Roteamento Tolerante a Falhas Baseado em Caminhos Robustos

Jonatan Schroeder, Elias P. Duarte Jr.

Universidade Federal do Paraná – Departamento de Informática
Caixa Postal 19081 CEP 81531-990 – Curitiba - PR - Brasil

{jonatan,elias}@inf.ufpr.br

Abstract. *Routing protocols present a latency, i.e. a time interval spent to update all routing tables, after the topology changes. During the convergence latency interval several packets and connections may be lost. This work proposes a new strategy for a dynamic routing, in which intermediate routers, that may hold more recent information about the current state of the topology, select an alternative path when the regular one is not working. This routing strategy is based on strongly connected paths, chosen using connectivity criteria. Experimental results, based on simulation, comparing this approach and one using Dijkstra's algorithm, on a faulty environment, show that path changes made on strongly connected paths are 40% shorter than those found by Dijkstra's algorithm.*

Resumo. *Protocolos de roteamento apresentam uma latência, isto é, um intervalo de tempo para atualizar suas tabelas de rotas em toda a rede, após uma alteração na topologia. Esta latência de convergência pode gerar potenciais perdas de pacotes ou conexões na rede. Este trabalho propõe uma estratégia para roteamento dinâmico, permitindo que roteadores intermediários que possuam informações mais recentes de alterações na topologia interfiram no caminho utilizado. Este roteamento é baseado em caminhos robustos, escolhidos utilizando critérios de conectividade, que valorizam a redundância de caminhos. Uma comparação de resultados experimentais obtidos através de simulação da abordagem baseada nestes caminhos com uma seleção de caminhos por Dijkstra, em um ambiente sujeito a falhas, demonstra que as alterações de caminho feitas em caminhos robustos são 40% menores que as encontradas por Dijkstra.*

1. Introdução

Os protocolos de roteamento na Internet necessitam de um certo tempo para atualizar as tabelas de rotas de todos os roteadores, de modo a refletir mudanças de estado que tenham ocorrido na topologia da rede. Este período de tempo é conhecido como *latência de convergência* do protocolo [10]. A latência média do protocolo BGP (*Border Gateway Protocol*), um dos mais utilizados na Internet atualmente [12], é de 3 minutos, sendo que já foram observados períodos de latência de até 15 minutos [10].

Durante o período de latência do protocolo, alguns roteadores possuem informações inconsistentes nas suas tabelas de rotas, gerando potenciais perdas de pacotes e de conexões entre as aplicações que se comunicam pela rede.

Diversas abordagens têm sido propostas para resolução deste problema. Chandrashekar et.al. [1] propõem um mecanismo para análise das dependências de caminhos, de forma a reduzir a exploração de novos caminhos e, por conseqüência, reduzir a latência

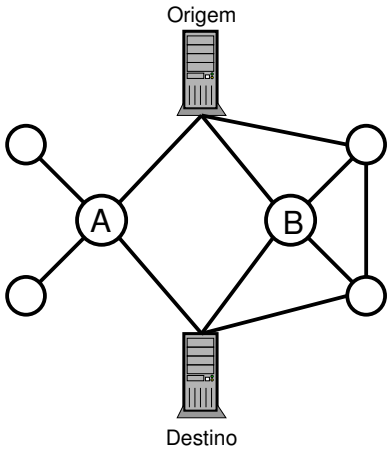


Figura 1: Exemplo de vantagem do uso da conectividade.

de convergência. Pei et.al. [11] propõem uma série de asserções a serem aplicadas nas redes, com o propósito de comparar caminhos similares e eliminar caminhos inviáveis. Wang et. al. [16] citam uma estratégia baseada em um mecanismo escalável, denominado FRTR (*Fast Routing Table Recovery*), para detecção e recuperação de inconsistências nas tabelas de roteamento do protocolo BGP.

Grande parte destas abordagens busca reduzir a latência de convergência, sem, no entanto, propor soluções a serem adotadas durante esta convergência. O que se propõe é reduzir os impactos da alta latência de convergência pela própria redução desta latência.

Outra abordagem para contornar esta situação [3, 13] se baseia na identificação de nodos altamente conexos na rede, encontrando-se caminhos que passem por estes nodos. Propõe-se que, em caso de ocorrência de falhas na conexão, se encontre um desvio que passa pelo nodo altamente conexo de forma a chegar ao destino. Esta abordagem pode ser visualizada na figura 1. Nesta figura, pode-se utilizar dois caminhos, passando um pelo nodo A e outro pelo nodo B. O caminho que passa por B pode ser considerado melhor, visto que há uma possibilidade maior da utilização de desvios, caso o caminho principal falhe.

Este trabalho propõe uma abordagem de roteamento tolerante a falhas e dinâmico. O roteamento é dinâmico no sentido que roteadores intermediários do caminho escolhido podem alterar este caminho utilizando informações sobre seu estado. Por exemplo, se um determinado roteador identifica que, no caminho escolhido para a comunicação, há um enlace falho, ele evita este caminho, e as conseqüentes perdas de pacote. Isto ocorre porque, após uma falha ou recuperação de um enlace, os roteadores mais próximos a este enlace recebem a informação deste evento antes dos demais pontos da rede. Este conceito pode se aplicar também a informações de congestionamento. O caminho alternativo encontrado após identificação da falha é chamado *desvio*.

Para que a possibilidade de determinar desvios sem falha e curtos seja maior, é necessário escolher caminhos com grande redundância, que chamaremos de caminhos robustos. Este trabalho propõe uma nova abordagem para a identificação de caminhos robustos, baseada em critérios de conectividade. Nesta abordagem, a conectividade média de *todos* os nodos do caminho é considerada na escolha dos caminhos. Em caminhos com maior conectividade, a possibilidade de encontrar desvios aumenta. Isto se deve à redundância de caminhos introduzida naturalmente pela alta conectividade.

Um dos principais critérios de conectividade utilizados neste trabalho é o número de conectividade, denotado por $\#C(v)$ e originalmente introduzido em [3]. Este número

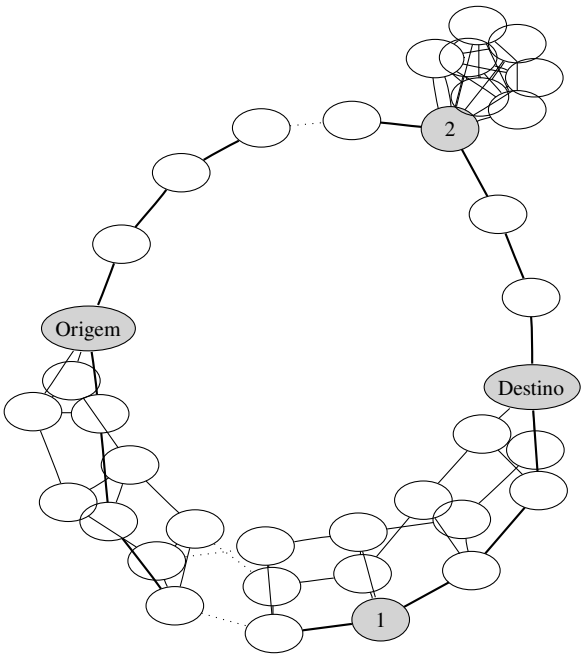


Figura 2: Exemplo de vantagem do uso da conectividade média.

é associado a um nodo específico, e corresponde à quantidade de caminhos disjuntos existentes entre o nodo em questão e seus vizinhos. Prova-se que o número de conectividade de um nodo está diretamente ligado à quantidade de caminhos alternativos que passam por esse nodo.

A figura 2 ilustra a vantagem do uso da conectividade média em relação à seleção de componentes de alta conectividade. No caminho que passa pelo nodo "1", todos os nodos possuem uma conectividade relativamente alta, isto é, $\#C(v) = 4$, em média. No outro caminho, o nodo "2" possui uma conectividade extremamente alta, com $\#C(v) = 8$, mas os demais nodos do caminho possuem uma conectividade baixa, $\#C(v) = 2$, reduzindo-se a possibilidade de desvios, dada uma falha no caminho original.

Deve ser destacado que este trabalho, em comparação com [3], apresenta duas contribuições principais: enquanto que em [3] a escolha do caminho é feita considerando apenas um nodo altamente conexo, este trabalho apresenta uma seleção baseado na conectividade média de todos os nodos do caminho; além disso, este trabalho apresenta uma proposta de roteamento dinâmico, no qual o caminho é alterado à medida em que informações mais atuais da topologia são descobertas.

É apresentada também uma avaliação baseada na comparação quantitativa entre o roteamento com a escolha do menor caminho pelo algoritmo de Dijkstra [2], e a nova abordagem apresentada. Esta comparação tem como objetivo avaliar a robustez e o custo do caminho, no que se refere à possibilidade de utilização de desvios para chegar ao destino, em caso de ocorrência de falhas no caminho original.

Este artigo está organizado como segue. Na seção 2 são descritos os principais critérios de conectividade utilizados neste trabalho. Na seção 3 é descrito o algoritmo utilizado para a seleção de caminhos robustos. Na seqüência, a seção 4 descreve a abordagem de roteamento tolerante a falhas utilizada, bem como o re-roteamento adotado em caso de identificação de falhas. A seção 5 a avaliação da confiabilidade dos caminhos encontrados, bem como apresenta os resultados experimentais obtidos. Por fim, a seção 6 apresenta as conclusões.

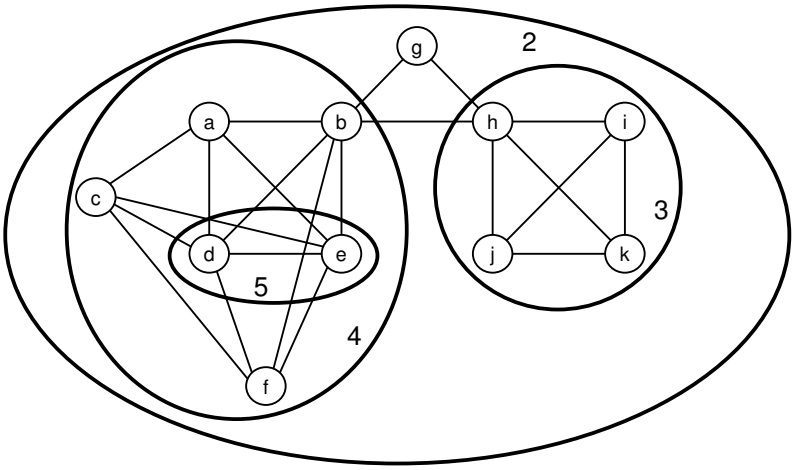


Figura 3: Um exemplo de cálculo do $\#C(v)$ e do $MCC(v)$.

2. Critérios de Conectividade

A seguir, são apresentados os critérios de conectividade utilizados neste trabalho, introduzidos originalmente em [3].

Em todas as definições, a topologia física (ou lógica) da rede é representada através de um grafo não-direcionado. Um grafo não direcionado é um par $G = (V, E)$, onde V é um conjunto (finito) de elementos denominados nodos (ou vértices) e E é um conjunto (finito) de elementos denominados arestas. Cada aresta é um conjunto $e = \{u, v\}$, onde $u, v \in V$.

Definição 2.1: *Dados dois vértices $u, v \in V$. Um corte entre esses dois vértices é um conjunto de arestas tais que, removendo-se as arestas do grafo original, os vértices u e v ficam desconexos no grafo G. Um corte mínimo entre dois vértices é um corte entre esses vértices com cardinalidade (ou seja, número de arestas) mínima.*

Definição 2.2: *Dado um vértice $v \in V$. Dado um subconjunto $V' \subseteq V$, com $v \in V'$, tal que a cardinalidade de qualquer corte mínimo em G que desconecta dois vértices em V' é máxima. O número de conectividade $\#C(v)$ é definido como a cardinalidade do corte mínimo entre qualquer par de vértices do subconjunto V'.*

Definição 2.3: *Dado um vértice $v \in V$. O componente de conectividade máxima $MCC(v)$ é o maior subconjunto de V contendo v tal que todo corte entre qualquer par de vértices do subconjunto possui cardinalidade maior ou igual a $\#C(v)$.*

A figura 3 ilustra um grafo juntamente com os valores dos números de conectividade e dos componentes de conectividade máxima desses nodos. Os círculos representam o $MCC(v)$ dos nodos em questão, juntamente com o número de conectividade associado. O número de conectividade de cada nodo corresponde ao número associado ao círculo mais restrito que corresponde ao mesmo. Por exemplo, considere o vértice a. O corte mínimo entre este vértice e o vértice b é 4, visto que é necessário remover 4 arestas para que ambos fiquem desconexos entre si. O mesmo ocorre com os vértices c, d, e e f. Desta forma, temos $\#C(a) = 4$ e $MCC(a) = \{a, b, c, d, e, f\}$.

2.1. Algoritmos de Obtenção dos Critérios de Conectividade

Uma propriedade importante do número de conectividade é a de que, dado um nodo $v \in V$, o valor do $\#C(v)$ é igual à cardinalidade máxima entre todos os cortes mínimos separando v de qualquer outro nodo de G. A prova pode ser verificada em [3].

Tendo em vista esta propriedade, utiliza-se a chamada árvore de corte, também conhecida por árvore de Gomory-Hu [5], para o cálculo dos critérios de conectividade.

Esta árvore tem algumas propriedades que auxiliam em cálculos relacionados ao corte mínimo em grafos não-direcionados. Diversos trabalhos, dentre os quais o trabalho de Gusfield [6], descrevem algoritmos para obtenção de uma árvore de corte.

Uma *árvore de corte* de um grafo $G = (V, E)$ é definida como uma árvore $T = (V_T, E_T, w)$, onde V_T é o conjunto de nodos da árvore, E_T é o conjunto de arestas da árvore, e $w : W \rightarrow \Re$ é uma função que determina o peso de uma aresta. Além disso, ela deve possuir as seguintes propriedades:

1. $V_T = V$, ou seja, os nodos da árvore de corte correspondem aos nodos do grafo G;
2. A cardinalidade de um corte mínimo entre dois vértices de G é dada pelo valor da aresta de menor peso pertencente ao caminho único que liga os dois vértices correspondentes em T;
3. Um corte mínimo entre dois vértices de G é obtido removendo a aresta mencionada acima e considerando os dois conjuntos de vértices induzidos pelas duas sub-árvores formadas pela remoção de tal aresta em T.

A partir da árvore de corte, é possível calcular os critérios de conectividade citados no início desta seção. Em [3] prova-se que, para um grafo G, uma árvore de corte T de G e um vértice v, o valor do $\#C(v)$ é igual ao valor da aresta de maior peso incidente a v em T. No mesmo trabalho também se prova que os vértices alcançáveis em T a partir de v, utilizando apenas arestas com peso igual ou superior ao valor do $\#C(v)$, pertencem ao conjunto $MCC(v)$.

Para obtenção da árvore de corte, foi utilizado o algoritmo de Gusfield [6]. Neste algoritmo, para obtenção do corte mínimo, foi utilizada uma simplificação do algoritmo de Ford e Fulkerson [4] para cálculo de fluxo máximo/corte mínimo. Esta simplificação é descrita em [14].

3. Cálculo do Caminho

Este trabalho propõe uma nova abordagem para roteamento baseada em critérios de conectividade. Para a seleção de um caminho, é considerada a conectividade média de todos os seus nodos; outros critérios como o tamanho do caminho também são considerados. O algoritmo para a avaliação e a seleção de caminhos é descrito a seguir.

3.1. Seleção dos Caminhos

O nosso objetivo, nesta seção, é o de identificar os caminhos mais robustos entre dois pontos de uma rede. Considerando que a rede em que o cálculo será efetuado está modelada em um grafo não-direcionado G, vamos selecionar os melhores caminhos entre dois vértices: uma origem s, e um destino t.

Inicialmente é realizada uma busca em profundidade no grafo G, partindo da origem s. Esta busca retorna todos os caminhos p sem ciclos entre s e t. Após esta busca, para cada um dos caminhos encontrados é atribuído um valor $\Gamma(p)$, que é obtido considerando os critérios de conectividade dos nodos do caminho, e o tamanho do caminho em termos de seu número de arestas. O valor final associado a cada caminho é dado pela seguinte equação:

$$\Gamma(p) = \omega_1.E\big[\#C(v)\big] + \omega_2.E\big[\,|MCC(v)|\,\big] + \omega_3.E\big[\,|p_i|\,\big] \qquad (1)$$

Nesta equação, são utilizados os seguintes componentes:

- $E\big[\#C(v)\big]$: média do número de conectividade ($\#C(v)$) de todos os nodos intermediários do caminho p (exceto s e t);
- $E\big[\,|MCC(v)|\,\big]$: média do tamanho do componente de conectividade máxima ($MCC(v)$) de todos os nodos intermediários do caminho p (exceto s e t);
- $E\big[\,|p|\,\big]$: tamanho do caminho p, em termos do número de arestas atravessadas pelo caminho.

Cada um dos componentes deste cálculo é multiplicado por um peso ω_i, de forma a possibilitar dar mais ênfase a um aspecto ou outro. Diversas combinações de pesos são possíveis, tendo em vista requisitos específicos de cada sistema. Para o tamanho do caminho, recomenda-se o uso de pesos negativos ($\omega_3 < 0$), visto que, quanto menor o caminho, mais adequado ele é. Para a conectividade média, é recomendado que o peso seja positivo ($\omega_1 > 0$), uma vez que o objetivo deste algoritmo é encontrar caminhos robustos, e a conectividade é fator determinante da robustez do caminho. O peso do tamanho do componente de conectividade máxima também deverá ser positivo ou nulo ($\omega_2 \geq 0$), pois possui relação direta com a robustez, porém recomenda-se que seja relativamente menor que o peso da conectividade, uma vez que nodos com conectividade baixa normalmente pertencem a um componente de conectividade máxima maior, afetando o resultado esperado da robustez. Este fato é ilustrado na figura 3. Nesta figura, o nodo g possui baixa conectividade, mas o $MCC(g)$ corresponde ao grafo por inteiro. Já o nodo d possui alta conectividade, porém o $MCC(d)$ contém apenas os nodos d e e.

Há também a possibilidade de trabalhar com uma variabilidade no peso específico para o cálculo da média do número de conectividade. Desta forma, em lugar de utilizar uma média simples, utiliza-se a seguinte fórmula:

$$E\big[\#C(v)\big] = \frac{\displaystyle\sum_{k=1}^{|p|-1} \alpha^{k-1} \#C(n_k)}{\displaystyle\sum_{k=1}^{|p|-1} \alpha^{k-1}} \tag{2}$$

Nesta equação, n_k são os nodos intermediários do caminho p, e α é o coeficiente de variação da média. Desta forma, se $0 < \alpha < 1$, teremos uma conectividade média com ênfase na conectividade dos nodos mais próximos à origem. Se $\alpha > 1$, a média da conectividade será mais afetada pelos nodos mais próximos ao destino. Esta situação, a princípio, será mais favorável, uma vez que a necessidade de que desvios sejam encontrados no final do caminho é maior que no início do caminho. Considerando $\alpha = 1$, a média da conectividade será linear, com todos os nodos sendo considerados igualmente.

O mesmo tratamento pode ser feito com o tamanho do componente de conectividade máxima, de acordo com a seguinte equação:

$$E\big[\,|MCC(v)|\,\big] = \frac{\displaystyle\sum_{k=1}^{|p|-1} \beta^{k-1}\,|MCC(n_k)|}{\displaystyle\sum_{k=1}^{|p|-1} \beta^{k-1}} \tag{3}$$

Da mesma forma que o resultado da média da conectividade é afetado pelo valor de α, a média do tamanho do componente de conectividade máxima é afetado pelo valor de β. Valores de β menores ou maiores que 1 darão mais ênfase aos nodos mais próximos à origem ou ao destino, respectivamente. Para β igual a 1, a média será linear.

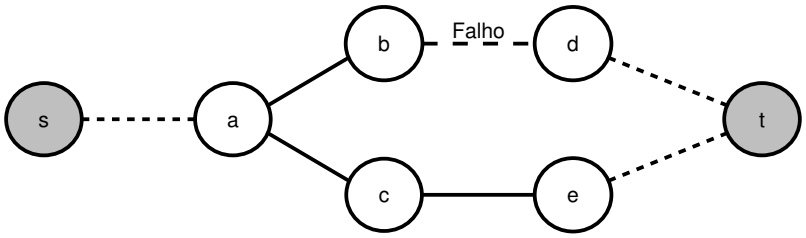


Figura 4: Exemplo da utilização de um desvio.

Após a atribuição do valor descrito acima, os caminhos são ordenados pelo valor $\Gamma(p)$ de forma decrescente. Assim, o caminho com maior valor é considerado o mais robusto, e deve ser a primeira escolha para o roteamento. Em caso de empate, foi considerado critério de desempate o tamanho do caminho. Caso dois caminhos possuam mesma avaliação e mesmo tamanho, um dos dois caminhos é selecionado aleatoriamente.

No caso em que há uma aresta ligando diretamente a origem e o destino, não há nodos intermediários para o cálculo da conectividade média. Podemos assumir, neste caso, que a aresta será o caminho mais apropriado. Desta forma, caso haja um caminho direto, com apenas uma aresta, entre a origem e o destino, este é considerado o primeiro caminho, sem que seja necessária avaliação do mesmo. Vamos considerar que, neste caso, $\Gamma(p) = +\infty$.

4. Uma Proposta de Roteamento Tolerante a Falhas

O principal objetivo do roteamento confiável proposto neste trabalho é a escolha de caminhos robustos, em que, quando uma falha é identificada, um desvio possa ser encontrado, ou seja, um novo caminho que chegue ao destino a partir da origem na presença da falha. Nesta seção descreve-se uma proposta de roteamento baseado nesta identificação de desvios.

O roteamento proposto neste trabalho se baseia na permissão para que roteadores intermediários de um caminho escolhido interfiram no restante do caminho. Esta permissão é concedida porque as informações de topologia normalmente são atualizadas mais rapidamente em nodos mais próximos aos eventos que geram sua alteração. Por exemplo, a ocorrência de uma falha em um enlace é primeiramente repassada para os nodos vizinhos, que por sua vez passam para seus vizinhos e assim sucessivamente. Tendo em vista este fato, roteadores que tenham o conhecimento de uma falha na seqüência do caminho proposto podem alterar o caminho para outro caminho que não esteja falho. Este novo caminho é denominado desvio.

A figura 4 ilustra uma situação em que a utilização de desvios é demonstrada. Supondo uma conexão entre os nodos s e t, em que s deseja enviar um pacote de dados para t. Vamos supor que o caminho escolhido seja o caminho que passa pelos nodos a, b e d. O procedimento para escolha do caminho está sendo ignorado, por enquanto.

Considere a situação em que a aresta que liga os nodos b e d falha. Se não for utilizada uma abordagem tolerante a falhas, o pacote enviado pelo nodo s será simplesmente perdido. Considerando a comunicação confiável, quando o nodo s identificar esta falha, seja por *timeout*, seja por informação por parte dos outros nodos, ele enviará um novo pacote. Este pacote possivelmente seguirá o mesmo caminho utilizado pelo pacote anterior, visto que o nodo s pode não ter sido informado da falha na aresta em questão. Este procedimento poderá se repetir até que s possua a informação mais recente da topologia, o que pode demorar até 15 minutos na Internet com o protocolo BGP, conforme já mencionado em seções anteriores.

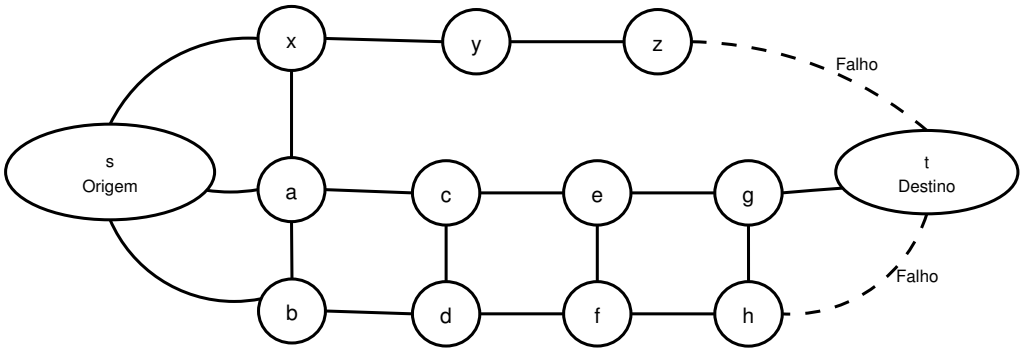


Figura 5: Exemplo de grafo com falhas.

Pela abordagem tolerante a falhas proposta neste trabalho, ao chegar no nodo b, a falha será identificada, e o nodo b fica, então, encarregado de escolher um novo caminho chegando em t, já desconsiderando o enlace entre os nodos b e d. Como b não possui outro enlace que chegue ao nodo t, é necessário retornar ao nodo a. Porém, para que a não tente enviar o pacote novamente por b, é necessário informar ao nodo a que o enlace $b - d$ está falho. Após a atualização das informações da topologia, a escolhe um novo caminho até t, que será, por exemplo, o caminho que passa por c e e.

Para que esta abordagem produza resultados satisfatórios, é necessário que haja uma redundância nos caminhos entre os roteadores intermediários e o nodo de destino. Esta redundância é necessária para que haja desvios que não precisem retornar até a origem para alcançar o destino. Desta forma, conforme já descrito nas seções anteriores, torna-se importante a escolha de caminhos robustos baseada em critérios de conectividade, que introduzem naturalmente a redundância dos caminhos.

5. Avaliação de Confiabilidade

Conforme já descrito nas seções anteriores, o objetivo do roteamento confiável baseado na escolha de caminhos robustos é possibilitar que, em caso de identificação de uma falha no caminho, seja possível encontrar um desvio, ou seja, um novo caminho que chegue ao destino sem que seja necessário retornar ao nodo inicial.

Para avaliar se o objetivo foi encontrado, este trabalho realiza uma comparação com o algoritmo de Dijkstra [2]. O objeto de avaliação é a robustez do caminho, ou seja, a possibilidade de encontrar um desvio em caso de falha. A métrica utilizada neste trabalho para comparação quantitativa da robustez de um caminho é o tamanho do caminho total atravessado pelos dois algoritmos. Este caminho inclui as arestas atravessadas até o nodo adjacente a uma aresta falha, e as arestas do novo caminho (desvio) entre esse nodo e o destino. O desvio foi calculado da mesma forma que o caminho original, porém desconsiderando a aresta identificada como falha e tendo como origem o nodo adjacente a esta aresta.

Na figura 5 podemos observar como ambos os algoritmos se comportam em relação a desvios. Pelo algoritmo de Dijkstra, o caminho escolhido será o caminho $p_D = (s, x, y, z, t)$. Porém, ao chegar ao nodo z, identifica-se uma falha que impossibilita a continuidade do caminho. Desta forma, um novo caminho é escolhido para chegar de z até o destino t. Tendo em vista que a aresta (z, t) já foi identificada como falha, ela não poderá ser utilizada, e o melhor caminho então é $p'_D = (z, y, x, a, c, e, g, t)$. Desta forma, o caminho total percorrido é de 10 arestas, 3 de s até z e 7 de z até t.

Pela abordagem descrita neste trabalho se utilizarmos como pesos os valores 2, 0.001 e -1 respectivamente para ω_1, ω_2 e ω_3, e com α e β valendo 1, os melhores caminhos

serão $p_S = (s, a, c, e, g, t)$ e $p'_S = (s, b, d, f, h, t)$, que ficarão com mesma avaliação ($\Gamma(p_S) = \Gamma(p'_S)$). Para fins de entendimento do funcionamento do algoritmo, suponhamos que o caminho escolhido é p'_S. Percorrendo este caminho, ao chegar em h, é identificada uma falha, que impossibilita a continuidade do caminho. É necessário, então, encontrar um novo caminho entre h e t que não contenha esta aresta. Chegamos facilmente ao caminho $p''_S = (h, g, t)$. Se verificarmos o caminho total, podemos verificar que foram percorridas 6 arestas, 4 de s até h e 2 de h até t.

O exemplo acima ilustra a robustez dos caminhos que estão sendo propostos neste trabalho. Porém, para uma avaliação mais completa, foi desenvolvida uma plataforma de simulação, de forma a verificar o comportamento dos algoritmos em situações diversas. Esta plataforma recebe como entrada um grafo $G = \langle V, E \rangle$ correspondente à topologia de uma rede, e realiza testes de robustez neste grafo.

A plataforma descrita acima considera, para o cálculo, todas as situações de falhas possíveis em que haja uma aresta falha. Para cada aresta, são considerados todos os pares possíveis de origem e destino na rede, em que a origem e o destino sejam nodos diferentes do grafo G. Temos, portanto, uma situação σ descrita da seguinte forma:

$$\sigma = \langle e, s, t \rangle, e \in E(G), s, t \in V(G), s \neq t$$

Para cada situação σ, o caminho p_D entre os nodos s e t é calculado pelo algoritmo de Dijkstra. Igualmente, o caminho p_S é calculado pelo algoritmo proposto neste trabalho. Em ambos os caminhos, verifica-se se a aresta e pertence ao caminho. Se pertencer, é calculado um novo caminho p'_D (ou p'_S) entre o nodo em que a falha é identificada e o nodo t, agora desconsiderando a aresta e. Neste caso, o tamanho total considerado é a soma entre o caminho até o ponto em que a falha ocorre e o tamanho do caminho p'_D (ou p'_S). Caso a aresta e não pertença ao caminho, o tamanho total considerado é o próprio tamanho do caminho p_D (ou p_S).

Os valores encontrados são armazenados, e após a avaliação de todas as situações, a média dos valores encontrados para o algoritmo de Dijkstra e para o algoritmo proposto é apresentado.

5.1. Implementação

De forma a possibilitar a avaliação da robustez de caminhos, foi implementada uma ferramenta de cálculo e avaliação dos conceitos apresentados neste trabalho. Esta implementação foi feita na linguagem Java 1.4.2 [8].

Foi utilizada uma biblioteca disponível na Internet para trabalhar com grafos, denominada JDigraph [9]. Além disso, foi utilizada a estrutura de Servlets e JSPs (*Java Server Pages*), introduzida pela plataforma J2EE (*Java 2 Enterprise Edition*) [7]. Para executar estas estruturas, foi utilizado o servidor Java Tomcat [15].

Foi criada uma interface gráfica de acesso aos cálculos, acessível através de um browser da Web. Esta interface é demonstrada na figura 6. Nesta figura, (A) ilustra a página inicial, em que é solicitado o arquivo de grafo ou a estrutura para criação manual. (C) mostra a página principal, que lista os critérios de conectividade encontrados, bem como permite a seleção de nodos para cálculo dos melhores caminhos. Estes caminhos são mostrados numa nova página, ilustrada em (B). A partir da página principal, também é possível realizar uma simulação completa do grafo, envolvendo uma avaliação de todos os caminhos possíveis, com todas as situações de falha possíveis, conforme descrito anteriormente nesta seção. O resultado desta avaliação é demonstrado na mesma figura, em (D).

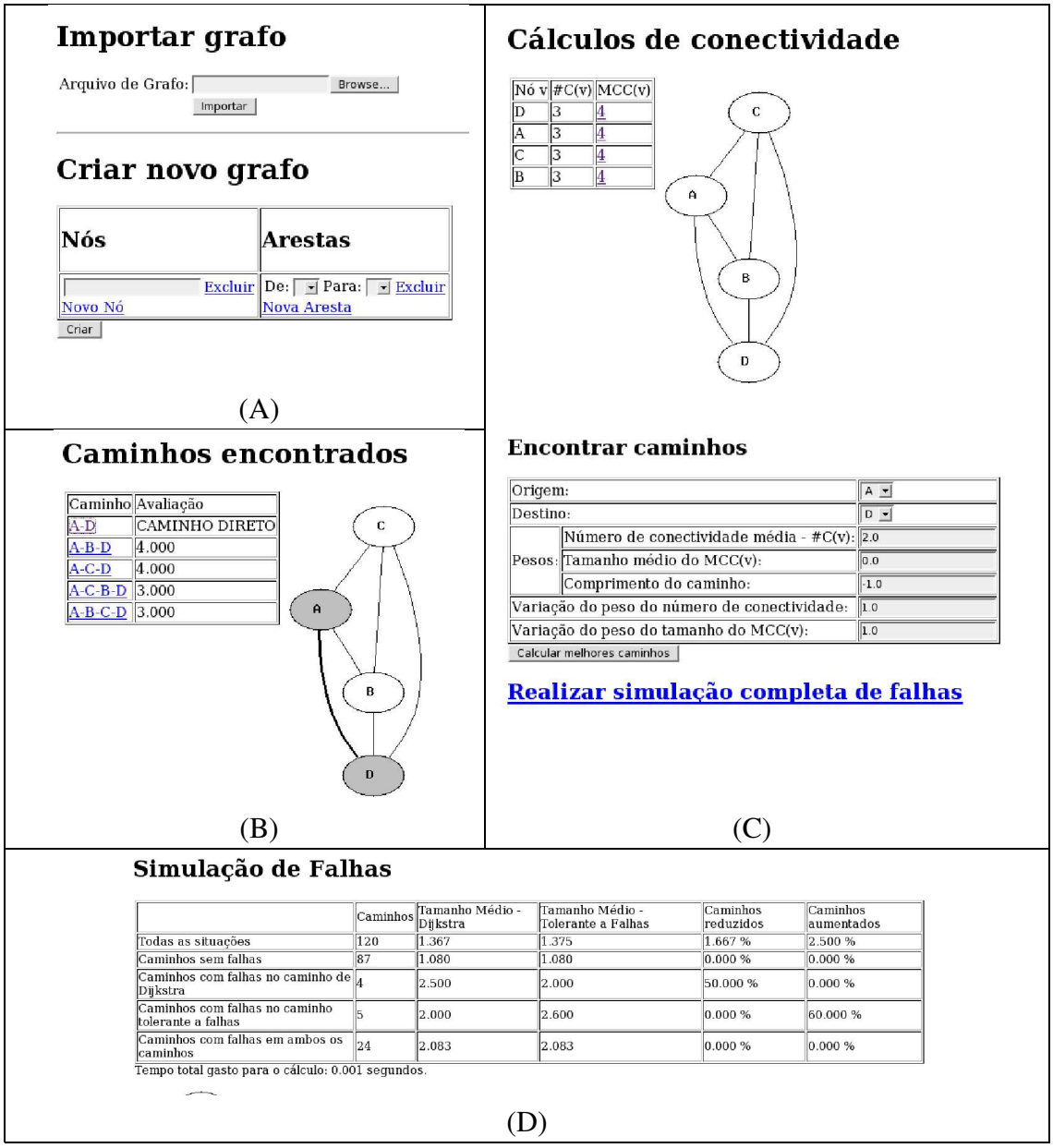


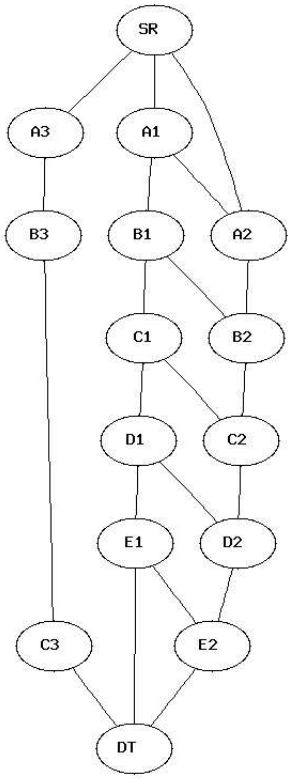
Figura 6: Exemplo da interface gráfica.

Um exemplo de grafo utilizado para testes é o mostrado na figura 7. Foi realizado um teste de simulação completa de falhas, conforme descrito no início desta seção. Os testes foram realizados com os parâmetros ω_1, ω_2, ω_3, α e β, descritos na seção 3, valendo, respectivamente, 2, 0.001, -1, 1 e 1.

O resultado da simulação de falhas pode ser visto na figura 7. Nesta figura, pode-se verificar que, em situações em que falha interrompe apenas o caminho de Dijkstra, o tamanho médio do caminho pela nova abordagem é 2,2 arestas menor que o caminho de Dijkstra, que necessitou do desvio. Porém, quando a falha interrompeu apenas o caminho da nova abordagem, a diferença entre os caminhos passou para 1,3, demonstrando que o desvio gerado neste caso é menor que o desvio pelo caminho de Dijkstra.

6. Conclusão

Este trabalho propôs uma estratégia de roteamento dinâmico e tolerante a falhas, que se baseia na escolha de caminhos robustos e na utilização de caminhos alternativos (desvios) em caso de falhas no caminho inicial utilizado. O roteamento proposto produziu



	Número de Caminhos	Tamanho Médio	
		Dijkstra	Robusto
Caminhos sem falhas	7371	2.664	2.673
Caminho de Dikstra com falha	261	6.418	4.261
Caminho robusto com falha	269	4.186	5.494
Caminho de ambos os tipos com falha	919	5.751	5.766
Todos os caminhos (média geral)	8820	3.143	3.128

Figura 7: Resultados obtidos com um grafo de exemplo.

resultados preliminares satisfatórios no que se refere à identificação de desvios. Os caminhos alternativos são relativamente curtos, tendo em vista que não há necessidade de retornar ao nodo de origem do pacote. Os caminhos robustos podem caracterizar uma possibilidade maior de desvios, dada a redundância introduzida por estes caminhos.

Trabalhos futuros devem incluir testes mais abrangentes, com grafos maiores e utilizando métodos como o de Waxman [17] para geração dos grafos, além do uso de valores distintos para os parâmetros ω_1, ω_2, ω_3, α e β. Também será estudada a utilização de técnicas e heurísticas para reduzir o tempo de escolha do caminho, como o descarte de alguns caminhos e o cálculo heurístico do $\widetilde{\#}C(v)$ proposto em [3]. Além disso, devem ser examinados aspectos relacionados ao roteamento dinâmico, além da realização de testes com situações de mais de uma falha, bem como com falhas de nodos em lugar de apenas de arestas. Também serão avaliados outros critérios para inclusão na avaliação dos caminhos, como a latência e o congestionamento do caminho, assim como medidas de dispersão dos critérios de conectividade, como o desvio padrão. Futuramente deve-se avaliar formas de avaliação de caminhos em redes com topologia parcialmente desconhecida, como no roteamento entre sistemas autônomos distintos.

Referências

[1] J. Chandrashekar, Z. Duan, Z. L. Zhang, and J. Krasky. Limiting path exploration in BGP. disponível em `http://www-users.cs.umn.edu/~jaideepc/papers/epic-tr.pdf`.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, second edition, 1990.

[3] E. P. Duarte Jr., R. Santini, and J. Cohen. Delivering packets during the routing convergence latency interval through highly connected detours. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'2004)*, pages 495–504, Florence, Italy, 2004.

[4] L. R. Ford Jr. and D. R. Fulkerson. Flows in networks. *Princeton University Press*, 1962.

[5] R. E. Gomory and T. C. Hu. Multi-terminal network flows. In *SIAM Journal on Applied Mathematics*, volume 9, pages 551–556, 1961.

[6] D. Gusfield. Very simple method for all pairs network flow analisys. In *SIAM Journal on Computing*, volume 19(1), pages 143–155, 1990.

[7] Java 2 Platform, Enterprise Edition (J2EE). `http://java.sun.com/j2ee`.

[8] Java Technology. `http://java.sun.com`.

[9] JDigraph. `http://jdigraph.dev.java.net`.

[10] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *SIGCOMM*, pages 175–187, 2000.

[11] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. Wu, and L. Zhang. Improving BGP convergence through consistency assertions. In *INFOCOM*, New York, 2002.

[12] Y. Rekhter and T. Li. *RFC 1771: A Border Gateway Protocol 4 (BGP-4)*, March 1995.

[13] R. Santini, E. P. Duarte Jr., J. Schroeder, P. R. Torres Jr., and J. Cohen. Roteamento tolerante a falhas baseado em desvios de alta conectividade. Technical report, Universidade Federal do Paraná, 2004.

[14] J. Schroeder, A. L. P. Guedes, and E. P. Duarte Jr. Computing the minimum cut and maximum flow of undirected graphs. Technical report, Universidade Federal do Paraná, 2004.

[15] The Jakarta Site - Apache Jakarta Tomcat. `http://jakarta.apache.org/tomcat`.

[16] L. Wang, D. Massey, K. Patel, and L. Zhang. FRTR: A scalable mechanism for global routing table consistency. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'2004)*, pages 465–474, Florence, Italy, 2004.

[17] B. M. Waxman. Routing of multipoint connections. In *IEEE Journal of Selected Areas in Communications*, volume 6(9), pages 1617–1622, 1988.