: especificação de serviço, projeto de protocolo, acéfalo, cérebro partido, alta disponibilidade.

Abstract

This dissertation describes a high availability architecture based on the VRRP protocol and the conditions brainless and split-brain, explaining why they are considered problem. This document cites proposals for improvements in high availability protocols that aim to solve these problems and presents a comparative study between protocols VRRP, CARP and Paxos, showing that the shortest path to a solution is a new protocol for high availability. This dissertation has among its objectives to serve as a starting point for creating a knowledge base for the design of new protocols in the area of high availability, and presents assumptions about the environment where a high availability service should operate and how high availability protocols can solve the challenges that arise in these environments. These assumptions lead to the main objective of this dissertation, a high availability service specification at a high level of abstraction, which is presented as a model, primitives, parameters, values, and access points. This service specification should be combined with an automaton, vocabulary and format of the messages, which will culminate in the new protocol for high availability.

Keywords: service specification, protocol design, brainless, split-brain, high availability.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ACID	Atomicity, Consistency, Isolation, Durability
ASIC	Application-Specific Integrated Circuit
BASE	Basically Available, Soft State, Eventually Consistent
CAP	Consistency, Availability, Partition Tolerance
CARP	Common Address Redundancy Protocol
CRC	Cyclic Redundancy Check
CSMA-CD	Carrier Sense Multiple Access With Collision Detection
DHCP	Dynamic Host Configuration Protocol
DIX	DEC (Digital), Intel and Xerox
DoS	Denial of Service
ECC	Error Correction Codes
FCFS	First Come First Served
FCS	Frame Check Sequence
FIFO	First In First Out
GBLP	Gateway Load Balancing Protocol
HALS	High Available Link Service
HSRP	Hot Standby Router Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IPS	Intrusion Prevention System
LAN	Local Area Network
LLC	Logical Link Control
MAC	Medium Access Control
MSDU	MAC Service Data Unit
MTBF	Mean Time Between Failures
OSI	Open Systems Interconnection
P&D	Pesquisa e Desenvolvimento
PPPoE	Point-to-Point Protocol over Ethernet
QoS	Quality of Service
SAP	Service Access Point

SCTP	Stream Control Transmission Protocol
SNA	Shared Nothing Architecture
SPoF	Single Point of Failure
SSL	Secure Sockets Layer
SWS	Silly Window Syndrome
TCP	Transmission Control Protocol
TI	Tecnologia da Informação
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
VRID	Virtual Router Identifier
VRRP	Virtual Router Redundancy Protocol
WAN	Wide Area Network

Capítulo 1

Introdução

A ubiquidade da Internet faz com que novas aplicações em rede, em particular as aplicações Web, sejam desenvolvidas tendo os protocolos da arquitetura Internet como base de sua comunicação. Além de presente nos mais variados lugares, o baixo custo de acesso[1], combinado com a disponibilidade aceitável da rede quando comparada com outros meios de comunicação, impulsionam seu uso tornando as aplicações oferecidas através dessa rede cada vez mais indispensáveis.

Pode-se dizer, então, que a Internet resolve o problema da comunicação de dados, restando ainda apresentar soluções de infra-estrutura[2] de tecnologia de informação (TI) de baixo custo e que entreguem uma disponibilidade maior que a da própria Internet. Infra-estrutura de TI compreende todo o aparato usado para manter computadores alimentados, refrigerados e conectados à rede, de forma ininterrupta, oferecendo uma gama aplicações que vão desde a transferência de arquivos, criada nos primórdios da Internet, até as contemporâneas redes sociais.

Para micro e pequenas empresas, o volume de dados muitas vezes leva a um custo proibitivo por transação, e foi visando a redução do custo dos serviços prestados para este tipo de empresa que surgiu um nicho de mercado no qual a mesma infra-estrutura abriga diversos servidores, permitindo assim o rateio do custo de infra-estrutura entre diversos clientes. Assim, tanto as pequenas empresas, que usam infra-estrutura de terceiros, quanto as médias e grandes, que têm infra-estrutura própria, têm em comum a necessidade de ter seus servidores disponíveis o máximo de tempo possível, para que as aplicações hospedadas por eles também estejam disponíveis.

Estas necessidades levaram os fornecedores de equipamentos de rede a desenvolver uma linha específica de produtos que ficou conhecida pelo termo "Alta Disponibilidade". Alta disponibilidade é a capacidade de um sistema de se manter disponível a despeito de eventuais falhas, para quantificar essa capacidade convencionou-se um parâmetro denomi-

[1] *Enfatiza a conexão com a rede de dados na função de usuário, em contraste com a função de servidor*

[2] A expressão infra-estrutura é utilizada ao longo do texto sempre com referência à infra-estrutura de tecnologia de informação (TI).

nado "Classe de Disponibilidade". Espera-se de um sistema classe 2 uma disponibilidade de 99%, e uma disponibilidade de 99,9% para um sistema classe 3, sendo o número da classe equivalente ao número de noves da porcentagem de disponibilidade esperada.

No cenário da infra-estrutura esta dissertação trata de manter computadores conectados à rede — trata mais especificamente dos protocolos utilizados, seus problemas, e como produzir um protocolo melhor que os que já existem e são usados para oferecer serviços de maneira ininterrupta. Atingir esse objetivo é equivalente, em outras palavras, a entregar alta disponibilidade. Mas conceitualmente o que vem a ser alta disponibilidade?

Disponibilidade é um dos três princípios do conceito de segurança da informação, que estabelece que informação segura é aquela que alia os adjetivos: íntegra; confidencial; e disponível [Lopes Filho 2008]. A semântica do termo *Alta Disponibilidade* pode variar dependendo do contexto em que é usado. Em essência, o termo se relaciona com o esforço, ou a intenção, de prover a maior, e a melhor, disponibilidade possível.

Quando se refere a serviços, pode-se dizer que *Disponibilidade* é a probabilidade de se estar disponível durante um determinado período de tempo. Por exemplo, servidores de *alta disponibilidade* estão disponíveis por mais de 99,999% do tempo que foram designados a operar. Diz-se ainda que *alta disponibilidade* é um padrão de projeto e construção de sistemas.

Por exemplo, um software pode ser implementado com uma técnica que minimize vulnerabilidades e com a capacidade de se recuperar automaticamente de eventuais falhas, inclusive falhas de hardware. Ou ainda, um hardware projetado com uma arquitetura redundante e construído com materiais que entreguem, ambos, o menor custo e o maior tempo médio entre falhas (MTBF - *Medium Time Between Failures*). Uma técnica comum em projeto e construção de sistemas de *alta disponibilidade* é a utilização de "Aglomerados" (*clusters*).

No que tange o tempo de resposta das transações, nos últimos anos, muito tem se falado sobre outra característica importante que é a Escalabilidade de Sistemas [Porto 2009]. Servidores tendem a degradar o tempo de resposta em função do volume de transações, sendo que escalabilidade (*Scalability*) é a habilidade que os sistemas escaláveis apresentam de oferecer desvio padrão próximo de zero no tempo de resposta mesmo na presença de alto volume de transações.

Escalabilidade horizontal [Porto 2009], na qual diversos servidores são disponibilizados para atender às requisições de um serviço, além da propriedade relativa ao desvio padrão no tempo de resposta, oferece também alta disponibilidade. Isto é, dado um conjunto de servidores, há uma grande probabilidade que, em um instante aleatório, haja um servidor disponível para atender à próxima transação. Há, portanto, relação entre escalabilidade e disponibilidade de sistemas.

Existem vários protocolos usados em arquiteturas de *alta disponibilidade* ou *escalabilidade* de sistemas. Neste trabalho foram analisados os protocolos: VRRP (*Virtual Redun-*

dant Router Protocol) — um protocolo proprietário largamente utilizado pelos maiores fornecedores mundiais de equipamentos de rede; CARP (*Common Address Redundancy Protocol*) — um protocolo de alta disponibilidade de domínio público; e Paxos — protocolo proposto para oferecer consenso entre vários participantes em um ambiente distribuído. Paxos foi escolhido por ter sido a base de diversos clones do *BigTable* [Chang et al. 2006] tais como: *Cassandra* [Cassandra 2009]; *HBase* [HBase 2008]; e *Hyper Table*.

Escalabilidade é a capacidade de os sistemas processarem cargas crescentes de trabalho, mantendo ou aumentando o desempenho de maneira satisfatória. A necessidade de escalabilidade para um sistema pode aparecer na forma de requisitos de desempenho. Por exemplo, imagine um sistema que deva atender às requisições em menos de cinco segundos, não importando se há uma ou milhares de requisições por segundo. A palavra "satisfatória" usada na especificação do requisito funcional indica que um sistema somente será dito escalável se atender a algum parâmetro pré-definido de desempenho. Neste contexto, "desempenho" é um valor, que geralmente possui relação com uma escala ou expectativa, cuja escalabilidade será o comportamento do desempenho perante à variação da carga de trabalho.

Não obstante Escalabilidade ser um requisito não funcional cada vez mais frequente nas especificações de sistemas distribuídos, nem sempre os desenvolvedores e arquitetos de sistemas estão a par dos desafios que um sistema desse tipo esconde. Servidores de Aplicação são uma peça fundamental dos sistemas distribuídos, e contam com a infraestrutura do sistema operacional, que permite criar e matar processos e *threads*, alocar e desalocar memória, comprometendo e liberando os recursos de hardware da máquina sob demanda. Some-se a isso as novas tecnologias de virtualização, que possibilitam que a aplicação alitize mais recursos de hardware, assim que a aplicação consome o que estava previsto inicialmente. Essa elasticidade conjugada com a multiplexação de recursos é um dos pilares dos sistemas distribuídos escaláveis.

No entanto, nem só de servidores de aplicação é composto um sistema escalável, interligando esses servidores há também os elementos de rede[3] tais como roteadores, e *switches*. Estes equipamentos executam tarefas específicas, curtas e repetitivas, cuja implementação no nível físico garante maior velocidade de operação. A velocidade é tão grande que não há tempo suficiente para o gerenciamento de processos, memória, etc. Há mesmo uma classe de dispositivos em que se aplica o conceito de Computação Desnuda (*Bare Computing* [Engler e Kaashoek 1995]), para os quais não se prevê nenhum vestígio de sistema operacional.

[3] *Network Elements: dispositivos especializados, geralmente implementam apenas as camadas mais baixas da arquitetura de rede.*

1.1 Motivação

Além de componentes redundantes uma infra-estrutura de rede de alta disponibilidade conta com protocolos de alta disponibilidade para lidar com a lógica da arquitetura. Esse modelo reduz o custo, uma vez que permite o uso de quaisquer equipamentos — independência de fabricante, incentivando a concorrência, bastando que eles implementem as mesmas interfaces e protocolos. Os protocolos de alta disponibilidade servem para eleger um equipamento — dentre os disponíveis no aglomerado (*cluster*) — apto a aceitar requisições de serviço, indicar substitutos para o eleito e empossá-los em caso de falhas, e monitorar constantemente a saúde do eleito.

Basicamente, o endereço conhecido publicamente (pelos clientes dos serviços) é o endereço virtual de um equipamento (denominado de *Master*) responsável por definir qual o equipamento do aglomerado (denominado de *Slave*) que responderá efetivamente pela requisição do serviço. Todos os equipamentos do aglomerado têm a capacidade de desempenhar quaisquer dos papéis *Master* ou *Slave*. Em um determinado instante, apenas um equipamento poderá responder pelo papel de *Master*.

O equipamento *Master* é então o cérebro na distribuição das requisições entre os demais equipamentos *Slaves* do aglomerado. Sabendo-se que os equipamentos são passíveis de falha, então, caso o equipamento *Master* venha a falhar, imediatamente é feita uma eleição para que um dos *Slaves* assuma o papel de *Master* e dali em diante passe a exercer as distribuições.

Apesar dos esforços para o desenvolvimento de equipamentos com maior MTBF , que em princípio visam oferecer sistemas altamente disponíveis, do surgimento de novas tecnologias, arquiteturas, e protocolos, porque ainda é possível perceber indisponibilidade em serviços oferecidos através da Internet?

A internet resolve o problema de comunicação de dados, principalmente para quem está diretamente ligado à espinha dorsal (*backbone*) da rede, sendo mais provável que a rede da empresa ou a interligação entre a Internet e a rede da empresa fiquem indisponíveis, do que a própria Internet. A alta disponibilidade entregue pelas soluções instaladas nas empresas hoje não é menor e nem o custo das soluções é maior, que as soluções anteriores. No entanto, há uma percepção mais aguçada, por parte do usuário, com relação à indisponibilidade de serviços. Em outras palavras, a alta disponibilidade se tornou um requisito não funcional mais rigoroso em relação aos sistemas mais antigos.

É exatamente esse o ponto onde se situa a motivação para produzir um protocolo de alta disponibilidade que esteja em uma classe de disponibilidade sem precedentes. Por outro lado, o esforço — em termos de pesquisa — que produziu um sistema de disponibilidade classe n é sempre maior que o que produziu um sistema classe $n-1$. Lógico que estar diante de um problema cuja solução se mostra desafiadora, com a possibilidade de produzir conhecimento e benefícios financeiros, só reforça a motivação.

Credita-se à percepção de indisponibilidade no serviço dois fenômenos que afetam o protocolo de alta disponibilidade, conhecidos como Acéfalo e Cérebro Partido. O equipamento *Master* de *facto* atua no encaminhamento do tráfego. Entretanto, se por algum motivo, na falha do *Master*, nenhum dos *Slaves* se habilite ao novo papel, o serviço ficará momentaneamente Acéfalo (*Brainless*) e, portanto, indisponível. Se, por outro lado, por algum motivo — como a partição da rede, houver a presença de mais de um cérebro (Cérebro Partido ou *Split Brain*), haverá mais de um equipamento tratando o tráfego, resultando na replicação de requisições, o que levará o protocolo de transporte a situações imprevisíveis (*communication reset*, *Silly Window Syndrome* - SWS, etc.) e à inacessibilidade.

1.2 Objetivos do Trabalho

Este trabalho objetiva aprofundar a análise das condições que concorrem para as situações de Acéfalo e Cérebro Partido à luz do exposto em [Hashimoto 2009] para o Protocolo VRRP e estendendo as análises aos protocolos CARP e Paxos. As condições de Acéfalo e Cérebro Partido são problemas clássicos da área de *Clustering*, onde vários computadores são usados para formar um sistema computacional. Geralmente um dos computadores assume o papel de *Master* e o restante ficam como *Slaves*. Internamente cada *Slave* obedece às ordens do *Master*, sendo que externamente vê-se uma entidade computacional única que provê recursos ou serviços.

Como observado em [Lopes Filho 2008] e [Hashimoto 2009], o protocolo VRRP é incapaz de entregar a alta disponibilidade pelas aplicações e padrões de tráfego contemporâneos, o que iniciou uma pesquisa para produzir um novo protocolo de alta disponibilidade. O objetivo principal desta dissertação é especificar um serviço para esse protocolo de alta disponibilidade, cuja função é oferecer disponibilidade melhorada em relação aos fenômenos de Acéfalo e Cérebro Partido.

Um objetivo secundário deste trabalho é oferecer uma visão mais precisa do que é um protocolo e mostrar a projetistas que mesmo protocolos bem projetados e implementados podem apresentar comportamentos que podem ser vistos como um problema. Este trabalho oferece subsídios a projetistas de protocolos especificando quais tópicos devem ser observados para que o projeto de um novo protocolo obtenha sucesso. Toda essa discussão está restrita aos protocolos para prover *alta disponibilidade*, embora as idéias descritas aqui possam ser usadas em outras áreas.

1.3 Contribuições

As principais contribuições deste trabalho são:

- especificação de serviço que compõe o projeto de um protocolo de alta disponibilidade, visando a diminuição das condições de Acéfalo e Cérebro Partido;
- apresentação de suposições a respeito do ambiente onde um serviço de alta disponibilidade deve operar e como os protocolos de alta disponibilidade podem resolver os desafios que surgem nesses ambientes;
- identificação de pontos críticos no projeto de protocolos de alta disponibilidade;
- estudo comparativo dos principais protocolos de alta disponibilidade da atualidade;
- identificação das características do protocolo Ethernet que podem provocar efeitos colaterais que até então eram creditados aos protocolos de alta disponibilidade e aos protocolos de camadas superiores, como é o caso do protocolo TCP.

1.4 Organização da Dissertação

No Capítulo 2 a arquitetura de alta disponibilidade é descrita com enfase no protocolo de alta disponibilidade, usando o protocolo VRRP para esclarecer alguns detalhes. Os problemas são delineados e os prejuízos que eles causam são discutidos, bem como algumas hipóteses de solução, auxiliando na formação do estado da arte e ajudando a identificar pontos críticos nos projetos de protocolos de alta disponibilidade.

O Capítulo 3 apresenta um estudo comparativo entre dois protocolos de *alta disponibilidade* e um protocolo de *escalabilidade* com relação à influência do protocolo Ethernet sobre os mesmos, coloca a *alta disponibilidade* e a *escalabilidade* como concorrentes, mostrando porque ainda há espaço para a *alta disponibilidade*.

Para garantir a completude de um protocolo, seu projeto deve conter pelo menos cinco elementos. O Capítulo 4 explica quais são esses elementos, reitera conceitos e tece um raciocínio norteador em direção à especificação de serviço, justificando as decisões de projeto e argumentando sobre o impacto das outras alternativas.

Por fim, o Capítulo 5 expõe os progressos obtidos na dissertação e justifica porque a especificação de serviço é um passo em direção a um protocolo de alta disponibilidade melhor. Apresenta ainda o que resta fazer, delineando as possibilidades de extensão da pesquisa.

Capítulo 2

Estado da Arte e Contexto do Problema

Este capítulo tem o objetivo de estabelecer as bases conceituais e filosóficas para o entendimento do contexto do problema. Além dos aspectos conceituais e filosóficos, é ainda objetivo deste capítulo introduzir as tecnologias existentes nos principais protocolos de alta disponibilidade, implementados pelos principais fornecedores de equipamentos.

Deste modo, introduz-se o conceito de Alta Disponibilidade, bem como os dois fenômenos que motivaram a concepção deste trabalho quais sejam as condições de Acéfalo e de Cérebro Partido. Em seguida são analisados os motivos que levaram à suspeita de que o principal problema dos protocolos de alta disponibilidade seria a camada de Transporte, o que levou a uma proposta de uma camada de transporte baseada no protocolo SCTP [Lopes Filho 2008]. Cabe também ao capítulo, explicar porque a suspeita deixou de pairar sobre a camada de Transporte e passou a pairar sobre a camada de Enlace, e analisar a extensão proposta ao protocolo VRRP em [Hashimoto 2009].

Resta ao capítulo ainda, uma vez que as duas propostas anteriores resolveram apenas parte do problema, mencionar as impressões sobre as soluções criadas para atuar no campo da escalabilidade, onde as bases de dados distribuídas como a *BigTable* e o Teorema CAP [Brewer 2000] nos pareceram especialmente úteis na solução do nosso problema.

2.1 Arquitetura de Alta Disponibilidade

No sentido amplo, a palavra arquitetura significa "princípios de construção", ocupa-se então das questões sobre a construção de um objeto, definindo seus componentes, sua organização e relacionamento. Num sentido mais restrito, a palavra arquitetura pode ser usada para enfatizar os detalhes de construção ou organização de determinado objeto. O termo "arquitetura de alta disponibilidade", pode então, ser usado tanto para tratar questões genéricas de construção, quanto para enfatizar detalhes de recursos ou serviços

altamente disponíveis.

Uma falha em componentes de um recurso ou serviço de alta disponibilidade pode levar à indisponibilidade, o que justifica o fato do princípio mais comum na construção de sistemas de alta disponibilidade ser a eliminação dos pontos únicos de falha, como ilustrado na Figura 2.1. Pontos únicos de falha podem ser removidos com a duplicação do componente considerado um ponto único de falha, mas em alguns casos essa redundância é economicamente inviável, sendo preferível então a utilização de um componente único mais robusto, ou ainda, a adoção de medidas que tornem o componente em questão menos suscetível a falhas.

Um projetista de rede experiente considerará que pode haver rompimento nos cabos ou fibras que interligam um sistema e providenciará interligações redundantes, passando por caminhos diferentes, de maneira que seja improvável um rompimento simultâneo nos dois caminhos. Outro problema a ser considerado é a possibilidade de o equipamento de rede, de onde os cabos e fibras chegam e partem, ficar sem energia. O projetista deve providenciar nesse caso, uma ou mais companhias de energia elétrica, além de fontes alternativas de energia como *no-breaks* e geradores, visando manter o equipamento sempre alimentado independente de falhas no fornecimento de energia.

Ainda que hajam cabos, fibras e energia, o próprio equipamento de rede pode deixar de funcionar ou apresentar mau funcionamento, que pode ser causado por um ataque explorando uma vulnerabilidade, ou uma atualização para a qual o hardware não é compatível [Caesar e Rexford 2008]. Alguns componentes simplesmente deixam de funcionar e fica claro que o substituto deve assumir a função, como geradores e *no-breaks* na falta de energia. Outros componentes, principalmente os ativos de rede, podem apresentar mau funcionamento, deixando grandes problemas por resolver, mesmo contando com componentes redundantes.

Enquanto os geradores e *no-breaks* foram criados com o propósito de suprir a ausência da fonte principal de alimentação, o que torna a substituição natural, os roteadores foram desenvolvidos para encaminhar os pacotes endereçados a eles, não sabendo como substituir a sua réplica. O preenchimento dessa lacuna exige a adição de um novo componente na arquitetura, cuja função é detectar o mau funcionamento e decidir qual réplica deve assumir a tarefa. Este trabalho se preocupa justamente em entender melhor o funcionamento deste componente, em explicar porque alguns comportamentos são considerados problemas, e apontar maneiras de melhorá-los.

Uma vez entendido o que se quer dizer com arquitetura e enfatizado qual (e porque) o componente da arquitetura é merecedor de todas as atenções, pode-se passar a explorar o conceito de alta disponibilidade. O termo "Alta Disponibilidade" pode ser visto sob dois aspectos sendo um deles o aspecto do desempenho dos sistemas, nos quais a alta disponibilidade significa manutenção do tempo de resposta aceitável para as transações. O outro é o aspecto da segurança, sendo a principal preocupação evitar o DoS (*Denial of*

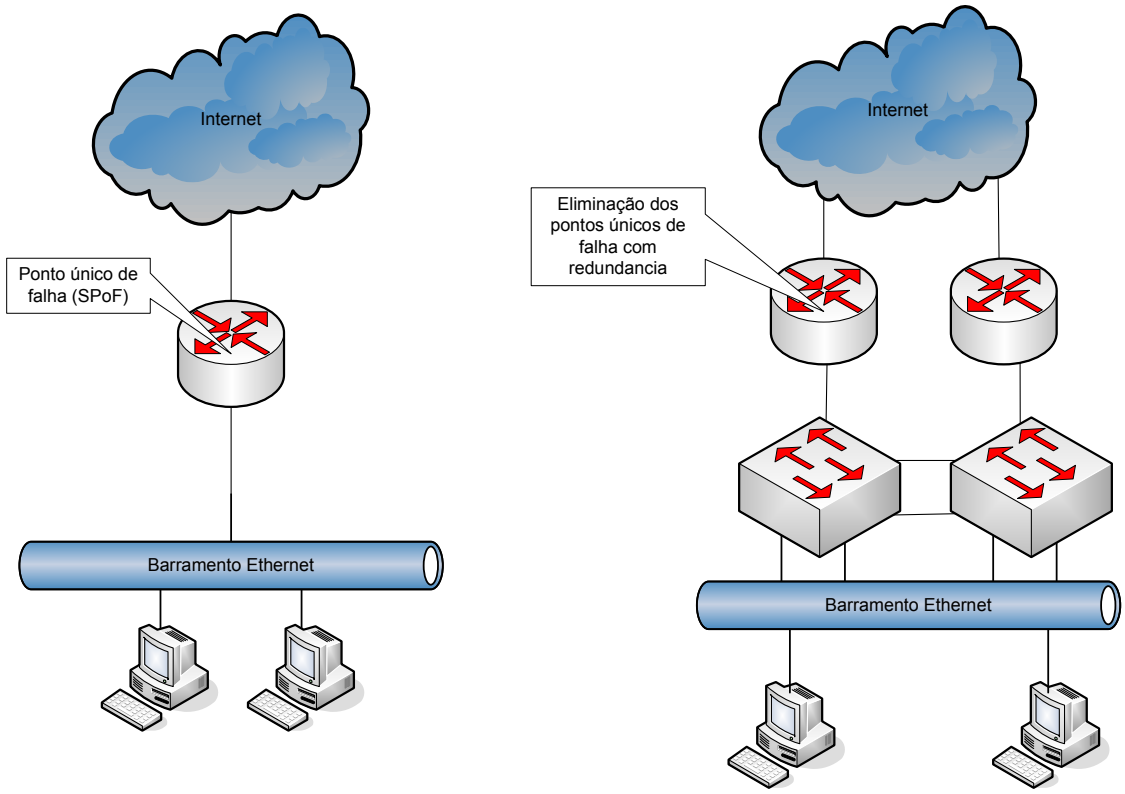


Figura 2.1: Eliminação dos Pontos Únicos de Falha

Service).

Neste trabalho, Disponibilidade será vista como um dos princípios da segurança da informação, que, como mencionado na introdução, estabelece que Informação Segura alia os atributos integridade, confidencialidade e disponibilidade [Lopes Filho 2008]. Dependendo do contexto, a semântica do termo *Alta Disponibilidade* pode variar significativamente, sendo que em essência o termo é relacionado ao esforço para prover a maior disponibilidade possível.

Do ponto de vista conceitual, independentemente do contexto, Disponibilidade é uma razão entre o tempo em que o sistema está efetivamente disponível e o período de tempo avaliado. Posto de outro modo, pode-se dizer que Disponibilidade é a probabilidade de um sistema estar disponível em determinado período. Para sistemas altamente disponíveis é comum considerar-se razão superior a 99%. Em função disto, criou-se a denominação "Classe de Disponibilidade" que conta o número de *noves* presentes na razão. Por exemplo, Classe de Disponibilidade 4, significa que o sistema está disponível 99,99% do tempo.

Para a implementação de um Sistema Altamente Disponível é primordial que se parta de um padrão de projeto e de construção, ou seja, é muito difícil, e em muitos casos impossível, desenvolver um sistema altamente disponível se ele não for originalmente projetado para tal. Por exemplo, é possível projetar um sistema (software) com uma técnica

que minimize vulnerabilidades e preveja a capacidade de se recuperar automaticamente na presença de eventuais falhas, inclusive falhas de hardware.

Um dispositivo físico (*hardware*), pode ser projetado com uma arquitetura redundante e construído com materiais que ofereçam o menor custo e o maior tempo médio entre falhas. Ressalte-se que mesmo com as considerações mencionadas, é evidente que este dispositivo intrinsecamente redundante (por exemplo com fonte redundante) é naturalmente mais caro que os não redundantes.

Para contornar a necessidade de hardware redundante, uma possibilidade real é o projeto e construção de Sistemas de Alta Disponibilidade a partir de uma Aglomerado (*Cluster*) de dispositivos de propósito geral. Um *Cluster* de computadores, por exemplo, pode entregar desempenho similar ao de super-computadores, com o conveniente de apresentar um custo bem menor.

Clusters podem ser construídos a partir de máquinas destinadas ao mercado doméstico, fáceis de se encontrar no mercado nacional, o que reduz os custos de construção e manutenção. É importante frisar que a menção a *Cluster* neste trabalho será sempre com o enfoque de disponibilidade e não de processamento de alto desempenho.

A palavra *Cluster* enseja idéias tais como agrupamento, conjunto, aglomeração, podendo ser utilizada em outras áreas do conhecimento, entre as quais química e música, servindo para denotar uma concentração incomum de elementos (químicos ou notas), que adquirem uma função sinérgica em relação a cada componente isolado.

As mesmas idéias valem para a área de computação, onde um *Cluster* pode ser usado para fornecer recursos computacionais como: processamento ou armazenamento. Na proposta de [Lopes Filho 2008] o termo *Cluster* é usado para entregar alta disponibilidade e o componente fundamental do *Cluster* é o protocolo VRRP.

Um *Cluster* de Alta Disponibilidade pode ser projetado para as mais variadas funções, mas em geral serve para eliminar a possibilidade de SPoF (*Single Point of Failure*). Em termos da segurança, há diversos elementos que são cruciais na interconexão de uma corporação à Internet e a outras empresas. Os dois primeiros elementos de interconexão importantes à segurança são o *Firewall* e o IPS (*Intrusion Prevention System*).

Um *Firewall* consiste em um método de proteção de uma rede contra outra potencialmente não confiável, através do controle de tráfego não autorizado [Lopes Filho 2008]. O método pode variar, mas um firewall, em geral, é constituído por dois componentes: um para o bloqueio de tráfego não autorizado; e outro para a habilitação de tráfego autorizado. Um *Firewall* se comporta como um *gateway* que repassa tráfego de um domínio de rede para outro, desde que o tráfego esteja em conformidade com uma regra de segurança específica para o tipo de tráfego em análise.

Um sistema de IPS monitora as atividades de uma rede ou sistema, buscando encontrar atividades maliciosas ou comportamento indesejável [Lopes Filho 2008]. Caso alguma atividade não autorizada seja detectada, ele pode reagir para bloquear ou prevenir a

concretização dessas atividades. IPSs de rede, por exemplo, operam em linha com a rede para monitorar todo o tráfego de rede, buscando encontrar e deter código malicioso e ataques. Quando um ataque é detectado ele pode excluir os pacotes ofensores, enquanto permite outros fluxos de pacotes.

Ao analisar as camadas de um sistema de interconexão, percebe-se que há necessidade de alta disponibilidade em diversas camadas tanto no aspecto da conectividade (switches, roteadores, etc.) quanto no aspecto da segurança (firewalls, IPS, etc.).

Como mencionado, apesar de todos os esforços para se obter alta disponibilidade, a realidade é que momentos de inacessibilidade existem por mais respeitável que seja o fornecedor da solução, em função das condições de Acéfalo e Cérebro Partido descritas oportunamente. As condições descritas foram detectadas a partir do protocolo VRRP e, portanto, as condições serão introduzidas considerando-se os fenômenos detectados em roteadores.

Contudo, as condições podem ser verificadas em todas as arquiteturas de Alta Disponibilidade que contem a filosofia de um nó *Master* e diversos nós *Slaves*, em que quaisquer dos *Slaves* possam assumir o papel de *Master* tão logo percebam sua inoperabilidade.

2.1.1 Protocolos de Alta Disponibilidade

Conforme já fora mencionado, as soluções de alta disponibilidade geralmente são formadas por hardware redundante e software pró-ativo (não necessariamente do mesmo fabricante) e apresentam um custo proibitivo para pequenas empresas. O que mais pesa no custo dessas soluções é o valor agregado (ter uma solução dessas como diferencial em relação aos concorrentes pode significar a manutenção ou ampliação da participação no mercado; ou um serviço de qualidade superior pelo qual os clientes pagarão a mais), seguido pelo retorno do investimento em P&D (projeto do hardware e parque industrial, especificação e implementação do software), e pela fabricação e comercialização propriamente dita.

Diversas instituições oferecem partes ou soluções de alta disponibilidade como: A Cisco que oferece roteadores e switches que implementam os protocolos [Li et al. 1998], VRRP [Hinden 2004] ou GBLP [Cisco 2010]. A Juniper que oferece firewalls que implementam o protocolo NSRP [Juniper 2010]. A comunidade do OpenBSD, que oferece o protocolo de domínio público [CARP 2004]

Os protocolos de Alta Disponibilidade mantêm os nós de uma infra-estrutura de Alta Disponibilidade em constante comunicação, por meio de primitivas de serviço[1] que essencialmente carregam informações de estado. O sentido e a quantidade de primitivas a serem trocadas entre os nós da infra-estrutura dependem do protocolo de Alta Disponibilidade utilizado.

[1]Neste trabalho os termos “Primitiva de Serviço”, “Primitiva” e “Mensagem” são tratados como sinônimos e intercambiáveis.

Em geral, o nó *Master* envia primitivas de requisição aos nós *Slaves*, através das quais reafirma o seu papel. Cada nó *Slave*, ao receber tal primitiva de indicação, atualiza seu estado e executa as transições pertinentes ao evento associado à recepção da primitiva.

Por exemplo, no caso do protocolo VRRP, o nó *Master* envia uma primitiva de serviço (*Advertisement*), que recebida pelos nós *Slaves* significa sua vivacidade e permanência em tal estado. Os nós *Slaves* atualizam seus estados (por exemplo, as temporizações) e permanecem no mesmo estado.

Os momentos de trocas de primitivas de serviços entre os nós (*Master* e *Slaves*) de uma infra-estrutura são especificados pelos protocolos de Alta Disponibilidade. Contudo, há basicamente duas abordagens de atuação dos protocolos:

Periódico: os nós trocam primitivas de serviço periodicamente (*State Driven*) entre si, as quais carregam essencialmente informações de estado; ou

Eventual: os nós trocam primitivas de serviço entre si em função da ocorrência de evento(s) (*Event Driven*), as quais carregam as condições da ocorrência do evento (e eventualmente informações de estado).

Evidentemente, não há uma mútua exclusão entre as duas abordagens, i.e., protocolos de Alta Disponibilidade podem fazer uso de ambas as abordagens. Em geral há uma preferência pela abordagem periódica em função da previsibilidade de tráfego na infra-estrutura de Alta Disponibilidade.

O **não recebimento** de uma primitiva de serviço que ateste a vivacidade do nó *Master*, provocará nos nós *Slaves* transições de estado na direção de assumir o papel de *Master*. Os critérios (por exemplo, a quantidade de não recepções, as temporizações, as prioridades, etc.) para um nó *Slave* assumir o papel de *Master* dependem de cada protocolo e são também aspectos de configuração da infra-estrutura. Em regime permanente, a inoperância do nó *Master* levará **um e apenas um** nó *Slave* a assumir o papel de *Master*.

Para facilitar a compreensão do problema, as introduções das condições de Acéfalo e Cérebro Partido serão feitas a partir da experiência adquirida com os protocolos de Alta Disponibilidade (em particular o VRRP) e, portanto, serão mencionadas condições específicas associadas ao problema do roteamento. Ressalte-se que esta abordagem não altera a especificação do problema.

2.1.2 O Protocolo VRRP

A redundância, como já foi citado, é um dos métodos (eliminação dos pontos únicos de falha por duplicação) usados para tornar um sistema de componentes críticos mais robusto, visando aumentar a sua confiabilidade. Essa elevação na confiabilidade só é

atingida quando se consegue resolver os problemas que surgem dessa multiplicação, ou seja, o gerenciamento dos componentes.

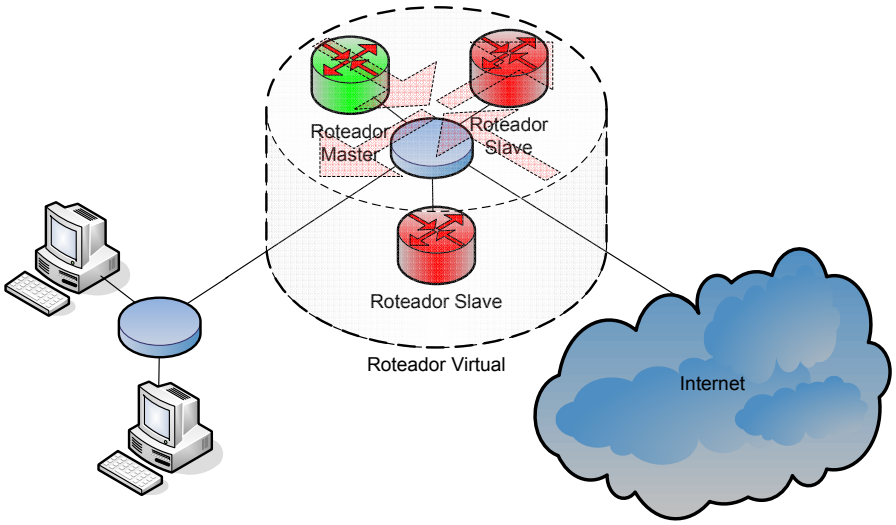


Figura 2.2: Roteador Virtual e respectivos Roteadores Master e Slaves

Suponha que dois dispositivos idênticos estejam configurados para executar o mesmo serviço em uma rede local. Um usuário conectado a essa mesma rede é informado sobre a existência do dispositivo 'A', requisita o serviço e começa a ser atendido. Em dado momento o dispositivo 'A' apresenta uma falha interrompendo o atendimento ao usuário. Esse problema pode ser resolvido de duas maneiras:

1. informando ao usuário sobre a existência de um dispositivo 'B' da mesma maneira que ele foi informado inicialmente sobre o dispositivo 'A'; ou
2. re-configurando o dispositivo 'B' para se passar pelo dispositivo 'A'.

De qualquer forma o usuário notará que o dispositivo 'A' ficou indisponível por algum tempo. A solução ideal seria não ter de re-informar aos usuários sobre outros dispositivos; nem ter de re-configurar dispositivos (pelo menos não com intervenção humana).

O Protocolo VRRP, protocolo de Alta Disponibilidade que opera com redundância por meio de um Roteador Virtual (*Master*), adota a idéia de replicação de equipamentos para prover um serviço de encaminhamento de tráfego capaz de continuar em caso de falhas. Resolve, deste modo, o dilema sobre re-informar aos usuários ou re-configurar os roteadores através do conceito de Roteador Virtual.

O Roteador Virtual, como ilustrado na Figura 2.2, existe apenas virtualmente (i.e., parece que existe mas não existe relativamente à tarefa de roteamento) e faz o papel do *Gateway* padrão para os *hosts* de uma rede local. O Roteador Virtual é identificado pelo VRID (*Virtual Router Identifier*), um conjunto de endereços de rede (IP), e um endereço de acesso ao meio (MAC - *Medium Access Control*), associados à sua interface.

O Roteador VRRP é um roteador no qual é habilitado e configurado o protocolo VRRP, e pode fazer parte de um ou mais domínios de Roteadores Virtuais. Somente o endereço IP do Roteador Virtual é divulgado, e cada roteador, seja ele virtual ou físico, possui configuração independente. Logo, não é necessário re-informar aos usuários ou re-configurar os roteadores em caso de falha.

No interior de um Roteador VRRP, existe um processo que mantém estados e reage a eventos (primitivas de serviço ou temporizadores) de acordo com a especificação do protocolo. Um dos Roteadores VRRP fica como *Master* do Roteador Virtual e assume a responsabilidade de encaminhar o tráfego e tratar em seu nome as requisições relacionadas à interface do Roteador Virtual. O restante dos Roteadores VRRP ficam como *Slaves* do Roteador Virtual[2] prontos para assumirem em caso de falha do *Master* [Hinden 2004].

As principais funções do protocolo VRRP são: manter funcional a interface do Roteador Virtual; eleger um *Master* para o Roteador Virtual sempre que for necessário; ser capaz de indicar caminhos preferenciais; e minimizar interrupções indesejáveis ao serviço. Isso é possível por dois motivos:

1. o endereço de rede do Roteador Virtual sempre resolve para um endereço de acesso ao meio do tipo *multicast*, logo todos os Roteadores VRRP recebem o tráfego enviado ao Roteador Virtual; e
2. as requisições (batimento cardíaco do *Master*) são transportadas por IP *multicasting*, o que permite normalmente informar simultaneamente a todos os Roteadores VRRP a um só tempo (curto).

Embora haja um ou mais Roteadores VRRP (roteadores físicos e portanto redundantes), seus endereços não são divulgados aos usuários dos serviços. Em seus lugares, é divulgado o endereço do Roteador Virtual que é de fato o roteador visto no mundo externo (por exemplo na Internet).

Ao entrar em operação o Roteador VRRP verifica o valor do parâmetro *Priority*, como mostrado na Figura 2.3. Caso o valor seja igual a 255 o Roteador VRRP torna-se imediatamente o *Master* do Roteador Virtual, caso o valor seja menor que 255 o Roteador VRRP torna-se *Master* imediatamente. Daí por diante uma falha no *Master* leva a uma eleição para escolher quem será o novo *Master*. Cada Roteador VRRP envia uma mensagem com o seu valor do parâmetro *Priority* para todos os outros Roteadores VRRP, e aquele que não receber nenhuma mensagem cujo valor seja maior que o seu próprio *Priority* automaticamente se tornará o novo *Master*.

Todos os Roteadores VRRP, participantes do domínio de um Roteador Virtual, têm o direito a votar nas eleições dessa instância (para a eleição do próximo *Master*). O vencedor de cada eleição fica com a responsabilidade de encaminhar os tráfegos em nome

[2]Eventualmente chamados de *Backups* ou Reservas.

do Roteador Virtual e com o dever de manter cientes os Roteadores VRRP dessa instância de que ele está encaminhando o tráfego.

Isso é feito através de primitivas de Anúncio (*Advertising*). Os perdedores recebem a denominação de *Slaves* e ficam com as responsabilidades de: monitorar as indicações de Anúncio enviadas pelo roteador *Master* (caso contrário ele será considerado inoperante); e realizar uma nova eleição caso não recebam as indicações de Anúncio. O diagrama de estados a seguir apresenta o comportamento do autômato VRRP.

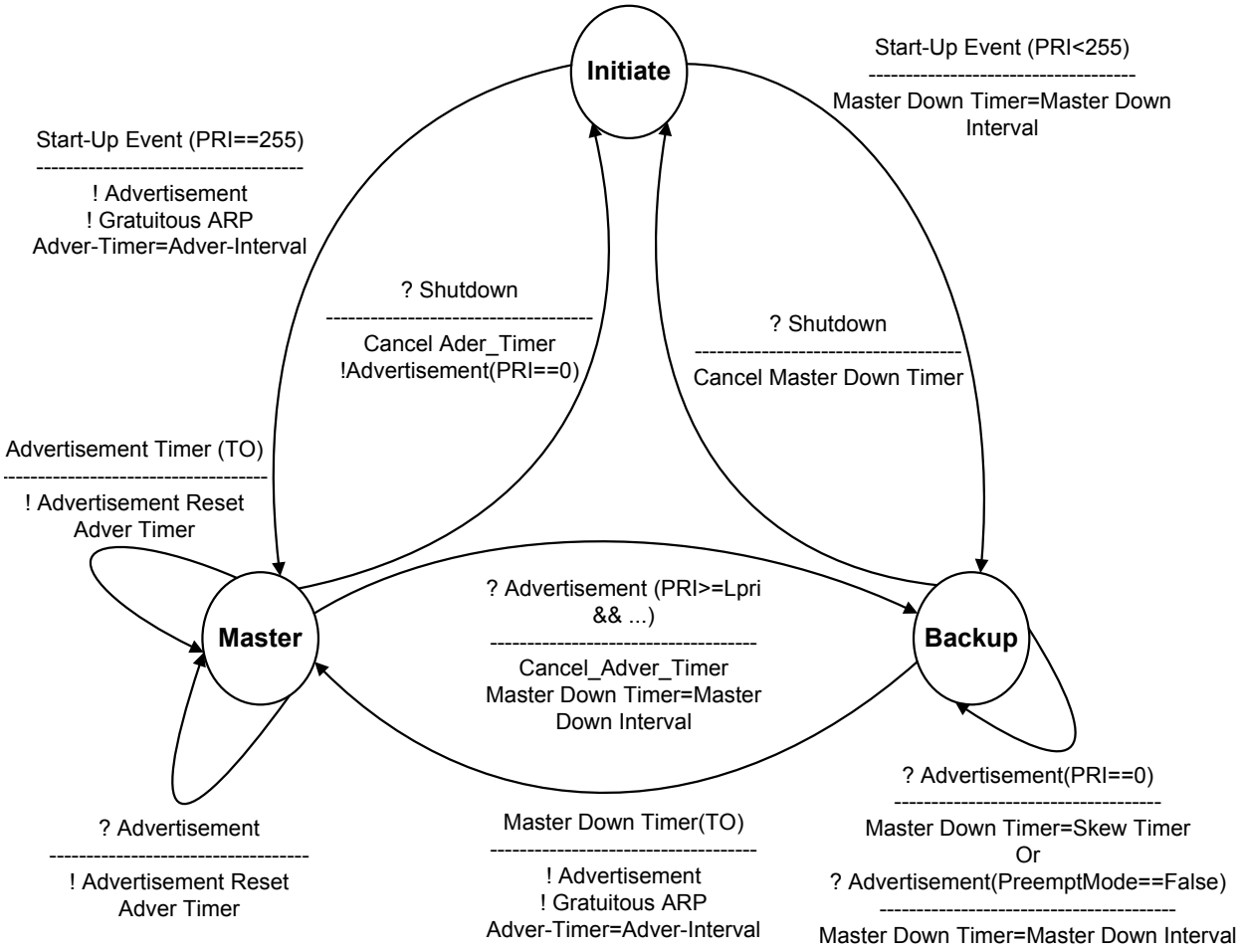


Figura 2.3: Diagrama de Estados do Protocolo VRRP [Hinden 2004]

2.1.3 Condição de Acéfalo

Definição:

"Acéfalo é a condição em que o nó de uma infra-estrutura de Alta Disponibilidade correntemente no papel de Master se torna inoperante e nenhum

outro nó (nós Slaves), como mostrado na Figura **??***, se habilita a assumir tal papel."*

Para o protocolo VRRP, o nó *Master* é denominado "Roteador Virtual" e os nós *Slaves* são denominados "Roteadores VRRP". Do ponto de vista do mundo externo (por exemplo, da Internet), o roteador visível é o Roteador Virtual, que abstrai (e representa) todos os Roteadores VRRP.

Para evitar que dois ou mais Roteadores VRRP assumam o papel de *Master*, caso o Roteador Virtual se torne inoperante, o protocolo VRRP atribui prioridades diferentes aos Roteadores VRRP e estas prioridades indicarão a preferência de um Roteador VRRP na eleição de um novo Roteador Virtual.

Essa prioridade pode ser baseada no custo ou velocidade do enlace, no desempenho do roteador, ou em qualquer outra política que se queira instituir para influir no resultado da eleição. Isto possibilita que os "melhores roteadores" exerçam o cargo de Roteador Virtual em uma ordem pré-determinada [Hinden 2004] por configuração.

O protocolo VRRP usa IP *Multicasting* [Deering 1989] para enviar as primitivas requisitadas pelo Roteador Virtual e endereçadas aos Roteadores VRRP. Endereçamento *Multicast* diminui drasticamente o tráfego de primitivas na rede, mas têm o inconveniente de serem suportadas apenas por Serviços Não-Orientados a Conexão (*Connecitonless Services*), que são essencialmente não confiáveis.

O uso de Endereçamento *Multicast* permite entender o motivo de certas perdas de primitivas, que circunstancialmente justificariam a condição de Acéfalo. A relação com a camada de Enlace será discutida com mais profundidade em 3.1.2.

Além da utilização de endereçamento *Multicast*, o protocolo VRRP não utiliza nenhum mecanismo de segurança (criptografia, autenticação) para garantir as autenticidade e integridade das primitivas.

Do ponto de vista da segurança, é possível para qualquer máquina situada na mesma rede local do Roteador Virtual se passar por um Roteador VRRP impostor alegando participar do domínio desse Roteador Virtual e, além disso, possuir prioridade superior ao Roteador Virtual atual.

Se a funcionalidade de indicar caminho preferencial estiver ativa, o Roteador Virtual abdicará do papel em prol do Roteador VRRP impostor, que se tornará, portanto, o novo Roteador Virtual sem, contudo, honrar o compromisso de encaminhar o tráfego aos Roteadores VRRP, resultando na indisponibilidade do serviço de encaminhamento de tráfego.

Essa situação é chamada Condição de Acéfalo, pois não há de fato um Roteador Virtual no comando, por exemplo, não há um cérebro. Em outras palavras, todos os Roteadores VRRP estão na condição de *Slaves* e o Roteador Virtual não desempenha o papel de *Master*.

Segundo [Lopes Filho 2008], o surgimento de *Worm*[3], possibilita esse tipo de ataque mesmo em redes que implementem mecanismos e políticas rígidas de segurança.

2.1.4 Condição de Cérebro Partido

Definição:

> *"Cérebro Partido é a condição em que um ou mais nós (correntemente no papel de Slaves) de uma infra-estrutura de Alta Disponibilidade se alçam ao papel de nó Master, como mostrado na Figura 2.4 por julgarem que o atual nó Master tornou-se inoperante."*

Em ambientes complexos é comum uma rede local se estender por vários *switches*, eventualmente, passando por *firewalls* e roteadores por meio de tecnologias como VLAN (*Virtual LANs*). Esses ambientes aumentam a probabilidade de erros em primitivas de serviços *Connectionless* (como são os casos dos protocolos Ethernet, IP e UDP) que podem levar temporariamente o Roteador Virtual à condição de Cérebro Partido. Sem mencionar as possibilidades de problemas de segurança e erros de implementação e configuração, que podem levar a infra-estrutura de Alta Disponibilidade a essa condição por períodos consideráveis.

Mesmo em outros ambientes a condição de Cérebro Partido é conhecida e considerada como um problema conforme o número de patentes [Armstrong 2008], [Hara 2005], [Schoenthal 2006], e [Wipfel 2005] sugere. No ambiente específico desta dissertação, O não recebimento de primitivas de Anúncio, que devem ser enviadas pelo *Master*, leva os *Slaves* a realizarem nova eleição na qual um novo *Master* é eleito. Os projetistas do protocolo VRRP consideraram que a única razão para não chegar uma primitiva de Anúncio é a falha do *Master*.

No entanto, ambientes de rede complexos mostram que essa não é a única razão, pois, por exemplo, erros de CRC (*Cyclic Redundancy Check*) detectáveis pelo protocolo Ethernet inutilizam ou descartam os quadros (*Frames*) enviados periodicamente pelo *Master*, resultando ao final de uma eleição na Condição de Cérebro Partido. Há então dois Roteadores VRRP no estado de *Master* disputando o comando do Roteador Virtual, ou seja, o comando está partido.

Uma possibilidade de resolver (ou reverter em caso de ocorrência) o problema introduzido pelo Cérebro Partido é a aplicação da técnica denominada de *Fencing*. Essa técnica consiste em 'cercar' os nós do *cluster*, que não deveriam estar ativos em determinado momento, e através de um Algoritmo de Isolamento decide se o nó deve permanecer como parte ativa do Roteador Virtual ou se deve deixar de se comunicar com os outros nós, isolando-se. A idéia de *Fencing* foi abandonada, pois isolar os nós seria desperdício de recursos à luz do conceito de Escalabilidade, introduzindo oportunamente.

[3]Um tipo específico de vírus conhecido como Verme, que ataca a rede e não os computadores

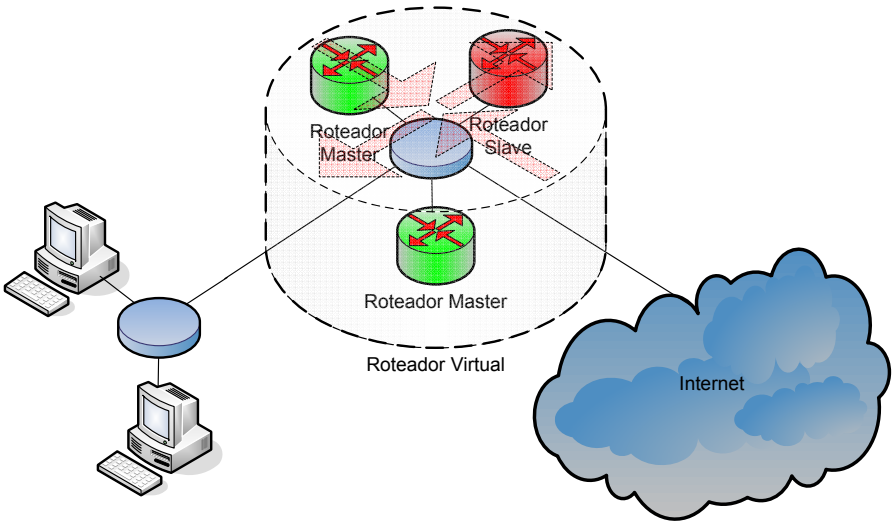


Figura 2.4: Roteador Virtual na Condição Cérebro Partido

Mesmo em um curto período de existência, a Condição de Cérebro Partido pode levar a sérias consequências, tanto no aspecto do desempenho quanto no aspecto da segurança (um acesso pode ser bloqueado por tentativas inesperadas e replicadas). Uma base de dados pode se tornar inconsistente, usuários podem ser desconectados de uma rede, pode ocorrer exposição inapropriada de informação ou a interrupção de um serviço.

2.2 Propostas de Melhoria em Protocolos de Alta Disponibilidade

Esta seção investiga a suspeita de que o principal problema dos protocolos de alta disponibilidade seria a camada de Transporte, o que levou a uma proposta de uma camada de transporte baseada no protocolo SCTP [Lopes Filho 2008]. Cabe também a ela, explicar porque a suspeita deixou de pairar sobre a camada de Transporte e passou a pairar sobre a camada de Enlace, e analisar a extensão proposta ao protocolo VRRP em [Hashimoto 2009]. Uma vez que as duas propostas anteriores resolveram apenas parte do problema decidiu-se observar também soluções criadas para atuar no campo da escalabilidade, onde bases de dados distribuídas como a *BigTable* e o Teorema CAP [Brewer 2000] nos despertaram a atenção como ferramentas úteis na solução do problema de acéfalo e cérebro partido no protocolo VRRP.

2.2.1 Camada de Transporte Baseada em SCTP

O fato do protocolo VRRP (bem como outros protocolos de Alta Disponibilidade) ter sido projetado para operar sobre os protocolos UDP/IP (protocolos *connectionless* das

camadas de Rede e Transporte) suscitou a suspeita de que o problema residia no tipo de transporte adotado.

[Lopes Filho 2008] examina então o protocolo VRRP [Hinden 2004] (como um mecanismo de *alta disponibilidade*), e conceitos de *Firewall* e IPS para prover um serviço seguro de *alta disponibilidade*, momento em que aponta um problema capaz de desencorajar o uso dessa solução.

O trabalho analisa o(s) problema(s) e propõe uma nova arquitetura, para prover o serviço de *alta disponibilidade*, alterando o protocolo VRRP para utilizar o protocolo de Transporte SCTP (*Stream Control Transport Protocol*) [Steward 2007] no envio das mensagens de Anúncio, ao invés dos protocolos UDP/IP *multicasting* [Deering 1989] usados originalmente.

O propósito do protocolo VRRP é manter o serviço de encaminhamento de pacotes sempre disponível, ou, em outras palavras, minimizar o número e a duração de períodos de indisponibilidade no serviço de roteamento. [Lopes Filho 2008] percebeu que no ambiente de uma empresa, que uma solução usando o protocolo VRRP apresentava momentos total indisponibilidade. E em seus experimentos constatou que conseguia fazer todos os nós do *Cluster* agirem como *Slaves*. Uma pesquisa preliminar mostrou que se tratava de um velho problema da área de *clustering*, a Condição de Acéfalo.

A Condição de Acéfalo é o fenômeno que identifica o problema do trabalho de [Lopes Filho 2008] e sua causa é a utilização de IP *multicasting* sem uma preocupação da camada superior em prover mecanismos de autenticação, que permita a hospedeiros (nós conectados a uma rede de computadores, dispositivos cuja função é hospedar informações ou serviços) se passarem por Roteadores VRRP através da inserção de primitivas que não pertencem a nenhum dos membros do Roteador Virtual.

O protocolo SCTP [Steward 2007] apresenta requisitos de segurança, que levaram [Lopes Filho 2008] a escolhê-lo como uma alternativa para o transporte das primitivas de Anúncio VRRP, substituindo o UDP/IP *Multicast*. Outro motivo que também influenciou nessa escolha foi o aproveitamento da característica de *Multihoming* do protocolo SCTP em conjunto com a utilização de interligações exclusivas para o tráfego de alta disponibilidade, inspirado em soluções proprietárias como os protocolos HSRP [Li et al. 1998] e NSRP [Juniper 2010].

diversas teses.

O protocolo SCTP é naturalmente orientado a conexão (*Connection Oriented*), conceito que sustenta a confiabilidade das comunicações de transporte (*Reliability*). Além disso, o conceito de conexão do protocolo TCP [Postel 1981] (*Mono-stream*) é estendido para suportar o conceito de associação de transporte (*Multi-stream*).

Durante o estabelecimento da Associação, cada *end-point* da comunicação fornece ao *end-point* parceiro uma lista de endereços de transporte por meio dos quais podem ser alcançados e a partir dos quais serão enviados os segmentos (primitivas da camada de

transporte) do protocolo SCTP.

Atualmente é comum um hospedeiro possuir mais de uma interface de rede e, geralmente, essas interfaces estão conectadas a redes totalmente independentes, permitindo que uma aplicação seja identificada por um *end-point* de um conjunto de endereços de rede.

Devido à capacidade de suportar múltiplas interfaces (*Multi-homing*) conectadas a redes independentes (distintas), o transporte de dados pode ser feito por diversos caminhos (*Streams*). O protocolo SCTP escolhe um caminho primário e monitora os caminhos alternativos fornecendo um serviço de transferência de dados altamente disponível.

A aplicação pode solicitar o transporte de informações por quaisquer dos caminhos alternativos; ou ainda, o próprio protocolo pode alterar transparentemente o caminho de transporte caso haja algum problema com o caminho principal. Para monitorar esses caminhos, o protocolo define duas primitivas de serviço (*HEARTBEAT* e *HEARTBEAT-ACK*), trocadas periodicamente. O tempo que as mensagens levam para atravessar a rede e a corrupção ou perda dessas mensagens são parâmetros utilizados para determinar o estado de cada caminho. O autômato do protocolo mantém os estados das associações, os quais servirão de base para a definição do melhor caminho de transporte.

Como parte dos requisitos de segurança, o SCTP oferece serviço de transporte de dados com as seguintes características: disponibilidade; confidencialidade; integridade; confiabilidade; e em tempo (*Timeliness*).

O SCTP reduz o risco de ataques de personificação [Nam et al. 2010] usando o *four-way handshake*. Uma vez que a troca inicial de mensagens usa o mínimo de memória o desperdício de recursos é aceitável em caso de ataque. Além disso, a mensagem INIT ACK contendo o *State Cookie* é retornada para o hospedeiro que originou a mensagem INIT, deixando o atacante sem qualquer informação que possa favorecer o ataque. O SCTP se protege contra a inserção de pacotes estranhos ao fluxo de uma associação estabelecida usando a *Verification Tag*.

Estas propriedades do protocolo SCTP excedem em muito as propriedades do protocolo UDP que, por ser não orientado a conexão (não confiável), não oferece qualquer meio para quaisquer verificações, por exemplo se uma primitiva chegou.

A análise desenvolvida em [Lopes Filho 2008], que aponta para as deficiências do protocolo UDP (não confiável) e que levaram à adoção do protocolo SCTP, mostrou que a hipótese estava parcialmente correta.

O protocolo UDP permite endereçamento *Multicast*, pois não tem a incumbência de estabelecer e manter conexões fim-a-fim [Postel 1980]. O fato de não ser um protocolo orientado a conexão faz com que a responsabilidade de verificar erros de transporte sejam legadas às camadas superiores, o que resulta em um transporte consideravelmente rápido (mas não confiável).

O protocolo SCTP impede que mensagens estranhas sejam inseridas no fluxo normal

de uma associação, o que é suficiente para evitar que o *Cluster* seja levado à condição de Acéfalo por ataques *Man in the middle* [Nam et al. 2010], por exemplo.

No entanto, o esforço do SCTP para manter a associação disponível acaba por contrariar um dos principais objetivos do protocolo VRRP, que é minimizar o número e a duração das interrupções de serviço. Esse efeito colateral é compreensível, pois o propósito do protocolo SCTP é oferecer serviços de transporte fim-a-fim (*end-to-end*) confiáveis e disponíveis. Isto é, ele envida todos os esforços para entregar as primitivas através das *Streams* da Associação.

A análise das facilidades introduzidas pela nova camada de Transporte, trouxe à luz um novo fenômeno na infra-estrutura de Alta Disponibilidade e que fragilizou momentaneamente a solução proposta no referido trabalho. Mesmo utilizando a camada de transporte confiável ainda era possível observar indisponibilidades momentâneas, e embora dessa vez o *cluster* pudesse se recuperar delas automaticamente, isso não diminuía a gravidade de situação. Como se verá a seguir, pesquisas mostraram que se tratava de um outro problema, também bastante conhecido na área de *clustering* e considerado irmão da Condição de Acéfalo, chamado a condição de Cérebro Partido.

2.2.2 Aspectos da Detecção de Erros na Camada de Enlace

Pesquisadores e desenvolvedores de protocolos da camada de Enlace[4], desde os primórdios das redes, enfrentaram o desafio de corrigir as distorções introduzidas pelo meio físico (erros de bit). Cientes de que produzir uma interface de rede capaz de detectar e corrigir todos os erros introduzidos pelo meio físico tornaria o produto mais caro e complexo (além do desempenho inferior), decidiram utilizar algoritmos[5] que tratariam distorções cujos comprimentos não fossem superiores a 'n' bits.

Os engenheiros da Digital, Intel e Xerox (DIX), responsáveis pelo projeto do protocolo Ethernet, utilizaram o algoritmo CRC-16, cuja distância de Hamming é capaz de detectar 100% de erros em rajadas de até 3 bits de erro. Para rajadas de erros superiores a 3 bits, o algoritmo CRC-16 não garante a detecção de erro. Neste caso, a primitiva, mesmo contendo bits distorcidos, será encaminhada à camada superior (do ponto de vista conceitual a camada de Rede).

Este comprimento de rajada era muito razoável para a época do projeto. Se considerarmos que os meios físicos da época (década de 70) apresentavam velocidade da ordem de 9.600*bps* e que a taxa de erro para um cabo TP (*Twisted Pair*) era de algo em torno de 10^{-4}, era de se esperar então, uma média de 1 erro de bit por segundo. Considerando-se 3 bits de erro, a taxa cai para 10^{-12}, então era necessário uma sequência de bits que levaria algo em torno de 3 anos.

[4]Em geral a sub-camada MAC (*Medium Access Control*) da camada de Enlace é hospedada na placa de rede)

[5]Tipicamente FCS (*Frame Check Sequence*) ou CRC (*Cyclic Redundancy Check*)

O fato de o protocolo Ethernet ter se estabelecido em praticamente todas as redes locais e, de modo semelhante, como as camadas Física e de Enlace da borda da Internet[6], torna-o um protocolo a ser analisado primordialmente. Contudo, os aspectos de erros mencionados para o caso do protocolo Ethernet variam em amplitude, mas não em essência, para os demais protocolos similares.

Como dito acima, esse protocolo usa um algoritmo de verificação de redundância cíclica (CRC-16) para detectar erros nos quadros (*Frames*), sendo que existem trabalhos, [Fujiwara et al. 1989] por exemplo, que apontam situações em que é possível erros passarem para as camadas superiores sem serem detectados. Não se trata de uma falha do algoritmo CRC, mas uma escolha dos projetistas do protocolo Ethernet.

Além disso, ressalte-se que os antecessores do protocolo VRRP foram projetados para trabalhar num ambiente onde o tráfego de controle era transportado por canais separados do tráfego de dados. Esses canais não necessariamente usavam o protocolo Ethernet. Foi justamente a popularidade e o baixo custo das interfaces Ethernet que levaram à convergência dos tráfegos de dados e controle, expondo o tráfego de controle a competir com o tráfego de dados.

A necessidade de minimizar o tráfego de controle e a característica de comunicação do protocolo VRRP (um *Master* envia primitivas de Anúncio para os Roteadores VRRP), demonstrou um carácter de endereçamento *Multicast*[7] que só é suportado por protocolos com serviços não orientados a conexão (serviços não confiáveis).

Isso explica porque os projetistas do protocolo escolheram o protocolo UDP (não orientado à conexão), para transportar as mensagens de Anúncio. Essas mensagens controlam as transições de estado dos roteadores VRRP, que ocorrem mais rápido uma vez que os roteadores não tem que se preocupar com estabelecer, manter e encerrar conexões para trocar essas mensagens.

Como o protocolo IP também não é orientado a conexão, forma-se então uma arquitetura propicia para que erros do meio físico (distorção dos bits da primitiva) cheguem até o protocolo da camada de Aplicação, neste caso o próprio protocolo VRRP. O cenário fica ainda mais complicado se considerarmos que os meios físicos da atualidade têm taxa de transmissão da ordem de *Gbps*.

enviados. Conforme trecho do IEEE Std 802-2001 o comportamento das redes com relação a erros deve respeitar as seguintes condições:

a) Para os meios físicos de cobre e fibra ótica: Dentro do mesmo segmento de rede, a probabilidade[8] de que um quadro MAC transmitido (excluindo o preâmbulo) não seja

[6]Acesso Banda Larga em todo o mundo tem sido feito através de variações do protocolo Ethernet, como é o caso do PPPoE

[7]Endereçamento que permite a comunicação 1 (envia) para N (recebe)

[8]O comportamento citado em (a) define uma característica de desempenho desejada nas LANs, uma vez que se baseia em outros aspectos do serviço entregue, como perdas de quadros e atrasos de transmissão causados pela necessidade de retransmissão. No entanto, essa medida não é realista para todos os tipos de meio físico; por exemplo, meios físicos sem fio podem ser incapazes de atingir esse nível de desempenho na

recebido corretamente na interface de serviço da entidade MAC de destino, levando em consideração apenas a operação da camada física, deve ser menor que $8x10^{-8}$ por octeto do quadro MAC.

b) Para meios físicos sem fio: Dentro do mesmo segmento de rede, a probabilidade de que um MAC Service Data Unit - MSDU não seja entregue corretamente no MSAP do usuário de serviço MAC de destino, levando em consideração a camada física e o protocolo MAC, deve ser menor que $8x10^{-8}$ por octeto do MSDU.

c) A probabilidade[9] de um MSDU entregue em um MSAP conter um erro não detectado, levando em consideração a operação do provedor de serviço MAC, deve ser menor que $5x10^{-14}$ por octeto do MSDU.

O protocolo VRRP assume que erros de bit são detectados pela camada de Enlace e, portanto, não toma nenhuma providência, a não ser aguardar pela perda de três mensagens consecutivas para realizar uma nova eleição. A taxa de erros atual do protocolo Ethernet, pode se transformar em um número significativo de eventos frente aos modernos padrões de tráfego. Atualmente é comum hospedeiros acumularem tráfego da ordem de Petabytes por dia, o que nos leva a dez erros para cada Petabyte de tráfego.

2.2.3 A solução com VRRP Estendido

[Hashimoto 2009] atribui os erros não detectados pelos algoritmos de controle de erro da camada de Enlace como causa para o problema da condição de Cérebro Partido e conclui pela necessidade de desenvolvimento de um novo protocolo de Alta Disponibilidade. Foi demonstrado que o autômato do protocolo VRRP é incompleto, pois ele não considera os erros não detectáveis pela camada de Enlace. [Hashimoto 2009] apresenta um novo autômato, que é um dos elementos na proposição de um protocolo [Holzmann 1991]. A modelagem em Redes de Petri especificou as perdas de mensagens de Anúncio em função dos fenômenos da camada de Enlace, o que resultou no novo autômato.

A especificação de [Hashimoto 2009] prevê 4 lugares (Equipamentos em Manutenção, Equipamentos não Atribuídos, Equipamentos no Estado *Acting* e Equipamentos no Estado *Slave*), inspirada nos estados do autômato do protocolo VRRP (*Initiate*, *Master* e *Backup*), como esclarece a Figura 2.5. Os demais lugares servem para tornar a especificação bem formada.

A transição t_7 representa o envio de mensagens de Anúncio por parte do *Master* e a transição t_8 representa a recepção das indicações dessas mensagens por parte dos *Slaves*.

camada física devido às suas características. Nesses casos a operação do protocolo MAC deve empregar mecanismos adicionais, por exemplo, detecção e correção de erros, para garantir que o provedor de serviço MAC atinja na interface do serviço MAC oferecido os níveis de desempenho que a condição implica.

[9]Por exemplo, (a) o pior caso de probabilidade de perder um quadro IEEE 802.3 com tamanho máximo de 1518 octetos por dano na camada física deve estar abaixo de $1.21x10^{-4}$, ou aproximadamente 1 em 8250; (c) o pior caso de probabilidade de um quadro similar, que contém um MSDU de 1500 octetos, seja entregue com um erro não detectado deve estar abaixo de $7.5x10^{-11}$, ou aproximadamente 1 em 13300000000.

Essa sequência de disparos passa por um lugar, P_6, que representa a mensagem no meio físico, com uma transição de saída, t_9, cujo disparo representa perda de mensagens no meio físico (incluindo aquelas descartadas pela camada de Enlace em função de conterem erros).

As transições t_3 e t_6, de saída dos lugares *Acting* e *Slave*, especificam os eventos relativos a falhas de operação de nós do *Cluster* e conduzem as respectivas fichas (*Tokens*) para o lugar que representa o estado de manutenção do equipamento.

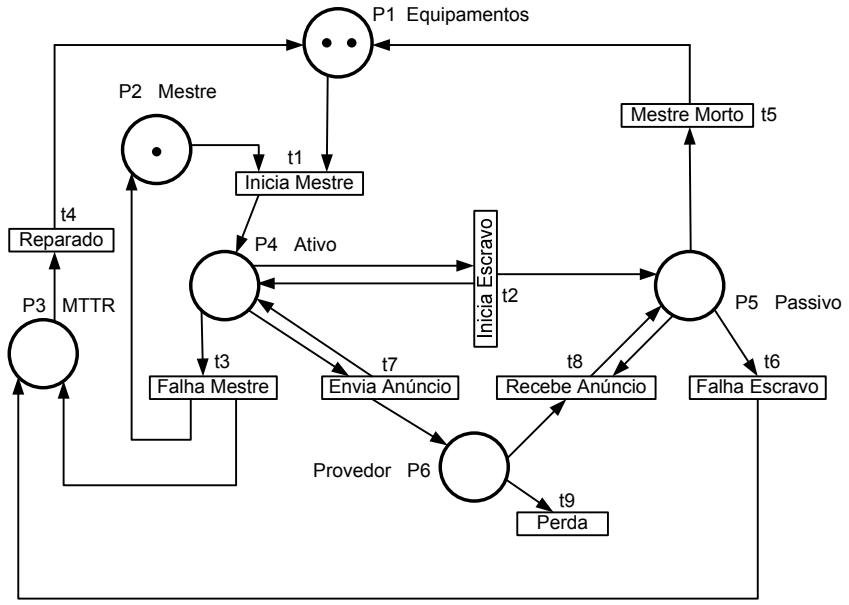


Figura 2.5: Rede de Petri: Modelagem do Problema de Erros não Detectáveis no Protocolo Ethernet [Hashimoto 2009]

O aspecto mais importante nessa especificação reside no lugar P_2 (*Toogle*), que limita a quantidade de fichas no lugar P_4 (que representa o número de nós no estado *Master*). Este fragmento da especificação é responsável por garantir que haja apenas um equipamento no estado *Master* em um determinado instante. No entanto, do ponto de vista operacional, isso não garante que não haverá mais de um nó ativo no *cluster* real. Como viu-se anteriormente o protocolo Ethernet pode danificar, atrasar ou perder quadros.

[Hashimoto 2009] especifica a possibilidade de perda, o que é uma simplificação válida, uma vez que ela é reprodutível na realidade. Todavia, o mesmo não pode ser dito da abstração introduzida pela transição t_5, que dispara toda vez que o intervalo de tempo entre os disparos das transições t_7 e t_8 ultrapassam um determinado limite. Prever o intervalo de tempo entre os disparos das transições t_7 e t_8 é factível na especificação, mas prever o tempo de transmissão de quadros usando o protocolo Ethernet como meio físico não é reprodutível na realidade.

Mesmo com uma sub-camada de controle de erros, não é possível garantir as ordem e tempo de envio em redes Ethernet (por princípio aleatório), como pode ser visto na

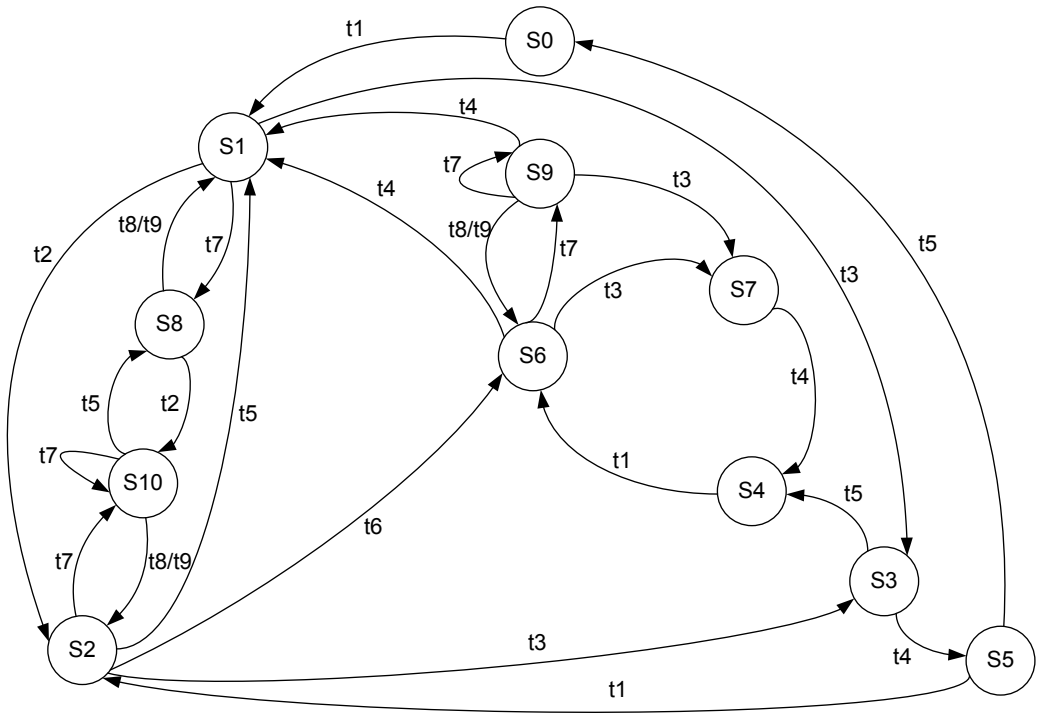


Figura 2.6: Autômato do Protocolo VRRP Estendido [Hashimoto 2009]

subseção 3.1.3. A rede de Petri, no entanto, demonstrou ter boas propriedades, produzindo o autômato mostrado na Figura 2.6. Podendo ainda ser refinado o modelo para obter melhores resultados.

2.2.4 Padrões e Diretrizes Arquiteturais para Escalabilidade de Sistemas

Esta Subseção, na realidade, não trata de uma proposta de melhoria em protocolos de alta disponibilidade. No entanto, resolveu-se incluí-la aqui, primeiro, porque as duas propostas anteriores resolveram apenas parte do problema, logo ela reflete a continuação da pesquisa. Segundo, devido ao relacionamento entre Alta Disponibilidade e Escalabilidade, cujo conceito é bem mais recente, logo trata-se de investigar se novas tecnologias já não haviam resolvido o problema.

Escalabilidade é a capacidade de processar cargas variáveis (em geral crescentes) de trabalho, mantendo, ou melhorando, satisfatoriamente a QoS (*Quality of Service*) especificada para o Sistema Escalável. À Escalabilidade interessa o subconjunto da QoS especificada por meio de Parâmetros Escalares (mensuráveis) que possam variar em função da carga (requisições de serviços).

Por exemplo, o parâmetro 'Tempo de Resposta', de um sistema que deva ser capaz de atender a todas as requisições em menos de oito segundos, não importando se há uma ou milhares de requisições em determinado instante. A palavra 'satisfatória' usada na

definição indica que um sistema somente será escalável se atender a algum parâmetro pré-definido, no caso, o requisito não funcional Tempo de Resposta (desempenho).

Alta Disponibilidade tem uma relação estreita com a Escalabilidade de Sistemas, sendo um dos requisitos de um sistema escalável. Contudo, Disponibilidade significa a probabilidade de um sistema estar disponível, enquanto Escalabilidade significa manutenção (ou melhora) de algum requisito não funcional (mensurável) ligado à QoS do serviço.

Mesmo havendo equipamentos para os quais Escalabilidade não se aplica totalmente, como é o caso dos Elementos de Rede (NE - *Network Element*), em particular os Roteadores, há alguns aspectos que devem ser considerados para que a Disponibilidade atinja seu objetivo. [Porto 2009] cataloga padrões e estabelece diretrizes para a Escalabilidade de sistemas, sendo que quatro deles merecem uma análise para o aspecto da Alta Disponibilidade em NEs.

Do ponto de vista da Segurança de Informação, Alta Disponibilidade é uma reação a 'fatores internos' que possam interromper a disponibilidade de serviços. Escalabilidade é também uma reação, mas a 'fatores externos' relacionados à variação da carga sobre o sistema que possam afetar a QoS (por exemplo, imprevisibilidade do tempo de reposta), levando inclusive à indisponibilidade.

A especificação da QoS para serviços não é uma tarefa trivial. É importante ter-se em mente que há requisitos não funcionais[10] mutuamente exclusivos, sendo os mais importantes, para o objetivo deste trabalho, aqueles introduzidos pelo Teorema CAP.

O Teorema CAP

O Teorema CAP [Brewer 2000] prova cabalmente que há três propriedades que não podem ser especificadas concorrentemente, sendo que, por brevidade, este trabalho ater-se-á à exposição das propriedades e não às provas.

O Teorema CAP (*Consistency, Availability and Partitioning*) prova que dadas as propriedades Consistência, Disponibilidade e Tolerância a Particionamento somente duas podem coexistir em um Sistema.

Neste teorema, o termo consistência é uma simplificação para Consistência Espacial. Um sistema é dito 'espacialmente consistente' se as cópias de objetos (dados por exemplo) existentes em dois ou mais nós forem 'iguais'. Ou seja, se dois ou mais nós compartilham dados, estes devem estar sincronizados (valores iguais) para que o sistema seja dito consistente.

Particionamento ocorre quando partes da infra-estrutura de rede perdem a capacidade de comunicação, tornando impossível as interações entre os nós do sistema. Tolerância a Particionamento é a capacidade de o sistema continuar operando mesmo quando há o particionamento da rede. A Figura 2.7 mostra o relacionamento entre as três propriedades,

[10] Em geral os parâmetros da QoS são requisitos não funcionais

representadas sobre um triangulo. Um sistema pode apoiar-se somente sobre uma das três arestas do triangulo e cada aresta une apenas duas das três propriedades do teorema. Logo, um sistema pode possuir apenas as propriedades cuja aresta na qual ele está apoiado une.

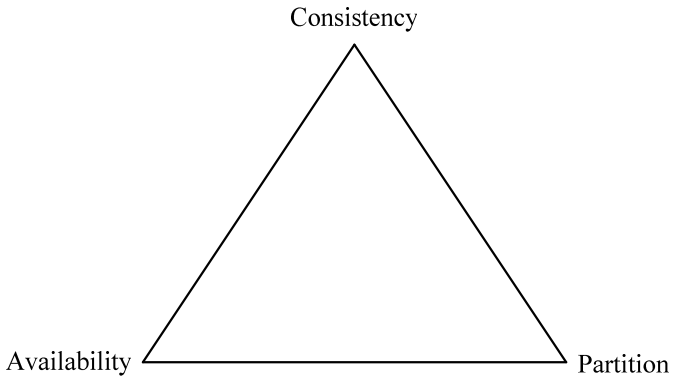


Figura 2.7: Teorema CAP [Brewer 2000]

Várias arquiteturas de sistemas escaláveis foram criadas com base nesse teorema, sendo que [Porto 2009], realiza as classificação e organização dessas arquiteturas, explicando os seus mecanismos e exemplificando o seu uso com situações hipotéticas. O resultado é um catálogo com padrões e diretrizes para a construção de aplicações escaláveis.

Padrões e Diretrizes

Embora existam diversas arquiteturas, que servem de base para o projeto de sistemas escaláveis horizontalmente[11], como citado anteriormente, este trabalho se concentra naquelas de interesse para os Elementos de Redes.

O trabalho de [Porto 2009] classifica os tipos de padrões de arquiteturas para o projeto de sistemas escaláveis horizontalmente, dentre os quais, para a finalidade deste trabalho, destacam-se os seguintes: SNA (*Shared Nothing Architecture*); BASE (*Basically Available, Soft state, Eventual Consistency*); SAGAS; e *Caches* Distribuídos.

Estas arquiteturas lidam com o que se chama de penalidade de persistência, para as quais, uma vez que a informação tenha de ser persistida, o sistema estará sujeito a: (i) dispender recursos computacionais para armazenar e recuperar a informação; (ii) sofrer restrições relacionadas a características do canal pelo qual a informação flui; e (iii) sofrer alterações na informação dependendo de como ela é armazenada ou transportada.

O padrão SNA auxilia na construção de sistemas facilmente escaláveis horizontalmente, a partir da estruturação do sistema em partes independentes, sem compartilhamento de

[11] Agregação de nós computacionais à infra-estrutura

estado (*Shared Nothing*), possibilitando escalabilidade linear.

O padrão BASE permite a construção de sistemas nos quais se troca a consistência de dados (cópias sincronizadas em nós remotos) por escalabilidade ou disponibilidade, através da construção de um sistema que é basicamente disponível, lidando com dados ligeiramente desatualizados e eventualmente consistentes.

O padrão *SAGAS* proporciona melhoria da escalabilidade e desempenho, evitando o uso de transações distribuídas e mantendo a consistência dos dados. Transações 'ACID[12] longas' são dividas em transações menores e ações compensatórias são definidas em caso de falha dessas transações menores para preservar a consistência dos dados.

O padrão Camada de *Caches* Distribuídos proporciona melhorias da escalabilidade horizontal e do desempenho através do uso de vários *caches* com o espalhamento dos fragmentos de dados entre os *caches*, evitando duplicação desnecessária de dados em memória.

Estes Padrões e Diretrizes são introduzidos rapidamente para dar uma base de conhecimento ao leitor, pois as Arquiteturas de Alta Disponibilidade precisam de armazenar e compartilhar Estados. Problemas na recuperação, nas transmissões ou nos armazenamentos locais levam às condições de Acéfalo ou Cérebro Partido. Deste modo, o armazenamento e a recuperação de Estados dos nós da infra-estrutura de alta disponibilidade dependem fundamentalmente das disponibilidades dos nós e da rede de interconexão.

Tendências para o Armazenamento de Dados Distribuídos

Sistemas gerenciadores de bancos de dados relacionais são conhecidos como a solução mais bem estabelecida quando o assunto é o armazenamento, modificação e recuperação de grandes quantidades de dados. Esse cenário vem mudando significativamente uma vez que sistemas escaláveis demandam não só uma quantidade crescente de dados e transações, mas também alta disponibilidade.

Nesse novo ambiente, os sistemas de gerenciamento de base de dados frequentemente se tornam um gargalo, e cedem lugar aos sistemas distribuídos de gerenciamento de dados, que implementam as arquiteturas catalogadas por [Porto 2009]. Além disso, Elementos de Rede não têm infra-estrutura de sistema operacional para suportar a instalação e configuração de um banco de dados relacional.

O padrão BASE [Pritchett 2008] chama a atenção por contrariar a lógica ACID, um princípio das bases de dados relacionais. Transações ACID significam: transações indivisíveis (**A**tomicidade); base de dados **C**onsistente após uma transação; transações simultâneas não devem provocar efeitos colaterais umas sobre as outras (**I**solamento); e **D**urabilidade, indicando que as alterações na base de dados devem ser duradouras.

Na busca por implementações da arquitetura BASE foram encontradas as arquiteturas

[12] ACID - Atomicidade, Consistência, Isolamento e Durabilidade

Bigtable [Chang et al. 2006], *Hypertable* [Hypertable 2010] e *Hbase* [HBase 2008], que são bases de dados distribuídas, escaláveis e de alto desempenho. Um componente essencial para o funcionamento dessas bases é o serviço de exclusão, que no caso da *Bigtable* é feito pelo *Chubby* [Burrows 2006]. Esse serviço tem como base o algoritmo de Paxos (ver seção 3.2).

Não é objetivo do presente trabalho detalhar as arquiteturas mencionadas, mas explicitar as bases que serviram de ponto de partida, sendo que os interessados em detalhes de seus projetos podem facilmente aprofundarem-se por meio das referências oferecidas. A análise destas arquiteturas de base de dados distribuídas é de especial interesse, pois o modo como elas trocam informações de estados são uma fonte importante de experiência na especificação dos serviços a serem propostos neste trabalho.

Capítulo 3

Estudo Comparativo entre os Protocolos VRRP, CARP e Paxos

Este capítulo apresenta uma análise dos protocolos VRRP, CARP e Paxos, sendo seu principal objetivo servir como ponto de partida para a criação de uma base de conhecimento para o projeto de novos protocolos na área de Alta Disponibilidade.

O aprofundamento da análise das características do protocolo Ethernet se deve ao fato se sua utilização ser muito comum por estes três protocolos. Será feita uma análise minuciosa sobre a relação entre "Alta Disponibilidade" e "Escalabilidade", motivados pelo uso de alta disponibilidade em elementos de rede, tais como roteadores, *Hubs*, *Switches*, multiplexadores, etc.

Como Elementos de Rede são uma classe de sistemas embarcados específicos para a interconexão de redes, eles não apresentam sistemas operacionais capazes de suportar totalmente o conceito de Escalabilidade, sendo, contudo, desejável a manutenção de certos requisitos de QoS (que é o objetivo de Escalabilidade).

Finaliza o capítulo, um estudo comparativo que mostra como as habilidades das camadas subjacentes afetam: a propriedade de Alta Disponibilidade dos protocolos VRRP e CARP; e a propriedade de Escalabilidade do protocolo Paxos.

3.1 Habilidades do Protocolo *Ethernet*

Esta seção reúne e discute as habilidades do Protocolo *Ethernet* (o mais utilizado protocolo da camada de Enlace em redes locais), cuja descrição é introduzida no Capítulo 2. Ressalte-se que os problemas relativos ao Controle de Erro (*Error Correcting Codes*) da camada de Enlace não são exclusividade do protocolo Ethernet, e acontecem em outros protocolos da camada de Enlace padronizados pelo IEEE, tais como os protocolos do comitê IEEE 802.

Essas características têm influência significativa nos protocolos de Alta Disponibilidade

existentes, sendo elas óbvias para administradores de rede experientes e acostumados a ambientes críticos.

No entanto, o fato de serem características de comunicação de baixo nível, torna-as difícil de serem percebidas, tanto na literatura quanto na vida real. Por exemplo, como separar os erros da Aplicação dos erros produzidos pelo meio físico e não detectados pela camada de Enlace?

Este trabalho reclama a contribuição de agrupá-las e discutí-las sob um novo ponto de vista que facilita o trabalho de: (i) administradores de rede menos experientes, na tentativa de usar os protocolos existentes para prover alta disponibilidade; e (ii) engenheiros de redes, ao projetar novos protocolos para prover alta disponibilidade.

3.1.1 Herança do Meio Compartilhado

O protocolo Ethernet foi originalmente desenvolvido para operar sobre meios físicos compartilhados (cabos Coaxiais e HUBs) e evoluiu para meios físicos comutados (*switches*). Todavia, algumas características que simulam o meio compartilhado foram criadas por questões de compatibilidade, como por exemplo, as transmissões do tipo *Broadcast* e *Multicast*, que possuem vantagens (economia de banda e redução da complexidade de protocolos) e desvantagens (acesso dos adaptadores de rede a transmissões que não lhe dizem respeito e potenciais ataques).

O fato do meio ser compartilhado facilita muito o aspecto do endereçamento de nós parceiros (*peers*) e permitiu desde o princípio no projeto ALOHA, que um adaptador de rede soubesse se o meio físico estava ou não em uso por outro nó. Um meio físico compartilhado significa ter sempre em um conjunto de n adaptadores, até $n-2$ (supondo que algum dos demais nós está em processo de recepção) adaptadores aguardando o término da transmissão corrente para iniciar suas respectivas transmissões.

Para otimizar o uso do meio físico, foram criados, então, dois mecanismos de endereçamento, que permitem que uma comunicação seja endereçada a mais de um nó da rede, sendo: *Broadcast* - todos os nós da rede são endereçados, inclusive quem enviou a primitiva; e *Multicast* - um subconjunto dos nós da rede são endereçados.

Esta possibilidade de receber comunicações, que são endereçadas a vários nós da rede, ou de enviar comunicações, a vários nós da rede, culminaram em uma fragilidades dos protocolos de Alta Disponibilidade, entre os quais o protocolo VRRP, cuja ausência de mecanismos de segurança entre os elementos de um sistema distribuído é o ponto fulcral de sua vulnerabilidade. É evidente que essa vulnerabilidade somente se tornará um problema se de *facto* houver uma ameaça. Para que não haja dúvida, há a necessidade de mecanismos de segurança tais como autenticação e criptografia.

3.1.2 Erros não Detectáveis

Embora ameaças sejam comuns em ambientes críticos, existe uma característica do protocolo Ethernet que muitos protocolos das camadas superiores não levam em consideração (por exemplo o Protocolo TCP). Isto se deve ao fato de o Modelo de Referência OSI atribuir o controle de erros de bits à camada de Enlace. Protocolos de camadas superiores podem ser capazes de detectar erros (caso dos protocolos orientados a conexão), mas não erros de bits.

Como já foi dito, o protocolo Ethernet não é capaz de detectar todos os erros que possam surgir em um quadro. Existe a probabilidade de erros não detectados passarem para a camada superior, mas, no entanto, esses erros ocorrem e dependem de fatores físicos ou externos tais como interfaces de rede de má qualidade ou a presença de campos eletromagnéticos, que introduzem erros em rajada. São fatores fáceis de ser percebidos, uma vez que o problema geralmente mantém um padrão de comportamento ao longo do tempo. Além disso, pode-se definir aspectos do projeto físico que diminuam a incidência de erros provocados por ruídos eletromagnéticos.

Como exemplo de erros não detectados pela camada de Enlace, é não raro alguém receber um e-mail com um arquivo em anexo, que apresenta erro ao se tentar abrir. Contudo, o reenvio do mesmo arquivo (se não for o caso de fonte danificada) se mostra um eficiente mecanismo de correção de erro. O que acontece na maioria das vezes é que houve um erro não detectado pela camada de Enlace. Infelizmente, quando se projeta sistemas autônomos, como é o caso dos protocolos de Alta Disponibilidade, não há a chance de se requerer o reenvio.

3.1.3 Ausência de Garantias de Tempo-Real

A sub-camada de Controle de Acesso ao Meio (MAC - *Medium Access Control*) do protocolo *Ethernet* implementa o método de acesso ao meio denominado CSMA-CD (*Carrier Sense Multiple Access-Colision Detection*), que é essencialmente aleatório. Isto é, por mais rápido que seja o equipamento e por mais rápido que a primitiva do protocolo de Alta Disponibilidade chegue à sub-camada MAC, ela aí permanecerá até que o meio físico esteja desocupado. Forma-se na sub-camada MAC uma fila FCFS (*First Come First Served*), sem mecanismos de prioridade ou de tempo-real.

Em outras palavras, uma vez que o protocolo de Alta Disponibilidade envie primitivas, que ficarão hospedadas na sub-camada MAC até que o meio físico esteja desocupado, não há como prever o instante de sua recepção, ou esperar que a recepção ocorra dentro de um intervalo de tempo, pelos nós parceiros. Isto torna o mecanismo de Temporização perigoso pelo risco de vencimento do temporizador (*Time Out*).

É justamente um mecanismo de temporização que o protocolo VRRP e a proposta de [Hashimoto 2009] utilizam para determinar se e quando um Roteador VRRP *Slave*

deve iniciar sua sequência de eventos para assumir o papel de *Master*.

Esse problema da imprevisibilidade do instante de acesso ao meio é bem mais comum (provocando o respectivo *Time Out* nos autômatos hospedados em nós parceiros) do que a probabilidade de uma primitiva conter erros não detectados, com os agravantes de: estar diretamente ligado a ambientes com tráfego intenso; e possuir comportamento não determinístico (aleatório).

3.2 Algoritmo de Paxos

O algoritmo de Paxos introduz a filosofia de "Consenso Distribuído" entre "Entidades Fracamente Acopladas" (*Loosely Coupled Entities*). Entenda-se por Entidades Fracamente Acopladas, entidades que se comunicam ou compartilham informações de Estado eventualmente e podem enfrentar o fenômeno do Particionamento com frequência.

O termo "Acoplamento" se refere ao grau de compartilhamento de estado que as entidades podem experimentar. Posto de outro modo, refere-se ao grau de conhecimento direto que uma entidade tem de outra entidade. Acoplamento Fraco dificulta o Consenso entre Entidades participantes, quando os meios de interconexão podem falhar.

O algoritmo de Paxos define uma sequência de passos que as entidades devem desempenhar para permitir que elas continuem de comum acordo (Consenso) a respeito de uma decisão (Estado). Esse acordo deve ser mantido mesmo se ocorrerem falhas em algumas entidades ou na presença de problemas de comunicação entre si.

Cada entidade participante deve guardar uma Lista de Decisões Tomadas. O primeiro requisito do protocolo de Paxos é a manutenção da consistência dessas listas, que enfatiza que duas listas não podem conter informações contraditórias sobre as Decisões Tomadas. Se a primeira decisão for a favor da opção 'a', então todas as Entidades devem apresentar a opção 'a' no primeiro lugar da lista.

No entanto, quando uma entidade falha, outra entidade é criada para substituí-la (a entidade que falhou). Essa nova entidade pode apresentar uma lista em branco (vazia), ou uma lista com algumas linhas em branco até que a entidade aprenda (anote) todas as decisões já tomadas.

Entretanto, manter a consistência das listas não é suficiente para atingir o consenso, uma vez que esse requisito pode ser atendido deixando todas as listas em branco (conteúdo nulo). Então, há necessidade de requisitos adicionais para garantir que decisões sejam tomadas e armazenadas nas listas. Essas decisões não podem ser validadas a menos que haja um número mínimo de Entidades quando ela for tomada e nenhuma Entidade falhe ou surja num período que antecede e prossegue o instante da decisão. Além disso, um número mínimo de Entidades deve sobreviver por tempo suficiente para que as decisões sejam transmitidas.

As entidades realizam eleições afim de escolher entre opções em determinada questão.

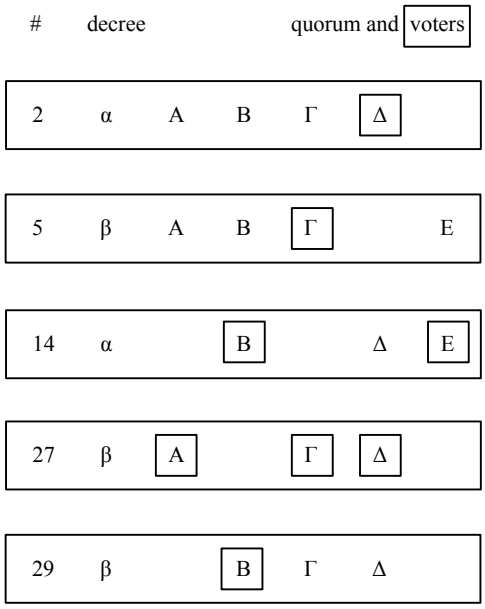


Figura 3.1: Conjunto de 5 Eleições que Satisfaz as Condições B1(β)-B3(β) [Lamport e Marzullo 1998]

Terminada a eleição a opção escolhida deve ser um consenso entre todas as entidades participantes. Cada entidade deve guardar o resultado de cada eleição em uma lista.

No artigo [Lamport e Marzullo 1998], Lamport usa uma metáfora para explicar as etapas da concepção do algoritmo. Segundo ele, em uma antiga ilha da Grécia, chamada Paxos, havia um parlamento que funcionava em tempo parcial. Devido à vocação mercantil dos habitantes da ilha, havia um fluxo constante de parlamentares entrando e saindo do parlamento, muitos sequer regressavam. Mesmo assim, um protocolo desenvolvido pelos matemáticos de Paxos mantinha as leis de Paxos sempre consistentes.

O protocolo do parlamento de Paxos evoluiu a partir do protocolo de um antigo cerimonial, onde sacerdotes se reuniam para escolher um decreto. Por séculos, o conselho escolheu o decreto usando um procedimento convencional, que exigia a presença de todos os sacerdotes. Mas à medida que o comércio florescia ficava mais difícil ter todos os sacerdotes presentes durante as cerimônias. De fato, não foi possível manter o ritual e um protocolo foi proposto para resolver esse problema.

O decreto era escolhido através de uma série de cédulas numeradas, sendo que cada cédula representava um referendo envolvendo somente um decreto. Em cada referendo, um Sacerdote tinha a chance de votar no decreto ou não votar. O conjunto de sacerdotes associados a cada célula era chamado de *Quorum*. Um referendo era considerado bem sucedido se cada sacerdote presente votasse no decreto. Formalmente uma cédula B era composta de quatro componentes:

- B_{dec}: decreto a ser referendado (ou não) pelos Sacerdotes;

- B_{qrm}: conjunto não vazio de Sacerdotes, também definido como *Quorum*;
- B_{vot}: conjunto de sacerdotes que votaram; e
- B_{bal}: número de cédulas.

Uma cédula B era considerada bem sucedida se $B_{qrm} \subseteq B_{vot}$. Os matemáticos de Paxos definiram três condições em um conjunto β de cédulas, mostrando que a consistência estava garantida e que o progresso era possível se o conjunto de cédulas satisfizesse às seguintes condições:

- B1(β): cada cédula em β contém um número único;
- B2(β): os *Quorums* de duas cédulas quaisquer em β têm pelo menos um sacerdote em comum; e
- B3(β): para cada cédula B em β, se qualquer sacerdote do *Quorum* de B votou em algum referendo anterior de β, então o decreto de B será igual ao decreto da célula mais recente entre as anteriores.

A figura 3.1 ilustra a condição B3(β) com cinco referendos realizados quando existiam cinco sacerdotes (A, B, Γ, Δ, e E). A figura mostra cinco cédulas representadas por retângulos, contendo o número, o decreto e o *quorum*, sendo que os sacerdotes que votaram aparecem dentro de um quadrado. Por exemplo, a cédula número 14, com o decreto α, possui um *quorum* de três sacerdotes (B, Δ, e E), sendo dois deles votantes (B e E).

A partir dessas três condições foi derivado um protocolo que permite a um sacerdote iniciar, conduzir uma eleição e divulgar o resultado em caso de sucesso. Este protocolo definiu seis passos, a saber:

1. O sacerdote p escolhe um novo número de cédula e envia uma mensagem *NextBallot(b)* para um conjunto de sacerdotes;
2. Um sacerdote q responde a mensagem *NextBallot(b)* enviando uma mensagem *LastVote(b,v)* para p, onde v é o voto da cédula mais recente antes de b que q efetuou, ou $null_q$ se q não votou em nenhuma cédula anterior a b;
3. Depois de receber uma mensagem *LastVote(b,v)* da maioria de sacerdotes do conjunto Q, o sacerdote p inicia uma nova eleição, com número b, *Quorum* Q, e decreto d, onde d é escolhido para satisfazer B3, anotando então as informações sobre a eleição em seu registro e envia uma mensagem *BeginBallot(b,d)* para todos os sacerdotes do conjunto Q;
4. Ao receber *BeginBallot(b,d)*, o sacerdote decide votar ou se abster em b (para não violar a promessa feita em *LastVote(b',v')*), sendo que, se q votar ele envia uma mensagem *Voted(b,q)* para p e anota o voto em seu registro;

5. Se p recebeu uma mensagem *Voted(b,q)* de cada sacerdote em Q, então ele anota d no seu registro e envia a mensagem *Success(d)* para cada sacerdote;
6. Ao receber a mensagem *Success(d)* o sacerdote anota o decreto d no seu registro.

O protocolo básico do Parlamento de Paxos mantém a consistência das decisões (sincronização de estados), mas não pode garantir a evolução do autômato, pois ele apenas estabelece o que um Sacerdote pode fazer, mas não especifica qualquer obrigatoriedade. É importante lembrar que originalmente os habitantes da ilha praticamente não se ausentavam. Então, sempre havia alguém apto a iniciar ou dar continuidade ao protocolo.

O protocolo completo possui os mesmos seis passos do protocolo básico, mas introduz requisito adicional de que os sacerdotes executem os passos de 2 a 6, o mais rapidamente possível. Evidentemente, a necessidade de alguém executar o passo 1, que inicia a eleição, continua. A chave está em determinar quando um sacerdote deve iniciar uma eleição.

É importante frisar que na democracia de Paxos, toda e qualquer decisão sobre assuntos relativos à coletividade era feita através de eleições. E as decisões tomadas nas eleições eram respeitadas rigorosamente.

Acreditava-se que não iniciar uma eleição (para alguma pretensão da ilha), significava travar o progresso. No entanto, iniciar muitas eleições também podia resultar no mesmo, isto é, não se traduzir em progresso. Deste modo, o progresso consistia em se garantir que eleições fossem iniciadas e conduzidas até que alguma obtivesse sucesso (uma decisão).

A frequência ótima de eleições (visando a maximização do progresso) depende da duração de uma eleição e a duração de uma eleição depende do tempo que uma mensagem leva para ir da origem (sacerdote remetente) ao destino (sacerdote destinatário). Como a evolução do tempo é primordial para a troca de mensagens, então, a condição básica para garantir o progresso é ser capaz de medir a passagem do tempo (uma abstração da progressão da eleição).

Alcançar a condição de progresso requer que novas eleições sejam iniciadas até que uma seja bem sucedida, e que as eleições não sejam iniciadas com uma frequência muito alta. Para chegar ao protocolo completo, os matemáticos de Paxos tiveram de descobrir quanto tempo era necessário para produzir e entregar uma mensagem. Eles determinaram que um mensageiro que não deixasse o templo[1] levaria cerca de 4 minutos para entregar uma mensagem, e um sacerdote que não deixasse o templo levaria cerca de 7 minutos para produzir uma mensagem. Assim, em condições normais o sacerdote p levaria 22 minutos para enviar uma mensagem para q e receber uma resposta (*Confirmed Service*).

Suponha que somente o sacerdote p inicie eleições e que o faça enviando mensagens a todos os sacerdotes no templo, conforme o passo 1 do protocolo. Se p iniciou uma eleição, quando a maioria dos sacerdotes estavam no templo, então ele espera atingir o 'Passo 3' em 22 minutos, após o início da eleição, e o 'Passo 5' após outros 22 minutos.

[1]Por exemplo, dedicação exclusiva ao templo

Se isto não acontecer (não for possível executar os passos mencionados nesses intervalos de tempo), então algum mensageiro ou sacerdote deixou o templo depois que p iniciou a eleição, ou uma eleição com um número maior já havia sido iniciada por outro sacerdote (antes de p se tornar o único sacerdote a iniciar eleições).

Para lidar com esta possibilidade, p tem de saber sobre todas as eleições com número maior que *lastTried[p]* usados por outros sacerdotes. Isso pode ser feito estendendo o protocolo para requerer que se um sacerdote q receber uma mensagem *NextBallot(b)* ou *BegunBallot(b,d)* de um sacerdote p com menor *nextBal[q]*, então deve enviar uma mensagem contendo *nextBal[q]* para p e o sacerdote p deve iniciar uma nova eleição com um número maior de votantes.

Ainda, assumindo que p seja o único capaz de iniciar eleições, suponha que ele deva iniciar uma nova eleição somente se: (i) ele não executou os passos 3 e 5 nos últimos 22 minutos; ou (ii) ele percebeu que outro sacerdote iniciara uma eleição com número maior de votantes. Se as portas do templo forem trancadas com p e a maioria dos sacerdotes dentro, então um decreto poderia ser aprovado e registrado por todos os sacerdotes em 99 minutos, isto é, levaria 22 minutos para p iniciar a próxima eleição, 22 minutos para saber que outro sacerdote já iniciara uma eleição com número maior e mais 55 minutos para executar os seis passos de uma eleição bem sucedida. Então a condição de progresso seria atingida se somente um sacerdote, que não deixasse o templo, iniciasse as eleições.

Portanto, no protocolo completo foi incluído um procedimento para escolher um presidente (sacerdote que iniciaria as eleições). O procedimento para a eleição de presidente, então, precisa atender a apenas o seguinte requisito: se ninguém entrar ou sair do templo em T minutos um dos sacerdotes pode se considerar o presidente.

Os matemáticos de Paxos decidiram que o presidente deveria ser o sacerdote cujo nome era o último em ordem alfabética. Cada sacerdote deveria enviar uma mensagem contendo o seu nome para os outros sacerdotes do templo a cada $T - 11$ minutos, e o sacerdote que não recebesse mensagens com um nome que viesse depois do seu por T minutos poderia de considerar o presidente.

3.3 VRRP, CARP, e Paxos

Nesta seção os protocolos VRRP, CARP, e Paxos são comparados em relação a quesitos como: licença de uso, balanceamento de carga, área de abrangência, tempo de convergência e recuperação, e sensibilidade ao tráfego.

Como já fora mencionado na Subseção 2.1.1, soluções de alta disponibilidade, e escalabilidade representam uma vantagem competitiva, possuindo portanto, alto valor agregado. Uma das maneiras de fazer o investimento em P&D retornar é através do licenciamento de uso da tecnologia.

Neste quesito o protocolo VRRP possui uma história controversa. Baseado em um

protocolo proprietário da Cisco, [Li et al. 1998], e desenvolvido pelo (IETF - *Internet Engineering Task Force*), o protocolo é alvo de litígios e ainda assim implementado pelos principais fabricantes de equipamentos de rede e implementado nos principais sistemas operacionais.

O protocolo CARP é, em parte, uma resposta da comunidade OpenBSD a esses conflitos e, em parte, inclui algumas melhorias, como ver-se-á nos outros quesitos. O algoritmo original de Paxos é altamente independente de sua aplicação e foi publicado sob domínio público. No entanto, o esforço da industria para tirar proveito do algoritmo levou ao surgimento de derivações e implementações, sobre as quais não se conhece todos os detalhes.

Balanceamento de carga é uma característica vital para se prover escalabilidade linear [Porto 2009], entretanto, a ausência de balanceamento não é um impedimento à provisão de alta disponibilidade, o que justifica o fato de protocolos como o [Li et al. 1998] e [Hinden 2004] não fornecerem soluções completas de balanceamento de carga.

No caso do VRRP, o balanceamento de carga pode ser obtido configurando subconjuntos de roteadores para pertencer a roteadores virtuais distintos. E configurando parte dos clientes para usar um dos roteadores virtuais e parte para usar o outro. Esse tipo de solução possui a desvantagem de requerer intervenção manual (para configurar o gateway padrão de cada cliente), ficando a cargo do administrador de rede determinar quanto de carga cada roteador virtual deve receber. Uma alternativa é utilizar protocolos de configuração automática de clientes como o DHCP [Droms 1999], note que ambos os casos aumentam a complexidade e vulnerabilidade da solução.

O protocolo CARP inclui algumas melhorias no quesito balanceamento de carga. Com ele é possível determinar se o balanceamento será feito com base no endereço MAC ou IP do remetente, usando apenas a configuração do protocolo. Neste caso, embora não seja uma solução de balanceamento perfeitamente justa, é uma solução completa no sentido de não depender de fatores externos ao protocolo, o que minimiza as vulnerabilidades.

O algoritmo de Paxos não lida diretamente com a questão do balanceamento de carga, mas pode ser usado para manter a informação sobre um balanceamento de carga (quem deve receber qual porcentagem da carga, por exemplo) de forma distribuída e consistente, possibilitando a negociação de transferência de carga entre os elementos de um sistema distribuído.

No quesito área de abrangência, enquanto os protocolos que nasceram da alta disponibilidade, como VRRP e CARP, permanecem restritos às LANs, embora tecnologias como VLAN e VPN permitam que as LANs se estendam a grandes distâncias, sob pena de redução da disponibilidade oferecida devido ao atraso de propagação. Mesmo em LANs de curta distância, um grande número de clientes conectados à rede também pode levar penalidades devido a aprendizagem de endereços MAC.

Neste quesito o algoritmo de Paxos é extremamente flexível, podendo ser usado em

	VRRP	CARP	Paxos
Licença de uso	Publicado inicialmente sob domínio público pelo IETF e atualmente com direitos autorais reclamados pela Cisco devido a semelhança com o HSRP.	Publicado pela comunidade OpenBSD sob domínio público como uma resposta a reclamação da Cisco.	Publicado sob domínio público, possui diversas derivações e implementações.
Balanceamento de carga	Não oferece uma solução completa	Permite balancear pelo MAC ou IP do remetente.	Não lida diretamente mas permite a implementação em sistemas distribuídos
Área de abrangência	Projetado para operar em LANs	Projetado para operar em LANs	Projetado sem uma preocupação com abrangência os nós podem estar espalhados em desde uma pequena LAN até pela Internet.
Tempo de convergência e recuperação	Se torna operacional em milésimos de segundo	Se torna operacional em milésimos de segundo	Varia de alguns segundos a alguns minutos dependendo da área de abrangência do sistema distribuído
Sensibilidade ao tráfego	Alta devido ao baixíssimo tempo de convergência e recuperação	Alta devido ao baixíssimo tempo de convergência e recuperação	Baixa e média, dependendo da área de abrangência e do tempo de convergência

Tabela 3.1: Comparação entre os Protocolos VRRP, CARP, e Paxos

nós espalhados desde de pequenas LANs até a Internet. Neste caso um alto número não é considerado um problema, ao contrário de um baixo número de nós, que pode impedir o algoritmo de progredir. Paxos possui um horizonte de tempo bem mais largo, se comparado aos protocolos VRRP e CARP, logo a influência da distância tem um impacto menor.

Uma vez iniciados, esses protocolos levam um período de tempo para o *cluster* se tornar funcional e diante de falhas levam algum tempo para reagir e mascarar a falha. Esses períodos tempo são chamados respectivamente de tempo de convergência e tempo de recuperação.

Equipamentos que implementam as funções mais baixas das arquiteturas de rede lidam com funções que se repetem a cada segmento (encaminhamento de pacotes ou quadros, por exemplo). Embora isto seja transparente para um usuário, sua conexão fim-a-fim passa por vários saltos (*hops*). Os protocolos de alta disponibilidade devem atuar em cada salto e as decisões devem ser tomadas em um espaço de tempo muito curto, uma vez que a soma dos períodos em cada salto é que resulta na percepção de indisponibilidade.

No caso do Paxos, não raramente, ele é o usuário de fim-a-fim, o que o coloca em uma perspectiva de tempo totalmente diferente. Enquanto o VRRP e o CARP geralmente trabalham para manter os pontos de salto disponíveis, os nós do Paxos podem trabalhar separados por vários pontos de salto, e como as mensagens do Paxos exigem confirmação, o tempo gasto para transmitir uma mensagem é dobrado.

Tanto a alta disponibilidade quanto a escalabilidade são características exigidas em ambientes críticos, onde alto volume de transações e de tráfego fazem parte do melhor caso. Portanto, o último quesito de comparação é a sensibilidade ao tráfego. Tanto no caso dos protocolos VRRP e CARP quanto no caso do algoritmo de Paxos, o tráfego de controle (comunicação gerenciamento entre os nós) e o tráfego de dados coexistem.

No primeiro caso a disputa ocorre entre tipos de tráfego diferentes com tempos de vida diferentes, enquanto no segundo a disputa ocorre entre tipos de tráfego diferentes com tempo de vida igual. Uma mensagem de anúncio que chega depois que o intervalo de anúncio expirou não tem mais serventia, enquanto uma conexão TCP continua ativa mesmo que um pacote demore poucos minutos para chegar. Neste sentido os protocolos de alta disponibilidade aqui comparados estão claramente prejudicados, donde se lhes atribui alta sensibilidade ao tráfego e baixa ou média sensibilidade ao Paxos, a depender de também aqui comparados.

Esta comparação não tem a intenção de classificar os protocolos, muito menos de eleger um como o melhor. O foco se situa mais nos pontos de comparação que nos objetos comparados propriamente ditos. Esses pontos de comparação influenciarão nas decisões de projeto, algumas descritas no Capítulo 4 a Tabela 3.1 resume a comparação.

3.4 Alta Disponibilidade versus Escalabilidade

Como já foi mencionado anteriormente, existe uma relação entre alta disponibilidade e escalabilidade. Especificamente para a escalabilidade horizontal [Porto 2009], na qual diversos nós de processamento são disponibilizados para atender às requisições de serviços, além da propriedade relativa ao desvio padrão no tempo de resposta, pode-se dizer que a escalabilidade é uma extensão da alta disponibilidade, ou ainda, que a escalabilidade é um super conjunto da alta disponibilidade. Isto é, dado um conjunto de equipamentos, há uma grande probabilidade que haja um nó disponível para atender à próxima indicação (requisição entrante).

Por exemplo, servidores tendem a degradar o tempo de resposta em função do volume de transações, sendo que escalabilidade (*Scalability*) é a habilidade que os sistemas escaláveis apresentam de oferecer desvio padrão próximo de zero no tempo de resposta mesmo na presença de alto volume de transações. Se há garantia de resposta, ainda que a restrição de manter o desvio padrão do tempo de resposta próximo a zero não seja atendida, haverá alta disponibilidade. Em outras palavras, a garantia de resposta já é suficiente para prover alta disponibilidade, enquanto que a escalabilidade requer a manutenção (ou melhora) do tempo de resposta.

A degradação no tempo de resposta em função do volume de transações se explica pela necessidade de alocação e liberação de recursos (alocação de memória, abertura de arquivos, processamento, etc.), manipulação das estruturas de dados que gerenciam esses recursos (tabela de páginas, fila de processos), bem como no chaveamento de contexto dos processos das respectivas requisições concorrentes. Essas tarefas são típicas do sistema operacional, o que torna essa relação de difícil percepção mesmo para desenvolvedores experientes.

	Alta Disponibilidade	Escalabilidade
Hardware	Especializado/Otimizado.	De propósito geral.
Rendimento	Depende apenas da capacidade do hardware	Depende da capacidade do hardware e também da capacidade do software (sistema operacional) em empregar o máximo de recursos na execução da tarefa e o mínimo no gerenciamento (criação e término de processos, chaveamento de contexto).
Objetivo	Responder todas as requisições de serviço	Responder todas as requisições de serviço dentro de um tempo previsto independente do número de requisições.

Tabela 3.2: Alta Disponibilidade versus Escalabilidade

No entanto, segundo o conceito de Computação Desnuda (*Bare Computing* [Engler e Kaashoek 1995]), elementos de rede (tais como pontes, hubs, switches, roteadores, firewalls, gateways, etc.) tendem a reduzir o número de camadas de abstração entre o sistema operacional e o hardware, ou a remover completamente o sistema operacional, visando aproveitar todo o desempenho que o hardware tem a oferecer para realizar a tarefa da aplicação. Isso só é possível porque esses dispositivos realizam tarefas altamente especializadas, possuindo frequentemente, um hardware otimizado para realizar determinada tarefa.

Por exemplo, um Roteador realiza a tarefa de verificar os pacotes na fila de ingresso, que foram recebidos por meio de suas interfaces de rede, colocando-os nas filas de egresso das interfaces de rede, para que atinjam o destino pretendido. Ao invés de alocação e liberação de recursos há a ocupação e desocupação à medida que os pacotes são processados (roteados) e a consequente manipulação das estruturas de dados, que gerenciam esses recursos, tornando o processo mais rápido do que uma máquina de propósito geral o faria (não há chaveamento de contexto de processos, por exemplo), uma vez que o processador possui as instruções específicas para a realização do roteamento.

É evidente que a ausência de um sistemas operacional (gerenciamento de arquivos, processos, entrada e saída, entre outras facilidades) torna a construção de programas para os elementos de rede uma tarefa mais complexa. Mas o tempo (processamento) tomado por tais infra-estruturas em muitos casos é proibitivo.

Gerenciamento de recursos mais específico (e frequentemente mais simples) e ausência de chaveamento de contexto leva a um consequente aumento de desempenho do elemento de rede (aumentando, por exemplo, a capacidade de encaminhamento de pacotes de um roteador) e, o mais importante, a uma degradação muito pequena na presença de um alto volume de requisições (transações). Essa pequena degradação, mesmo na presença de um alto volume de transações, é atingida não somente às custas da eliminação do sistema operacional, mas também de uma arquitetura de hardware guiada por padrões de escalabilidade, como o *Shared Nothing* [Porto 2009], de maneira que esses dispositivos dispensem o uso de software para prover escalabilidade.

No entanto, esse dispositivo de alto desempenho apresenta o problema de ser um ponto único de falha (SPoF - *Single Point of Failure*) e, sua falha, pode reduzir o serviço eficiente prestado por ele a um serviço indisponível. Como já discutido na Seção 2.1, a melhor maneira de eliminar um SPoF é substituí-lo por componentes redundantes. Essa substituição, especialmente para os elementos do núcleo da rede, requer a adição de um componente de software para gerenciar essa redundância, resultando em um dispositivo único (virtual) altamente disponível. Pode-se dizer que os fenômenos denominados de condição de Acéfalo e de condição de Cérebro Partido merecem cada dia mais atenção pelos seguintes motivos:

- a necessidade de eliminar os pontos únicos de falhas (SPoF) nos elementos do núcleo

da rede, considerada a demanda por serviços de alta disponibilidade;

- a consolidação do 1 Gigabits por segundo, o surgimento do 10 Gigabits por segundo e a aspiração do protocolo Ethernet à tecnologia de enlace das redes WANs [Gorshe et al. 2004] [Meddeb 2005]; e
- o alto volume de requisições de serviços críticos no núcleo da rede.

De fato, estes três itens levam ao pior cenário que se pode imaginar para desencadear o problema de Cérebro Partido. Elementos de Rede redundantes interligados pelo protocolo Ethernet e a presença de um alto volume de requisições de serviços (transações).

Considerando que em essência as condições de Acéfalo e Cérebro Partido ocorrem devido a um problema de consistência distribuída, então, a aplicação do algoritmo de Paxos pode ser de grande valia.

Isto é, se todos os nós do *Cluster* tivessem sempre cópias idênticas (consistência espacial), sobre o estado global do aglomerado, seria fácil tomar decisões a respeito de quem deveria ser, ou deixar de ser, o nó *Master*. A próxima seção apresenta uma especificação de Serviços para um novo Protocolo de Alta Disponibilidade capaz de operar confiavelmente neste cenário.

Capítulo 4

Elementos para um de Protocolo de Alta Disponibilidade

Embora Escalabilidade seja um super-conjunto de Alta Disponibilidade, o exposto até o momento demonstra que há uma categoria de dispositivos, em particular os Elementos de Rede, para os quais não há como prover escalabilidade em função da especificidade e da demanda de processamento de entrada e saída.

Visto que a alta disponibilidade é uma característica extremamente importante e que há necessidade de serviços de alta disponibilidade com qualidade superior aos existentes, este capítulo introduz os conceitos de serviço, primitivas e pontos de acesso.

Apresenta suposições a respeito do ambiente onde um serviço de alta disponibilidade deve operar e como os protocolos de alta disponibilidade podem resolver os desafios que surgem nesses ambientes.

Estas suposições levam à definição de uma especificação de serviço de alta disponibilidade em alto nível de abstração, que é apresentada na forma de modelo, primitivas, parâmetros, valores e pontos de acesso.

4.1 Conceito de Serviço

Segundo [Holzmann 1991], projetar um protocolo é um processo complexo. Depois de escolher um meio de transmissão deve-se determinar um conjunto de regras que permita usá-lo, definindo como uma mensagem é codificada e como uma transmissão é iniciada e terminada. Dois tipos de erros são difíceis de evitar durante esse processo: criar um conjunto de regras incompleto, ou criar regras contraditórias. Para guiar o projetista durante o processo [Holzmann 1991] definiu cinco elementos essenciais. Para ser completa cada especificação de protocolo deve incluir explicitamente:

1. O **serviço** provido pelo protocolo;
2. As **suposições** a respeito do ambiente em que o protocolo será executado;

3. O **vocabulário** de mensagens usadas para implementar o protocolo;
4. A **formato** de cada mensagem do vocabulário;
5. As **regras** que mantém a consistência nas trocas de mensagens.

[Vissers e Logrippo 1985] faz a seguinte definição: O modelo mais comum para a especificação de protocolos mostra algumas entidades, os "usuários de serviço" (*user*), se comunicando via outras entidades, os "provedores de serviço" (*provider*). Essa comunicação entre entidades segue um conjunto de regras, chamado popularmente de "protocolo". Contudo neste trabalho toda vez que for citada a palavra protocolo ela ensejará os elementos introduzidos por [Holzmann 1991]. Note que [Vissers e Logrippo 1985] cita dois dos elementos enumerados por [Holzmann 1991].

Para se comunicar via um provedor de serviço uma entidade utiliza as tão faladas "primitivas de serviço". Uma primitiva de serviço pode ser considerada uma interação elementar entre o usuário e o provedor de serviço, durante a qual os valores para vários parâmetros da primitiva são estabelecidos, e aos quais ambos, usuários e provedores, podem se referir. As interações são executadas na "fronteira comum" (interface) entre o usuário e o provedor de serviço, sobre a qual residem os pontos de acesso ao serviço (SAP - *Service Access Point*).

Uma vez que esses SAPs representam pontos de acesso nos sistemas reais, as primitivas de serviço devem ser definidas e expressadas no mais alto nível de abstração, de modo a não restringir qualquer implementação válida. Isso implica que o desenvolvedor pode usar qualquer mecanismo que ele considere útil para invocar as primitivas de serviço, como chamadas de procedimento ou interfaces de hardware.

Um provedor de serviço é, então, visto como uma máquina abstrata (*Protocol Machine*) acessível a partir de alguns pontos de acesso (SAPs). A invocação de uma primitiva de serviço geralmente resulta na invocação de outra primitiva de serviço, cujos valores dos parâmetros podem depender dos valores dos parâmetros da primitiva que a invoca. A máquina abstrata também é capaz de realizar ações espontâneas (*Event Notification*) que resultam na requisição de serviços. Assim, a especificação de um serviço pode ser expressa em termos de passiveis ordens de primitivas de serviço e suas dependências de valores de parâmetros. Como essa maneira de especificar o serviço não revela nenhum detalhe interno do provedor de serviço, é frequentemente chamada de especificação observacional ou extensional: ela define o comportamento do provedor na visão dos seus usuários.

As entidades de protocolo da Figura 4.1 podem ser consideradas como uma camada de funções sobre o provedor de serviço. Em arquiteturas multi camadas, como a arquitetura OSI, um provedor de serviço pode ser construído usando entidades (N)-protocolo (protocolo da camada N), que se comunicam usando o provedor de serviço (N-1)-serviço (serviço da camada inferior em relação a N). A execução do (N)-protocolo pelas entidades (N)-protocolo usando o serviço (N-1)-serviço realiza o serviço (N)-serviço. Em outras

palavras, um (N)-serviço sempre pode ser descrito como resultado da ação combinada do (N-1)-serviço e do (N)-protocolo. Provavelmente essa maneira de descrever o (N)-serviço deve ser chamada de especificação do (N)-protocolo já que ela provê uma base arquitetural muito melhor para a especificação e verificação de um protocolo que os conceitos vagos geralmente usados. Em contraste com a abordagem extensional mencionada acima essa maneira de especificar o serviço é chamada de intencional ou generativa, pois revela detalhes internos sobre o provedor de serviço.

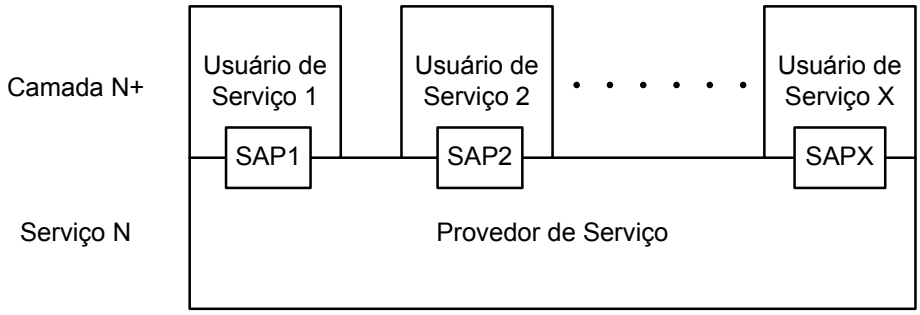


Figura 4.1: Modelo de Serviço [Vissers e Logrippo 1985]

A decomposição acima pode ser repetida para o (N-1)-serviço, (N-2)-serviço, e assim por diante, resultando em um sistema de protocolos em camadas com um conjunto de serviços aninhados. (Note que o Modelo OSI, aparentemente, não reconhece o meio físico como um serviço.) Portanto, também encontrou-se a seguinte definição: "A especificação do (N)-serviço define o comportamento global das (entidades nas) camadas abaixo da camada (N+1) de acordo com o que as (entidades nas) camadas (N+1) podem observar".

O exposto acima implica que existem a princípio duas formas de especificar o serviço:

- abordagem extensional, que será chamada especificação (N)-serviço e,
- abordagem intencional, que será chamada especificação (N)-protocolo.

Felizmente, mas também necessariamente, o (N)-serviço pode quase sempre ser descrito de maneira bem mais simples que o (N)-protocolo, como é mostrado nos dois exemplos a seguir:

Como primeiro exemplo, considere duas entidades da camada (N+1) que trocam fluxos de mensagens. O provedor (N)-serviço que elas precisam para realizar as trocas pode ser descrito em termos bem simples, por exemplo, um canal simples que não perde nem "inventa" mensagens. O (N)-protocolo necessário para realizar este serviço simples pode, no entanto, ser bastante complexo, se por exemplo o provedor (N-1)-serviço pode perder ou criar mensagens.

Como segundo exemplo, suponha que duas entidades (N+1) precisem de um provedor (N)-serviço que, além de não perder ou criar mensagens, não altere a ordem e tenha capacidade de armazenar algumas mensagens. Novamente esse provedor de serviço

pode ser descrito simplesmente como uma fila FIFO. Suponha, no entanto, que esse serviço seja executado usando provedores (N-1)-serviço que reordenem as mensagens. Um (N)-protocolo óbvio para executar o (N)-serviço desejado é o que verifica a ordem das mensagens numerando-as de algum modo. Então mecanismos de armazenamento ou retransmissão podem ser usados para corrigir as falhas na ordem.

Ainda segundo [Vissers e Logrippo 1985], nos dois exemplos, pode-se ver que uma descrição de serviço simples esconde uma variedade de descrições de protocolo mais complexas. Uma característica de sistemas de comunicação de dados bem definidos é exatamente que a descrição de um serviço pode ser muito mais simples que a descrição dos protocolos e serviços da camada inferior combinados.

4.2 Suposições Sobre o Ambiente

Enquanto [Holzmann 1991] frisa que as ações realizadas durante a execução dos serviços dependem das suposições feitas sobre o ambiente em que o protocolo executará, [Vissers e Logrippo 1985] defende que abordagem extensional, ou especificação (N)-serviço, é, na maioria das vezes, mais simples que a abordagem intencional, ou especificação (N)-protocolo. Nesta dissertação adota-se a especificação (N)-serviço, cujos resultados serão mostrados na próxima seção. Nesta seção, no entanto, a discussão ganha um viés de especificação (N)-protocolo, uma vez que se preocupa com uma série de detalhes internos, servindo como um degrau para chegar à abordagem extensional.

O Teorema CAP, discutido na Subseção 2.2.4, não se aplica apenas ao ambiente Web, de onde foi originalmente derivado, mas a qualquer sistema distribuído. Conforme a arquitetura apresentada na Seção 2.1, o foco deste trabalho está em um componente para prover um serviço de alta disponibilidade, cuja função é detectar o mau funcionamento em elementos do conjunto de dispositivos redundantes e decidir qual deles deve assumir a tarefa.

Um sistema pode tornar-se distribuído por diversas razões: (i) organização (redução da complexidade de construção e manutenção); (ii) desempenho (quando um único dispositivo é incapaz de atender a demanda usa-se vários em conjunto para atingir o nível desejado); ou ainda alta disponibilidade (que é o caso deste trabalho), para não estender a enumeração. A natureza distribuída do sistema torna o componente que provê o serviço de alta disponibilidade distribuído, por consequência, uma vez que ele envolve todos os nós do sistema, e como tal, está sujeito ao Teorema CAP.

Do ponto de vista do usuário de um sistema a execução de um serviço é simples e o usuário tem como preocupações apenas enviar uma requisição e aguardar uma resposta. Do ponto de vista interno do sistema, geralmente devido à complexidade, ela requer a divisão do problema em várias etapas. Essa divisão obriga o sistema a manter informações sobre a execução do serviço, de outro modo não seria possível distinguir quais etapas já

foram realizadas e quais ainda estão pendentes.

Essas informações devem persistir durante a realização do serviço, uma vez que representam o estado do atendimento da requisição. Em sistemas centralizados é fácil manter a consistência deste estado, mas ao se considerar um volume grande de transações, a velocidade em que essas informações podem ser lidas ou escritas constitui-se invariavelmente em um gargalo. Uma das maneiras de resolver este problema é distribuir a tarefa, de maneira que as requisições (ou parte delas) sejam atendidas em nós diferentes.

Essa solução é o que se pode chamar de sistema distribuído, cuja vantagem está em obter um desempenho relativamente superior ao do sistema centralizado, e a desvantagem está em perder a facilidade para manter a consistência do estado. O Roteador Virtual do protocolo VRRP é um sistema distribuído que não mantém o estado global. Apesar de cada Roteador VRRP saber do próprio estado, o Roteador Virtual não mantém o estado de todos os nós e nem usa essa informação para prestar o serviço.

Alguns sistemas necessitam de persistência mesmo após a execução do serviço. Devido à hierarquia de memória na arquitetura dos sistemas computacionais, quanto maior o grau de persistência, mais o serviço é penalizado. Com relação à informação, essa penalidade pode aparecer tanto na forma de inconsistência, quanto na forma de indisponibilidade.

As arquiteturas [Porto 2009] para sistemas escaláveis fornecem escalabilidade atacando essas penalidades. A arquitetura *Shared Nothing* lida com penalidade de persistência na memória principal, as arquiteturas *Sharding*, Base e Sagas lidam com penalidade de persistência na memória secundária, e a arquitetura de Caches Distribuídos lida com a penalidade de persistência em vários níveis.

A persistência vem da necessidade de manter informações sobre os usuários de um serviço durante uma sessão (ou sobre o serviço executado mesmo após o término da sessão). Durante a pesquisa chegou-se a considerar um projeto de arquitetura de serviço de sessão, uma vez que essa funcionalidade foi identificada como a mais crítica nas aplicações escaláveis modernas. No entanto, percebeu-se em um estágio muito precoce que nem todas as aplicações requerem todas as funcionalidades do serviço de sessão, sendo que cada aplicação possui suas particularidades, que transformariam esse serviço em um entrave à escalabilidade do próprio sistema.

Restou desse estudo a conclusão de que todos os serviços de sessão requerem dois componentes principais: (i) um para a manipulação da sessão, que trata do estabelecimento, manutenção e encerramento das sessões; (ii) um para persistência de informação, que trata do armazenamento e recuperação de dados relacionados a decisões que o usuário tomou ou situações que o usuário atingiu.

Do ponto de vista de um autômato qualquer mudança de estado depende da transição de um arco e a transição de um arco só acontece quando o evento que está relacionado ao arco é detectado. Esse evento pode ser a indicação ou confirmação de um serviço (primitiva), uma mudança de estado interno (informação, recurso de hardware), ou a pas-

sagem de determinado período de tempo (timeout). Esses eventos podem ser classificados em três grupos: tempo, comunicação, e estado do serviço. Os eventos do grupo tempo possuem o tempo como restrição.

Por exemplo, se nenhum evento ocorrer dentro de certo prazo, ou se algum evento só puder ser disparado depois de certo prazo, são situações consideradas evento de tempo. Os eventos do grupo comunicação envolvem o envio ou recebimento de primitivas. Por exemplo, a requisição, indicação, resposta, ou confirmação de um serviço. Os eventos do grupo estado de serviço envolvem qualquer alteração que possa afetar o andamento do atendimento de uma requisição de serviço ou o conteúdo da resposta que será enviada do requisitante.

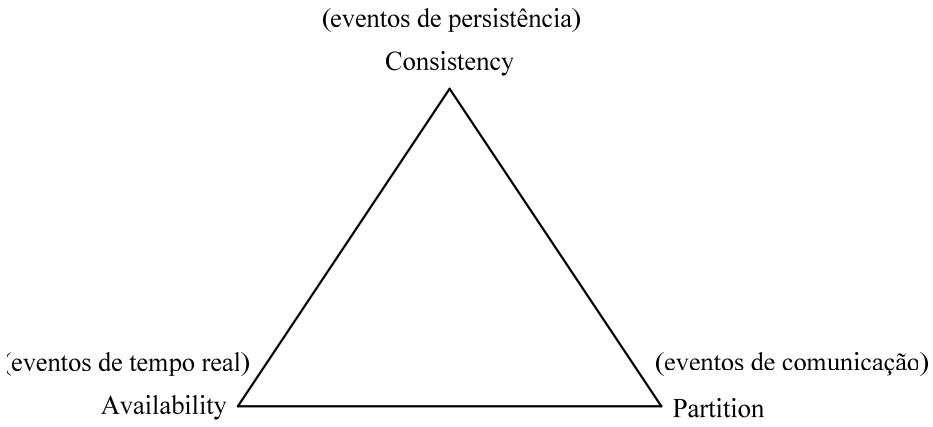


Figura 4.2: Relação entre Eventos de um Protocolo e o Teorema CAP

Cada grupo de eventos está estritamente relacionado às propriedades do teorema CAP, como mostra a Figura 4.2. Os eventos de tempo estão relacionados com a propriedade de disponibilidade, os eventos de comunicação estão relacionados com a propriedade de tolerância a partição da rede, e os eventos de estado de serviço estão relacionados com a propriedade de consistência.

O protocolo VRRP permite a ocorrência dos fenômenos acéfalo e cérebro partido e ainda entrega o serviço ao qual se propõe, minimiza os períodos não funcionais fornecendo a maior disponibilidade possível. Ele usa um autômato de três estados e apenas eventos dos grupos comunicação e tempo podem provocar mudanças de estado nesse autômato. O VRRP abdicou das propriedades de consistência em favor das propriedades de tolerância a particionamento da rede e disponibilidade. Assim, o serviço apresenta a propriedade de disponibilidade enquanto não é necessário tolerar partições na rede.

O intervalo de tempo entre o envio das mensagens de anúncio é que determina quanto tempo fenômenos como acéfalo e cérebro partido podem durar. Quando não há necessidade constante de comunicação algoritmos de inteligência artificial podem ser usados, como mostra [Sen et al. 1998]. A desvantagem é que esses algoritmos possuem um tempo

de convergência alto e por vezes não há garantia de convergência.

Tendo dois computadores interligados via uma rede Ethernet e cada um deles podendo mostrar dois valores na tela, zero ou um. Sempre que o serviço for iniciado cada computador sabe qual valor deve ser mostrado inicialmente na tela e qual deles deve iniciar a troca de mensagens (isso nos poupa o esforço de definir os detalhes que evitam o *starvation* na inicialização).

O computador que inicia ou está com o valor um envia uma primitiva pela rede e altera o seu valor para zero. A mensagem sempre deve ser enviada dentro de um período de tempo aleatório e finito após o computador mostrar o valor um. O computador que inicia ou está com o valor zero deve aguardar o recebimento da mensagem e então alterar o valor para um. Durante um segundo é aceitável que os computadores mostrem valores iguais.

É fácil perceber que a regra de mostrar valores iguais por no máximo um segundo será violada se for possível interceptar ou injetar mensagens falsas no meio de comunicação. O mesmo raciocínio é valido para mensagens descartadas ou corrompidas. Fora esses casos esse serviço deve funcionar sem problemas em ambientes de rede não críticos. Em um ambiente crítico o comportamento da rede torna-se menos previsível fazendo com que esse serviço simples apresente baixa qualidade.

Uma hipótese seria usar uma abordagem oposta à do protocolo VRRP, considerar a consistência espacial como condição obrigatória para o serviço de encaminhamento de pacotes, permitindo a ocorrência de acéfalos e cérebros partidos que serão resolvidos quando a base estiver sincronizada.

A impossibilidade de garantir tempo real é uma característica marcante do protocolo Ethernet, e ainda assim pouco conhecida por administradores de rede experientes. Isto se dá, primeiro, pelo fato de haver sempre um mal entendimento do conceito de tempo real. Tende-se a interpretar a expressão tempo real como algo que pode ser resolvido rapidamente, instantaneamente, quando na verdade está relacionado com a precisão na medição de tempo. O protocolo Ethernet não pode garantir que uma mensagem chegará instantaneamente, ou chegará exatamente em determinado tempo, ou chegará depois de determinado tempo.

O uso de uma rede (interfaces) Ethernet dedicada para o tráfego do controle do *cluster* vai na contra-mão dos conceitos de virtualização e computação em nuvem [Chowdhury e Boutaba 2010], que estão ganhando cada vez mais espaço. O compartilhamento de recursos, a unificação de gerenciamento, e a redução de custos são características imprescindíveis para se estabelecer no mercado. O próprio protocolo Ethernet cuja simplicidade é o principal motivo para justificar o seu alastramento.

4.3 Especificação do Serviço de Alta Disponibilidade

O trabalho de [Hashimoto 2009] contém uma proposta do quinto elemento definido por [Holzmann 1991], isto é, o autômato, para o novo protocolo de Alta Disponibilidade que essa pesquisa vislumbra. Geralmente o quinto elemento é a última parte da especificação de um protocolo, mas o elemento proposto é baseado em um protocolo que já existe, o que justifica a inversão da ordem. Como a especificação deve conter todos os elementos para ser considerada completa, fica clara a validade da nossa preocupação nesta dissertação com a especificação do serviço.

O objetivo final da pesquisa é desenvolver um protocolo que forneça um serviço de alta disponibilidade semelhante ao dos protocolos citados na Subseção 2.1.1, capaz de gerenciar dispositivos redundantes conforme a arquitetura descrita na Seção 2.1, apresentando ainda qualidade superior no tratamento de problemas como a Condição de Acéfalo (Subseção 2.1.3) e de Cérebro Partido (Subseção 2.1.4). Este protocolo justifica o objetivo desta dissertação, que é apresentar uma especificação de serviço, que comporá o protocolo ao se juntar com os outros elementos.

4.3.1 Modelo de Serviço

[Zimmermann 1980] apresenta treze princípios para justificar a arquitetura de sete camadas do modelo OSI:

1. Não criar muitas camadas de maneira a dificultar o projeto de sistema descrevendo e integrando essas camadas.
2. Estabelecer limites entre as camadas de maneira que as descrições de serviço possam ficar pequenas e o número de interações entre as camadas seja minimizado.
3. Criar camadas separadas para lidar com funções distintas.
4. Agrupar funções similares na mesma camada.
5. Estabelecer limites entre as camadas em pontos onde a experiência já demonstrou sucesso.
6. Criar camadas onde as funções estão bem localizadas, de maneira que toda a camada possa ser reestruturada e os protocolos trocados para aproveitar avanços arquiteturais, de hardware ou software sem interferir nas camadas adjacentes.
7. Criar um limite onde ele possa ser útil no futuro, mantendo a interface correspondente padronizada.
8. Criar uma camada onde existe a necessidade de um nível diferente de abstração na manipulação dos dados (i. e., morfologia, sintaxe, semântica).

9. Permitir alterações nas funções e protocolos em uma camada sem afetar as outras camadas.
10. Cada camada deve conter interfaces somente para as camadas adjacentes.
11. Organizar as funções distintas de uma camada em sub-camadas.
12. Criar, onde for necessário, duas ou mais sub-camadas de funcionalidade comum para permitir interface de operação com camadas adjacentes.
13. Permitir (desvio) passar por camadas ou sub-camadas sem utilizar o serviço das mesmas.

Neste trabalho usa-se os mesmos princípios para justificar a arquitetura do modelo de serviço. Primeiro, para garantir completude e corretude em relação aos aspectos filosóficos da comunicação de dados, assume-se que o modelo de serviço teria as mesmas sete camadas do modelo OSI (baseado no princípio número 5).

Visto que já existe um meio físico, um serviço de enlace (fornecidos pelo protocolo Ethernet), um serviço de rede (fornecido pelo protocolo IP), um serviço de transporte (fornecido pelo protocolo UDP) e que um serviço de apresentação é desnecessário , eliminou-se as camadas física, de rede, de transporte, e de apresentação (baseado no princípio número 1).

Restaram então as camadas de aplicação, para desempenhar funções específicas do serviço de alta disponibilidade, a camada de sessão, para desempenhar as funções de gerenciamento de grupo, e camada de enlace, para fornecer um serviço de controle de erro. Seguindo os princípios número 1 e 4 as funções da camada de aplicação foram agrupadas na camada de sessão, minimizando o serviço em duas camadas. No entanto, este serviço será prestado na arquitetura Internet, que já fornece alguns dos serviços que o modelo necessita.

A Figura 4.3 ilustra a arquitetura final do modelo de serviço de alta disponibilidade, formado por duas camadas (sessão e enlace) estrategicamente separadas pelos serviços de rede (IP) e transporte (UDP) não orientados à conexão.

Camada de sessão

O serviço de sessão foi definido para atacar o problema do acéfalo e encerra portanto, todas as questões de autenticação e criptografia do serviço de alta disponibilidade. Este serviço foi colocado exatamente onde estão outras tecnologias com funções similares na arquitetura Internet (como o SSL), baseando então no princípio número 5. Cabe ao desenvolvedor garantir que o uso deste serviço e de todos os serviços abaixo dele sejam oferecidos somente a usuários autorizados, caso o protocolo de alta disponibilidade use autenticação.

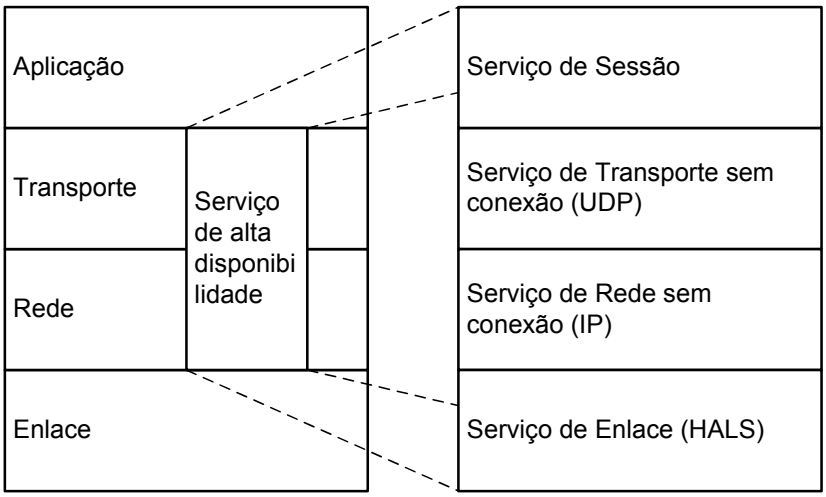


Figura 4.3: Modelo de serviço de alta disponibilidade

[Hinden 2004] remove a criptografia nessa versão do VRRP, alegando que embora capaz de inibir a condição de acéfalo, não pode impedir o cérebro partido, havendo, na visão dele, outros meios menos complexos de obter o mesmo resultado. O contra argumento aqui é, um serviço deve ser definido no nível mais abstrato possível, nada mais coerente então, que deixar a possibilidade de oferecer um serviço de criptografia. Uma vez constatada qualquer desvantagem no uso do serviço, o mesmo pode ser anulado baseado no princípio 13, ou, um novo protocolo pode ser desenvolvido para oferecer um serviço melhor, já que a interface foi bem definida.

O Gerenciamento de grupo se preocupa com a criação e remoção de grupos (por exemplo, um Roteador Virtual pode ser considerado um grupo), admissão e demissão dos elementos (Roteadores VRRP) em cada grupo. Os sub-serviços de *Connection*, *Disconnection*, e *Configuration* (são orientados à conexão visando aumentar a confiabilidade) endereçam questões de segurança e gerenciamento de grupo, e permitem incluir no protocolo de alta disponibilidade características como autenticação e criptografia.

Como mencionado na Seção 3.4, geralmente o *hardware* dos elementos de rede é especializado e otimizado, sendo o único fator do qual o rendimento do sistema depende. Logo o serviço de balanceamento torna-se interessante pois é possível atender a pequenos aumentos de demanda com custo e complexidade baixos, uma vez que é necessário apenas a adição de novos módulos de *hardware* e a configuração do protocolo.

O balanceamento de carga possibilita o aumento da expectativa de vida de equipamentos com tecnologia obsoleta, o que permite uma melhor preparação de transição para uma nova tecnologia, e permite esperar a acomodação de preço da nova tecnologia, ambas as ações resultando em economia.

Esta camada da arquitetura de serviço também tem a responsabilidade de permitir

que protocolos implementem métodos para definir os papéis (*Master* e *Slave* no caso do VRRP) de cada elemento da arquitetura, bem como suas relações. O protocolo VRRP realiza uma eleição onde apenas um nó sai como vencedor, mas se considerarmos o balanceamento de carga é possível usar outro método, uma classificação por exemplo, permitindo que todos os nós realizem o serviço, recebendo uma carga de acordo com a sua classificação.

O sub-serviço *Configuration* foi definido justamente com esse tipo de problema em mente. O mais importante a se notar é que a interface busca atingir o mais alto nível de abstração possível, ao invés de fazer restrições para uma situação específica.

Camada de enlace

O serviço de enlace foi definido para atacar o problema do cérebro partido, que ocorre devido ao *jitter* (variação do atraso) na entrega de quadros quando o protocolo Ethernet é exposto a um volume alto de tráfego. Este serviço foi colocado como usuário do serviço de enlace Ethernet, podendo desse modo oferecer um serviço de enlace com valor agregado (HALS - *High Availability Link Services*), respeitando portanto o princípio número 4. Como o protocolo Ethernet, por exemplo, tende a incorporar serviços de qualidade de serviço, segundo [Gorshe et al. 2004] e [Meddeb 2005], o princípio 13 permite aproveitar a nova tecnologia sem alterar a definição de serviço.

O sub-serviço de *Data* prevê a troca normal de mensagens (o mesmo é válido para a camada de sessão) e o sub-serviço de *Error Control* (não orientados à conexão visando aumentar a disponibilidade) permite que os protocolos de alta disponibilidade implementem métodos de controle de erro capazes de detectar e corrigir os erros que o protocolo Ethernet deixa passar, ou ainda, considerar mensagens com entrega atrasada como um erro e notificar a camada superior, descartando a mensagem quando, e se ela chegar.

4.3.2 Primitivas, Parâmetros, e Valores

Um mal entendido particular, que leva ao mau uso do conceito de serviço reside na interpretação da natureza da primitiva de serviço. Na maioria dos casos uma primitiva de serviço é interpretada como "um tipo de mensagem" que é passada através da fronteira entre serviços. Na documentação do antigo protocolo de transporte ECMA essas frases podem ser literalmente encontradas "a direção da primitiva de serviço", ou "uma primitiva de serviço é passada através do limite entre as camadas".

A interpretação acima parece inofensiva se estabelecer-se que o termo "primitiva de serviço" define a passagem de informação somente em uma direção. O serviço de sessão OSI, no entanto, já mostrou um exemplo contrário definindo a troca bi-direcional de informações em algumas de suas primitivas de serviço. A interpretação do conceito arquitetural sobre o qual o conceito de primitiva de serviço foi construído está claramente

mal entendido.

Enfatiza-se aqui que a primitiva de serviço está construída sobre o conceito arquitetural de interação, que é reforçado pelo nível de abstração requerido pelas fronteiras entre processos (ou entidades) comunicantes dentro de sistemas. Esse conceito de interação permite uma variedade muito maior de relacionamentos entre parâmetros e valores ao invés da simples passagem de valores.

Como exemplos dessa variedade menciona-se a negociação e verificação de valores. Ambos mecanismos extremamente úteis e necessários na resolução de problemas que são encontrados em serviços orientados à conexão, para o estabelecimento e distinção entre conexões. O desconhecimento dessas possibilidades leva a frases como essa nos documentos de especificação "todas as primitivas de serviço devem conter este identificador de conexão", no entanto, o identificador não é mostrado como um parâmetro das primitivas.

Mesmo se a execução de primitivas implicasse em passagem de informação somente em uma direção, a interpretação do conceito não seria tão inofensiva quanto se pode esperar. Uma interpretação pobre leva a um desenho de arquitetura pobre, por exemplo, nos primórdios do projeto 802 do IEEE existia uma situação, como ilustrado na Figura 4.4 onde o provedor de serviço aceitava uma requisição de datagrama de um usuário e então respondia com uma confirmação.

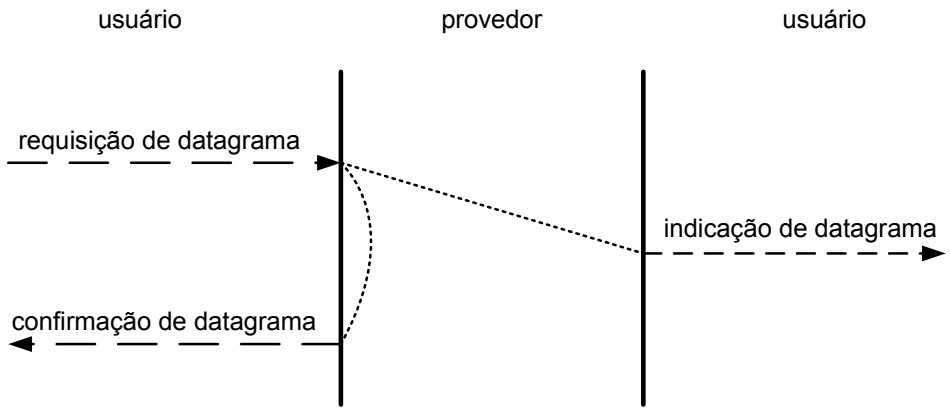


Figura 4.4: Definição de Datagrama Confirmado [Vissers e Logrippo 1985]

O significado da primitiva de confirmação não tem haver com o recebimento de uma mensagem, por exemplo, a confirmação de que a indicação de um datagrama ocorreu, mas que a parte local do provedor fez o melhor possível para entregar a requisição de datagrama. Aparentemente a introdução desta primitiva de confirmação foi inspirada na idéia de que primitivas de serviço podem ser não confiáveis: a informação pode ser passada pela fronteira entre os serviços mas o outro lado pode ignorá-la, ou ela pode nunca chegar lá.

Essa é uma má interpretação do conceito de interação: no nível de especificação não se pode aceitar a possibilidade de primitivas não confiáveis: interação requer a participação

de todas as partes envolvidas, quando uma informação passada não é aceita pelo outro lado, aparentemente um dos parceiros da interação não estava envolvido e a execução da primitiva de serviço simplesmente não aconteceu. Essa é uma questão que deve ser tratada na implementação, e não na especificação. Verificar se as primitivas de serviço são executadas de maneira confiável e tomar as devidas medidas na presença de erros é uma tarefa do implementador, que está em posição muito mais favorável para tomar esse tipo de decisão, ao contrário do especificador.

Do ponto de vista do usuário, a primitiva de confirmação não traz, claramente, nenhuma novidade. Se a confirmação é positiva, não há evidencia sobre o que fazer com ela, uma vez que é independente da indicação gerada para o outro usuário, que pode ou não ocorrer. Ainda, o que o usuário deve fazer caso a confirmação seja negativa? Na opinião dos autores essa primitiva inútil constitui mau uso do conceito de serviço, concluindo portanto, tratar-se de uma mistura de preocupações de especificação e implementação.

Primitivas da Camada de Sessão

O sub-serviço *Configuration* foi idealizado para acomodar a noção de interação (discutida na Sessão 4.3.2). Sendo sua função permitir a negociação e verificação de valores durante o estabelecimento da sessão. Como um serviço orientado à conexão ele define as primitivas *S-Configuration.request*, *S-Configuration.indication*, *S-Configuration.response* e *S-Configuration.confirmation*.

O sub-serviço de *Data* da camada de Sessão prevê a troca normal de dados entre a aplicação e a sessão. Como um serviço não orientado à conexão ele define apenas as primitivas *S-Data.request* e *S-Data.indication*. Uma primitiva de requisição de transmissão de dados do protocolo UDP é usada pelo serviço de sessão de alta disponibilidade para enviar dados ao serviço de transporte não orientado à conexão (UDP). A primitiva *S-Data.indication* é usada pelo serviço de transporte não orientado à conexão (UDP) para avisar o serviço de sessão de alta disponibilidade, que ele deve receber os dados recém chegados, ou, que houve um erro dependendo do valor status do parâmetro *sessionStatus*.

O sub-serviço *Connection* da camada de Sessão prevê o estabelecimento da conexão de sessão. Como um serviço orientado à conexão ele define as primitivas *S-Connection.request*, *S-Connection.indication*, *S-Connection.response* e *S-Connection.confirmation*.

O sub-serviço *Disconnection* da camada de Sessão prevê o encerramento da conexão de sessão. Como um serviço orientado à conexão ele define as primitivas *S-Disconnection.request*, *S-Disconnection.indication*, *S-Disconnection.response*, e *S-Disconnection.confirmation*.

Primitivas da Camada de Enlace

O sub-serviço *Data* da camada de Enlace prevê a troca normal de dados, sejam dados de estabelecimento, encerramento ou configuração de sessão, ou dados da aplicação. Como

Serviço de Sessão			
Serviço	Primitiva	Parâmetros	Valores
Connection	S-Connection.request	sessionIdentifier	identifier
		sessionParameter	value
	S-Connection.indication	sessionIdentifier	identifier
		sessionParameter	value
	S-Connection.response	sessionIdentifier	identifier
		sessionStatus	status
	S-Connection.confirmation	sessionIdentifier	identifier
		sessionStatus	status
Disconnection	S-Disconnection.request	sessionIdentifier	identifier
	S-Disconnection.indication	sessionIdentifier	identifier
	S-Disconnection.response	sessionIdentifier	identifier
		sessionStatus	status
	S-Disconnection.response	sessionIdentifier	identifier
		sessionStatus	status
Configuration	S-Configuration.request	sessionIdentifier	identifier
		sessionParameter	value
	S-Configuration.indication	sessionIdentifier	identifier
		sessionParameter	value
	S-Configuration.response	sessionIdentifier	identifier
		sessionStatus	status
	S-Configuration.confirmation	sessionIdentifier	identifier
		sessionStatus	status
Data	S-Data.request	sessionIdentifier	identifier
	S-Data.indication	sessionIdentifier	identifier
		sessionStatus	status

Tabela 4.1: Primitivas, Parâmetros, e Valores do Serviço de Sessão de Alta Disponibilidade

um serviço não orientado à conexão ele define apenas as primitivas *L-Data.request* e *L-Data.indication*. A primitiva *L-Data.request* é usada pelo serviço de rede não orientado à conexão (IP) ao enviar dados para o serviço de enlace de alta disponibilidade. A primitiva *L-Data.indication* é usada pelo serviço de enlace de alta disponibilidade para avisar o serviço de rede não orientado à conexão (IP), que ele deve receber os dados recém chegados, ou, que houve um erro dependendo do valor status do parâmetro *linkStatus*.

O sub-serviço *Error Control* tem a função de entregar o serviço de controle de erros da sub-camada *Logical Link Control* com valor agregado, em outras palavras, deve ser capaz de detectar erros não detectados pelo protocolo Ethernet e indicar como erro uma mensagem que atrasou. Não há preocupações nesse momento em determinar se a mensagem atrasou ou não, o que deseja-se aqui é deixar claro que as camadas superiores podem informar qual o tipo de erro deve ser controlado, qual tipo de mensagem deve ser monitorada, e de qual em qual intervalo, usando o parâmetro *linkParameter* da primitiva *L-ErrorControl.request*. A camada de Enlace deve indicar se o controle de erro requisitado será ou não realizado dependendo do valor do parâmetro *linkStatus* da primitiva

L-ErrorControl.indication.

Serviço de Enlace			
Serviço	Primitiva	Parâmetros	Valores
Data	L-Data.request	linkIdentifier	identifier
		linkParameter	value
	L-Data.indication	linkIdentifier	identifier
		linkStatus	status
Error Control	L-ErrorControl.request	linkIdentifier	identifier
		linkParameter	value
	L-ErrorControl.indication	linkIdentifier	identifier
		linkStatus	status

Tabela 4.2: Primitivas, Parâmetros, e Valores do Serviço de Enlace de Alta Disponibilidade

4.3.3 Pontos de Acesso

Um usuário que deseje usar o serviço de alta disponibilidade deve enviar uma primitiva ao provedor, cujo valor do parâmetro define o tipo do serviço requerido. Eventualmente, mais de uma usuário pode requerer o uso do mesmo tipo de serviço, faz-se necessário, então, o uso de um parâmetro para distinguir os serviços de mesma natureza, prestados à usuários diferentes. Por esse motivo, tanto as primitivas da camada de enlace quanto as de sessão possuem os respectivos parâmetros *linkIdentifier* e *sessionIdentifier*.

O par tipo de serviço e o parâmetro identificador constituem então, um identificador único do serviço prestado ao usuário. Esse identificador único é o ponto de acesso ao serviço, não havendo outra maneira de usar o serviço. Geralmente o usuário fica com a responsabilidade de informar ao provedor qual o tipo de serviço desejado, então o provedor devolve uma primitiva cujo valor do parâmetro identificador combinado com o tipo de serviço formam o identificador único (em alguns casos o usuário também pode passar o valor do parâmetro identificador na primitiva de requisição), cabe ainda ao usuário verificar se o valor do parâmetro de status da primitiva de indicação ou confirmação (no caso de serviços orientados à conexão) é de sucesso.

O ponto de acesso (SAP) é na verdade um objeto (um conjunto de informações) situado na fronteira entre o usuário e o provedor de serviço. Cada conexão entre usuário e provedor cria uma instância desse objeto, que permite que o mesmo serviço seja prestado a vários usuários simultaneamente e sem conflitos.

4.4 Benefícios da Especificação de Serviço

A divisão de sistemas em camadas, as interfaces de abstração, e o conceito de serviço foram inspirados, segundo [Vissers e Logrippo 1985], nos conceitos desenvolvidos em engenharia de software. A divisão em camadas é crucial para a organização de sistemas

complexos, e sem ela seria mais difícil entende-los. As interfaces e o conceito de serviço colaboram nesse processo isolando os detalhes indesejáveis. No princípio, o conceito de serviço prometeu uma revolução nos sistemas de comunicação de dados, pois vislumbrava-se que seria muito mais fácil testar e provar formalmente sua corretude.

A idéia de divisão em camadas foi largamente aceita do desenvolvimento de software à comunicação de dados, o mesmo não pode ser dito para as interfaces e o conceito de serviço. Na área de comunicação de dados, o conceito de serviço esteve quase sempre acompanhado de controvérsia. Talvez o fato dos testes e provas formais não terem se tornado uma realidade expliquem porque o conceito de serviço não foi bem aceito. No modelo de referencia OSI, que nasceu na mesma época em que as idéias da engenharia de software floresceram, é comum encontrar esse conceito. No entanto, na arquitetura Internet (de longe a mais usada atualmente) ele é raramente visto.

A ausência de padronização de conceitos nos documentos de especificação da arquitetura Internet leva a um esforço desnecessário, os benefícios da especificação de serviço tornam-se evidentes neste trabalho, uma vez que ele melhora o entendimento do problema e influi significativamente na qualidade da solução. A especificação de serviço permite ainda, acomodar futuras mudanças com uma curva de esforço muito mais suave, e aumenta a inter-operabilidade dos sistemas.

Capítulo 5

Conclusão e Trabalhos Futuros

5.1 Conclusão

Os protocolos VRRP e CARP conseguem prover alta disponibilidade muito bem, mas em ambientes críticos, onde os requisitos desses protocolos conflitam com as restrições do serviço oferecido pelo protocolo Ethernet, fica clara a possibilidade da ocorrência dos fenômenos acéfalo e cérebro partido. Esta dissertação considera estes fenômenos como sérios problemas, pois uma vez atingidos trazem graves consequências.

O fenômeno acéfalo está estritamente ligado à ausência de mecanismos de segurança, em suas considerações, [Hinden 2004] aponta a criptografia das mensagens de anúncio como medida capaz de impedir a condição de acéfalo, como observado em versões anteriores do protocolo VRRP. Ainda segundo [Hinden 2004], a criptografia é incapaz de evitar a condição de cérebro partido, sendo portanto preterida nessa versão em favor da medida de ajustar o tempo de vida das mensagens de anúncio para o valor 255, cuja verificação no recebimento garante que o pacote foi originado no mesmo segmento de rede. Esta medida, combinada com outras visando impedir que dispositivos não autorizados originem pacotes VRRP de dentro deste segmento de rede, seriam então, suficientes para resolver o problema de acéfalo.

No entanto, ([Lopes Filho 2008] cita o exemplo do *Worm*), a evolução das tecnologias e o estudo dos ambientes não raramente cria ou descobre vulnerabilidades [Nam et al. 2010]. A especificação determina no modelo o local do ponto de acesso a serviços de segurança para garantir que a condição de acéfalo não ocorrerá independente de outras medidas de segurança.

O fenômeno cérebro partido está estritamente ligado do serviço oferecido pelo protocolo Ethernet ser do tipo melhor esforço e não garantir ausência de erros e tempo real na entrega dos quadros. Neste caso a condição de cérebro partido pode durar exatamente o tempo que o protocolo Ethernet atrasar para entregar uma mensagem de anúncio (da ordem de milésimos de segundo). Ainda assim, esse curto tempo de cérebro partido pode trazer

graves consequências (retransmissão de pacotes no tráfego de dados, que pode levar a um atraso de segundos. Levar sistemas que dependem do serviço de alta disponibilidade a reiniciar, o que pode significar uma indisponibilidade de minutos).

A especificação do serviço de alta disponibilidade é um passo em direção a protocolos de alta disponibilidade mais robustos. Até então a solução para os problemas de Acéfalo e Cérebro partido eram atacados por soluções únicas, quando a especificação de serviço deixa claro que eles são totalmente distintos e devem ser combatidos em camadas totalmente diferentes na arquitetura.

Para os novos protocolos de alta disponibilidade a especificação de serviço guia a implementação, sem impor no entanto qualquer restrição. Uma vez respeitadas as interfaces fica a cargo do implementador decidir se a camada realmente oferecerá ou não determinado serviço, o que torna a especificação útil mesmo para outras arquiteturas de rede, cujos serviços ditos necessários já são oferecidos. Para os antigos protocolos de alta disponibilidade, fica uma nova visão sobre o problema, permitindo que ferramentas auxiliares combatam os pontos problemáticos na medida do possível.

5.2 Trabalhos Futuros

Vislumbra-se como trabalhos futuros o refinamento da especificação de serviço, o refinamento do autômato ilustrado na Figura 2.6, que trabalhará combinado com as primitivas da camada de enlace para fornecer um serviço de enlace com controle de erros para atrasos de quadros; uma pesquisa sobre como o protocolo Ethernet está resolvendo o problema de atrasos na entrega de quadros e quais parâmetros de QoS podem ser utilizados para fornecer um serviço de enlace melhor à camada de enlace de alta disponibilidade; e o projeto de um protocolo de sessão para trabalhar na camada de sessão de alta disponibilidade. As primitivas de cada sub-serviço definido na Seção 4.3 contribuirão na definição do formato e do vocabulário dos protocolos. Uma vez que o protocolo de alta disponibilidade esteja completo, restará ainda, conforme já foi dito por [Holzmann 1991], avaliar se ele está correto.

Referências Bibliográficas

[Armstrong 2008] Armstrong, Joseph W. (2850 Fresno Street, S. C. C. . U. H. S.-c. . P. P. C. F. C. . U. S. M. . Y. C. S. J. C. . U. (2008). **Method of solving a split-brain condition in a cluster computer system**. (EP1550036).

[Brewer 2000] Brewer, E. A. (2000). **Towards robust distributed systems**. In *PODC '00: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pp. 7+, Portland, Oregon, United States, New York, NY, USA. ACM.

[Burrows 2006] Burrows, M. (2006). **The Chubby lock service for loosely-coupled distributed systems**. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pp. 335–350, Seattle, Washington, Berkeley, CA, USA. USENIX Association.

[Caesar e Rexford 2008] Caesar, M. e Rexford, J. (2008). **Building bug-tolerant routers with virtualization**. In *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pp. 51–56, Seattle, WA, USA, New York, NY, USA. ACM.

[CARP 2004] CARP (2004). **PF: Firewall Redundancy with CARP and pfsync**. http://www.openbsd.org/faq/pf/carp.html. Comunidade OpenBSD. Relatório Técnico. Acessado em 30 de outubro de 2008.

[Cassandra 2009] Cassandra (2009). **Apache Cassandra Project**. http://cassandra.apache.org. Site do projeto. Acessado em 16 de agosto de 2010.

[Chang et al. 2006] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., e Gruber, R. E. (2006). **Bigtable: a distributed storage system for structured data**. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pp. 205–218, Seattle, Washington, Berkeley, CA, USA. USENIX Association.

[Chowdhury e Boutaba 2010] Chowdhury, N. M. K. e Boutaba, R. (2010). **A survey of network virtualization**. *Computer Networks*, 54(5):862 – 876.

[Cisco 2010] Cisco (2010). **Cisco Systems**. http://www.cisco.com. Site da empresa. Acessado em 16 de agosto de 2010.

[Deering 1989] Deering, S. (1989). **Host Extensions for IP Multicasting**. IETF, RFC 1112.

[Droms 1999] Droms, R. (1999). **Automated configuration of TCP/IP with DHCP**. *Internet Computing, IEEE*, 3(4):45 –53.

[Engler e Kaashoek 1995] Engler, D. e Kaashoek, M. (1995). **Exterminate all operating system abstractions**. pp. 78 –83.

[Fujiwara et al. 1989] Fujiwara, T., Kasami, T., e Lin, S. (1989). **Error detecting capabilities of the shortened Hamming codes adopted for error detection in IEEE Standard 802.3**. *Communications, IEEE Transactions on*, 37(9):986 –989.

[Gorshe et al. 2004] Gorshe, S., Parsons, G., Truskowski, M., e Vissers, M. (2004). **Ethernet WAN transport**. *Communications Magazine, IEEE*, 42(3):62 – 63.

[Hara 2005] Hara, Norie (Hiratsuka, J. A. K. O. J. (2005). **Cluster computing system and its failover method**. (20050005001).

[Hashimoto 2009] Hashimoto, G. T. (2009). **Uma Proposta de Extensão para um Protocolo para Arquiteturas de Alta Disponibilidade**. Dissertação de Mestrado, Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2009.

[HBase 2008] HBase (2008). **HBase: The Hadoop Database**. http://hbase.apache.org. Site do projeto. Acessado em 30 de outubro de 2008.

[Hinden 2004] Hinden, R. (2004). **Virtual Router Redundancy Protocol (VRRP)**. IETF, RFC 3768.

[Holzmann 1991] Holzmann, G. J. (1991). **Design and Validation of Computer Protocols**. Prentice Hall, 1991.

[Hypertable 2010] Hypertable (2010). **Hypertable: An Open Source, High Performance, Scalable Database**. http://www.hypertable.org. Site do projeto. Acessado em 16 de agosto de 2010.

[Juniper 2010] Juniper (2010). **Juniper Networks**. http://www.juniper.net. Site da empresa. Acessado em 16 de agosto de 2010.

[Lamport e Marzullo 1998] Lamport, L. e Marzullo, K. (1998). **The part-time parliament**. *ACM Transactions on Computer Systems*, 16:133–169.

[Li et al. 1998] Li, T., Cole, B., Morton, P., e Li, D. (1998). **Cisco Hot Standby Router Protocol (HSRP)**.

[Lopes Filho 2008] Lopes Filho, E. (2008). **Arquitetura de Alta Disponibilidade para Firewall e IPS Baseada em SCTP**. Dissertação de Mestrado, Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2008.

[Meddeb 2005] Meddeb, A. (2005). **Why ethernet WAN transport?** *Communications Magazine, IEEE*, 43(11):136 – 141.

[Nam et al. 2010] Nam, S. Y., Kim, D., e Kim, J. (2010). **Enhanced ARP: preventing ARP poisoning-based man-in-the-middle attacks**. *Communications Letters, IEEE*, 14(2):187 –189.

[Porto 2009] Porto, I. O. (2009). **Padrões e Diretrizes Arquiteturais para Escalabilidade de Sistemas**. Dissertação de Mestrado, Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2009.

[Postel 1980] Postel, J. (1980). **User Datagram Protocol - UDP**. IETF, RFC 768.

[Postel 1981] Postel, J. (1981). **Transmission Control Protocol - TCP**. IETF, RFC 793.

[Pritchett 2008] Pritchett, D. (2008). **BASE: An Acid Alternative**. *Queue*, 6(3):48–55.

[Schoenthal 2006] Schoenthal, Scott (San Ramon, C. U. R. S. H. M. V. C. U.-R. A. L. S. J. C. U. S. S. J. M. C. U. C. S. M. C. C. U. (2006). **Mirror split brain avoidance**. (7111194).

[Sen et al. 1998] Sen, S., Sekaran, M., e Hale, J. (1998). **Learning to coordinate without sharing information**. pp. 509–514.

[Steward 2007] Steward, R. (2007). **Stream Control Transmission Protocol - SCTP**. IETF, RFC 4960.

[Vissers e Logrippo 1985] Vissers, C. A. e Logrippo, L. (1985). **The importance of the service concept in the design of data communications protocols**. In *Proceedings of the IFIP WG6.1 Fifth International Conference on Protocol Specification, Testing and Verification V*, pp. 3–17, Amsterdam, The Netherlands, The Netherlands. North-Holland Publishing Co.

[Wipfel 2005] Wipfel, Robert (Sandy, U. U. G. R. K. S. L. C. U. U. (2005). **Method for detecting and resolving a partition condition in a cluster**. (6965936).

[Zimmermann 1980] Zimmermann, H. (1980). **OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection**. *Communications, IEEE Transactions on*, 28(4):425 – 432.