ROBER MAYER

MECANISMO DE PROTEÇÃO PARA SOBREVIVÊNCIA DE FLUXOS MULTICAST EM REDES DE SOBREPOSIÇÃO P2P

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

CURITIBA

2012

ROBER MAYER

MECANISMO DE PROTEÇÃO PARA SOBREVIVÊNCIA DE FLUXOS MULTICAST EM REDES DE SOBREPOSIÇÃO P2P

Dissertação apresentado ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Manoel Camillo De Oliveira Penna Neto.

CURITIBA

2012

Mayer, Rober

Mecanismo de Proteção para Sobrevivência de Fluxos Multicast em Redes de Sobreposição P2P. Curitiba, 2012. 100p.

Dissertação – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada.

1. Multicast 2. P2P 3. Proteção 4. Redes de Sobreposição 5. Sobrevivência. I. Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática Aplicada II-t

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

PUCPR

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO Nº 18/2012

Aos 11 dias do mês de Dezembro de 2012 realizou-se a sessão pública de Defesa da Dissertação **"Mecanismo de Proteção para Sobrevivência de Fluxos Multicast em Redes de Sobreposição P2P"** apresentada pelo aluno **Rober Mayer**, como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Manoel Camillo Penna de O. Neto PUCPR (Orientador)	(assinatura)	APROVADO (Aprov/Reprov.)
Profª. Drª Marcelo Pellenz PUCPR	(assinatura)	APROVADO (Aprov/Reprov.)
Prof. Dr. Edgard Jamhour PUCPR	(assinatura)	APROVADO (Aprov/Reprov.)
Prof. Dr. Carlos Marcelo Pedroso UFPR	(assinatura)	Aprov. (Aprov/Reprov.)

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado APROVADO (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof. Dr. Fabrício Enembreck
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática PPGIa PUCPR

Rua Imaculada Conceição, 1155 Prado Velho CEP 80215-901 Curitiba Paraná Brasil
Tel.: (41) 3271-1669 Fax: (41) 3271-2121 www.ppgia.pucpr.br

Dedico este trabalho aos meus pais,

Osmar Mayer e Eliane Maria Acco Mayer

pelo incentivo na continuidade de meus estudos,

e a minha namorada Carline Marquetti

pela companhia, compreensão e paciência.

Agradecimentos

Agradeço primeiramente a todos os professores do Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná, sobretudo os da linha de pesquisa de Redes de Computadores e Telecomunicações, pelo compartilhamento de seus conhecimentos. Agradeço ao professor Manoel Camillo De Oliveira Penna Neto pelas orientações prestadas a este trabalho e ao professor Gilberto Souto da UTFPR campus Pato Branco pela aula de *Branch and cut* e *branch and bound*. Um agradecimento especial para Krzysztof Walkowiak, pela gentileza em sempre responder aos meus e-mails, respondendo as meus questionamentos. Por fim, mas não menos importante, agradeço aos meus colegas de disciplinas do mestrado pelos momentos de descontração, e ao colega Marcelo Riedi, amigo nas incontáveis e cansativas viagens até Jaraguá Do Sul, nos cerca de quinhentos quilômetros percorridos nos finais de semana.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

API	*Application Programming Interface*
CDN	*Content Distribution Network*
CSL	*Cost Selection with Level Control*
DHT	*Distributed Hash Table*
DONet	*Data-driven Overlay Network*
HON-P2P	*Hybrid Overlay Network*
IP	*Internet Protocol*
IPTV	*Internet Protocol over Television*
ISP	*Internet Service Provider*
LP	*Linear Programming*
MDC	*Multiple Description Coding*
MIP	*Mixed Integer Programming*
NACK	*Negative Acknowledgements*
P2P	*Peer-to-Peer*
PRIME	*Peer-to-Peer Receiver-drIven Mesh-based Streaming*
PRM	*Probabilistic Resilient multicast*
RSL	*Random Selection with Level Control*
SHE	*Super-head-ends*
SuTeC	*Survivable Trees Constructor*
TCP	*Transmission Control Protocol*

TTL	*Time-To-Live*
VHO	*Video Hub Offices*
VoD	*Video over Demand*
VoIP	*Voice over Protocol Internet*

Resumo

Este trabalho propõe um mecanismo para sobrevivência de fluxos multicast em redes de sobreposição P2P, capaz de proteger a transmissão do fluxo contra falhas do nó de origeme dos nós intermediários da árvore multicast. Com base em um modelo de programação inteira mista, é proposto um algoritmo para construção das árvores de transmissão de fluxos multicast (árvores de multicast) e suas alternativas de reconfiguração, de modo a proteger os fluxos contra falhas de nós intermediários. Um modelo de conservação de fluxo tradicional é usado para construção de um conjunto de árvores de multicast capazes de proteger o fluxo em caso de falha do nó de origem. A proteção contra falhas de nós intermediários é alcançada através de um mecanismo de proteção que usa as árvores de reconfiguração pré-calculadas.

O mecanismo proposto é avaliado por meio de experimentos numéricos, que demonstram a eficácia do novo mecanismo de proteção, com relação aos tempos de reconfiguração das árvores em caso de falhas e a qualidade das árvores de reconfiguração geradas.

Palavras-chave: Multicasting, P2P, Proteção, Redes de Sobreposição, Sobrevivência.

Abstract

This dissertation proposes a mechanism for survival of multicast streams on P2P overlay network, capable of protecting against failures of the transmission of flows due to source nodes and intermediate nodes of the multicast tree. Based on a mixed integer programming model, an algorithm is proposed to construct trees to support the transmission of multicasting streams and their reconfiguration alternatives, in order to protect the flow against intermediate node failures. A traditional model of flow conservation is used to construct a set of multicast trees capable of protecting the flow in case of failure of the source node. The protection of intermediate nodes is achieved through a mechanism that uses pre-calculated reconfiguration trees.

The proposed mechanism is evaluated by means of numerical experiments, which demonstrate the effectiveness of the new mechanism of protection in relation to the time of reconfiguration of the trees in case of failures and quality of trees generated reconfiguration.

Keywords: Multicasting, P2P, Protection, Overlay Networks, Survival.

Capítulo 1

Introdução

A transmissão *multicast* é um mecanismo eficiente para suportar aplicações de comunicação em grupo, tais como conferência de áudio e vídeo, *games on-line* e distribuição de conteúdo, que elimina a maioria dos pacotes redundantes na rede [Birrer e Bustamante, 2005]. É particularmente útil para as aplicações de distribuição de fluxos de informação de uma origem para múltiplos destinos, por exemplo, a aplicação denominada vídeo *streaming* na qual um fluxo de vídeo é produzido em um nó de origem e distribuído para vários nós de destino. Um fluxo contínuo de informação que é transmitido de uma origem para múltiplos destinos é denominado fluxo *multicast*.

Em resposta às dificuldades de implantação da transmissão *multicast* nos protocolos da camada de rede IP (*Internet Protocol*), uma série de protocolos recentemente propostos adotam a alternativa *Peer-to-Peer* (P2P), com todas as funcionalidades da transmissão *multicast* implementadas exclusivamente nos sistemas finais (*hosts* de usuários) e não nos roteadores, e o objetivo do protocolo *multicast* é construir e manter uma cobertura eficiente para transmissão de dados [Birrer e Bustamante, 2006]. Neste mecanismo autoadaptativo, implementado na camada de aplicação, os pares participantes configuram-se em uma topologia de sobreposição (*overlay*), ou seja, uma rede lógica sobre uma rede física existente, para entrega de dados. Na topologia de sobreposição cada enlace corresponde a um caminho *unicast* entre dois sistemas finais na Internet. Várias soluções para transmissão *multicast* em redes (P2P) têm recebido um ganho de popularidade nos últimos anos, demonstrando que podem efetivamente dar suporte a aplicações de comunicação ao vivo com conteúdos variados, pela Internet.

Com o objetivo de simplificação, facilidades de transmissão *multicast* implantadas em redes de sobreposição P2P serão denominadas *multicasting* P2P.

1.1. Necessidade da Sobrevivência em P2P *Multicast*

Os pares em um sistema P2P são mais vulneráveis a falhas por serem compostos por *hosts* finais, na maioria das vezes *desktops* ou *notebooks* de uso comum. Assim, a interrupção de um único *host* pode interferir em diversos fluxos *multicast*.

Neste contexto, a sobrevivência a falhas em redes P2P é quesito importante no projeto e operação destas redes. A sobrevivência a falhas é a capacidade de uma rede não prejudicar, ou não interromper, as conexões de seus usuários quando ocorrer a falha de algum recurso da rede. Nas redes convencionais em malha, os mecanismos que oferecem sobrevivência a falhas, também denominados mecanismos de sobrevivência, são classificados, basicamente, em dois tipos [Banerjee et al., 2006], [Fei e Yang, 2007], [Walkowiak, 2009] e [Walkowiak e Przewozniczek, 2011]: os mecanismos de proteção, que pré-computam e pré-alocam os recursos de recuperação; e os mecanismos de restauração, que tratam dos recursos de recuperação de maneira reativa apenas quando ocorre a falha. Aplicar o conceito de sobrevivência a falhas a uma rede de sobreposição P2P, significa selecionar enlaces e nós alternativos que possibilitam que a rede continue a oferecer os serviços a que foi destinada, mesmo na presença de falhas.

Neste trabalho, considera-se que fluxos *multicast* serão distribuídos aos usuários finais através de redes de sobreposição P2P. Assume-se que o conteúdo dos fluxos é tal que necessita de garantias de entrega, isto é, que mecanismos de sobrevivência são necessários para suportar a entrega confiável dos fluxos *multicast*. Assume-se também que os nós que compõem a rede de sobreposição não deixam a sessão de transmissão *multicast* de forma voluntária e independente, ou seja, assumindo-se a hipótese de rede de sobreposição estável (ver seção 2.1).

1.2. Objetivo Geral

O objetivo desse trabalho é propor um mecanismo para sobrevivência de fluxos *multicast* em redes de sobreposição P2P, capaz de proteger a transmissão do fluxo contra falhas do nó origem e dos nós intermediários da árvore *multicast*. Com base em um modelo de programação inteira mista, é proposto um algoritmo para construção das árvores de transmissão de fluxos *multicast* (árvores de *multicast*) e suas alternativas de reconfiguração, de modo a proteger os fluxos contra falhas de nós intermediários. Um modelo de conservação de fluxo tradicional é usado para construção de um conjunto de árvores de *multicast* capazes

de proteger o fluxo em caso de falha do nó de origem. A proteção contra falhas de nós intermediários é alcançada através de um mecanismo de proteção que usa as árvores de reconfiguração pré-calculadas.

1.3. Objetivos Específicos

Além do objetivo geral já enunciado, esse trabalho também proporcionou as seguintes contribuições:

- Descrição e classificação de métodos de proteção *multicasting* P2P;
- Formulação através de programação inteira mista de um mecanismo de proteção para sobrevivência de sistemas *multicasting* P2P;
- Especificação dos esquemas de uso e distribuição de informação de proteção para implantação dos modelos nos nós das redes P2P, e
- Avaliação do mecanismo proposto, por meio de experimentos numéricos, para demonstrar a eficácia do novo mecanismo de proteção, com relação aos tempos de reconfiguração das árvores em caso de falhas e a qualidade das árvores de reconfiguração geradas.

1.4. Hipóteses e Restrições

Neste trabalho considera-se que a transmissão de fluxos *multicast* em redes de sobreposição P2P será usada para distribuir conteúdo com requisitos de grande importância, como por exemplo, sistemas de previsão de tempestades, de dados financeiros, de alertas de segurança, IPTV, o que determina a necessidade de características de sobrevivência.

A inspiração para o trabalho são os métodos de sobrevivência desenvolvidos para redes com pares participantes estáveis. Essa visão diferenciada de uma rede P2P, permite que os trabalhos avancem no desenvolvimento de novos métodos de proteção para falhas na rede, não se preocupando com as constantes entradas e saídas no sistema dos pares, amplamente estudadas em uma rede P2P dinâmica.

O mecanismo desenvolvido atua pré-calculando árvores para o caso de falhas na transmissão de nós intermediários das redes *multicasting* P2P, limitando a proteção dentro da própria árvore. A rede P2P considerada é uma rede com organização estruturada, ou seja, são caracterizadas por um modelo de ligações que segue uma determinada hierarquia.

As simulações para testar o mecanismo proposto por meio deste trabalho estão

restritas a um número máximo de 18 nós. Essa limitação do tamanho das redes, se deve pelo problema ser do tipo NP-completo, para o qual a ferramenta usada para realizar a otimização (CPLEX) não executaria com um tempo de resposta adequado. Deixa-se para os trabalhos futuros, a implementação de uma heurística para aplicar o mecanismo de proteção proposto para redes com maior número de nós.

1.5. Declaração do Problema

Este trabalho define um novo mecanismo de proteção de fluxos *multicast* em redes P2P estáticas, que possibilita a proteção dos nós intermediários, sem a necessidade de envio simultâneo de dados através de árvores de proteção, como descrito em trabalhos anteriores. Para o caso de falha do nó fonte (raiz), mantém-se o método tradicional de proteção por múltiplas árvores. O desenvolvimento do trabalho foi conduzido utilizando-se um algoritmo *off-line*, que pré-calcula árvores de *multicast* segundo os critérios de otimização definidos por meio da programação inteira mista, implementados no ambiente de otimização CPLEX. A partir de uma árvore inicial com custo mínimo, o modelo proposto calcula novas árvores de proteção (reconfiguração) considerando as falhas de nós intermediários da árvore de *multicast*. Por fim, a partir das árvores de proteção pré-calculadas, são construídas as tabelas de reconfiguração de cada nó, uma para cada falha de nó intermediário na árvore inicial, que são ativadas pelos mecanismos propostos, conforme as falhas venham a ocorrer.

1.6. Estrutura do Trabalho

O capítulo 2 contém o levantamento bibliográfico que faz referencia aos principais conceitos que se fizeram necessários para o estudo e desenvolvimentos dos trabalhos e os trabalhos relacionados com este estudo, apresentando as principais propostas para redes P2P *multicasting*, e descrevendo com detalhes os trabalhos que inspiraram a proposta.

O capítulo 3 apresenta o mecanismo de proteção proposto, demonstrando o método de proteção (modelo de programação inteira mista), bem como as tabelas de reconfiguração construídas a partir das árvores de proteção e as trocas de mensagens entre os nós da rede.

No capítulo 4 é feita a avaliação do mecanismo proposto, sendo apresentada a comparação dos resultados obtidos com outros trabalhos publicados na área.

Por fim, ao final do documento estão as conclusões e discussão sobre as limitações desse trabalho e sugestão de possíveis trabalhos futuros, bem como as referências

bibliográficas utilizados no trabalho e os anexos que comprovam os ensaios realizados.

Capítulo 2

Levantamento Bibliográfico

Este capítulo inicia apresentando o conceito de redes P2P, uma tecnologia que permite a utilização de recursos dos próprios usuários do sistema para efetuar a distribuição dos dados de maneira colaborativa.

A segunda parte discute a necessidade de proteção em redes P2P, principalmente com relação a transmissões sensíveis à perda e ao atraso, como Vídeo sob Demanda (VoD) ou a IPTV (*Internet Protocol over Television*).

A terceira parte discorre a respeito da sobrevivência em redes *multicasting* P2P e a quarta parte sobre os trabalhos relacionados, e que serviram de base para o desenvolvimento deste trabalho.

2.1. Redes P2P

2.1.1. Conceitos e Arquitetura

Em uma rede P2P, ilustrada na figura 1, as aplicações utilizam a comunicação direta entre pares de *hosts* conectados, denominados pares. Os pares não são de propriedade dos provedores de serviços (*Internet Service Provider* - ISPs), mas sim, são controlados por usuários de *desktops* e *laptops*, sendo que a maioria dos pares se aloja em residências, universidades e escritórios. Não há nenhuma exigência de que os pares sempre estejam em funcionamento ou conectados à rede. Como os pares se comunicam sem passar por nenhum servidor dedicado, a arquitetura é denominada par-a-par (*peer-to-peer* – P2P). Isso quer dizer que os computadores da rede estão todos interligados em uma cadeia descentralizada, onde cada um possui funções equivalentes não havendo uma hierarquia entre eles. Todos os usuários são clientes e servidores, funcionando, assim, de forma totalmente independente e

livre da existência de um servidor central [Kurose e Ross, 2010].

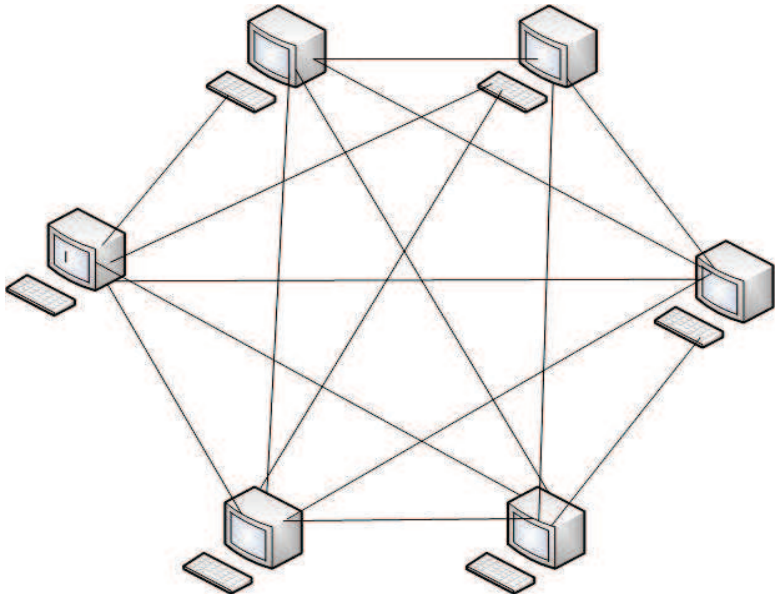

Figura 1: Uma Rede P2P

Uma das principais vantagens das redes P2P é sua escalabilidade. Cada par possui uma conexão individual com a Internet, a qual contribui com sua largura de banda nos dois sentidos da comunicação (*download* e *upload*). Desse modo, cada par que adere a uma rede P2P incrementa a sua capacidade total de comunicação.

Um bom exemplo para ilustrar a escalabilidade das redes P2P é a aplicação de compartilhamento de arquivos. Nesse caso, embora cada par gere uma carga de trabalho solicitando arquivos, cada par também acrescenta capacidade de serviço ao sistema distribuindo arquivos a outros pares. Nesse caso, um par da rede pode ser um nó interessado em receber arquivos de outros pares, como um nó folha em uma árvore, mas pode também ter a função de distribuir arquivos para outros pares interessados, funcionando como um nó intermediário de outra árvore. O compartilhamento de arquivos em sistemas P2P é mais eficiente que em sistemas cliente-servidor convencionais, principalmente para a distribuição de arquivos grandes, como conteúdos de vídeo de alta qualidade [Shyu e Lee, 2009]. Na distribuição de um grande arquivo a partir de um único servidor a um grande número de hospedeiros, na arquitetura cliente-servidor, o servidor deve enviar uma cópia do arquivo para cada um dos clientes, sobrecarregando o servidor e consumindo uma grande quantidade de banda. Na distribuição de arquivos em arquiteturas P2P, cada par pode redistribuir qualquer

parte do arquivo que recebeu para outros pares, auxiliando, assim, o nó fonte no processo de distribuição. Atualmente, o protocolo de distribuição de arquivos P2P mais utilizado é o Bit Torrent [Kurose e Ross, 2010]. Outro exemplo de aplicação que funciona de modo eficiente em redes P2P é a distribuição de fluxos de vídeo. O problema é semelhante ao de distribuição de arquivos quando a exibição não ocorre em tempo real, mas também pode aparecer na distribuição dos fluxos de vídeo para pessoas que assistem ou compartilham o mesmo vídeo [Magharei e Rejaie, 2007].

As redes P2P também possuem uma boa relação custo-benefício, pois normalmente não requerem uma infraestrutura de servidores significativa [Kurose e Ross, 2010]. Outra vantagem é que cada par atua como um servidor, portanto o desligamento ou mau funcionamento de algum par atuando como servidor não interrompe todos os serviços da rede.

As redes P2P representam uma forma de construir sistemas e aplicativos distribuídos, onde os dados e recursos computacionais são derivados da colaboração de muitos pontos na Internet de maneira uniforme. Os pontos que compõe uma rede P2P podem atuar tanto como servidores quanto como clientes sem uma coordenação centralizada [Lua et al., 2005].

De acordo com [Kurose et al., 2010] os sistemas P2P podem servir para a distribuição de arquivos, para construção de um banco de dados distribuído também conhecido como *Distributed Hash Table* (DHT) e telefonia pela Internet (VoIP).

Algumas características P2P podem ser entendidas como desvantagens. A redundância é uma delas, pois devido ao tamanho das redes, é improvável que um *host* obtenha o mesmo resultado ao efetuar duas buscas na rede. Uma segunda desvantagem é a perda de conteúdo, pois um par possui a facilidade de entrar e deixar a rede, assim um conteúdo compartilhado pode deixar de existir na rede com certa facilidade. Por fim, o desempenho também pode ser uma desvantagem, já que pode ser reduzido bastante caso uma rede preparada para trabalhar com um grande número de pares conectados, venha a possuir uma baixa participação de pares, pois as buscas e recursos serão ilimitados [Kurose e Ross, 2010].

Um ponto crítico da operação das redes P2P, que influencia o seu desempenho, seus protocolos e a arquitetura das aplicações é a entrada e saída de forma voluntária e independente dos seus participantes (*churn*). Em redes P2P, o *churn* faz com que os caminhos tenham que ser reconstruídos com frequência.

Segundo [Kurose e Ross, 2010] os provedores de serviço estão mais interessados em utilizar as redes P2P para suas aplicações para reduzir custos, e definem três desafios que

devem ser trabalhados em relação a futuras aplicações P2P. O primeiro desafio advém de os serviços residenciais providos pela maioria dos ISPs terem sido projetados para uso da largura de banda assimétrica, ou seja, para muito mais capacidade de entrada (*download*) do que de saída (*upload*). Mas, as aplicações em redes P2P requerem, em geral, tanto tráfego de entrada quanto tráfego de saída, gerando sobrecarga significativa no enlace de *upload*. O segundo desafio é a segurança, porque as aplicações P2P são altamente perigosas devido a sua natureza altamente distribuída e exposta. O terceiro desafio é o incentivo, onde o sucesso das futuras aplicações P2P depende de usuários dispostos a participar oferecendo largura de banda, armazenando arquivos e recursos da computação às aplicações [Kurose e Ross, 2010].

2.1.2. Características das Redes P2P

Segundo [Ranjan et al., 2006] as redes P2P apresentam características relacionadas à organização da rede, organização dos dados e a forma de roteamento.

Como os nós que compõe uma rede P2P podem ser organizados logicamente, os mesmo podem utilizar duas categorias básicas [Milojicic et al., 2002]: estruturadas e não estruturadas.

Um sistema não estruturado é descrito por um modelo de ligações aleatórias que são baseadas na popularidade das informações de cada nó [Ranjan et al., 2006]. Neste modelo não existe uma preocupação em criar e manter uma organização lógica [Milojicic et al., 2002]. São exemplos de sistemas não estruturados o Napster, Gnutella e Kazaa.

Já os sistemas estruturados são caracterizados por um modelo de ligações que segue uma determinada hierarquia, como a apresentada pelo DHT [Ranjan et al., 2006]. Além de seguir uma hierarquia, os sistemas estruturados se preocupam em manter uma organização lógica denominada *overlay* (rede de sobreposição) [Jin et al., 2008]. Essa organização é mantida para que a busca por um recurso possua o menor número de passos dentro da rede de sobreposição. São exemplos de sistemas estruturados o CAN, Pastry, Chord e Tapestry.

A organização dos dados segue o modelo da organização da rede [Ganesan et al., 2004]. Se a rede for não estruturada, significa que os nós não farão parte de uma topologia específica e nem de um domínio específico uma vez que os dados são espalhados pela rede de forma aleatória.

No modelo estruturado, os dados são organizados na rede seguindo uma topologia específica que depende da rede de sobreposição. Essa característica é utilizada para limitar a

complexidade da busca, o balanceamento de carga e a limitação na sobrecarga do gerenciamento da localidade dos dados [Ganesan et al., 2004].

Os dados podem ser as informações referentes aos nós, como capacidade de processamento e armazenamento, ou arquivos. Esses dados formam a *hash* que é utilizada para localizar outros nós dentro da rede (Bienkowski et al., 2005).

A consulta e o roteamento são relacionados à forma como a rede é estruturada. Normalmente as redes não estruturadas utilizam técnicas como busca em largura e profundidade ao procurar por um determinado recurso [Androutsellis-Theotokis e Spinellis, 2004].

Já as redes que utilizam sistemas estruturados possuem um sistema de busca que utiliza a hierarquia da rede [Ganesan et al., 2004]. Esse método proporciona o controle de carga no roteamento, uma vez que cada nó recebe aproximadamente o mesmo número de consultas e mantém um número limitado de informações sobre os nós da rede. As informações que um nó mantém se referem apenas aos nós mais próximos a ele [Ganesan et al., 2004].

2.1.3. Redes P2P de Sobreposição

Sistemas P2P normalmente implementam redes de sobreposição, que são criadas na camada de aplicação, sobre as redes físicas. Essas redes são usadas para descobrir e identificar os pares tornando a rede P2P independente da topologia da rede física. O conteúdo é compartilhado diretamente sobre a rede IP. Redes de sobreposição consistem de todos os pares participantes como nós da rede. Existem ligações diretas entre todos os nós que conhecem a localização um do outro na rede [Kurose e Ross, 2010]. Portanto, redes de sobreposição são redes lógicas construídas sobre uma rede física já existente.

As redes de sobreposição são constituídas de enlaces lógicos criados entre os nós participantes da rede. Assim, como pode ser visto na figura 2, existe a ideia de que a rede de sobreposição forma uma camada acima da rede física. Outro detalhe que precisa ser notado é que um enlace lógico da rede de sobreposição não precisa corresponder a um enlace físico da rede física [Coutinho, 2006].

Outro exemplo de redes de sobreposição é a própria Internet, que no início era uma rede de dados sobreposta ao sistema telefônico público e ainda hoje conta com muitas conexões desse tipo.

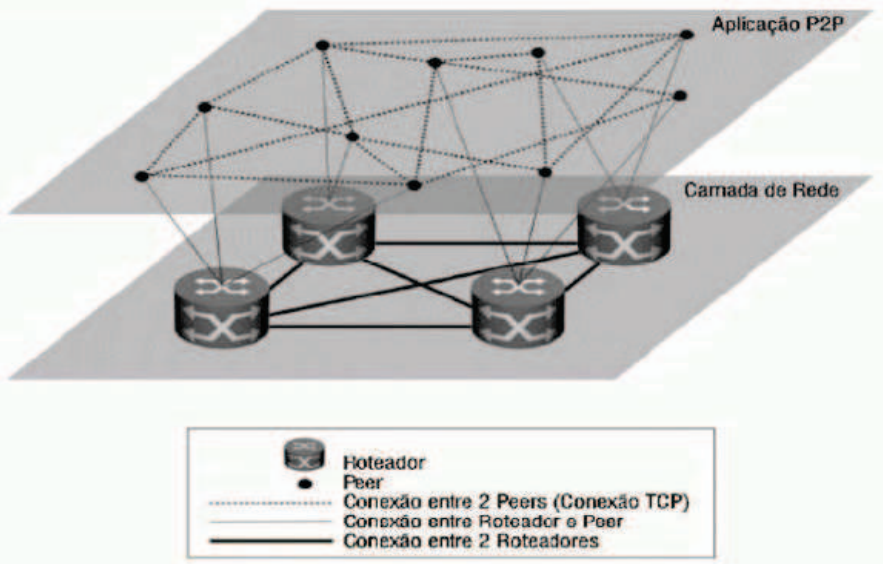

Figura 2: Rede de sobreposição para P2P [Coutinho, 2006]

2.1.4. *Multicast* em redes P2P

A transmissão *multicast* em redes de sobreposição P2P, denominada *multicasting* P2P, usa uma árvore de distribuição *multicast* construída entre os pares na rede de sobreposição. Em contraste com a transmissão *multicast* em redes IP, os nós intermediários da transmissão *multicast* não são roteadores, e sim *hosts* finais, que são denominados nós de *upload*. O *multicasting* P2P pode ser usado para distribuir uma ampla gama de dados, incluindo conteúdo elástico (por exemplo, arquivos de dados) e o fluxo de conteúdos com requisitos específicos de taxa de bits (por exemplo, fluxos de mídia) [Walkowiak e Przewozniczek, 2011].

Na maioria dos protocolos para transmissão *multicast* em redes de sobreposição, os pares se organizam automaticamente em duas topologias: uma rede de sobreposição para controle das tarefas relacionadas aos membros do grupo, e uma árvore de distribuição para encaminhamento dos dados. De acordo com [Birrer e Bustamante, 2006] com base na sequência adotada na construção destas topologias e nas suas propostas de estruturação, quatro tipos de arquiteturas são identificados: *tree-first, mesh-first, DHT-first e implicit*. Nas redes *tree-first*, os pares constroem uma árvore de distribuição de dados selecionando seus pais entre pares conhecidos. A adição dos enlaces é feita depois para definir a topologia de controle. As redes *mesh-first* constroem *spannings trees* de caminho mais curto enraizadas em qualquer um dos pares, sobre o grafo mais densamente conectado. Na arquitetura *DHT-first*,

os pares se organizam dentro de uma estrutura geométrica bem definida sobre a qual uma topologia de distribuição de dados é construída. Por fim, na proposta *implicit*, os pares criam apenas uma topologia de controle, enquanto que a árvore de distribuição de dados é implicitamente definida por regras de encaminhamento de pacotes baseadas na árvore de controle inicial.

2.2. Classificação das Propostas de *Multicasting* P2P

As propostas de *multicasting* P2P podem ser organizadas dentro de três classes, de acordo com a construção da rede de sobreposição: baseadas em árvore (*tree-based*), baseadas em redes em malha (*mesh-based*) e dirigidas por dados (*data-driven*) [Hongyun et al., 2008].

Devido as características deste trabalho estarem voltadas para um problema de árvore, descreve-se a seguir as propostas de multicasting P2P baseadas em árvore.

2.2.1. Propostas de *Multicasting* P2P Baseadas em Árvore

Os protocolos apresentados a seguir organizam os pares dentro de árvores lógicas simples ou múltiplas sobre as quais os dados *multicast* são distribuídos. As árvores são muito dependentes da confiabilidade de nós não folhas cujas falhas podem resultar em partições temporárias da árvore, podendo causar perda de pacotes de dados. Recentemente, para reduzir o impacto da dinâmica de entrada e saída dos nós no desempenho dos protocolos *multicast* das redes de sobreposição, diversas técnicas têm sido propostas na tentativa de melhorar a tolerância a falhas explorando a ideia de diversidade de caminhos e diminuindo a dependência em relação aos nós. As diversas propostas na classe com base em árvores, qui apresentadas, podem ser classificadas, de acordo com o tipo de redundância, em: redundância por arestas cruzadas (*cross-edge*), redundância interna da árvore de *multicast* (*In-tree*) e redundância por múltiplas árvores de *multicast* (*multiple tree*) [Hongyun et al., 2008].

a) Redundância por arestas cruzadas

O PRM (*Probabilistic Resilient Multicast*), definido por [Banerjee, 2006] foi o primeiro esquema proposto para *multicast* com tolerância a falhas. O PRM adota um componente proativo para encaminhamento aleatório no qual cada nó da rede de sobreposição, com uma pequena probabilidade, escolhe um número constante de outros nós da rede de sobreposição e encaminha transmissões adicionais ao longo da árvore de distribuição e sobre algumas arestas cruzadas. O componente randomizado acrescenta pouca

sobrecarga e pode garantir elevados índices de entrega de dados mesmo com alta quantidade de falhas de nós da rede de sobreposição. O PRM possui um segundo componente, este reativo, chamado *trigerred* NACKs (*negative acknowledgments*) para controlar a perda de dados devido a erros de enlace e congestionamento da rede, que foi originalmente implementado usando NICE [Banerjee et al., 2002], um protocolo de aplicação, que tem como ideia principal a redução de complexidade e o ganho de desempenho utilizando uma estrutura hierárquica para sua rede, organizada em níveis.

b) Redundância interna da árvore de *multicast*

Nesta categoria os receptores de uma árvore são organizados dentro de uma hierarquia de *clusters* onde cada nível tem definido um líder e um líder associado. O Zigzag é um protocolo *multicasting* P2P hierárquico para redes de sobreposição que se encaixa nessa descrição. Nele, a árvore é construída com os pares organizados em uma hierarquia multicamada recursiva de *clusters*. Com um líder e um líder associado em cada nível, o Zigzag separa o controle da manutenção da árvore e da distribuição de dados, melhorando a tolerância a falhas. O líder é o responsável por monitorar os membros do *cluster* enquanto que o líder associado é o responsável por transmitir o conteúdo para os membros do *cluster*, que por sua vez trocam mensagens de controle para entrar e manter suas posições na árvore. A falha do líder não afeta a continuidade do serviço de outros membros, já que ele não é o responsável pela transmissão de dados. Quando um líder associado deixa a rede devido a uma falha ou por decisão individual, o líder define um novo líder associado. Devido ao uso de dois pares para compartilhar as responsabilidades na presença de falhas, a recuperação de falhas pode ser feita localmente no *cluster*, com impacto, no máximo em um número constante de receptores existentes e principalmente sem sobrecarregar o nó fonte com a manutenção da árvore e a distribuição de dados na árvore [Tran et al., 2004].

A proposta de [Birrer e Bustamante, 2005] é um protocolo *multicasting* P2P, para redes de grande escala, heterogêneas em ambientes altamente dinâmicos. Essa proposta define o protocolo Nemo, para *multicasting* P2P, organizando os pares dentro de uma hierarquia lógica baseada na proximidade da rede, sobre a qual as árvores de distribuição de dados estão implicitamente definidas por um conjunto de regras de encaminhamento.

O Nemo atinge a tolerância a eventos inesperados através da inserção de co-líderes, líderes alternativos que compartilham a carga de encaminhamento com os líderes dos *clusters*, de maneira semelhante ao descrito no protocolo ZigZag. Quando um líder de um *cluster*

recebe uma transmissão de um membro qualquer de seu *cluster*, ele deve encaminhar esta transmissão para os demais membros de seu *cluster* e para o líder do próximo *cluster*, com a ajuda do co-líder de seu *cluster* alternando o receptor das mensagens entre eles. Como os co-líderes compartilham a carga das mensagens de encaminhamento com os líderes, eles também ajudam a reduzir a demanda de largura de banda dos líderes de *clusters*, melhorando a escalabilidade do sistema como um todo. O Nemo utiliza número de sequências e NACKs, para detectar perda de pacotes, através de um algoritmo heurístico para realizar a manutenção da rede de sobreposição. O Nemo ainda reduz o custo da manutenção da rede de sobreposição através da adoção de uma aproximação probabilística, onde as operações são executadas com alguma prioridade, ou de forma alternativa, adiadas para o próximo intervalo.

c) Redundância por múltiplas árvores de *multicast*

Para resolver o problema de gargalos no encaminhamento de dados em árvores isoladas, várias propostas surgiram com o objetivo de usar múltiplas árvores disjuntas sobre as quais os fluxos de dados são disseminados. Encaminhando diferentes fluxos sobre cada árvore e tornando cada par um nó interno em pelo menos uma árvore, a redundância é atingida através da distribuição da carga de dados de forma mais igualitária entre os pares participantes.

Seguindo essa ideia, [Castro et al., 2003] apresentaram o Splitstream, que introduz a difusão de conteúdo através de múltiplas árvores. A implementação do Splitestream é baseada no Scribe, um sistema de comunicação de grupo construído sobre Pastry que por sua vez é uma rede de sobreposição P2P estruturada. O Splitstream consegue atingir esse objetivo dividindo o conteúdo original em vários fluxos, e encaminhando cada fluxo usando uma árvore separada. Os pares juntam-se a tantas árvores quantos forem os fluxos que eles desejam receber e especificam um limite máximo de fluxos que eles estão dispostos a encaminhar. O desafio é construir uma floresta de árvores *multicast* em que um nó interno em uma árvore é um nó folha em todas as demais árvores e que a restrições de largura de banda especificada pelos nós sejam satisfeitas, ou seja, respeitando a capacidade de banda de cada par. Isso garante que a carga de encaminhamento possa ser espalhada através de todos os pares participantes. Dividindo o fluxo através de múltiplas árvores também se melhora a tolerância a falhas para falha de nós. O Splitestream oferece robustez para falhas dos nós e para as partidas voluntárias dos nós, explorando a diversidade de caminhos, garantindo que a maioria dos nós sejam nós internos em apenas uma árvore. Além disso, a falha de um simples

nó causa a perda temporária de, no máximo, um fluxo.

O CoopNet, proposto por [Padmanabhan et al., 2003], utiliza múltiplas árvores de distribuição e segue uma abordagem que mescla aspectos da arquitetura cliente-servidor com aspectos da arquitetura P2P. Assim como na arquitetura cliente-servidor, há um servidor central que provê todas as mídias (pré-gravadas ou ao vivo), além de servir como "porta de entrada" para as novas estações que queiram se conectar à rede. Por outro lado, assim como nas redes P2P, há conexões diretamente entre as estações que, aproveitando os recursos ociosos que possuem, auxiliam o servidor na distribuição do conteúdo, provendo uma grande escalabilidade ao sistema. Trata-se de um sistema P2P de fluxo de mídia ao vivo que complementa, ao invés de substituir, a arquitetura cliente-servidor.

No CoopNet uma distribuição sincronizada de conteúdo aos clientes é realizada. Consequentemente, árvores de distribuição são formadas através de conexões *unicast*, estando o servidor em suas raízes. Cada cliente é incluído como nó nessas árvores, podendo ser admitido diretamente pelo servidor, caso este tenha recursos para isso, ou por outros nós que já estejam conectados no sistema, escolhidos aqueles que ainda tenham recursos excedentes. Cada cliente, então, repassa o fluxo recebido em uma determinada árvore de distribuição a todos os seus descendentes, até que o fluxo atinja os nós folhas daquela árvore. O grau de saída de cada nó do sistema é limitado pela banda de saída da respectiva entidade (servidor ou cliente). Em geral, o servidor apresenta uma banda de saída bem superior às disponibilizadas pelos clientes.

Para minimizar os problemas de rupturas em árvores decorrentes da constante entrada e saída de clientes, o sistema CoopNet utiliza, além de múltiplas árvores de distribuição, a codificação em múltiplos descritores (*Multiple Description Coding* ou MDC). A MDC é um método de codificação de áudio e/ou vídeo no qual um sinal é codificado em certo número de fluxos distintos, sendo cada fluxo chamado de descritor. Dessa forma, utilizando uma árvore de distribuição distinta para o transporte de cada descritor, o fluxo não é totalmente interrompido quando uma desconexão ocorre. Ao contrário, apenas um subconjunto dos descritores deixa de ser entregue temporariamente, ocasionando uma queda momentânea na qualidade de fluxo recebida.

Ainda, o CoopNet emprega a sugestão do Splitstream, descrita anteriormente, que indica que cada cliente deve ser admitido como nó interior em uma única árvore, sendo nó folha nas demais. Com isso, consegue-se construir árvores o mais disjuntas possível,

diversificando os caminhos de rede para aumentar a tolerância a falhas, bem como para melhor aproveitar as bandas de saída dos clientes (se um nó fosse folha em todas as n árvores, a sua banda de saída não seria aproveitada).

[Walkowiak, 2009] aborda o problema para transmissão *multicast* em redes P2P estáticas, formulando um problema de otimização para criação de árvores disjuntas em redes de sobreposição para proteger o sistema contra os seguintes tipos de falhas: falhas dos nós raízes, falhas de enlaces de sobreposição, falhas de nós de *upload* entre as árvores e a falhas de enlaces de ISP. Os objetivos são dois: minimizar o custo de transmissão e maximizar o *troughput* de todos os pares receptores da rede. Para resolver o problema, são usados um modelo de programação inteira mista (MIP) e simulações. No primeiro caso, os melhores resultados gerados pela ferramenta de otimização CPLEX são apresentados e discutidos. O último caso representa um ambiente distribuído e um simulador de rede simples é aplicado para fazer os experimentos. Os resultados obtidos em ambos os métodos da pesquisa indicam que os requisitos adicionais necessários para a proteção não tem uma influência significativa sobre o desempenho do sistema *multicasting* P2P.

[Walkowiak e Przewozniczek, 2011] apresentam uma extensão do trabalho anterior, considerando, também, a métrica de atraso. O problema foi resolvido por modelos de programação inteira mista e por um método heurístico. No primeiro caso foram utilizadas duas formulações, a mesma formulação por fluxos anteriormente estudada e uma formulação por níveis que permite que restrições referentes ao número de níveis das árvores sejam implementados, reduzindo o tempo de execução do sistema. Novamente os testes foram executados sobre o CPLEX concluindo-se que os sistemas P2P *multicasting* podem ser melhorados com métodos de proteção adicionais, sem redução significativa do desempenho do sistema. Também é apresentando um algoritmo heurístico, chamado SuTeC (*Survivable Trees Constructor*) para resolver o problema de sobrevivência em P2P *multicasting*, com o objetivo de minimizar o custo dos fluxos, em redes de grande porte, para os quais os modelos de programação inteira mista não conseguem encontrar solução em tempo adequado. O SuTeC é um método heurístico estocástico que utiliza um algoritmo de propósito geral e que não utiliza mecanismo de restrições.

2.3. Sobrevivência em Redes *Multicasting* P2P

A tolerância a falhas ou tolerância a eventos inesperados (*resilience*) em uma rede de

multicasting P2P deve ser tratada em dois casos, quando um nó falhar ou quando ele deixar a sessão *multicast* voluntariamente. A questão chave é como reconstruir a árvore de sobreposição dentro de um tempo adequado após a ocorrência desses eventos. Na transmissão *multicast* em redes IP, os nós intermediários na árvore de distribuição são roteadores, que são relativamente estáveis e não deixam a árvore *multicast*. Já em uma rede *multicasting* P2P, um nó corresponde a um processo executando em um *host* final, que é potencialmente mais susceptível a falhas do que um roteador, principalmente devido à possibilidade de o *host* final poder deixar a sessão *multicast* unilateralmente. Esses eventos causam a desconexão de nós que dependem do nó que falhou (ou que deixou espontaneamente a rede) até a reconstrução da árvore de *multicast*. Perdas devido a falhas nos nós da rede de sobreposição são mais significativos que perda de pacotes regulares e podem causar interrupção dos fluxos de dados por períodos na ordem de dezenas de segundos [Banerjee et al., 2006].

A recuperação da árvore de *multicast* na rede de sobreposição pode ser realizada de duas maneiras: reativa e pró-ativa. No primeiro caso a restauração da árvore, que acontece somente após a ocorrência do evento, costuma levar um tempo maior para reparar a árvore de *multicast*. A dificuldade é encontrar um novo pai para cada nó desconectado, devendo contatar vários nós na árvore antes de encontrar uma locação apropriada [Fei e Yang, 2007]. No segundo caso, a ideia básica para a recuperação é que cada nó não folha na árvore pré-calcule um plano de resgate antes das falhas ou partidas dos nós [Fei e Yang, 2007].

Ainda no caso de recuperação de forma ativa, é possível trabalhar com um método *off-line*, que executa um algoritmo para pré-calcular (antes da operação) um plano de restauração. Assim, ao se iniciar o sistema, todos os pares de uma rede *multicasting* P2P, já conhecem todas as possíveis falhas e seus respectivos planos de restauração. Isso é uma vantagem, pois libera os pares de realizarem trabalhos extra durante uma falha, ou durante o tempo em que estão transmitindo dados, ganhando em desempenho e até exigindo menor uso de recursos computacionais.

Desconsiderando a dinamicidade nas redes P2P, [Walkowiak, 2009] e [Walkowiak e Przewozniczek, 2011] assumem em seus trabalhos que o cenário de *multicasting* P2P é relativamente estático. Isto é, que os nós participantes têm interesse no sucesso da comunicação, e que não deixam a rede voluntariamente. Consequentemente, seus trabalhos focam nas falhas de rede não considerando questões relacionadas ao dinamismo de sistemas P2P. Com base nessas premissas os autores argumentam que mecanismos de sobrevivência

semelhantes aos estabelecidos na transmissão *unicast* podem ser usados em transmissões *multicast*. Duas estratégias são primordialmente empregadas na proteção de redes *unicast*: proteção e restauração. Essas técnicas já foram amplamente discutidas na sobrevivência das redes convencionais, e a distinção entre as duas consiste na escala de tempo diferente nas quais elas operam. Enquanto a proteção aloca recurso para a sobrevivência da rede antecipadamente, a restauração usa dinamicamente os recursos de sobrevivência. A principal vantagem da proteção é a rápida reação a falhas garantindo pequeno tempo de recuperação.

2.4. Trabalhos Relacionados

Nesta seção são descritos os principais trabalhos que inspiraram o mecanismo de proteção proposto nessa dissertação (seção 2.4.1 e 2.4.2) e que serão usados para comparação no capítulo de avaliação de resultados (seção 2.5.3).

2.4.1. Proteção por Árvores com Múltiplas Fontes

Como descrito na seção 2.2 o cenário exposto por [Walkowiak, 2009] e [Walkowiak e Przewozniczek, 2011] é relativamente estático, ou seja, não considera saídas voluntárias da rede por parte dos destinatários e dos nós intermediários. Tais cenários são utilizados em sistemas críticos (como os de previsão de tempestades), IPTV ou Rede de Distribuição de Conteúdo (*Content Distribution Network* - CDN).

Neste tipo de configuração de rede, todos os nós são conhecidos, e os autores definem todos os parâmetros da rede, na entrada do algoritmo. Assim, a quantidade de nós da rede, bem como quais serão os nós raízes e capacidades de *donwload* e *upload* dos nós já são definidas. As informações referentes às árvores também são conhecidas, portanto, a quantidade de árvores, bem como seus níveis possíveis e ainda a taxa de *streaming* são passadas na inicialização do programa. Por fim, os custos dos enlaces e a quantidade de ISPs e quais nós estão presentes em cada ISP também são pré-determinadas. O algoritmo constrói as árvores deixando os nós, com exceção da raiz, livres para se adaptarem à rede de acordo com as restrições iniciais de forma a atender as métricas propostas, definidas por meio da programação inteira mista.

A proteção é garantida sempre por múltiplas árvores disjuntas, não importando o tipo de falha que ocorra. Como possíveis cenários de falhas foram considerados a falha de enlaces de sobreposição, a falha de nós de *upload* entre as árvores e a falha de enlace do ISP. Portanto,

ao ocorrer uma falha em uma árvore, a proteção ocorre, sempre por meio de um fluxo redundante proveniente de outras árvores.

A figura 3 ilustra o modelo de proteção por múltiplas árvores. Nela, duas árvores *A* e *B* são estabelecidas na rede de sobreposição conectando oito nós *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*. O par *a* é a raiz da árvore *A* e o par *b* é a raiz da árvore *B*. Os demais nós *c*, *d*, *e*, *f*, *g*, *h* são receptores. No caso da árvore *A*, os nós *a*, *c* e *f* são nós de *upload*, enquanto os nós *d*, *e*, *g*, *h* são folhas. Os nós *a*, *c*, *d* e *e* pertencem ao ISP 1. Os nós *b*, *f*, *g* e *h* pertencem ao ISP 2. Por definição, o nó *a* está no nível 1 da árvore *A*, os nós *c* e *f* estão no nível 2 da árvore *A*, e assim por diante. A árvore *A* tem 2 níveis de nós de *upload*, e a árvore *B* possui 3 níveis.

O exemplo de configuração mostrado na figura 3(a) provê proteção apenas contra a falha do nó raiz, já que existem duas árvores com duas raízes separadas. Entretanto, esta configuração é vulnerável a outras possíveis falhas de rede. Por exemplo, o enlace de sobreposição (*c, d*) pertence a ambas as árvores, consequentemente, no caso de falha desse enlace, o nó *d* não estará mais conectado a nenhuma árvore. O par *c* é nó de *upload* nas duas árvores e a sua falha desconecta todos os sucessores. Finalmente, o enlace entre ISP 1 e ISP 2 é compartilhada por ambas as árvores, pelo enlace (*a*, *f*) da árvore *A* e na árvore *B* pelo enlace (*b*, *c*). A figura 3(b) mostra uma configuração de árvores que possui proteção completa contra falha de enlaces de sobreposição ou falha de nós de *upload*. Como no exemplo, existem apenas dois ISPs, não se pode garantir a tolerância a falhas para falha do enlace de ISPs.

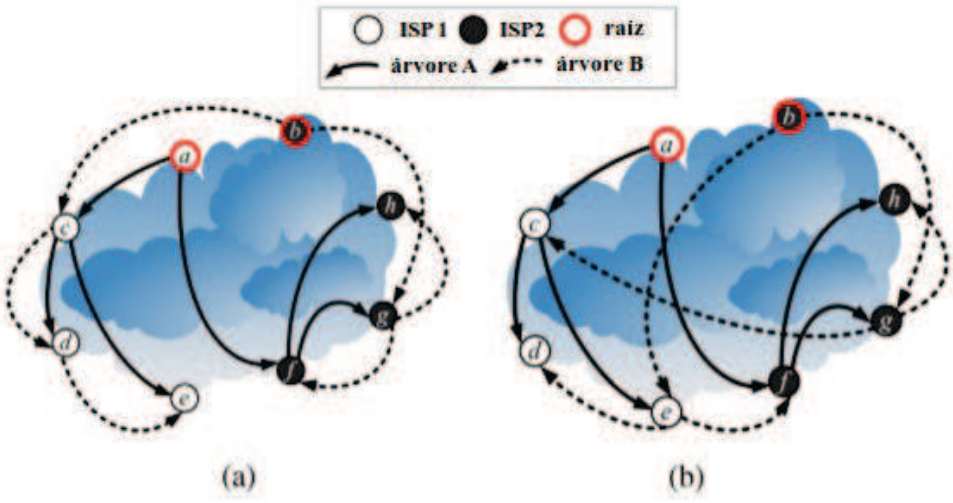

Figura 3: Exemplo de Árvores P2P *Multicast* [Adaptado de Walkowiak 2009]

Em [Walkowiak, 2009] os resultados foram avaliados através de duas métricas, custo

do fluxo (*streaming)* e vazão do sistema (*throughput*). O primeiro critério pode ser interpretado como consumo de banda ou como atraso induzido pela rede. O segundo critério denota a vazão do sistema da sessão de distribuição do conteúdo, ou seja, o volume de recepção de cada par [Walkowiak, 2009].

Os autores ainda definem uma proposta distribuída para quando as árvores *multicasting* P2P são criadas de uma maneira distribuída, ou seja, cada nó é responsável pela sua própria conexão à árvore de *multicast*. Nesse caso, os testes foram realizados por meio de um simulador, que se concentra no processo de criação das árvores. Duas estratégias para a construção de árvores foram definidas: a estratégia RSL (*Random Selection with Level Control*) onde se limita o número de níveis em da árvore em três tornando-a tanto menor quanto possível; e a estratégia CSL (*Cost Selection with Level Control*) que usa como objetivo o custo da rede também tentando limitar os níveis da árvore.

Como resultados, os autores demonstram que, no caso do custo dos fluxos em função do número de níveis, o custo cai com o aumento de níveis, entretanto, a partir de cinco níveis o custo converge para um valor estável. Já quando é levado em consideração o custo dos fluxos em função da taxa de transmissão dos fluxos (taxa de *streaming*), conforme a taxa de *streaming* aumenta, o custo passa a ser mais alto, já que a partir de uma determinada taxa alguns nós nós não podem enviar o *streaming* . Quanto à vazão do sistema, foi verificado que, para cerca de 60% das redes testadas, a vazão do sistema foi a mesma.

O estudo conclui que a exigência para a construção de árvores com disjunção de falhas não influenciam significativamente os objetivos das funções de custo e vazão do sistema. Consequentemente, a aplicação do *multicasting* P2P pode ser melhorada com critérios adicionais de sobrevivência sem degradação substancial do sistema em termos do custo de transmissão e a taxa de transferência do sistema.

Em [Walkowiak e Przewozniczek, 2011] o cenário a ser utilizado continua sendo o de uma rede P2P estática com a diferença de que quatro situações de falhas são consideradas ao invés das três do trabalho anterior: falhas de nó raiz, falhas de enlaces de sobreposição, falhas de enlaces do ISP e falhas de nós de *upload* entre as árvores. Além disso, as métricas de desempenho ganham mais um objetivo: o atraso máximo, além do custo de fluxos e da vazão do sistema. O terceiro critério reflete o atraso máximo de transmissão tendo em conta todos os pares de recepção.

Nesta nova abordagem, os autores implementam dois modelos para aplicar as métricas

definidas, utilizando-se da programação inteira mista. No primeiro modelo, chamado de modelo de fluxo, as métricas implementadas, foram de custo de fluxos e atraso máximo. Como em redes P2P as conexões entre os pares formam um grafo completo, ou seja, pode existir um enlace direto entre quaisquer dois pares, o número de variáveis de fluxo individuais é muito grande. Assim, foi introduzida uma formulação alternativa para reduzir a complexidade do modelo, denominada, modelo de níveis, que permite a definição de um valor limite na profundidade máxima da árvore.

Os testes foram realizados sobre a ferramenta CPLEX, sendo avaliado o custo de fluxos e o atraso máximo em função da taxa de transmissão. Os autores notaram que a taxa de transmissão de 256 kbps é uma espécie de limiar, ou seja, neste ponto o custo de transmissão aumenta drasticamente. Esta é uma consequência do fato de que a capacidade do enlace de *upload* de alguns nós é de 256 kbps. Assim, se a taxa for maior que 256 kbps, os enlaces de sobreposição não têm capacidade para carregar fluxos e outros com taxa de transmissão maior e devem ser usados para construir as árvores viáveis. Quanto ao atraso máximo em função da taxa de transmissão, o comportamento de todos os modelos é bastante estável, observando-se uma pequena redução do atraso máximo com a redução da taxa de transmissão. A conclusão é que os custos de fluxos são mais sensíveis à variação da taxa de transmissão do que o atraso máximo.

Quanto à vazão do sistema, os resultados mostram que as restrições adicionais de sobrevivência em sua maioria não influenciam a métrica. Apenas no caso da falha de enlaces de ISP e raízes localizadas no mesmo ISP é que a diferença é considerável. Observou-se que, se as raízes estão no mesmo ISP, o fluxo de todas as árvores estabelecidas devem ser enviados para pares em outros ISPs usando a capacidade de *upload* de todos os pares localizados no ISP. Em contraste, no caso de raízes localizadas em diferentes ISPs, a capacidade de *upload* de todos estes ISPs pode ser usada. Assim, a degradação de transferência do sistema em comparação com o cenário de servidor separado é muito maior no caso de raízes dentro do mesmo ISP. Além disso, nota-se que a diferença percentual é quase independente do número de níveis.

Os resultados foram promissores mas as desvantagens foram que a funcionalidade adicional do sistema, para garantir a sobrevivência em *multicasting* P2P, sempre provoca uma sobrecarga de gerenciamento. No caso de um sistema centralizado e otimização *off-line*, o elemento de controle deve levar em conta restrições de capacidade de sobrevivência para

estabelecer várias falhas em árvores separadas. Assim, os modelos de programação inteira mista são maiores (em termos do número de variáveis e restrições) em comparação com casos em que a sobrevivência não é levada em consideração, o que aumenta o tempo de execução. Assim, os resultados ótimos podem ser obtidos em um tempo razoável de uma hora em média, para redes relativamente pequenas composta por até vinte pares. Em um sistema distribuído, cada par é responsável por controlar as árvores que ficarão desconectadas. Consequentemente, gera-se tráfego de rede adicional para distribuir atualizações da topologia de árvores e os custos de processamento extra para analisar as informações recebidas.

2.4.2. Proteção por Árvores com Mesma Fonte

[Allen e Kubart, 2010] estabeleceram um trabalho para tolerância a falhas em transmissões de vídeo cujo foco é a transmissão em longas distâncias entre os chamados *super-head-ends* (SHEs) e os *video hub offices* (VHOs) usados em provedores de conteúdo de vídeo, explorando o uso da teoria de *Steiner trees* em transmissões por meio de árvores *multicast*, para formular modelos e propor uma solução de rede de custo mínimo.

O problema da *Steiner tree* consiste em achar a menor árvore que conecta um conjunto de pontos dados. As árvores geradoras mínimas conectam um conjunto de pontos dados e possuem custo mínimo, mas requerem que todas as conexões sejam entre estes pontos. A diferença, no problema da *Steiner tree*, é que se podem utilizar pontos extras, de modo que o custo da árvore gerada seja ainda menor do que o custo da árvore geradora mínima. O problema da *Steiner tree* mínima é formulado por meio da programação inteira como segue. Uma unidade de conteúdo a ser transmitida é criada para todo par origem/destino. Variáveis binárias representam o fluxo de conteúdo através da rede, as variáveis de decisão binárias descrevem a seleção de um enlace. Restrições de conservação de fluxo garantem sempre que uma unidade de conteúdo está sendo roteada continuamente da origem para o destino. As restrições de viabilidade forçam os fluxos a utilizarem somente os enlaces que foram selecionados pelo algoritmo, e as restrições de integridade proíbem a divisão do fluxo e proíbem a escolha de enlaces fracionados na solução [Allen e Kubart, 2010].

A proteção para uma única falha de um enlace em uma *Steiner tree* é bastante simples. O procedimento requer em um primeiro passo que uma *Steiner tree* seja encontrada, para que num segundo passo, para cada enlace nessa *Steiner tree*, seja criado uma unidade de conteúdo entre os pares do enlace. O terceiro passo consiste na formulação e resolução de um problema

de múltiplos fluxos de conteúdos com adições de restrições, as quais proíbem o uso do enlace selecionado para cada conteúdo criado no passo 2. Por fim, no passo 4, realiza-se o cálculo do custo total da solução, ou seja, a soma do custo da *Steiner tree* e o custo do enlace mantidos na reserva no passo três.

Os autores estendem o problema da *Steiner tree* para promover dupla conectividade. Para isso, explicam que a forma mais fácil é transmitir um segundo conjunto de conteúdos originadas do mesmo SHE, e que terminam nos VHOs. Assim, todo nó de destino recebe conteúdo duplicado da fonte, mas por caminhos completamente diferentes.

[Allen e Kubart, 2010] concluem seu trabalho demonstrando resultados que comprovam que o uso de vários esquemas de proteção por meio de compartilhamento de comprimentos de onda, torna adequada a substituição de sistemas tolerantes a falhas com um único comprimento de onda.

2.4.3. Redundância por Arestas Cruzadas

O *Probabilistic Resilient Multicast* (PRM) é um esquema para melhorar a tolerância à falhas do *multicast* em redes de sobreposição [Banerjee et al., 2006] através de encaminhamento randomizado, um componente proativo.

O esquema do PRM é definido como PRM (*n*, *p*) onde *n* é o número de parceiros aleatórios na rede P2P e *p* é a probabilidade de encaminhamento. No PRM, um nó descobre continuamente membros aleatórios da sessão e encaminha todos os pacotes recebidos para qualquer um destes membros com uma probabilidade especificada. Estes pacotes aleatórios ajudam a detectar e recuperar partições temporárias na árvore.

Como consequência das arestas adicionais, alguns nós podem receber múltiplas cópias do mesmo pacote. Além disso, cada nó da rede de sobreposição mantém um pequeno *cache* de supressão de duplicatas, que temporariamente aloja o conjunto de pacote de dados recebidos sobre uma pequena janela de tempo. Os pacotes que perdem o prazo de latência são descartados [Banerjee et al., 2006]. Por isso, o tamanho do *cache* é limitado pelo prazo da latência desejada pela aplicação. Na prática, o cache de supressão de duplicatas pode ser implementado usando ou *buffer* de reprodução já mantido pelas aplicações de transmissão de dados.

Cada nó da rede de sobreposição periodicamente descobre aleatoriamente um conjunto de outros nós na rede de sobreposição e avalia o número de perdas que ele compartilha com

estes nós aleatórios. [Banerjee et al., 2006] implementam um mecanismo de descoberta de nós no PRM. Nele, um nó descobrindo um grupo de membros aleatórios transmite uma mensagem *Discover* com um campo de tempo de vida (*time-to-live* – TTL) para os seu pai na árvore. A mensagem é randomicamente encaminhada de vizinho para vizinho, sem um novo rastreamento do seu caminho ao longo da árvore e o campo TTL é decrementado em cada salto. O nó em que o TTL chegar a zero é escolhido como o nó aleatório.

O encaminhamento aleatório é efetivo porque quando um grande número de nós falha, um nó presente em uma sub-árvore não particionada pode encaminhar dados, através de enlaces aleatórios, para um nó em uma sub-árvore particionada. Os nós da rede de sobreposição que receberem dados ao longo dessas arestas cruzadas escolhidas aleatoriamente, posteriormente também encaminharão os dados ao longo das arestas de árvores regulares e de arestas escolhidas aleatoriamente. Como as arestas cruzadas são escolhidas uniformemente de forma randômica, uma sub-árvore maior terá uma probabilidade maior de incidência de arestas cruzadas nela. Assim, conforme o tamanho da partição da árvore aumenta, o mesmo acontece com a sua probabilidade de reparação utilizando as arestas cruzadas.

O PRM adota um segundo componente, este reativo, conhecido por *triggered negative acknowledgement*, da mesma forma que uma proposta da classe de redundância por arestas cruzadas, encaminhando uma fração adicional da transmissão sobre enlaces transversais conectando pares aleatórios na árvore, onde cada nó da rede sobreposição utiliza uma *bit-mask* para cada pacote de dados encaminhado indicando quais os números de sequência anteriores ele já recebeu corretamente. Os receptores de pacotes de dados detectam os pacotes faltantes usando os intervalos nas sequências recebidas e enviam NAKs para o nó anterior para requisitar as retransmissões apropriadas.

As simulações realizadas por [Banerjee et al., 2006] consideram as métricas de taxa de distribuição de dados, atraso de distribuição e sobrecarga de dados. Os resultados demonstraram que o PRM pode ser usado em cenários reais envolvendo aplicações de transmissão de mídias. A baixa largura de banda, armazenamento e sobrecargas de processamento do PRM o tornam atrativo para aplicações de transmissões de dados com baixa e alta largura de banda. Além disso, as grandes taxas de recuperação, frequentemente acima de 97% sobre condições adversas, permitem ao PRM ser usado mais eficientemente em codificações de mídias (as quais normalmente possuem menor tolerância a perda de dados).

Entretanto, os pacotes de dados encaminhados aleatoriamente sobre as arestas cruzadas, potencialmente criam pacotes duplicados em alguns nós, na ordem de n x p que gera desperdício de parte da largura de banda disponível nos nós, já que o PRM adota uma taxa constante de pacotes duplicados mesmo com um número baixo de falhas e também utiliza muitas mensagens de controle por segundo para descobrir e manter a lista de pares de encaminhamento aleatórios que também aumenta a sobrecarga na largura de banda.

A proposta deste trabalho será comparada ao PRM, ambos mecanismos proativos para que se possa verificar a sobrecarga na largura de banda gerada no momento da reconfiguração das árvores, e assim determinar se as mensagens de divulgação para reconfiguração das árvores acarretarão em perdas de recursos dos nós.

2.5. Considerações Finais

Nesta dissertação, o trabalho dá continuidade aos estudos desenvolvidos por [Walkowiak e Przewozniczek, 2011]. Porém, as diferenças estão na realização de proteção para falhas dos nós intermediários em uma única árvore, utilizando-se do modelo de fluxo reproduzido e da programação inteira mista, respeitando-se as métricas e restrições desenvolvidas por [Walkowiak e Przewozniczek, 2011]. Assim, pode-se definir um novo modelo, que proteja a árvore primeiramente na própria árvore, ao invés de sempre recorrer a outras árvores para qualquer tipo de falha da rede *multicasting* P2P. Ressalta-se que neste trabalho não será abordada a proteção de ISP.

Para montar-se as árvores de proteção, para cada falha de um nó intermediário, é adotada a mesma ideia ensinada por [Allen e Kubart, 2010], ou seja, na montagem das árvores pelo algoritmo, retiram-se os nós intermediários que podem falhar, e ainda garante-se que não haverá transmissões de nenhum nó para os nós intermediários, e que nenhum desses nós de intermediários transmita para qualquer outro nó da árvore.

A proteção dos nós intermediários realizada dentro da própria árvore será mais um mecanismo de garantia de tolerância a falhas. Unindo-se a solução de [Walkowiak e Przewozniczek, 2011] para proteções utilizando-se de múltiplas árvores, mais as proteções a partir de uma única árvore, melhora-se consideravelmente as possibilidades dos sistemas considerados críticos e de transmissão de conteúdo ao vivo permanecerem ativos.

O método utilizado para o novo modelo, continuará utilizando a programação inteira mista, com a otimização realizada pela ferramenta CPLEX, pois sempre retorna o resultado

ótimo e ainda compreende uma grande quantidade de restrições de corte que garantem que a árvore esteja realmente conectada. Observa-se que esse trabalho contempla a obtenção de árvores de proteção apenas através de modelos de programação inteira, estando o desenvolvimento de heurísticas fora do escopo desta dissertação.

Capítulo 3

Mecanismo de Proteção de Fluxos *Multicast* em Redes P2P

Neste capítulo está relatada a contribuição deste trabalho, com a exposição do algoritmo gerador das árvores de proteção, bem como a descrição do mecanismo de proteção para sobrevivência para redes P2P *Multicast*.

3.1. Descrição do Problema

Em um modelo tradicional de uma rede P2P, os nós de comunicação são *hosts* finais e os enlaces são os *links* de *download* e *upload* de cada um desses *hosts*. A comunicação de fluxos *multicast* é realizada através de uma árvore, na qual o *host* que origina os fluxos (chamado de nó fonte) é a raiz da árvore, sendo o responsável pela transmissão dos fluxos da origem para os demais componentes da rede. Os *hosts* que são responsáveis por encaminhar os fluxos são os nós de *upload* (chamados nós intermediários), que também são consumidores dos fluxos gerados. E, por fim, temos os *hosts* de destino que apenas consomem os fluxos gerados, que correspondem às folhas na árvore de *multicast*, denominados nós folhas.

Em uma aplicação de transmissão de fluxos *multicast*, onde o conteúdo é transmitido aos demais nós da rede por um nó raiz, a falha desse nó é bastante crítica, já que inviabiliza os demais nós de receberem os fluxos. Para resolver esse problema, pode-se optar por delegar um novo nó raiz na mesma árvore, com as possibilidades devidamente alocadas no início do sistema, ou utilizar um mecanismo de proteção que gera múltiplas árvores, cada qual com um nó raiz, que envia o mesmo fluxo de dados em suas respectivas árvores. Dessa forma, no caso de falha do nó raiz, a redundância é garantida, pelo fluxo de dados da raiz de outra árvore.

Na ocorrência da falha de um nó intermediário, todos os seus sucessores são desconectados da árvore. A estratégia de uso de múltiplas árvores originadas em fontes distintas também se aplica a esse caso, desde que as árvores sejam planejadas para esse fim.

Uma solução ainda não explorada é a de se proteger a árvore de uma origem contra falhas dos nós intermediários, a partir da sua própria reconfiguração, por outras árvores que excluem os nós que falharam. Nessa abordagem, impede-se o desperdício de banda com transmissões simultâneas provenientes de múltiplas árvores fonte, mantendo-se um mecanismo de proteção eficiente. É importante, que ao se iniciar uma transmissão, todos os nós já possuam todas as configurações possíveis para este tipo de falha de modo a tornar eficiente a reconfiguração.

Na recuperação de um nó intermediário, a determinação da árvore de reconfiguração pode ser feita no momento de ocorrência da falha (como acontece no caso dos estudos de redes de *multicast* P2P dinâmicas), ou a *priori*.

No primeiro caso, as possíveis estratégias de recuperação são várias. Conforme discutido no capítulo 2, essas estratégias podem criar uma concentração de tráfego no nó avô do nó intermediário que falhou ou na raiz. O tempo para a árvore se recuperar pode ser longo, porque um nó pode ser redirecionado várias vezes antes que ele possa encontrar um nó com capacidade apropriada para aceitá-lo como filho. Caso opte-se por deixar que um nó filho tente a reconexão, de forma randômica, pode resultar em rejeição ao nó folha porque o grau do nó folha pode estar exaurido.

No segundo caso, a reconfiguração de menor custo é calculada antes da falha e armazenada nos nós em tabelas de reconfiguração. Cada nó possuirá uma tabela de reconfiguração apropriada para cada possível falha de um nó intermediário. Dessa forma, ao ocorrer uma falha de um nó intermediário, seus filhos podem selecionar uma nova reconfiguração que é ativada pelo mecanismo de proteção após um pequeno tempo de interrupção. Como o cálculo das árvores de proteção é realizado *off-line*, o tempo de cálculo deixa de ser crítico, e a nova árvore utilizada em caso de falha será sempre a de menor custo após a reconfiguração da árvore de *multicast*.

O caso mais simples de recuperação de falha de um nó é quando este é um nó folha. Neste caso nada precisa ser feito para reestruturar a árvore, pois ele apenas irá deixar vago um espaço na hierarquia mais baixa da árvore.

Outra possibilidade de falha estaria relacionada aos enlaces, mas em uma rede

multicasting P2P esse tipo de falha não faz sentido, já que os enlaces físicos entre os nós não existem, existindo apenas enlaces virtuais de uma rede de sobreposição. Nesse caso, o mecanismo de proteção poderia tratar a falha de um ISP, como feita por [Walkowiak e Przewozniczek, 2011], o que está fora do escopo deste trabalho.

O modelo de rede considerado é de uma rede P2P estática. O mecanismo proposto é um mecanismo de proteção que trata falhas do nó de origem a partir do cálculo *off-line* de múltiplas árvores com origens distintas e trata as falhas de nós intermediários sem troca da origem, através do cálculo *off-line* de árvores de reconfiguração, acionadas por um mecanismo de reconfiguração *on-line* através da utilização de tabelas de reconfiguração que são armazenadas em todos os nós da rede, antes das transmissões de dados iniciarem.

Como objetivo principal considera-se a descrição do mecanismo proposto, e a forma de recuperação da árvore no momento em que uma falha de nó intermediário ocorrer, descrevendo-se as tabelas de reconfiguração armazenadas e utilizadas pelos nós para a reconfiguração da árvore, e o esquema de troca de mensagens entre os nós pelos pseudo-códigos presentes em cada tipo de nó de uma árvore, que demonstram como os nós atuam.

3.2. Cálculo de Árvores para Recuperação de Nó Origem

Para geração de árvores para recuperação de nó origem, utiliza-se o modelo de programação inteira mista proposto por [Walkowiak e Przewozniczek, 2011], que é reproduzido a seguir. O estudo de [Walkowiak e Przewoznicze, 2011] propõe um modelo baseado no custo do fluxo de *multicast* e um modelo baseado nos níveis da árvore *multicast*, calculam múltiplas árvores levando em conta diversas restrições de sobrevivência. O modelo descrito a seguir, que é usado neste trabalho, é o modelo baseando no custo no fluxo de *multicast*. Foram retiradas do modelo as equações relacionadas com a proteção de ISP, já que tal cenário não é considerado neste trabalho.

Nesse modelo os índices v, w = 1, 2, …., V denotam os nós da rede de sobreposição que são participantes do *multicasing* P2P. Cada nó v possui capacidade de *upload* u_v e capacidade de *download* d_v. Há T árvores de transmissão indexadas por t = 1, 2, …, T. Se o par v é a raiz da árvore t (v é o nó de transmissão), então r_{vt} = 1, caso contrário, r_{vt} = 0. Para simplificar a notação, assume-se que o par $v = (V - T + 1)$ é a raiz da árvore t =1, e o par $v = (V - T + 2)$ é a raiz da árvore t = 2, e assim por diante. Cada par que recebe o fluxo da transmissão e não é o nó raiz, é representado pelo índice k = 1, 2, …, K. O número de pares

receptores é $K = (V - T)$. Considera-se a restrição de saltos *multicastin*g, ou seja, há um limite máximo L no número de saltos entre o nó raiz e qualquer outro nó. O limite de saltos pode ser interpretado como o limite máximo no número de níveis da árvore *multicast*.

O fluxo individual para cada par receptor, é representado pela variável binária chamada Y_{wvkt}, a qual é atribuído o valor 1, se o caminho da transmissão na árvore t para o nó receptor k inclui um enlace de sobreposição do nó w para o nó v (sem nenhum outro par entre eles), e atribui 0 caso contrário. Os enlaces de sobreposição são representados pela variável binária Y_{wvt}, a qual é atribuído o valor 1, se o enlace de sobreposição é usado na árvore *multicast t*, e o valor 0, caso contrário. A constante C_{wv} é o custo da transmissão no enlace de sobreposição (w,v), representando o atraso entre os nós. O objetivo é minimizar o custo de transmissão total dado por $\sum_{w}\sum_{v}\sum_{t} Y_{wvt} C_{wv}$. A tabela 1 apresenta os índices, constantes e variáveis usados no problema.

Índices	**Significado**
$v, w = 1, 2, \ldots, V$	Pares na rede P2P (raiz, intermediários e folhas)
$k = 1, 2, \ldots, K$	Pares receptores na rede P2P (intermediários e folhas)
$t = 1, 2, \ldots, T$	Árvores de *multicast*
Constantes	
d_v	Capacidade de *download* do nó v (kbps)
u_v	Capacidade de *upload* do nó v (kbps)
r_{vt}	= 1, se o nó v é a raíz (nó de *streaming*) da árvore t, = 0, caso contrário
q	Taxa de tranmissão (Kbps)
C_{wv}	Custo de transmissão dos fluxos no enlace de sobreposição (w, v)
L	Número de níveis
Variáveis	
Y_{wvkt}	= 1, se na árvore de *multicast t* o caminho de fluxos do nó raiz ao nó k inclui o enlace (w, v); = 0, caso contrário.
Y_{wvt}	= 1, se o enlace (w, v) está na árvore t; = 0, caso contrário.

Tabela 1: Índices, constantes e variáveis - Modelo de fluxo [Adaptado de Walkowiak e Przewozniczek, 2011]

A função objetivo e as restrições são apresentadas na tabela 2.

Função Objetivo	
Minimizar $\sum_{w}\sum_{v}\sum_{t} Y_{wvt} C_{wv}$	(1)
Restrições	
$\sum_{w} Y_{wvkt} - \sum_{w} Y_{vwkt} = 1,\ v=k\ \ v=1,2,\ldots,V\ \ k=1,2,\ldots,K\ \ t=1,2,\ldots,T$	(2)
$\sum_{w} Y_{wvkt} - \sum_{w} Y_{vwkt} = -1,\ r_{vt}=1\ \ v=1,2,\ldots,V\ \ k=1,2,\ldots,K\ \ t=1,2,\ldots,T$	(3)
$\sum_{w} Y_{wvkt} - \sum_{w} Y_{vwkt} = 0,\ v\neq k\ \ r_{vt}=1\ \ v=1,2,\ldots,V\ \ k=1,2,\ldots,K\ \ t=1,2,\ldots,T$	(4)
$Y_{wvkt} \leq Y_{wvt},\ v=1,2,\ldots,V\ \ w=1,2,\ldots,W\ \ k=1,2,\ldots,K\ \ t=1,2,\ldots,T$	(5)
$\sum_{w} Y_{wvt} = \left(1 - r_{vt}\right),\ v=1,2,\ldots,V\ \ k=1,2,\ldots,K\ \ t=1,2,\ldots,T$	(6)
$\sum_{v}\sum_{t} Y_{vvt} = 0$	(7)
$\sum_{w}\sum_{t} Y_{wvt}\, q \leq d_{v},\ v=1,2,\ldots,V$	(8)
$\sum_{v}\sum_{t} Y_{vwt}\, q \leq u_{v},\ v=1,2,\ldots,V$	(9)
$\sum_{w}\sum_{v} Y_{wvkt} \leq L,\ r_{vt}\neq 1\ \ k=1,2,\ldots,K\ \ t=1,2,\ldots,T$	(10)

Tabela 2: Função objetivo e restrições - Modelo de Fluxo [Adapatado de Walkowiak e Przewozniczek, 2011]

O objetivo (1) é minimizar os custos de transmissão. As equações (2)-(4) são as restrições de conservação de conservação de fluxo. Como se considera o *multicasting*, em cada árvore existe no máximo um par transmitindo pelo enlace de sobreposição (w, v) indiferente do número de nós receptores k que usam este enlace. A restrição (5) garante a ligação entre a variável de fluxo Y_{wvkt} e a variável de enlace Y_{wvt}. Para atender a exigência que o nó raiz da árvore t ($r_{vt} = 1$) não pode ser um nó intermediário, adiciona-se a restrição (6). A condição (7) assegura que o fluxo interno no nó seja zero. As restrições (8) e (9) são as restrições de capacidades de *download* e *upload*, respectivamente. Por fim, a restrição (10) define o limite máximo de níveis no caminho para cada nó receptor.

Como o problema é do tipo NP-completo, um método *branch-and-cut* é usado para reduzir a complexidade computacional na obtenção da solução ótima. A restrição (11) define os cortes que permitem cálculos com objetivos (*bound*) mais efetivos. O corte (11), representado pela equação da tabela 3, advém diretamente do fato de que o número de enlaces

em cada árvore deve ser $K = (V - T)$.

Restrição	
$\sum_{w}\sum_{k} Y_{wkt} = (V - T),\ t = 1,2,\ldots,T$	(11)

Tabela 3: Restrição de corte [Adaptado de Walkowiak e Przewozniczek, 2011]

A seguir são descritas as variáveis e restrições relativas à sobrevivência do *multicasting* P2P para os cenários de falha do enlace de sobreposição, falha do nó de *uploading* e falha do enlace de ISP. As variáveis adicionais são definidos na tabela 4, e as restrições de sobrevivência são mostradas na tabela 5.

Variável adicional	**Significado**
Y_{vt} (binária)	= 1, se o nó *v* está enviando *(uploading*) na árvore t; = 0, caso contrário

Tabela 4: Índices e variáveis para restrições de sobrevivência [Adaptado de Walkowiak e Przewozniczek, 2011]

Restrições	
Falha do Enlace de Sobreposição	
$\sum_{t}\left(Y_{wvt} + Y_{vwt}\right) \le 1,\ v = 1,2,\ldots,V\ \ w = 1,2,\ldots,W\ \ v < w$	(12)
Falha do nó de *uploading*	
$\sum_{v}\left(Y_{wvt} \le MY_{wt}\right),\ v = 1,2,\ldots,V\ \ t = 1,2,\ldots,T$	(13)
$Y_{wt} \le \sum_{v} Y_{wvt},\ v = 1,2,\ldots,V\ \ t = 1,2,\ldots,T$	(14)
$\sum_{t} Y_{vt} \le 1,\ v = 1,2,\ldots,V$	(15)

Tabela 5: Restrições de sobrevivência [Adaptado de Walkowiak e Przewozniczek, 2011]

A restrição (12) protege o *multicasting* P2P contra a falha de enlace de sobreposição. Nesse caso, ambos os enlaces direcionados, (*w*, *v*) e (*v*, *w*), são quebrados. Isso vem do fato que normalmente uma falha de rede influencia a transferência em ambas as direções.

As restrições (13)-(15) são para a falha do nó de *uploading*. A variável adicional Y_{vt} denota se um nó em particular está transmitindo na árvore *t*.

A proteção também pode ser realizada visando a minimização do tempo de transmissão. Nesse caso, cada enlace de sobreposição (w, v) está associado com um atraso na comunicação C_{wv} (dado em milissegundos). Para cada par receptor k e árvore t calcula-se o tempo de transmissão da raiz da árvore t até k através da fórmula $\sum_{w}\sum_{v}\sum_{t} Y_{wvt} C_{wv}$. O objetivo é minimizar o valor máximo do tempo de transmissão sobre todos os pares receptores em todas as árvores de transmissão.

Define-se a variável x, contínua e não negativa como sendo o atraso máximo e a função objetivo torna-se a minimização de x (16). A tabela 6 apresenta a função objetivo e as restrições utilizadas para esse objetivo.

Função Objetivo	
Minimizar x	(16)
Restrições	
Todas de (2)-(10) e todas de (11)-(15)	
$\sum_{w}\sum_{v} Y_{wvkt}\, C_{wv} \leq x,\ k=1,2,\ldots,K\ t=1,2,\ldots,T\ r_{vt}=1$	(17)

Tabela 6: Restrições para atraso máximo [Adaptado de Walkowiak e Przewozniczek, 2011]

A restrição (17) garante que x é o tempo máximo de transmissão de todos os fluxos individuais Y_{wvkt} considerando-se todos os receptores k = 1, 2, …, K e todas as árvores t = 1, 2, …, T.

3.3. Cálculo de Árvores para Recuperação de Nó Intermediário

A árvore de *multicast* inicial para cada origem é calculada pela formulação descrita na seção 3.2. Para o cálculo das árvores de recuperação dos nós intermediários, tal algoritmo foi modificado como apresentado na tabela 7. O objetivo é que as árvores de recuperação sejam calculadas sem a presença de cada um dos nós intermediários existentes na árvore inicial, correspondente à árvore de recuperação relativa à falha de cada um deles. A constante *node* incluída no modelo corresponde ao nó intermediário que deve ser excluído da árvore de recuperação.

Funções Objetivo	
Minimizar $\sum_{w}\sum_{v}\sum_{t} Y_{wvt} C_{wv}$	(1)
Minimizar x	(16)
Restrições	
$\sum_{w} Y_{wvkt} - \sum_{w} Y_{vwkt} = 1,\ v = k\ \ v = 1,2,\ldots,V\ \ k = 1,2,\ldots,K\ \ t = 1,2,\ldots,T$ $k \neq node\ \ v \neq node\ \ w \neq node$	(18)
$\sum_{w} Y_{wvkt} - \sum_{w} Y_{vwkt} = -1,\ r_{vt} = 1\ \ v = 1,2,\ldots,V\ \ k = 1,2,\ldots,K\ \ t = 1,2,\ldots,T$ $k \neq node\ \ v \neq node\ \ w \neq node$	(19)
$\sum_{w} Y_{wvkt} - \sum_{w} Y_{vwkt} = 0,\ v \neq k\ \ r_{vt} = 1\ \ v = 1,2,\ldots,V\ \ k = 1,2,\ldots,K\ \ t = 1,2,\ldots,T$ $w \neq node\ \ v \neq node\ \ k \neq node$	(20)
$Y_{wvkt} \leq Y_{wvt},\ v = 1,2,\ldots,V\ \ w = 1,2,\ldots,W\ \ k = 1,2,\ldots,K\ \ t = 1,2,\ldots,T$ $v \neq node w \neq node$	(21)
$\sum_{w} Y_{wvt} = \left(1 - r_{vt}\right),\ v = 1,2,\ldots,V\ \ k = 1,2,\ldots,K\ \ t = 1,2,\ldots,T\ \ v \neq node$	(22)
$\sum_{v}\sum_{t} Y_{vvt} = 0,\ v \neq node$	(23)
$\sum_{w}\sum_{t} Y_{wvt}\, q \leq dv,\ \ v = 1,2,\ldots,V\ \ v \neq node\ w \neq node$	(24)
$\sum_{v}\sum_{t} Y_{vwt}\, q \leq uv,\ \ v = 1,2,\ldots,V\ \ w \neq node\ \ v \neq node$	(25)
$\sum_{w}\sum_{v} Y_{wvkt} \leq L,\ r_{vt} \neq 1\ \ k = 1,2,\ldots,K\ \ t = 1,2,\ldots,T\ \ k \neq node\ \ w \neq node\ \ v \neq node$	(26)

Tabela 7: Restrições do novo modelo

As funções objetivo (1) (16) permanecem inalteradas, conforme o modelo que se deseja usar (minimização do custo do fluxo ou do tempo de transmissão). As restrições (18)-(26) possuem as mesmas funções descritas em (2)-(10), mas são alteradas para não permitir que o nó especificado pela constante *node* esteja presente na solução. Isto é, remove-se o nó intermediário desejado da lista de nós clientes, assim ele não está presente na árvore de reconfiguração.

Da mesma forma é necessário alterar as restrições de corte (27) e as restrições de sobrevivência (28)-(32) para excluir o nó intermediário desejado do modelo para a geração das novas árvores. Essas restrições são indicadas na tabela 8. As restrições referentes a falha do enlace de ISP não são incluídas, pois esse tipo de falha não é considerado.

Restrição de corte	
$\sum_{w}\sum_{k} Y_{wkt} = (V - T),\ t = 1,2,\ldots,T\ \ w \neq node\ \ v \neq node$	(27)
Restrição para falha do enlace de sobreposição	
$\sum_{t} \left(Y_{wvt} + Y_{vwt} \right) \leq 1,\ v = 1,2,\ldots,V\ \ w = 1,2,\ldots,W\ \ v < w\ \ w \neq node\ \ v \neq node$	(28)
Restrição para falha do nó de *uploading*	
$\sum_{v} \left(Y_{wvt} \leq MY_{wt} \right),\ v = 1,2,\ldots,V\ \ t = 1,2,\ldots,T\ \ w \neq node\ \ v \neq node$	(29)
$Y_{wt} \leq \sum_{v} Y_{wvt},\ v = 1,2,\ldots,V\ \ t = 1,2,\ldots,T\ \ w \neq node\ \ v \neq node$	(30)
$\sum_{t} Y_{vt} \leq 1,\ v = 1,2,\ldots,V\ \ v \neq node$	(31)
Restrição para atraso máximo	
$\sum_{w}\sum_{v} Y_{wvkt} C_{wv} \leq x,\ k = 1,2,\ldots,K\ \ t = 1,2,\ldots,T\ \ r_{vt} = 1\ \ k \neq node\ \ w \neq node\ \ v \neq node$	(32)

Tabela 8: Restrições de corte e de sobrevivência do novo modelo

Para proteger a rede *multicast* contra falhas de nós intermediários, múltiplas árvores de reconfiguração são construídas para a mesma origem, cada uma considerando a falha de um dos nós intermediários. O procedimento é apresentado na figura 4.

Passo 1: Para cada nó fonte (F):

Passo 1.1: Calcular a Árvore de multicast ótima para F (MF) de acordo com o modelo descrito no item 3.2.

Passo 1.2: Para cada nó de upload U de MF:

Passo 1.2.1: Calcular as árvores de multicast ótimas para cada MF do passo 1.1 de acordo com o modelo alterado e com as restrições adicionais que proíbem o uso do nó U, ambos descritos no item 3.3.

Figura 4: Algoritmo para Proteção de Falha de um Nó Intermediário

3.4. Mecanismo de Proteção

Essa seção descreve o mecanismo de proteção explicando-se a construção das tabelas de reconfiguração a partir das árvores de proteção e o protocolo de reconfiguração dos nós em

caso de falha do nó intermediário.

3.4.1. Tabela de Reconfiguração

Na seção 3.3 foi descrito o algoritmo capaz de gerar as árvores de reconfiguração para falhas de nós intermediários da rede. A partir das árvores de reconfiguração são produzidas as tabelas de reconfiguração que são distribuídas para cada nó. A tabela contém a configuração correspondente à árvore inicial, e as demais configurações de árvores possíveis correspondentes às árvores de reconfiguração de cada possível falha de um nó intermediário da árvore inicial.

Cada nó deverá manter várias tabelas de reconfiguração, uma para cada falha de nó intermediário. Por exemplo, em uma rede com dez nós, cuja árvore de *multicast* ótima possui três nós intermediários, para cada um é calculada uma árvore de reconfiguração e a tabela de reconfiguração correspondente. O mecanismo é ainda baseado em múltiplas árvores raiz, uma para cada falha de nó raiz, calculadas pelo algoritmo descrito na seção 3.2.

As árvores de reconfiguração são identificadas de acordo com as falhas de nós intermediários (apenas uma falha está sendo considerada) e falhas de nós raiz. Seja I a quantidade de nós intermediários na árvore inicial, que são identificados por números sequencias: 0, 1, 2, … I. As árvores são identificadas por $A_{i,j}$, onde i é o identificador do nó raiz ativo (i = 0, …,T), e j é o identificador do nó intermediário que falhou (j = 1, …, I). Quando não ocorrer falha em nenhum nó intermediário, usa-se o identificador 0. Por exemplo, $A_{1,0}$ identifica a árvore com nó raiz 1 e nenhuma falha de nó intermediário e $A_{5,3}$ identifica a árvore com nó raiz 5 e falha no nó intermediário 3.

A figura 5 ilustra o conceito de árvore de reconfiguração e sua forma de identificação. Para uma rede com dez nós e com limite de saltos de três níveis, a árvore principal (calculada pelo algoritmo da seção 3.2) tem a raiz no nó 0 e três nós intermediários (3, 4 e 8).

Serão geradas três árvores de reconfiguração, sendo que a figura 5 ilustra a árvore $A_{0,0}$ (inicial) e as árvores de reconfiguração correspondentes às falhas dos nós intermediários 3, 4 e 8, ou seja, as árvores de reconfiguração $A_{0,3}$, $A_{0,4}$ e $A_{0,8}$. As árvores para os demais nós raiz estão incluídas nos anexos dessa dissertação. Destaca-se que a árvore de reconfiguração $A_{0,8}$ possui apenas dois níveis, mesmo com o número inicial de saltos do exemplo estando limitado a três saltos. Isso acontece devido as características do modelo que geram árvores sempre de menor custo. Dessa forma, sem o nó intermediário 8, que falhou, o mecanismo consegue gerar

uma árvore de baixo custo com apenas dois níveis.

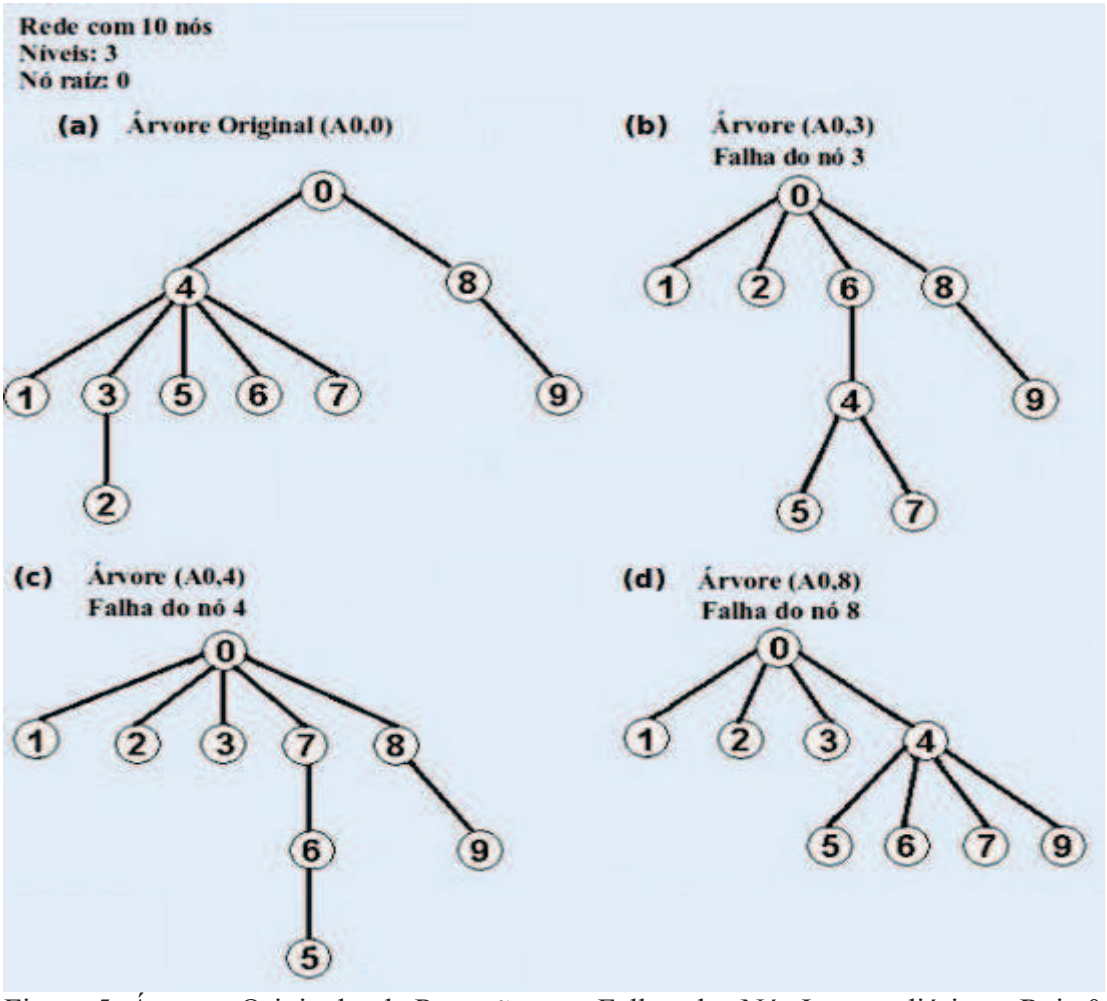

Figura 5: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 0

Na obtenção do menor custo de *streaming* e o menor atraso, os nós são livres para se reconfigurar de acordo com as restrições do modelo definidos na seção 3.3. Por isso, a estrutura de nós de algumas árvores altera-se bastante, mas como as árvores são sempre criadas antes de qualquer transmissão, todos os nós já possuem as tabelas de reconfiguração quando alguma falha acontecer. A construção de novas árvores, não se dará apenas no momento da falha, tornando a recuperação eficiente.

As tabelas de reconfiguração são identificadas por R_i, onde *i* é o identificador do nó raiz ativo. A tabela de reconfiguração para árvore com raiz no nó 0 (R_0), do exemplo da figura 5 está representada na tabela 9. As demais tabelas para árvores com outras raízes possíveis estão representadas nos anexos deste trabalho. Será necessária uma tabela de reconfiguração

para cada nó raiz considerado. Como pode ser observado, as tabelas de reconfiguração são representadas por matrizes, onde as colunas representam os nós em falha (0 para quando não há falha) e as linhas representam os nós. Em cada célula, é armazenada a lista de nós filhos.

Falha / Nó	0	3	4	8
0	4,8	1,2,6,8	1,2,3,7,8	1,2,3,4
1	-	-	-	-
2	-	-	-	-
3	2	-	-	-
4	1,3,5,6,7	5,7	-	5,6,7,9
5	-	-	-	-
6	-	4	5	-
7	-	-	6	-
8	9	9	9	-
9	-	-	-	-

Tabela 9: Tabela de reconfiguração para uma Árvore com Raiz no nó 0

A estrutura hierárquica e a distribuição da tabela de reconfiguração favorecem a construção deste mecanismo de tolerância a falhas. A não ser que mais do que um nó intermediário falhem ao mesmo tempo, o reescalonamento automático das árvores ativas tem como objetivo garantir o funcionamento da rede *multicasting* P2P em caso de falhas, melhorando a qualidade dos serviços, com menos desperdício de recursos, pela utilização de múltiplas árvores.

3.4.2. Protocolo de Reconfiguração

O esquema geral do protocolo de reconfiguração está representado nesta seção. O nó raiz também é um nó de *upload*, porém, não pode ser um nó intermediário, já que é a origem dos dados, e por consequência, não recebe transmissões de outros nós, apenas transmite. Os nós intermediários devem receber os fluxos de transmissão da raiz e de outros nós intermediários, e transmiti-los adiante para os demais nós da árvore.

Durante o processo de transmissão, dois nós trocam periodicamente mensagens *KEEPALIVE* para testar conectividade de rede e verificar se ambos os nós continuam

funcionando.

Existem duas razões por que o mecanismo usa mensagens *keepalive* [Comer, 2006]. Primeiro, a troca periódica de mensagens é necessária para que ambos os nós saibam se a conexão entre eles falhou. Segundo, *keepalives* conservam largura de banda em comparação com outras mensagens. Muitos protocolos de roteamento antigos usavam troca periódica de informações de roteamento para testar conectividade. Entretanto, como as informações de roteamento raramente mudam, o conteúdo da mensagem dificilmente é alterado. Além disso, uma vez que as mensagens de roteamento normalmente são grandes, reenviar a mesma mensagem desperdiça largura de banda de rede. Para evitar a ineficiência, o mecanismo proposto separa a funcionalidade de envio e recepção dos fluxos de dados do teste de conectividade, permitindo, que o mecanismo proposto envie pequenas mensagens *KEEPALIVE* frequentemente e reservando mensagens RECONFIGURAR para situações em que reconfigurações de árvores forem necessárias.

Um nó receptor especifica certo período (*timeout*) quando está recebendo dados; o *timeout* define um tempo mínimo que o mecanismo proposto deve esperar sem receber uma mensagem.

Assim, durante o processo de transmissão, caso não exista transmissão por parte do nó pai durante certo período (*timeout*), os nós filhos enviam mensagens *KEEPALIVE* para testar a conectividade com seus pais e se certificarem que eles ainda estão presentes. Em caso de constatação de falha, os filhos que perceberem a ausência do nó intermediário, enviarão mensagens RECONFIGURAR informando para todos os nós qual a nova árvore da tabela de reconfiguração deve ser utilizada.

O pseudo-código que especifica esse protocolo para envio e recepção dos fluxos foi idealizado utilizando-se o conceito de múltiplas *threads* rodando em paralelo. Dessa forma consegue-se separar o envio e recepção em relação aos fluxos de dados do teste de conectividade.

As figuras 6, 7 e 8 demonstram os códigos respectivamente para as *threads* de envio de mensagens KEEPALIVE, recepção de mensagens e *timeout*.

Envio de mensagens KEEPALIVE – *thread* 1

Conforme demonstra a figura 6, para o envio de mensagens de KEEPALIVE para o teste de conectividade entre os nós, quando um nó identificado pelo node_ID for diferente do

nó de origem (raiz), recupera-se o pai desse nó na árvore original (Recupera PAI em Arvore[Corrente]). Dessa forma, o node_ID fica enviando mensagens do tipo KEEPALIVE (Enviar(SENDER, TIPO_MENSAGEM, F_ID)), enquanto não há necessidade de reconfiguração da árvore (variável Reconfigura = FALSE) , pois não há falha (F_ID = 0).

Quando uma falha ocorrer, o node_ID deixará de enviar mensagens do tipo KEEPALIVE e ficará em estado de Pause, pois Espera = TRUE, significando que a *thread* deve ser interrompida, até que Espera = TRUE, pois o nó está aguardando a recepção de uma mensagem do tipo ATIVAR, que tem a função de iniciar a nova árvore, após todos os nós forem informados de uma falha. Após receber a mensagem do tipo ATIVAR, a variável Reconfigura deve ser FALSE, e Espera deve ser TRUE, para indicar que não há mais necessidade de uma reconfiguração de árvore. Por fim, Recupera-se o pai da árvore de reconfiguração corrente (Recupera PAI em Arvore[Corrente]).

```
Variáveis Globais
node_Id                 // Identificador do nó
Reconfigura = FALSE     // Flag para sincronização de reconfiguração
Espera = TRUE           // Flag para sincronização de reconfiguração
Arvore                  // Vetor de árvores
                        // Arvore[0] é a árvore original
                        // Arvore[ID] é a árvore correspondente à falha do nó ID
Corrente = 0            // Índice da árvore corrente

Thread 1 // Enviar mensagens KEEPALIVE
SENDER = node_Id
TIPO_MENSAGEM = KEEPALIVE
F_ID = 0                                    // O campo F_ID informa a arvore corrente
Se node_ID ≠ RAIZ
        Recupera PAI em Arvore[Corrente]    // Árvore original
        Enquanto (TRUE) faça
                Enquanto (Reconfigura = FALSE) faça
                        Enviar (SENDER, TIPO_MENSAGEM, F_ID) para PAI
                        Sleep(time)         // espera mínima para aguardar sem enviar mensagens
                Fim Enquanto
                Pause (Espera)              // Interrompe a thread enquanto Espera = TRUE
                                            // Estaria esperando recepção de mensagem ATIVAR
                Reconfigura = FALSE
                Espera = TRUE
                Recupera PAI em Arvore[Corrente]   // Árvore de reconfiguração
        Fim Enquanto
Fim Se
```

Figura 6: Envio de mensagens KEEPALIVE - *thread* 1

Recepção de mensagens – thread 2

O código para a recepção de mensagem é demonstrado na figura 7.

```
Thread 2 // Receber mensagens
Recupera FILHOS em Arvore[Corrente]                    // Árvore original
Para cada FILHO em Arvore[Corrente] faça
        Inicia Timer[FILHO]
        Retentativa[FILHO] = 0
Fim Para
Enquanto (TRUE) faça
        Receber (SENDER, TIPO_MENSAGEM, F_ID)
        Se (TIPO_MENSAGEM = KEEPALIVE)          // Recebe mensagem KEEPALIVE
                Reinicia Timer[SENDER]          // Reinicia temporizador do filho que enviou
        Fim Se
        Se (TIPO_MENSAGEM = RECONFIGURAR)       // Recebe mensagem RECONFIGURAR
        // Os nós que estão no caminho entre o nó que iniciou a recuperação e o nó raiz são avisados
        // para reconfigurar e propagam mensagem para a raiz
                Reconfigura = TRUE              // Interrompe envio de mensagens KEEPALIVE
                Se node_ID ≠ RAIZ
                        Recupera PAI em Arvore[F_ID]   // O campo F_ID informa a arvore corrente
                        TIPO = RECONFIGURAR            // Tipo da mensagem
                        SENDER = node_Id               // Informa quem está enviando
                        Enviar (SENDER, TIPO_MENSAGEM, F_ID) para PAI
                        Receber (SENDER, TIPO_MENSAGEM, F_ID)
                        Enquanto (TIPO_MENSAGEM != ATIVAR)   // Espera uma mensagem ATIVAR
                                Receber (SENDER, TIPO_MENSAGEM, F_ID)
                        Fim Enquanto
                Fim Se
                // Ativa nova configuração
                Corrente = F_ID
                Recupera FILHOS em Arvore[Corrente]
                TIPO_MENSAGEM = ATIVAR
                SENDER = node_Id                       // Informa quem está enviando
                // Propagar nova configuração para os filhos
                Para cada FILHO em Arvore[Corrente] faça
                        Enviar (SENDER, TIPO_MENSAGEM, F_ID) para FILHO  // Ativar filho
                        Inicia Timer[FILHO]
                        Retentativa[FILHO] = 0
                Fim Para
                Espera = FALSE                         // Reinicia envio de mensagens KEEPALIVE
        Fim Se
        Se (TIPO_MENSAGEM = ATIVAR)             // Recebe mensagem ativar
        // Os nós que não estão no caminho entre o nó que iniciou a recuperação e o nó raiz
        // são avisados para ativar nova reconfiguração
                Reconfigura = TRUE              // Interrompe envio de mensagens KEEPALIVE
                // Ativa nova configuração
                Corrente = F_ID                 // O campo F_ID informa a arvore corrente
                Recupera FILHOS em Arvore[Corrente]
                TIPO = ATIVAR
                SENDER = node_Id                // Informa quem está enviando
                // Propagar nova configuração para os filhos
                Para cada FILHO em Arvore[Corrente] faça
                        Enviar (SENDER, TIPO, F_ID) para FILHO  // Ativar filho
                        Inicia Timer[FILHO]
                        Retentativa[FILHO] = 0
                Fim Para
                Espera = FALSE                  // Reinicia envio de mensagens KEEPALIVE
        Fim Se
Fim Enquanto
```

Figura 7: Recepção de mensagens - *thread* 2

Para a recepção de mensagens, primeiramente é necessário recuperar os filhos da árvore inicial (Recupera FILHOS em Arvore[Corrente]). Dessa forma, para cada filho da árvore inicial inicia-se o *timer* de cada nó (Timer[Filho]) e zera-se o contador de retentativas de cada nó (Retentativa[FILHO] = 0), para que novas mensagens possam ser trocadas entre os nós, pois não existe falha nesse momento.

Ao receber uma mensagem (SENDER, TIPO_MENSAGEM, F_ID), o nó receptor receberá três informações nela, isto é, a identificação do nó que enviou a mensagem, o tipo da mensagem e a identificação do nó que falhou (0 quando não há falha). São possíveis três tipos de mensagens (TIPO_MENSAGEM) na recepção: uma mensagem do tipo KEEPALIVE, uma mensagem do tipo RECONFIGURAR e uma última mensagem do tipo ATIVAR.

Dessa maneira, quando um nó recebe uma mensagem, é verificado o tipo da mensagem. Se o tipo da mensagem for do tipo KEEPALIVE, a única tarefa a ser feita é a de se reiniciar o *timer* do nó que enviou a mensagem, pois as mensagens foram trocadas com sucesso, não havendo falhas.

Quando o tipo da mensagem for do tipo RECONFIGURAR, todos os nós que estão no caminho entre o nó que iniciou a recuperação e o nó raiz são avisados para se reconfigurarem e propagam essa mensagem de reconfiguração até o nó de origem (raiz). Nesse momento a variável Reconfigura deve ser TRUE, para que os nós que estão enviando mensagens fiquem em estado de Pause para aguardarem uma mensagem de ATIVAR, que significa que todos os nós já se reconfiguraram para usar a nova árvore de transmissão dos fluxos. Se o node_ID não for o nó de origem, recupera-se o pai da árvore de reconfiguração correspondente a árvore de reconfiguração sem o nó que falhou (F_ID). Assim, o nó informa que está enviando uma mensagem do tipo RECONFIGURAR para o seu pai na árvore de reconfiguração (Enviar(Sender, TIPO_MENSAGEM, F_ID)) e recebe mensagem do tipo RECONFIGURAR do seu pai na árvore de reconfiguração corrente (Receber(SENDER,TIPO_MENSAGEM, F_ID). O passo seguinte é receber as mensagens do tipo RECONFIGURAR até que uma mensagem do tipo ATIVAR seja recebida, dessa forma indicando que todos os nós da árvore de reconfiguração corrente receberam a mensagem de reconfiguração estando cientes da nova tabela a ser utilizada.

A seguir inicia-se o processo de ativação da nova configuração de árvore, para isso o campo F_ID informa a árvore corrente para que se possa recuperar os filhos da árvore atual (Recupera FILHOS em Arvore[Corrente]. Nesse momento o tipo da mensagem é do tipo

ATIVAR, e assim pode-se propagar a nova configuração para os filhos. Portanto, para cada filho na árvore de reconfiguração corrente, envia-se uma mensagem do tipo ATIVAR (Enviar(SENDER, TIPO_MENSAGEM, F_ID)), inicia-se o *timer* de cada nó filho (Timer[FILHO]), para que as mensagens de teste de conectividade sejam enviadas e zera-se o contador de retentativas novamente. Nesse ponto, a variável Espera passa a ser FALSE, para que o envio de mensagens entre os nós na *thread* 1 seja possível novamente.

Por último, é necessário tratar o caso em que os nós que não estão no caminho entre o nó que iniciou a recuperação e o nó de origem (raiz) são avisados para ativarem a nova reconfiguração. Assim, quando o tipo da mensagem recebida for do tipo ATIVAR, esses nós devem se reconfigurar. Nesse momento, a variável Reconfigura deve ser TRUE para que os nós fiquem em estado de Pause até receberem a mensagem ATIVAR. A seguir, recupera-se os filhos da árvore de reconfiguração atual, e propaga-se a nova configuração para os filhos, exatamente da mesma forma feita anteriormente, ou seja, enviando-se uma mensagem do tipo ATIVAR, com indicação do nó que falhou para identificar a árvore de reconfiguração a ser utilizada, reinicia-se o *timer* dos nós filhos e zera-se o contador de retentativas dos nós filhos. Finalmente a variável Espera deve voltar a ser FALSE, para que as mensagens de teste de conectividade voltem a ser envidadas entre os nós.

***Timeout – thread* 3**

A figura 8 demonstra o código da *thread* de *timeout*. Esta *thread* é invocada sempre que o *Timer* de um nó estourar (Timer[ID]). Assim, um contador de retentativas é iniciado (Retentativa[ID] = Retentativa[ID] + 1) e após um número limite de retentativas (Retentativa[ID] < Limite) uma mensagem de reconfiguração (TIPO_MENSAGEM = RECONFIGURAR) do node_ID par si mesmo deve ser enviada (Enviar(SENDER, TIPO_MENSAGEM, F_ID)), informando o nó que falhou (F_ID=ID). Caso o limite de retentativas não ultrapasse o limite, ou seja, houve uma confirmação de que não há falhas, o *timer* do node_ID é reiniciado.

```
Thread 3 // Timeout
Timeout(ID)                                   // Invocada quando Timer[ID] estoura
Retentativa[ID] = Retentativa[ID] + 1
Se (Retentativa[ID] < Limite )                // Somente quando estourar o limite deve reconfigurar
        Inicia timer[ID]
        Exit
Fim Se
// Enviar uma mensagem de reconfiguração para si mesmo
SENDER = node_Id                              // Informa quem está enviando a mensagem
F_ID = ID                                     // Informa nó que falhou
TIPO_MENSAGEM = RECONFIGURAR
Enviar (SENDER, TIPO_MENSAGEM, F_ID) para node_ID
```

Figura 8: *Timeout* - *thread* 3

Exemplificando

Para exemplificar o protocolo, observa-se novamente o exemplo da figura 5. A primeira árvore, ilustrada pela figura 5(a) é a árvore original. A segunda árvore, figura 5(b), é a árvore de reconfiguração da falha do nó intermediário 3. O desenho da topologia lógica quando o nó 3 falha está representado na figura 9. Nela, representa-se no topo da figura a árvore inicial, com uma falha ocorrendo no nó intermediário 3, e na parte intermediária da figura estão representadas as mensagens de reconfiguração. Nesse exemplo, o mecanismo de reconfiguração é iniciado pelo nó filho 2. Na parte inferior da figura está representada a árvore reconfigurada após a falha do nó intermediário 3. Por fim, à direita estão as tabelas que demonstram os enlaces para as árvores de reconfiguração para as falhas de todos os nós intermediários possíveis na árvore inicial.

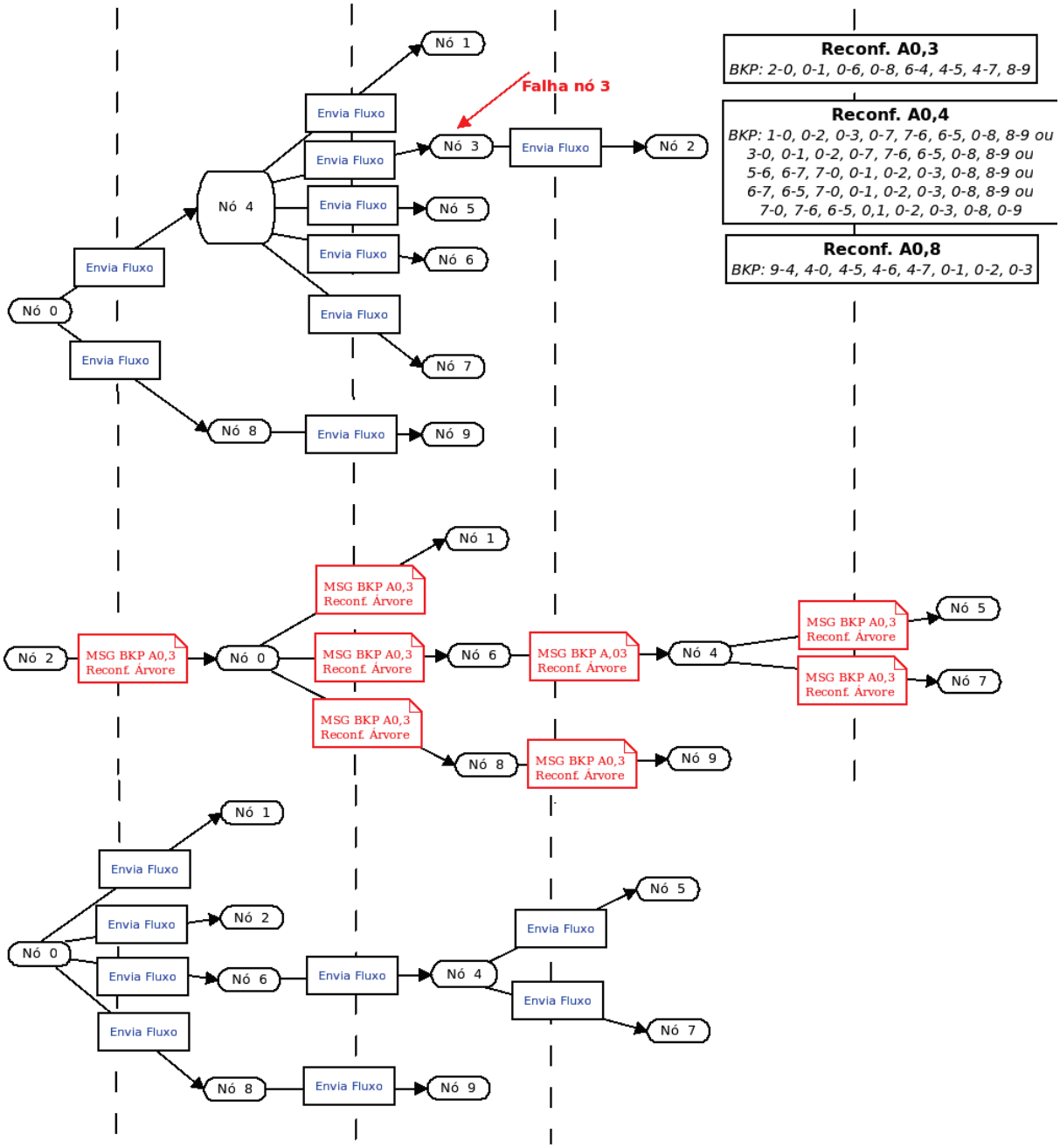

Figura 9: Topologia lógica para árvore com falha no nó 3

A troca de mensagens entre os pares da rede, representando a falha do nó intermediário 3, está representada na figura 10.

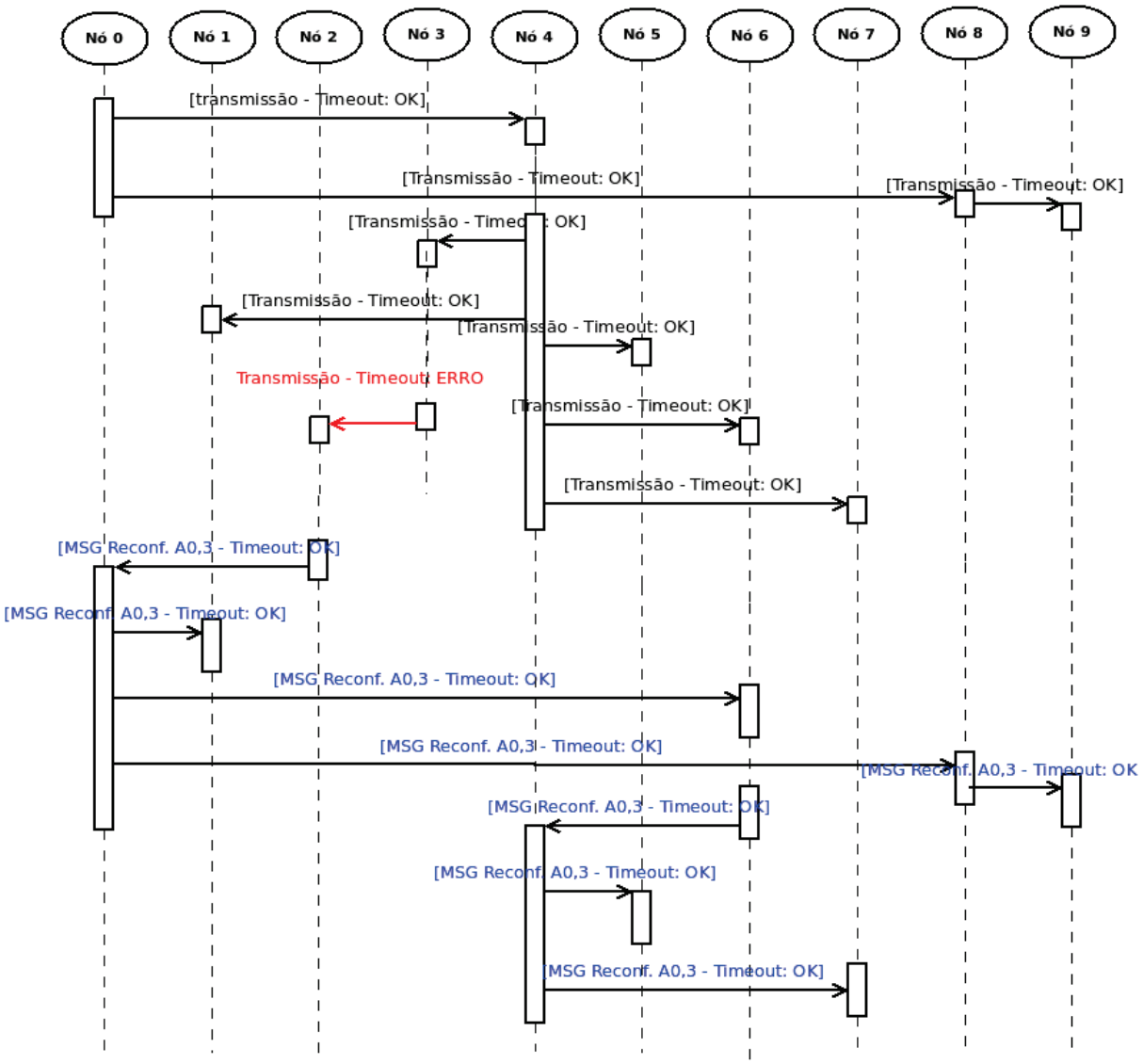

Figura 10: Troca de mensagens para reconfiguração da falha do nó 3

Como mostra a figura 10, após um período de tempo (*timeout*), o nó 2 envia uma mensagem de *KEEPALIVE*, ao não receber resposta do seu pai o nó 2 saberá que está isolado do restante da rede e inicia a reconfiguração da árvore através da divulgação de uma mensagem *RECONFIGURAR*. Nesse instante, o nó 2, através da árvore de reconfiguração correspondente a configuração que exclui o nó intermediário 3, informa ao nó raiz, através de seu novo pai (o nó 0) de que a partir daquele momento a tabela de reconfiguração precisa ser alterada devido a ausência do nó 3. O nó 0 (a raiz) por sua vez divulgará a mensagem de reconfiguração com a nova tabela de reconfiguração válida para os nós 2, 6 e 8 a ser usada. O nó 6 avisará o nó 4 de que a tabela de reconfiguração foi alterada, que por sua vez repassará a

mensagem para os nós 5 e 7. Por fim, o nó 8 enviará a mensagem para o nó 9, informando-o da tabela de reconfiguração a ser usada a partir deste momento.

No caso de falha do nó intermediário 4, cuja árvore de reconfiguração é representada pela figura 5c, a situação é um pouco mais crítica, pois existem mais nós afetados, já que os nós 1, 3, 5, 6 e 7, são filhos deste nó intermediário. Se o nó 4 falhar, em algum momento um ou mais desses nós filhos perceberão a ausência do nó 4 (após o *timeout*, e após tentar contatá-lo sem sucesso). Assim, pelas características da árvore os nós filhos 1 e 3, usando a nova tabela de reconfiguração (que exclui o nó 4) podem enviar as mensagens de reconfiguração para o nó 0 (a raiz) que passa a ser o pai destes nós que enviaram as mensagens de reconfiguração. Como a raiz receberá avisos redundantes sobre a mesma falha de um nó, simplesmente ignorará o aviso. O nó 6 envia a mensagem de reconfiguração para o nó 7, seu novo pai, e para o nó 5 que devido a nova configuração, passou a ser seu filho. O nó 7 por usa vez informará seu pai também utilizando a nova tabela de reconfiguração, nesse caso o nó 0 (raiz), e também enviará uma mensagem de reconfiguração da árvore para o seu filho, o nó 6. Aqui também haverá redundância de mensagens, e alguma delas será descartada. Por fim, o nó 0 (raiz) informará os nós 0 e 8 da nova tabela a ser utilizada. E finalmente o nó 8 informará ao nó 9 sobre a nova árvore a ser utilizada. A figura 11, representa a troca de mensagens para falha do nó 4.

Como o nó 4 possuía mais de um filho, dessa forma é bem provável que mais de um dos seus filhos perceba a ausência do nó 4, antes que a nova tabela de reconfiguração seja disseminada para todos os nós da árvore. Os nós filhos do nó 4, ao descobrirem um evento de falha, enviam aos respectivos pais na árvore de reconfiguração de falha do nó 4 uma mensagem de recuperação avisando-os do novo evento ocorrido. Essas mensagens se propagarão até o nó raiz que ao receber uma mensagem de recuperação já divulgada por algum outro nó, este descarta a mensagem redundante. Observa-se ainda que o nó 2 poderia ser induzido a acreditar na falha do nó 3, mas essa situação não ocorre devido à mensagem de *KEEPALIVE* enviada após o *timeout*, que será respondida pelo nó 3.

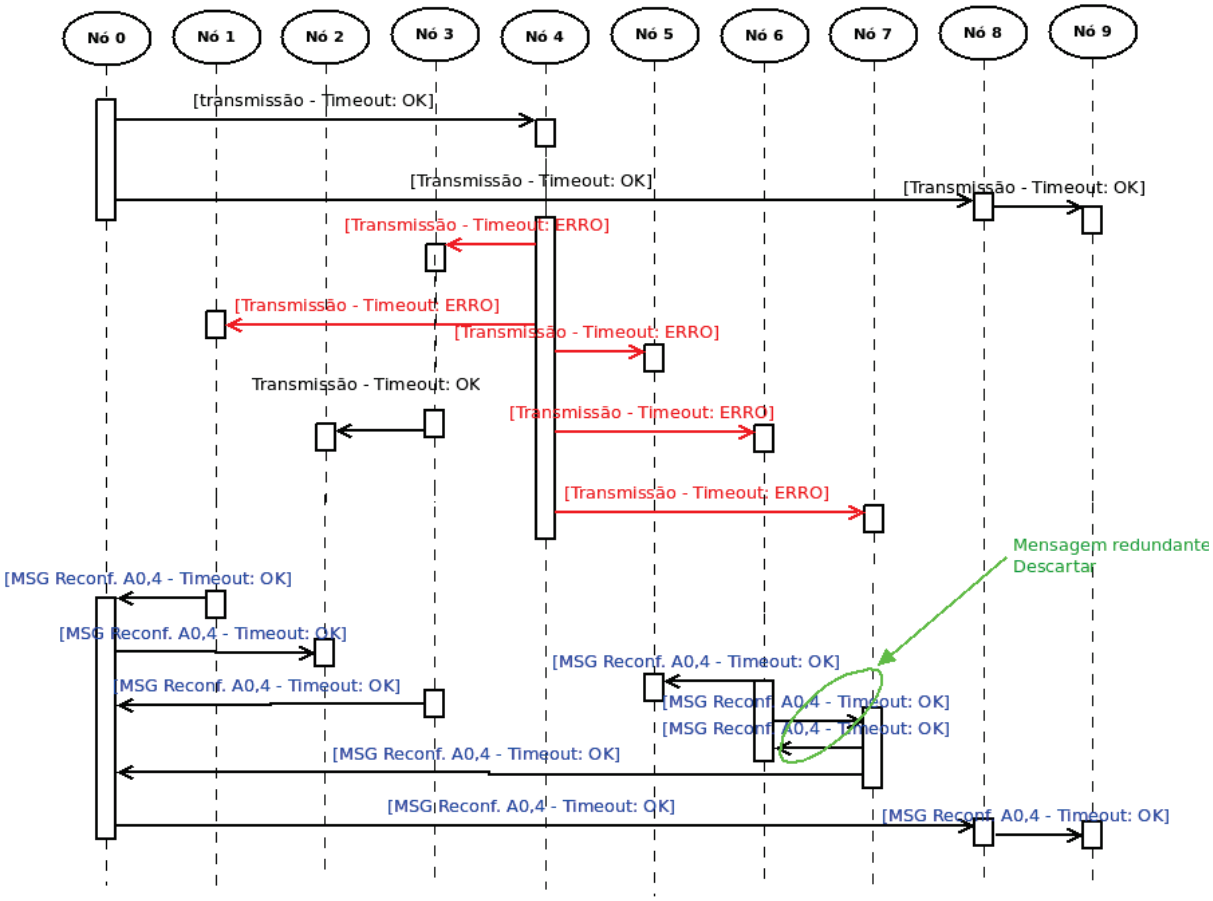

Figura 11: Troca de mensagens para reconfiguração da falha do nó 4

No último caso, representado pela figura 5(d), o último nó de *upload* que pode vir a falhar é o nó 8. Nesta situação caberá ao seu filho, o nó 9, após o *timeout*, tentar comunicar-se com o nó 8, através do envio de mensagem *KEEPALIVE*. Não havendo sucesso, o nó 9, através da tabela de reconfiguração sem a presença do nó 8, informará o nó 4, seu novo pai, da ausência do nó 8 e consequentemente da nova tabela a ser utilizada. Na nova configuração da árvore, o nó 4, envia mensagem de informação para o seu pai, o nó 0 (raiz) e para seus filhos, os nós 5, 6 e 7 do ocorrido. Por fim, caberá ao nó raiz informar aos seus filhos, os nós 1, 2 e 3, da nova tabela a ser utilizada. As trocas de mensagens para esse caso estão representadas na figura 12.

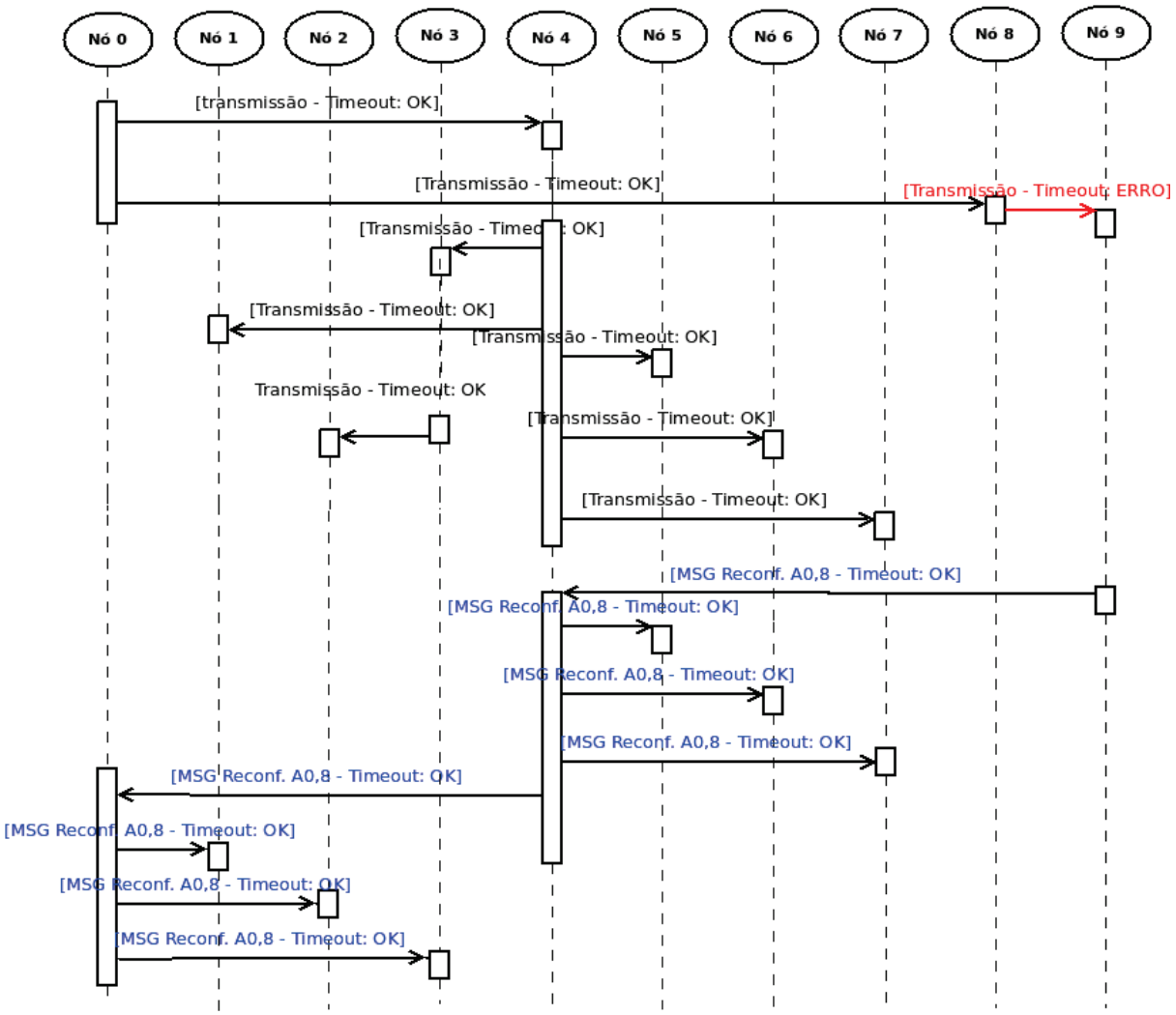

Figura 12: Troca de mensagens para reconfiguração da falha do nó 8

Com esse exemplo, nota-se que possuindo todas as tabelas antes das falhas ocorrerem, o sistema de proteção torna-se bastante robusto, não gerando sobrecarga em um único nó, geralmente o nó raiz ou o nó pai do nó intermediário que falhou, como em muitos trabalhos desenvolvidos, onde os recursos dos nós podem ser exauridos quando falhas acontecerem. Dessa forma, não se alteram os objetivos iniciais (custo do *streaming* e atraso máximo) ao se estabelecerem novas árvores. Com isso, a proteção ocorre dentro da própria árvore, não havendo necessidade de uso de múltiplas árvores, para envio de fluxos de dados, não consumindo recursos da rede de forma desnecessária por meio de fluxos redundantes por outros enlaces, que podem nem mesmo serem utilizados.

O mecanismo descrito pressupõe que apenas uma falha ocorre de cada vez, e que

durante a execução da reconfiguração nenhum outro nó não pode falhar e o nó que falhou não pode recuperar-se antes que o evento anterior tenha sido disseminado pela rede e que a falha de um nó não particiona a rede.

Capítulo 4

Avaliação do Mecanismo Proposto

4.1. Considerações Iniciais

O modelo de [Walkowiak e Przewozniczek, 2011] foi desenvolvido para proteção de árvores *multicast* em redes de sobreposição P2P, para manter as transmissões ativas quando ocorrer falha no nó de origem das transmissões, no enlace dos ISPs e no nó de *upload* entre as árvores, sempre utilizando ao menos duas árvores. Tal característica faz com que o custo das árvores desse modelo seja maior, visto que várias árvores são mantidas, visando a proteção.

Quando ocorrer alguma das falhas citadas no modelo de [Walkowiak e Przewozniczek, 2011] a solução é que a árvore afetada sempre receba as transmissões de uma das outras árvores. Esse mecanismo tem um tempo de recuperação eficiente, já que quando uma transmissão é interrompida, essa é imediatamente restaurada, já que os fluxos estão sendo enviados simultaneamente por todas as árvores de proteção. Portanto, a quantidade de largura de banda necessária para tal proteção é maior do que a necessária para transmissões por meio de uma única árvore, visto que alguns enlaces receberão vários fluxos repetidos, que, quando não necessários são descartados, desperdiçando-se recursos da rede.

Outra consideração sobre o modelo de [Walkowiak e Przewozniczek, 2011] é que a única proteção de nó realizada é quando ocorre falha do nó fonte, ou seja, a origem das transmissões, onde a transmissão dos dados é feita pelo nó fonte de outra árvore. Não há proteção na própria árvore, para falhas de nós intermediários ou nós folhas.

No modelo apresentado nesta dissertação a proteção foi aplicada para falha de nós intermediários em uma única árvore. Dessa forma mantendo-se uma única árvore, obtém-se um custo menor do que várias árvores juntas. Isso porque apenas uma árvore está ativa por vez, mesmo que todas tenham sido geradas de forma *off-line* e armazenadas nos nós antes das transmissões iniciarem.

Por outro lado no modelo proposto nesse trabalho, haverá um tempo de recuperação, já que todos os nós precisam saber qual é a nova árvore de reconfiguração a ser utilizada. Nesse ponto, o mecanismo, devido ao seu tempo de reconfiguração, passa a ser dinâmico.

4.2. Implementação do modelo de cálculo de árvores de proteção

Para efetuar os cálculos dos modelos, e se chegar a resultados ótimos, foi utilizado a ferramenta IBM ILOG CPLEX [CPLEX, 2009], que oferece bibliotecas em C, C++, Java,. NET e Python que resolvem problemas de programação linear (LP – *linear programming*) e problemas relacionados. Especificamente, ele resolve de forma linear ou de forma quadrática problemas de otimização restrita onde o objetivo a ser otimizado pode ser expresso como uma função linear ou uma função quadrática convexa.

Pelas características apresentadas, a linguagem C++ é a opção adotada para a aplicabilidade da proposta, pois é uma linguagem altamente estruturada e que possibilita a criação de bons algoritmos para estimar os benefícios de tempo de computação de funções, principalmente quando estas são chamadas com muita frequência, e ainda podem ser responsáveis por uma pequena porção do tempo de execução do programa, como por exemplo, uma função de avaliação de força bruta ou otimização estocástica de um processo.

Embora este trabalho aborde um mecanismo de proteção para falha de apenas um único nó intermediário, a falha simultânea de mais de um nó intermediário pode ser tratada seguindo o mesmo princípio. Para isso, bastaria excluir da reconfiguração da árvore todos os nós em falha (no caso de múltiplas falhas). Porém se a rede não possuir muitos nós presentes, haverá a possibilidade dos nós restantes formarem uma rede com poucos recursos, resultando em uma rede com solução inviável.

4.3. Metodologia de Validação

O mecanismo proposto foi avaliado em três aspectos distintos: o tempo de recuperação da transmissão multicast em caso de falha, a qualidade da árvore de multicast restaurada medindo-se o custo das árvores e a sobrecarga de largura de banda introduzida pelo protocolo de recuperação.

A seleção da métrica foi o primeiro passo antes de realizar os experimentos para observar o tempo de reconfiguração de uma única árvore quando a falha de algum nó intermediário ocorrer. Para a seleção da métrica, como apresentado por [Jain, 1991] antes de

iniciar uma avaliação de desempenho é preciso definir quais serão os fatores fixos (fatores que não irão sofrer mudanças em todos os experimentos, mantendo o mesmo ambiente para todo o conjunto de experimento) e fatores variáveis (fatores que passarão pela avaliação de desempenho para definir qual proporciona melhores resultados).

A tabela 10 apresenta os fatores que foram fixados para a realização dos experimentos. A primeira parte, relacionado à máquina, é o computador real onde a simulação foi realizada. Já as informações referentes ao simulador são como foi testado o ambiente de uma rede P2P.

Máquina Virtual	**Valores**
Processador	Intel Core 2 Duo
Memória	512MB
Sistema Operacional	Windows XP
Ambiente de Virtualização	VirtualBox
Ambiente de Programação	Microsoft Visual Studio 2005
Ferramenta de Otimização	ILOG CPLEX 12.1
Simulador	
Sistema Operacional	GNU/Linux Ubuntu 10.10
Ambiente de Programação	GCC - GNU C/C++
Número de Árvores	1

Tabela 10: Fatores fixos definidos para todos os experimentos

São utilizados quatro fatores variáveis. No primeiro fator de aleatoriedade estão o número de nós, cujas simulações foram feitas com 10, 12, 14, 16 e 18 nós.

[Linnolahti, 2004] apresenta duas métricas principais no desenvolvimento de políticas de transmissão: quantidade de saltos (*hops*) e latência entre um nó de origem e um nó de destino. Apesar de tais metas serem voltadas às políticas de roteamento, tal metodologia pode ser empregada em redes P2P, já que nelas os pares acabam agindo como roteadores nas transmissões na rede.

Dessa forma, o segundo fator definido foi a quantidade de saltos, que dependem da restrição de número de saltos máximos, definido na entrada do modelo. Neste caso, para todos os experimentos foi considerado um limite máximo de três saltos por nós, e são armazenados para cada nó, sempre que o programa estabelece o caminho de menor custo para cada nó. Já o terceiro fator foi um tempo aleatório representando a propagação dos dados pelos enlaces da

rede P2P. Nesse caso optou-se por utilizar a biblioteca GSL, da linguagem C, que permite realizar uma distribuição exponencial, com uma média de tempo de propagação por um enlace fixada em 5 ms [Kurose e Ross, 2010].

O quarto fator aleatório foram os próprios custos dos enlaces do modelo que representam o atraso entre os nós da rede, em *ms* [Walkowiak e Przewozniczek, 2011]. Apesar da rede P2P possuir um comportamento estático e, portanto, possuir seus custos bem definidos e conhecidos, essa variação é interessante para simular que os pares da rede P2P estão localizados em diferentes regiões, com possíveis escalas de distâncias diferentes. Neste trabalho, foram gerados os custos aleatórios, com valores entre 3 ms e 109 ms, apenas para a árvore original, já que a geração aleatória dos custos, no momento em que o modelo passa a gerar as árvores de reconfiguração, descaracterizariam a proteção, pois os custos da árvore original seriam alterados e os resultados seriam incorretos.

O simulador para o cálculo do tempo de reconfiguração dos nós nos momentos das falhas é bastante simples. Após a execução do modelo, já possuindo, portanto, os nós intermediários e seus filhos, os experimentos iniciam-se através de uma heurística.

Em um primeiro momento, sorteiam-se aleatoriamente um dos possíveis nós intermediários que podem falhar. No passo seguinte executa-se um algoritmo de caminho mais curto (*shortest-path*) para obter-se o número de saltos e o custo total desde a fonte dos dados (a raiz) até todos os outros nós receptores. Na etapa seguinte, excluindo-se o nó intermediário que falhou, calcula-se o tempo de reconfiguração da árvore, seguindo os critérios explicados nesta seção anteriormente, para então, na última etapa armazenar o menor tempo de subida e o maior tempo de descida dos nós no momento da recuperação, e assim calcula-se o tempo de reconfiguração da árvore para uma falha de um nó intermediário.

	Média	**Desvio Padrão**	**Coeficiente de Variação**
10 Nós	114,71	8,97	8,00%
12 Nós	127,08	11,2	9,00%
14 Nós	129,25	10,71	8,00%
16 Nós	131,29	13,44	10,00%
18 Nós	133,02	12,52	9,00%

Tabela 11: Média, desvio padrão e coeficiente de variação

As simulações foram executadas cem vezes, até os valores estabilizarem-se estatisticamente. Ao final da simulação, calculou-se a média, o desvio padrão e o coeficiente de variação do tempo de reconfiguração (em milissegundos), representados na tabela 11, para avaliação dos resultados.

4.4. Avaliação dos Resultados

4.4.1. Tempo de Recuperação

O tempo de recuperação indica o quão rápido o sistema pode restaurar a árvore após a falha de um nó intermediário. O tempo de recuperação é composto pelo tempo de propagação da árvore de recuperação, pelo tempo introduzido pela perda de mensagens e pelo tempo de detecção de falhas.

Tempo de propagação da árvore de recuperação

A primeira métrica utilizada para medir a capacidade de resposta é o tempo médio de propagação da árvore de reconfiguração, que é o tempo médio correspondente à troca de mensagens após um nó intermediário falhar, simulado de acordo com a descrição na seção 4.3. Para cada execução do programa, calculou-se a soma dos tempos de propagação, seguido do cálculo da média e da variância e do desvio padrão.

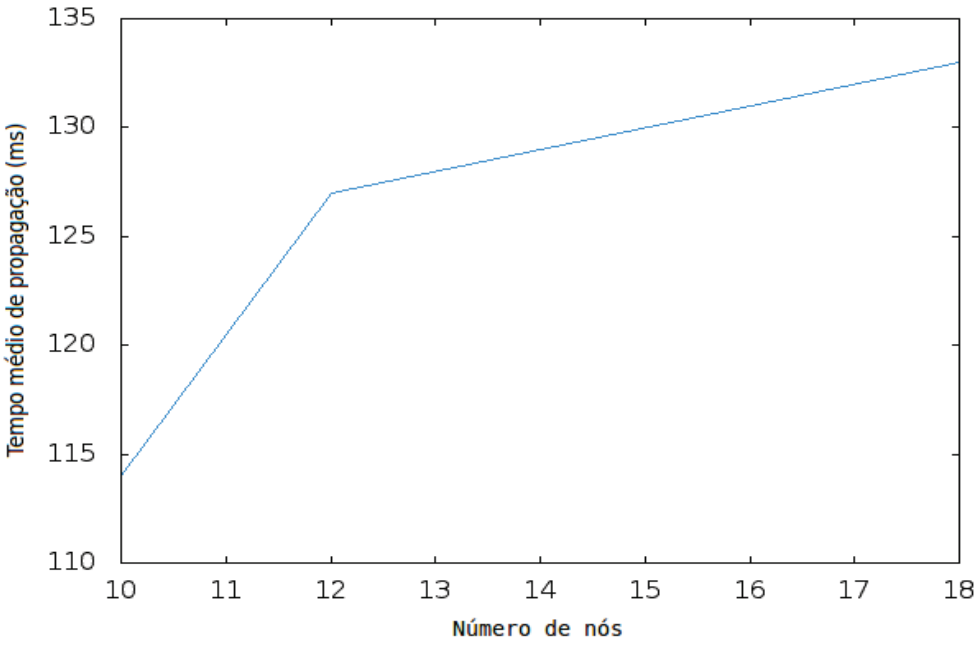

Figura 13: Tempo médio de propagação da árvore de reconfiguração

O gráfico da figura 13 apresenta o tempo médio de propagação da árvore de reconfiguração para a quantidade de 10 a 18 nós.

Pode-se observar que o tempo de propagação aumenta, conforme o número de nós aumenta, pois o custo do sistema aumenta. Mas, o tempo de propagação tende a estabilizar depois de cerca de 130 ms. Isso vem do fato que, o aumento do número de nós torna mais abundante a capacidade de *upload* da rede. Como resultado, os pares podem estabelecer mais conexões para os nós filhos, o que pode decrementar o número de níveis da árvore e em consequência reduzir o atraso. A figura 14 apresenta o gráfico com os tempos médios de propagação com relação aos nós, mostrando o intervalo de confiança, com nível de confiança de 99%.

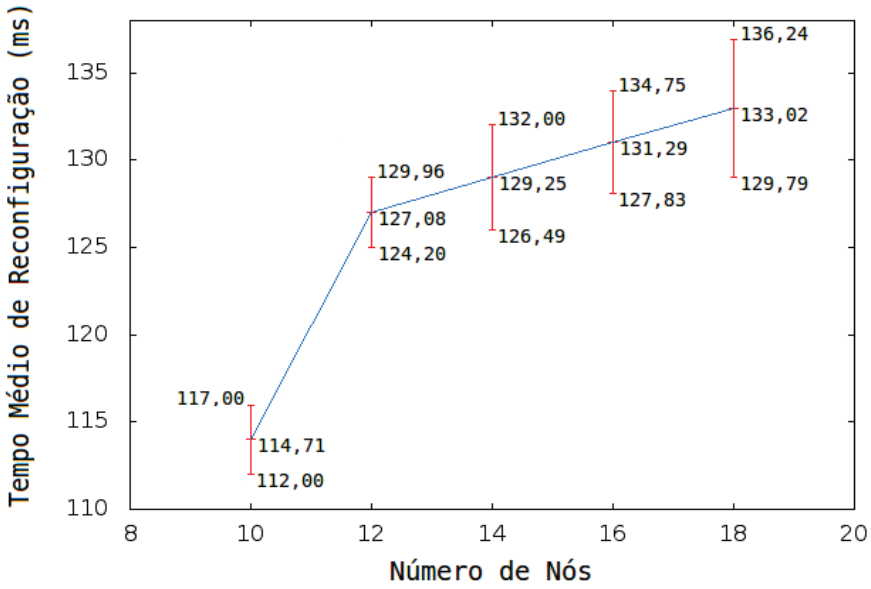

Figura 14: Diferença nos tempos de reconfiguração

Tempo introduzido pela perda de mensagens

A seguir avaliou-se o impacto da perda de mensagens no tempo de reconfiguração. A mensagem de reconfiguração enviada por um nó para o seu novo pai, através da árvore de reconfiguração pode ser perdida. Neste caso, o nó terá que retransmitir a mensagem, e isto pode aumentar o tempo de reconfiguração. Assume-se que a taxa de perda de enlace na topologia subjacente é $p\_enlace = 1\%$. A taxa de perda no caminho fim-a-fim será de $p = 1 - (1 - p\_enlace)^e$, onde e é o número de enlaces entre um par de nós [Fei e Yang, 2007]. Na

média $e = 1{,}89$, e assim a taxa de perda no caminho é $p \simeq 2\%$.

Na figura 15, o tempo médio de reconfiguração considerando-se o tempo de propagação e o tempo introduzido pela perda de mensagens, que se mostra similar ao da figura 13, exceto que todas as curvas moveram-se um pouco para cima. Observa-se que o impacto do mecanismo proposto não é grande, pois quando um nó se reconfigura ele contata poucos pares e a mensagem de reconfiguração passa por poucos enlaces. Além disso, a chance de mensagens serem perdidas é pequena e a penalidade no tempo de reconfiguração é menos severa.

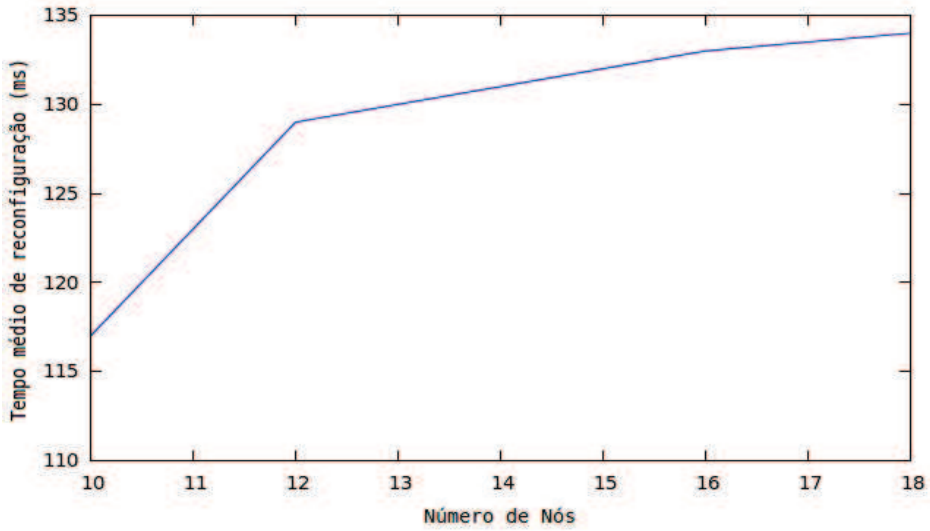

Figura 15: Tempo médio de reconfiguração com perda de mensagens

Tempo de detecção da falha

Outro parâmetro relacionado ao tempo de reconfiguração da rede é o tempo de detecção da falha. Um ponto importante a ser considerado é se o tempo de detecção pode ser um fator dominante na melhoria do tempo de recuperação pelo mecanismo proposto, comparado a métodos reativos. Isso depende do mecanismo de detecção usado e a percentagem das falhas nos nós.

Escolhendo-se cuidadosamente os parâmetros da rede, pode-se tornar o tempo de detecção similar, se não menor que o tempo de reconfiguração de um método reativo, porém mecanismos reativos levam vantagem nesse tipo de detecção por possuírem em sua maioria, métodos colaborativos de detecção. Além disso, a melhoria do mecanismo proposto pode ser

mais significante na redução do tempo de interrupção dos serviços. Especialmente se considerarmos o tempo máximo de propagação no lugar do tempo médio de propagação, o mecanismo proposto pode diminuir substancialmente o tempo de reconfiguração.

Uma ideia a ser aplicada é a de que o tempo de detecção fosse a aplicação do tempo de *timeout* transcorrido no momento da falha multiplicado pelo número de retentativas, mais o tempo de envio das mensagens de reconfiguração dos nós filhos do nó intermediário que falhou até seus novos pais. Nesse caso deve-se considerar o envio de apenas uma mensagem de reconfiguração.

Por exemplo, considerando-se que uma mensagem do tipo KEEPALIVE levaria o mesmo tempo de transmissão do que o tempo de propagação dos dados por um enlace, ou seja, 5ms, pode-se concluir que o tempo de Sleep após o envio de uma mensagem na *thread* 1 do código da seção 3.4.2 ficaria em torno de 10ms. Dessa forma, com o Sleep em 10ms, a vazão máxima de mensagens KEEPALIVE em um segundo ficaria em torno de 10 mensagens, consequentemente o *timeout* da rede seria de 33ms. Ainda, considera-se empiricamente um número limite de retentativas, isto é, o número limite do teste de conectividade entre os nós, fixado em 3 retentativas.

Assim, pode-se calcular o tempo de detecção, ou seja, seria 33ms do *timeout* multiplicado pelo numero de retentativas (3). Consequentemente, o tempo de detecção médio ficaria em torno de 100ms. O gráfico da figura 16 demonstra o tempo médio de reconfiguração levando-se em consideração o tempo de detecção da falha.

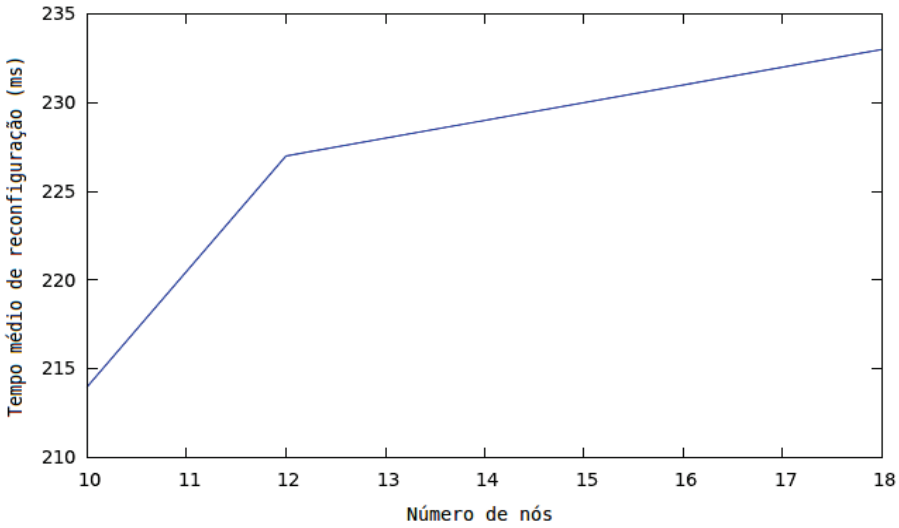

Figura 16: Tempo de reconfiguração com detecção de falha

4.4.2. Qualidade da Árvore Restaurada

A qualidade da árvore restaurada pode ser medida pelo custo da árvore que por sua vez mede os recursos utilizados da árvore. A figura 17 mostra o custo médio da árvore de reconfiguração quando o número de nós na árvore varia de 10 a 18 em comparação a árvore original. Após a reconfiguração de cada nó, calcula-se o custo total da árvore, que é a soma dos pesos (atrasos) de todos os enlaces da camada de aplicação na árvore e divide-se pelo número de enlaces da árvore. Estes custos são medidos sobre todas as árvores reconfiguradas geradas durante os experimentos.

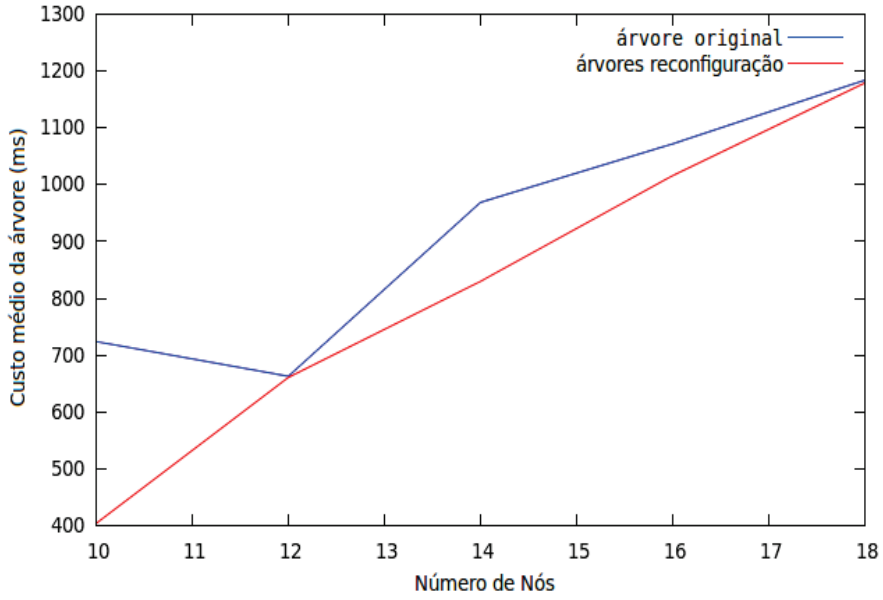

Figura 17: Custo médio variando número de nós

Na figura 17 foi usado o custo médio da árvore como limite inferior. Quando o número de nós na árvore aumenta, o custo total aumenta em todos os casos, com exceção da árvore com 12 nós na árvore original. Analisando com mais cuidado o modelo gerador da árvore original, chegou-se a conclusão de que essa anomalia acontece na realidade na árvore com 10 nós e não na árvore de 12 nós. Isso acontece porque a árvore de 10 nós possui um nó intermediário com custo alto mas que comporta outros nós, devido a sua boa capacidade de *upload*. Pode-se observar também que o mecanismo proativo como o empregado possui um custo da árvore em torno do custo da árvore original, mesmo com um nó a menos na árvore, quando, a princípio o custo deveria aumentar pelo menor número de nós diminuir o *upload* da

rede. Isso aconteceu porque o nó intermediário que falhou era um nó com custo alto, assim em alguns casos o custo torna-se semelhante.

A figura 18 apresenta o gráfico do custo da árvore em função do número de nós para o momento de falha de cada nó intermediário das árvores, incluindo os tempos de reconfiguração para a árvore original e para cada árvore de reconfiguração. Pode-se observar que se mantém uma boa qualidade da árvore nessas situações.

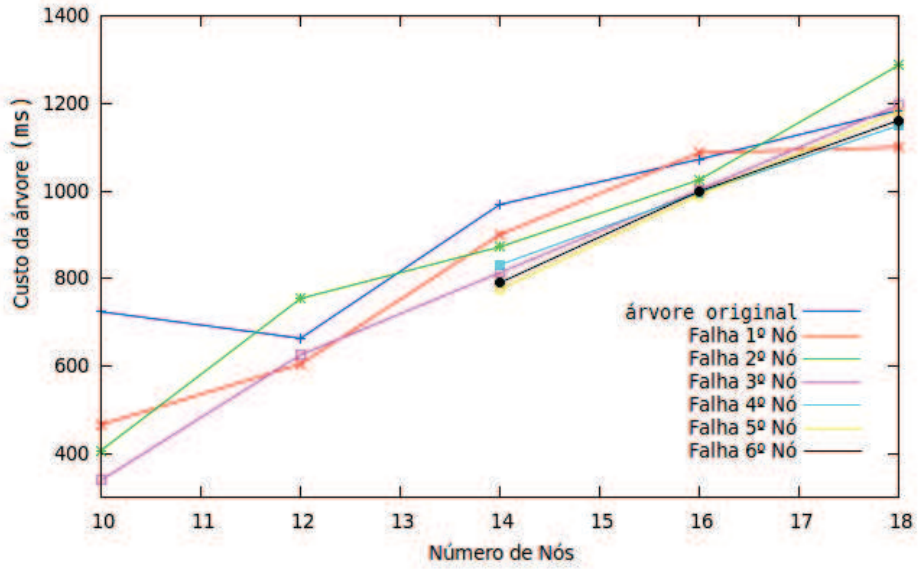

Figura 18: Custo das árvores para cada falha de um nó intermediário

Podemos comparar as figuras 17 e 18 com o trabalho de [Walkowiak e Przewozniczek, 2011] observando o custo das árvores. A árvore original é a árvore do modelo de fluxo, calculada pelo modelo de [Walkowiak e Przewozniczek, 2011]. Dessa forma podemos notar que mesmo com as restrições adicionadas para proteção de nós intermediários e com um nó a menos nas árvores de reconfiguração (o nó que falhou) o custo da árvore permanece semelhante. Isso é importante, pois dependendo qual for o nó intermediário que possa falhar, a tendência é o custo da árvore aumentar devido a menor quantidade de recursos de rede disponíveis.

Outra forma de avaliar a qualidade da árvore é observando o custo médio por enlace das árvores de reconfiguração. Assim para uma rede com 10 nós, o custo médio por enlace para a árvore original foi de 80,44 ms, enquanto o custo médio por enlace para as árvores de reconfiguração foi 50,5 ms. Para 12 nós o custo médio por enlace da árvore original foi de

60,27 ms, enquanto o custo médio para as árvores de reconfiguração foi de 66,1 ms por enlace. Já para 14 nós, o custo médio dos enlaces foi de 74,54 ms, sendo 69,18 ms o custo médio por enlace das árvores de reconfiguração. Para 16 nós, o custo médio por enlace foi de 71,47 ms, com 74,76 ms para o custo médio das árvores de reconfiguração. Por fim, o custo médio por enlace para 18 nós ficou em 69,65 ms, enquanto o custo médio por enlace para as árvores de reconfiguração foi de 73,6ms.

4.4.3. Sobrecarga de Largura de Banda

Analisou-se a sobrecarga na reconfiguração medindo a largura de banda consumida para o propósito de reconfiguração. Isto é, a média do número de *bytes* por segundo enviados na árvore *multicast*, em adição a taxa de transmissão normal de dados.

Para o mecanismo proposto, as mensagens de controle são iniciadas pelos filhos dos nós intermediários que falharam e trocadas com os seus novos pais através das árvores de reconfiguração.

A figura 19 mostra o gráfico da sobrecarga que representa o tráfego médio gerado quando o número de nós varia de 10-18 entre o mecanismo proposto e o mecanismo PRM, descrito na seção 2.4.3. O PRM é um método proativo que trabalha no plano de dados para *multicast* de sobreposição, onde cada nó da rede de sobreposição escolhe uniformemente de forma aleatória um número constante de outros nós da rede de sobreposição e encaminha os dados para cada desses nós com uma baixa probabilidade.

Para simulações do PRM a implementação do mesmo foi desenvolvida e disponibilizada por [Birrer, 2008]. Para os experimentos com o PRM foram utilizados de 10-18 nós distribuídos em *sites* do PlanetLab [PlanetLab, 2012] e com os parâmetros apresentados em [Banerjee et al., 2006], ou seja, o esquema de entrada do PRM é definido com um número randômico de pares e uma probabilidade de encaminhamento. Os parâmetros do PRM foram definidos para um baixo número de *hosts* finais, isto é, todo nó envia os dados para somente um único nó extra com uma probabilidade de 1%, assim como descrito na seção 2.4.3, $n = 1$ e $p = 0{,}01$. O tamanho do pacote de controle foi definido em 64 *bytes* e a taxa de transmissão dos dados é de 256Kb/s. A quantidade de mensagens trocadas entre os nós na árvore de reconfiguração depende do número de nós da árvore. No PRM à medida que o número de nós aumenta, a quantidade de mensagens de controle também aumenta, pois ele necessita descobrir e manter a lista de pares aleatórios de encaminhamento.

Para o mecanismo proposto teorizou-se que é possível estimar a quantidade de mensagens de controle a mais que o protocolo utiliza e consequentemente tem um efeito sobre a sobrecarga. Assim como para o PRM, o tamanho do pacote de controle foi definido em 64 *bytes* e a taxa de transmissão de dados em 256Kb/s.

Como abordado no cálculo de detecção de falha, a sobrecarga de largura de banda do mecanismo proposto para mensagens do tipo KEEPALIVE seria de 10 mensagens por segundo. Assim, incluindo a estimativa de mensagens de controle, teríamos 640 bytes por segundo (64 * 10), ou seja, 5120 b/s para cada nó. Portanto, a sobrecarga gerada pelo envio de mensagens KEEPALIVE para 10, 12, 14, 16 e 18 nós seria respectivamente 51200 b/s (51Kb/s por segundo), 61440 b/s (61Kb/s por segundo), 71680 b/s (72Kb/s por segundo), 81920 b/s (82Kb/s) e 92160 b/s (92Kb/s).

A sobrecarga devido a mensagens de recuperação é desprezível, porque uma única mensagem é enviada pelo nó que detectou e pelos nós pais até atingir a raiz (poucas mensagens, menor que o número de nós). E também, depois uma para cada nó para ativar a nova configuração (cada nó só recebe a mensagem uma vez). Pode-se estimar por 2**n* mensagens trocadas, onde *n* é o número de mensagens. No caso de 10 nós, 20 mensagens, de 64 *bytes*. Ou seja desprezível.

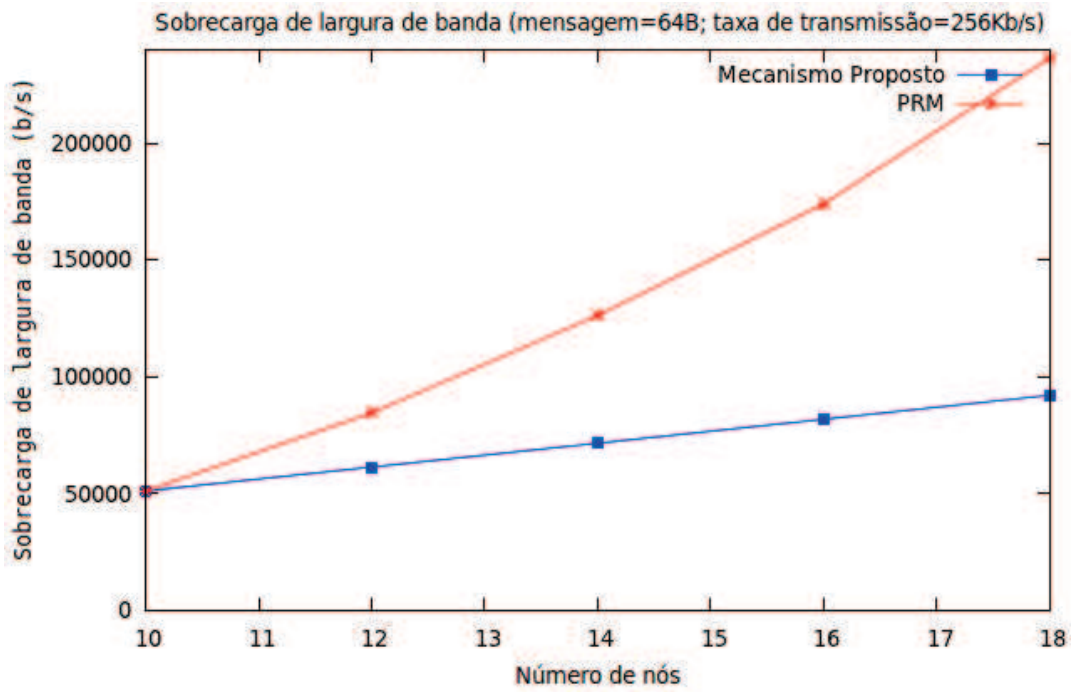

Figura 19: Sobrecarga média na largura de banda com número de nós variáveis

Nota-se, pela figura 19, que no PRM, a sobrecarga de largura de banda aumenta linearmente com o aumento do tamanho do grupo *multicast* pois um número maior de mensagens de controle deve ser encaminhado de acordo com a probabilidade de encaminhamento definida e muitos nós podem ser sobrecarregados, já que o PRM define aleatoriamente os nós que receberão pacotes de dados extras. Já a curva do mecanismo proposto também aumenta linearmente, porém de maneira mais plana. Pode-se notar que inicialmente, para 10 nós a sobrecarga entre os dois mecanismos é semelhante, mas no ponto mais alto, ou seja para 18 nós, a curva do mecanismo proposto ainda é 2,5 vezes menor em relação ao PRM. O aumento da sobrecarga do mecanismo é típico de mecanismos proativos, pois eles só aplicam um mecanismo de recuperação definido antes de ocorrerem falhas. A sobrecarga do mecanismo proposto, dá-se unicamente pela quantidade de mensagens KEEPALIVE necessárias para testar a conectividade entre os nós da rede P2P.

4.5. Considerações Finais

Não é possível comparar o tempo de resposta de um modelo de múltiplas árvores com um modelo de uma única árvore, visto que o tempo de restauração quando se utilizam múltiplas árvores é zero, entretanto alguns pontos puderam ser avaliados nesse estudo.

Um ponto que deve ser ressaltado na análise dos resultados obtidos é que não há sobrecarga gerada em um único nó no momento da reconfiguração, pois os nós já possuem as possíveis árvores para falhas antes das transmissões de fluxos *multicast* ocorrerem. Os tempos de resposta obtidos nos experimentos são da ordem de uma centena de milissegundos, interrompendo os fluxos *multicast* por um tempo aceitável. Porém para aplicações de tempo real, podem ser elevados.

O tempo de propagação das árvores de reconfiguração entre os nós da rede estabiliza em determinado ponto, pois a maior quantidade de nós parceiros aumenta os recursos da rede e assim os nós conseguem mais conexões para os nós filhos, reduzindo o custo da árvore pela redução do número de níveis.

O impacto das perdas de mensagens e do tempo de detecção de falha no tempo de reconfiguração da árvore também não é grande, já que as mensagens de reconfiguração são da ordem de alguns *bytes* e considera-se que apenas uma mensagem pode ser enviada de uma par a outro, e que nesse instante a rede não pode falhar.

A qualidade da árvore também se mostrou adequada, já que a falha de nós

intermediários podem gerar árvores de reconfiguração com custo mais elevado devido a redução de recursos. Entretanto, os custos permaneceram semelhantes, comprovando que a restrições adicionais do modelo não geram sobrecarga na busca por novas árvores de reconfiguração.

Quanto ao PRM, a ideia do mecanismo é que um nó continuamente descobre aleatoriamente membros de sessão e encaminha todos os pacotes de dados recebidos para qualquer um destes membros com uma específica probabilidade. Estes pacotes aleatórios ajudam a detectar e recuperar as partições temporárias da árvore. Sobre condições estáveis, entretanto, tal abordagem introduz pacotes duplicados na ordem de o número de nós que foram sorteados vezes a probabilidade de encaminhamento, que sobrecarrega a largura de banda dos pares de forma linear. Já no mecanismo proposto a sobrecarga da largura de banda acontece devido a quantidade de mensagens de controle necessárias para testar-se a conectividade dos elementos da rede.

Conclusões

Considerações Iniciais

O *multicast* para redes de sobreposição P2P difere do tradicional *multicat* IP na questão de transmissão de dados de importância elevada, devido à instabilidade dos pares de uma rede P2P. Porém, considerando-se que os participantes da rede P2P estão interessados no conteúdo das transmissões enviadas e não deixam a rede *multicast* com frequência, pode-se focar os estudos nas falhas da rede e não nas constantes entradas e saídas dos pares, o que torna o problema de restauração da árvore *multicat* após falhas um pouco diferente.

Neste trabalho, propôs-se um mecanismo proativo *off-line* para recuperação de falhas de nós intermediários na rede P2P *multicast*, com o objetivo de manter a sobrevivência de fluxos *multicast* em redes de sobreposição P2P. No mecanismo proposto todos os pares da rede *multicast* já possuem todas as árvores de reconfiguração para as possíveis falhas dos nós intermediários antes de iniciar as transmissões do nó de origem. As árvores são calculadas *offline* através de um modelo de programação inteira mista, ou seja, o plano de resgate é definido antes das falhas ocorrerem. O mecanismo proposto também possui a vantagem de não precisar de recursos computacionais elevados durante as transmissões dos fluxos *multicast*, já que os *hosts* da rede nada precisam computar no momento de uma falha de nó intermediário, apenas reconfigurar as transmissões de acordo com a nova árvore.

Conclusões

As falhas do nó de origem são protegidas por múltiplas árvores que são calculadas por um modelo de conservação de fluxo proposto na literatura. As árvores de proteção contra as falhas de nós intermediários são calculadas por um algoritmo proposto nesse trabalho, baseado no algoritmo anterior. Ambos foram desenvolvidos utilizando-se a programação inteira mista, e executados através da ferramenta CPLEX. Após a execução do algoritmo gerador das árvores, são criadas tabelas de reconfiguração para cada falha de nó intermediário da rede P2P. Essas tabelas são utilizadas por um protocolo de recuperação do fluxo *multicast*.

O mecanismo proposto (geração de árvores e protocolo de recuperação) foi avaliado

segundo dois aspectos, o tempo de reconfiguração da árvore de *multicast*, e a qualidade da árvore de *multicast*, e foi comparado com um método proativo descrito na literatura. Para isso realizou-se simulações, via uma heurística simples, para se observar o tempo de reconfiguração das árvores e a qualidade da árvore restaurada, durante as trocas de mensagens entre os pares nos momentos em que falhas ocorreram.

Observou-se que o tempo de reconfiguração da árvore é eficiente em relação a modelos reativos. O impacto no tempo de reconfiguração das árvores, gerado pelo tempo de propagação das mensagens, foi medido através do tempo de propagação das mensagens, pelo tempo introduzido pela perda de mensagens e pelo tempo de detecção da falha. Pode-se concluir que o tempo de recuperação não sofre impacto relevante com o acréscimo das mensagens de reconfiguração da rede, pois quando um nó se reconfigura ele contata poucos nós e a mensagem de reconfiguração passa por poucos enlaces. Trata-se de um problema de níveis, já que o custo converge para um valor estável conforme o número de níveis aumenta. Além disso, o tamanho das mensagens de reconfiguração é pequeno, pois as mensagens atuam apenas como avisos sobre a nova árvore a ser utilizada após a ocorrência de uma falha, não carregando nenhuma informação do formato da árvore a ser utilizada. Notou-se que conforme aumenta o número de nós da rede, o tempo de reconfiguração também aumenta, mas ao mesmo tempo, os recursos de *upload* da rede aumentam, e o tempo de reconfiguração tende a estabilizar-se, devido aos nós poderem estabelecer mais conexões para os nós filhos, o que pode decrementar o número de níveis da árvore e em consequência reduzir o atraso.

A qualidade da árvore restaurada também se mostrou adequada, devido as características de geração do modelo de fluxo descrito. Através do custo da árvore, pôde-se notar que mesmo com as restrições adicionais para geração de árvores sem a presença de um nó intermediário que pode falhar, o custo da árvore permanece semelhante ao da árvore inicial sem falhas. E mesmo sem a presença de um nó intermediário que falhou, o custo permaneceu estável, já que dependendo do nó que falhou, o custo da árvore poderia aumentar consideravelmente devido aos poucos recursos que restaram na rede.

Através da comparação do mecanismo proposto com outro mecanismo de recuperação proativo descrito na literatura, foi possível avaliar a sobrecarga na largura de banda gerada no momento de reconfiguração da árvore. Pode-se concluir que o outro mecanismo aumenta a sobrecarga de largura de banda linearmente conforme aumentam o número de nós na rede P2P, devido a sua característica de enviar fluxos adicionais aleatoriamente como mecanismo

preventivo. O incremento observado para o mecanismo proposto também é linear, mas a sua variação, mesmo inicialmente sendo semelhante ao mecanismo concorrente é muito menos acentuada do que a variação do mecanismo concorrente, conforme o numero de nós aumenta.

Em resumo, a partir dos resultados apresentados no capítulo 4, podemos concluir que, através do mecanismo de proteção para árvores *multicast* proposto neste trabalho, é possível melhorar a confiabilidade dos fluxos multicast em redes P2P estáticas com eficiência e escalabilidade com relação à sobrecarga na rede. Entretanto, as limitações de escala introduzidas pelo modelo de programação inteira mista devem ser resolvidas.

Trabalhos Futuros

Como trabalhos futuros podem ser levantados os seguintes pontos:

- Estender o mecanismo para abranger mais de uma falha de nó intermediário simultaneamente.
- Desenvolver uma heurística eficiente para trabalhar com um número maior de pares na rede. É possível supor que com um número elevado de nós que desejem participar da rede, o custo da árvore permaneça constante, garantindo uma boa qualidade da árvore reconfigurada, pois, com um número maior de nós na rede, a falha de poucos nós não acarretariam grandes impactos no aumento do custo.
- Adaptar o mecanismo para múltiplas árvores, porém, tentando realizar a proteção dentro da própria árvore, somente após essa tentativa, receber transmissões de outra árvore.
- Utilizar o novo mecanismo para realizar a proteção de ISP pode ser uma interessante questão a ser tratada. Basta dizer, que nesse caso, sempre deverão existir pelo menos dois ISPs no caso de falha de um terceiro, para que a proteção seja possível.

Referências Bibliográficas

[Allen e Kubart, 2010] J. David Allen; Peter Kubat. **Sreliable Video Broadcasts via Protected Steiner Trees**. IEEE Communications Magazine, Fev. 2012. 70-76. 0163-6804/10.

[Androutsellis-Theotokis e Spinellis, 2004] S. Androutsellis-Theotokis; D. Spineliis. **A survey of peer-to-peer content distribution technologies**. ACM Computing Surveys, v. 36, p. 335–371, 2004.

[Banerjee et al., 2002] S. Banerjee; S. Lee; B. Bhattacharjee; A. Srinivasan. **Scalable Application Layer Multicast**. SIGCOMM'02, Aug. 19-23, Pittsburgh, Pennsylvania, USA. Copyright 2002 ACM 1-58113-570-X/02/0008.

[Banerjee et al., 2006] S. Banerjee; S. Lee; B. Bhattacharjee; A. Srinivasan. **Resilient Multicast Using Overlays.** IEEE/ACM Transactions on Networking, Vol. 14, nº. 2, Apr. 2006, 063-6692, 2006.

[Bienkowski et al., 2005] M. Bienkowski; M. Korzeniowski; F. Heide. **Dynamic load balancing in distributed hash tables**. Peer-to-Peer Systems IV, p. 217–225, 2005.

[Birrer e Bustamante, 2005] S. Birrer; F. E. Bustamante. **Resilient Peer-to-Peer Multicast without the Cost.** Proc. of MMCN, Jan. 2005.

[Birrer e Bustamante, 2006] S. Birrer; F. E. Bustamante. **Resilience in Overlay Multicast Protocols.** Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '06) 0-7695-2573-3/06, 2006.

[Birrer, 2008] S. Birrer. **Addresing the Limitations of Tree-based Approaches to High-**

Bandwith Streaming Multicast. UMI Number: 3303707. Evanston, Illinois. Jun. 2008.

[Castro et al., 2003] M. Castro; P. Druschel; A. Rowstron; A. Kermarrec; A. Singh; A. Nandi. **SplitStream: High-Bandwidth Multicast in Cooperative Environments.** OSP'03, Oct. 19–22, 2003, Bolton Landing, New York, USA. ACM 1-58113-757-5/03/0010.

[Comer, 2006] D. E. Comer; **INTERLIGAÇÃO DE REDES COM TCP/IP, vol. 1 Princípios, protocolos e arquitetura**. Tradução da 5ª Edição Revista e Atualizada. Rio de Janeiro: Elsevier, 2006.

[Coutinho, 2006] G. L. Coutinho. **Delay Tolerant Networks (DTN)**. Universidade Federal do Rio de Janeiro – UFRJ - Escola Politécnica. 2006. Disponível em: http://www.gta.ufrj.br/grad/06_2/gustavo/inicial.htm. Acesso em 11/2011.

[CPLEX, 2009] **IBM ILOG CPLEX v12.1 User's Manual for CPLEX.** Copyright International Business Machines Corporation, 1987 – 2009.

[Fei e Yang, 2005] Z. Fei; M. Yang. **Cooperative failure detection in overlay multicast**. in Proc. IFIP Networking 2005, LNCS 3462, Waterloo, Canada, May 2005, pp. 881–892.

[Fei e Yang, 2007] Z. Fei; M. Yang. **A Proactive Tree Recovery Mechanism for Resilient Overlay Multicast**. IEEE/ACM Transactions on Networking, Vol. 15, nº. 1, Feb. 2007, 1063-6692.

[Ganesan et al., 2004] P. Ganesan; B. Yang; H. Garcia-Molina. **One torus to rule them all: multidimensional queries in p2p systems**. In: Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004, ACM, 2004, p.19–24.

[Hongyun et al., 2008] Y. Hongyun; H. Ruiming; C. Jun; C. Xuhui. **A Review of Resilient Approaches to Peer-to-Peer Overlay Multicast for Media Streaming.** IEEE, 978-1-4244-2108-4/08, 2008.

[Jain, 1991] R. Jain. **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling**. Wiley, 685 p., 1991.

[Jin et al., 2008] H. Jin; Y. Tao; S. Wu; X. Shi. **Scalable dht-based information service for large-scalegrids**. In: CF 08: Proceedings of the 5th conference on Computing frontiers, New York, NY,USA: ACM, 2008, p. 305–312.

[Kurose e Ross, 2010] J. F. Kurose; K. W. Ross. **Redes de Computadores e a Internet – Uma abordagem Top-Down**. 5ª Edição. São Paulo: Pearson, 2010.

[Linnolahti, 2004] J. Linnolahti. **Qos routing for p2p networking**. In: HUT T-110.551 Seminar on Internetworking, 2004. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.7192. Acesso em 10/2012.

[Lua et al., 2005], E. Lua; J. Crowcroft; M. Pias; R. Sharma; S. Lim. **A survey and comparison of peer-to-peer overlay network schemes**. Communications Surveys & Tutorials, IEEE, v. 7, n. 2, p. 72–93, 2005.

[Magharei e Rejaie, 2007] N. Magharei; R. Rejaie, **PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streamin**g. INFOCOM, 2007.

[Milojicic et al., 2002] D. Milojicic; V. Kalogeraki; R. Lukose; K. Nagaraja; J. Pruyne; B. Richard; S. Rollins; Z. Xu. **Peer-to-peer computing**. 2002.

[Padmanabhan et al., 2003] V. N. Padmanabhan; H. J. Wang; P. A. Chou. **Resilient Peer-to-Peer Streaming.** Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03)1092-1648/03, 2003.

[Pioró e Medhi, 2004] M. Pioró; D. Medhi. **Routing, Flow, and Capacity Design in Communication and Computer Networks**. San Francisco, CA. Elsevier, 2004.

[PlanetLab, 2012] PlanetLab Consortium. Disponível em: http://www.planet-lab.org. Acesso em 10/2012.

[Ranjan et al., 2006] R. Ranjan; A. Harwood; R. Buyya. **A study on peer-to-peer based discovery of grid resource information**. University of Melbourne, Australia, Technical Report GRIDS-TR-2006-17, 2006.

[Shyu e Lee, 2009] W. Shyu; T. Lee. **ThrustTorrent: a Sender-selected P2P Protocol**. IEEE, 978-1-4244-5626-0/09, 2009.

[Tran et al., 2004] D. A. Tran; K. A. Hua; T. T. Do. **A Peer-to-Peer Architecture for Media Streaming.** IEEE Journal On Selectede Areas In Communications, Vol. 22, nº. 1, Jan. 2004, 0733-8716/04.

[Walkowiak, 2009] K. Walkowiak. **Survivability of P2P Multicasting**. IEEE 7th International Workshop on the Design of Reliable Communication Networks, 978-1-4244-5048-0/09, 2009.

[Walkowiak e Przewozniczek, 2011] K. Walkowiak; M. Przewozniczek. **Modeling and Optimization of Survivable P2P Multicasting**. Computer Communications 34 (2011), p.1410–1424.

Anexos

Nesta seção encontram-se as figuras e tabelas referentes a seção 3.4.1.

Árvores

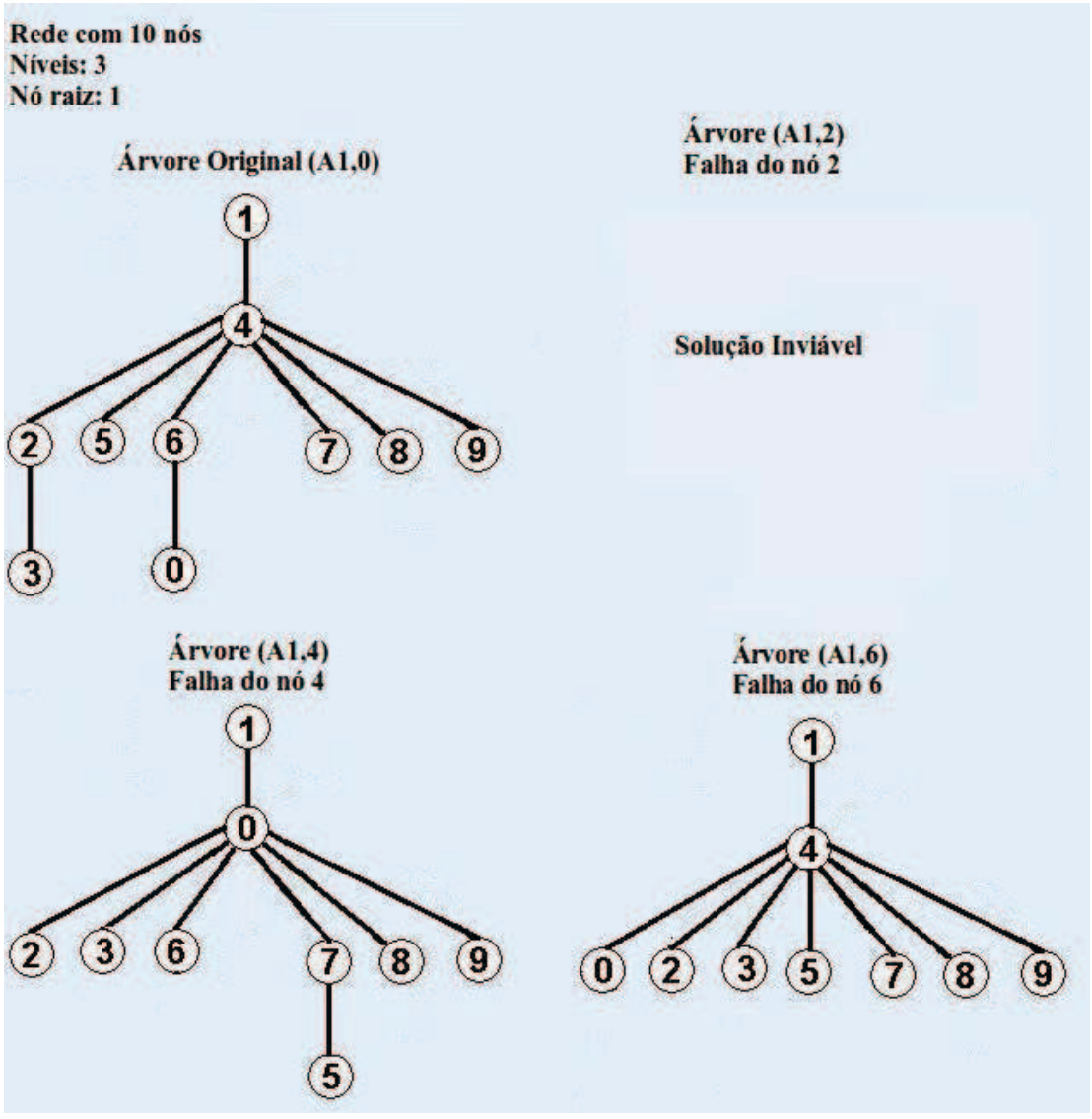

Figura 20: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 1

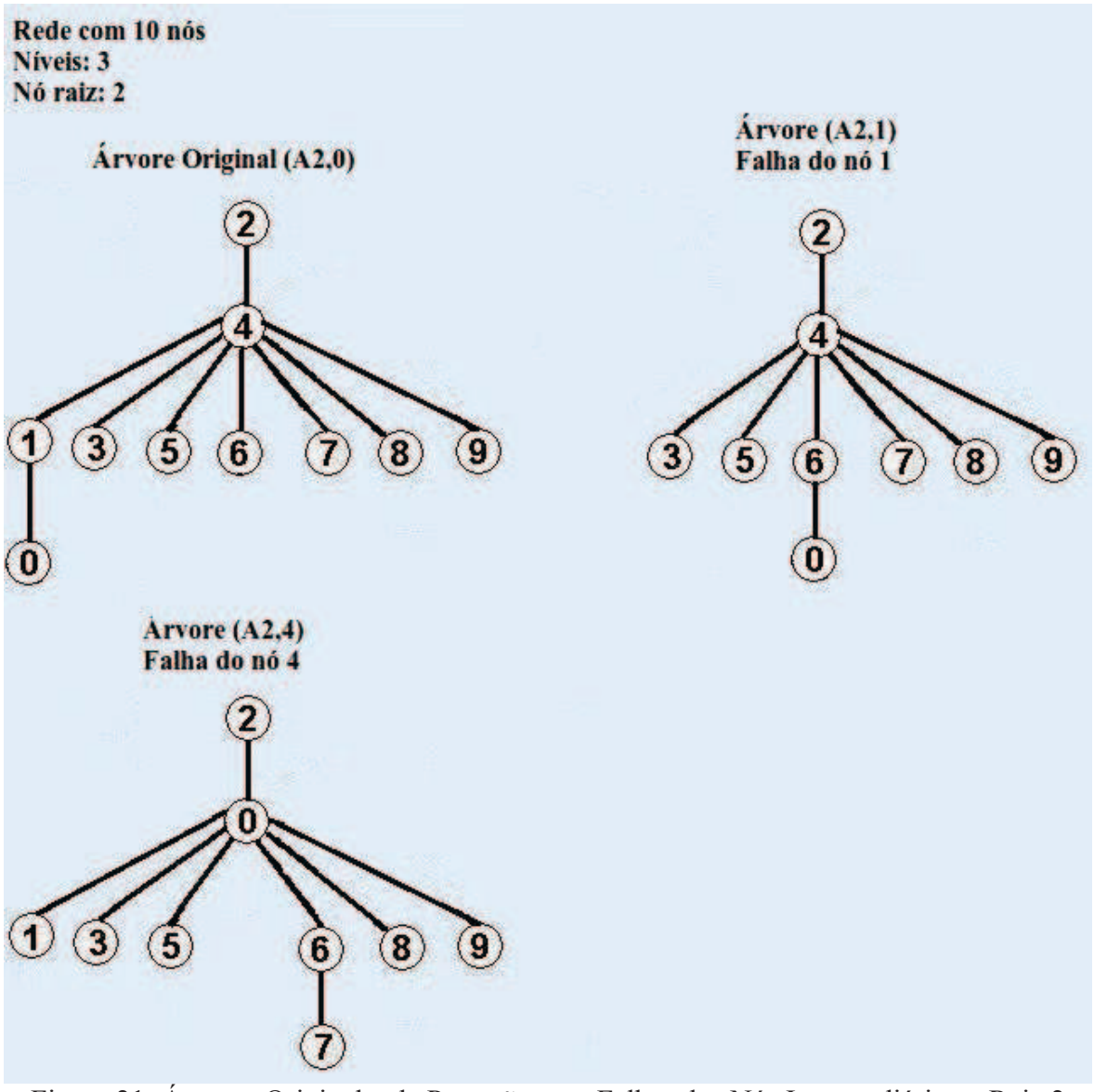

Figura 21: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 2

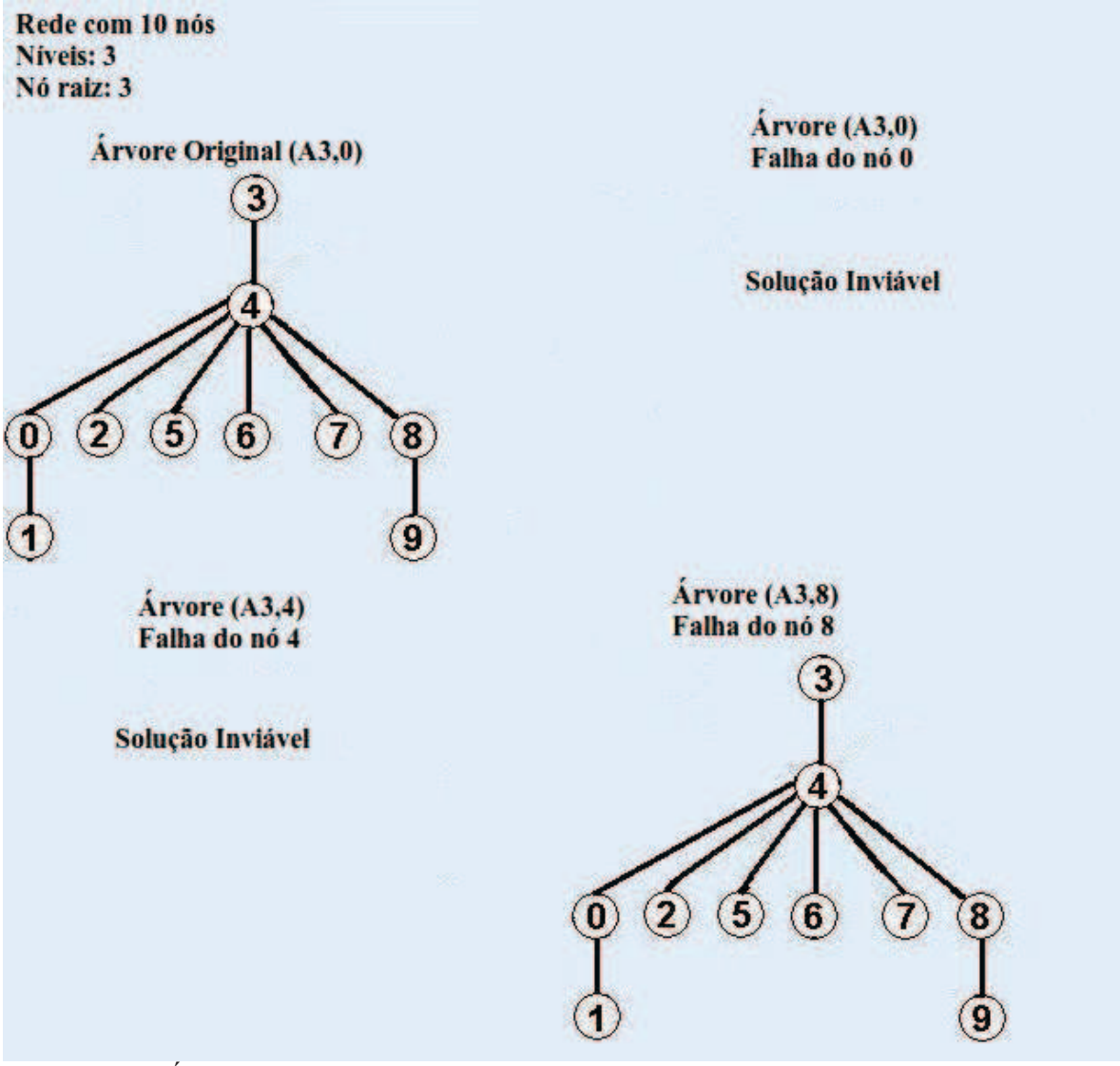

Figura 22: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 3

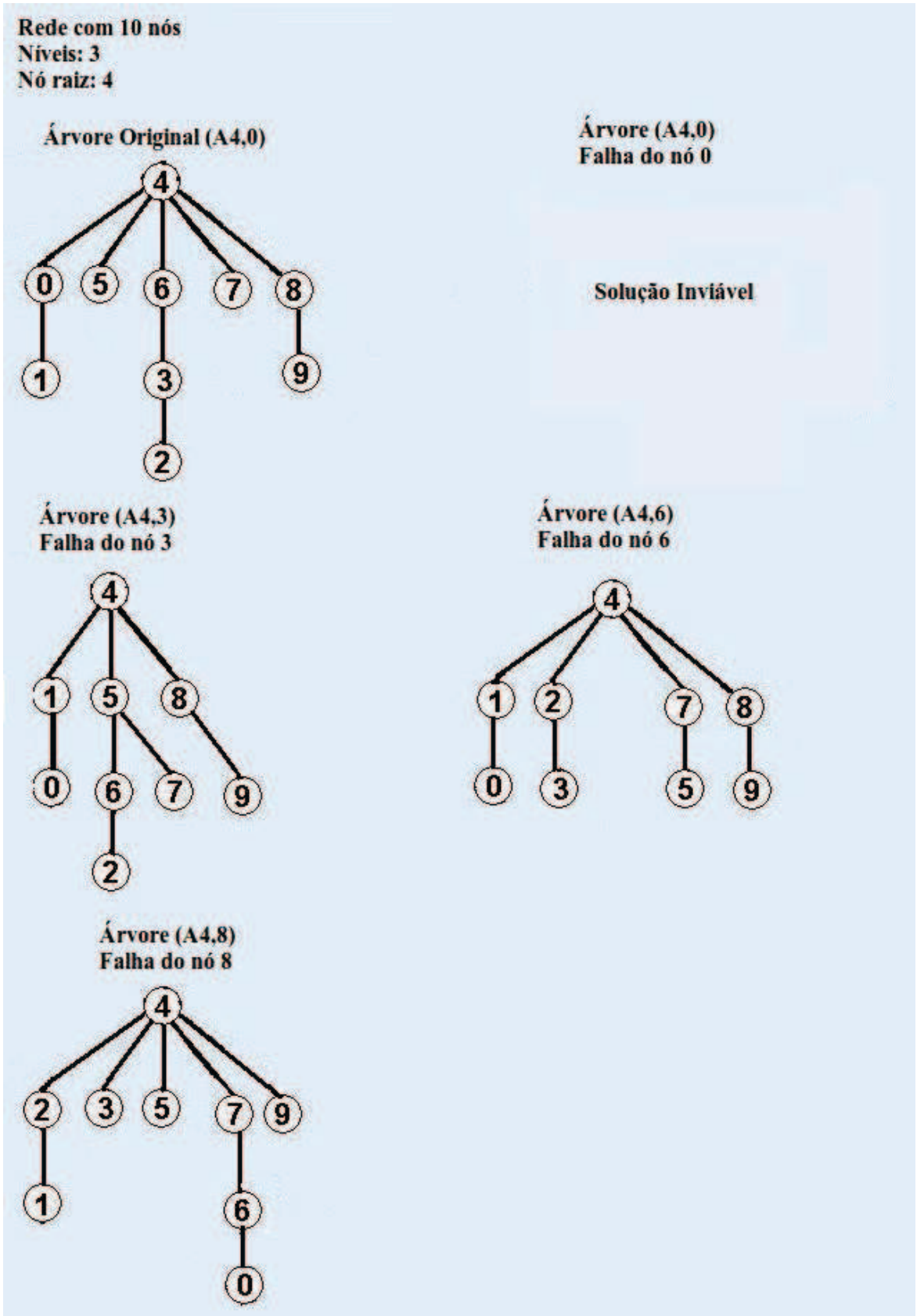

Figura 23: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 4

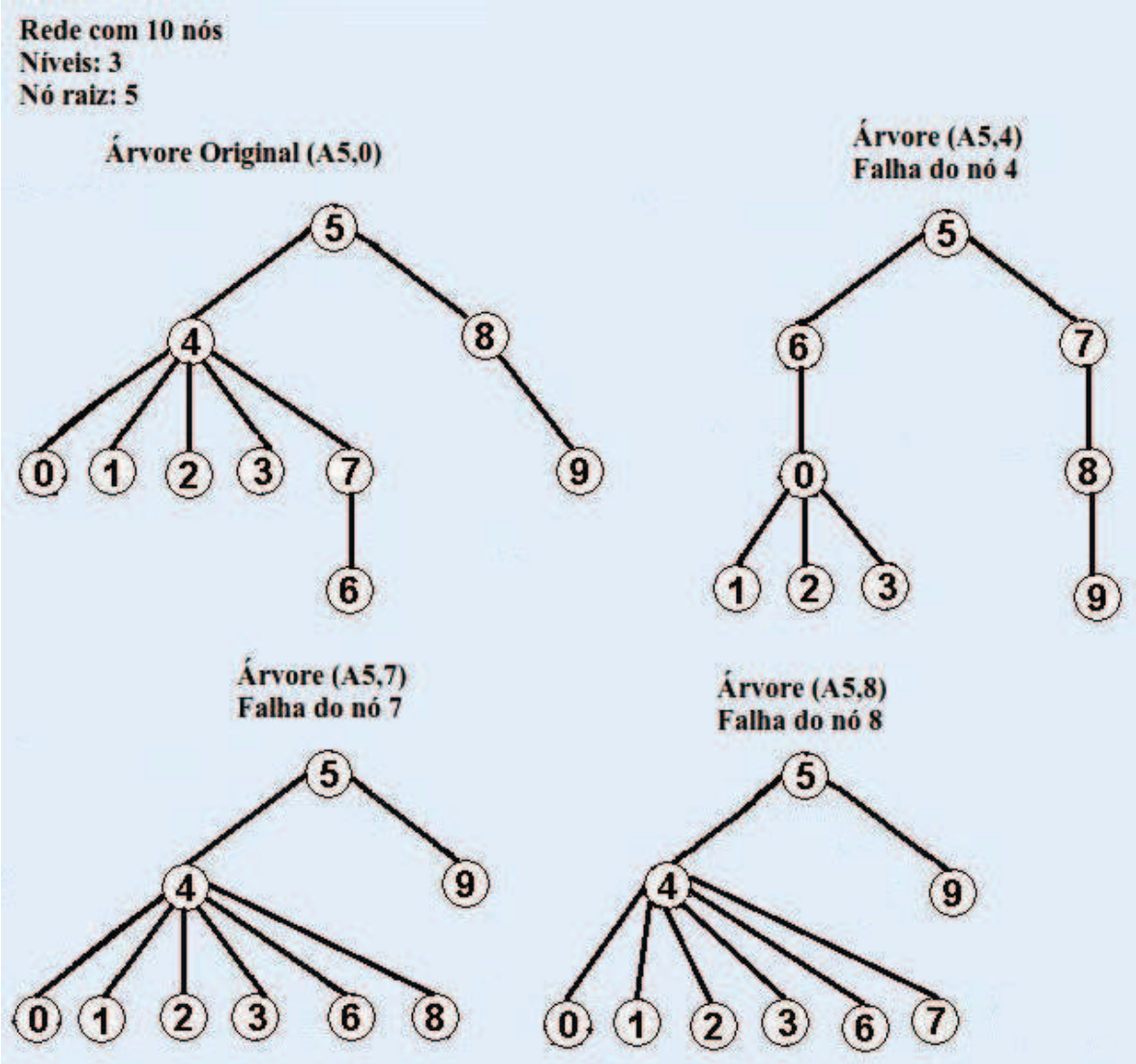

Figura 24: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 5

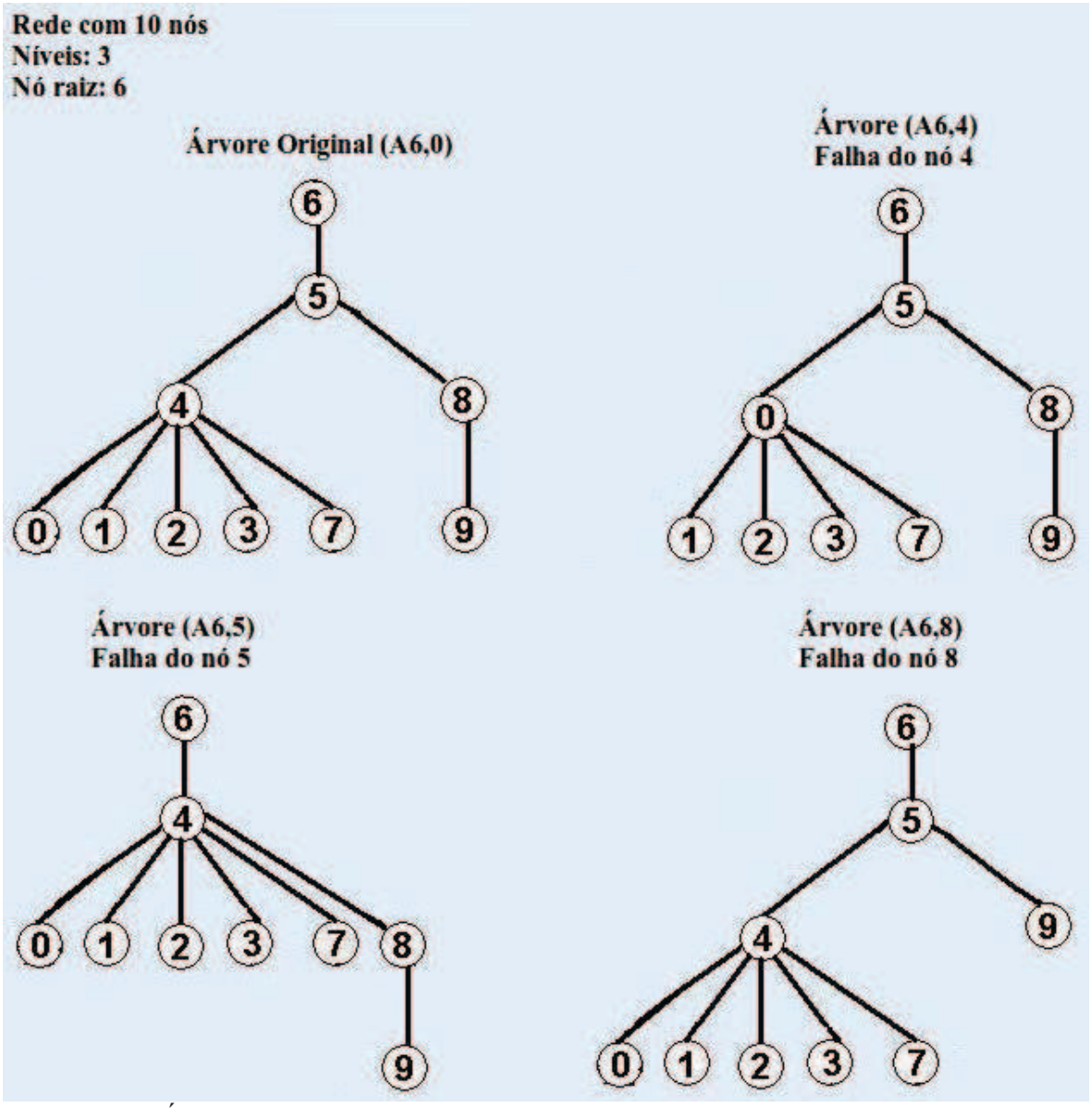

Figura 25: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 6

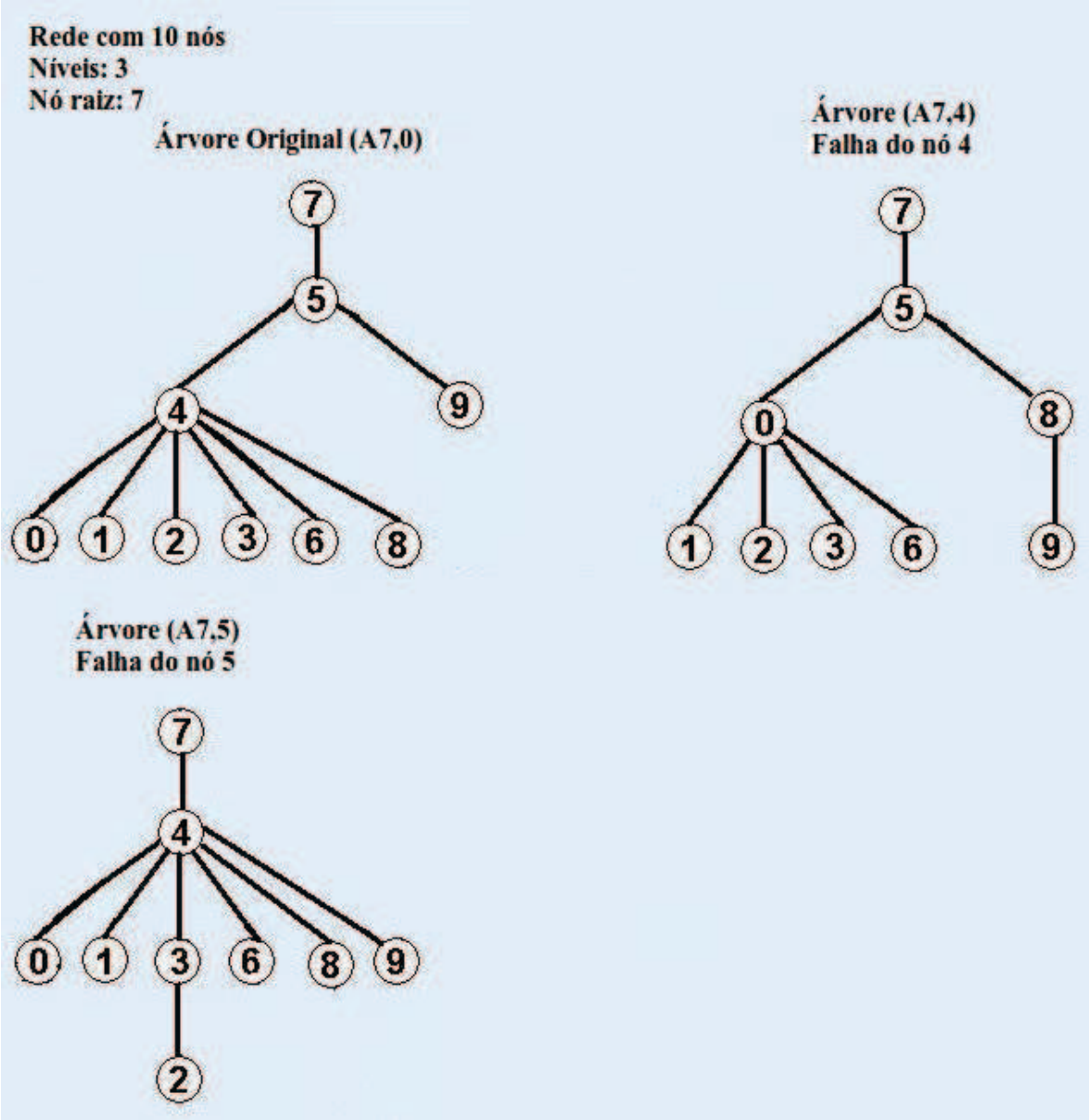

Figura 26: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 7

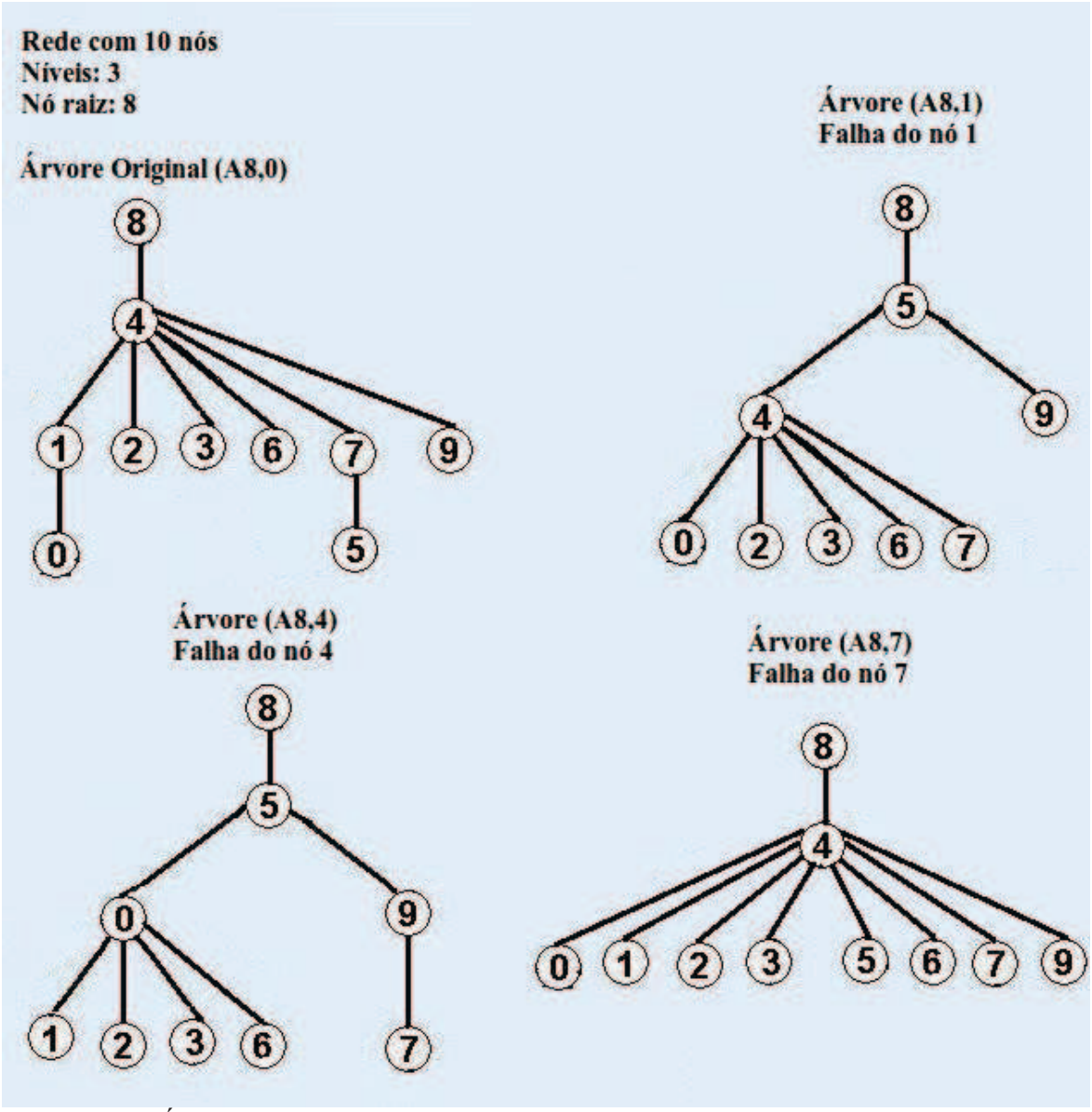

Figura 27: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 8

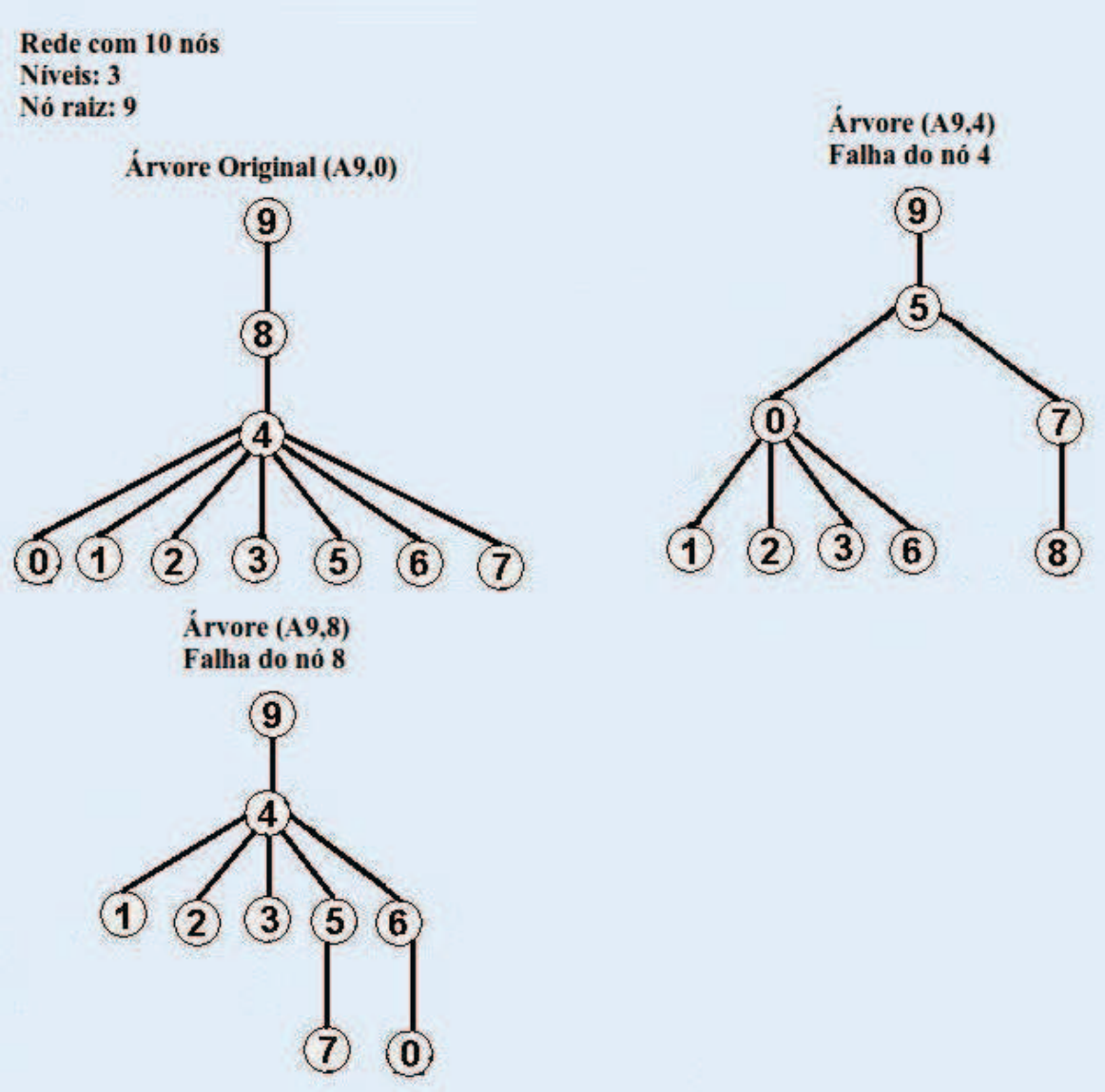

Figura 28: Árvores Original e de Proteção para Falhas dos Nós Intermediários – Raiz 9

Tabelas

Falha / Nó	0	2	4	6
0	-	-	2,3,6,7,8,9	-
1	4	-	0	4
2	3	-	-	-
3	-	-	-	-
4	2,5,6,7,8,9	-	-	0,2,3,5,7,8,9
5	-	-	-	-
6	0	-	-	-
7	-	-	5	-
8	-	-	-	-
9	-	-	-	-

Tabela 12: Tabela de reconfiguração para uma Árvore com Raiz no nó 1

Falha / Nó	0	2	4	6
0	-	-	2,3,6,7,8,9	-
1	4	-	0	4
2	3	-	-	-
3	-	-	-	-
4	2,5,6,7,8,9	-	-	0,2,3,5,7,8,9
5	-	-	-	-
6	0	-	-	-
7	-	-	5	-
8	-	-	-	-
9	-	-	-	-

Tabela 13: Tabela de reconfiguração para uma Árvore com Raiz no nó 2

Falha / Nó	0	0	4	8
0	1	-	-	-
1	-	-	-	0
2	-	-	-	-
3	4	-	-	4
4	0,2,5,6,7,8	-	-	1,2,5,7,9
5	-	-	-	-
6	-	-	-	-
7	-	-	-	6
8	9	-	-	-
9	-	-	-	-

Tabela 14: Tabela de reconfiguração para uma Árvore com Raiz no nó 3

Falha / Nó	0	0	3	6	8
0	1	-	-	-	-
1	-	-	0	0	-
2	-	-	-	3	1
3	2	-	-	-	-
4	0,5,6,7,8	-	1,5,8	1,2,7,9	2,3,5,7,9
5	-	-	6,7	-	-
6	3	-	2	-	0
7	-	-	-	5	6
8	9	-	9	9	-
9	-	-	-	-	-

Tabela 15: Tabela de reconfiguração para uma Árvore com Raiz no nó 4

Falha / Nó	0	4	7	8
0	-	1,2,3	-	-
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	0,1,2,3,7	-	0,1,2,3,6,8	0,1,2,3,6,7
5	4,8	6,7	4,9	4,9
6	-	0	-	-
7	6	8	-	-
8	9	9	-	-
9	-	-	-	-

Tabela 16: Tabela de reconfiguração para uma Árvore com Raiz no nó 5

Falha / Nó	0	4	5	8
0	-	1,2,3,7	-	-
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	0,1,2,3,7	-	0,1,2,3,7,8	0,1,2,3,7
5	4,8	0,8	-	4,9
6	5	5	4	5
7	-	-	-	-
8	9	9	9	-
9	-	-	-	-

Tabela 17: Tabela de reconfiguração para uma Árvore com Raiz no nó 6

Falha / Nó	0	4	5
0	-	1,2,3,6	-
1	-	-	-
2	2	-	-
3	-	-	2
4	0,1,2,3,6,8	-	0,1,3,6,8,9
5	4,9	0,8	-
6	-	-	-
7	5	5	4
8	-	9	-
9	-	-	-

Tabela 18: Tabela de reconfiguração para uma Árvore com Raiz no nó 7

Falha / Nó	0	1	4	7
0	-	-	1,2,3,6	-
1	0	-	-	-
2	-	-	-	-
3	-	-	-	-
4	1,2,3,6,7,9	0,2,3,6,7	-	0,1,3,5,6,9
5	-	4,9	0,9	-
6	-	-	-	2
7	5	5	-	-
8	4	5	5	4
9	-	-	7	-

Tabela 19: Tabela de reconfiguração para uma Árvore com Raiz no nó 8

Falha / Nó	0	4	8
0	-	1,2,3,6	-
1	-	-	-
2	-	-	-
3	-	-	-
4	0,1,2,3,5,6,7	-	1,2,3,5,6
5	8	0,7	7
6	-	-	0
7	5	8	-
8	4	-	-
9	8	5	4

Tabela 20: Tabela de reconfiguração para uma Árvore com Raiz no nó 9