Protocolos de Roteamento e Roteamento Baseado em Estado de Enlaces

Diego Passos

29 de Maio de 2017

1 Roteamento: Introdução

O conceito de roteamento já foi definido em aulas anteriores, bem como seu relacionamento com o conceito de encaminhamento: enquanto o encaminhamento diz respeito à movimentação dos pacotes das portas de entrada para as portas de saída dos roteadores, o roteamento fornece os subsídios necessários para esta primeira tarefa. Em particular, o roteamento é o processo através do qual são montadas as tabelas de roteamento que, posteriormente, são consultadas para guiar o encaminhamento dos pacotes.

Esta montagem da tabela de roteamento pode ser realizada de várias formas na Internet. Para um *host* que chega a uma nova rede, é comum que o protocolo DHCP — estudado em aulas anteriores — forneça, além do endereço IP, uma ou mais entradas que devem ser adicionadas à tabela de roteamento. Outra alternativa é a adição manual de entradas à tabela, feita normalmente por um administrador de rede em cenários de configuração basicamente estática.

Para roteadores mais próximos do núcleo da rede, no entanto, é desejável que estas configurações sejam realizadas de forma mais dinâmica e automática. Eventos como a adição de uma nova grande sub-rede à Internet — pense, por exemplo, no surgimento de um novo grande ISP —, a adição ou remoção de enlaces e a falha de um ou mais equipamentos muitas vezes demandam a alteração das tabelas de roteamento de roteadores do núcleo. Reduzir o grau de intervenção humana necessária para lidar com este tipo de situação é positivo, entre outros motivos, pela melhora no tempo de resposta da rede a estas mudanças.

Nesta aula — e nas demais aulas até o final desta disciplina —, estaremos interessados especificamente nesta abordagem de roteamento dinâmico. Este tipo de roteamento é efetuado através da execução de protocolos particulares conhecidos como **protocolos de roteamento**.

Antes de discutirmos como funcionam estes protocolos, é útil introduzir — ou revisar — uma estrutura de dados extensivamente utilizada nesta área e em outras áreas da computação: os grafos. Um grafo é uma estrutura de dados extremamente flexível composta por **vértices** — ou nós — e **arestas**. Os vértices são interligados pelas arestas, que podem ser direcionadas — *i.e.*, saindo de um vértice e chegando a outro — ou não. Arestas direcionadas muitas vezes são chamadas de arcos. Muitas vezes, grafos são ponderados, o que significa que suas arestas são associadas a **pesos**: valores numéricos indicando alguma qualidade ou característica de cada aresta. Em uma notação matemática mais formal, um grafo é geralmente identificado por $G = (N, E)$, onde N é o conjunto dos vértices e E é o conjunto das arestas.

Na área de roteamento, grafos são úteis para representar a **topologia da rede**. Isto é, grafos provêm uma abstração que permite a representação do estado atual da rede, incluindo seus nós — roteadores e *hosts* — e os enlaces que os interconectam. Nesta representação, os nós da rede são mapeados para vértices, enquanto os enlaces são representados pelas arestas.

Os pesos das arestas em um grafo utilizado para representar a topologia de uma rede geralmente representam alguma característica de desempenho do enlace, como a sua taxa de transmissão, a ocupação de sua fila de pacotes, a probabilidade de perda de pacotes transmitidos por ele, ou o percentual de utilização do enlace. Esta representação numérica de uma destas características de desempenho do enlace é comumente chamada de **métrica de roteamento**. As métricas de roteamento também permitem que um estabeleçamos o **custo de uma rota**, *i.e.*, um valor numérico representando o quão bom — ou ruim — é um determinado caminho. Embora isso possa variar, em geral pesos e custos maiores estão associados a enlaces e caminhos piores. Além disso, o custo total de uma rota geralmente é dado pelo somatório dos pesos dos enlaces que a compõem.

Dada uma topologia de rede representada na forma de um grafo contendo pesos que representam a qualidade de seus enlaces, a pergunta que gostaríamos de responder no contexto das próximas aulas é: quais são os melhores caminhos entre cada par de nós da nossa rede? Assumindo que pesos e custos

mais baixos sejam melhores, esta pergunta pode ser formulada da seguinte maneira equivalente: quais são os caminhos de custo mínimo no grafo da topologia?

Felizmente, o problema de encontrar caminhos de custo mínimo em grafos já foi extensivamente estudado em computação — muito antes do advento da Internet, inclusive. Hoje, são conhecidos diversos algoritmos eficientes, com características variadas, para a solução deste problema. Dois, em particular, são comumente aplicados à área de roteamento e, portanto, serão estudados em detalhes nesta disciplina: os algoritmos de Dijkstra e de Bellman-Ford.

Estes dois algoritmos são os mais utilizados por duas vertentes diferentes de protocolos de roteamento: os protocolos baseados em Estado de Enlaces — que comumente usam o Algoritmo de Dijkstra — e os protocolos baseados em Vetor de Distância — que comumente empregam o Algoritmo de Bellman-Ford.

Os protocolos baseados em Estado de Enlaces demandam que cada roteador conheça toda a topologia da rede, armazenada na forma de um grafo em memória principal. Neste tipo de protocolo, cada roteador acompanha o estado dos seus enlaces para com os seus vizinhos. Periodicamente, esta informação é disseminada em *broadcast* para o restante da rede através do envio de um pacote de controle contendo a listagem dos enlaces locais. Ao receber um pacote deste tipo, outro roteador da rede deve processar o pacote e atualizar sua própria visão da rede como um todo, adicionando e removendo enlaces ou atualizando seus pesos. Toda vez que um roteador detecta uma mudança na sua visão da topologia, um algoritmo — tipicamente o Algoritmo de Dijkstra — é executado localmente com o objetivo de determinar as melhores rotas na rede — ao menos às que concernem o roteador local.

Já nos protocolos baseados em Vetor de Distâncias, não é necessário que os nós conheçam toda a topologia da rede. Cada roteador precisa conhecer apenas sua vizinhança. Roteadores, então, trocam pacotes de controle do protocolo de roteamento e executam um **algoritmo distribuído** que após algumas iterações converge para determinar o melhor próximo salto a ser utilizado para alcançar cada destino na rede.

Há muitas outras características que podem ser utilizadas para classificar protocolos de roteamento. Alguns exemplos incluem:

1. **Estático ou dinâmico**. A rigor, todo protocolo de roteamento é dinâmico — do contrário, poderíamos simplesmente utilizar a abordagem de construção manual das tabelas de roteamento. No entanto, certas redes são mais dinâmicas que outras. A topologia da Internet, por exemplo, tende a mudar de forma muito mais lenta que a topologia de uma rede sem fio composta por nós móveis. Neste sentido, pode-se dizer que protocolos de roteamento para a Internet tendem a ser **menos dinâmicos** que protocolos de roteamento utilizados para estas redes sem fio.

2. **Pró-ativo ou reativo**. Protocolos pró-ativos — os mais comuns na Internet — são aqueles que sempre buscam encontrar rotas para todos os pares de nós da rede, independentemente de se há ou não tráfego atualmente necessitando de tais rotas. Em contrapartida, um protocolo reativo é aquele que só se dá ao trabalho de encontrar uma rota entre um determinado par de nós da rede quando, de fato, há demanda por esta rota (ou seja, quando é necessário encaminhar um ou mais pacotes entre estes nós). Protocolos reativos são geralmente mais populares em redes nas quais há grande preocupação com o *overhead* associado ao processo de roteamento e considera-se "mais barato" adiar a decisão de roteamento para quando ela é estritamente necessária.

2 O Algoritmo de Dijkstra e Protocolos Baseados em Estado de Enlaces

Conforme explicado na seção anterior, em protocolos baseados em estado de enlace, cada roteador mantém uma visão completa da topologia da rede na forma de um grafo armazenado na sua memória principal. Isso é realizado através da difusão, por cada nó, das informações dos enlaces pertencentes à sua vizinhança — daí o nome **Estado de Enlaces**.

A ideia é que, através deste processo de difusão de informações, a rede convirja para uma situação na qual todos os roteadores possuam exatamente a mesma visão da topologia. Assumindo que isso seja verdade, cada roteador pode aplicar um mesmo algoritmo de caminhos de custo mínimo em grafos, chegando às mesmas melhores rotas obtidas pelos demais roteadores da rede. Isso nem sempre é verdade — *i.e.*, nem sempre todos os roteadores possuirão a mesma visão da rede — o que, na prática, traz problemas que serão discutidos mais tarde.

Além da topologia da rede, o Algoritmo de Dijkstra recebe como entrada um **nó de origem**. O algoritmo, então, determina as rotas de custo mínimo **do nó de origem para todos os outros nós**

no grafo da topologia. Esta saída do algoritmo é usada pelo roteador local para construir sua tabela de roteamento, associando cada destinatário ao enlace de próximo salto no caminho determinado pelo algoritmo.

O Algoritmo de Dijkstra opera de forma iterativa, sendo que ao final de cada iteração descobre-se uma nova rota ótima até um determinado destinatário. Se há n nós na rede, portanto, o Algoritmo de Dijkstra precisará de $n-1$ iterações — uma para cada nó, exceto a origem.

Além do grafo da topologia e do nó de origem, o algoritmo utiliza uma série de variáveis auxiliares:

1. u. Denota o nó de origem, *i.e.*, o nó que executa o algoritmo.

2. $c(x, y)$. Denota o peso do enlace entre do nó x para o nó y. A rigor, esta informação é parte da entrada do algoritmo, sendo basicamente o conjunto de pesos associados às arestas. Deve-se notar, no entanto, que o algoritmo assume que, caso não exista um enlace de x para y, o peso é **infinito**.

3. $D(v)$. Custo da melhor rota conhecida até agora entre o nó de origem e o nó v. Estes valores são dinamicamente alterados à medida que o algoritmo opera.

4. $p(v)$. Predecessor do nó v — *i.e.*, último salto — na melhor rota conhecida até o momento entre o nó de origem e o nó v. Esta também é uma informação atualizada ao longo do processamento do algoritmo.

5. N'. Conjunto dos nós para os quais já conhecemos a melhor rota possível — *i.e.*, temos certeza de que a melhor rota conhecida até agora para aquele destinatário é, de fato, a melhor rota dentre todas as existentes. Às vezes chamado de **conjunto de nós definitivos**. Este conjunto é inicialmente vazio e um novo nó é adicionado a ele a cada iteração do algoritmo.

A ideia central do Algoritmo de Dijkstra é relativamente simples: se w é o predecessor da melhor rota de u para v, então o trecho desta rota entre u e w é o melhor caminho de u para w. De forma alternativa, podemos afirmar que a melhor rota de u para um certo destinatário v necessariamente é igual à melhor rota de u para algum outro destinatário w anexada do enlace de w para v. Logo, se em um dado passo do algoritmo conhecemos as melhores rotas de u para todos os nós do conjunto N', com certeza há uma rota ótima para algum dos demais destinatários — digamos, v — formada pela concatenação de uma das rotas ótimas já encontradas — digamos, até um certo destinatário w — com o enlace de w para v.

A primeira etapa do algoritmo é a inicialização das suas variáveis auxiliares. Em particular, o conjunto N' é inicializado para conter apenas o nó de origem u. Essa inicialização faz sentido porque a melhor rota de u para ele mesmo é sempre trivial. Além disso, já durante a inicialização podemos computar alguns valores iniciais para o vetor $D(.)$: $D(u) = 0$ e $D(v) = c(u, v)$ para todo o vizinho v do nó de origem u. Repare que segunda parte é um chute inicial: estamos apenas afirmando que conhecemos *um possível caminho* de u para v formado pelo enlace direto entre eles, mas nada impede que encontremos um caminho melhor, indireto, posteriormente.

Após a inicialização, o algoritmo entra em seu *loop* principal. A cada iteração deste *loop*, o algoritmo percorre todos os nós que ainda não foram inseridos no conjunto N', comparando os custos das melhores rotas conhecidas para eles até o momento. Aquele nó cuja melhor rota conhecida até o momento possuir o menor custo é adicionado ao conjunto N', denotando que a rota conhecida atualmente é, de fato, a melhor[1]. Além de inserir este novo nó — digamos w — no conjunto de definitivos, o algoritmo verifica para cada vizinho de w se descobrimos uma rota melhor do que aquela que conhecemos até aqui. Para isso, o algoritmo avalia se a melhor rota da origem para w anexada do enlace de w para cada um de seus vizinhos v tem custo menor que $D(v)$ — o melhor custo conhecido entre a origem e v atualmente. Se sim, atualiza-se o valor de $D(v)$ e marca-se w como o predecessor de v na melhor rota conhecida até agora para v.

Repare que esta comparação de custos das melhores rotas executada a cada iteração do algoritmo pode resultar em empates. Neste caso, o critério de desempate é arbitrário. Normalmente, é empregado como primeiro critério o número de saltos das rotas que estão sendo comparadas. Quando este algoritmo é utilizado em um protocolo de roteamento baseado em Estado de Enlaces é fundamental que todos os roteadores utilizem os mesmos critérios de desempate e que estes critérios sejam determinísticos, já que espera-se que todos obtenham o mesmo resultado final.

A condição de parada para o Algoritmo de Dijkstra é simples: o algoritmo é executado até que todos os nós da topologia sejam adicionados ao conjunto de definitivos. Neste momento, teremos,

[1] O Algoritmo de Dijkstra não permite que arestas possuam pesos negativos. Com isso, o custo do melhor caminho até um nó nunca será menor que o custo dos melhores caminhos até seus predecessores. Essa propriedade justifica esta decisão do Algoritmo de Dijkstra.

com certeza, os melhores custos $D(v)$ para cada destinatário v, além da informação de qual é seu predecessor na rota correspondente.

Repare que o conjunto dos melhores caminhos obtidos através do Algoritmo de Dijkstra forma uma árvore enraizada no nó de origem. Revisitaremos este fato em aulas posteriores. Repare, ainda, que através da informação dos nós predecessores de cada destinatário, é possível determinar o enlace de primeiro salto na melhor rota até aquele destinatário. Basta que se proceda recursivamente, percorrendo os predecessores até que se chegue ao próprio nó de origem: se o predecessor de um nó v é o próprio nó de origem, o enlace de próximo salto para v é o próprio enlace $u \rightarrow v$; caso contrário, o enlace de próximo salto para v é igual ao enlace de próximo salto de seu predecessor. Com isso, é possível construir a tabela de roteamento.

O Algoritmo de Dijkstra é extremamente popular. Entre outros motivos, este algoritmo é muito eficiente, tendo uma complexidade assintótica de tempo de execução $O(n^2)$, em implementações simples, ou $O(n \cdot log\ n)$, com o emprego de estruturas de dados mais sofisticadas. Isso viabiliza o emprego deste algoritmo mesmo em redes relativamente grandes.

É importante notar que protocolos de roteamento executam o Algoritmo de Dijkstra toda vez que alguma mudança é verificada no estado dos enlaces da rede. Isso pode introduzir um processo oscilatório nas decisões de roteamento. Considere, por exemplo, que a métrica de roteamento usada pela rede consiste no percentual de tempo de utilização de cada enlace. Neste caso, ao decidir pela utilização de uma certa rota, o protocolo poderá estar, ainda que indiretamente, direcionando tráfego da rede para os enlaces que a formam. Logo, estes enlaces passarão ser mais utilizados, tendo sua métrica piorada. Isso faz com que, em uma próxima avaliação, o custo desta rota piore como um todo, o que pode resultar na escolha de uma nova rota. Neste caso, os enlaces desta nova rota terão suas métricas prejudicadas, o que facilitará a seleção de uma nova rota na próxima execução do algoritmo[2].

Por fim, lembre-se que uma das hipóteses dos protocolos baseados em Estado de Enlaces é que cada roteador terá exatamente a mesma visão da topologia de rede, o que garante que todos encontrarão as mesmas rotas ótimas em suas execuções locais do Algoritmo de Dijkstra. Na prática, no entanto, isso nem sempre é verdade. Note que os pacotes de controle usados pelos protocolos de roteamento para a difusão das informações de topologia podem eventualmente ser perdidos. Além disso, ainda que um destes pacotes não seja perdido, ele demora algum tempo para alcançar todos os roteadores da rede. Em particular, roteadores mais próximos da origem do pacote receberão a informação mais rapidamente, enquanto os mais distantes demorarão mais. Logo, são comuns as inconsistências nas visões que cada roteador possui da topologia da rede. De fato, quanto mais dinâmica for a topologia de uma rede, mais frequentes e representativas tendem a ser estas discrepâncias entre as visões dos roteadores.

Mas qual é a consequência prática destas inconsistências? A depender do caso — e isto não é raro — pode não haver qualquer consequência negativa. É possível também que esta inconsistência simplesmente leve alguns pacotes a serem encaminhados por rotas não-ótimas. Entretanto, uma consequência relativamente comum e bem mais danosa ao funcionamento da rede é a ocorrência dos chamados *loops* de roteamento. Neste caso, como cada roteador é responsável por determinar apenas o seu próximo salto no caminho até o destinatário, a depender das rotas diferentes encontradas por eles, os pacotes podem passar indefinidamente por uma mesma sequência cíclica de roteadores[3]. Como esta ocorrência é relativamente comum na Internet, o protocolo IP inclui um mecanismo para mitigar seus efeitos: o campo TTL dos datagramas.

[2]Note que esta característica oscilatória também pode ocorrer com protocolos baseados em Vetor de Distâncias. De fato, esta oscilação diz respeito muito mais à métrica de roteamento que ao protocolo.

[3]Um exemplo trivial envolvendo dois roteadores, A e B: A acredita que o melhor caminho até o destinatário tem como próximo salto B, enquanto B acredita que o melhor caminho para o destinatário tem como próximo salto A.