Um Algoritmo Heurístico para Roteamento Robusto Baseado em Avaliação de Fluxo Máximo

Maverson E. S. Rosa, Elias P. Duarte Jr.

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 15.064 – 91.501-970 – Curitiba – PR – Brazil

{maverson,elias}@inf.ufpr.br

Abstract. *Maximum flow evaluation allows the selection of robust routes as it represents the topology redundancy in terms of the amount of disjoint routes available. In this paper we propose an efficient heuristic algorithm that allows the construction of routing tables using maximum flow and route length as criteria for route selection. The algorithm builds routing tables that map destination addresses to multiple output interfaces ranked according to estimates of robustness and cost. The algorithm employs backtracking to allow messages to try alternatives if the route in use fails. A proof of correctness is presented as well as experimental results obtained with simulation. Results show that the proposed strategy devises robust routes that are in average slightly larger than the corresponding shortest paths.*

Resumo. *A avaliação de fluxo máximo permite selecionar rotas robustas que possibilitam, na ocorrência de falhas, o uso de alternativas (desvios) que partem do ponto em que a falha é conhecida até o destino final da mensagem. O presente trabalho propõe um algoritmo heurístico eficiente para a construção de tabelas de roteamento baseadas em fluxo máximo e distância como critério secundário. O algoritmo permite o uso de* backtracking*: quando um caminho falha, as alternativas são avaliadas. São apresentados a prova formal da correção do algoritmo proposto e resultados de comparativos obtidos através de simulação. Esses resultados comprovam que, com pouca alteração no formato tradicional das tabelas de roteamento e um pequeno aumento no comprimento médio dos caminhos percorridos, é possível efetuar o roteamento robusto de mensagens de forma eficiente.*

1. Introdução

Organizações e indivíduos têm se tornado cada vez mais dependentes do bom funcionamento das redes de computadores, e da Internet em particular, para realizar as mais diferentes tarefas. Uma rede robusta depende necessariamente de estratégias de roteamento robustas, que sejam capazes de recuperar rapidamente após a ocorrência de falhas e eventos na topologia da rede. O roteamento robusto é imprescindível para que haja uma confiança realista no funcionamento da rede, em particular a Internet.

Diversas abordagens têm sido propostas recentemente para melhorar a confiança no roteamento da Internet. As soluções variam desde o uso de redes sobrepostas, como as *Resilient Overlay Networks* [ron 2009], para realizar o roteamento robusto, até propostas de modificações nos protocolos da Internet. Wang e Gao em [Wang and Gao 2009b]

propõem duas variações do protocolo BGP (*Border Gateway Protocol*) que usam a própria diversidade de rotas para destinatários para aumentar a confiabilidade ou, mais precisamente, a reatividade deste protocolo. Campisano *et. al.* [Campisano et al. 2008] atuam sobre outro ponto de vista: dada uma mudança de configuração do BGP, uma estratégia é proposta para identificar a causa primária desta mudança. Os autores apresentam dados mostrando que rotas entre sistemas autônomos são muito mais variáveis do que a intuição sugere.

Ainda outras abordagens para o roteamento tolerante a falhas na Internet consistem na atuação direta no próprio protocolo IP, por exemplo através dos endereços not-via [Wang and Gao 2009a], que representam uma forma de informar roteadores que ainda não atualizaram suas tabelas, que devem evitar certos caminhos da rede. O trabalho de Cicik e outros [Cicik et al. 2009] é uma alternativa para IPFRR (*IP Fast ReRoute*) que mantém diversas visões concorrentes da topologia da rede, permitindo a recuperação rápida após o ocorrência de falhas.

Em todas as abordagens para roteamento robusto é possível perceber que na raiz do problema está a definição e o uso de rota alternativa, a partir do momento que a rota pré-configurada como prioritária deixa de funcionar. Desta forma, a possibilidade de utilização, ou ao menos a disponibilidade de informações sobre múltiplas rotas para um destino, é útil para o roteamento. A avaliação de fluxo máximo [Ford Jr. and Fulkerson 1962, Cormen et al. 2003] é um critério viável para a seleção de rotas robustas [Schroeder and Duarte Jr 2006]. Esse critério é utilizado para computar a redundância dos caminhos, visto que, quanto maior o fluxo máximo, maior a quantidade de caminhos disjuntos e, consequentemente, maior é o número de desvios que podem ser utilizados em caso de falha.

Ohara apresenta uma nova família de algoritmos [Ohara et al. 2009] que, dada a topologia da rede, calcula um grafo direcionado acíclico (DAG - *Directed Acyclic Graph*) que maximiza o fluxo máximo. A eficiência do método proposto é avaliada através da complexidade dos algoritmos desenvolvidos e também através da simulação da estratégia para diversas topologias de sistemas autônomos da Internet. Apesar de que se trata de um trabalho relacionado com a proposta em [Schroeder and Duarte Jr 2006], na medida em que ambos utilizam a avaliação de fluxo para o roteamento robusto, a principal diferença é que para a construção do DAG é necessário um conhecimento prévio da topologia completa da rede, sendo portanto somente aplicável para roteamento interno a um sistema autônomo. Em [Schroeder and Duarte Jr 2006], é apresentado um algoritmo de roteamento baseado em avaliação de fluxo máximo que não necessita da topologia e pode ser aplicado para roteamento externo. Apesar de polinomial, o algoritmo apresenta custo elevado o que inviabiliza sua aplicação prática em protocolos de roteamento da Internet.

Neste trabalho é apresentado um algoritmo heurístico que permite a implementação prática de roteamento baseado em avaliação de fluxo máximo. O algoritmo proposto constrói uma tabela na qual para cada destino as múltiplas alternativas de saída são ordenadas de acordo com os critérios obtidos a partir da avaliação de fluxo máximo e do tamanho dos caminhos. O algoritmo utiliza *backtracking*: quando um caminho falha, uma mensagem pode retornar para tentar outra alternativa. O roteamento funciona corretamente mesmo que a tabela de roteamento mantida pelo nó que executa o algoritmo não reflita as últimas alterações ocorridas na rede. Inicialmente basta que cada

nó conheça seus vizinhos na rede. Além da prova de correção do algoritmo proposto, são apresentados resultados experimentais obtidos a partir de simulação. Esses resultados comprovam que, com uma alteração no formato das tabelas de roteamento e um pequeno aumento no comprimento médio dos caminhos percorridos, é possível efetuar o roteamento robusto de mensagens de forma eficiente.

O restante deste trabalho está organizado da seguinte maneira. Na seção 2 é descrita a aplicação de avaliação de fluxo máximo para o roteamento. Na seção 3, é apresentado o algoritmo heurístico proposto neste trabalho, incluindo a prova formal de correção. A seção 4 apresenta resultados experimentais obtidos através de simulação, comparando o algoritmo proposto ao algoritmo do caminho mais curto de Dijkstra [Dijkstra 1959] e ao algoritmo de [Schroeder and Duarte Jr 2006]. Trabalhos relacionados são apresentados na seção 5. A seção 6 conclui o trabalho.

2. Roteamento Baseado em Avaliação de Fluxo Máximo

A avaliação de fluxo máximo [Cormen et al. 2003] permite a determinação do caminho de maior capacidade entre dois vértices de um grafo $G = (V, E)$. É natural a aplicação de avaliação de fluxo máximo para o roteamento robusto pois o fluxo máximo é equivalente ao corte mínimo. A cardinalidade do corte mínimo m entre u e v é um conjunto de arestas A tais que, removendo todas as arestas em A 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.

Em [Schroeder and Duarte Jr 2006] a avaliação de fluxo máximo é proposta como critério para o roteamento robusto. O objetivo principal é selecionar rotas robustas, escolhidas de acordo com critérios de conectividade que possibilitem, na ocorrência de falhas, a utilização de caminhos alternativos. Um caminho alternativo, também chamado de desvio, parte do ponto em que a falha é conhecida até o destino final da mensagem. Além da avaliação de fluxo máximo, outros critérios de avaliação são utilizados, em particular o tamanho das rotas para sua seleção.

Um algoritmo é apresentado, que permite a cada nó, a partir da origem, determinar o próximo nó da rota entre os vizinhos, até que a mensagem chegue ao seu destino. Para a selecionar o próximo nó da rota de uma mensagem, o algoritmo avalia as arestas adjacentes ao nó em que é executado. Uma função de avaliação $\Gamma(G, e)$ é utilizada por um nó i que, tendo em vista sua representação interna da topologia G, retorna um valor numérico para a aresta e adjacente a i. A função $\Gamma(G, e)$ pode utilizar divesos critérios na avaliação, sendo possível definir um peso parametrizável para cada critério. Usando $\Gamma(G, e)$ cada nó tem uma estimativa de quão bom é um determinado enlace para o roteamento de uma mensagem. A aresta que apresenta o melhor compromisso entre esses critérios é selecionada para o roteamento.

Os pesos parametrizáveis permitem ajustar e enfatizar diversos critérios. Por exemplo, podem ser usados dois critérios: um para avaliar a conectividade de caminhos entre o nó correspondente à aresta avaliada e o nó de destino, representado pela cardinalidade do fluxo máximo; e um critério que avalia a distância entre esses nós, isto é, o

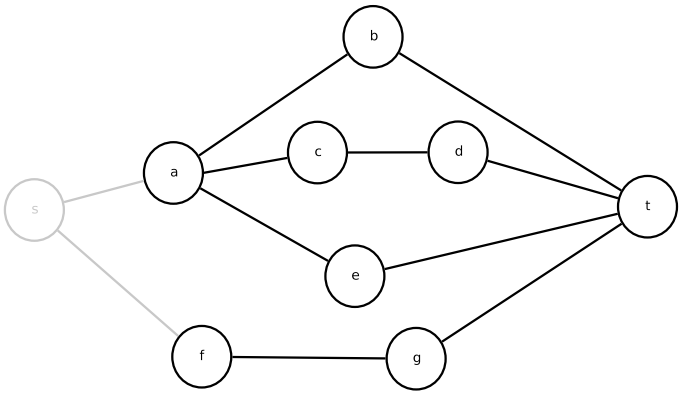


Figura 1. A representação do grafo utilizado para a avaliação das arestas iniciais em uma rede.

comprimento do menor caminho. Porém, é possível incluir outros critérios baseados em outras métricas como, por exemplo, o tráfego da rede. Desta forma, a função de avaliação pode ser representada por:

$$\Gamma(G, e) = \sum_{c_n \in N} w_n \times c_n(e)$$

Nessa expressão, N é o conjunto de critérios e w_n é o peso que está associado ao critério c_n. A aresta que possuir o maior valor para $\Gamma(G, e)$ será a aresta escolhida para o roteamento da mensagem.

Para exemplificar a avaliação das arestas adjacentes a um nó, através de critérios de avaliação, é apresentada a figura 1 que representa uma rede, tendo s como origem e t como destino.

Considerando que, inicialmente, o algoritmo esteja sendo executado no nó s e que s deva enviar uma mensagem a t; as arestas (s, a) e (s, f) são avaliadas de acordo com a função $\Gamma(G, e)$. Porém, o grafo utilizado na avaliação desconsidera dois conjuntos de arestas do grafo original: arestas percorridas pela mensagem em roteamento; e arestas adjacentes ao nó que executa o processo de roteamento. Desse modo, para avaliação das arestas (s, a) e (s, f) é utilizado o grafo formado pelas linhas mais escuras.

Para a avaliação da aresta $e = (s, a)$, usando a função $\Gamma(G, e)$, o nó s calcula os critérios de fluxo máximo e caminho mínimo considerando o nó a como origem e o nó t como destino. O cálculo do fluxo máximo, que é a principal contribuição desse algoritmo, é efetuado utilizado o algoritmo clássico de Ford-Fulkerson [Ford Jr. and Fulkerson 1962, Cormen et al. 2003]. Na avaliação do primeiro critério, fluxo máximo entre os nós avaliados e o destino, para a aresta (s, a) é encontrado um valor de fluxo máximo igual a 3, ou seja, existem três caminhos disjuntos do nó a ao nó t. Como o foco do algoritmo é a tolerância a falhas, são sugeridos valores positivos relativamente altos como peso para esse critério. O cálculo do segundo critério, comprimento do caminho mínimo, é efetuado utilizando o algoritmo de Dijkstra [Dijkstra 1959].

Na avaliação do segundo critério para a aresta (s, a) seria encontrado o valor 2, correspondendo ao comprimento do menor caminho entre o nó a e f. Nesse caso, basta considerar o primeiro caminho encontrado pela busca em largura, por exemplo (a, e, t).

Como quanto maior o comprimento do caminho mínimo, maior é o valor desse critério, o peso correspondente a esse critério deve ser negativo.

O algoritmo apresentado em [Schroeder and Duarte Jr 2006] desconsidera os caminhos disjuntos que utilizam nós já visitados por cada mensagem roteada para calcular o fluxo máximo efetivo. Dessa forma, esse processo de roteamento é dependente do contexto de *cada* mensagem, mais especificamente, do conjunto de nós visitados por cada mensagem a cada nó percorrido. O custo do algoritmo é de $O(E^2)$ para a escolha de cada aresta da rota, e portanto $O(E^3)$ para definir uma rota completa da origem ao destino. Esta ordem de complexidade por mensagem roteada não propicia uma implementação prática desse algoritmo para redes de alta velocidade. Neste trabalho é proposta uma heurística que mantém os benefícios do uso de roteamento baseado em avaliação de fluxo máximo que possa ser utilizada por protocolos na prática.

3. O Algoritmo Heurístico de Roteamento Robusto

Esta seção apresenta o algoritmo heurístico proposto para o roteamento robusto baseado em avaliação do fluxo máximo. A heurística utilizada permite a geração de estimativas da robustez dos caminhos da rede. Essas estimativas são baseadas em avaliação de fluxo máximo e são utilizadas para atualizar as tabelas de roteamento sempre que são detectadas alterações na topologia. O algoritmo efetua o encaminhamento de mensagens individuais consultando a tabela de roteamento para a escolha da saída de cada pacote. Após a especificação do algoritmo, são apresentados exemplos de funcionamento, bem como a prova formal de correção.

3.1. Especificação do Algoritmo Proposto

Um nó que executa o algoritmo mantém uma representação local da topologia da rede no grafo *KnownTopology*. Este grafo não precisa refletir todas as últimas alterações na rede, pois o algoritmo foi proposto justamente para viabilizar o roteamento nestes casos. Alterações na topologia são ou detectadas em arestas adjacentes ou informadas em mensagens de manutenção de topologia. Quando entra na rede, o nó dispara o procedimento de manuntenção da topologia. Esse procedimento é, a partir de então, executado periodicamente: a cada α segundos a tabela de roteamento é atualizada caso a descrição da topologia mantida no grafo $KnownTopology$ tenha sido alterada.

Um nó que recebe uma mensagem de atualização de topologia, deve propagar esta informação. Ao receber uma mensagem de atualização, se a aresta pela qual a mensagem foi recebida não está em $KnownTopology$, ela é incluída. O nó que recebe a mensagem deve verificar, em seu conteúdo, todas as informações recebidas sobre a alteração da topologia.

O procedimento usado para atualizar a tabela de roteamento, `GenerateRoutingTable`, é apresentado na figura 2. Como dito acima, caso a representação da topologia tenha sido alterada, esse procedimento é disparado pelo processo de manutenção da topologia. Esse procedimento gera a tabela de roteamento contendo todos os destinos possíveis. A heurística proposta para a atualização da tabela de roteamento inicia com o nó s (que executa o procedimento `GenerateRoutingTable`) inicialmente removendo a si próprio de sua representação interna da topologia. Assim, o nó calcula uma estimativa do fluxo máximo. Para cada destino (entrada da tabela),

```
GenerateRoutingTable(s)
 1. Let WorkingGraph ← s.KnownTopology − {s}
 2. For each destination node t in WorkingGraph
 3.         Let CandidateLinks ← an empty list to possible links out
 4.         For each link l adjacent to s in s.KnownTopology
 5.                 If l does not have a path to t
 6.                         Ignore l
 7.                 Else
 8.                         Add l to CandidateLinks
 9.         If CandidateLinks contains more than one link
10.                 Sort links in CandidateLinks according to the evaluation function
    Γ(WorkingGraph, l')
11.         Let RoutingTable[t] ← CandidateLinks
```

Figura 2. Especificação do algoritmo de roteamento: geração da tabela de roteamento.

é mantida uma lista de arestas candidatas ao roteamento, *CandidateLinks*. Essa lista é inicializada vazia. Uma aresta adjacente é incluída em *CandidateLinks* se tem um caminho ao destino final em *KnownTopology*. Ao final as arestas em *CandidateLinks* são ordenadas conforme a função de avaliação $\Gamma(WorkingGraph, e)$, apresentada na seção 2.

A tabela de roteamento gerada é usada como base para o procedimento de encaminhamento de mensagens (*Packet Forwarding*). Esse procedimento, apresentado na figura 3 e explicado a seguir, é responsável por consultar a tabela de roteamento, escolher a aresta candidata ao roteamento e encaminhar a mensagem por esta aresta.

O encaminhamento de mensagens é efetuado quando um nó s deseja enviar uma mensagem a um nó t, ou quando o nó s recebe uma mensagem de um nó x destinada um nó t. Quando o nó s recebe uma mensagem qualquer, é executado o procedimento `ReceiveMessage`. Esse procedimento é responsável por efetuar os seguintes passos: encaminhar a mensagem recebida ao nível de aplicação, caso seja o destino final da mensagem; encaminhar a mensagem ao procedimento de recebimento de mensagem de atualização, caso seja uma mensagem de atualização de topologia; enviar uma mensagem de confirmação de recebimento caso a mensagem recebida necessite de tal confirmação; encaminhar a mensagem ao procedimento de recebimento de confirmação de mensagens, caso seja uma mensagem de confirmação; e, finalmente, por encaminhar a mensagem ao procedimento `PacketForwarding`, caso a mensagem não tenha atingido seu destino final.

Considerando o caso acima, em que o nó s recebe uma mensagem de um nó x destinada a um nó t, o procedimento de encaminhamento efetua primeiramente uma verificação para detectar se a mensagem a ser roteada não está contida em um ciclo. Os ciclos são permitidos pelo algoritmo proposto quando o nó que efetua o roteamento não possui uma alternativa de rota, na qual o próximo nó não esteja na lista de nós visitados pela mensagem, até o destino final da mensagem. Caso seja necessária a criação de um ciclo, a mensagem é retornada ao nó imediatamente anterior no caminho percorrido. Caso a mensagem não esteja contida em um ciclo, é efetuada uma busca na tabela de roteamento pela aresta que possui maior resultado de avaliação e que ainda não tenha sido visitada. Se existir uma aresta que atenda a essas condições a mensagem é roteada para essa aresta, caso contrário é gerado um ciclo. Se o nó for o inicial, e não há alternativa de rota, é gerada uma mensagem de erro informando à aplicação que a mensagem não pode ser

```
PacketForwarding(msg, node)
 1. If msg.Dest is adjacent to node
 2.         Let l ← the link that goes directly from node node to msg.Dest
 3.         Send message msg through link l
 4. Else If msg.Visited does not contain the node node
 5.         Add node node to msg.Visited
 6.         If RoutingTable[msg.Dest] contains nodes that are not in msg.Visited
 7.                 Let l ← the link with largest RoutingTable[msg.Dest] whose destination is not in
    msg.Visited
 8.                 Send message msg through l
 9.         Else // Outdated topologies OR it is not a connected graph
10.                 If msg.Visited contains at least one node before node
11.                         Let l ← the link to the node before node in msg.Visited
12.                         Send msg back to l
13.                 Else
14.                         Return Error: There is no route to msg.Dest
15. Else // A cycle was detected
16.         If RoutingTable[msg.Dest] contains nodes that are not in msg.Visited
17.                 Let l ← the link with largest RoutingTable[msg.Dest] whose destination is not in
    msg.Visited
18.                 Send message msg through l
19.         Else // Outdated topologies OR it is not a connected graph
20.                 If msg.Visited contains at least one node before node
21.                         Let l ← the link to the node before node in msg.Visited
22.                         Send msg back to l
23.                 Else
24.                         Return Error: There is no route to msg.Dest
```

Figura 3. Especificação do algoritmo de roteamento: encaminhamento de mensagem.

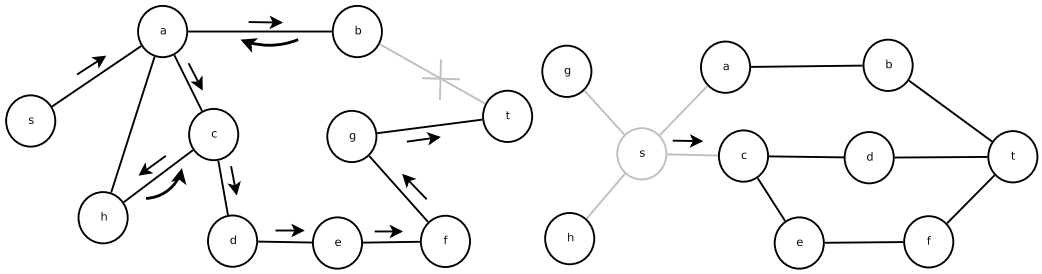
roteada.

Na ocorrência de um ciclo, são realizados os mesmos passos descritos acima, mas como o nó que gera o ciclo está marcado como visitado ele não será escolhido novamente para o roteamento. Ou seja, os ciclos apenas retrocedem ao nó imediatamente anterior para a tentativa de outros caminhos. Caso não seja encontrado nenhum caminho até o destino, a mensagem é retornada nó a nó, até que seja gerado um erro de falta de rota.

Para realizar esse processo são necessárias duas estruturas, uma para armazenar apenas o caminho principal percorrido pelas mensagens, e outra para gravar os nós visitados. Quando um nó gera um ciclo, ele deixa de pertencer ao caminho principal, sendo considerado apenas como um nó visitado. Com isso, o nó gerador do ciclo não é escolhido para roteamento novamente, e caso seja necessário que a mensagem retroceda mais um nó, esta será retrocedida para o nó imediatamente anterior no caminho principal de roteamento.

3.2. Exemplos de Utilização do Algoritmo Proposto

As figuras 4(a) e 4(b) apresentam exemplos de roteamento utilizando o algoritmo proposto em topologias simples. A figura 4(a) apresenta uma situação de falha no enlace (b, t), o qual seria utilizado para o roteamento da mensagem. No momento em que a mensagem considerada é enviada pelo nó s, apenas o nó b tem o conhecimento da falha. Dessa forma, inicialmente o nó s envia a mensagem para o nó a, pois é sua única alternativa de roteamento. Considerando que a tabela de roteamento apresenta o nó b como a alternativa que possui o maior valor para a função de avaliação, a mensagem é então roteada ao nó b. Porém, como o nó b não possui alternativas de roteamento que não utilizem nós já visitados pela mensagem, a mensagem é então retornada ao nó imediatamente anterior de sua rota percorrida, a.



(a) Uma rede com falha em uma de suas arestas. (b) A escolha de uma aresta em um ponto de corte.

Figura 4. Exemplos de roteamento.

Ao receber a mensagem novamente, a entrada da tabela que possui maior valor para função de avaliação, e que não tenha sido visitada previamente pela mensagem roteada é a entrada que indica o nó c como destino. Então, o nó a roteia a mensagem ao nó c. Nesse ponto o caminho principal percorrido pela mensagem é representado como $s- > a- > c$, sendo também o nó b marcado como visitado, porém não pertencente ao caminho principal.

Considerando que a tabela de roteamento do nó c apresenta para o destino t as alternativas a, h, d, ordenadas segundo a função de avaliação Γ, o nó c irá, inicialmente optar por a. Porém como o nó a já está marcado como visitado na mensagem roteada é então escolhido o nó h para o roteamento da mensagem. Vale destacar que a avaliação do nó h apresenta um valor maior que a do nó d. Pois, quando é desconsiderado o nó c e todas as suas arestas adjacentes para o cálculo da tabela em c, apesar da avaliação dos nós h e d possuírem o mesmo valor para o critério de fluxo máximo (um), o caminho mínimo para o nó t é menor por h ($h- > a- > b- > t$), dado que c desconhece a falha da aresta (b, t).

Da mesma forma que em b, ao chegar no nó h, como não existem alternativas para o roteamento que não tenham sido visitadas pela mensagem, esta é retornada ao nó imediatamente anterior da rota percorrida, c. Então, é considerado o nó d para o roteamento, pois é o único que não foi percorrido pela mensagem. A partir do nó d, é percorrido o caminho $e- > f- > g- > t$ até o destino final da mensagem.

A utilização do formato proposto para a tabela de roteamento apresenta várias situações satisfatórias ao roteamento, duas delas são apresentadas na figura 4(b), e descritas a seguir. Para o exemplo apresentado na figura 4(b), que representa uma situação de escolha de aresta em ponto de corte, ao calcular-se a tabela de roteamento do nó s, são considerados os nós a, c, g e h. Porém, como o nó s e todas suas arestas são desconsideradas durante o cálculo, os únicos nós que possibilitam o roteamento de mensagens ao nó t são os nós a e c e, portanto, são as únicas entradas inseridas na tabela de s para o destino t.

Como a função de avaliação retorna um valor melhor para o nó c, pois através desse nó é possível rotear uma mensagem ao nó t através de dois caminhos disjuntos, a aresta (s, c) é escolhida preferencialmente à aresta (s, a). Desse modo, a mensagem é roteada ao nó c. Na ocorrência de falhas na rede é clara a vantagem dessa escolha, pois, caso seja necessária a utilização de um desvio, a escolha do nó c aumenta a probabilidade

desse desvio ser menor.

Nesse caso, como a rede não apresenta falhas, a mensagem é roteada através do caminho $c- > d- > t$ até o seu destino final. Esse caminho, além de ser um dos caminhos mínimos possíveis, é o que apresenta maior probabilidade de utilização de um desvio mais curto, na ocorrência de falhas na rede.

3.3. Prova de Correção do Algoritmo

A prova de correção consiste do **Lema 4.1** e do **Teorema 4.2**. Assume-se que não há mudança de topologia da rede durante o roteamento de uma mensagem. Para garantir a correção do algoritmo uma condição suficiente (não necessária) é 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. Essa condição é suficiente, porém não é necessária para o funcionamento do algoritmo. Uma mensagem pode ser roteada mesmo que o nó de origem não conheça nenhum caminho válido até o destino.

Lema 4.1. Considere $G = (V, E)$ um grafo, e $s, t \in V$ dois nós não falhos desse grafo. Considere que para uma ordenação das arestas candidatas, listadas na entrada correspondente ao destino t das tabelas de roteamento de cada nó, o algoritmo entrega a mensagem no destino. Se for alterada a ordenação das arestas candidatas ao roteamento nas tabelas dos nós do grafo G, e s enviar uma mensagem para t utilizando o algoritmo proposto nesse trabalho, a mensagem chegará até t.

Prova. Esta prova considera, sem perda de generalidade, um nó x pertencente a um caminho válido da origem ao destino, $x \neq t$. Assume-se que a tabela de roteamento de x, para a entrada de destino t, apresente a seguinte lista de arestas candidatas ao roteamento $e_1, e_2, ..., e_{n-1}$. Existe ao menos uma aresta nesse conjunto que leva a um nó pertencente a um caminho válido para o qual a mensagem deve ser enviada.

Como base da indução, é assumido que o conjunto de arestas candidatas contém apenas um elemento. Neste caso, a prova de que a alteração da ordem de escolha das arestas candidatas não afeta o roteamento é trivial, pois o conjunto continua ordenado da mesma maneira. Como hipótese de indução prova-se que se o lema é verdadeiro para a ordenação de arestas $e_1, e_2, ..., e_{n-1}$ é também verdadeiro para $e_1, e_2, ..., e_{n-1}, e_n$.

Assume-se, novamente sem perda de generalidade, que a aresta escolhida para o roteamento é a aresta e_n. Portanto, a mensagem será enviada para ao outro nó adjacente à aresta, chamado de u. Se o nó u fizer parte de um caminho válido para o destino t está provado que o lema é válido para o novo conjunto de arestas. Caso contrário, após o nó u tentar enviar a mensagem para todas as alternativas possíveis de roteamento que não tenham passado por nós visitados pela mensagem, a mensagem é retornada ao nó x e a aresta e_n, conforme o algoritmo, não será utilizada novamente para o roteamento. Portanto, o conjunto de arestas disponíveis será $e_1, e_2, ..., e_{n-1}$, para o qual o lema é verdadeiro. Desta forma o roteamento é realizado com sucesso caso seja alterada a ordenação das arestas candidatas nas tabelas de roteamento. □

Teorema 4.2. Considere $G = (V, E)$ um grafo, e $s, t \in V$ dois nós não falhos desse grafo. Considere que os nós possuem conhecimento do estado de seus nós vizinhos.

Considere, também, que o nó s conhece pelo menos um caminho para o destino t que esteja sem-falhas. Se s enviar uma mensagem para t utilizando o algoritmo proposto neste trabalho, a mensagem chegará até t.

Prova. Como base para a indução, assume-se que o grafo G possui dois nós. Esses nós correspondem aos nós s e t que, por definição são diferentes. A única aresta candidata ao roteamento presente na tabela de roteamento do nó s é a aresta (s, t), que não apresenta falha, portanto o roteamento tem sucesso.

Assumindo como hipótese para a indução que o teorema é válido para um grafo com $(n - 1)$ nós, deve-se provar que o teorema também é válido para um grafo com n nós. Considerando, então, que o grafo H possui $(n - 1)$ nós e o teorema é satisfeito para esse grafo conforme a hipótese anterior. Suponha que o grafo I seja um grafo resultante da inclusão de um nó n e da inclusão de até $(n - 1)$ arestas, conectadas ao nó n, no grafo H.

Considerando, sem perda de generalidade, dois nós s e t comuns aos grafos H e I, suponha que o nó s envia uma mensagem destinada ao nó t. Nessa situação, existem dois casos iniciais $CASO1, CASO2$ que precisam ser provados para aceitar o teorema como válido para qualquer número de nós. No primeiro, a inclusão do nó n e das arestas que conectam esse nó aos demais nós do grafo I não interfere, mesmo considerando possíveis alterações nas tabelas de roteamento geradas nos nós, no envio da mensagem do nó s ao nó t. Nesse caso o teorema é satisfeito de acordo com a hipótese.

No segundo caso possível ($CASO2$), a inclusão do nó n, e suas arestas adjacentes, interfere no envio da mensagem do nó s ao nó t. Vale ressaltar que essa inclusão pode apenas alterar a ordenação das arestas candidatas nas tabelas de roteamento e/ou incluir uma aresta que aponte para o nó n. Nesse caso, existem algumas possibilidades listadas a seguir.

A primeira possibilidade ($CASO2.1$), é que essa inclusão tenha apenas alterado a ordenação das arestas candidatas e que o nó n não tenha sido selecionado para rotear a mensagem. Nesse caso, pelo lema 4.1, está provado teorema.

Outro caso possível ($CASO2.2$), é o nó n ser selecionado para rotear a mensagem. Nesse ponto, caso o nó n roteie a mensagem para um nó pertencente a um caminho válido, pelo algoritmo, o nó n será marcado como visitado e o roteamento considera os demais $(n - 1)$ nós. Caso em que, pelo lema 4.1, o teorema é provado, pois o nó n é marcado como visitado e não é mais selecionado para roteamento. Caso o nó roteie a mensagem apenas para nós que não façam parte de um caminho válido, que não inclua nós visitados anteriormente pela mensagem, a mensagem, esta retornará ao nó n até que este fique sem alternativas de roteamento. Nesse ponto, o nó n retorna a mensagem ao nó que a enviou e, como é marcado como nó visitado, não será mais escolhido para o roteamento da mensagem. Portanto, o roteamento continua com, no máximo, $(n - 1)$ nós, caso em que o teorema é válido, pois mesmo que tenha sido mudada a ordenação das arestas candidatas, o lema 4.1 prova que o algoritmo continua sendo válido. □

4. Resultados Experimentais

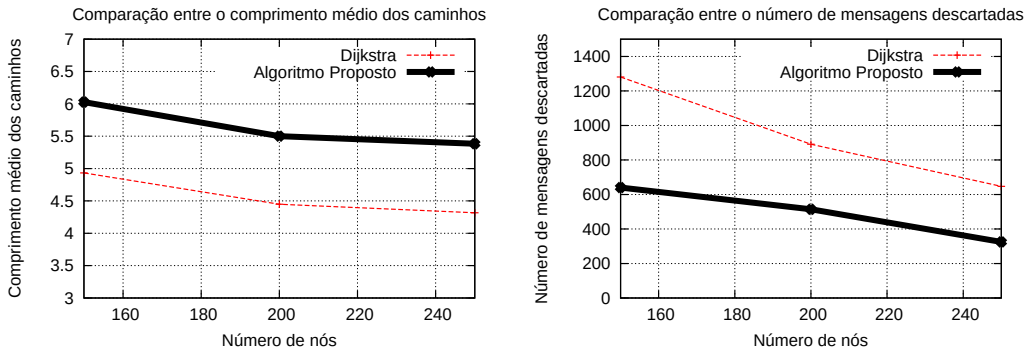
Nesta seção, são apresentados resultados experimentais obtidos através de simulação. O algoritmo proposto foi comparado com o algoritmo de Dijkstra [Dijkstra 1959] e também

ao algoritmo apresentado na seção 2. Para validar a representatividade dos resultados obtidos, as simulações foram efetuadas de modo a garantir que essa probabilidade esteja dentro de limites aceitáveis (a meia amplitude dos intervalos de confiança de 95% e menor que 10% da média dos valores obtidos).

A comparação com o algoritmo de Dijkstra considerou o comprimento médio de caminhos percorridos por mensagens entregues e número de mensagens descartadas. Em seguida, na comparação com o algoritmo descrito na seção 2 são considerados o comprimento médio dos caminhos percorridos com e sem *backtracking*; o número de mensagens descartadas e o custo do cálculos de avaliação de rotas.

O principal objetivo foi a avaliação dos algoritmos em situações dinâmicas de funcionamento, em que falhas ocorrem por períodos relativamente curtos de 10 segundos. Esses períodos são curtos o suficiente para não serem percebidos nas atualizações normais de topologia. Produzindo-se falhas por períodos curtos, consegue-se apurar principalmente as situações de interesse para avaliar a capacidade do algoritmo proposto de contornar as falhas encontradas no percurso das mensagens. Desta forma os nós da rede não tem tempo para atualizar suas representações locais da topologia. O algoritmo pode ser avaliado em circunstâncias em que é necessário. Foi utilizado um intervalo de envio de mensagens de 50 milissegundos, garantindo a geração de 200 mensagens aleatórias durante a ocorrência de cada falha.

A figura 5(a) apresenta o comprimento médio dos caminhos percorridos por mensagens entregues em redes de 150, 200 e 250 nós. O algoritmo proposto é comparado com o algoritmo de Dijkstra. Pode-se observar que uma rota do algoritmo proposto é em média cerca de um nó maior que a rota correspondente do algoritmo de Dijkstra.



(a) Comprimento médio das rotas. (b) Número de mensagens descartadas.

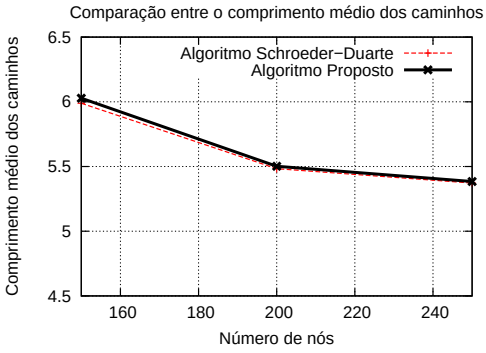
Figura 5. O algoritmo proposto vs. Dijkstra.

Observa-se, ainda, que o algoritmo proposto inclui mensagens entregues que contornaram falhas ocorridas na rede e que percorreram um caminho maior por esse motivo. Essas mensagens foram descartadas pelo algoritmo de roteamento baseado no algoritmo de Dijkstra. O número exato de mensagens descartadas pode ser observado na figura 5(b). Esta figura compara o número de mensagens descartadas na execução do algoritmo proposto e do roteamento baseado no algoritmo de Dijkstra.

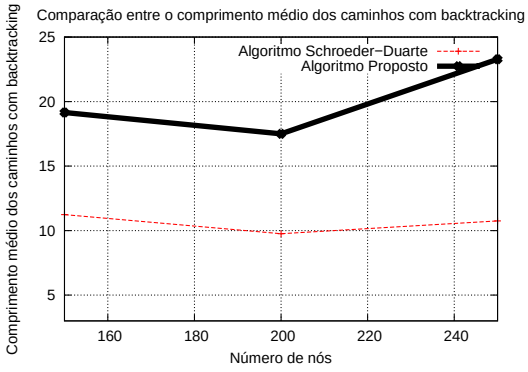
Nota-se que o algoritmo proposto reduziu significativamente o número de mensagens descartadas. De fato, nos casos avaliados, o algoritmo proposto descartou apenas

mensagens que não tinham possibilidade de roteamento, devido ao particionamento das redes pelas falhas inseridas na simulação.

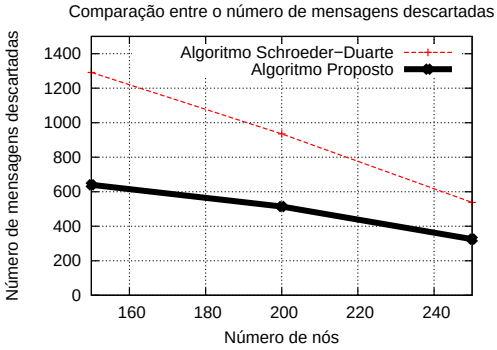
Quando comparado ao algoritmo de [Schroeder and Duarte Jr 2006], o algoritmo proposto quase não apresenta diferença quando se avalia o comprimento médio dos caminhos percorridos por mensagens. A figura 6(a) apresenta essa diferença para redes de 150, 200 e 250 nós. Porém, de acordo com a figura 6(b), pode-se constatar uma diferença maior no comprimento médio de caminhos de mensagens que efetuaram o *backtracking*. Essa diferença deve-se ao fato do algoritmo de [Schroeder and Duarte Jr 2006] enviar uma mensagem de atualização de topologia ao nó imediatamente anterior, nos casos em que é necessário efetuar o *backtracking* de uma mensagem. Isto faz com que o nó anterior atualize a sua representação da topologia da rede e, portanto, roteie a mensagem por um caminho mais adequado.



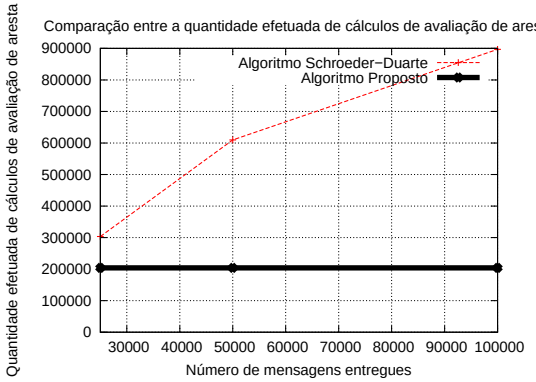
(a) Comprimento médio das rotas.



(b) Comprimento médio das rotas com *backtracking*.



(c) Número de mensagens descartadas.



(d) Custo do algoritmo.

Figura 6. O algoritmo proposto vs. [Schroeder and Duarte Jr 2006].

A característica de adiantar as mensagens de atualização de topologia, em casos de descoberta de falhas na rede, pode ser prejudicial quando o elemento falho retorna ao seu estado normal. Este fato pode ser observado na figura 6(c), que compara a quantidade de mensagens descartadas pelos algoritmo de [Schroeder and Duarte Jr 2006] e pelo algoritmo proposto. Ao retomar o seu estado de funcionamento normal, as mensagens somente serão roteadas a esses elementos quando as topologias dos demais nós pertencentes ao caminho normal da mensagem forem atualizadas. No pior caso, esse tempo de convergência está vinculado ao diâmetro da rede em questão.

Uma comparação do custo de processamento do algoritmo proposto e de [Schroeder and Duarte Jr 2006] pode ser observada na figura 6(d). Para um número pequeno de mensagens o algoritmo de [Schroeder and Duarte Jr 2006] efetua um processamento inferior ao processamento imposto pelo algoritmo proposto. Porém, os valores logo invertem com o aumento do número de mensagens.

5. Trabalhos Relacionados

Diversas estratégias relacionadas têm sido propostas no contexto dos protocolos de roteamento na Internet. Pei *et. al.* [Pei et al. 2005] propõem um novo mecanismo, chamado BGP-RCN (*BGP with Root Cause Notification*, que limita superiormente o tempo de convergência do BGP em $O(d)$, onde d é o diâmetro da rede medido pelo número de *hops* de AS. Com o mesmo intuito de diminuir o tempo de convergência do BGP, Bremler-Barr *et. al.* [Bremler-Barr et al.] sugerem uma pequena mudança no protocolo, permitindo a geração e propagação "mensagens de retirada" (*withdrawal messages*), mensagens que informam falhas na rede e são propagadas rapidamente.

Echenique *et. al.* [Echenique et al. 2004] sugerem o uso de algoritmos que apresentam um melhor desempenho quando comparados ao algoritmo padrão de roteamento entre AS's. Esses algoritmos utilizam tanto informações da topologia quanto informações de tráfego da rede. Outro trabalho que leva em consideração o tráfego da rede é proposto por Yan *et. al.* [Yan et al. 2006], que propõe uma estratégia para melhorar a eficiência do transporte em redes complexas.

Existe um conjunto emergente de tecnologias, denominado *IP Fast Reroute (IP-FRR)* [Gjoka et al.], que tem por objetivo efetuar o reparo através do re-roteamento do tráfego através de alternativas pré-computadas. Kvalbein *et. al.* [Kvalbein et al. 2006] apresentam um novo esquema para o tratamento de falhas em redes IP. A idéia central desse esquema, chamado de *Multiple Routing Configurations (MRC)*, consiste em utilizar a topologia conhecida da rede e construir um conjunto de configurações de roteamento. Cicic *et. al.* [Cicic et al. 2008] apresentam um mecanismo aprimorado para o re-roteamento IP (*IP Fast Rerouting*). Esse trabalho aprimora o no trabalho proposto em [Kvalbein et al. 2006].

6. Conclusão

Neste trabalho foi proposto um algoritmo heurístico de roteamento cujo objetivo é o de possibilitar a implementação prática de um protocolo de roteamento que utiliza a avaliação de fluxo máximo como um dos critérios de escolha de rotas. O algoritmo utiliza tabelas de roteamento, permitindo que mensagens sejam encaminhas de forma eficiente, com baixa complexidade. Além da especificação e descrição do algoritmo, foi apresentada uma prova formal de sua correção. Foram também apresentados resultados experimentais comparativos, mostrando que o algoritmo resulta em baixa perda mensagens, com rotas de tamanho próximo ao mínimo.

Trabalhos futuros contemplam a utilização de outros critérios, como tráfego e parâmetros de qualidade de serviço, além do fluxo máximo e distância. A elaboração de um protocolo baseado no algoritmo proposto também é prevista como trabalho futuro.

Referências

(2009). Resilient overlay networks. `http://nms.csail.mit.edu/ron/`.

Bremler-Barr, A., Afek, Y., and Schwarz, S. Improved BGP convergence via ghost flushing. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2.

Campisano, A., Cittadini, L., Battista, G., Refice, T., and Sasso, C. (2008). Tracking Back The Root Cause of a Path Change in Interdomain Routing. *The 11th IEEE/IFIP Network Operations and Management Symposium (NOMS)*.

Cicic, T., Hansen, A., Kvalbein, A., Hartman, M., Martin, R., and Menth, M. (2008). Relaxed Multiple Routing Configurations for IP Fast Reroute. *NOMS 2008*, pages 457–464.

Cicik, T., Kvalbein, A., Hansen, A. F., Gjessing, S., and Lysne, O. (2009). Multiple Routing Configurations for Fast IP Network Recovery. *IEEE/ACM Transactions on Networking*.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2003). *Introduction to Algorithms, 2nd Ed.* McGraw-Hill.

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

Echenique, P., Gomez-Gardenes, J., and Moreno, Y. (2004). Improved routing strategies for Internet traffic delivery. *Physical Review E*, 70(5):56105.

Ford Jr., L. R. and Fulkerson, D. R. (1962). Flows in networks. *Princeton University Press*.

Gjoka, M., Ram, V., and Yang, X. Evaluation of IP Fast Reroute Proposals. *IEEE International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE)*.

Kvalbein, A., Hansen, A. F., Cicic, T., Gjessing, S., and Lysne, O. (2006). Fast IP Network Recovery Using Multiple Routing Configurations. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11.

Ohara, Y., Imahori, S., and V., M. R. (2009). MARA: Maximum Alternative Routing Algorithm. *The 28th IEEE Conference on Computer Communications (INFOCOM)*.

Pei, D., Azuma, M., Massey, D., and Zhang, L. (2005). BGP-RCN: Improving BGP convergence through root cause notification. *Computer Networks*, 48(2):175–194.

Schroeder, J. and Duarte Jr, E. P. (2006). Roteamento Dinâmico Tolerante a Falhas Baseado em Avaliação de Fluxo Máximo. *Workshop de Testes e Tolerância a Falhas (WTF'2006), Anais do SBRC'2006*, pages 197–207.

Wang, F. and Gao, L. (2009a). IP Fast Reroute Using Not-via Addresses. *IETF Draft*.

Wang, F. and Gao, L. (2009b). Path Diversity Aware Interdomain Routing. *The 28th IEEE Conference on Computer Communications (INFOCOM)*.

Yan, G., Zhou, T., Hu, B., Fu, Z., and Wang, B. (2006). Efficient routing on complex networks. *Physical Review E*, 73(4):46108.