RemapRoute: Reduzindo o custo do remapeamento de mudanças de roteamento na Internet

Ítalo Cunha[1] **Renata Teixeira**[2] **Darryl Veitch**[3] **Christophe Diot**[4]

[1]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais

[2]UPMC Sorbonne Universités & conseil Nationale de la Recherche Cientifique

[3]Department of Electrical and Electronic Engineering, University of Melbourne

[4]Technicolor

cunha@dcc.ufmg.br, renata.teixeira@lip6.fr, dveitch@unimelb.edu.au, christophe.diot@technicolor.com

Abstract. *Internet topology maps collected with traceroute may be incomplete or out-of-date because we cannot measure frequently enough to detect all routing changes without overloading the network. In this paper, we show that routing changes usually affect few routers. We exploit this property to design RemapRoute, a tool to locally remap Internet routing changes that probes only the few affected routers instead of the whole route. Our evaluation with trace-driven simulations and a deployment shows that RemapRoute significantly reduces the number of probes needed to remap routes, with no impact on remapping accuracy or latency. This reduction in remapping cost allows us to build more complete and up-to-date topology maps.*

Resumo. *Mapas topológicos da Internet coletados com traceroute podem estar incompletos ou desatualizados porque não podemos realizar medições em frequência suficiente para detectar todas as mudanças de rota. Neste artigo mostramos que mudanças de rota geralmente afetam poucos roteadores. Exploramos essa propriedade no RemapRoute, nossa ferramenta para remapeamento local de mudanças de roteamento que sonda apenas os poucos roteadores afetados ao invés de toda a rota. Nossa avaliação via simulação e um protótipo real mostra que RemapRoute reduz significativamente o número de sondas de remapeamento, sem comprometer a exatidão e a latência de remapeamento. Esta redução do custo de remapeamento nos propicia construir mapas topológicos mais completos e atualizados.*

1. Introdução

Sistemas para identificação de falhas na Internet coletam medições frequentes de rotas na rede [Duffield 2006, Dhamdhere et al. 2007, Kompella et al. 2007, Katz-Bassett et al. 2012]. De forma similar, redes de distribuição de conteúdo medem rotas na Internet para escolher o melhor servidor para atender uma requisição [Dilley et al. 2002]. Esses e outros sistemas medem rotas frequentemente na esperança de rastrear mudanças de roteamento à medida que elas acontecem.

Medições de rota na Internet são frequentemente coletadas com traceroute [Jacobson 1989, Augustin et al. 2007, Veitch et al. 2009], que envia sondas para identificar

uma sequência de roteadores entre uma origem e um destino. A banda disponível para enviar sondas é finita. Medir rotas para mapear a topologia requer um grande número de sondas e pode levar de vários minutos a alguns dias [Cunha et al. 2011a, Sherwood et al. 2008, Claffy et al. 2009]. É impossível medir com frequência suficiente para detectar todas as mudanças de roteamento sem sobrecarregar a rede. Consequentemente, mapas da topologia da Internet podem estar desatualizados ou inconsistentes pois mudanças de roteamento podem acontecer durante o processo de medição.

Nosso sistema DTRACK rastreia mudanças de roteamento na Internet para manter mapas da topologia da Internet mais atualizados [Cunha et al. 2011b]. O DTRACK separa as tarefas de detectar e remapear mudanças de roteamento. Para detectar mudanças, o DTRACK usa um processo de sondagem leve que combina duas ideias: (1) redirecionar sondas de caminhos estáveis onde mudanças de roteamento são improváveis para caminhos instáveis onde mudanças são mais prováveis; e (2) espalhar sondas de forma uniforme na rede para reduzir medições redundantes. Para remapear mudanças, o DTRACK usa o Paris traceroute [Augustin et al. 2007, Veitch et al. 2009] para medir a nova rota por inteiro. O Paris traceroute é uma versão moderna do traceroute capaz de identificar roteadores que fazem balanceamento de carga. Usamos o Paris traceroute por que é impossível inferir mudanças de roteamento de forma precisa sem informação sobre roteadores que fazem balanceamento de carga [Cunha et al. 2011a].

O DTRACK mantém um banco de dados com a última rota observada em cada um dos caminhos monitorados. Para detectar mudanças, o DTRACK envia uma sonda num ponto s' do caminho e compara a resposta da sonda com a última rota observada. Se a resposta da sonda for incompatível com a última rota observada, e.g., o endereço IP do roteador que enviou a resposta não pertence à última rota observada, uma mudança é detectada e o processo de remapeamento é disparado. Atualmente, o DTRACK remapeia o caminho por inteiro usando o Paris traceroute. Esta abordagem garante medição correta da nova rota, mas desperdiça várias sondas pois mudanças de caminho envolvem poucos roteadores (seção 3). Em particular, esta abordagem ignora duas informações disponíveis quando o processo de remapeamento é disparado: a última rota observada e o ponto s' onde a mudança foi detectada.

Neste artigo propomos RemapRoute, uma ferramenta para reduzir o custo do remapeamento de mudanças de roteamento na Internet (seção 4). Dadas a última rota observada e um ponto onde uma mudança foi detectada, o RemapRoute envia sondas em pontos estratégicos para localizar a mudança e remapeá-la localmente, sem desperdiçar sondas nos roteadores que não estão envolvidos na mudança. Nossa avaliação via simulação dirigida por dados reais mostra que RemapRoute reduz pela metade o custo do remapeamento de 88% das mudanças de roteamento em nossos dados (seção 5). A redução de custo é ainda maior em rotas longas ou com roteadores que fazem balanceamento de carga. Nossa avaliação de um protótipo do RemapRoute usando o PlanetLab confirma nossos resultados via simulação e demonstra a eficácia da ferramenta (seção 6). Sumarizando, neste artigo fazemos as seguintes contribuições:

- Caracterizamos mudanças de roteamento na Internet e mostramos que elas envolvem uma fração pequena dos roteadores numa rota (seção 3);
- Propomos métodos para localizar e remapear mudanças de roteamento que reduzem o desperdício de sondas (seção 4);

- Mostramos que nossa ferramenta reduz o custo de remapeamento de mudanças de rota via simulação e em cenários reais (seções 5 e 6).

A economia de sondas no processo de remapeamento de mudanças de roteamento aumenta o número de sondas disponíveis para mapeamento topológico. Podemos utilizar estas sondas para monitorar mais caminhos na Internet e melhorar a cobertura dos mapas da topologia, ou aumentar a frequência de sondagem e melhorar o rastreamento de mudanças de roteamento. RemapRoute é mais um passo na construção de mapas da topologia da Internet mais completos e consistentes.

2. Definições e fundamentos

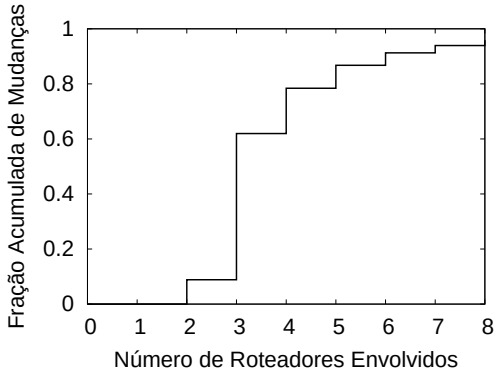
Seguindo a nomenclatura proposta por Paxson [Paxson 1997], chamamos de *caminho virtual* a conectividade entre uma origem e um destino na Internet. Em um dado momento, um caminho virtual é instanciado por uma *rota*. Devido a mudanças de roteamento, um caminho pode ser visto como um processo contínuo $C(t)$ que muda de uma rota a outra ao longo do tempo. Uma rota é composta de *saltos* (*hops*) que são instanciados por roteadores. Saltos são enumerados a partir da origem e nos referimos a um salto s numa rota $C(t)$ por $C(t)[s]$. Uma rota pode ser *simples* se ela tem apenas uma sequência de roteadores da origem ao destino; ou *ramificada* se ela tem roteadores que realizam balanceamento de carga e múltiplas sequências sobrepostas de roteadores da origem ao destino. Todos os saltos numa rota simples têm apenas um roteador e pelo menos um salto numa rota ramificada tem mais de um roteador.

Dadas duas medições consecutivas de um caminho nos instantes $C(t_i)$ e $C(t_{i+1})$, definimos uma *mudança de caminho* como uma sequência de saltos contíguos no novo caminho que altera o caminho antigo. Computamos mudanças de caminho minimizando o número de edições (adição, remoção e substituição de saltos) necessárias para transformar a nova rota na rota antiga. Definimos o *salto de divergência* s_d e o *salto de convergência* s_c de uma mudança como os saltos imediatamente anterior e posterior aos saltos editados pela mudança, respectivamente. Dizemos que o salto de divergência, o salto de convergência e todos os saltos entre eles estão *envolvidos* na mudança de roteamento. Exemplificando, se $C(t_i) = \{a, b, c, d, e, f, g\}$ e $C(t_{i+1}) = \{a, b, e, x, y, g\}$, temos uma mudança com $s_d = 1$ e $s_c = 2$ (remoção de c e d), e outra mudança com $s_d = 2$ e $s_c = 5$ (troca de f por x e y).

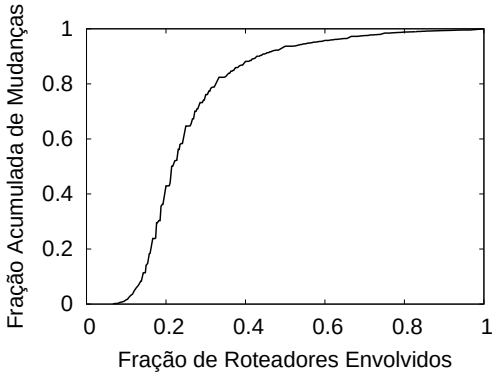
3. Caracterização de mudanças de caminhos

Nesta seção caracterizamos um conjunto de mudanças de caminho reais e verificamos que mudanças de caminho na Internet afetam poucos roteadores.

Implantamos o DTRACK [Cunha et al. 2011b] para medir caminhos em 72 monitores no PlanetLab por uma semana a partir de 4 de março de 2011. Cada monitor escolhe 1.000 destinos aleatoriamente de uma lista com 34.820 destinos alcançáveis escolhidos aleatoriamente na Internet. Em cada monitor, configuramos o DTRACK para rastrear mudanças nos caminhos escolhidos enviando 8 sondas por segundo, similar à taxa de sondagem do DIMES [Shavitt and Weinsberg 2009]. Em uma semana observamos 1.202.960 mudanças de caminho. Os caminhos medidos atravessam 7.315 sistemas autônomos e 97% dos sistemas autônomos de grande porte [Oliveira et al. 2010].



(a) Distribuição do número roteadores envolvidos em mudanças de caminho



(b) Distribuição da fração de roteadores de uma rota envolvidos em mudanças de caminho

Figura 1. Caracterização de mudanças de caminho na Internet

A figura 1(a) mostra a distribuição do número de saltos envolvidos em mudanças de caminho na Internet. O número de saltos envolvidos numa mudança de caminho é o mínimo de saltos que precisamos remapear para saber qual é a nova rota. Vemos que 78% das mudanças de caminho afetam quatro ou menos saltos, um número pequeno visto que a mediana do tamanho das rotas em nossos dados é 17 saltos. O tipo de mudança de caminho mais comum é quando o roteador de apenas um salto muda, resultando em mudanças que envolvem três saltos: o salto onde o roteador mudou mais os saltos de convergência e divergência. Notamos que 9% das mudanças de caminho envolvem dois saltos. Estas mudanças apenas removem saltos da rota antiga (a nova rota está contida na rota antiga) e os únicos saltos envolvidos são os saltos de convergência e divergência. Causas típicas de mudanças de caminho que envolvem dois saltos são falhas de conectividade na Internet e erros de medição onde é impossível medir os saltos atrás da falha ou erro.

A figura 1(b) mostra a distribuição da fração de saltos de um caminho envolvidos numa mudança, i.e., o número de roteadores envolvidos na mudança dividido pelo tamanho da nova rota, para todas as mudanças nos nossos dados. Vemos que 76% das mudanças de caminho envolvem menos de 30% dos saltos no caminho. Isso mostra o potencial do remapeamento local para redução de sondas de medição, se comparado com a abordagem atual de remapear o caminho por inteiro. A curva começa em $x = 0.066 = 2/30$ acontece porque uma mudança envolve pelo menos dois saltos e porque o DTRACK só mede os 30 primeiros saltos num caminho (o padrão do traceroute e do Paris traceroute [Jacobson 1989, Augustin et al. 2007]).

A figura 2 mostra a distribuição do número de sistemas autônomos envolvidos em mudanças de caminho. Convertemos endereços IPs medidos pelo DTRACK em sistemas autônomos combinando as tabelas de mapeamento do Team Cymru[1] e iPlane [Madhyastha et al. 2006]. Cada endereço IP que não aparece na tabela de mapeamento combinada é associado a um sistema autônomo que contém apenas um endereço IP. Essa é uma medida conservadora que pode sobrestimar o número de sistemas autônomos envolvidos em uma mudança. A figura 2 explica porque mudanças de roteamento envolvem poucos saltos. Mesmo fazendo a conversão de endereços IP para sistemas autônomos de forma conservadora, vemos que 60% das mudanças de caminho é interna a um sistema autônomo e apenas 7% envolve mais de dois sistemas autônomos.

[1]Team Cymru, IP to ASN Mapping, http://www.team-cymru.org/Services/ip-to-asn.html

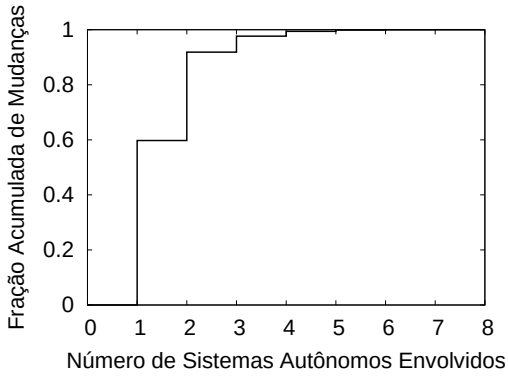


Figura 2. Distribuição do número de sistemas autônomos envolvidos em mudanças de caminho

4. REMAPROUTE

Agora apresentamos nosso algoritmo para remapeamento local de mudanças de caminho implementado no RemapRoute. O algoritmo recebe como entrada a rota antiga observada antes da mudança de roteamento $C(t_{i-1})$ e o salto s' do roteador onde a mudança foi detectada, $C(t_i)[s'] \neq C(t_{i-1})[s']$. Para remapear a mudança de rota, RemapRoute opera em duas etapas: primeiro ele localiza onde a mudança aconteceu (seção 4.1) e depois faz o remapeamento local da mudança (seção 4.2).

O processo de localização e remapeamento da mudança de rota envolve remapear saltos da rota atual e compará-los com saltos na rota antiga. Para remapear cada salto, o RemapRoute envia múltiplas sondas, modificando sistematicamente os campos do cabeçalho IP como o Paris traceroute [Augustin et al. 2007, Veitch et al. 2009], para identificar roteadores que fazem balanceamento de carga.

4.1. Localização da mudança de roteamento

O RemapRoute parte de um salto s' onde o roteador na rota atual é diferente do roteador na rota antiga, $C(t_i)[s'] \neq C(t_{i-1})[s']$. Se o roteador da rota atual no salto s' não pertencer à rota antiga, $C(t_i)[s'] \notin C(t_{i-1})$, então s' é um dos saltos envolvidos na mudança de caminho e podemos passar para a próxima etapa para remapear a mudança (seção 4.2).

Se o roteador da rota atual no salto s' pertencer à rota antiga em outro salto s'', $C(t_i)[s'] = C(t_{i-1})[s'']$, então temos uma mudança de roteamento em saltos anteriores a s' que adicionou ou removeu roteadores na rota. A figura 3 mostra um exemplo de uma mudança de roteamento onde o roteador d foi substituído por dois roteadores x e y. Uma sonda para o sétimo salto detecta uma mudança de roteamento pois a resposta da sonda vem do roteador $g = C(t_i)[7]$, que não é a resposta esperada na rota antiga, $h = C(t_{i-1})[7]$.

Para encontrar um roteador envolvido na mudança de caminho, i.e., um roteador na rota atual que não está presente na rota antiga, o RemapRoute faz uma busca binária no caminho. O RemapRoute inicializa $s_{\text{esq}} = 0$ e $s_{\text{dir}} = s'$. A cada iteração da busca, o RemapRoute remapeia o salto $s = (s_{\text{esq}} + s_{\text{dir}})/2$ e procura o roteador no salto s da rota atual na rota antiga. Se o roteador no salto s da rota atual não pertencer à rota antiga, $C(t_i)[s] \notin C(t_{i-1})$, a busca termina e passamos para a etapa de remapeamento. Se o

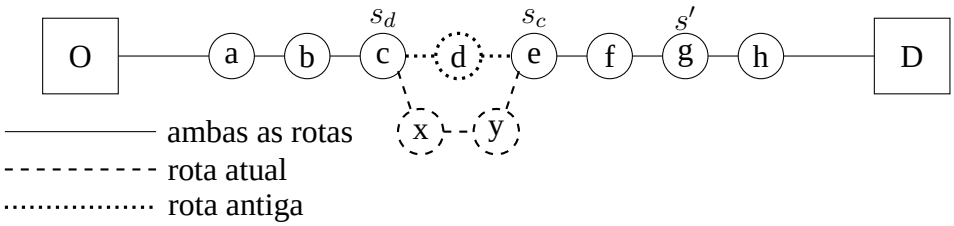


Figura 3. Exemplo de mudança de roteamento com adição de roteadores à rota.

roteador no salto s da rota atual estiver no salto s da rota antiga, $C(t_i)[s] = C(t_{i-1})[s]$, então a mudança está à direita de s e fazemos $s_{\text{esq}} = s$. Se o roteador no salto s da rota atual estiver em outro salto s'' na rota antiga, $C(t_i)[s] = C(t_{i-1})[s'']$, a mudança está à esquerda de s e fazemos $s_{\text{dir}} = s$.

Não temos como comparar a rota atual com a rota antiga se o roteador no salto s não responde a sondas. Neste caso tomamos a decisão conservadora de decrementar s e continuar a busca no salto anterior, sem alterar s_{esq} e s_{dir}.

Se uma mudança de caminho apenas remove saltos da rota antiga, então não existe salto na rota atual que não pertença à rota antiga. Neste caso, o processo de busca termina com $s_{\text{esq}} = s_{\text{dir}} + 1$ e nenhum remapeamento é necessário (seção 4.2).

4.2. Remapeamento local

O remapeamento local parte de um salto s onde o roteador na rota atual não pertence à rota antiga, $C(t_i)[s] \notin C(t_{i-1})$. O RemapRoute remapeia sequencialmente os saltos da rota atual posteriores a s até encontrar o salto de convergência $s_c > s$ com um roteador que pertence à rota antiga, $C(t_i)[s_c] \in C(t_{i-1})$. Se uma das rotas não chega até o destino, o salto de convergência pode não existir. Neste caso, o RemapRoute remapeia a rota atual até o destino ou até o último salto alcançável. Se o caminho não chega até o destino, o RemapRoute identifica o último salto alcançável após encontrar três saltos consecutivos que não respondem a sondas (igual o traceroute).

De forma similar, o RemapRoute remapeia sequencialmente os saltos da rota atual anteriores a s até encontrar o salto de divergência $s_d < s$ com um roteador que pertence à rota antiga, $C(t_i)[s_d] \in C(t_{i-1})$. No pior caso, a busca pelo salto de divergência termina na origem, que pertence a qualquer rota no caminho. Se o salto s_d não for idêntico nas duas rotas, $C(t_i)[s_d] \neq C(t_{i-1})[s_d]$, então existe outra mudança de roteamento anterior a s_d e voltamos à primeira etapa (seção 4.1), fazendo $s' = s_d$, para localizá-la e depois remapeá-la. Esse processo é realizado recursivamente até que não reste nenhuma mudança para remapear. O remapeamento dos roteadores nos saltos entre s_d e s_c precisa ser sequencial pois todos estão envolvidos na mudança de roteamento. Para mudanças de roteamento que apenas removem saltos, temos $s_d = s_{\text{esq}}$ e $s_c = s_{\text{dir}}$ e nenhum remapeamento é necessário.

4.3. Exemplo

Considere que a mudança de roteamento mostrada na figura 3 tenha sido detectada com uma sonda para o quinto salto no caminho. Temos $C(t_{i-1})[5] = f$ e $C(t_i)[5] = e$. Como $e \in C(t_{i-1})$, um nó foi inserido antes do quinto salto e o RemapRoute inicia uma

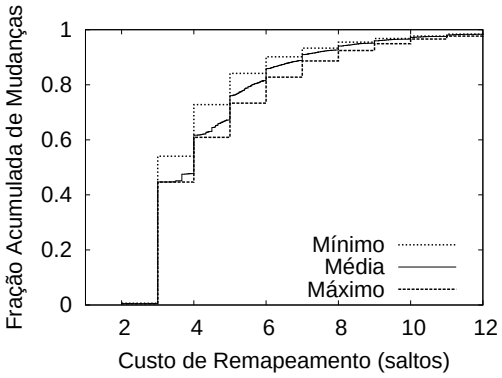


Figura 4. Custo de remapeamento com RemapRoute variando o salto s' de detecção da mudança.

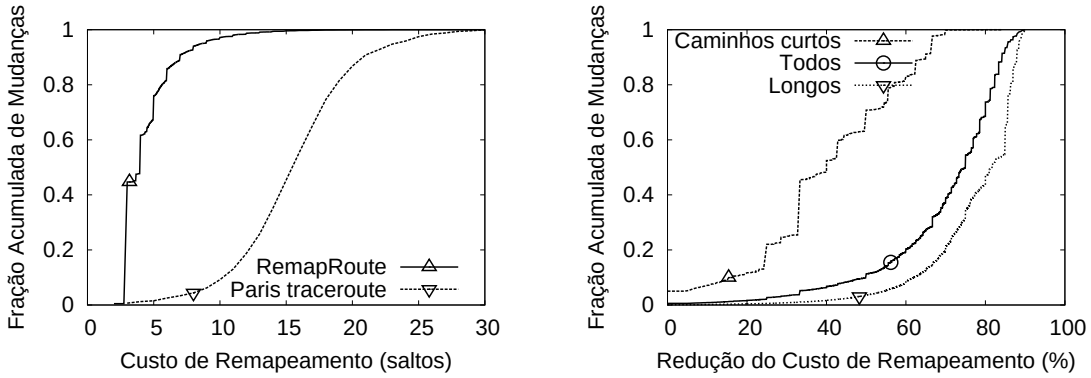
busca binária para identificar o local da mudança. O RemapRoute começa remapeando o segundo salto, onde $C(t_{i-1})[2] = C(t_i)[2] = c$, indicando que a inserção foi realizada entre o segundo e o quinto salto. O RemapRoute remapeia então o terceiro salto, onde $C(t_{i-1})[3] = d$ e $C(t_i)[3] = x$. Como $x \notin C(t_{i-1})$ o RemapRoute passa para a fase de remapeamento local da mudança, onde remapeia o quarto salto e termina.

5. Avaliação via simulação com dados reais

Nesta seção avaliamos o RemapRoute usando simulações dirigidas com dados reais. Utilizamos o mesmo conjunto de dados descrito na seção 3. Nosso foco é comparar o custo de remapear mudanças de caminho usando o Paris traceroute com o custo de remapear usando o RemapRoute.

O custo de remapeamento usando o RemapRoute varia de acordo com o salto s' onde a mudança é detectada. Calculamos o custo de remapeamento do RemapRoute para todos os saltos do caminho onde a mudança pode ser detectada. A figura 4 mostra a distribuição do custo de remapeamento de mudanças para o RemapRoute. Mostramos a distribuição dos custos mínimo, médio e máximo, calculados sobre todos os saltos envolvidos em uma mudança. Vemos que o custo mínimo e máximo em geral são muito próximos. Uma razão para este comportamento é que várias mudanças de caminho envolvem poucos roteadores, logo não existe muita variação em função do salto s' onde a mudança é detectada. No resto deste artigo usaremos o custo médio como representativo do custo de remapeamento com RemapRoute.

A figura 5(a) compara o custo de remapeamento do RemapRoute com o custo de remapeamento do Paris traceroute. Comparando com a figura 1(a) vemos que o RemapRoute frequentemente precisa sondar um número de saltos maior do que o número de saltos envolvidos numa mudança. Este aumento deve-se a mudanças de roteamento que precisam ser localizadas usando a técnica de pesquisa binária antes de serem remapeadas. Independente do processo de localização, o RemapRoute tem custo de remapeamento significativamente menor que o do Paris traceroute. Note que o custo de remapeamento do RemapRoute raramente é menor do que três saltos, mesmo com 9% das mudanças envolvendo apenas dois saltos (figura 1(a)). As mudanças que envolvem apenas dois roteadores apenas removem saltos da rota antiga, e precisamos localizar o salto onde a remoção aconteceu usando busca binária. A busca binária requer pelo menos três sondas a



(a) Custo de remapeamento (b) Redução relativa do custo de remapeamento

Figura 5. Comparação do custo de remapeamento entre RemapRoute e Paris traceroute.

não ser que a mudança seja detectada nos três primeiros saltos do caminho, o que acontece em 0,4% das mudanças em nossos dados.

A figura 5(b) mostra a distribuição da redução no custo de remapeamento quando usamos RemapRoute em vez do Paris traceroute. Calculamos a redução do custo de remapeamento como $(C_{\text{Paris}} - C_{\text{RemapRoute}})/C_{\text{Paris}}$, onde C_{Paris} é o número de saltos remapeados pelo Paris traceroute e $C_{\text{RemapRoute}}$ é o número de saltos remapeados pelo RemapRoute. A linha sólida, calculada para todas as mudanças de caminho em nossos dados, mostra que a redução no custo é significativa. O RemapRoute reduz pra menos da metade o custo de remapeamento de 88% das mudanças de caminho. As linhas tracejadas na figura 5(b) mostram a redução no custo para mudanças em rotas menores que 10 saltos (curtas) e maiores que 20 saltos (longas). Vemos que a redução no custo é mais acentuada para rotas longas, onde o Paris traceroute desperdiça sondas em vários roteadores que não estão envolvidos na mudança, e que o RemapRoute traz redução de custos para remapeamento mesmo em rotas curtas.

5.1. Erros de remapeamento

O RemapRoute remapeia mudanças num caminho dado um salto s' onde uma mudança foi detectada. Isso pode levar a inconsistências caso um caminho sofra duas ou mais mudanças disjuntas antes de detectarmos a primeira mudança. Por exemplo, um caminho $\{a, b, c, d, e, f, g\}$ pode mudar para $\{a, x, c, d, y, f, g\}$ entre duas medições consecutivas com Paris traceroute. Neste caso, podemos detectar duas mudanças nos saltos $s' = 1$ e $s'' = 4$. Infelizmente, dado s' ou s'', o RemapRoute remapeará apenas uma mudança.

Para avaliar a gravidade desse problema, a figura 6 mostra a distribuição do número de mudanças disjuntas para os pares de medições consecutivas dos caminhos em nosso conjunto de dados.[2] Vemos que 79% das medições consecutivas remapeiam apenas uma mudança disjunta, que o RemapRoute remapeará corretamente para qualquer salto s' onde a mudança for detectada. Apenas 3% das medições consecutivas remapeiam três ou mais mudanças disjuntas, indicando que o RemapRoute remapeará a nova rota corretamente na maioria dos casos.

[2]No nosso conjunto de dados só medimos caminhos com Paris traceroute quando uma mudança é detectada. Todos os pares de medições consecutivas remapeiam pelo menos uma mudança disjunta.

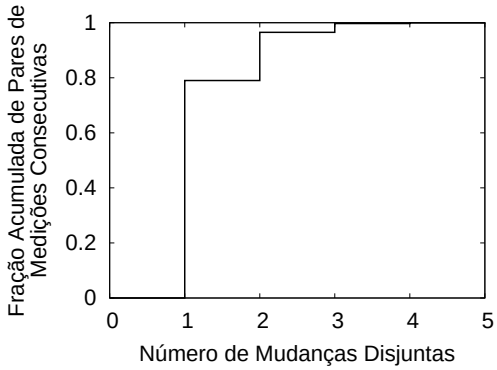


Figura 6. Distribuição do número de mudanças disjuntas entre duas medições com Paris traceroute.

Três outros fatores contribuem para minimizar o impacto de mudanças disjuntas. Primeiro, o RemapRoute pode remapear mudanças disjuntas anteriores ao salto de detecção s' caso ele as detecte durante o processo de remapeamento. Em particular, a probabilidade do RemapRoute remapear uma mudança disjunta anterior ao salto de detecção s' no nosso conjunto de dados é 43%. Segundo, uma mudança disjunta que não for remapeada quando executarmos RemapRoute será detectada e remapeada no futuro (assumindo que o processo de sondagem para detecção de mudanças de caminho seja capaz de detectar todas as mudanças possíveis). Qualquer mudança disjunta que não for remapeada causa apenas uma inconsistência temporária nos dados. Terceiro, a redução do custo de remapeamento obtida com RemapRoute pode ser utilizada para aumentar a frequência de sondagem para detecção de mudanças e reduzir a chance de ocorrerem duas mudanças em um caminho antes de detectarmos a primeira.

Outra limitação do RemapRoute é que o mecanismo de busca binária pode falhar quando a ordem relativa de dois saltos se inverte da rota antiga para a nova rota. Um exemplo extremo, mas ilustrativo, é uma mudança de $C(t_{i-1}) = \{a, b, c, d, e, f\}$ para $C(t) = \{a, e, d, c, b, f\}$. Apenas 0,9% das mudanças de caminho em nossos dados invertem a ordem relativa de saltos. Como inversão da ordem relativa de saltos é um evento raro, tomamos a decisão conservadora de remapear o caminho por inteiro (como o Paris traceroute) quando uma inversão é detectada durante o processo de remapeamento. Como mostram os resultados anteriores, essa limitação não compromete a utilidade do RemapRoute.

6. Avaliação com protótipo real no PlanetLab

Nesta seção avaliamos um protótipo do RemapRoute no PlanetLab. Implantamos o Paris traceroute e o RemapRoute em 140 nós PlanetLab e coletamos medições por 18 horas de 30 de Novembro de 2012. Cada nó executa o Paris traceroute para medir caminhos periodicamente. Como no conjunto de dados utilizado nas seções anteriores, cada nó monitora caminhos para 1.000 destinos escolhidos aleatoriamente de uma lista com 34.820 destinos alcançáveis na Internet. Quando duas medições consecutivas com Paris traceroute detectam uma mudança, executamos o RemapRoute para remapeá-la. Cada nó leva em média 7 horas e 42 minutos para medir os 1.000 caminhos. Devido à baixa frequência de sondagem, este conjunto de dados contém apenas 87.848 mudanças. Os caminhos medidos

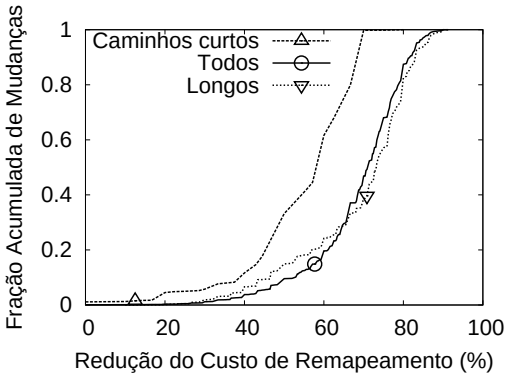


Figura 7. Redução do custo de remapeamento com RemapRoute em cenários reais.

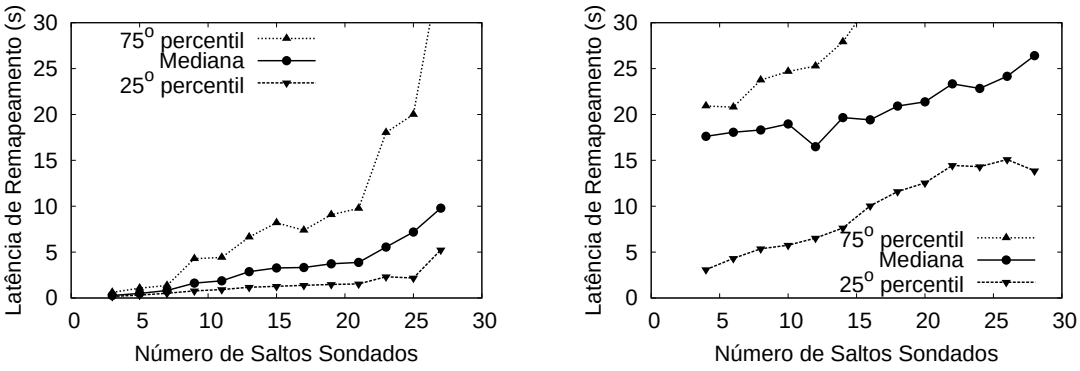
atravessam 7.143 sistemas autônomos e 95% dos sistemas autônomos de grande porte na Internet [Oliveira et al. 2010].

A figura 7 mostra a redução do custo de remapeamento quando usamos o RemapRoute em vez do Paris traceroute. Comparando com a figura 5(b), a redução média do custo de remapeamento no cenário real é quantitativamente similar aos resultados obtidos via simulação (linha sólida). Por exemplo, reduzimos pra menos da metade o custo de remapeamento de 90% das mudanças de caminho no cenário real.

A redução de custo para rotas curtas e longas é mais similar à redução de custo geral no cenário real que nas simulações. Em outras palavras, as linhas tracejadas na figura 7 estão mais próximas da linha sólida que na figura 5(b). Atribuímos essa mudança a três fatores: (i) o menor número de mudanças observadas no cenário real pode limitar a variedade de mudanças observadas; (ii) diferenças no conjunto de caminhos monitorados; e (iii) a diferente forma de detecção de mudanças (os dados utilizados nas simulações foram coletados com o DTRACK).

A figura 8(a) mostra os 25^o, 50^o e o 75^o percentis da latência de remapeamento em função do número de saltos sondados durante o processo de remapeamento. Avaliamos a latência de remapeamento por que o RemapRoute sonda saltos sequencialmente: a decisão do próximo salto a sondar depende do resultado do último salto sondado. O Paris traceroute, em contrapartida, poderia paralelizar a sondagem de saltos (apesar da implementação padrão não fazê-lo). Como a maior parte dos remapeamentos com RemapRoute requer sondagem de poucos saltos (figura 5(a)), a latência de remapeamento geralmente é menor que 5 segundos. A figura 8(b) mostra os 25^o, 50^o e 75^o percentis da latência de remapeamento com Paris traceroute no cenário real. Nosso objetivo não é comparar a latência de remapeamento do RemapRoute com Paris traceroute, pois a latência é diretamente afetada por decisões de implementação da ferramenta. Nosso objetivo é mostrar que a latência de remapeamento com RemapRoute é aceitável para uso em sistemas reais. Notamos ainda que um sistema de mapeamento topológico como o DTRACK pode executar o RemapRoute simultaneamente em caminhos diferentes caso mais de uma mudança seja detectada num curto intervalo de tempo.

Por último, avaliamos se o remapeamento com o RemapRoute é equivalente a utilizar o Paris traceroute para medir o novo caminho por inteiro. Para cada mudança



(a) RemapRoute (b) Paris traceroute

Figura 8. Latência de remapeamento em cenários reais.

observada no cenário real, comparamos os saltos remapeados pelo RemapRoute com a rota medida pelo Paris traceroute. Apenas 0,6% das medições com RemapRoute são diferentes das medições com Paris traceroute. A identificação de roteadores que fazem balanceamento de carga usando Paris traceroute ou RemapRoute é probabilística [Veitch et al. 2009]. Por exemplo, a configuração padrão do Paris traceroute e do RemapRoute identifica todos os roteadores que fazem balanceamento de carga numa rota com 95% de confiabilidade. Erros de medição são inevitáveis e causam diferenças entre remapeamentos independente da ferramenta utilizada. Outra causa para diferença nos remapeamentos são mudanças de roteamento que acontecem no intervalo entre a medição com Paris traceroute e a medição com RemapRoute.

Sumarizando, o remapeamento de mudanças de caminho na Internet com RemapRoute é tão preciso quanto remapeamento com Paris traceroute, reduz significativamente o número de sondas enviadas e tem pouco impacto na latência de remapeamento.

7. Trabalhos relacionados

Operadores podem utilizar mensagens dos protocolos de roteamento (e.g., OSPF e IS-IS) e arquivos de configuração dos roteadores para mapear a topologia de sua rede [Feamster and Balakrishnan 2005, Turner et al. 2010, Markopoulou et al. 2008]. Esta abordagem resulta em mapas completos e precisos da topologia, mas só está disponível para operadores de redes e é restrita a uma única rede. Para mapear múltiplas redes, podemos utilizar coletores públicos de mensagens BGP[3] para construir um mapa dos sistemas autônomos da Internet [Oliveira et al. 2010, Dhamdhere and Dovrolis 2008]. Infelizmente, o BGP não expõe todos os enlaces da rede e coletores públicos de mensagens BGP não cobrem todos os sistemas autônomos da Internet [Oliveira et al. 2010, Cohen and Raz 2006]. Neste trabalho tomamos a abordagem ortogonal de medir a topologia da rede no nível de roteadores usando medições ativas.

Pesquisa sobre mapeamento topológico no nível de roteadores usando medições ativas têm três objetivos principais: (i) aumentar a cobertura da Internet, (ii) aumentar a precisão da topologia construída e (iii) aumentar a frequência das medições.

[3]The University of Oregon Routeviews Project, http://www.routeviews.org
RIPE Routing Information Service, http://www.ripe.net/data-tools/stats/ris

A abordagem clássica para aumentar a cobertura da Internet é monitorar um grande número de caminhos. A plataforma Skitter/Ark da CAIDA [Claffy et al. 2009] tenta cobrir toda a Internet usando alguns monitores para medir caminhos para todos os prefixos $/24$ anunciados na Internet. O problema é que o Skitter/Ark demora de dois a três dias para coletar a topologia devido ao grande número de caminhos monitorados e limitações de banda nos monitores. Uma alternativa é dividir a carga de sondagem das medições entre vários monitores, como nos sistemas DIMES [Shavitt and Weinsberg 2009] e Ono [Choffnes et al. 2010]. Neste trabalho assumimos que o conjunto de monitores e destinos é fixo. Porém, utilização do RemapRoute é ortogonal ao conjunto de monitores e destinos. O RemapRoute pode ser usado por qualquer um dos sistemas acima para reduzir a sobrecarga de rede.

Técnicas para aumentar a precisão da topologia coletada tentam inferir mais informações sobre a rede do que medições tradicionais com traceroute. O Paris traceroute envia sondas adicionais variando sistematicamente os valores nos campos do cabeçalho IP para detectar todos os roteadores que fazem balanceamento de carga em uma caminho [Augustin et al. 2007, Veitch et al. 2009]. O RemapRoute detecta roteadores que fazem balanceamento de carga utilizando o mesmo algoritmo que o Paris traceroute. O DisCarte usa traceroute e sondas com a opção *Record Route* do protocolo IP ativada para coletar duas sequências relacionadas de roteadores em caminhos da Internet [Sherwood et al. 2008]. O DisCarte pós-processa essas sequências com ferramentas de aprendizado de máquina para combiná-las em uma topologia mais precisa. Estas e outras técnicas enviam sondas adicionais e aumentam o custo de mapeamento da topologia. O objetivo do RemapRoute é complementar: reduzir o custo do remapeamento de mudanças de roteamento, aumentando a disponibilidade de sondas para coleta de topologias mais precisas.

Nosso trabalho é mais relacionado com técnicas para aumentar a frequência de medições da topologia da Internet. Em geral, monitores têm banda de rede limitada para mapear a topologia. Reduzir o custo de cada medição da topologia aumenta diretamente a frequência com a qual medições podem ser coletadas. O RocketFuel, por exemplo, reduz o custo para mapear a topologia de um sistema autônomo alvo descartando caminhos que têm roteadores de ingresso e egresso no sistema autônomo alvo idênticos a outro caminho já medido [Spring et al. 2002]. Outra abordagem é reduzir o custo de medições escolhendo apenas um destino em cada sub-rede num sistema autônomo [Beverly et al. 2010]. O Doubletree reduz sondas redundantes nos roteadores próximos a monitores (compartilhados pelos caminhos partindo do monitor) e nos roteadores próximos a destinos (compartilhados pelos caminhos que terminam no destino) [Donnet et al. 2005]. O RemapRoute foi desenvolvido para reduzir o custo do remapeamento de mudanças no DTRACK, nosso sistema de mapeamento topológico [Cunha et al. 2011b]. O RemapRoute e o DTRACK complementam as técnicas existentes para redução do custo de mapeamento topológico. O DTRACK reduz sondas redundantes para roteadores próximos aos monitores como o Doubletree e é compatível com técnicas descritas acima para selecionar quais caminhos mapear.

8. Conclusões e trabalhos futuros

A manutenção de mapas completos e atualizados da Internet é difícil devido à grande quantidade de sondas necessárias e restrições de banda nos monitores. Neste trabalho propomos o RemapRoute, uma ferramenta para reduzir o custo de remapeamento de

mudanças de roteamento na Internet. Dadas a rota anterior a uma mudança de roteamento e um salto (*hop*) onde a mudança foi detectada, o RemapRoute (1) realiza uma busca binária pelo salto onde a mudança aconteceu e (2) faz o remapeamento local da mudança. Comparado com a abordagem atual de remapear a nova rota por inteiro usando traceroute, o RemapRoute reduz significativamente o número de saltos sondados para remapear mudanças de roteamento na Internet. Essa redução de saltos sondados aumenta a disponibilidade de sondas, potencializando a medição de mais caminhos ou aumento da frequência de medições. O remapeamento com RemapRoute é tão preciso quanto remapear o caminho inteiro com traceroute, e a latência de remapeamento é satisfatória para utilização do RemapRoute em sistemas reais. RemapRoute é mais um passo na construção de mapas da topologia da Internet mais completos e atualizados.

Como trabalho futuro, pretendemos integrar o RemapRoute no DTRACK e prover um serviço de mapeamento da Internet aberto para disponibilizar informações sobre a topologia para pesquisadores e aplicações. Queremos também reduzir ainda mais o custo de remapeamento de mudanças no DTRACK. Atualmente, o DTRACK remapeia cada caminho separadamente. Esta abordagem desperdiça sondas caso vários caminhos sejam afetados pela mesma mudança. Pretendemos desenvolver mecanismos para prever quais caminhos são afetados por uma mesma mudança e remapear apenas um deles.

Referências

Augustin, B., Friedman, T., and Teixeira, R. (2007). Measuring Load-balanced Paths in the Internet. In *Proc. IMC*.

Beverly, R., Berger, A., and Xie, G. (2010). Primitives for Active Internet Topology Mapping: Toward High-Frequency Characterization. In *Proc. IMC*.

Choffnes, D. R., Bustamante, F. E., and Ge, Z. (2010). Crowdsourcing Service-level Network Event Monitoring. In *Proc. ACM SIGCOMM*.

Claffy, K., Hyun, Y., Keys, K., Fomenkov, M., and Krioukov, D. (2009). Internet Mapping: from Art to Science. In *Proc. IEEE CATCH*.

Cohen, R. and Raz, D. (2006). The Internet Dark Matter - on the Missing Links in the AS Connectivity Map. In *Proc. IEEE INFOCOM*.

Cunha, I., Teixeira, R., and Diot, C. (2011a). Measuring and Characterizing End-to-End Route Dynamics in the Presence of Load Balancing. In *Proc. PAM*.

Cunha, I., Teixeira, R., Veitch, D., and Diot, C. (2011b). Predicting and Tracking Internet Path Changes. In *Proc. ACM SIGCOMM*.

Dhamdhere, A. and Dovrolis, C. (2008). Ten Years in the Evolution of the Internet Ecosystem. In *Proc. IMC*.

Dhamdhere, A., Teixeira, R., Drovolis, C., and Diot, C. (2007). NetDiagnoser: Troubleshooting Network Unreachabilities Usind End-to-end Probes and Routing Data. In *Proc. ACM CoNEXT*.

Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. (2002). Globally Distributed Content Delivery. *IEEE Internet Computing*, 6(5):50–58.

Donnet, B., Raoult, P., Friedman, T., and Crovella, M. (2005). Efficient Algorithms for Large-scale Topology Discovery. In *Proc. ACM SIGMETRICS*.

Duffield, N. (2006). Network Tomography of Binary Network Performance Characteristics. *IEEE Trans. on Inf. Theory*, 52(12):5373–5388.

Feamster, N. and Balakrishnan, H. (2005). Detecting BGP Configuration Faults with Static Analysis. In *Proc. USENIX NSDI*.

Jacobson, V. (1989). traceroute. Available at ftp://ftp.ee.lbl.gov/traceroute.tar.gz.

Katz-Bassett, E., Scott, C., Choffnes, D. R., Cunha, I., Valancius, V., Feamster, N., Madhyastha, H. V., Anderson, T., and Krishnamurthy, A. (2012). LIFEGUARD: Practical Repair of Persistent Route Failures. In *Proc. ACM SIGCOMM*.

Kompella, R., Yates, J., Greenberg, A., and Snoeren, A. (2007). Detection and Localization of Network Blackholes. In *Proc. IEEE INFOCOM*.

Madhyastha, H., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., and Venkataramani, A. (2006). iPlane: an Information Plane for Distributed Services. In *Proc. USENIX OSDI*.

Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C. N., Ganjali, Y., and Diot, C. (2008). Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Trans. Netw.*, 16(4):749–762.

Oliveira, R., Pei, D., Willinger, W., Zhang, B., and Zhang, L. (2010). Quantifying the Completeness of the Observed Internet AS-level Structure. *IEEE/ACM Trans. Netw.*, 18(1):109–122.

Paxson, V. (1997). End-to-end Routing Behavior in the Internet. *IEEE/ACM Trans. Netw.*, 5(5):601–615.

Shavitt, Y. and Weinsberg, U. (2009). Quantifying the Importance of Vantage Points Distribution in Internet Topology Measurements. In *Proc. IEEE INFOCOM*.

Sherwood, R., Bender, A., and Spring, N. (2008). DisCarte: a Disjunctive Internet Cartographer. In *Proc. ACM SIGCOMM*.

Spring, N., Mahajan, R., and Wetherall, D. (2002). Measuring ISP Topologies with Rocketfuel. In *Proc. ACM SIGCOMM*.

Turner, D., Levchenko, K., Snoeren, A., and Savage, S. (2010). California Fault Lines: Understanding the Causes and Impact of Network Failures. In *Proc. ACM SIGCOMM*.

Veitch, D., Augustin, B., Friedman, T., and Teixeira, R. (2009). Failure Control in Multipath Route Tracing. In *Proc. IEEE INFOCOM*.