Roteamento Broadcast e Multicast

Diego Passos

14 de Junho de 2017

1 Introdução

Até aqui, nesta disciplina, nos preocupamos quase exclusivamente com a comunicação *unicast* — *i.e.*, um modo de comunicação no qual o destinatário de uma mensagem é um único nó específico da rede. Em particular, os protocolos de roteamento estudados até aqui todos se preocupam apenas com este modo. Nesta aula, no entanto, discutiremos brevemente dois outros modos de comunicação: o *broadcast* e o *multicast*. Mais especificamente, nosso enfoque será nos problemas de encaminhamento e roteamento eficiente de pacotes nestes modos de comunicação.

2 Roteamento *Broadcast*

Em uma comunicação *broadcast*, um nó de origem transmite um pacote **destinado a todos os nós de um certo domínio de *broadcast***. Para os propósitos desta disciplina, um domínio de *broadcast* pode ser entendido como uma rede — por exemplo, uma sub-rede IP. Desta forma, quando um certo nó conectado a uma sub-rede IP gera um datagrama IP cujo endereço de destino é o endereço de *broadcast* daquela rede, o objetivo é que este datagrama seja recebido por todas as interfaces conectadas àquela sub-rede.

Considere um domínio de *broadcast* compreendido por quatro roteadores, R1, R2, R3 e R4, e os seguintes enlaces bidirecionais: $R1 \leftrightarrow R2$, $R2 \leftrightarrow R3$, $R2 \leftrightarrow R4$, $R3 \leftrightarrow R4$. Suponha que R1 deseje enviar um pacote em *broadcast* nesta rede — ou seja, o pacote deve ser recebido por R2, R3 e R4.

Uma possível abordagem seria a utilização de comunicação *unicast* para emular a comunicação *broadcast*. Em outras palavras, R1 poderia gerar três cópias do pacote, cada um endereçada a um dos outros três roteadores da rede. Desta forma, o problema de roteamento se resumiria ao roteamento *unicast*, estudado nas aulas anteriores.

Esta solução, no entanto, não é eficiente — e, em certos casos, seque viável. Considere, por exemplo, o conjunto de pacotes que passam pelo enlace $R1 \leftrightarrow R2$: nesta solução, este enlace carrega três cópias do mesmo pacote. Esta replicação de várias cópias de um mesmo pacote pelo enlace é um desperdício de recursos. Idealmente, gostaríamos que um pacote de uma comunicação em *broadcast* passasse, no máximo, uma única vez por cada enlace da rede.

Repare que, neste exemplo, seria muito mais eficiente que R1 gerasse apenas uma cópia do pacote, transmitindo-a para R2 através de $R1 \leftrightarrow R2$. R2, por outro lado, poderia replicar o pacote original pelos enlaces $R2 \leftrightarrow R3$ e $R2 \leftrightarrow R4$, resultando no mesmo ser recebido por todos os nós da rede.

Outro inconveniente do uso de comunicação *unicast* para emular uma comunicação *broadcast* é a necessidade da fonte conhecer de antemão todos os destinatários. Isso não é viável em várias aplicações de comunicação *broadcast*. Como um exemplo já discutido nesta disciplina, considere a mensagem *DHCP Discovery* utilizada pelo protocolo DHCP. Esta mensagem é utilizada quando um *host* se conecta a uma rede — potencialmente pela primeira vez — e, sem nenhum conhecimento prévio, precisa descobrir um ou mais servidores DHCP disponíveis. O *host* que origina a mensagem gostaria que esta alcançasse todas as interfaces conectadas à mesma sub-rede, mas não é razoável assumir que ele possa emular esta funcionalidade transmitindo múltiplos pacotes *unicast* pelo simples fato de não conhecer sequer qual é a faixa de endereços IP utilizada naquela sub-rede.

A solução ideal para o roteamento *broadcast*, portanto, passa por mover a funcionalidade de replicação do pacote *broadcast* da origem para cada roteador da rede. Em outras palavras, novas cópias do pacote devem ser replicadas apenas quando, de fato, um roteador precisar realizar transmissões do pacote por vários enlaces diferentes. O roteamento *broadcast* trata justamente disso: determinar por qual ou quais portas de saída um roteador deve replicar um pacote *broadcast* recém recebido.

Há basicamente duas estratégias para roteamento *broadcast*: a **inundação** e o emprego de **árvores geradoras**. Árvores geradoras garantem maior eficiência no encaminhamento, mas precisam ser

previamente estabelecidas, enquanto a inundação pode resultar em envios desnecessários, embora tenha implementação mais simples.

A ideia da inundação é simples: ao receber um pacote destinado ao endereço de *broadcast*, um roteador deve **replicá-lo para todas as suas portas de saída, exceto para aquela pela qual o pacote foi recebido**. A opção por não enviar o pacote pelo mesmo enlace pelo qual ele foi recebido é facilmente justificável, já que, se um roteador R2 acabou de receber o pacote de R1, R1 certamente já o recebeu.

A simples inundação do pacote, no entanto, pode resultar — e geralmente resulta — em uma série de problemas. Se a topologia da rede contém ciclos, então este procedimento de inundação inevitavelmente resulta em roteadores recendo o pacote de *broadcast* múltiplas vezes. Neste caso, cada nova cópia do pacote recebido seria replicada novamente pelos enlaces de saída, gerando novas cópias. Este processo continuaria por um tempo indeterminado, progressivamente introduzindo mais e mais cópias do pacote na rede, causando desperdício de recursos que poderiam ser empregados para outros pacotes. Este fenômeno, bastante danoso ao desempenho da rede, recebe o nome de **tempestade de *broadcast***[1].

Como topologias com ciclos são comuns, para que a inundação funcione é necessária a introdução de algum mecanismo de controle que limite o escopo desta inundação. Isso dá origem a uma técnica chamada de **inundação controlada**.

Uma possível implementação da inundação controlada é através do uso de identificadores de pacotes e do armazenamento de estado nos comutadores. Assumindo que cada pacote de *broadcast* possua um identificador único, quando um roteador recebe um pacote deste tipo ele pode utilizar este identificador para consultar uma tabela local dos pacotes por ele já inundados. Caso o pacote ainda não tenha sido inundado por ele, o roteador realiza sua inundação — *i.e.*, replica o pacote para todas as suas portas de saída, exceto aquela pela qual o pacote foi recebido — e armazena esta informação na sua tabela. Caso contrário, o pacote é simplesmente descartado. Ainda que o pacote seja recebido novamente no futuro, o roteador evitará realizar nova inundação. Desta forma, cada roteador da rede transmitirá o pacote em *broadcast* uma única vez, garantindo que este processo de transmissão em *broadcast* do pacote eventualmente pare.

Embora o IPv4 possua no seu cabeçalho campos que possam ser utilizados para esta identificação, isso nem sempre é verdade para todos os protocolos que precisam dar suporte a comunicação em *broadcast*. Além disso, assim como praticamente todos os mecanismos estudados nesta disciplina que necessitavam de armazenamento de estado, tal solução provavelmente empregaria o conceito de *soft state*, o que significa que as informações armazenadas seria eventualmente descartadas. A depender dos tempos envolvidos no encaminhamento dos pacotes na rede, é possível que um roteador recebesse uma nova cópia de um pacote de *broadcast* já encaminhado *após* o descarte da informação da inundação anterior, fazendo com que o roteador repetisse a inundação de qualquer forma.

Por todos estes motivos, existe uma segunda abordagem para a implementação da inundação controlada chamada de **encaminhamento baseado em caminho reverso** (ou RPF, do inglês *Reverse Path Forwarding*). Neste método, quando um roteador recebe um pacote destinado ao endereço de *broadcast*, ele verifica o endereço do **nó de origem e sua tabela de roteamento *unicast***. Se a porta pela qual o pacote foi recebido é a mesma utilizada por este roteador para enviar pacotes *unicast* para o nó de origem, então o roteador realiza a inundação. Assim como no método anterior, um roteador pode receber múltiplas cópias do mesmo pacote de *broadcast*, mas apenas uma vez esta recepção ocorrerá através da porta utilizada para enviar pacotes *unicast* para a origem, fazendo com que cada roteador realize a inundação uma única vez.

Claramente, o método do encaminhamento baseado em caminho reverso depende da existência em cada roteador de uma tabela de roteamento corretamente preenchida com informações de roteamento *unicast*. Entretanto, este método não necessita de identificadores nos pacotes ou de armazenamento de estado nos roteadores.

A segunda estratégia genérica para o problema do roteamento *broadcast* — o emprego de árvores geradoras (ou *spanning tree*, em inglês) — assume que o conhecimento prévio destas tais árvores. Formalmente, uma árvore geradora de um grafo G é sub-grafo acíclico e conexo que contém todos os vértices de G. Em termos mais simples, a árvore geradora de um grafo contém todos os vértices do grafo, mas apenas um sub-conjunto de suas arestas de forma que a topologia resultante não contenha ciclos. Aplicando este conceito à realidade das redes de computadores, a árvore geradora define um sub-conjunto dos enlaces da rede de forma que haja exatamente um único caminho entre cada par de nós.

Assumindo que uma árvore geradora tenha sido construída sobre a rede em questão — *i.e.*, que cada roteador saiba exatamente quais as suas portas pertencentes a esta árvore geradora —, o enca-

[1] Este assunto é estudado em mais profundidade no contexto da camada de enlace.

minhamento de um pacote de *broadcast* se torna trivial: ao receber um pacote deste tipo, **o roteador deve replicá-lo para todas as suas portas pertencentes à árvore geradora, exceto aquela pela qual o pacote foi recebido**. A natureza acíclica da árvore garante que cada nó receberá o pacote uma única vez, efetivamente descartando qualquer possibilidade de tempestade de *broadcast* e resultando em um encaminhamento mais eficaz que os métodos baseados em inundação.

A desvantagem nesta abordagem é justamente a necessidade da construção da árvore, embora existam algoritmos relativamente simples para isso. Um algoritmo distribuído simples para isso funciona da seguinte maneira:

1. **Eleição de um nó central**. Através de algum método, os nós da rede escolhem um nó para atuar como "centro da árvore". Esta escolha tem pouca relevância para o funcionamento do algoritmo, bastando que todos os nós da rede considerem o mesmo nó como centro.

2. **Envio de mensagens de *join***. Cada nó da rede "se junta" à árvore geradora enviando uma mensagem *join unicast* destinada ao centro da árvore. À medida que esta mensagem percorre o caminho entre a origem e o nó central, os enlaces utilizados são marcados como pertencentes à árvore. Se esta mensagem chega ao nó central ou a algum nó que já faça parte da árvore geradora naquele momento da execução, seu encaminhamento pode ser interrompido.

3 Roteamento *Multicast*

Em uma comunicação *multicast*, mensagens são destinadas a um **sub-conjunto dos nós de uma rede.** Este sub-conjunto é geralmente próprio — *i.e.*, não inclui todos os nós da rede — já que, do contrário, a comunicação seria *broadcast*. Este conjunto de nós aos quais a mensagem *multicast* interessa recebe o nome de um **grupo *multicast***. O problema de roteamento *multicast*, portanto, consiste em encontrar uma árvore — ou várias — que interconecta todos os membros do grupo *multicast*.

Note que uma árvore geradora atende a este requisito: é uma árvore que interconecta todos os nós da rede — e, portanto, todos os membros do grupo *multicast*, qualquer que seja este grupo. Entretanto, é comum que existam árvores muito menores que uma árvore geradora — *i.e.*, que envolvam menos nós da rede — que ainda interconectem todos os nós do grupo *multicast*. Desta forma, ao contrário de uma árvore geradora, uma **árvore *multicast*** geralmente contém apenas um sub-conjunto (potencialmente pequeno) do total de nós da rede. É importante ressaltar, no entanto, que normalmente não é possível interconectar todos os membros de um grupo *multicast* sem utilizar alguns outros nós que não fazem parte do grupo. Logo, as árvores *multicast* contêm todos os nós membros e, potencialmente, alguns nós não membros.

As abordagens para roteamento *multicast* podem ser divididas em duas famílias: as baseadas em **árvore compartilhada** e as baseadas em **árvores enraizadas na fonte**. Na primeira família, uma única árvore é definida sobre a rede, sendo utilizada para o encaminhamento independentemente da origem de cada pacote. Já na segunda, existirão múltiplas árvores potencialmente diferentes, cada uma utilizada para o encaminhamento de pacotes originados em cada membro do grupo *multicast*.

3.1 Soluções Baseadas em Árvores Enraizadas na Fonte

Nesta família de soluções, uma primeira abordagem é a utilização da chamada *árvore de caminhos mais curtos*. Em um protocolo baseado em estado de enlaces, quando o algoritmo de Dijkstra é executado, o nó que efetua a execução descobre os melhores caminhos completos entre ele e todos os demais nós da rede. Lembre-se que este conjunto de caminhos forma uma árvore que conecta o nó local a todos os demais nós da rede. Assumindo, portanto, que haja um protocolo baseado em estado de enlaces utilizado para roteamento *unicast* na rede, basta que estas árvores de caminhos mais curtos encontradas por cada nó sejam disseminadas por toda a rede. Assim, quando um roteador recebe um pacote de *broadcast*, basta que ele consulte quais de suas portas de saída pertencem à árvore de caminhos mais curtos do nó de origem daquele pacote e faça o encaminhamento por elas.

Outra solução nesta família é o uso do encaminhamento baseado em caminho reverso, exatamente como utilizado no caso do roteamento *broadcast*: ao receber um pacote *multicast*, o roteador verifica este foi recebido pela porta utilizada para encaminhar pacotes *unicast* para o nó de origem; em caso afirmativo, o roteador realiza a inundação do pacote para todas as suas portas de saída, exceto aquela pela qual o pacote foi recebido; caso contrário, o pacote é descartado.

Repare que a solução de encaminhamento baseado em caminho reverso é menos eficiente que a baseada na árvore de caminhos mais curtos, no sentido de que cada roteador pode receber um mesmo pacote múltiplas vezes — embora só encaminhe uma única vez. Por outro lado, a solução baseada nas

árvores de caminho mínimo apresenta maior complexidade, dado que a informação de cada uma das árvores precisa ser disseminada por toda a rede e armazenada em cada roteador. Além disso, repare que ambas as soluções exigem a execução conjunta de um protocolo de roteamento *unicast*.

Por fim, note que ambas as soluções, como descritas até aqui, resultam em um *broadcast* — *i.e.*, os pacotes *multicast*, na prática, chegam a todos os nós da rede. Evitar este tipo de desperdício era justamente o propósito de estudarmos métodos específicos para o roteamento *multicast*. Na verdade, a diferença entre estas soluções e suas equivalentes aplicadas ao roteamento *broadcast* é a possibilidade de uso de um mecanismo adicional ainda não explicado até aqui: a **poda**.

A poda consiste em cortar das árvores *multicast* ramos que não são úteis. Por *não úteis*, entenda-se ramos que não contêm nem membros do grupo *multicast*, nem roteadores necessários para que pacotes alcancem membros.

Em uma solução distribuída, a poda geralmente funciona da seguinte forma. Ao receber um pacote destinado a um grupo *multicast*, um roteador decide por quais portas de saída deve encaminhá-lo. Caso o roteador decida que **nenhuma das suas portas de saída faz parte da árvore e que ele próprio não é membro do grupo *multicast***, então ele não é necessário àquele grupo. Com isso, este roteador envia uma mensagem de poda para seu pai na árvore. Ao receber esta mensagem, o roteador pai remove porta pela qual a mensagem de pode chegou da árvore *multicast*. Note que isso pode fazer com que este próprio roteador pai chegue à conclusão de que agora não é mais necessário à árvore e, portanto, envie uma mensagem de poda para o seu pai. Este processo se repete até que a mensagem de pode alcance um roteador que ainda seja necessário à árvore.

3.2 Soluções Baseadas em Árvores Compartilhadas

Nesta família de soluções, nosso objetivo é determinar uma única árvore compartilhada por todos os membros do grupo *multicast*. Em geral, existem muitas possíveis árvores interconectando todos os membros de um grupo *multicast*. É importante, portanto, definir um critério de qual dessas devemos selecionar.

Assim como o roteamento *unicast* comumente se utiliza de métricas de roteamento para comparar dois ou mais caminhos, podemos, aqui, assumir que cada enlace da rede possui um peso — dado por uma métrica — o que possibilita comparar duas árvores *multicast* através da comparação dos custos totais de cada árvore. Em geral, o custo de uma árvore é dado pela soma dos pesos dos enlaces que a compõem. Assim, chegamos a um critério de escolha de uma árvore *multicast* que parece razoável: dentre as várias disponíveis, optaremos por aquela de menor custo total.

No entanto, dada uma topologia de rede e o conjunto de nós membros do grupo *multicast*, como podemos determinar esta árvore *multicast* de custo mínimo? Este é, na verdade, um problema clássico em computação — mais especificamente, na área de otimização — chamado de **Problema da Árvore de Steiner em Grafos**.

O Problema de Árvore de Steiner em Grafos tem grande aplicabilidade em diversos domínios, incluindo, por exemplo, o projeto de circuitos eletrônicos. Além disso, há grande interesse teórico no problema. Entretanto, foge ao escopo desta disciplina um estudo mais profundo sobre o problema e os algoritmos existentes para solucioná-lo. Para os nossos propósitos, é suficiente entendermos que este problema é classificado como NP-Difícil, o que significa — entre outras coisas — que, ao menos no momento, não são conhecidos algoritmos eficientes (*i.e.*, de tempo polinomial no tamanho da instância) para a sua solução. Dada a sua grande relevância, ao longo dos anos várias boas heurísticas — algoritmos que encontram soluções "boas", mas não necessariamente ótimas — foram criadas para este problema. Mesmo assim, a aplicabilidade destas heurísticas para instâncias práticas de larga escala tipicamente encontradas na área de roteamento *multicast* não é considerada prática, por conta de fatores como: a (ainda) alta complexidade computacional e de troca de mensagens, a necessidade (em certos métodos) de conhecimento sobre a topologia completa da rede e a natureza monolítica de várias destas heurísticas (*i.e.*, pequenas alterações nos participantes do grupo *multicast* demandam a re-execução completa do algoritmo).

Uma solução mais viável computacionalmente são as chamadas **árvores baseadas em nó central**. A ideia é simples e muito similar à do algoritmo distribuído para construção de uma árvore geradora descrito anteriormente: elege-se, dentre os membros do grupo *multicast*, um nó para atuar como nó central; em seguida, cada um dos demais membros do grupo *multicast* envia uma mensagem *unicast* do tipo *join* endereçada ao nó central; os enlaces percorridos pelas mensagens de *join* são adicionados à árvore *multicast* e posteriormente utilizados para o encaminhamento de pacotes no grupo.

De fato, a única diferença deste algoritmo para o de construção da árvore geradora é o fato de que apenas os membros do grupo *multicast* enviam mensagens de *join*. Isso faz com que a árvore resultante possa não incluir todos os nós da topologia.