Implementação e teste do PIM-SM no Network Simulator

António Costa[1], Maria João Nicolau[2], Alexandre Santos[1], e Vasco Freitas[1]

[1] Departamento de Informática,
Universidade do Minho, Campus de Gualtar,
4710 Braga, Portugal
{costa,alex,vf}@uminho.pt
[2] Departamento de Sistemas de Informação,
Universidade do Minho, Campus de Azurém,
4800 Guimarães, Portugal
joao@uminho.pt

Resumo O Multicast é um modelo de comunicação onde a transmissão de dados é feita simultaneamente para múltiplos receptores e por isso particularmente útil para suportar aplicações que envolvem vários intervenientes, de que são exemplo, a vídeo-conferência, o ensino-à-distância, o trabalho cooperativo, etc.
Apesar de ser um tópico com crescente interesse, onde todos os dias surgem novas propostas, ainda não existem soluções passíveis de serem genericamente adoptadas pelos Fornecedores de Serviço IP. Umas das principais razões apontadas para este facto é a complexidade da maioria das soluções, que atrasa de forma drástica o desenvolvimento alargado dos protocolos de encaminhamento multicast. Neste contexto, o recurso à simulação é particularmente útil. A simulação pode ajudar não só a configurar correctamente alguns dos parâmetros em causa, mas também conduzir à proposta de novas soluções.
Neste artigo é apresentada uma implementação do PIM-SM que integramos no Network Simulator (NS). O NS é uma ferramenta muito usada na simulação e teste de protocolos de comunicações. Por sua vez o PIM-SM é um dos protocolos de encaminhamento multicast que está mais desenvolvido na Internet e, como tal, faz sentido estudar as suas limitações com vista à proposta de soluções que as ultrapassem, ou pelo menos, minimizem.

1 Introdução

O Multicast é um modelo de comunicação onde a transmissão de dados é feita simultaneamente para múltiplos receptores. O objectivo é poupar recursos, partilhando-os. Assim, em vez de se transmitirem os dados do emissor para cada receptor separadamente, os mecanismos de encaminhamento multicast estabelecem rotas que partilham os mesmos *links* de forma a transmitirem os pacotes apenas uma vez, enquanto isso for possível.

Este modelo de comunicação é particularmente útil para implementar as aplicações multimédia que envolvem vários intervenientes, por exemplo, aplicações que implementam vídeo-conferência, ensino-à-distância, trabalho cooperativo, etc.

As soluções propostas nesta área baseiam-se na construção de árvores a partir de grafos. Existem basicamente duas razões para este facto: numa topologia tipo árvore os dados podem ser transmitidos simultaneamente ao longo dos seus ramos; e pode ser transmitido apenas um número mínimo de cópias, sendo a duplicação feita apenas quando for necessário, ou seja, quando um ramo se sub-divide.

Existem vários protocolos de encaminhamento multicast que podem ser usados para construir árvores para grupos multicast. Alguns dos mais conhecidos são o PIM-SM (Protocol-Independent Multicast-Sparse Mode)[1], o PIM-DM (Protocol-Independent Multicast-Dense Mode)[2], o DVMRP (Distance-Vector Multicast Routing Protocol)[3], o CBT (Core Based Trees)[4] e o MOSPF (Multicast Open Shortest Path First)[5].

Todos eles têm limitações bem conhecidas, vantagens e desvantagens uns em relação aos outros. O recurso à simulação é particularmente útil no estudo destes protocolos, do seu comportamento e das suas limitações. A simulação pode ajudar não só a configurar correctamente alguns dos seus parâmetros, mas também conduzir à proposta de novas soluções.

De entre todos, o PIM-SM é umas das propostas mais promissoras já desenvolvido por diversos fabricantes. Existem várias razões para esse facto, nomeadamente, o facto de ser independente do protocolo de encaminhamento unicast, ser concebido para redes alargadas e apresentar uma grande flexibilidade por suportar dois tipos de árvores: as árvores partilhadas e as árvores centradas nas fontes.

O Network Simulator (NS)[6], que é um ambiente de simulação largamente utilizado pela comunidade científica no estudo de protocolos de comunicação, não inclui (nem é conhecida qualquer contribuição que inclua) implementações do protocolo PIM-SM, apesar de este protocolo já ter importantes implementações em encaminhadores de produção comercial. Neste artigo apresenta-se então uma implementação do PIM-SM no NS, identificam-se os métodos tornados disponíveis por esta implementação e avaliam-se ainda os resultados da simulação de encaminhamento multicast (PIM-SM) numa rede de *backbone* de razoável dimensão.

O artigo está estruturado da seguinte forma: na próxima secção é feita uma descrição genérica do simulador usado e da forma como ele implementa o encaminhamento multicast. Na secção 3 é feita a descrição da implementação desenvolvida. Começa-se por uma descrição genérica do protocolo PIM-SM, seguida da descrição da implementação da sua estratégia de encaminhamento no NS. Além da estratégia de encaminhamento, que no caso do multicast passa pela forma como se constroem e mantém as árvores, o encaminhamento multicast envolve também a actividade de envio de tráfego multicast em cada nó. A subsecção 3.3 explica as alterações tiveram que ser introduzidas de acordo com o PIM-SM. Por fim, aborda-se a questão da obtenção de resultados da simulação e a sua respectiva análise.

2 Implementação do Encaminhamento Multicast no NS

O NS é um simulador escrito em duas linguagens orientadas a objectos: o C++ e o OTcl. As tarefas muito frequentes (e que por isso exigem rapidez) são normalmente escritas em C++, enquanto que acções de controlo, que exigem flexibilidade, são implementadas em OTcl. O NS inclui assim duas hierarquias de classes que se podem usar conforme o que se julgar mais conveniente. Uma classe numa das hierarquias tem normalmente uma equivalente na outra hierarquia. Uma instância de uma dessas classes reúne os métodos e as variáveis definidos em ambas as linguagens.

No caso do encaminhamento multicast, a estratégia de encaminhamento (responsável pela troca de mensagens de controle entre os nós de forma a construir as árvores multicast), está quase toda implementada em OTcl, enquanto que a componente que processa o envio de tráfego multicast em cada nó está implementada em C++.

A figura 1 mostra como é feito o encaminhamento de pacotes num qualquer nó multicast do simulador NS. O nó é constituído por um conjunto de "classificadores"por onde os pacotes vão passando e sendo "classificados"por forma a chegarem ao destino desejado. A classificação ocorre normalmente por endereço de destino do pacote e porta do pacote. Os destinos podem ser agentes que recebem os pacotes destinados ao próprio nó (endereço de destino= endereço do nó), ou *links* para outros nós.

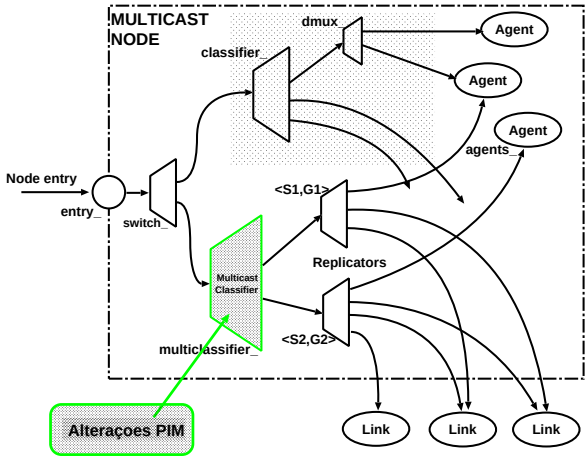


Figura 1. *Encaminhamento num nó NS*

Os pacotes são entregues aos classificadores através da invocação do método **recv(packet)** que eles implementam. Da mesma forma, um classificador depois de "classificar"o pacote, isto é, depois de decidir a quem o entregar, invoca também um método **recv(packet)** nesse objecto. Desta forma o pacote viaja dos agentes para os classificadores e destes para os *links* de saída, e assim sucessivamente até serem entregues aos agentes destinatários.

Os classificadores estão normalmente encadeados conforme ilustrado na figura 1. O ponto de entrada no nó é identificado pela variável **entry_**, e pode ser obtido invocando o método **$node entry**. Este é o primeiro classificador que manipula os pacotes que chegam ao nó. No caso dos nós multicast, este primeiro elemento é um **Classifier/Addr**, referenciado pela variável **switch_**, e que basicamente separa os pacotes unicast dos pacotes multicast com base no endereço destino. A heurística é muito simples: se o primeiro bit do endereço for 0, trata-se de um endereço unicast, se for 1 trata-se de um endereço multicast. Na prática o espaço de endereçamento é dividido a meio, quando a simulação envolve protocolos de multicast.

Os pacotes classificados como unicast são depois entregues a um outro classificador (um **Classifier/Hash/Dest** referenciado pela variável **classifier_**), que os separa de acordo com o endereço de destino, entregando os dirigidos ao próprio nó a um classificador de

portas **Classifier/Port** referenciado pela variável **dmux_**, e os dirigidos a outros nós aos *links* respectivos.

Quanto aos pacotes multicast, passam então por um segundo classificador, do tipo **Classifier/Multicast**, referenciado pela variável **multiclassifier_**, que olha para o endereço de grupo constante nos pacotes e procura encontrar entradas (S,G) (relativas a árvores centradas na fonte S) e/ou (*,G) (relativas a árvores partilhadas) associadas a esse endereço, encaminhando os pacotes por todos os interfaces de saída constantes nessa entrada. O trabalho de replicação por todos os interfaces é feito por um classificador-replicador, do tipo **Classifier/Replicator**, que se limita a manter uma lista de *links* ou agentes locais e entregar uma réplica dos pacotes de dados a cada um deles.

Os classificadores referenciados por **classifier_** e **multiclassifier_**, guardam respectivamente a tabela de encaminhamento unicast e multicast do nó, bem como a lógica de encaminhamento associada.

Cabe aos módulos que implementam a estratégia de encaminhamento, inserir e remover rotas de cada um destes classificadores. A estes fica a tarefa de armazenar as rotas em estruturas de fácil pesquisa, e de as consultar a quando da recepção de um novo pacote de dados, encaminhando-os de acordo com a informação aí constante.

2.1 Estratégias de encaminhamento existentes no NS

A versão actual do simulador NS inclui quatro estratégias de encaminhamento: centralizada (CtrMcast), modo-denso (DM), árvores-partilhadas (ST) e árvores-partilhadas-bidireccionais (BST), implementando cada delas um mecanismo distinto de cálculo das árvores de difusão com semelhanças óbvias com os principais protocolos de multicast actualmente existentes.

A estratégia centralizada utiliza um único agente de controlo (daí a sua designação) que calcula árvores de difusão centradas num qualquer nó da topologia e depois instala as entradas necessárias nas tabelas de encaminhamento de todos os nós seleccionados para fazerem parte da árvore. Podem-se assim construir árvores partilhadas, árvores centradas numa fonte, ou ambas, com os receptores a comutarem de umas árvores para as outras, com relativa facilidade. Esta estratégia tem muitas semelhanças com a estratégia PIM-SM, diferindo fundamentalmente na forma como se calcula e instala a informação de estado nos nós, pois no PIM-SM isso é feito à custa do envio explícito de mensagens de controlo *join* e *prune*. Esta diferença pode não ser aceitável em função da simulação e das medidas que se pretendam efectuar.

A estratégia de encaminhamento DM é uma implementação típica de um protocolo baseado na difusão não selectiva, mais adequada a topologias densas, à semelhança dos protocolos bem conhecidos DVRMP e PIM-DM. O módulo DM pode aliás ser previamente configurado para se comportar como qualquer um desses dois protocolos, através da alteração de uma variável de classe. Por omissão, a estratégia seguida é a do PIM-DM.

A terceira estratégia incluída no simulador é o ST e constrói apenas árvores partilhadas centradas num RP. Trata-se de uma estratégia mais adequada a topologias esparsas, semelhante em alguns aspectos ao PIM-SM. Qualquer nó que se queira juntar ao grupo, envia uma mensagem de *join* em direcção ao RP, instalando entradas nas tabelas de

encaminhamento de todos os nós ao longo do caminho que vai sendo percorrido. Para abandonar o grupo os nós enviam mensagens de *prune* ao RP, desfazendo o estado instalado ao longo do percurso. A implementação actual não reage a alterações dinâmicas do estado da rede durante a simulação.

A quarta estratégia disponível no simulador, designa-se por BST, e constrói árvores partilhadas bidireccionais centradas num RP. Tal como acontece no ST, a árvore partilhada é construída e destruída com base no envio explícito de mensagens de *join* e *prune* dos nós para o RP. Tal como a implementação ST, também esta não reage ao dinamismo da rede ao longo da simulação.

Analisando as quatro estratégias incluídas no NS conclui-se que embora duas delas (a centralizada e a ST) tenham algumas semelhanças com a estratégia de encaminhamento PIM-SM, nenhuma a implementa de facto. Na estratégia centralizada não são enviadas mensagens de controlo e a comutação entre árvores ocorre de forma brusca, enquanto a estratégia ST não constrói sequer árvores centradas nas fontes.

3 Implementação do PIM-SM no NS

3.1 O protocolo PIM-SM

No PIM-SM as árvores de distribuição multicast são construídas e mantidas através de pedidos explícitos de *join/prune* dos vários membros do grupo.

Um novo membro começa por juntar-se a uma árvore partilhada centrada num encaminhador pré-estabelecido designado por RendezVous Point (RP). Assim, numa primeira fase, os dados fluem das várias fontes para o RP e deste para os múltiplos destinos.

Um novo receptor que deseje juntar-se a um grupo multicast G, informa o encaminhador mais próximo (adiante designado por encaminhador nomeado) desse facto através do protocolo IGMP (Internet Group Management Protocol)[7]. Esse encaminhador reage ao pedido de admissão de um membro criando de imediato uma entrada (*,G) na tabela de encaminhamento. Uma vez criada a entrada, o encaminhador envia uma mensagem *join/prune* ao próximo encaminhador na direcção do RP.

Cada encaminhador no caminho até ao RP deve criar ou simplesmente actualizar uma entrada (*,G) na sua tabela de encaminhamento sempre que receber uma mensagem de *join/prune*. Sempre que a entrada (*,G) é criada de novo, a mensagem de *join/prune* deve ser reenviada ao próximo nó no caminho mais curto até ao RP.

Quando o tráfego de dados provenientes de uma mesma fonte ultrapassa um determinado limite, um receptor pode, se assim o desejar, enviar um pedido de junção à fonte, iniciando assim a construção de um ramo de uma árvore centrada nessa fonte. Para isso, o encaminhador nomeado (com membros directamente ligados) coloca uma entrada (S,G) na sua tabela de encaminhamento e envia uma mensagem *join/prune* ao próximo encaminhador, na direcção da fonte. Todos os encaminhadores no trajecto até à fonte devem criar ou actualizar as entradas (S,G) em conformidade.

Um encaminhador ao receber o primeiro pacote proveniente da fonte através da árvore centrada na fonte, faz um *prune* daquela fonte na árvore partilhada, para garantir que não

recebe pacotes duplicados da fonte, via árvore centrada na fonte e via árvore partilhada. Este *prune* só deverá ser feito se o nó pertencer á árvore partilhada.

Para cada grupo multicast, podem coexistir árvores multicast centradas nas fontes, identificadas por pares (S,G), e uma árvore partilhada identificada por (*,G).

3.2 Estratégia de encaminhamento do PIM-SM no NS

Tendo-se concluído que nenhuma das estratégias existentes no NS implementa com suficiente proximidade a estratégia de encaminhamento PIM-SM, partiu-se para a sua implementação de raíz, usando como ponto de partida a estratégia ST.

O primeiro passo foi identificar os módulos e funções que seria necessário escrever do zero e aqueles que poderiam ser simplesmente derivados, aproveitando o facto de o simulador ser orientado a objectos. Pelo que ficou dito na secção 2, o encaminhamento multicast envolve duas actividades distintas que fazem uso de uma estrutura de dados comum que são as tabelas de encaminhamento. Uma das actividades consiste em manter (acrescentar e remover) entradas dessas tabelas, e que sendo tipicamente uma actividade de controlo deve ser escrita em linguagem OTcl e outra que consiste na consulta dessas tabelas sempre que um pacote de dados precisa de ser encaminhado e que pela frequência com que se realiza deve ser implementada em C++. No primeiro caso as actividades são realizadas por um agente de controlo multicast a instalar em todos os nós da topologia enquanto no segundo caso são realizadas pelos classificadores também instalados em todos os nós da topologia.

Dado que a estratégia PIM-SM permite que os nós comutem de árvore de difusão a qualquer momento, as tabelas de encaminhamento podem conter, em simultâneo e para o mesmo grupo G, entradas (*,G) relativas à árvore partilhada e entradas (S,G) relativas a árvore(s) centrada(s) na(s) fonte(s) S. Para além disso e por forma a evitar que a transição de árvore se faça bruscamente (e com perdas de dados), as entradas (S,G) são criadas num primeiro instante num estado inactivo e são mantidas nesse estado até que o primeiro pacote de dados chegue pela nova árvore. Este facto só por si é determinante para a maioria das decisões de implementação tomadas.

Por um lado torna-se óbvio que é necessário rever a estrutura de dados que implementa as tabelas encaminhamento nos nós. A nova estrutura deve permitir a coexistência de entradas (*,G) e (S,G) e, além disso, incluir *flags* de estado nas várias entradas. Como as tabelas de encaminhamento são mantidas nos classificadores, é preciso pelo menos derivar uma nova sub-classe que inclua a nova estrutura de dados. Mas as implicações não são apenas essas, porque como é o classificador que recebe e replica os pacotes de dados dos vários grupos multicast, tem de ser ele também a detectar quando chega o primeiro pacote de dados que completa a transição de árvore de difusão.

Por outro lado o agente de controlo multicast que recebe e envia as mensagens de controlo *join* e *prune* em cada nó, tem de ter pelo menos um novo método (*join-group* <*fonte*>) que possa ser invocado para dar início à comutação de árvore. Este novo método originará mensagens de controlo que podem ser dirigidas não às fontes. Isto faz com que cada agente passe a receber mensagens dos seus vizinhos dirigidas ora ao RP ora a uma das fontes, e tem de reagir de maneira diferente e de acordo com o contexto local a cada uma

delas. Esta constatações sugerem um nível de complexidade que obriga a implementar uma nova classe OTcl para o agente de controlo PIM.

Conclui-se assim pela necessidade de (1) derivar um novo classificador PIM, revendo as estruturas de dados da tabela de encaminhamento e (2) implementar um novo agente multicast PIM. Estas duas decisões de implementação, vão ser analisadas com mais detalhe nas secções seguintes.

3.3 Implementação de um classificador PIM-SM

Tendo-se optado por derivar um novo classificador PIM, para inclusão de novas funcionalidades, a questão que se coloca é qual dos classificadores já existentes usar como classe de base. Os classificadores multicast que existem são apenas dois e derivam ambos do classificador unicast. Um deles é usado apenas na estratégia BST para implementar as árvores bidireccionais que nada têm a ver com o PIM. O outro, designado por *MCast-Classifier*, serve de base a três das quatro estratégias de encaminhamento implementadas no simulador (centralizada, DM e ST) e é por isso o candidato natural.

Como a tabela de encaminhamento é mantida em tabelas de *hashing*, onde cada entrada representa uma rota, houve necessidade de derivar o registo de base por forma a incluir algumas *flags* de caracterização das rotas e temporizadores associados à rota. Uma das *flags* a incluir foi a SPT que classifica uma entrada (S,G) como inactiva (0) ou activa (1). A entrada é sempre criada como inactiva, quando um nó envia a mensagem de *join* à fonte e passa a activa quando chega o primeiro pacote de dados vindo dessa fonte.

Parte das funcionalidades do classificador de base puderam ser aproveitadas, nomeadamente as tabelas de *hashing* e respectivas funções de pesquisa e inserção, bem como os métodos usados na recepção dos pacotes pelo classificador. Apenas um teve de ser rescrito: o método *classify()*, que efectivamente classifica o pacote de dados extraindo do cabeçalho informação necessária para pesquisar na tabela de encaminhamento a rota de saída mais adequada. O algoritmo de classificação foi rescrito por forma a procurar em primeiro lugar entradas do tipo (S,G) relativas a árvores centradas na fonte e só depois entradas (*,G) relativas à árvore partilhada. Se existirem entradas (S,G) ainda inactivas (*flag* **SPT** ainda a zero) deve-se invocar o procedimento que finaliza de forma adequada a comutação da árvore partilhada para uma árvore centrada na fonte.

Como resultado foram escritos os módulos C++ **classifier-pim.cc** e **classifier-pim.h**.

3.4 Implementação de um agente PIM-SM

Relativamente ao agente de controlo PIM, as suas funções são enviar, receber e processar mensagens de controlo *join* e *prune* de forma a construir a árvore partilhada e as árvores centradas nas fontes.

Podemos organizar os estímulos a que o agente tem de responder em duas grandes categorias: (a) o nó que hospeda o agente abandona ou junta-se a um grupo ou a uma fonte desse grupo (b) o agente recebeu mensagens de *join* e *prune* dos agentes seus vizinhos destinados ao RP ou a uma das fontes. Para a primeira situação foram concebidos dois

métodos que podem ou não ser invocados com uma fonte como parâmetro adicional: *join-group* [<*fonte*>] e *leave-group* [<*fonte*>]. Quando não é especificado nenhum parâmetro, significa que se trata de um operação de subscrição/abandono de um grupo na totalidade, e ao especificar a fonte que se trata de um junção/abandono à árvore centrada nessa fonte. Para a segunda situação (b) foram concebidos igualmente dois métodos: *recv-join* e *recv-prune*, que são invocados consoante o agente recebeu uma mensagem de *join* ou de *prune*.

O agente deve reagir a cada invocação destes quatro métodos de acordo com o contexto actual do nó em que se encontra. Podemos também identificar quatro situações distintas, que se podem identificar consultando a tabela de encaminhamento e que são (a) o nó já está na árvore partilhada do grupo (b) o nó já está na árvores centrada na fonte (c) o nó ainda não está na árvore partilhada e (d) o nó ainda não está na árvore centrada na fonte. A reacção do agente em cada um destas situações pode ser uma ou mais das seguintes actividades: não fazer nada, criar, remover ou alterar entradas na tabela de encaminhamento do nó e enviar mensagens de controlo aos nós vizinhos.

O comportamento global resultante, foi alinhado com a norma que descreve o protocolo PIM-SM, e a sua descrição aqui resultaria demasiado exaustiva. A título de exemplo, diga-se que se o agente receber uma mensagem de *join* dirigida ao *RP*, e ainda não se encontra na árvore partilhada, cria uma nova rota (*,G) e reenvia a mensagem de *join* ao nó vizinho que se encontra no caminho mais curto em direcção ao RP. O módulo foi testado exaustivamente, incluindo situações de falha de nós ou de ligações.

O agente PIM-SM implementado constitui uma nova classe OTcl (**PIM.tcl**). Esta classe, à semelhança das estratégias de encaminhamento multicast que integram o NS, deriva da classe **MCastProto**.

4 Teste da Implementação

O teste da implementação do PIM-SM no NS passa pela resposta a duas questões fundamentais: o que avaliar e como obter os dados para efectuar essa avaliação.

Sendo o objectivo do PIM-SM a construção de árvores de difusão, interessará obter dois tipos de indicadores: indicadores da qualidade da(s) árvore(s) geradas e indicadores da sobrecarga introduzida pelo protocolo na geração dessa(s) mesma(s) árvores. Os dados necessários para obter esses indicadores podem ser registados pelo simulador durante a execução da simulação, quer pela activação de opções de *trace* já predefinidas, quer pela inclusão cirúrgica de novos registos de dados no código OTcl produzido. A combinação desses dois métodos melhora a qualidade dos resultados obtidos, e foi esse o procedimento seguido para obtenção dos resultados apresentados em 4.2.

4.1 Indicadores

A avaliação das árvores geradas é normalmente feita recorrendo a gráficos que mostram como varia o custo médio da árvore com o número membros que fazem parte do grupo. Isto porque sempre que um membro sai ou entra do grupo a árvore modifica-se em conformidade e muda de custo. Dado que os membros podem juntar-se ou abandonar o grupo a

qualquer momento, a cardinalidade do mesmo vai mudando ao longo do tempo, podendo haver valores de custo diferentes para o mesmo número de elementos. Esta situação leva a que se tenha de considerar um valor de custo médio da árvore, calculado com base em todos os valores de custo registados para um dado número de membros.

O custo da árvore pode ser medido de várias formas. Uma das formas consiste em contar o número de *links* entre encaminhadores que fazem parte da árvore. No caso do grupo ter mais do que uma árvore (centrada no RP ou centradas em fontes) a contagem do número de *links* deve incluir todas elas, porque são de facto todas necessárias para a distribuição de dados no grupo. Naturalmente que a contagem do número de *links* não tem em conta o tipo de *link* nem o seu custo, mas é um bom indicador da qualidade da árvore.

Outra possibilidade é contar o número de réplicas que são feitas, ao longo da árvore, de cada pacote de dados. Esta medida seria equivalente à anterior se existisse apenas uma árvore, dado que os pacotes seriam replicados exactamente uma vez por cada ramo da árvore, o que corresponderia exactamente ao número de *links*. Quando existe mais do que uma árvore por grupo, o valor do número de réplicas corresponde aos links das várias árvores que são de facto usados durante a transmissão de pacotes. Pode portanto ser diferente do número total de ramos que formam as diferentes árvores. Neste caso o número de réplicas constitui uma melhor métrica do que o número de *links*, uma vez que mede os recursos que estão a ser efectivamente usados.

O NS, já regista no ficheiro de *traces* todo o percurso de todos os pacotes, o que pode ser usado para obter o número médio de réplicas. Quanto à contagem do número de *links* que constituem a árvore, só pode ser feita incluindo no código do módulo **PIM.tcl** um registo de *trace* (recorrendo ao método **annotate** do objecto **McastProto**) sempre que uma entrada é adicionada ou removida da tabela de encaminhamento.

Se para além de anotar os novos *links* se acrescentar no registo o seu custo (a métrica usada pelos algoritmos de encaminhamento), podemos obter um custo da árvore mais realista com base na soma dos custos dos *links* que a constituem.

Em relação ao esforço necessário para construir as árvores de difusão, e dada a natureza do protocolo PIM, pode-se usar como indicador a sobrecarga de mensagens de controlo que os nós da rede tem de suportar para as realizar. As mensagens de controlo são de dois tipos apenas: *join* e *prune*, e podem ser contadas à entrada ou à saída de cada nó. O resultado das contagens apresenta-se normalmente em valores médios por nó em função do número de operações que alteram de alguma forma a árvore de difusão: abandono do grupo, adesão de novos membros e comutações da árvore partilhada para árvores centradas na fonte. O ficheiro de *trace* normal do NS também permite calcular estes valores.

4.2 Simulação e Análise de Resultados

Para testar a implementação do PIM-SM desenvolvida utilizámos a topologia apresentada na figura 2. Esta topologia é constituída por 18 nós. Todos eles têm potenciais receptores (um associado a cada nó), com excepção do RP. Existem duas fontes activas, uma no nó 3 e outra no nó 9.

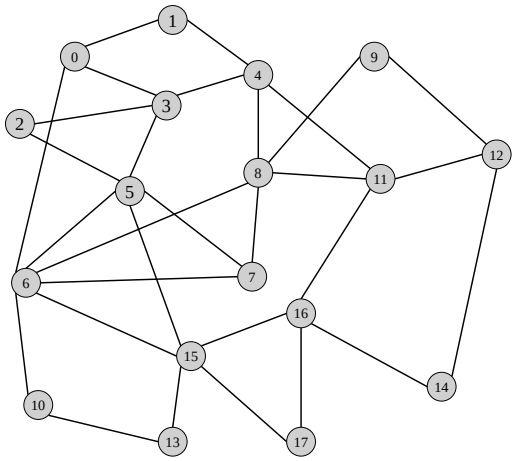


Figura 2. *Topologia de rede utilizada*

Foram simuladas duas situações com o objectivo de comparar as árvores partilhadas com as árvores centradas nas fontes. Na primeira situação todos os receptores se juntam em instantes de tempo distintos à árvore partilhada e nenhum deles comuta de árvore. Na segunda situação todos os receptores, que se juntaram ao grupo, ao fim de algum tempo comutam para uma árvore centrada na fonte associada ao nó 3 e mais tarde comutam também para uma árvore centrada na fonte associada ao nó 9.

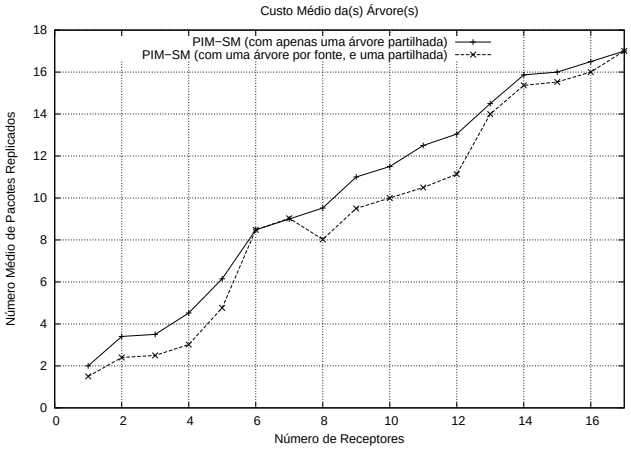
Os resultados obtidos constam dos gráficos apresentados na figura 3.

O gráfico 3(a) mostra o custo das árvores em termos do número de réplicas feito por cada pacote de dados. Este custo é melhor para a situação em que os receptores optaram por comutar para árvores centradas nas fontes. Já o mesmo não acontece se optarmos por medir o custo das árvores através do número de links que as constituem 3(b). Obviamente, neste caso, o número de links envolvidos na situação em que os receptores optam por se juntar a árvores centradas nas fontes é maior.

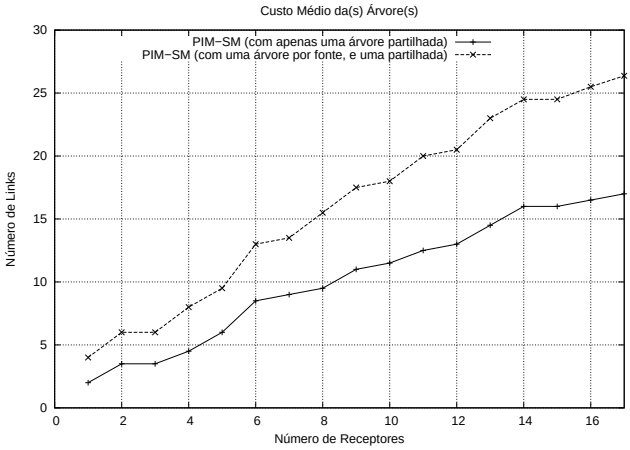
O gráficos 3(c) e 3(d) medem a sobrecarga, medida em termos de mensagens de controle introduzidas, que as operações de *join* e *prune* representam, numa e noutra situação. Na situação em que todos os receptores se juntam a árvores centradas nas fontes existe uma maior sobrecarga, uma vez que novas mensagens são necessárias para completar esse processo.

5 Conclusões

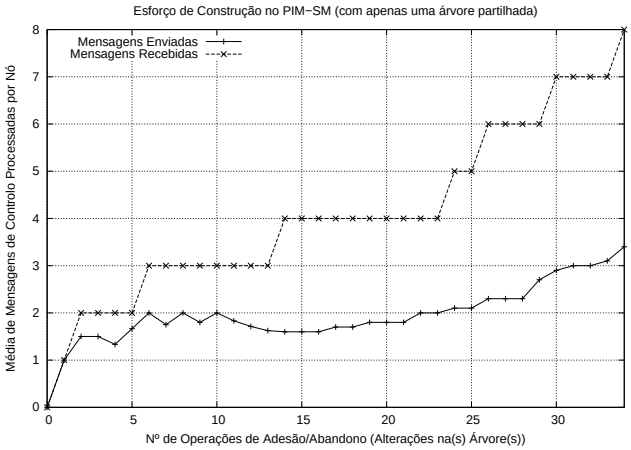
O NS é um ambiente de simulação de eventos discretos que tem sido muito utilizado pela comunidade científica para a simulação, estudo e desenvolvimento de protocolos de comunicação. Este ambiente inclui já suporte para a simulação de protocolos dos níveis de rede e transporte (p.ex. IP, TCP e UDP), bem como para diversos protocolos de encaminhamento e mesmo para aplicações (como p.ex. ftp). Para além do desenvolvimento inicial deste ambiente, na UC Berkeley, LBL, trata-se de um ambiente que é o resultado



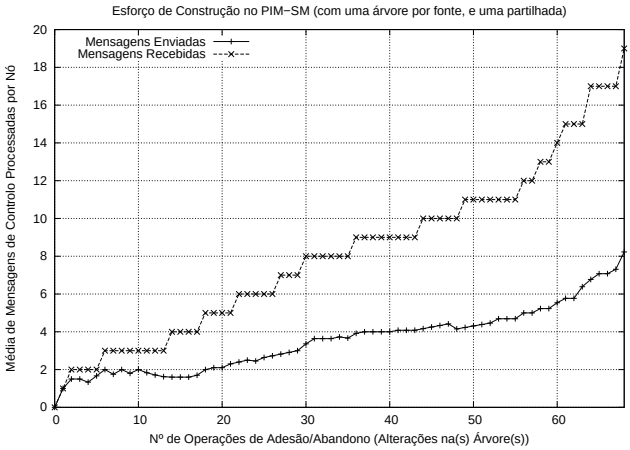
(a) Custo das Árvores (Réplicas)



(b) Custo das Árvores (Links)



(c) Sobrecarga de Mensagens de Controlo



(d) Sobrecarga de Mensagens de Controlo

Figura 3. *Resultados da Simulação*

do esforço contínuo da comunidade científica a nível mundial que tem vindo a fornecer diversas contribuições.

A contribuição apresentada neste trabalho refere-se à implementação e teste de um novo módulo para encaminhamento multicast, usando o Protocol Independent Multicast-Sparse Mode (PIM-SM). Embora o protocolo PIM-SM seja já bem conhecido e bastante utilizado, não existia até à data (pelo menos do conhecimento dos autores) nenhuma implementação deste protocolo para o NS.

Da experiência resultante da implementação aqui apresentada dever-se-á realçar a dificuldade encontrada em incorporar o código desenvolvido, integrando-o simultaneamente nas hierarquias C++ e OTcl, como um novo módulo de encaminhamento multicast para o

NS. Sendo um ambiente de simulação já bastante complexo e com múltiplas contribuições independentes, as interdependências entre os diferentes módulos nem sempre são fáceis de deduzir, até porque alguns dos módulos têm falhas importantes ao nível da respectiva documentação.

Para além desta implementação do PIM-SM, desenvolvida em C++ e OTcl e já integrada na hierarquia de classes OTcl deste ambiente de simulação, realizou-se ainda a simulação de algumas situações de encaminhamento de tráfego multicast num *backbone* com 18 nós e com uma topologia que proporciona múltiplos caminhos entre quaisquer dois dos seus nós. Simularam-se situações em que se utilizam árvores de distribuição multicast, quer árvores partilhadas quer árvores centradas na fonte, e analisaram-se os custos das diferentes árvores criadas, tanto em termos do número médio de réplicas de pacotes como em termos do número médio de *links* que o tráfego multicast atravessa, sendo em qualquer dos casos analisada a sobrecarga introduzida pelas mensagens de controlo geradas pelo próprio protocolo.

Finalmente será importante realçar que a implementação do PIM-SM não se constitui como um fim em si mesmo, mas antes como uma etapa preliminar e essencial no estudo de estratégias de encaminhamento multicast com requisitos de Qualidade de Serviço que o mesmo grupo de investigação da Universidade do Minho, em projecto conjunto com a Universidade de Coimbra, está a realizar no âmbito de um projecto de I&D apoiado pela FCT enquadrado no Programa Operacional Sociedade da Informação[1](POSI).

Referências

1. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol independent multicast-sparse mode (PIM-SM): protocol specification. Request for Comments 2362, Internet Engineering Task Force, June 1998.
2. D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, P. Sharma, and A. Helmy. Protocol independent multicast-dense mode (pim-dm): Protocol specification. Work in Progress.
3. D. Waitzman, C. Partridge, and S. E. Deering. Distance vector multicast routing protocol. Request for Comments 1075, Internet Engineering Task Force, November 1988.
4. A. Ballardie. Core based trees (CBT version 2) multicast routing. Request for Comments 2189, Internet Engineering Task Force, September 1997.
5. J. Moy. MOSPF: analysis and experience. Request for Comments 1585, Internet Engineering Task Force, March 1994.
6. K. Fall and K. Varadhan. *The NS Manual*, Jan 2001.
   URL=http://www.isi.edu/nsnam/ns/ns-documentation.html.
7. W. Fenner. Internet group management protocol, version 2. Request for Comments 2236, Internet Engineering Task Force, November 1997.

[1] Este trabalho foi realizado com o apoio parcial da FCT, através do projecto POSI EEI/10168/98.