# Roteamento Dinâmico Tolerante a Falhas Baseado em Avaliação de Fluxo Máximo

**Jonatan Schroeder, Elias Procópio Duarte Jr.**

Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19018 CEP 81531-990 – Curitiba - PR - Brasil

{jonatan,elias}@inf.ufpr.br

***Abstract.*** *This work proposes a fault-tolerant dynamic routing strategy, in which intermediate routers, having more recent information about topology changes, are able to switch the path employed. The proposed routing strategy chooses network edges for routing based on maximum flow evaluation, in order to increase the number of disjoint paths, enhancing the path redundancy, and so extending the possibility of using detours, or alternative paths. Route distance is employed as a secondary criterion. Formal proofs for correctness of the algorithm are also presented.*

***Resumo.*** *Este trabalho propõe uma estratégia de roteamento tolerante a falhas e dinâmico, que permite aos roteadores intermediários, que podem possuir informações mais recentes de alterações na topologia, interferirem na escolha do caminho utilizado. A estratégia de roteamento proposta baseia-se na escolha de arestas para roteamento utilizando cálculo de fluxo máximo em grafos, aumentando o número de caminhos disjuntos, o que valoriza a redundância de caminhos e, por conseqüência, ampliam a possibilidade de utilização de desvios ou caminhos alternativos. Critérios de distância da rota são utilizados como critérios secundários. Provas formais da correção do algoritmo são apresentadas.*

## 1. Introdução

Os protocolos de roteamento, após a ocorrência de uma alteração na topologia da rede, necessitam de um tempo de convergência para que atualizar suas tabelas de rotas em todos os roteadores. Esse período de tempo é conhecido como *latência de convergência* do protocolo [1]. Por exemplo, a latência média do protocolo BGP (*Border Gateway Protocol*), um dos mais utilizados na Internet atualmente [2], é de 3 minutos, sendo que já foram observados períodos de latência de até 15 minutos [3]. Durante a latência de convergência, pacotes enviados pela rede podem ser potencialmente perdidos, bem como conexões podem ser rompidas.

Neste trabalho é proposta uma abordagem de roteamento que faz a seleção de rotas tendo como um dos principais objetivos a geração de rotas robustas, no sentido de que, no caso da ocorrência de falhas em parte do caminho (nós ou enlaces), é possível encontrar outro caminho (ou *desvio*) que parte do ponto em que a falha é conhecida em direção ao destino da mensagem sendo roteada. A avaliação da robustez do caminho é feita com base no tamanho desse desvio em relação à distância dos nós de origem e destino na rede.

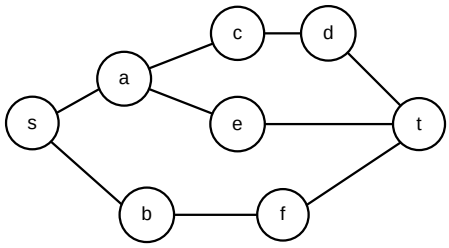


**Figura 1. Um grafo utilizado como exemplo para o algoritmo.**

A robustez de uma rota utilizada é maior quanto mais próximo for o número total de nós visitados do caminho mínimo entre os nós na rede.

Além da robustez dos caminhos escolhidos, outra característica importante é que o roteamento proposto neste trabalho é dinâmico. Em um roteamento dinâmico, cada nó que recebe a mensagem a ser roteada escolhe apenas a próxima aresta da rota, permitindo que roteadores intermediários da rota possam escolher uma rota melhor que as conhecidas pelo nó de origem da mensagem. Esse roteamento explora o fato de que nós mais próximos a um evento de alteração de estado de nó ou enlace recebem a informação do evento antes dos demais nós da rede. Este conceito pode ser estendido também a informações de congestionamento, caso em que uma aresta pode ser considerada temporariamente falha. O algoritmo funciona mesmo que a tabela não esteja atualizada, ou possua uma posição incorreta da topologia da rede.

A figura 1 ilustra o funcionamento do roteamento proposto. Suponha que o nó $s$ precise enviar uma mensagem para o nó $t$. Se $s$ enviar a mensagem para o nó $a$, este nó terá duas alternativas (dois caminhos disjuntos) para chegar no destino $t$. O nó $b$, por sua vez, possui apenas um caminho possível. Desta forma, o nó $a$ terá prioridade na escolha do caminho a ser usado para o roteamento.

Diversos trabalhos relacionados têm sido propostos para evitar a perda de pacotes e conexões durante a latência de convergência dos protocolos de roteamento. Chandrashekar et.al. [4] propõem um mecanismo para análise das dependências de caminhos, de forma a reduzir a exploração de novos caminhos e, por conseqüência, reduzir a latência de convergência. Pei et.al. [5] propõem uma série de asserções a serem aplicadas nas redes, com o propósito de comparar caminhos similares e eliminar caminhos inviáveis. Wang et. al. [6] 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.

Uma abordagem para aumentar a tolerância a falhas dos caminhos escolhidos, proposta em [7], é baseada na identificação de nós altamente conexos na rede, com o intuito de encontrar caminhos que passem por esses nós. Nós conexos são nós que possuem grande quantidade de conexões com outros nós da rede. Propõe-se que, em caso de ocorrência de falhas na conexão, se encontre um desvio que passa pelo nó altamente conexo de forma a chegar ao destino. Há uma ênfase na conectividade individual dos nós altamente conexos, sem que haja uma preocupação com a conectividade dos demais nós do caminho escolhido para a comunicação.

Foi desenvolvida uma implementação do algoritmo, utilizando a linguagem Java

1.4.2 [8]. Os detalhes da implementação, bem como os resultados obtidos em redes com topologia similar à da Internet, estão disponíveis em [9]. A implementação está disponível para execução na Web, em `http://www.inf.ufpr.br/jonatan/mfrp`.

Este trabalho está organizado como segue. Na seção 2 é especificado o algoritmo. A seção 3 apresenta as provas de correção do algoritmo. Por fim, a seção 4 apresenta as conclusões.

## 2. O Algoritmo Proposto

O algoritmo de roteamento proposto nesse trabalho recebe como entrada um par de nós da rede, correspondentes à origem e ao destino de uma mensagem a ser enviada. O algoritmo, então, é executado em cada nó da rede, iniciando pelo nó de origem, escolhendo o próximo nó da rota, dentre os seus nós vizinhos. Quando a mensagem chega a um nó escolhido, esse nó executa o mesmo algoritmo para escolher o nó seguinte, e assim por diante, até que o destino seja alcançado.

A abordagem desse algoritmo é similar à adotada pelo algoritmo de Bellman-Ford, no sentido de que apenas o próximo nó para comunicação é escolhido. Uma das principais vantagens dessa abordagem é o funcionamento do mesmo ainda que a topologia conhecida pela origem não possua as alterações mais recentes da topologia real da rede [1].

Pelo algoritmo proposto, cada nó da rede escolhe o próximo nó da rota avaliando cada aresta adjacente ao mesmo, avaliação esta baseada em um compromisso (*trade-off*) entre redundância e distância dos caminhos da aresta avaliada. Após avaliar as arestas, a aresta com melhor avaliação é escolhida.

As fórmulas e equações para o cálculo das métricas utilizadas nesse trabalho assumem que cada nó possui uma representação local da topologia da rede. Essa representação da topologia, porém, não é necessariamente completa e atualizada. A representação é feita através de uma estrutura de grafos direcionados, com um conjunto de nós e um conjunto de arestas. Essa estrutura é atualizada periodicamente através de troca de mensagens, conforme descrito na seqüência desse trabalho.

O modelo de falhas considerado neste trabalho é o das falhas *crash*, e o sistema é considerado parcialmente síncrono, ou seja, existe um limite de tempo finito e não necessariamente conhecido para o atraso na comunicação entre dois nós quaisquer.

### 2.1. Especificação do Algoritmo

Inicialmente nesta seção são apresentadas algumas definições preliminares utilizadas no algoritmo proposto.

Um *grafo* (ou *rede*) direcionado $G$ é um par $(V, E)$ de conjuntos, em que $V$ é um conjunto de nós (ou vértices) e $E$ é um conjunto de arestas (ou enlaces). Cada aresta é um par ordenado de exatamente dois nós diferentes.

Seja $G = (V, E)$ um grafo; seja $c : E \rightarrow \Re$ uma função correspondente à capacidade das arestas do grafo; sejam $u, v \in V$ nós do grafo $G$. Um *fluxo* entre $u$ e $v$ é uma função $f : E \rightarrow \Re$ tal que:

$$\forall e \in E, f(e) \leq c(e)$$

$$\forall t \in V - \{u, v\}, \sum_{e=(t,t') \in E} f(e) = \sum_{e=(t',t) \in E} f(e)$$

Dizemos que o tamanho (ou cardinalidade) de um fluxo $f$, denotado por $|f|$, é igual a:

$$|f| = \sum_{e=(u,t) \in E} f(e) - \sum_{e=(t,u) \in E} f(e)$$

Dizemos que um fluxo $f$ é máximo se para todo fluxo $f'$ entre o mesmo par de nós, $|f| \leq |f'|$.

Sejam $u, v \in V$ nós do grafo $G$. Um *corte* entre $u$ e $v$ é um conjunto de arestas $C$ tais que, removendo todas as arestas em $C$ do grafo $G$, $u$ e $v$ deixam de ser conexos. Dizemos que o tamanho (ou cardinalidade) de um corte $C$, denotado por $|C|$, é igual à cardinalidade do conjunto de arestas. Um corte $C$ é dito mínimo se, para todo corte $C'$ entre o mesmo par de nós, $|C| \leq |C'|$. Para qualquer par de nós da rede, o fluxo máximo e o corte mínimo possuem a mesma cardinalidade, e podem ser calculados utilizando os mesmos algoritmos [10]. Desta forma, o restante do trabalho utiliza os dois conceitos em conjunto.

Quando um nó de origem precisa enviar uma mensagem para um nó de destino, ou um nó intermediário recebe uma mensagem da qual não é o destinatário, o seguinte algoritmo é executado ($n$ corresponde ao nó que realiza o roteamento):

1. Inclui o nó $n$ à lista de nós visitados da mensagem.
2. Se houver uma aresta que liga o nó $n$ ao destino, envia a mensagem utilizando essa aresta e termina o processo.
3. Cria um grafo auxiliar $G'$ a partir da topologia conhecida, removendo os nós visitados da mensagem.
4. Avalia cada uma das arestas adjacentes ao nó $n$ utilizando a função $\Gamma(e)$ no grafo $G'$. Arestas que levam a nós visitados ou que não possuam caminhos disponíveis até o destino não são avaliados.
5. Se pelo menos uma aresta foi avaliada, envia a mensagem pela aresta com melhor avaliação.
6. Se nenhuma aresta foi avaliada:
   (a) Remove $n$ da lista de nós visitados.
   (b) Se $n$ é a origem da mensagem, retorna um erro.
   (c) Se $n$ é um nó intermediário, envia uma mensagem de atualização seguida da mensagem roteada para o último nó na lista de nós visitados.

A avaliação das arestas a serem utilizadas no roteamento é feita em um subgrafo do grafo que corresponde à representação local da topologia da rede. Este subgrafo é gerado pela remoção dos nós já visitados pela mensagem, e pelas arestas adjacentes aos mesmos. Esse tratamento é feito para evitar ciclos no caminho percorrido.

Há, porém, um caso específico em que arestas que geram ciclos são consideradas. Levando em conta que as informações nos nós não estão necessariamente atualizadas, um nó pode ter sido escolhido para o roteamento em virtude de caminhos que não existem mais, ou que estão falhos. No momento em que a mensagem alcança um nó que já possua essa informação mais atualizada, esse nó poderá verificar que as únicas arestas que

possuem caminhos até o destino levam a nós já previamente visitados. Neste caso, é necessário retornar a mensagem à aresta a partir da qual a mensagem chegou ao nó. Porém, o nó que receber essa mensagem provavelmente terá informações desatualizadas sobre a rede, visto que selecionou para o roteamento um nó sem opções de rotas. Logo, para que esse nó possua informações mais atuais, a mensagem de roteamento programada para ser enviada para esse nó é antecipada, e é enviada antes de retornar a mensagem para o nó. Desta forma, o novo nó poderá tomar a decisão de um novo caminho baseando-se em informações mais atuais de topologia, evitando assim caminhos sem opções de rotas.

Para possibilitar que as informações de rotas disponíveis reflitam as alterações de topologia à medida em que essas ocorrem, há um processo de atualização executado em cada nó. Esse processo é executado a cada $\alpha$ segundos, com $\alpha$ podendo ser parametrizado. O processo envia, para todos os seus vizinhos, uma lista contendo as alterações de topologia que teve conhecimento. Ao recebimento de uma mensagem por uma aresta que era considerada falha, o nó que recebe a mensagem inclui a aresta na sua representação local. Caso nenhuma mensagem seja recebida de um determinado nó após um *timeout* parametrizado, a aresta correspondente é considerada falha. Este *timeout* é chamado de $\beta$ ($\beta > \alpha$).

Após o recebimento de uma mensagem de atualização, cada nó envia uma mensagem de confirmação (*Acknowledgement*) para o nó que enviou a atualização. O recebimento de uma mensagem de confirmação permite ao nó que o recebe saber que o nó vizinho já recebeu a informação de atualização com sucesso, não sendo mais necessário enviar as mesmas informações novamente.

### 2.2. Avaliação das Arestas - Redundância *versus* Distância

O cálculo da avaliação das arestas se baseia em uma série de critérios quantitativos relacionados à redundância e ao comprimento dos caminhos relacionados às arestas. Para cada um desses critérios, é associado um peso parametrizável, de forma a possibilitar a ênfase maior ou menor em um critério ou outro.

A equação correspondente ao cálculo da avaliação das arestas está descrita a seguir:

$$\Gamma(G, e) = \sum_{c_n \in C} \omega_n \times c_n(e) \tag{1}$$

Nessa equação, $\Gamma(G, e)$ é a função de avaliação (*trade-off*), $e$ é a aresta sendo avaliada, $C$ é o conjunto de critérios (que será descrito na seqüência) e $\omega_n$ é o peso associado ao critério $c_n$.

Os critérios utilizados para avaliação das arestas são funções que, a partir de uma aresta, retornam um valor numérico. Inicialmente são utilizados como critérios: a cardinalidade do fluxo máximo (ou do corte mínimo) entre o nó correspondente à aresta avaliada e o nó de destino ($c_1$) e o comprimento do menor caminho entre esses nós ($c_2$).

O principal critério utilizado neste trabalho para avaliação das arestas para roteamento é o fluxo máximo (ou corte mínimo) entre o nó correspondente à aresta avaliada e o destino. Este critério é chamado de $c_1$. Um algoritmo clássico para a avaliação do fluxo máximo é o algoritmo de Ford-Fulkerson [10, 11]. O algoritmo utiliza a estrutura de grafo valorado, em que, para um grafo $G = (V, E)$ temos uma função $c : E \rightarrow \Re$,

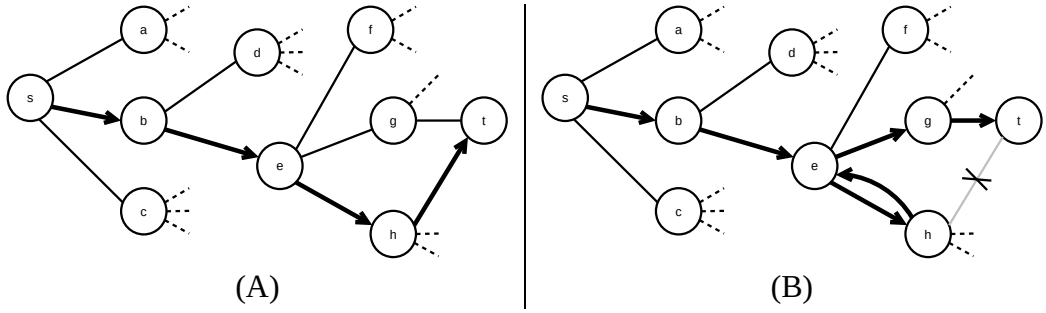


**Figura 2. Um exemplo de como o algoritmo funciona.**

que associa cada aresta a um valor. Assume-se que a função $c(e)$ retorna 1 para todas as arestas do grafo. A complexidade do cálculo do fluxo máximo, para custos com valor inteiro, é $O(NM)$, com $N$ sendo o número de nós e $M$ o número de arestas do grafo [11]. Para o peso correspondente a esse critério ($\omega_1$) é utilizado um valor positivo, visto que a avaliação deve ser diretamente proporcional ao resultado do mesmo.

Como forma de possibilitar a manutenção de um compromisso com a distância do caminho, um dos critérios utilizados na avaliação das arestas é a distância mínima entre o nó correspondente à aresta avaliada e o destino. Esse critério é chamado de $c_2$. Tendo em vista que as arestas não possuem pesos ou custos, utilizamos uma busca em largura, na qual o número de rodadas, ou níveis, percorridos para encontrar o nó de destino, partindo do nó avaliado, é utilizado como o valor do critério. A busca em largura é executada em $O(M)$, com $M$ sendo o número de arestas da rede [12, 11]. Para o peso correspondente a esse critério ($\omega_2$) é utilizado um valor negativo, visto que a avaliação deve ser inversamente proporcional ao resultado do mesmo.

### 2.3. Exemplos de Execução do Algoritmo

A figura 2 ilustra o funcionamento do algoritmo proposto neste trabalho. Nessa figura, em (A), vamos supor que o nó $s$ precisa enviar uma mensagem para o nó $t$. O algoritmo de roteamento é executado em $s$, avaliando todas as arestas adjacentes, ou seja, as arestas $(s, a)$, $(s, b)$ e $(s, c)$, com o objetivo de escolher qual dessas arestas será utilizada no caminho. Supondo que a aresta escolhida seja $(s, b)$, então a mensagem é enviada de $s$ para $b$.

Quando o nó $b$ recebe a mensagem enviada de $s$ para $t$, ele avalia todas as suas arestas adjacentes, $(b, d)$ e $(b, e)$. A aresta $(b, s)$ é descartada, visto que o nó $s$ já foi visitado e, portanto, é removido do grafo utilizado para avaliação. Suponha que a aresta escolhida seja $(b, e)$. A mensagem é então enviada para o nó $e$. Da mesma forma, $e$ avalia as arestas vizinhas, descartando $(e, b)$ e escolhendo $(e, h)$. Finalmente, o nó $h$ avalia suas arestas adjacentes. Como ele possui uma aresta que vai diretamente para o destino final $t$, as demais arestas são descartadas, e a aresta $(h, t)$ é utilizada. Então, a mensagem é enviada para o nó $t$, chegando ao destino final.

Considere agora um outro exemplo. Suponha que, na mesma rede descrita acima, ocorre uma falha na aresta $(h, t)$, que é utilizada no caminho entre $s$ e $t$, conforme descrito. A figura 2 ilustra em (B) o resultado da rede após essa falha. Suponha, novamente, que $s$ precisa enviar uma mensagem para $t$, pouco depois da ocorrência da falha. Suponha que, no momento do envio da mensagem, apenas os nós $h$ e $t$ (vizinhos da aresta falha) têm o

conhecimento dessa falha. O algoritmo é executado no nó $s$, enviando a mensagem para $b$, que por sua vez envia a mensagem para $e$, que envia a mensagem para $h$. O procedimento ocorre da mesma forma que o exemplo anterior nestes nós, visto que suas informações de topologia ainda não foram alteradas.

Quando o nó $h$ receber a mensagem que se destina a $t$, ele verificará que os únicos caminhos possíveis para $t$ são através de arestas que ligam a nós já visitados, como o nó $e$. Visto que não há outra alternativa para o envio dessa mensagem, o nó $h$ envia uma mensagem de atualização de topologia para $e$. Com o recebimento dessa mensagem, $e$ estará apto a tomar uma decisão baseado em uma informação mais recente da topologia. Finalmente, após o envio da mensagem de atualização, $h$ envia a mensagem original para $e$. Quando o nó $e$ receber novamente a mensagem original, esse tomará uma nova decisão sobre a aresta que deverá ser utilizada. Essa decisão é tomada com base em informações mais recentes dos caminhos disponíveis, tendo conhecimento de que a aresta $(h, t)$ não está funcional. Desta forma, uma outra aresta é escolhida, no caso, a aresta $(e, g)$, levando a mensagem ao nó $g$. Esse nó envia a mensagem para o nó $t$ através da aresta $(g, t)$, e a mensagem chega ao seu destino.

## 3. Provas

Esta seção apresenta provas de correção do algoritmo, assim como resultados obtidos para o número e tamanho das mensagens utilizadas, da complexidade do processamento e da latência de correção do algoritmo proposto.

### 3.1. Correção

O primeiro teorema a ser provado corresponde à correção do algoritmo, ou seja, se houver um caminho entre dois nós, o algoritmo efetuará o roteamento de uma mensagem com sucesso entre esses dois nós. Esse teorema assume algumas hipóteses para sua correção. A primeira hipótese afima que nós adjacentes a uma aresta (ou a outro nó) possuem conhecimento do estado dessa aresta (ou nó). A segunda hipótese utilizada é a de que o nó de origem do roteamento precisa conhecer pelo menos um caminho não falho até o destino, ou seja, na representação local da topologia no nó de origem, pelo menos um dos caminhos disponíveis entre a origem e o destino no grafo não está falho na rede, não necessariamente o caminho utilizado. Na terceira hipótese assume-se que não haja mudança de topologia da rede entre o início do envio de uma mensagem até o momento em que a mensagem chega ao destino final, ou até o momento em que a origem toma conhecimento da inexistência de caminhos disponíveis ao destino.

Para a prova deste teorema, inicialmente é provado um lema que indica que, se a mensagem retornar à origem após o envio de mensagens por todas as arestas com caminho conhecido, a origem terá o conhecimento de que não há caminho conhecido sem falhas e ficará sem alternativas de roteamento. Em seguida é provado um lema que indica que, se houver um caminho funcional iniciando pela aresta escolhida para o roteamento, esse caminho é utilizado. Na seqüência, é provado um lema que indica que, se houver um ou mais caminhos conhecidos pela origem sem falhas, então uma aresta que pertença a um desses caminhos é selecionada para o roteamento. Finalmente, utilizando os lemas provados, é provado o teorema da correção do algoritmo.

**Lema 3.1.** *Considere $G = (V, E)$ um grafo, e $s, t \in V$ dois nós não falhos desse grafo. Considere que todos os nós possuem o conhecimento do estado de seus nós (ou arestas)*

*vizinhos. Se $s$ enviar uma mensagem que tenha destino $t$ utilizando o algoritmo proposto neste trabalho, e todas as arestas avaliadas forem desprezadas ou, após o envio, retornarem a mensagem para a origem, então a origem $s$ terá conhecimento de que não possui alternativas para o roteamento.*

**Prova.** Assume-se que o conjunto de arestas adjacentes e sem falhas a $s$ seja o conjunto $\{e_1, e_2, \ldots e_n\}$. Nesse conjunto, são desprezadas arestas que não possuam caminhos conhecidos até $t$ e que serão naturalmente desprezados pelo algoritmo. Como base de uma indução, é assumido que esse conjunto esteja vazio, ou seja, $s$ não possui arestas adjacentes. Neste caso, a prova é trivial, pois $s$ não possui alternativas para o roteamento.

Como desenvolvimento da indução, é assumido como hipótese que o lema é verdadeiro para o conjunto de arestas $\{e_1, e_2, \ldots e_{n-1}\}$. Deve-se provar que o mesmo lema é verdadeiro caso o conjunto de arestas adjacentes à origem $s$ seja o conjunto $\{e_1, e_2, \ldots e_{n-1}, e_n\}$. Assume-se, sem perda de generalidade, que a aresta escolhida para o roteamento é a aresta $e_n$. Portanto, a mensagem é enviada para o próximo nó utilizando essa aresta, nó que será chamado de $u$. Uma das hipóteses afirma que a mensagem retorna para a origem através dessa aresta. Pelo algoritmo, o retorno de uma mensagem ocorre apenas após o envio, do nó $u$ para o nó $s$, de uma mensagem com todas as informações atualizadas de caminhos disponíveis utilizando o nó $u$. Esse retorno ocorre apenas quando o nó $u$ não possui alternativas de roteamento até o destino $t$, e portanto não possui caminhos disponíveis até o destino $t$. Ao receber a mensagem de atualização, o nó $s$ atualizará sua lista de roteamento. Com o retorno da mensagem original, o nó $s$ fará uma nova avaliação de suas arestas, e a aresta $e_n$ é então desprezada, visto que a informação mais atual recebida dessa aresta aponta a inexistência de caminhos até o destino $t$. Portanto, o conjunto de arestas disponíveis será $\{e_1, e_2, \ldots e_{n-1}\}$, para o qual o lema é verdadeiro. Logo, prova-se que $s$ ficará sem alternativas para roteamento. □

**Lema 3.2.** *Considere $G = (V, E)$ um grafo, $s, t \in V$ dois nós não falhos desse grafo. Considere que todos os nós possuem o conhecimento do estado de seus nós (ou arestas) vizinhos. Considere que há um caminho de $s$ até $t$, e que $(s, u)$ é a primeira aresta desse caminho ($u$ pode ser igual a $t$). Se $s$ enviar uma mensagem para $t$ através da aresta $(s, u)$, utilizando o algoritmo proposto neste trabalho, a mensagem chegará ao destino $t$, ou retornará até a origem $s$.*

**Prova.** Como base de uma indução, assume-se que o grafo $G$ possui 2 nós. Esses nós correspondem aos nós $s$ e $t$ que, por definição, são diferentes. A única aresta que pode ser avaliada pelo nó $s$ é a aresta $(s, t)$, visto não haver outra possibilidade de existência de arestas nesse grafo, logo $t = u$. Se essa aresta estiver falha, pela hipótese $s$ tem conhecimento desta falha, e $(s, u)$ não pode ser utilizado para o roteamento, caso em que a hipótese não se aplica. Se, porém, a aresta não estiver falha, então a mensagem é enviada pelo nó $s$ diretamente ao nó $t$ pela aresta $(s, t)$, chegando ao nó $t$ e confirmando a base da indução.

Assumindo como hipótese de indução que o lema é válido para um grafo com $(n - 1)$ nós, deve-se provar que o lema também é válido para $n$ nós. Suponha, então, que $G$ possui $n$ nós. Quando $s$ envia a mensagem usando a aresta $(s, u)$, assume-se que $s$ sabe que a aresta não está falha, visto que $s$ possui conhecimento do estado das arestas

vizinhas. O nó $u$, que recebe a mensagem, ao realizar a avaliação das arestas, remove o nó $s$ do grafo utilizado para a avaliação, e portanto continua o roteamento em um grafo com $(n-1)$ nós, no qual o lema é válido. Logo, por indução, prova-se que o lema é válido para qualquer número de nós. □

**Lema 3.3.** *Considere $G = (V, E)$ um grafo, $s, t \in V$ dois nós não falhos deste grafo. Considere que todos os nós possuem o conhecimento do estado de seus nós (ou arestas) vizinhos. Considere que $s$ conhece um caminho válido até o destino $t$. Se $s$ deseja enviar uma mensagem para $t$, utilizando o algoritmo proposto neste trabalho, uma aresta que pertença a um caminho válido é selecionada.*

**Prova.** Como base de uma indução, considere que $s$ possui apenas um vizinho, denominado de $u$. Como há um caminho válido de $s$ até $t$, esse caminho precisa passar por $u$ (que pode ser o próprio $t$). Na seleção das arestas para avaliação, a aresta $(s, u)$ é considerada (visto que $s$ conhece um caminho sem falhas que passa pela aresta), e como é a única aresta disponível, é selecionada para o roteamento. Pelo lema anterior, como a mensagem é enviada através dessa aresta, e a aresta pertence a um caminho válido, então a mensagem chega ao destino.

Em seguida, considerando que o lema é válido para $(n-1)$ vizinhos, considere que $s$ possui $n$ vizinhos, denominados de $u_1$, $u_2$, $\ldots u_{n-1}$, $u_n$. Pelo algoritmo, algumas arestas serão desprezadas por não pertencerem a caminhos válidos conhecidos até o destino. Suponha que a aresta $(s, u_k)$ seja desprezada, qualquer que seja $1 \leq k \leq n$. Nesse caso, o algoritmo continua sem a aresta, como se o nó possuísse $(n-1)$ arestas, caso em que o lema é válido.

Considere que nenhuma aresta é desprezada. Nesse caso, sem perda de generalidade, considere que a aresta com melhor avaliação é a aresta $(s, u_n)$. O algoritmo envia uma mensagem por essa aresta. Se a aresta pertencer a um caminho válido até $t$, a mensagem chegará ao destino, conforme provado no lema 3.2. Se, porém, não houver caminho válido, pelo lema 3.1, a mensagem retorna ao nó $u_n$ sem alternativas de roteamento, e este nó envia a mensagem de volta para o nó $s$, com informações mais atuais referentes ao estado das arestas da rede. A aresta $(s, u_n)$ é, então, desprezada, por não pertence a nenhum caminho válido até o nó $t$. O algoritmo continua com $(n-1)$ arestas, caso em que o lema é válido. Com isto, provamos que o lema é válido para qualquer número de vizinhos da origem. □

**Teorema 3.4.** *Considere $G = (V, E)$ um grafo, e $s, t \in V$ dois nós não falhos desse grafo. Considere que os nós possuem o conhecimento da situação de seus nós vizinhos. Considere, também, que o nó $s$ conhece pelo menos um caminho para o destino $t$ que esteja funcional (sem falhas). Se $s$ enviar uma mensagem para $t$ utilizando o algoritmo proposto neste trabalho, a mensagem chegará até $t$.*

**Prova.** Como primeiro caso, considere a existência de uma aresta sem falha que liga diretamente $s$ a $t$. Nesse caso, como $s$ é vizinho à aresta e, portanto, possui conhecimento de sua funcionalidade, pelo algoritmo, a mensagem de $s$ a $t$ é enviada pela aresta $s - t$, chegando ao nó $t$, conforme proposto.

Como segundo caso, considere a não existência de uma aresta ligando diretamente os nós $s$ e $t$. Esse segundo caso se aplica também quando a aresta ligando os dois nós

existe e está falha. Pelo lema 3.3, como é assumido que a origem $s$ conhece um caminho sem falhas até o destino $t$, logo foi concluído que uma aresta que possui um caminho sem falhas é escolhida. Pelo lema 3.2, quando a aresta escolhida possui um caminho sem falhas, a mensagem chega ao destino, logo, conclui-se que a mensagem chega ao destino $t$. □

### 3.2. Avaliação do Algoritmo de Roteamento

O trabalho [9] prova que o algoritmo de roteamento proposto neste trabalho envia para atualização da topologia $O(M)$ mensagens a cada $\alpha$ segundos, e que cada mensagem tem tamanho $O(M)$. O trabalho também prova que o tempo necessário para escolha de uma rota entre uma origem e um destino é $O(M^3)$.

Em relação à latência de funcionamento do algoritmo, é provado, também no trabalho [9], que quando a origem de uma mensagem já conhece um caminho que está sem falhas até o destino, o tempo máximo necessário a partir de um evento de falha de aresta ou nó para que a mensagem chegue ao destino corretamente é de $\beta$ segundos. Quando a origem não conhece um caminho até o destino, o tempo máximo necessário a partir de um evento de criação ou recuperação de uma aresta ou nó para que a mensagem chegue ao destino corretamente é $O(D(G)\alpha)$ segundos, sendo $D(G)$ o diâmetro do grafo correspondente à topologia da rede. Este último caso pode ocorrer quando a aresta que sofre alteração de estado particiona a rede, ou no início do funcionamento de uma rede, quando nenhum dos nós possui informações suficientes para roteamento.

## 4. Conclusão

Neste trabalho foi apresentada uma proposta de roteamento tolerante a falhas, fundamentada em uma escolha dinâmica de caminhos, selecionados com base em uma avaliação de fluxo máximo, utilizando como critério secundário a distância dos caminhos. A abordagem de roteamento proposta foi especificada. A correção do algoritmo foi provada em situações definidas, e concluiu-se que a complexidade tanto do roteamento quanto das mensagens de atualização é polinomial em termos do número de nós e arestas da rede. Foi obtida também a latência de convergência do roteamento proposto. Resultados experimentais obtidos através de simulação em redes com topologias similares às da Internet foram apresentados, utilizando diferentes valores para os parâmetros $\alpha$, $\beta$, $\omega_1$ e $\omega_2$.

Outros critérios podem ser utilizados no futuro para avaliação das arestas para roteamento. Um desses critérios é o número total de caminhos disponíveis entre o nó avaliado e o destino. Outro critério é o comprimento médio desses caminhos. A utilização desses critérios foi avaliada neste trabalho, porém algoritmos triviais para identificação desses valores possuem complexidade não-polinomial, logo não foram utilizados. Trabalhos futuros podem também utilizar critérios relacionados a QoS, como o atraso, o custo, a largura de banda, entre outros. Pretende-se, também, avaliar a funcionalidade do protocolo para redes móveis. Futuramente pretende-se formalizar um protocolo utilizando o algoritmo proposto neste trabalho, através da descrição das mensagens para uso em uma rede real.

## Referências

[1] C. Huitema. *Routing in the Internet.* Prentice Hall, Upper Saddle River, $2^{nd}$ edition, 1999.

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

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

[4] J. Chandrashekar, Z. Duan, Z. L. Zhang, and J. Krasky. Limiting path exploration in BGP, 2005. disponível em `http://www-users.cs.umn.edu/~jaideepc/papers/epic-tr.pdf`.

[5] 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.

[6] 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.

[7] 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.

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

[9] Jonatan Schroeder. Roteamento Dinâmico Tolerante a Falhas Baseado em Avaliação de Fluxo Máximo. Master's thesis, Universidade Federal do Paraná - UFPR, Curitiba, March 2006.

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

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

[12] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.