PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE ENGENHARIA
FACULDADE DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Desenvolvimento do Protocolo IGMP para Switches Ethernet

ROBERTO PORT DE OLIVEIRA

TRABALHO DE CONCLUSÃO DE CURSO

Prof. Dr. Ney Laert Vilar Calazans
Orientador

Porto Alegre, 13 de dezembro de 2010

SUMÁRIO

DESENVOLVIMENTO DO PROTOCOLO IGMP PARA SWITCHES ETHERNET

RESUMO

Este trabalho tem por objetivo melhorar a qualidade do serviço *multicast* em redes com repetidores passivos que tratam o tráfego multicast como *broadcast*, através da implementação do protocolo Internet Group Management Protocol (IGMP) Snooping em um *switch Ethernet*. O protocolo possui funcionalidades dependentes de *hardware*. Contudo, a implementação desenvolvida pode ser utilizada em outros equipamentos, somente modificando a Hardware Abstraction Layer (HAL) que separa as funções dependentes de hardware do algoritmo principal. A tabela de grupos é armazenada e verificada em hardware para não sobrecarregar o processador do switch com a análise de cada pacote multicast, da mesma forma como ocorre com a tabela de Media Access Control (MAC) tradicional de um switch comum. Ao trocar um switch sem IGMP por um com switch com IGMP, pode-se diminuir significativamente o tráfego multicast em hospedeiros que não têm interesse em receber um determinado conteúdo multicast, reduzindo o consumo de recursos neste para a análise de pacotes, além de diminuir a banda utilizada nos hospedeiros.

Palavras Chave: IGMP, multicast, switch Ethernet.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ARM	*Advanced RISC Machine*
ATM	*Asynchronous Transfer Mode*
BOOTP	*Bootstrap Protocol*
CPU	*Central Processing Unit*
HAL	*Hardware Abstraction Layer*
HTTP	*Hypertext Transfer Protocol*
IANA	*Internet Assigned Numbers Authority*
ICMP	*Internet Control Message Protocol*
IGMP	*Internet Group Management Protocol*
IP	*Internet Protocol*
IPC	*Inter Process Communication*
IPTV	*Internet Protocol Television*
LAN	*Local-Area Network*
MAC	*Media Access Control*
MRD	*Multicast Router Discovery*
OSI	*Open Systems Interconnection*
OSPF	*Open Shortest Path First*
RFC	*Request For Comments*
RISC	*Reduced Instruction Set Computer*
RTP	*Real-time Transfer Protocol*
RIP	*Router Information Protocol*
SSM	*Source Specific Multicast*
TCC	*Trabalho de Conclusão de Curso*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TFTP	*Trivial File Transfer Protocol*
TTL	*Time-to-Live*
UDP	*User Datagram Protocol*
VLAN	*Virtual Local-Area Network*

1. INTRODUÇÃO

Redes de telecomunicação possuem a capacidade de enviar mensagens de uma fonte a um ou mais destinos. Quando uma fonte envia mensagens a exatamente um destino diz-se tratar-se de uma comunicação *unicast*. *Multicast* por outro lado é a entrega de uma mesma informação para múltiplos destinatários, usando a estratégia mais eficiente possível, onde mensagens só passam por um enlace uma única vez e somente são duplicadas quando o enlace para os destinatários se divide em duas direções. Um caso especial de transmissão *multicast* de mensagens é a transmissão *broadcast*, onde uma fonte envia uma mensagem para todos os possíveis destinos alcançáveis [WIT01].

O suporte a *multicast* é difundido nos equipamentos que implementam funcionalidades a partir da camada de rede do modelo Open Systems Interconnection (OSI), porém não em equipamentos da camada de enlace. Com isto, os benefícios de multicast são perdidos ao utilizar em uma rede *switches* que não possuem suporte a multicast e tratam o tráfego multicast como broadcast [SCH08]. Com isto, surgiu a necessidade de se implementar o suporte a multicast em switches para melhorar a qualidade dos serviços que os utilizam, pois em uma rede é mais simples adicionar um switch para aumentar o número de máquinas, do que inserir um novo roteador que possua um *firmware* mais especializado em diversos protocolos e por isso possui poucas portas.

O termo *multicast* é tipicamente associado com IP Multicast, que é a transmissão de um datagrama IP para um "grupo de hospedeiros IP", representado por um conjunto de zero ou mais hospedeiros identificados por um único endereço IP de destino, endereço este que deve ser universalmente reconhecido como multicast. Um datagrama deste tipo é entregue a todos os membros pertencentes a este grupo com a mesma confiabilidade *best-effort* existente em datagramas unicast IP. Ou seja, não há garantia que o datagrama chegue a todos os membros do grupo destino ou que cheguem na mesma ordem em que foram enviados [GAS96]. Este tipo de transmissão é comumente associado a aplicações de áudio/vídeo. Um exemplo é o Real-Time Transport Protocol (RTP) [SCH03]. Outro exemplo são as redes Asynchronous Transfer Mode (ATM), que possuem mecanismos para prover conexões ponto-para-multiponto ou multiponto-para-multiponto.

A associação a um grupo é dinâmica, hospedeiros podem participar ou abandoná-lo a qualquer momento. Não há restrição quanto ao posicionamento geográfico ou quanto ao número de membros em um grupo de hospedeiros. Um hospedeiro pode ser membro de um ou mais grupos ao mesmo tempo. Um hospedeiro não precisa ser membro de um grupo para enviar datagramas a este.

Um grupo de hospedeiros pode ser *permanente* ou *transitório*. Um grupo permanente tem um IP bem conhecido. É o endereço, e não a associação que é permanente. A qualquer momento, um grupo permanente pode ter um número qualquer de membros, até mesmo zero. Endereços *multicast* que não são reservados para grupos permanentes estão disponíveis para atribuições dinâmicas a grupos transitórios, que existem somente enquanto houver hospedeiros membros.

O roteamento de datagramas IP Multicast na Internet é manipulado por roteadores multicast que podem co-existir ou não com gateways Internet. Um hospedeiro transmite um datagrama IP Multicast como um *multicast* na rede local, sendo que este alcança todos os membros vizinhos do grupo de hospedeiros de destino. Se o datagrama possui um Time-to-Live (TTL) maior que um, o roteador *multicast* associado a esta rede tem a responsabilidade de repassar este datagrama a todas as outras redes que possuam membros deste grupo destino. Nas redes alcançáveis (TTL), o roteador *multicast* a elas associado completa a entrega transmitindo o datagrama como um *multicast* local aos membros do grupo.

O protocolo de gerenciamento de grupos da Internet (*Internet Group Management Protocol* ou IGMP) é usado por hospedeiros para reportar sua participação em grupos multicast a roteadores vizinhos. Como o *Internet Control Message Protocol* (ICMP), IGMP é uma parte integral do IP. É um requisito básico de implementações a todos os hospedeiros que desejem enviar e/ou receber pacotes *multicast*.

1.1 Motivação

Este trabalho envolve diversas disciplinas estudadas ao longo do curso de graduação do Autor, além de tecnologias utilizadas em empresas de telecomunicações, o que proporciona um diferencial para ingressar neste ramo no mercado de trabalho.

A primeira utilização de *multicast* em redes locais foi feita com a finalidade de descobrir recursos, conforme pode ser observado nos protocolos de roteamento *Routing Internet Protocol* (RIP) e *Open Shortest Path First* (OSPF) [MAL98] [MOY98]. Se um roteador OSPF quer encontrar outros roteadores OSPF que se encontram na rede, ele simplesmente faz o multicast de um pacote "hello" para o endereço convencional "todos os roteadores OSPF". Este endereço é fixo, arbitrado pela *Internet Assigned Numbers Authority* (IANA), sendo que pode ser previamente codificado em implementações do protocolo OSPF. Existe uma maneira alternativa para fazer isto em redes que não possuem suporte para broadcast: para enviar um pacote "hello" a todos vizinhos, os roteadores devem ser configurados com uma lista de endereços dos mesmos. O envio de pacotes *multicast* possibilita que o roteador descubra esta lista com apenas uma transmissão, minimizando incômodos ao administrador da rede.

O uso de multicast não está limitado a protocolos de roteamento. Tome-se como exemplo o protocolo *Bootstrap Protocol* (BOOTP) [CRO85], que é utilizado para carregar estações com um valor inicial de parâmetros do sistema. Ao enviar um pacote multicast ao endereço "todos os servidores bootstrap", a estação automaticamente descobrirá o servidor local BOOTP.

Transmissões multimídia aproveitam as vantagens da alta capacidade de processamento de estações de trabalho. Sinais de áudio e vídeo são digitalizados e comprimidos em uma máquina origem, que as envia como uma sequência de pacotes User Datagram Protocol (UDP) para um endereço de grupo. Isto é, um endereço IP que designa a comunidade de hospedeiros que pretende e tem permissão de acesso ao conteúdo em questão. As estações receptoras se "sintonizam" pedindo para receber pacotes enviados a este grupo. Ao receber estes dados, decodificam-os e estes são apresentados aos usuários das estações.

A inovação maior associada a este trabalho se dá com a implementação do protocolo IGMPV3 em

switches, algo ainda pouco comum no mercado brasileiro. Pode-se citar como benefícios associados a serviços baseados em multicast:

- a diminuição de tráfego na rede;
- a descoberta automatizada de recursos;
- a distribuição de canais para vários assinantes;
- acesso a Bolsa de Valores e videoconferências;

Estes benefícios se devem ao fato do multicast usar apenas uma transmissão para enviar pacotes com informação para um determinado grupo, este pacote somente é replicado ao chegar aos seus destinatários, diminuindo o tráfego da rede e consumindo menos banda e recursos. A distribuição de canais para diversos assinantes pode ser feita usando a transmissão multicast. Cada cliente pode entrar em um grupo multicast para receber conteúdo de um determinado canal, esta aplicação é chamada de Internet Protocol Television IPTV, em [WEI08] encontra-se um trabalho que usa o IGMP em switches Ethernet para obter os benefícios do multicast e melhorar este serviço ao consumidor.

O IGMPV3 possibilita o uso de *Source Specific Multicast* (SSM), o que facilita o gerenciamento de endereços (vários conteúdos com um único grupo), provê maior segurança, simplifica o plano de controle, possibilita rastreamento dos receptores e a redução da tabela de roteamento multicast.

1.2 Objetivo

O presente trabalho teve por objetivo o desenvolvimento, em *software*, do protocolo IGMP Snooping. A implementação do protocolo é uma aplicação executando no sistema operacional Linux [TAN08], em espaço de usuário, que utiliza uma camada de abstração de acesso ao hardware (Hardware Abstraction Layer ou HAL) para configurar o *switch* de uma placa de desenvolvimento utilizando um *driver* específico [COR05]. O processo de desenvolvimento do protocolo IGMP foi feito utilizando a linguagem C, nos sistemas operacionais Linux Ubuntu e Debian, utilizando a versão 2.6 do *kernel* destes sistemas operacionais (SO). O resultado foi um *switch* capaz de gerenciar grupos *multicast* utilizando as versões IGMPV1 e IGMPV2 do protocolo IGMP. Além disto, foi também alvo do trabalho a implementação da versão IGMPV3 deste protocolo para esta fazer parte do mesmo *switch.*

Este trabalho propõe implementar o protocolo IGMP para uso em switches Ethernet com Linux embarcado, resultando em uma rede com menor consumo de banda dos clientes ao utilizar protocolos *multicast*, além de diminuir a carga de trabalho destes clientes e do roteador da rede em questão. Para tal, utiliza-se no desenvolvimento diversas RFCs relacionadas ao IGMP, estudando as mesmas. Também, utiliza-se documentação de sites de fabricantes de switches e artigos técnico/científicos resultantes de trabalhos similares. O algoritmo é independente de hardware, sendo necessário para seu porte somente a adaptação/substituição da HAL por outra adequada a qualquer plataforma alvo diferente da citada e usada aqui. Para testar as funcionalidades do algoritmo uma arquitetura de rede e um simulador foram criados. Ambos têm por finalidade avaliar o desempenho do algoritmo com um maior número possível de clientes.

1.3 Estrutura do Restante do Documento

O Capítulo 2 apresenta o ambiente e os recursos necessários para o desenvolvimento do projeto, desde o sistema operacional até a parte de *hardware e software* necessária para testes simulando ambientes reais. O Capítulo 3 apresenta a arquitetura da plataforma de desenvolvimento, sua configuração para executar o protocolo, além da implementação da HAL e dos testes realizados. O Capítulo 4 apresenta o protocolo IGMPV1 e seu desenvolvimento, detalhando a estrutura de dados utilizada. O Capítulo 5 apresenta o protocolo IGMPV2 e seu desenvolvimento, mostrando as diferenças e adaptações realizadas no IGMPV1. O Capítulo 6 apresenta o desenvolvimento do simulador de hospedeiros IGMP e switch multicast, detalhando o funcionamento do simulador, as características do sistema e como se dá a ligação entre o simulador e a implementação do protocolo. Finalmente, o Capítulo 7 apresenta a evolução do algoritmo, as conclusões e trabalhos futuros.

2. AMBIENTE DE DESENVOLVIMENTO E TESTES

Para o desenvolvimento do trabalho utilizou-se, na parte de *software*, sistema operacional Linux [MAS07], nas distribuições Ubuntu e Debian. No ambiente de programação, foi utilizado o Eclipse, software livre muito difundido. Para o controle de versão dos códigos gerados, foi utilizado o SVN – *Subversion*, juntamente com o Eclipse, este último servindo como intermediário com o SVN para gerenciar as versões dos códigos produzidos.

Para análise de pacotes criados pelo protocolo e pelas demais entidades que produzem tráfego na rede, utilizou-se o programa Wireshark [WIR10]. Este foi utilizado para analisar protocolos de rede, realizando o papel de *sniffing*, ou seja, analisar continuamente os pacotes que trafegam na rede, informando suas características gerais e detalhes do protocolo utilizado para encapsulamento.

Foi utilizada a plataforma de desenvolvimento KS8695P da MICREL, a fim de analisar o protocolo desenvolvido em uma situação real, similar a aplicações reais em que este protocolo pode ser usado. A plataforma de desenvolvimento trabalha com Linux embarcado. Assim, o Linux necessário para a plataforma é compilado na distribuição Debian incluindo o protocolo desenvolvido, sendo em seguida carregado na placa, executando assim como um *daemon* nativo do Linux.

A plataforma de desenvolvimento apresenta em sua arquitetura, um switch com quatro portas Ethernet, uma porta Ethernet para controle e uma porta serial para controle da plataforma de hardware. Foi utilizado o software MINICOM para a comunicação serial. Além disso, foi utilizada uma arquitetura de rede para testes, usando o software Xorp para emular um roteador multicast e o software VLC para distribuir o conteúdo e gerar as requisições de grupos dos clientes.

O protocolo IGMP foi desenvolvido em dois ambientes, o primeiro usando a plataforma de desenvolvimento KS8595P da Micrel, que possui um switch Ethernet com suporte a IGMP. O segundo foi desenvolvido com um switch Ethernet e sua tabela de grupos simulados em software. O algoritmo principal é o mesmo nos dois ambientes, o que os diferencia é a camada de abstração de hardware HAL que separa as funções dependentes de hardware do algoritmo principal. A Figura 1 a seguir mostra a relação entre os dois ambientes.

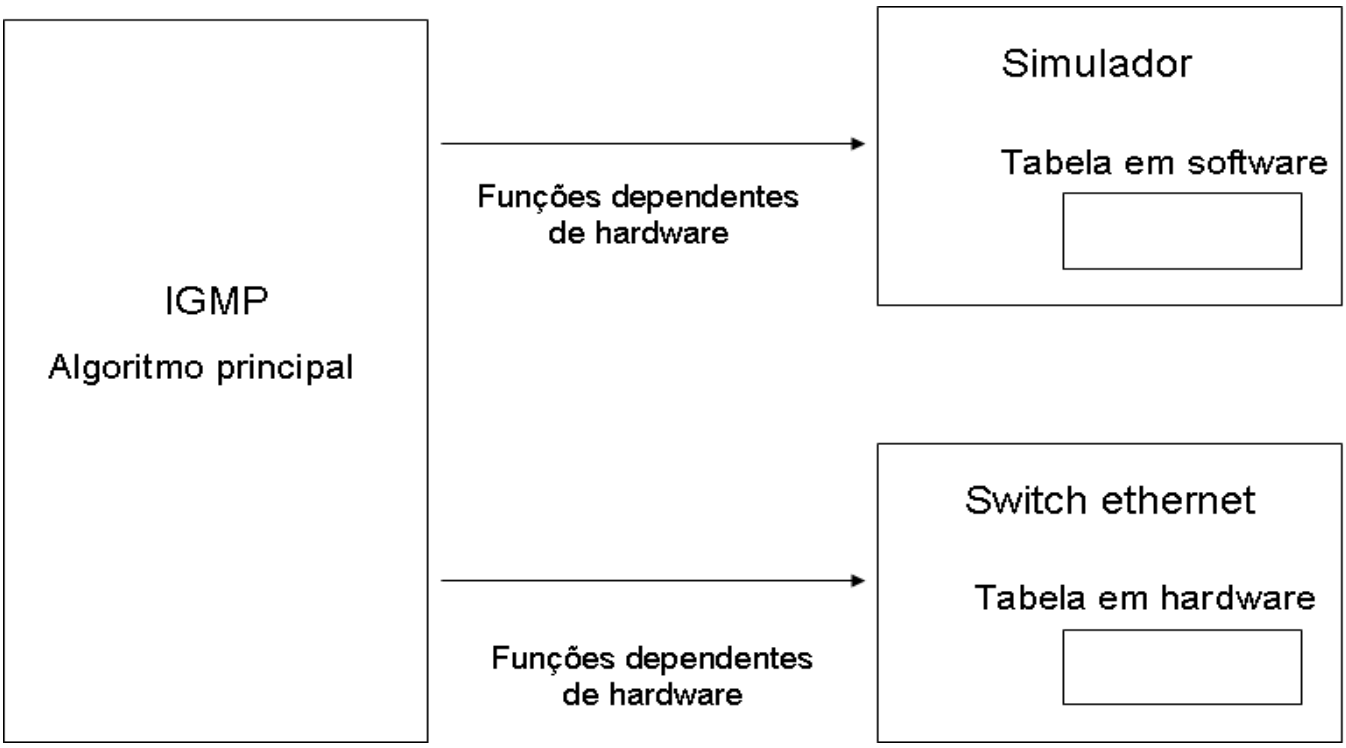

Figura 1 - Relação do algoritmo com os ambientes de simulação e de implementação em hardware.

2.1 Arquitetura do Ambiente de Testes

Visando a realização de testes para verificação da implementação, foi necessário um ambiente com um roteador *multicast*, um servidor e clientes para o conteúdo *multicast*. O roteador utilizado foi o Xorp [XOR10], um software livre que executa protocolos de roteamento, podendo ser usado para emular um roteador em um sistema operacional Linux em um computador pessoal comum. O servidor é um computador com sistema operacional Linux executando o software VLC [VID10]. Este aplicativo é usado ao mesmo tempo como cliente e servidor para diversos tipos de conteúdos multimídia e para distribuir vídeos utilizando o protocolo RTP. Os clientes são computadores solicitando o conteúdo *multicast* para diversos grupos através do software VLC. A Figura 2 mostra a arquitetura geral do ambiente de teste.

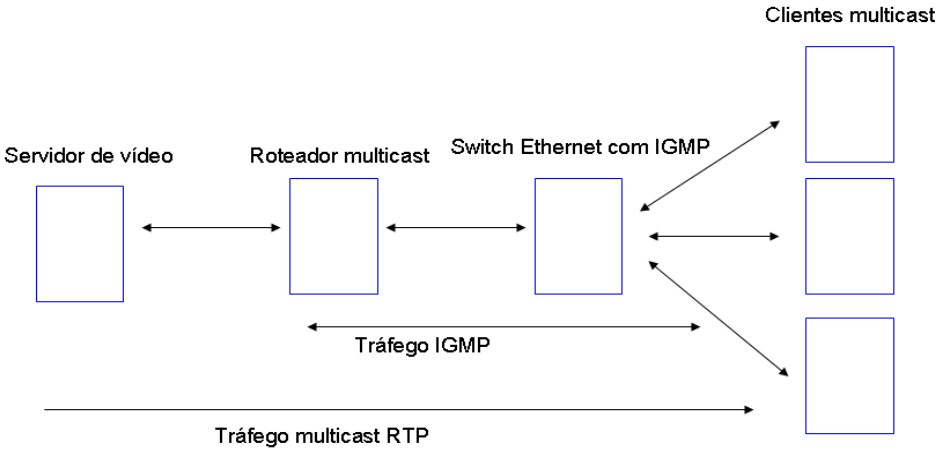

Figura 2 - Arquitetura da rede para realização de testes da implementação.

2.2 Utilizando o Xorp

O software Xorp foi utilizado para simular um roteador, ao invés de usar um roteador comercial somente para executar o protocolo IGMP. Com um computador comum executando o sistema operacional Linux na distribuição Debian, configurou-se o Xorp em suas duas interfaces Ethernet.

A configuração do software é feita através de um arquivo que descreve os protocolos a serem executados nas interfaces de rede. No escopo deste trabalho somente configurou-se o software para fazer o roteamento multicast e usar o IGMP para o controle dos grupos multicast. A Figura 3 mostra a configuração do IGMP nas interfaces *eth0* e *eth1* através de arquivo de entrada utilizado pelo Xorp.

```
protocols {
    igmp {
        disable: false
        interface eth0 {
            vif eth0 {
                disable: false
                version: 1
                /* enable-ip-router-alert-option-check: false */
                query-interval: 2
                /* query-last-member-interval: 1 */
                /* query-response-interval: 10 */
                /* robust-count: 2 */
            }
        }
        interface eth1 {
            vif eth1 {
                disable: false
                version: 1
                /* enable-ip-router-alert-option-check: false */
                query-interval: 2
                /* query-last-member-interval: 1 */
                /* query-response-interval: 10 */
                /* robust-count: 2 */
            }
        }
    }
}
```

Figura 3 - Parte do arquivo de configuração do Xorp, mostrando a configuração das interfaces *eth0* e *eth1* para operar com IGMP.

Além da configuração através de arquivo, o Xorp possui um *shell*, como os de roteadores comercias. Neste pode-se mudar configurações mesmo depois que o roteador já esteja executando. No contexto do trabalho este *shell* é usado para mudar as variáveis de configuração do IGMP e a versão do protocolo. Assim, se pode testar o algoritmo desenvolvido em diferentes configurações, utilizando diferentes versões do IGMP. A Figura 4 mostra a mudança de versão do IGMP em uma das interfaces Ethernet através do *shell* do Xorp.

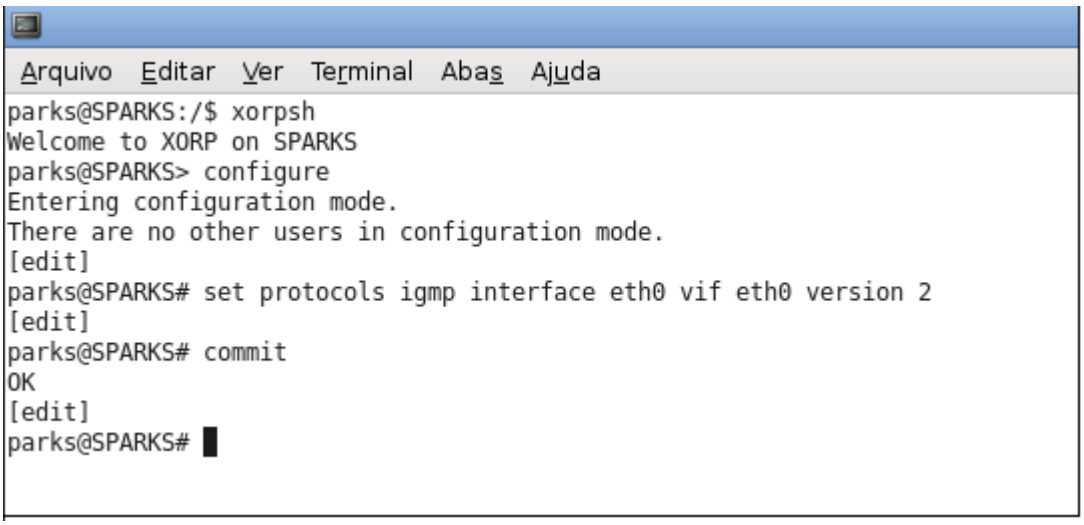

Figura 4 - Shell de configuração do Xorp mostrando uma mudança de versão de protocolo IGMP durante a operação do simulador de roteador.

2.3 Utilizando o VLC

O software livre VLC foi utilizado para gerar as requisições de hospedeiros para sua inclusão em grupos multicast e para gerar conteúdo para um determinado grupo. Este software facilitou os testes da implementação, pois gera os pacotes IGMP necessários de acordo com os grupos escolhidos e de acordo com a versão do IGMP utilizada no Linux do hospedeiro.

Clientes são computadores com sistema operacional Linux distribuição Ubuntu. Neles executa-se o VLC para receber um vídeo de um grupo multicast qualquer. A Figura 5 mostra a configuração do VLC para gerar um cliente IGMP para um determinado grupo.

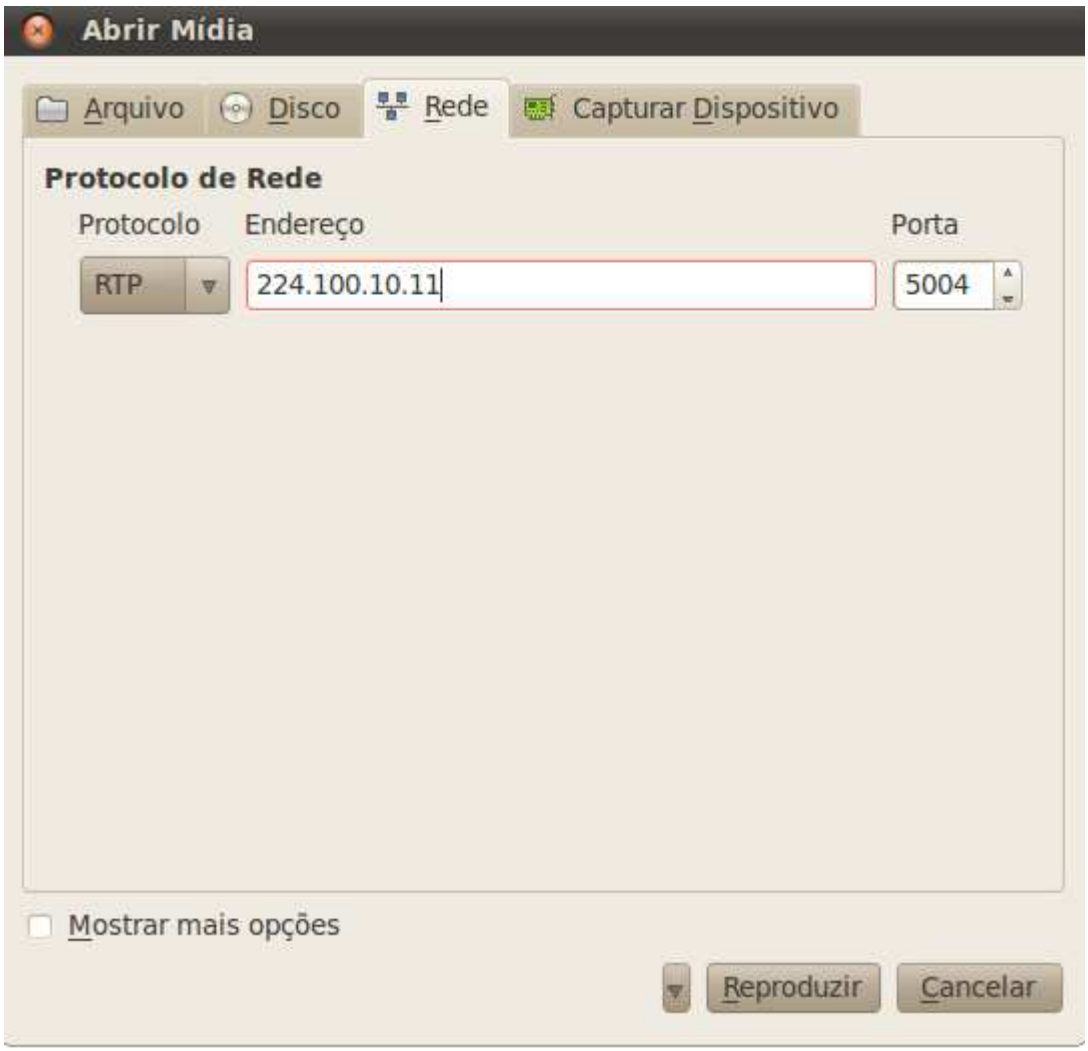

Figura 5 - Configuração do cliente multicast usando o VLC.

Nesta configuração é usado o protocolo RTP e sua porta padrão para a recepção do vídeo que deve ser o mesmo usado pelo servidor. O importante nesta configuração é o endereço multicast que é usado pelo aplicativo para gerar uma chamada ao sistema operacional a fim de incluí-lo no grupo. Assim o sistema operacional gera as requisições necessárias usando o protocolo IGMP como será visto no Capítulo 4.

O servidor funciona do mesmo modo que os clientes, porém o VLC é configurado para gerar *streams* de vídeo para um grupo multicast. Esta configuração é feita através de um comando em um shell do Linux, no qual são passados parâmetros. Destes, os principais são o arquivo de vídeo a ser usado para o stream, o grupo multicast para distribuição e o TTL dos streams. Como no cliente o protocolo RTP é usado e a porta padrão 5004. Um exemplo do comando utilizado e seu resultado são mostrados na Figura 6 e na Figura 7, respectivamente.

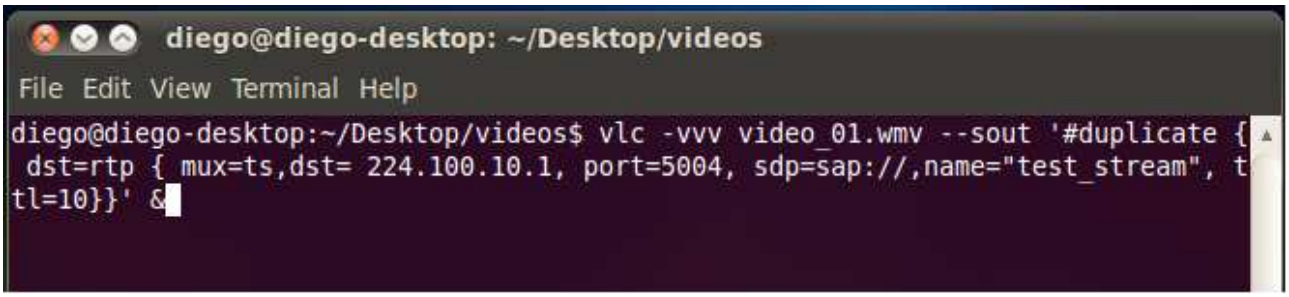

Figura 6 - Exemplo de comando para gerar um servidor multicast com o VLC.

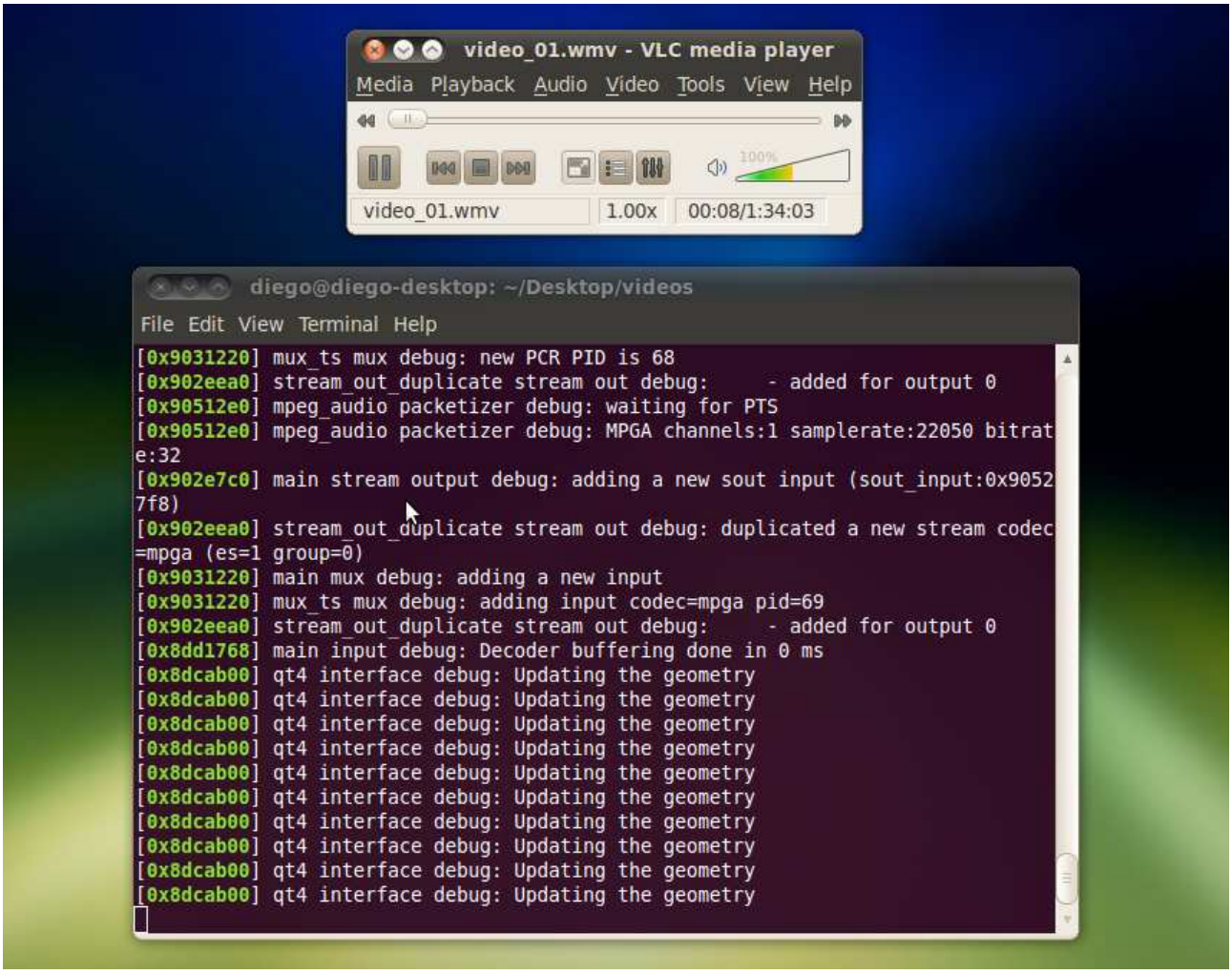

Figura 7 - VLC gerando streams multicast.

Podemos ver que o software faz a decodificação do vídeo gerando como resultado os pacotes para o grupo multicast com os frames do vídeo, que poderão ser visualizados nos clientes do grupo multicast.

2.4 Linux embarcado

A plataforma de desenvolvimento utilizada possui um sistema Linux embarcado, que deve ser compilado a partir de código fonte fornecido pelo fabricante. Para isso é necessário um compilador cruzado para a arquitetura ARM. Este compilador também é disponibilizado no site do fabricante da plataforma. Com o compilador cruzado instalado compila-se o Linux especialmente modificado para a plataforma de hardware. Para monitorar e verificar sua correta execução a partir de um computador hospedeiro instala-se o software

livre de comunicação serial MINICOM, para gerar um terminal emulado do Linux que está executando na plataforma, usando para tanto uma conexão do hospedeiro à plataforma via porta serial. Um exemplo de interface fornecida pelo MINICOM para controlar a plataforma a partir do hospedeiro aparece na Figura 8.

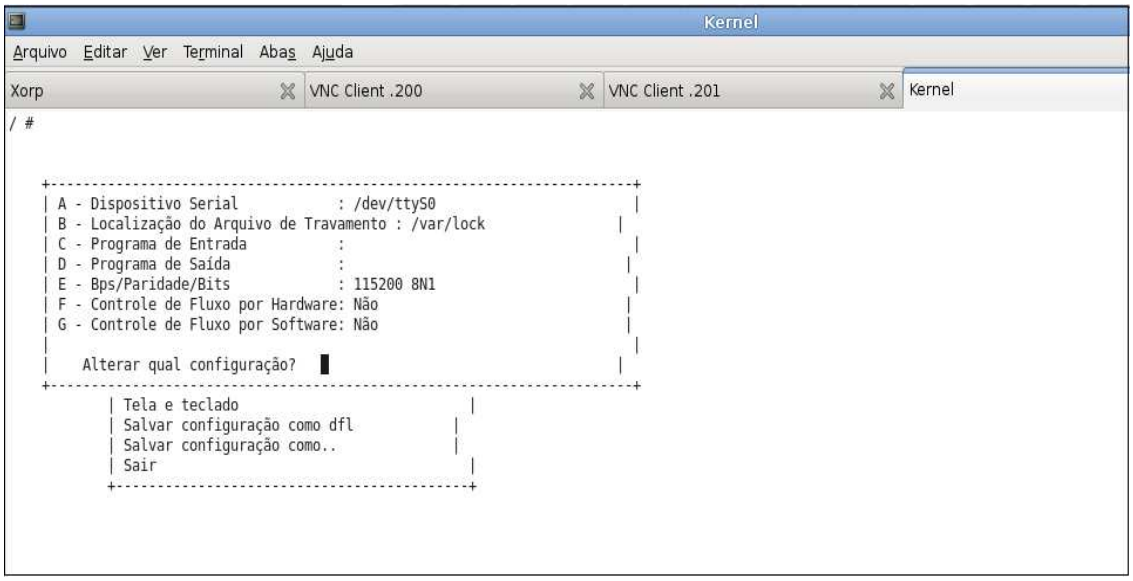

Figura 8 - Configuração da comunicação serial no MINICOM.

Ao compilar o Linux para a plataforma, gera-se uma imagem que pode ser carregada pelo bootloader da plataforma, o *u-boot*, que já vem instalado de fábrica. Este software carrega a imagem do Linux através do protocolo TFTP. Desta maneira, foi necessário instalar um servidor TFTP no hospedeiro. A Figura 9 mostra um exemplo de interação em que se faz a carga do Linux na plataforma via TFTP.

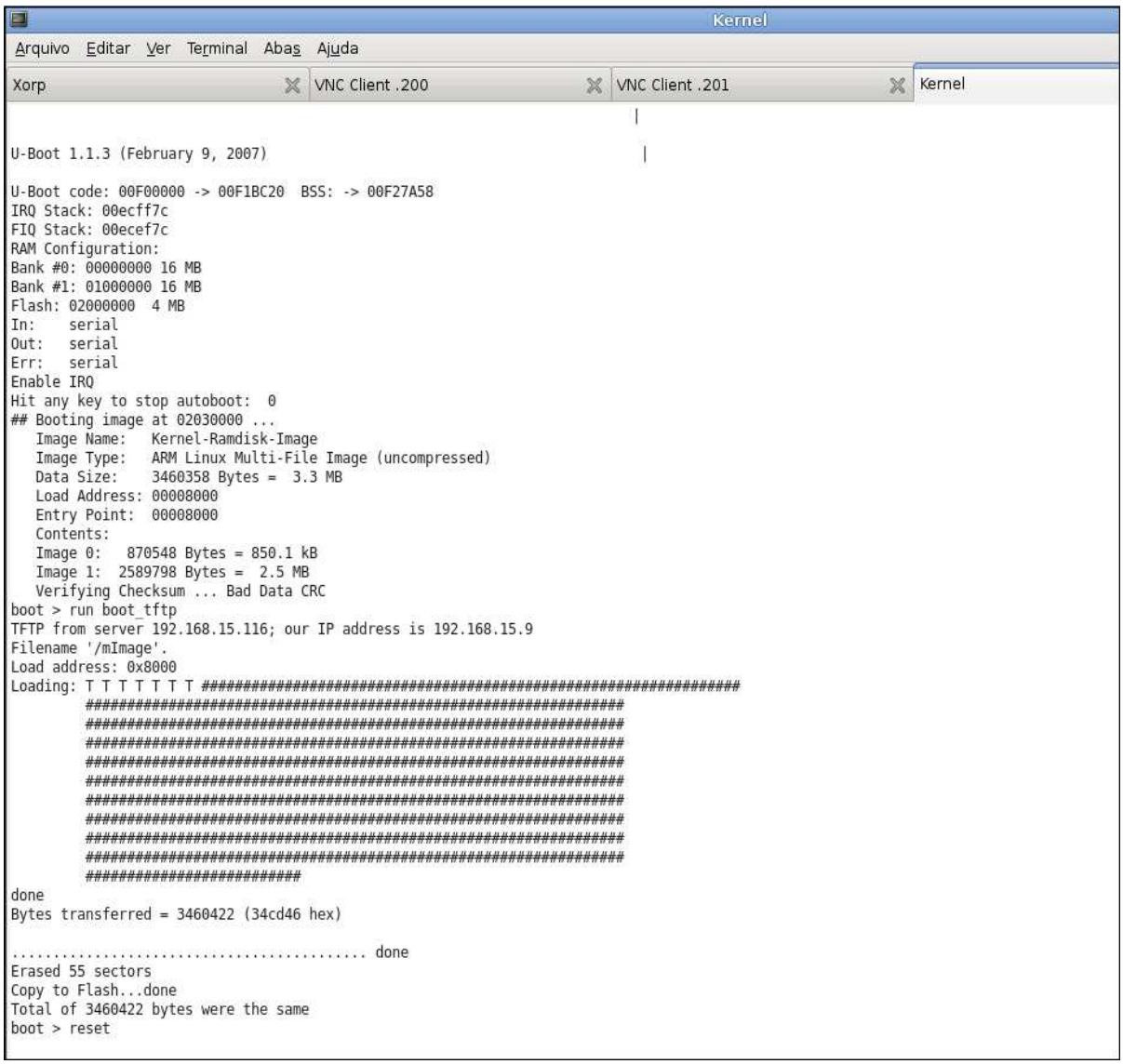

Figura 9 - Boot da imagem do Linux embarcado através de TFTP.

Além do servidor TFTP também foi necessário criar um servidor HTTP no hospedeiro para carregar o código executável do código fonte desenvolvido, pois caso contrário seria necessário criar novamente a imagem toda vez que o código fosse atualizado e isso iria exigir muito tempo, atrasando e dificultando o desenvolvimento. Com o servidor HTTP carrega-se o novo executável rapidamente no Linux embarcado em execução com um programa desenvolvido por um dos membros do grupo de pesquisa do Laboratório da empresa PARKS na PUCRS, o programa é chamado YAWGET. Ele faz a transferência de arquivos através do protocolo HTTP. A Figura 10 mostra um exemplo de uso do YAWGET.

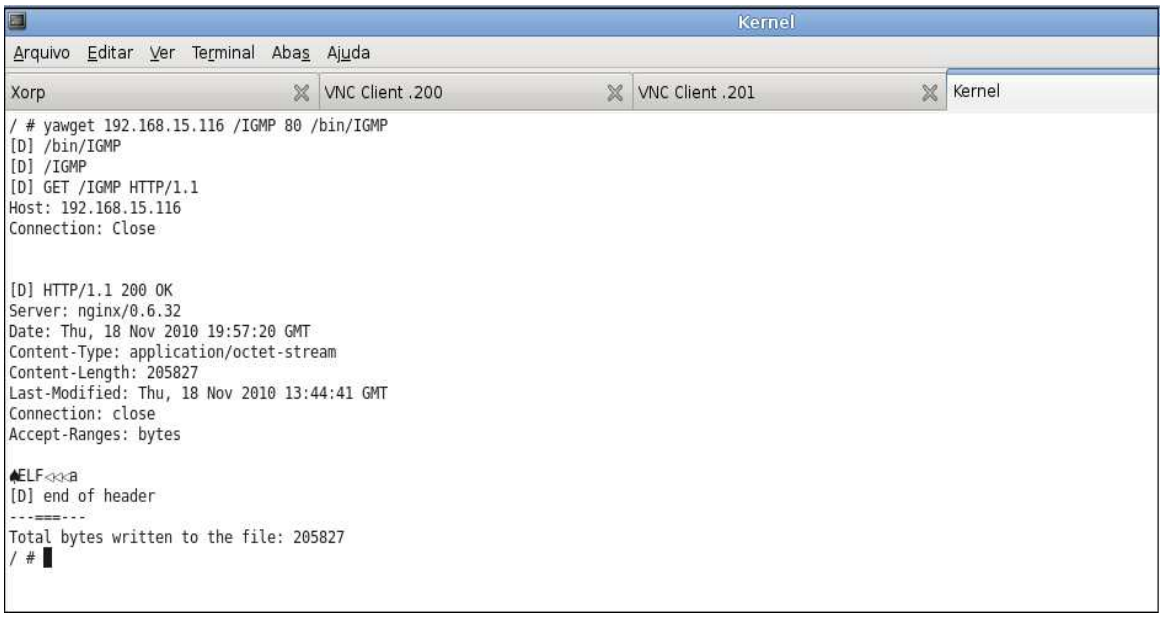

Figura 10 - Carregando o executável para o Linux embarcado, através do software YAWGET.

Outro software bastante utilizado neste trabalho foi o Wireshark, usado para verificar a troca de pacotes entre clientes e servidor e para verificar se o switch da plataforma faz o encaminhamento dos pacotes corretamente.

2.5 Ambiente de testes e desenvolvimento

Foi criada uma rede para o desenvolvimento e realização de testes do protocolo IGMP. Esta rede possui três clientes que são computadores com o sistema operacional Linux distribuição Ubuntu. Estes computadores não possuem monitores acoplados individualmente e ficam à disposição para a realização dos testes. Eles são acessados pela rede através do software livre VNCVIEWER, utilizado para criar conexões de forma remota para um ambiente gráfico em sistemas Linux (entre outros). Estes clientes estão ligados à plataforma de desenvolvimento. Esta por sua vez está conectada ao roteador e a rede. O roteador é um computador rodando Xorp, e a plataforma está ligada a rede local do laboratório de desenvolvimento de forma que pode receber o Linux embarcado e o executável do protocolo via esta última rede. O roteador está ligado ao servidor, um computador com VLC configurado como servidor gerando tráfego para um grupo multicast. A Figura 11mostra a arquitetura completa com seus componentes e suas conexões.

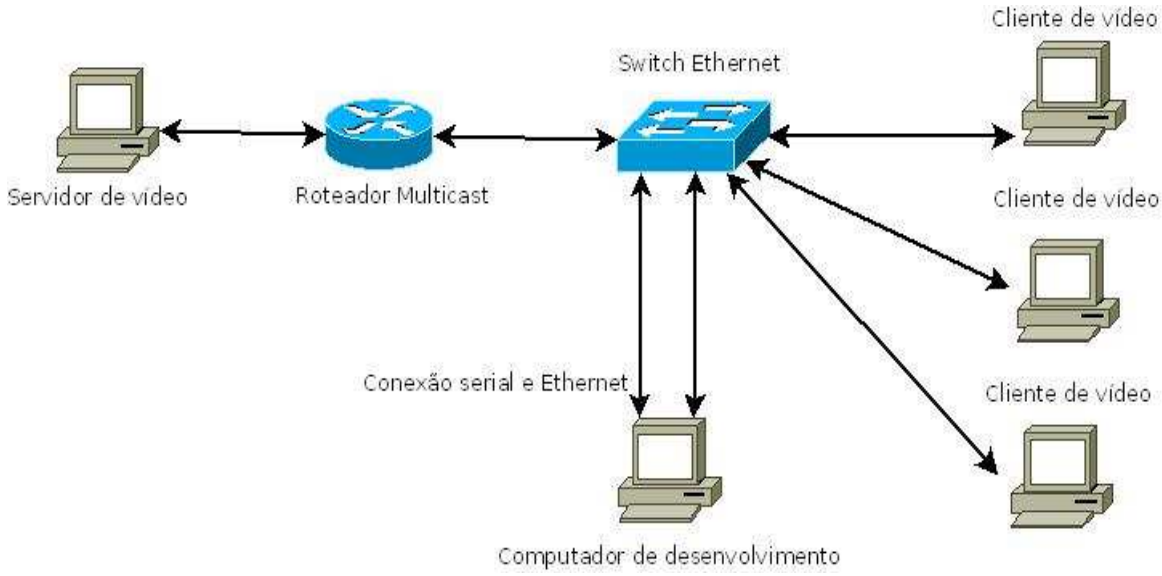

Figura 11 Arquitetura completa do ambiente de desenvolvimento da implementação do protocolo IGMP.

O servidor de vídeos gera os pacotes de dados para os grupos multicast. O roteador somente repassa esses pacotes se existir algum cliente no grupo destino do pacote. O switch tem a tarefa se repassar os pacotes somente para os clientes que fazem parte do grupo. Os clientes fazem as requisições de entrada e saída de grupos multicast e renderizam os vídeos recebidos usando o VLC. A conexão serial do computador de desenvolvimento serve como um terminal emulado do sistema operacional embarcado no switch. A conexão Ethernet serve carregar a imagem do Linux e os executáveis dos aplicativos desenvolvidos utilizando os servidores TFTP e HTTP configurados no computador de desenvolvimento. A Figura 12 mostra uma visão do ambiente de desenvolvimento.

Figura 12 – Visão do ambiente de desenvolvimento.

2.6 Controle de versões

O controle de versões é necessário devido ao fato de existirem diversas pessoas na equipe trabalhando no mesmo código. Um servidor central foi configurado para utilizar o SVN. Assim, cada membro da equipe realiza e recebe atualizações no código em desenvolvimento. O software eclipse com o *plugin* SVN foi usado para fazer as atualizações de maneira mais simples, pois pode empregar interface gráfica conforme ilustra a Figura 13. Assim, pode-se facilmente verificar quem atualizou os arquivos com o comentário do que foi modificado. No caso de problemas, pode-se voltar a versões anteriores para verificar o que foi modificado erroneamente.

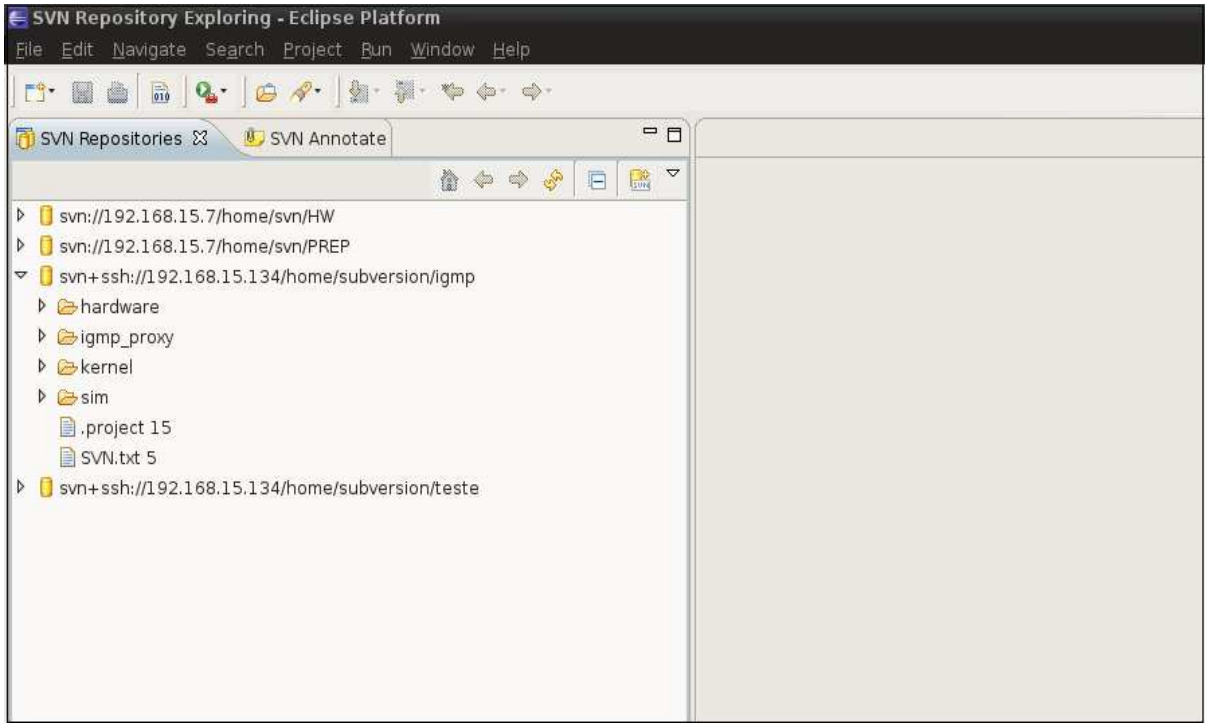

Figura 13 – Exemplo de visualização do repositório SVN do IGMP utilizando a interface gráfica do Eclipse.

3. PLATAFORMA DE DESENVOLVIMENTO

A plataforma utilizada para o desenvolvimento do protocolo IGMP é a KS8695P da Micrel [MIC05]. Esta plataforma possui um switch Ethernet com suporte a IGMP. No site do fabricante foi obtido o Linux modificado especialmente para a plataforma, além do compilador cruzado para a arquitetura ARM e os *datasheets* necessários para realizar as configurações e manipulações do switch através de seus registradores. A Figura 14 mostra a arquitetura da plataforma de desenvolvimento em detalhe.

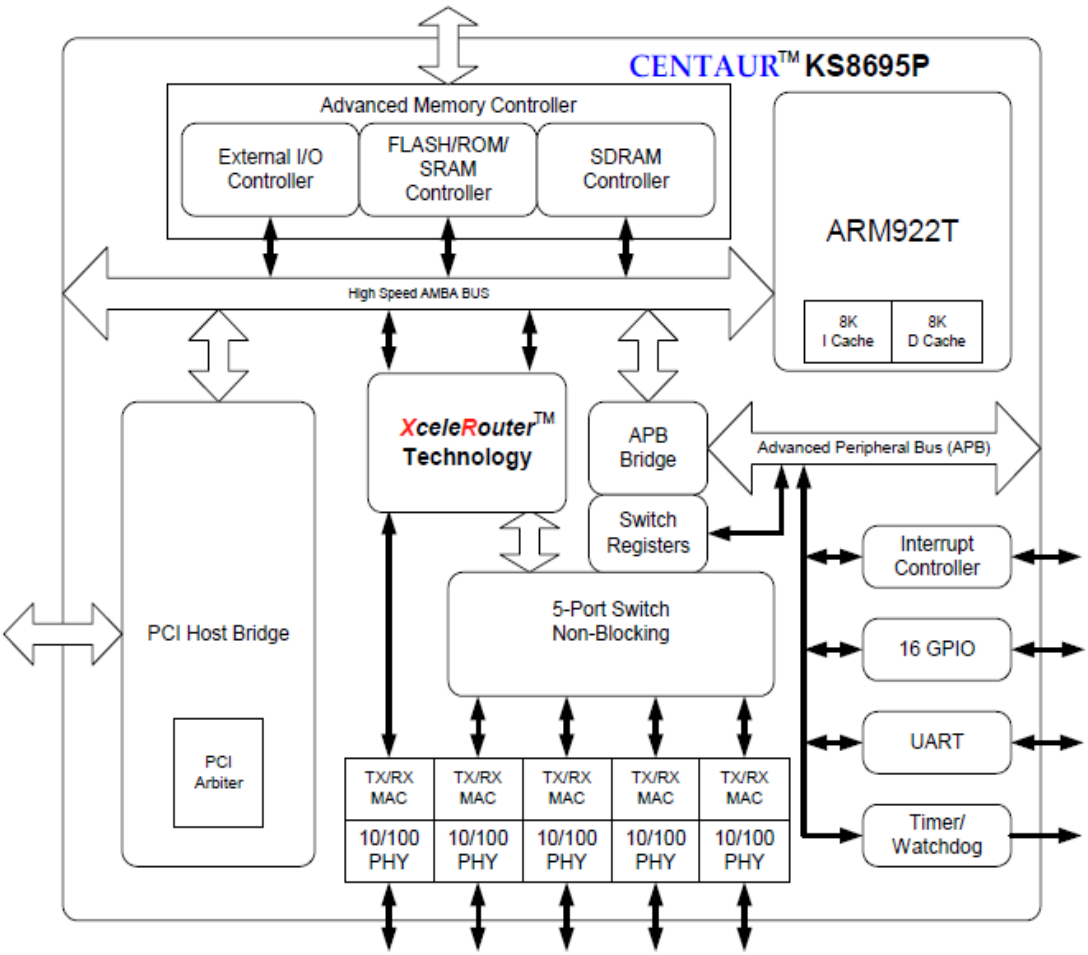

Figura 14 - Arquitetura da plataforma de desenvolvimento [MIC05].

O switch Ethernet da plataforma possui cinco portas, uma das portas esta conectada a CPU e as outras são portas Ethernet. Existe ainda uma quinta porta Ethernet independente do switch que foi utilizada para carregar os executáveis para o Linux embarcado utilizando transferência de arquivos de um servidor Hypertext Transfer Protocol (HTTP) em uma estação de trabalho que irá compilar o código fonte para a arquitetura Advanced RISC Machine (ARM) utilizando o *toolchain* fornecido no site do fabricante. Uma foto ilustrativa da plataforma de desenvolvimento aparece na Figura 15.

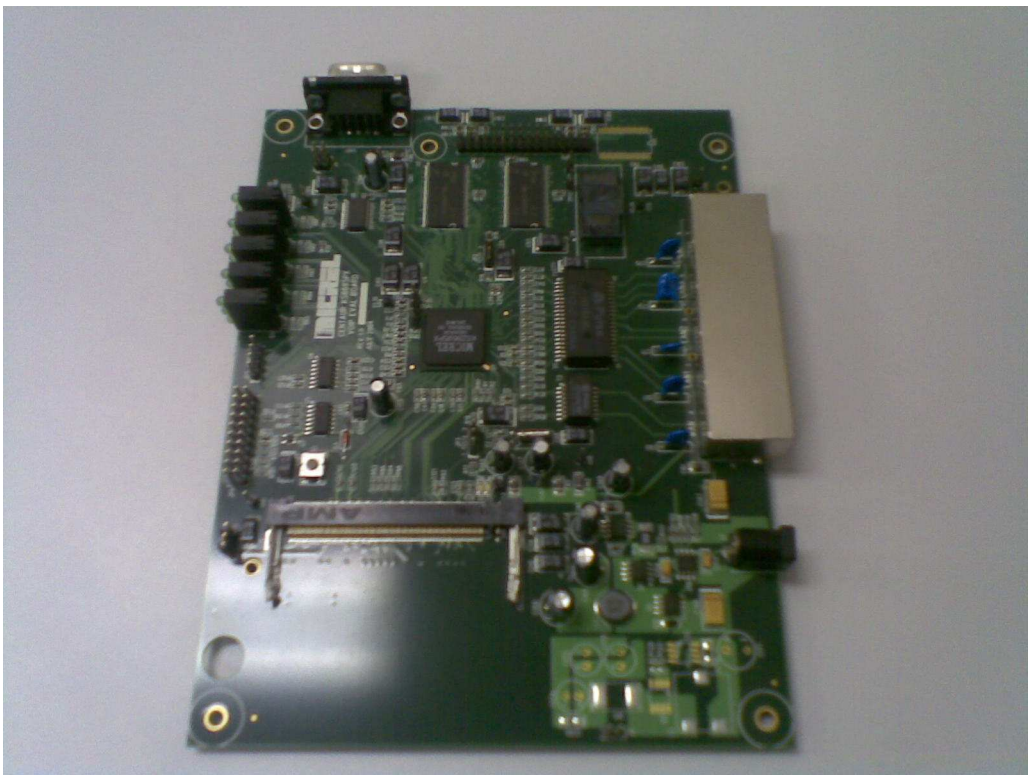

Figura 15 - Foto ilustrativa da plataforma de desenvolvimento.

Primeiramente foi necessário o estudo do *driver* que faz a comunicação com os periféricos da plataforma, em especial o *switch Ethernet*, as funções de configuração e manipulação que utilizam este driver foram separadas em uma HAL, tornando o algoritmo principal independente do tipo do *switch*, sendo necessária somente a substituição da HAL para utilizar o protocolo desenvolvido em outro *switch Ethernet*. Dentre as funções de configuração está a utilização de *tags* para indicar ao switch em que porta o pacote deve ser enviado e para saber de que porta do switch o pacote foi recebido, pois o Linux embarcado trata todas as portas Ethernet do switch como apenas uma interface local *eth1*. Tendo em vista esta utilização de *tags* o envio e recebimento de pacotes também serão dependentes do hardware. Diferentes switches podem ter diferentes tipos de *tags*. O envio de pacotes foi realizado utilizando *sockets*. Para o recebimento de pacotes foi utilizada a biblioteca de captura de pacotes LIBPCAP. Uma vez que o Linux obtido no site do fabricante não possui esta biblioteca instalada, foi necessário portá-la para o ambiente embarcado. Outra configuração necessária foi a ativação do suporte ao protocolo IGMP no *switch*, que faz com que este reconheça pacotes IGMP analisando se o próximo protocolo na camada IP é o IGMP. Caso seja um pacote IGMP, o *switch* repassa os pacotes para a CPU. Estas configurações são realizadas no início da execução do protocolo e não sofrem mais alterações, o que é constante na comunicação com o *switch* é a manipulação da tabela de grupos que este possui em hardware. Na medida em que o algoritmo principal atualiza suas informações, estas serão gravadas na tabela do *switch*. O *switch* utiliza o endereço Ethernet Multicast desta tabela para verificar em que portas ele deve enviar o tráfego *multicast* de determinado grupo.

Para o desenvolvimento na plataforma de hardware foi utilizado um computador como servidor de arquivos, para compilar e editar o código fonte, além de monitorar e controlar o sistema em execução na plataforma. O servidor de arquivos serve para carregar a imagem do Linux na plataforma através do protocolo Trivial File Transfer Protocol (TFTP), utilizando o U-boot que já está instalado de fábrica na

memória flash da plataforma. O U-boot se encarrega de inicializar o Linux [YAG03] a partir desta imagem que possui o *Kernel* e o sistema de arquivos. Este é o *bootloader* mais usado em sistemas embarcados Linux. O controle do Linux e do U-boot é feito através da porta serial utilizando o software MINICOM, configurado com o protocolo serial utilizado pela plataforma. Com este, emula-se um terminal Linux em uma estação de trabalho, o que facilita o desenvolvimento na plataforma de hardware.

3.1 Desenvolvimento da HAL

Após a criação do ambiente e o domínio da geração e manipulação do Linux embarcado, o desenvolvimento da camada de abstração de hardware para isolar as funções dependentes de hardware, tornando o trabalho mais facilmente portável para diferentes plataformas de hardware, juntamente com o estudo do datasheet da plataforma. Nele foram focadas as funções para a execução do protocolo IGMP e para o encaminhamento dos pacotes para os grupos multicast. Uma das dificuldades foi encontrar a maneira na qual os registradores de configuração do switch eram gravados na plataforma, pois no datasheet eram apresentados somente os endereços de memória mapeados para os registradores de configuração do switch.

3.2 Configuração do switch Ethernet

As funções de manipulação criadas fazem a inserção da tag nos pacotes que entram e saem do switch ethernet, pois este possui quatro portas e o Linux trata todas estas portas como somente uma interface física ethernet *eth1*. Uma das funções ativa o suporte do switch ao IGMP, com este suporte o switch identifica os pacotes IGMP e os repassa para o Linux para serem tratados pelo protocolo desenvolvido. A Tabela 1 mostra as funções criadas para a manipulação do switch com os seus protótipos e suas descrições.

Tabela 1- Funções de manipulação do switch

Protótipo	Descrição
void switch_init(char *interface)	Inicializa as configurações do switch para inserção de tag e captura de pacotes IGMP.
void switch_exit(void)	Libera os recursos ao terminar a execução do programa.
void add_group_table(int posicao, int porta, u_int8_t* grupo)	Adiciona um grupo a tabela de MACs estática do switch na posição indicada com as portas que pertencem ao grupo.
void remove_group_table(int posicao)	Remove um grupo da tabela do switch, desativando o bit de validade do índice indicado.

3.3 Acesso direto à memória

O acesso direto à memória no Linux não era um tema dominado pelo grupo de pesquisa que realizou este trabalho, pois se trata de um aspecto tecnológico muito específico que não é estudado na graduação, somente são estudados os conceitos e não tecnologias específicas, com isto diversas tentativas de acesso direto a memória foram realizadas. Uma destas tentativas foi através do dispositivo de memória virtual do Linux dev/mem, que não obteve sucesso, pois não acessa endereços físicos de memória somente endereços virtuais. Neste ponto sabíamos que deveríamos desenvolver um driver para ter acesso ao kernel do Linux e com isto aos endereços físicos da memória, porém a maneira como isto é feito em um driver também não foi facilmente encontrada. Depois de diversas tentativas resolvemos encontrar um driver parecido na implementação do Linux embarcado oferecido pela Micrel, lá encontramos o driver que faz toda a configuração dos diversos dispositivos de hardware da plataforma além do switch Ethernet. Analisando o código deste driver descobrimos a maneira na qual é feito o acesso direto à memória, que é através de ponteiros para o endereço físico em nível de execução do kernel do Linux.

A Figura 16 mostra o código em linguagem C no qual são criadas duas macros de funções de leitura e escrita em um endereço de memória. São definidos o endereço base dos registradores da plataforma e os deslocamentos para os registradores específicos do switch Ethernet e dos outros componentes da plataforma de acordo com o datasheet. Para fazer acesso a um registrador o endereço base é somado ao *offset* do registrador alvo, este endereço é usado em um ponteiro que pode ser lido e escrito estando no nível de execução do kernel como um endereço físico acessando o registrador mapeado em memória.

```
/*SWITCH registers offset difinitions*/
#define REG_SWITCH_CTRL0        0xE800
#define REG_SWITCH_CTRL1        0xE804
#define REG_SWITCH_PORT1        0xE808
#define REG_SWITCH_PORT2        0xE80C
#define REG_SWITCH_PORT3        0xE810
#define REG_SWITCH_PORT4        0xE814
#define REG_SWITCH_PORT5        0xE818
#define REG_SWITCH_AUTO0        0xE81C
#define REG_SWITCH_AUTO1        0xE820
#define REG_SWITCH_LUE_CTRL     0xE824
#define REG_SWITCH_LUE_HIGH     0xE828
#define REG_SWITCH_LUE_LOW      0xE82C
#define REG_SWITCH_ADVANCED     0xE830

/*host communication registers difinitions*/
#define REG_DSCP_HIGH           0xE834
#define REG_DSCP_LOW            0xE838
#define REG_SWITCH_MAC_HIGH     0xE83C
#define REG_SWITCH_MAC_LOW      0xE840

/*miscellaneours registers difinitions*/
#define REG_MANAGE_COUNTER      0xE844
#define REG_MANAGE_DATA         0xE848
#define REG_DEVICE_ID           0xEA00
#define REG_REVISION_ID         0xEA04

#define REG_HPNA_CONTROL        0xEA08
#define REG_WAN_CONTROL         0xEA0C
/*newly added registers*/
#define REG_WAN_POWERMAGR       0xEA10
#define REG_WAN_PHY_CONTROL     0xEA14
#define REG_WAN_PHY_STATUS      0xEA18


#define REG_IO_BASE             0x03FF0000
#define IO_READ(p)          ((*(volatile unsigned int *)(REG_IO_BASE + p)))
#define IO_WRITE(p, c)      (*(unsigned int *)(REG_IO_BASE + (unsigned int)p) = (c))
```

Figura 16 - Código para acesso direto à memória em nível de execução do kernel do Linux

3.4 Adaptando o driver

Com o driver encontrado criamos um programa que se comunica com ele através do *ioctl* do Linux, que são chamadas de funções que fazem a comunicação entre o nível de usuário e o do kernel. Esta comunicação foi adaptada usando como base um programa que faz a depuração dos registradores da plataforma, este programa vem em conjunto com o Linux embarcado oferecido no site da Micrel.

Para modificar as configurações do switch desenvolvemos uma função que grava em um registrador o valor passado usando o endereço de memória constante no datasheet. Com esta função mapeamos os registradores necessários para desenvolvimento do protocolo para criarmos as funções de manipulação do switch.

3.5 Tabela de grupos em Hardware

A função mais importante de configuração do switch foi a adição e remoção de um grupo multicast, para isto é usada uma tabela de MACs estática que o switch possui. Nesta tabela configuramos o MAC do grupo multicast e as portas que fazem parte deste grupo, além de informar se esta entrada da tabela é válida ou não. Uma restrição importante é o limite de apenas oito entradas da tabela, o que restringiu as possibilidades de testes com um número de grupos maior que oito. A Tabela 2 mostra o formato do registrador para inserção de MACs na tabela estática interna do switch.

Tabela 2 - Registrador para inserção dos grupos na tabela do switch [MIC05]

Format of static MAC table for writes (8 entries)

Bit	Name	R/W	Description	Default
59-56	FID	W	Filter VLAN ID, representing one of the 16 active VLANs.	0000
55	Use FID	W	=1, use (FID+MAC) to look up in static table =0, use MAC only to look up in static table	0
54	override	W	=1, override spanning tree “transmit enable=0” or “receive enable=0” setting. This bit is used for spanning tree implementation =0, no override	0
53	valid	W	=1, this entry is valid, the look up result will be used =0, this entry is not valid	0
52-48	Forwarding ports	W	The 5 bits control the forward ports, ex 00001, forward to port 1 00010, forward to port 2 ..... 10000, forward to port 5 00110, forward to port 2 and port 3 11111, broadcasting (excluding the ingress port)	00000
47-0	MAC address	W	48 bit mac address	0x0

3.6 Testes em Hardware

Depois de criadas as funções elas foram testadas para verificar se as configurações estavam corretas. A inserção da tag foi testada enviando pacotes Ethernet com a tag para determinadas portas da plataforma, conectadas em um computador analisando o tráfego com o Wireshark. Nesta análise foi verificado se o pacote era recebido na porta correta.

Para verificar o suporte ao IGMP foram gerados pacotes IGMP para o switch da plataforma. No switch foram capturados estes pacotes mostrando que este estava repassando os pacotes IGMP para o Linux, pois o comportamento de um switch comum sem suporte ao IGMP seria repassar estes pacotes em todas as suas portas sem repassá-los a CPU de seu sistema, tratando o tráfego como broadcast.

O teste mais importante foi o da tabela de grupos em hardware que foi realizado inserindo o grupo 224.100.10.10 no switch manualmente usando as funções desenvolvidas, somente uma das portas foi adicionada a esse grupo. Depois, foi gerado tráfego para o grupo cadastrado com o VLC em modo servidor multicast. Em um switch comum sem suporte ao IGMP os três clientes receberiam o vídeo do servidor. Foi verificado se somente a porta cadastrada no grupo recebia o seu tráfego capturando os pacotes no Wireshark em cada cliente usando um filtro para pacotes UDP, que são os pacotes usados pelo servidor para distribuir o vídeo. A Figura 17 mostra a arquitetura do teste com o grupo configurado no servidor.

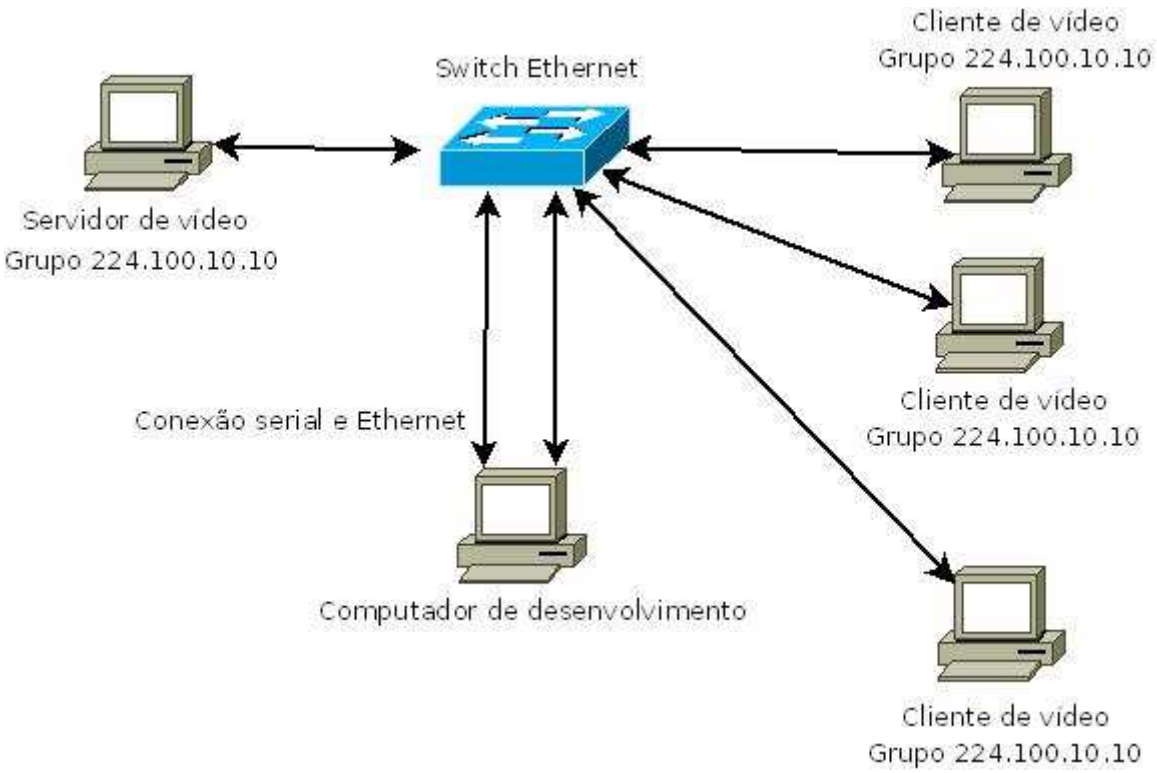

Figura 17 - Arquitetura para o teste da tabela de grupos do switch

Como pode ser visto na Figura 18, somente a porta cadastrada manualmente no switch no grupo 224.100.10.10 recebeu o stream de vídeo. Nesta figura são mostradas três janelas de captura do Wireshark, cada janela é uma sessão do VNCVIEWER em um dos clientes. Na janela do cliente que está conectado a porta cadastrada no grupo, podemos ver os pacotes UDP com destino ao grupo multicast sendo capturados. Este teste mostrou que a HAL estava funcionando corretamente, agora o próximo passo era implementar o

IGMP Snooping para fazer o gerenciamento dos grupos multicast.

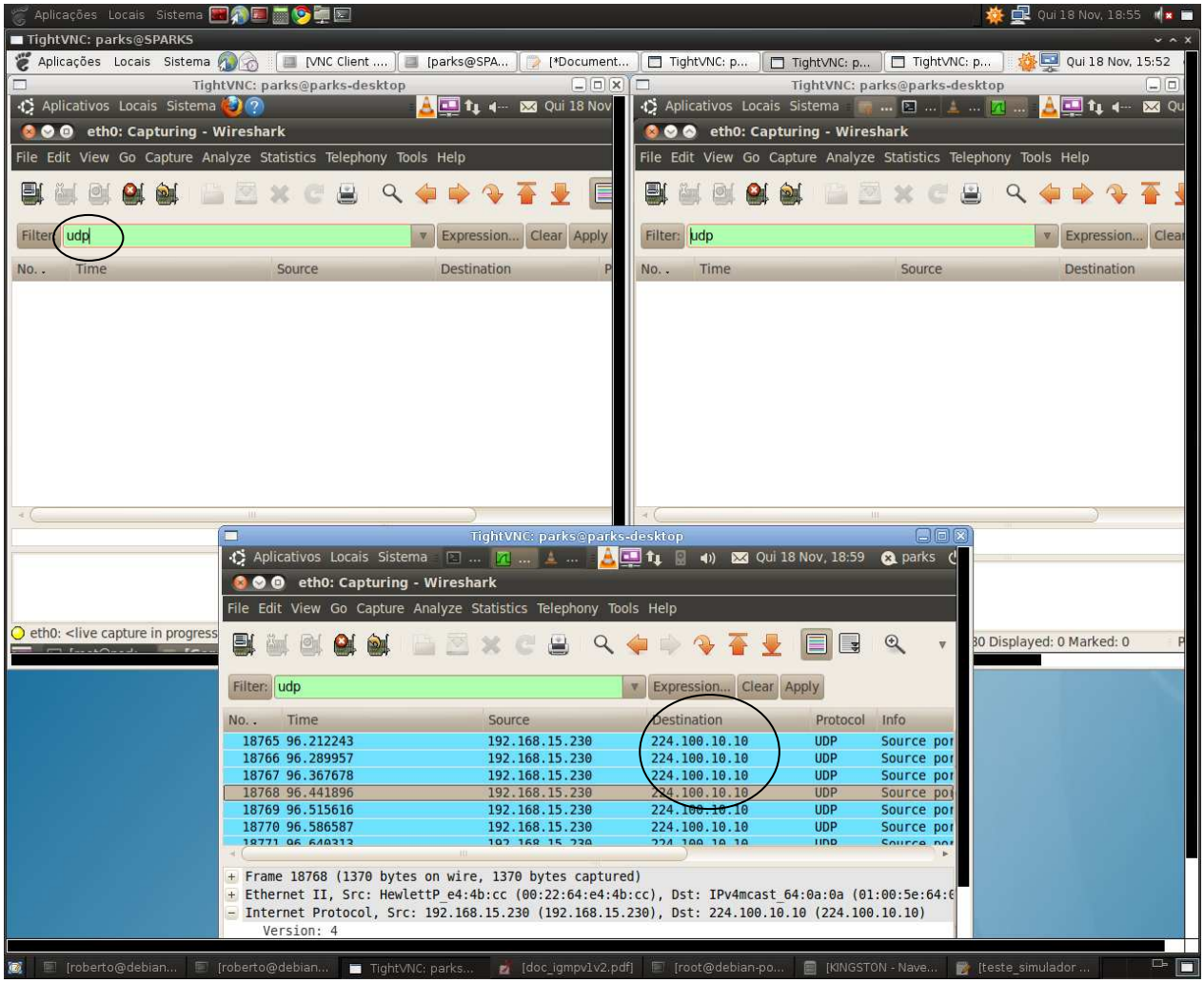

Figura 18 - Resultado do teste da tabela de grupos do switch

4. IMPLEMENTAÇÃO DO PROTOCOLO IGMPV1

Este Capítulo descreve a implementação do algoritmo IGMPV1 Snooping no switch Ethernet embarcado da plataforma de desenvolvimento. Primeiramente, é apresentado o referencial teórico estudado nas Seções 4.1 a 4.4, incluindo: as adaptações que devem ser realizadas no sistema operacional para suportar multicast (Seções 4.1 a 4.3) e em seguida o protocolo IGMP nas Seções 4.4 e 4.5. São descritos os protocolos IGMPV1 (Seção 4.4), IGMP Snooping e IGMP Proxy (Seção 4.5 e 4.6) que servem como base para a implementação do trabalho. O restante do Capítulo (Seções 4.7 a 4.13) apresenta o desenvolvimento do algoritmo, a estrutura de dados utilizada, a captura e o envio de pacotes e a implementação do algoritmo em si.

Existem três níveis de adaptação de um hospedeiro para dar suporte a IP multicasting [DEE89]:

- **Nível 0: sem suporte para IP Multicast** - Não existe a necessidade que todas as implementações IP suportem *multicast*. Hospedeiros neste nível, em geral, não serão afetados pela atividade *multicast*. A única exceção ocorre em alguns tipos de rede local, onde a presença de hospedeiros nível 1 ou 2 pode causar erros de entrega de datagramas IP Multicast para hospedeiros do nível 0. Estes datagramas, no entanto, podem ser facilmente identificados pela presença de um endereço classe D o campo de endereço destino, e devem ser descartados por hospedeiros que não dêem suporte a IP Multicast.
- **Nível 1: suporte para o envio mas não recebimento de datagramas IP Multicast** - Permite a um hospedeiro participar de alguns serviços baseados em *multicast*, como locação de recursos, mas não o permite participar de nenhum grupo de hospedeiros.
- **Nível 2: suporte completo a IP Multicast** - Permite a um hospedeiro participar ou abandonar grupos, além de permitir o envio de datagramas a estes grupos. Requer, no entanto, a implementação de um protocolo de gerência de grupos (IGMP - Internet Group Management Protocol) [DEE89], bem como a extensão do IP e das interfaces de serviço de rede local acopladas aos hospedeiros.

A implementação destes níveis baseia-se na necessidade de adaptação de diversos módulos, conforme pode ser visto na Figura 19 [DEE89].

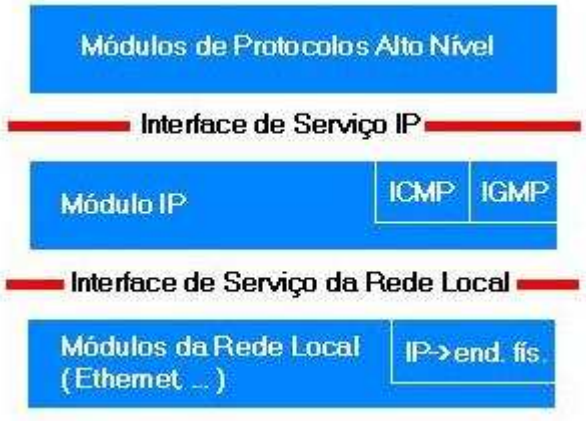

Figura 19 - Níveis estruturais passíveis de modificação no suporte a multicast do protocolo IGMP [DEE89].

4.1 Endereços de Grupo

Grupos de hospedeiros são identificados pelos endereços IP classe D (possuem "1110" nos quatro bits de mais alta ordem), conforme ilustra a Figura 20.

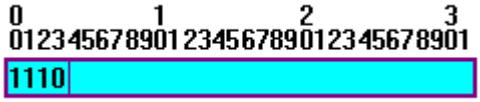

Figura 20 - Formato de endereços de grupo [GAS96].

Na notação padrão da Internet, endereços de grupos variam de 224.0.0.0 a 239.255.255.255. O endereço 224.0.0.0 não pode ser atribuído a nenhum grupo, e 224.0.0.1 á atribuído ao grupo permanente de todos os hospedeiros IP (incluindo gateways). É usado para endereçar todos os hospedeiros *multicast* na rede diretamente conectada. Não há endereço *multicast* para todos os hospedeiros de toda a Internet. Os endereços de grupos permanentes são publicados pela entidade Internet Assigned Numbers Authority (IANA), conforme consta em [IAN10].

4.2 Envio de Datagramas IP Multicast

4.2.1 Extensões da Interface de Serviço IP

Datagramas IP Multicast são enviados usando a mesma operação de envio de datagramas IP Unicast. Um módulo de protocolo de nível superior especifica unicamente um endereço de grupo como destino, ao invés de um endereço IP individual. Contudo, é necessário gerar um conjunto de extensões para a funcionalidade desta interface [DEE89]:

- A interface de serviço deve prover uma maneira dos protocolos de níveis superiores especificarem o parâmetro *Time-to-Live* TTL de um datagrama *multicast* que esteja sendo enviado. Se o nível superior não especifica este parâmetro, é utilizado o valor default 1 para todos os datagramas IP Multicast, de

modo que uma escolha explícita é necessária para propagar ao mensagem *multicast* além da rede corrente.

- Para hospedeiros que possam estar associados a mais de uma rede, o serviço de interface deve prover uma maneira para que os níveis superiores identifiquem a interface de rede a utilizar para realizar transmissões *multicast*. Apenas uma interface é utilizada para a transmissão inicial. Roteadores *multicast* são responsáveis pelo repasse destes datagramas para qualquer outra rede, se necessário. Se nenhuma interface é identificada pelo nível superior, deve-se eleger uma interface padrão.
- Se o hospedeiro que está enviando um datagrama é membro do grupo destino na interface de saída, uma cópia do datagrama deve ser fornecida a este hospedeiro em uma entrega local, a menos que tenha sido inibida em nível superior.

4.2.2 Extensões do Módulo IP

Para dar suporte ao envio de datagramas IP Multicast, o módulo IP deve ser estendido para reconhecer endereços de grupo quando rotear datagramas de saída. Grande parte das implementações incluem a seguinte lógica [DEE89]:

- Se IP destino está na mesma rede local, envia-se datagrama localmente para o IP destino.
- Senão envia-se datagrama localmente para o *Gateway* (IP destino).
- Para dar suporte a transmissões *multicast*, a lógica de roteamento deve ser mudada para:
- Se IP destino está na mesma rede local ou IP destino é um grupo de hospedeiros, envia-se datagrama localmente para IP destino.
- Senão, envia datagrama localmente para o *Gateway* (IP destino).

O endereço IP fonte do datagrama que está saindo deve ser um dos endereços individuais correspondentes à interface de saída. Um endereço *multicast* nunca deve ser colocado no campo de endereço fonte de um datagrama IP que esteja sendo enviado.

4.2.3 Extensões à Interface de Serviço da Rede Local

Nenhuma mudança na interface de serviço da rede local é necessária para enviar datagramas IP Multicast. O módulo IP apenas especifica um endereço destino de grupo, e não um endereço individual, quando requisita a operação existente "*Send Local*".

4.2.4 Extensões a um Módulo de Rede Local Ethernet

O padrão Ethernet dá suporte direto ao envio de pacotes *multicast* locais, permitindo endereços *multicast* no campo destino dos pacotes Ethernet. Tudo o que é necessário para dar suporte ao envio de datagramas IP Multicast é um procedimento para mapear endereços de grupo para endereços Ethernet Multicast. Um endereço IP de grupo é mapeado para um endereço Ethernet Multicast atribuindo os 23 bits de ordem mais baixa do endereço IP aos 23 bits de mais baixa ordem do endereço Ethernet Multicast 01-00-5E-00-00-00. Pelo fato de haver 28 bits significativos em um endereço IP Multicast, mais de um endereço de

grupo pode ser mapeado para um mesmo endereço Ethernet Multicast.

4.3 Recebimento de Datagramas IP Multicast

4.3.1 Extensões da Interface de Serviço IP

Datagramas IP Multicast que chegam são recebidos por protocolos de nível superior usando a mesma operação "*Receive IP*", como se fossem datagramas *unicast*. A seleção de um protocolo de nível superior destino é baseada no campo protocolo no cabeçalho do IP, independente do endereço IP destino. Contudo, antes que quaisquer datagramas destinados a um grupo particular sejam recebidos, um protocolo de nível superior deve pedir ao módulo IP para participar do grupo em questão [DEE89]. Deste modo, a interface de serviço IP deve ser estendida para oferecer duas novas operações:

- JoinHostGroup (endereço-grupo, interface)
- LeaveHostGroup(endereço-grupo, interface)

A operação JoinHostGroup faz uma requisição solicitando que o hospedeiro torne-se membro do grupo de hospedeiros identificado pelo "endereço-grupo" em uma determinada interface de rede. O argumento interface pode ser omitido em hospedeiros que dêem suporte a apenas uma interface. Para hospedeiros associados a mais de uma rede, o protocolo de nível superior pode não especificar a interface. Neste caso, a requisição será atendida através de uma interface padrão para enviar datagramas *multicast*.

É possível participar do mesmo grupo em mais de uma interface, sendo que neste caso datagramas *multicast* duplicados podem ser recebidos. Também é possível que mais de um protocolo de nível superior faça a requisição para participar de um mesmo grupo.

Ambas as operações devem retornar imediatamente indicando sucesso ou falha. Elas são portanto não-bloqueantes. Uma destas operações pode falhar devido a endereços de grupo ou interfaces inválidas. JoinHostGroup pode falhar ainda pela falta de recursos locais. LeaveHostGroup pode falhar se o hospedeiro não pertencer ao grupo definido. Esta operação pode retornar com sucesso, mas a associação pode continuar existindo se mais de um protocolo superior tiver requisitado participação no mesmo grupo.

4.3.2 Extensões do Módulo IP

Para dar suporte à recepção de datagramas IP Multicast, o módulo IP deve ser estendido para manter uma lista de participantes de grupos de hospedeiros associada a cada interface de rede. Quando um datagrama destinado a um destes grupos é recebido, opera-se exatamente da mesma maneira que com datagramas destinados a um endereço de hospedeiro individual.

Pacotes chegando destinados a grupos aos quais o hospedeiro não pertença são descartados sem gerar nenhuma informação de erro ou mensagem de diagnóstico. Em hospedeiros com mais de uma interface de rede, se um datagrama chega via uma dada interface destinado a um grupo para o qual o hospedeiro pertence em outra interface, o datagrama também é descartado.

Um datagrama não é rejeitado por ter um Time-to-Live de 1 (isto é, o TTL não é automaticamente decrementado em datagramas que chegam e que não sejam reenviados). Um datagrama que chegue com um endereço de grupo em seu campo endereço fonte é descartado. Uma mensagem de erro ICMP (destino inalcançável, tempo excedido ou problema com parâmetros) nunca é gerada em resposta a um datagrama destinado a um grupo de hospedeiros.

A lista de participantes de grupos é atualizada sempre que as primitivas JoinHostGroup e LeaveHostGroup são requisitadas por níveis superiores. Cada associação deve ter um contador de referência ou mecanismo similar para manipular várias requisições de *join* e *leave* para o mesmo grupo. Na primeira requisição de *join* e na última requisição de *leave* em uma determinada interface, o módulo de rede local daquela interface é notificado, de modo que ele possa atualizar seus filtros de recepção *multicast*.

4.3.3 Extensões à Interface de Serviço de Rede Local

Pacotes *multicast* que chegam à rede local são entregues ao módulo IP utilizando a mesma operação "*Receive Local*" que pacotes *unicast* da rede local. Para permitir ao módulo IP informar ao módulo de rede local que pacotes *multicast* aceitar, estende-se a interface de serviço de rede local com duas novas operações:

- JoinLocalGroup (endereço-grupo)
- LeaveLocalGroup (endereço-grupo)

Nestas operações endereço-grupo é o endereço de grupo de hospedeiros. A operação JoinLocalGroup requisita ao módulo de rede local que aceite e entregue os pacotes subsequentes destinados a um dado endereço de grupo. A operação LeaveLocalGroup requisita que o módulo de rede local pare de entregar pacotes destinados a um dado endereço de grupo de hospedeiros. Exige-se que o módulo de rede local mapeie endereços IP Multicast para endereços de rede local, para atualizar seus filtros de recepção *multicast*. Caso seja incapaz de filtrar pacotes, qualquer módulo de rede local pode ignorar livremente requisições LeaveLocalGroup, e pode entregar pacotes destinados a mais endereços que apenas aqueles especificados nas requisições JoinLocalGroup.

O módulo de rede local não deve entregar a ele mesmo pacotes *multicast* que tenham sido transmitidos por ele próprio. O *loopback* de *multicast* é manipulado no nível IP ou superior.

4.3.4 Extensões do Módulo de Rede Local Ethernet

Para dar suporte à recepção de datagramas IP Multicast, um módulo Ethernet deve ser capaz de receber pacotes destinados a endereços Ethernet Multicast que correspondam a endereços de grupo. É altamente desejável aproveitar as capacidades de filtragem, de modo que o hospedeiro receba apenas pacotes destinados a ele. Infelizmente, muitas interfaces Ethernet possuem um limite pequeno de número de endereços que o hardware é capaz de reconhecer. Deste modo, uma implementação deve ser capaz de "escutar” um número arbitrário de endereços *multicast*, que pode significar "abertura" dos filtros de endereço para aceitar todos os pacotes *multicast* durante períodos em que o número de endereços excede o limite do filtro.

4.4 IGMP Versão 1

O Protocolo Internet de Gerenciamento de Grupos (em inglês, Internet Group Management Protocol ou IGMP) é usado por hospedeiros para vincular participantes de um grupo de hospedeiros a roteadores *multicast* vizinhos [DEE89]. É um protocolo assimétrico e é especificado do ponto de vista de um hospedeiro, e não do ponto de vista de um roteador *multicast*.

Como o ICMP [POS81], IGMP é uma parte integral do IP. É um requisito básico de implementações a todos os hospedeiros que desejem enviar e receber pacotes *multicast*. As mensagens IGMP são encapsuladas em datagramas IP, com um número de protocolo IP igual a 2. Todas as mensagens importantes do ponto de vista do hospedeiro possuem o formato descrito na Figura 21 e detalhados a seguir.

Figura 21 - Formato das mensagens de protocolo IGMPV1 [RNP98].

- Version: Versão 1 do IGMP. A versão 0 é especificada na RFC-988 [DEE86] e está obsoleta.
- Type: há dois tipos de mensagens:
  - Host Membership Query
  - Host MemberShip Report
- Unused: campo não utilizado, zerado quando enviado e ignorado quando recebido.
- Checksum: usado para verificar a integridade do pacote, o cálculo é explicado em [BRA88].
- Group Address: em uma mensagem Host Membership Query, o campo de endereço de grupo é zerado quando enviado e ignorado quando recebido. Por outro lado, em uma mensagem Host Membership Report, este campo contém o endereço de grupo do grupo sendo relatado.

Roteadores *multicast* enviam mensagens Host Membership Query para descobrir que grupos de hospedeiros têm membros nas suas redes locais associadas. As requisições são endereçadas a todos os grupos de hospedeiros (endereço 224.0.0.1) e carregam um T*ime-to-Live* igual a 1.

Os hospedeiros, por sua vez, respondem à consulta gerando Host Membership Reports, relatando cada grupo de hospedeiro ao qual ele pertence na interface de rede a partir da qual a consulta foi recebida. Com o objetivo de evitar uma explosão de relatos concorrentes e reduzir o número de relatos transmitidos, duas técnicas são usadas:

- Quando um hospedeiro recebe uma consulta, ao invés de enviar os relatos imediatamente, ele inicia um temporizador de atraso para cada um dos participantes dos grupos na interface de rede que recebeu a consulta. A cada temporizador é atribuído um tempo aleatoriamente escolhido entre 0 e um valor D. Quando o temporizador expira, um relato é gerado para o membro do grupo de hospedeiro correspondente. Assim, relatos são espalhados em um intervalo de D segundos e não ocorrerão todos ao mesmo tempo. O valor padrão de D é 10 segundos [DEE89].

- Um relato é enviado com um endereço de IP destino igual ao endereço de grupo de hospedeiros sendo relatado, e com o *Time-to-Live* igual a 1. Se um hospedeiro escuta um relato para um grupo ao qual pertence, ele pode parar seu temporizador para aquele grupo e não gerar relato para ele. Assim, no caso normal, apenas um relato será gerado para cada grupo presente na rede, pelo membro do grupo cujo atraso seja o menor. Conclui-se daí que roteadores *multicast* recebem todos os datagramas IP *multicast* e, portanto não precisam ser endereçados explicitamente. Além disto, note-se que os roteadores não precisam saber quais hospedeiros pertencem a um grupo, apenas necessitam saber que pelo menos um hospedeiro pertence a um grupo em uma rede particular.

Roteadores *multicast* fazem consultas periódicas para atualizar suas tabelas de grupos presentes em uma rede particular. Se nenhum relato é recebido de um grupo particular após certo número de consultas, os roteadores assumem que aquele grupo particular não possui mais membros e eles não precisa mais redirecionar pacotes *multicast* originados remotamente para aquele grupo, naquela rede específica. As consultas são feitas com um intervalo razoável (em torno de 1 minuto) a fim de não sobrecarregar a rede.

Quando um hospedeiro passa a participar de um grupo, ele deve transmitir imediatamente um relato para aquele grupo, ao invés de esperar pela consulta feita pelos roteadores, no caso de ele ser o primeiro membro daquele grupo na rede. Para cobrir a possibilidade do relato inicial ser perdido ou danificado, é recomendável que este seja repetido uma ou duas vezes após intervalos curtos.

O comportamento IGMP é mais formalmente especificado pelo diagrama de transição de estados descrito na Figura 22 [DEE89]. Um hospedeiro pode estar em um dos três estados possíveis, para cada grupo de hospedeiro IP em qualquer interface de rede. Descreve-se a seguir os estados em detalhe.

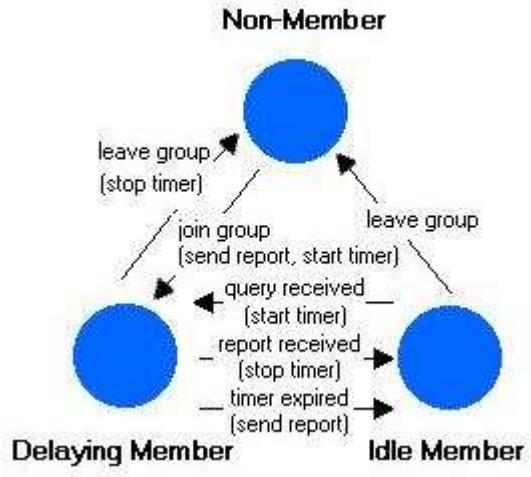

Figura 22 - Diagrama de transição de estados do protocolo IGMPv1 [DEE89].

- **Estado Non-Member** - quando o hospedeiro não pertence ao grupo na interface. Este é o estado inicial para todos os grupos em todas as interfaces de rede; não requer armazenamento no hospedeiro.
- **Estado Delaying Member** - quando o hospedeiro pertence ao grupo na interface e tem um temporizador de relatos executando para aquela associação.
- **Estado Idle Member** - Quando um hospedeiro pertence ao grupo na interface e não possui um temporizador executando para aquela associação.

Há cinco eventos significativos que podem causar transições de estado:

- "**join group**" ocorre quando o hospedeiro decide participar do grupo na interface. Pode ocorrer apenas no estado Non-Member.
- "**leave group**" ocorre quando um hospedeiro decide abandonar um grupo na interface. Pode acontecer apenas nos estados Delaying Member e Idle Member.
- "**query received**" ocorre quando um hospedeiro recebe uma mensagem Host Membership Query. Para ser válido, este deve de ter pelo menos 8 octetos, ter um checksum correto e um endereço IP destino de 224.0.0.1. Se aplica uma consulta simples a todos os grupos na interface que recebe a solicitação. Este evento é ignorado para grupos nos estados Non-Member ou Delaying Member.
- "**report receive**" ocorre quando o hospedeiro recebe uma mensagem Host Membership Report. Para ser válido, este deve ter pelo menos 8 octetos, ter um *checksum* correto, e conter o mesmo endereço IP de grupo no seu campo de destino e no campo de endereço de grupo IGMP. Um relato se aplica apenas para participantes no grupo identificado pelo relato, na interface onde este é recebido. É ignorado para grupos no estado Non-member ou Idle Member.
- "**timer expired**" ocorre quando o temporizador de atraso para o grupo na interface expira. Só pode ocorrer no estado Delaying Member.

Há três ações possíveis que podem ser dadas em resposta aos eventos:

- "**send report**" para o grupo na interface.
- "**start timer**" para o grupo na interface, usando uma atraso aleatório entre 0 e D segundos.
- "**stop timer**" para o grupo na interface.

4.5 IGMP Snooping

O protocolo IGMP assume que a rede é formada por switches passivos que tratam o tráfego *multicast* como se fosse broadcast [SCH08]. Com isto, se perde muito dos benefícios do multicast, pois os hospedeiros receberão todo o tráfego multicast destinado a rede, como o hospedeiro deve examinar cada pacote para verificar se este é multicast e passar para camada IP para verificar se o hospedeiro faz ou não parte do grupo, com um número grande de *Streams* multimídia o hospedeiro vai sobrecarregar seu processamento somente examinando pacotes. Devido a esta necessidade, recentemente fabricantes de switches introduziram no mercado uma nova funcionalidade chamada IGMP Snooping [CHR06]. Estes dispositivos não seguem mais o modelo OSI, pois utilizam informação de uma camada superior para fazer encaminhamento dos pacotes.

Um switch com IGMP Snooping provê o benefício de conservar a banda dos hospedeiros que não desejam tráfego *multicast*, além de não repassar tráfego de grupos dos quais o hospedeiro não faz parte. Não existe uma norma para o desenvolvimento do protocolo, este é proprietário e depende de cada fabricante. Em [CHR06] existem fundamentos para o desenvolvimento do IGMP Snooping. O que se deve fazer é estudar o comportamento do roteador e do hospedeiro quanto ao IGMP e realizar a implementação.

O switch deve ter suporte em hardware para a implementação do IGMP Snooping. Ele deve reconhecer datagramas IGMP e repassá-los à CPU. Com estas informações, deve-se atualizar a tabela de grupos *multicas*t, fazendo com que o tráfego seja direcionado corretamente em suas portas. O switch deve manter

uma lista dos roteadores *multicast* com as suas respectivas portas, esta lista pode ser construída de uma das seguintes formas:

- Usando o protocolo Multicast Router Discovery (MRD) [HAB05];
- Verificando em que portas chegaram General Queries;
- Configurando administrativamente as portas.

4.6 IGMP Proxy

O IGMP Proxy amplia as funcionalidades do IGMP Snooping, minimizando o tráfego IGMP e a carga de trabalho do roteador. Para isto, replica comportamentos tanto do hospedeiro como do roteador [SCH08] e gera uma base de dados com a união das informações de grupos *multicast* de todas as suas portas. Em [FEN06], existem recomendações para o seu desenvolvimento.

Uma diferença na aplicação dos dois protocolos encontra-se em [WEI08], onde switches de menor porte utilizam IGMP Snooping e switches de maior porte utilizam o IGMP Proxy, pois este demanda mais processamento por ter de replicar comportamentos de hospedeiros e roteadores gerando datagramas, enquanto que no IGMP Snooping os datagramas são somente verificados e repassados.

4.7 Escolha da estrutura de dados

Tendo estudado as características do protocolo IGMP, tanto do ponto de vista do hospedeiro quanto do roteador para a implementação na plataforma de desenvolvimento do protocolo IGMP Snooping, foi constatado que os componentes mais importantes seriam uma estrutura de dados para guardar os grupos multicast e seus membros ativos e o algoritmo pra sua manipulação. Os grupos são representados pelo seu IP Multicast convertido para um valor inteiro, para facilitar sua manipulação. Os hospedeiros membros de algum grupo são representados pelo número da porta no qual estão conectados.

Foram cogitadas diferentes estruturas de dados e algoritmos para a implementação do sistema, mas a que se destacou por ter um melhor desempenho, O(log n), para a pesquisa foi a árvore balanceada AVL (Adelson Velsky e Landis). Ela foi escolhida pois a operação mais realizada no algoritmo é a pesquisa de grupos e portas toda vez que se recebe um report de um hospedeiro. Deve-se pesquisar se o grupo já está incluído na base de dados dos membros multicast. Caso já esteja incluído deve-se pesquisar se a porta já está incluída no grupo. Estes testes são constantemente realizados no algoritmo toda vez que é incluído um novo grupo ou porta em um grupo existente. Também são realizados para verificar se o grupo e a porta já existem, pois o hospedeiro responde constantemente a general queries do roteador para se manter membro dos grupos multicast de seu interesse.

A implementação de AVL que foi utilizada pode ser encontrada em [ROC09], esta estrutura foi testada para verificar se ela operava corretamente. Depois de constatada a sua correta operação ela foi adaptada para incluir as variáveis necessárias para o controle dos grupos. Este mesmo modelo também foi utilizado para o

controle das portas. Com isto a base de dados ficou com uma árvore de grupos no qual cada nodo possui uma árvore de portas. A estrutura é alocada dinamicamente e é organizada utilizando o endereço IP dos grupos e o número das portas. A Figura 23 mostra esta estrutura.

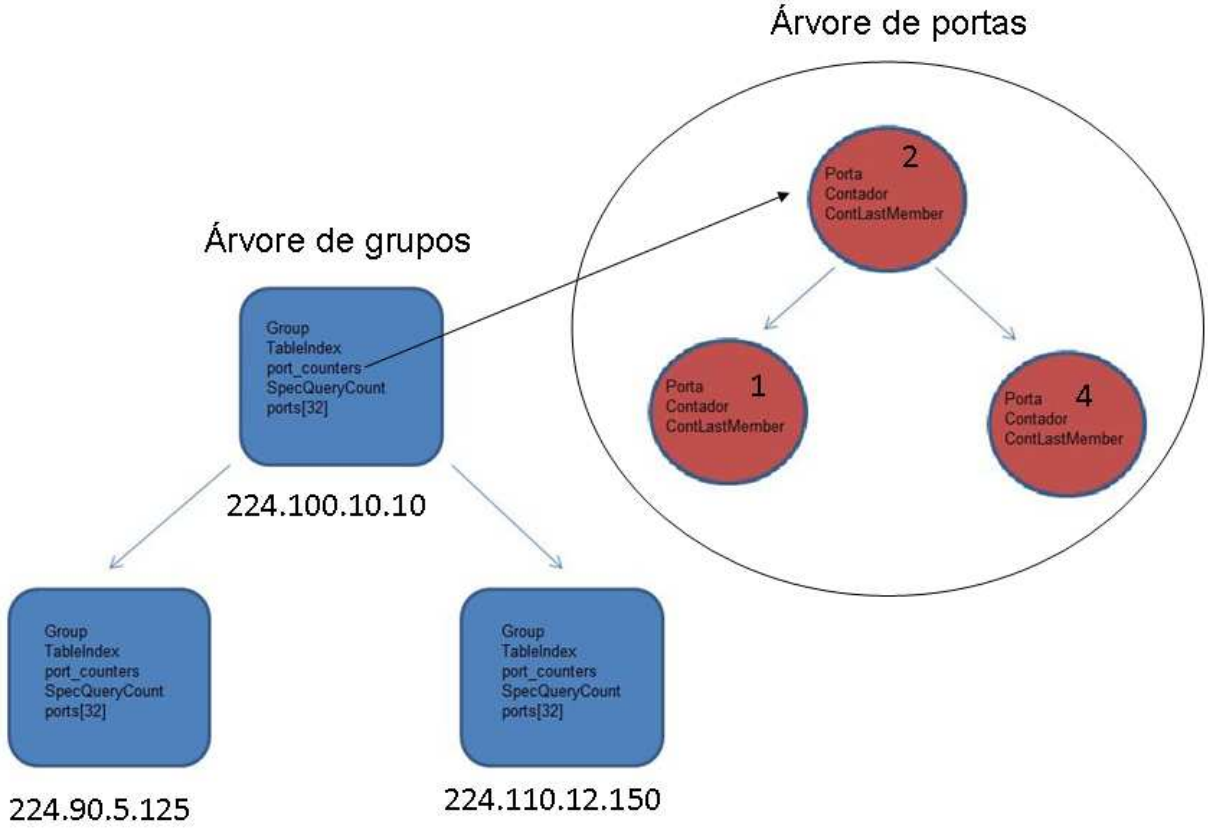

Figura 23 - Estrutura de dados utilizada para manutenção dos grupos multicast

A Figura 24 detalha a estrutura de dados mostrando o código em linguagem C das estruturas da árvore de grupo e da árvore de portas. Estas variáveis são usadas para a implementação do algoritmo, elas são detalhadas nas próximas sessões e no Capítulo 5.

```
/* Definição dos ponteiros para as estruturas */
typedef struct noavl_group *PtAVL;
typedef struct noavl_port *PtAVLPort;

struct noavl_group        /* definição de um nó da árvore de grupos */
{
    int Group;          /* variável que representa o grupo multicast armazenado na AVL */
    int TableIndex;     /* representa o índice deste grupo na tabela de hardware */
    PtAVLPort port_counters; /* ponteiro para a árvore de portas */
    int SpecQueryCount; /* contador de queries específicas */
    unsigned int ports[32]; /* vetor auxiliar para facilitar a pesquisa de portas ativas */

    /* variáveis de controle da árvore existentes na implementação original*/
    PtAVL PtEsq;        /* ponteiro para o nó esquerdo */
    PtAVL PtDir;        /* ponteiro para o nó direito */
    int Altura;         /* altura do nó */
    };


struct noavl_port       /* definição de um nó da árvore de portas */
{
    int Porta;      /* variável que representa a porta em que um cliente está conectado */
    int Contador;   /* Conta o número de general queries que o cliente respondeu */
    int ContLastMember; /* Conta o número de queries específicas que o cliente respondeu */

    /* variáveis de controle da árvore existentes na implementação original*/
    PtAVLPort PtEsq;    /* ponteiro para o nó esquerdo */
    PtAVLPort PtDir;    /* ponteiro para o nó direito */
    int Altura;     /* altura do nó */
};
```

Figura 24 - Código das estrutras das árvores de grupos e portas

4.8 Tag para diferenciar portas

Como foi visto no capítulo 3, a plataforma possui um switch que é configurado para utilizar tags para identificar a porta em que deve ser encaminhado o pacote ou a porta em que o pacote chegou, pois o Linux trata as quatro portas do switch como somente uma interface Ethernet física *eth1*. Com isto, o pacote capturado não terá uma estrutura padrão, pois é inserida informação no cabeçalho Ethernet entre os campos MAC source e o Type/Lenght.

Para tornar esta característica de captura facilmente configurável, com intuito de facilitar seu uso em outros switches e em configurações de sistema operacional que tratem o pacote diferentemente da plataforma de desenvolvimento utilizada, foi criada uma variável de deslocamento, que é o número de bytes da tag especial do switch que devem ser deslocados no pacote para obter o cabeçalho IGMP. Com a estrutura padrão Ethernet/IP/IGMP o cabeçalho IGMP se obtém a partir do primeiro byte após a camada IP do pacote. Com a inserção da tag de 4 bytes basta somar esta variável de deslocamento com o valor padrão para obtermos o cabeçalho IGMP. Além disso, a tag deve ser decodificada para identificar a porta em que o pacote foi capturado, usando a descrição da Special Tag em [MIC05]. A tag possui 4 bytes com o valor x“8100” que é somado ao *nibble* que representa as portas, este nibble é decodificado segundo a Tabela 3.

Tabela 3 - Decodificação do nibble de identificação das portas

Valor do nibble	Descrição
“0001”	Envia pacote somente pela porta 1 ou pacote recebido da porta 1
“0010”	Envia pacote somente pela porta 2 ou pacote recebido da porta 2

“0100”	Envia pacote somente pela porta 3 ou pacote recebido da porta 3
“1000”	Envia pacote somente pela porta 4 ou pacote recebido da porta 4
“0011”	Envia pacote para as portas 1 e 2
“1111”	Envio broadcast, envia pacote para as portas 1, 2, 3, e 4
“0000”	Pacote normal é tratado usando a tabela de MACs usual do switch

Para separar esta funcionalidade dependente de hardware do algoritmo principal foram utilizadas macros nas funções de captura e envio de pacotes, com isto somente é necessária a mudança nesta parte do algoritmo caso seja utilizada outra plataforma que utiliza tags diferentes ou que não possui a diferenciação de portas por tags.

Para o envio de pacotes também é adicionada a tag para indicar em que porta o switch deve enviar o pacote, os procedimentos de captura e envio de pacotes serão descritos a seguir.

4.9 Captura de pacotes com a Libpcap

A biblioteca de captura de pacotes Libpcap, foi utilizada devido à facilidade que ela oferece ao programador para realizar captura de pacotes com diversas opções de configurações, além de possuir uma excelente documentação. Em [TCP10] existem diversos tutorias e manuais que foram estudados para realizar a implementação, além das atualizações e código fonte da biblioteca.

As funções implementadas para realizar a captura de pacotes formam uma biblioteca que é usada no algoritmo principal para capturar os pacotes na interface do Linux que corresponde as portas do switch na plataforma de hardware. Na Tabela 4 são mostradas as funções da biblioteca Libpcap utilizadas, suas finalidades e características.

Tabela 4 - Funções utilizadas da bibiloteca Libpcap

Protótipo da função	Descrição
pcap_lookupnet(dev, &net, &mask, errbuf)	Obtém informações da interface “dev”, sua máscara e rede
descriptor = pcap_open_live(dev, BUFSIZ, 0, 1000, errbuf)	Cria uma sessão de captura na interface “dev”, criando um descritor para manipulá-la
pcap_setdirection(descriptor,PCAP_D_IN)	Seta a direção em que os pacotes devem ser capturados, se somente pacotes enviados, somente pacotes recebidos ou os dois fluxos
pcap_compile(descriptor, &fp, filter_exp, 0, net)	Compila um filtro, que é uma expressão lógica para capturar somente tráfego multicast
pcap_setfilter(descriptor, &fp) == -1)	Aplica o filtro à sessão criada
packet = pcap_next(descriptor, &hdr)	Captura o próximo pacote que combinar com o filtro

A função “pcap_open_live” possui alguns argumentos que definem o modo da captura, o “0” usado na implementação significa que a interface não estará em modo promíscuo, que é um modo no qual se captura pacotes para qualquer destino não somente para o IP da interface que está fazendo a captura. Como o tráfego a ser capturado é multicast o modo promíscuo não precisa ser configurado, pois os tráfegos multicast e broadcast são capturados no modo padrão.

4.10 Envio de pacotes com sockets

Sockets são uma forma de IPC Inter Process Communication, que fornece comunicação entre processos residentes em sistema único ou processos residentes em sistemas remotos. Sockets criados por diferentes programas são referenciados através de nomes, esses nomes devem ser traduzidos em endereços, espaço no qual o endereço é especificado é chamado de domínio [FER10].

Domínios básicos:

- INTERNET (AF_INET) - os endereços consistem do endereço de rede da máquina e da identificação do número da porta, o que permite a comunicação entre processos de sistemas diferentes.
- Unix (AF_UNIX) - os processos se comunicam referenciando um *pathname*, dentro do espaço de nomes do sistema de arquivos.

Tipos de sockets:

- STREAM SOCKET - provê seqüenciamento e fluxo bidirecional. Transmite dados sobre uma base confiável no domínio UNIX, trabalha igual a um *pipe*. No domínio INTERNET é implementado sobre TCP/IP.
- DATAGRAM SOCKET - suporta fluxo de dados bidirecional, não oferece um serviço confiável. Mensagens duplicadas, perdidas, e em ordem diferente (não seqüenciadas), são problemas que podem aparecer.
- RAW SOCKET - permite o acesso a interface de protocolos de rede. Disponível para usuários avançados e que possuam autoridade de usuário *root*. Permite acesso direto a protocolos de comunicação de baixo nível. Permite a construção de novos protocolos sobre os protocolos de baixo nível já existentes.

O envio de pacotes no algoritmo foi feito através de RAW SOCKET e foi desenvolvida uma biblioteca para facilitar o envio de pacotes, sendo necessário somente chamar a função de inicialização do socket no início da execução do algoritmo. Depois só é necessário chamar a função de envio de pacotes quando for conveniente.

Na Tabela 5 são mostradas as funções desenvolvidas, suas finalidades e características.

Tabela 5 - Funções de manipulação de sockets desenvolvidas

Protótipo da função	Descrição
int CreateRawSocket(int protocol_to_sniff)	Cria o RAW SOCKET para o envio de pacotes

int BindRawSocketToInterface(char *device, int rawsock, int protocol)	Atribui a interface de envio no socket
int Set_Raw_Socket_To_Send(char *device, int rawsock, int protocol,struct sockaddr_ll *PacketInfo, int AddressSize)	Prepara as estruturas no socket para poder enviar os pacotes
int sendPacket(int port, const u_char *packet,int size)	Envia o pacote já com a tag correspondente a porta

Foi escolhida esta forma para o envio de pacotes devido à familiaridade adquirida durante a graduação do autor com as disciplinas de redes de computadores e programação que abordavam o tema e principalmente durante os estágios que foram realizados pelo autor na área de telecomunicações, onde o conhecimento de implementação de sockets em Linux foi aprofundado. Outra razão é que o pacote que deve ser gerado não é um pacote padrão e está se implementando um protocolo, portanto precisa-se de uma manipulação mais direta de seu conteúdo que o RAW SOCKET do Linux oferece.

4.11 Implementação das estruturas de dados

O algoritmo desenvolvido utiliza diversas estruturas de dados, com as implementações base de AVL, fila e pilha, adaptações foram realizadas para incluir as informações necessárias ao algoritmo, como IP multicast do grupo e número da porta do membro. Para facilitar e encapsular a manipulação da árvore, todas as funções que a modificam foram incluídas na própria árvore, tratando a AVL como se fosse uma classe de programação orientada a objetos. As principais funções implementadas são mostradas na Tabela 6.

Tabela 6 - Funções de manipulação da estrutura de dados principal

Protótipo da função	Descrição
void AVLQuery(PtAVL *avl, PtQueue fq)	Verifica quais clientes estão ativos em todos os grupos
void AVLInsert (PtAVL *avl, int elemento, int index)	Insere um nodo na árvore com informações do grupo onde: “elemento” é o grupo multicast “index” é o índice a ser ocupado pelo grupo na tabela de grupos do switch
int AVLSearchAndSet(PtAVL* tree_root_p,int ipGroup,int porta,int* index,int *ports,int *join)	Usado caso receba um report. Implementa parte do algoritmo de Join, procura o grupo se este existir seta as estruturas, senão informa que é um Join (report de um hospedeiro que não está no grupo)

void AVLInsertAndAdd(PtAVL* tree_root_p,int ipGroup ,int index,int porta)	Usado caso receba um report. Implementa parte do algoritmo de Join. Insere um novo grupo na árvore de grupos e insere a porta correspondente setando as estruturas de controle
void AVLQueryC (PtAVLPort avl)	Verifica quais clientes estão ativos na árvore de portas de um grupo se o contador não foi incrementado por um report. Então este nodo deve ir para uma fila para depois ser excluído
void AVLInsertC (PtAVLPort *avl, int elemento)	Insere uma porta (indicada por elemento) na árvore de portas, o contador é inicializado em 1 para não retirar acidentalmente uma porta que recentemente fez um Join

Além da árvore outra estrutura de controle foi implementada. Uma fila com os índices livres na tabela de grupos do switch, que é inicializada com todos os índices e a cada inserção de grupo um índice é retirado da fila e é atribuído ao grupo. Assim ao atualizar um grupo gravamos diretamente no índice indicado por ela na tabela de grupos do switch, facilitando a sua atualização, pois não precisamos reescrever toda a tabela do switch toda vez que algum grupo for atualizado e no caso de deleção de um grupo o índice é apenas desativado marcando o bit correspondente indicado em [MIC05] como foi mostrado no capítulo 3.

Esta estrutura foi utilizada levando em conta que existiriam no máximo 256 portas, pois um número maior de portas acabaria usando muita memória se existissem muitos grupos. No pior caso cada árvore de grupo pode ter uma árvore de portas com o número máximo de portas, para tal caso seria necessário um switch de grande porte com bastantes recursos de processamento e memória. No trabalho foi possível testar com um número máximo de 8 grupos e 4 clientes devido a limitação do hardware usado, com o simulador foi possível testar com um número máximo de 16 clientes e 10 grupos. Este simulador será visto em detalhes no Capítulo 6.

4.12 Implementação do Join

Com o conhecimento do protocolo adquirido ao estudar diversos RFCs, programamos o Join de hospedeiros. Como estudado no protocolo o Join é quando um host deseja fazer parte de um grupo multicast, para isso ele envia uma requisição ao entrar no grupo ou espera que o roteador faça uma general query e responde com o grupo de interesse. Ele também deve continuar respondendo as general queries subseqüentes do roteador (o que é referido como report) para se manter no grupo caso o hospedeiro ainda queira fazer parte do grupo. O papel de IGMP Snooping no algoritmo desenvolvido é capturar esta comunicação entre o roteador e o hospedeiro e atualizar sua tabela de encaminhamento multicast.

Para realizar o Join de hosts várias funções foram adicionadas a árvore de grupos e a árvore de portas como foi visto anteriormente. A principal, que é a função que implementa o Join, verifica se existe o grupo, caso não exista adiciona o grupo e a porta e incrementa o contador da porta. Caso exista o grupo verifica se a porta já está incluída através de um vetor que possui as portas ativas, com isto somente um cálculo é

necessário para saber se a porta já está incluída no grupo. Se a porta não existir adiciona a porta e incrementa o contador, senão apenas incrementa o contador da porta. Além disso, cada grupo possui uma variável que indica qual a sua posição na tabela de grupos do switch ethernet, com isto se torna mais simples a sua manutenção, não sendo necessário reescrever toda a tabela do switch quando mudar a base de dados dos grupos no algoritmo principal. Esta fila é inicializada com todas as posições livres da tabela de grupos do switch ethernet, a cada grupo que é incluído um índice é retirado desta fila, ao eliminar o grupo o índice é adiciona a fila.

A Figura 25 mostra o fluxo da implementação do Join realizada.

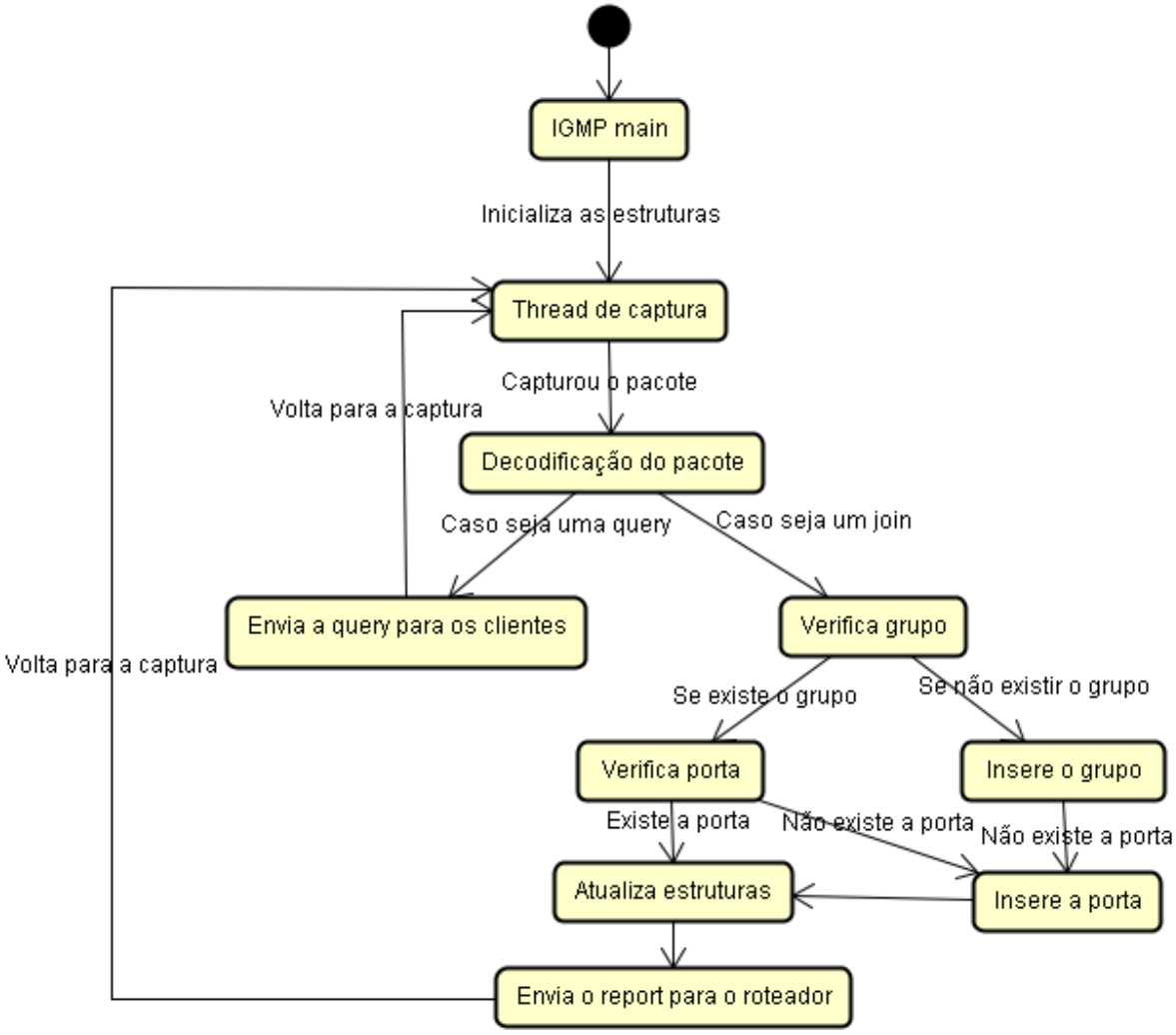

Figura 25 - Fluxo do Join do IGMPV1

4.13 Timeout de hospedeiros

No protocolo IGMP o roteador faz periodicamente aos hospedeiros general queries para verificar se existe algum membro em algum grupo e para eliminar hospedeiros que não responderem com um report no tempo máximo de 3 general queries mais o Max Response Time. Este tempo é padronizado no IGMPV1 em 10 segundos.

Ao invés de possuir um contador de timeout para cada grupo, como é descrito no protocolo que o

roteador deve possuir, o timeout de hospedeiros foi implementado contando-se o número de queries recebidas do roteador e contando-se o número de reports recebidos de cada porta em cada grupo multicast que esta porta está cadastrada. Ao chegar ao número limite de três queries recebidas do roteador volta-se a capturar os pacotes e uma thread é criada, esta thread aguarda o Max Response Time e depois faz uma varredura completa no banco de dados dos membros multicast eliminando as portas que não receberam nenhum report e eliminando os grupos que não possuírem mais portas. Para eliminar os grupos é necessário guardar os grupos que não possuírem mais portas em uma fila, pois enquanto está se fazendo recursivamente a varredura nas árvores se torna difícil manter as referências corretas e saber qual nodo já foi verificado devido às rotações que irão ocorrer na árvore para balanceá-la. Ao terminar a varredura os contadores são reinicializados.

A Figura 26 mostra o fluxo da implementação do timeout de hospedeiros realizada.

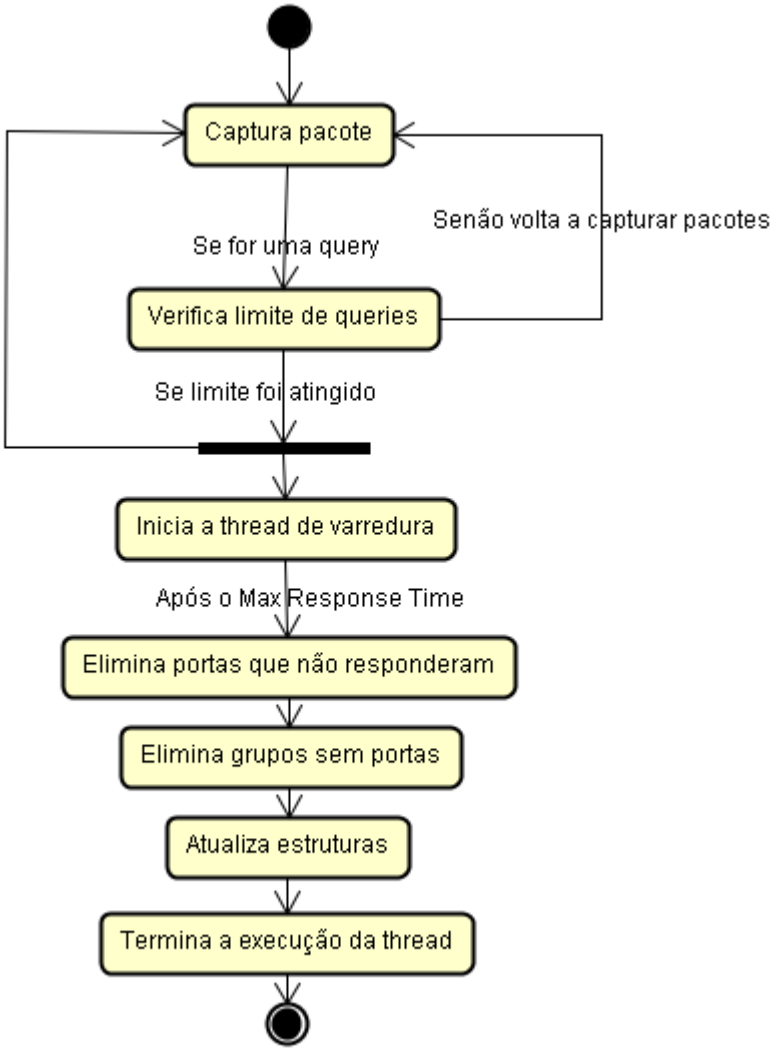

Figura 26 - Fluxo do timeout de hospedeiros

O acesso mutuamente exclusivo a árvore na sua varredura é necessário por existir múltiplos fluxos de execução, para isso foi usado um *mutex*. Ele faz parte da implementação de *pthreads* que foi utilizada também para criar as threads, esta biblioteca é a forma mais usual e prática para se criar programas com execuções paralelas no Linux e foi amplamente estudada na graduação do autor.

O fluxo completo do algoritmo do protocolo IGMPV1 desenvolvido é mostrado na Figura 27.

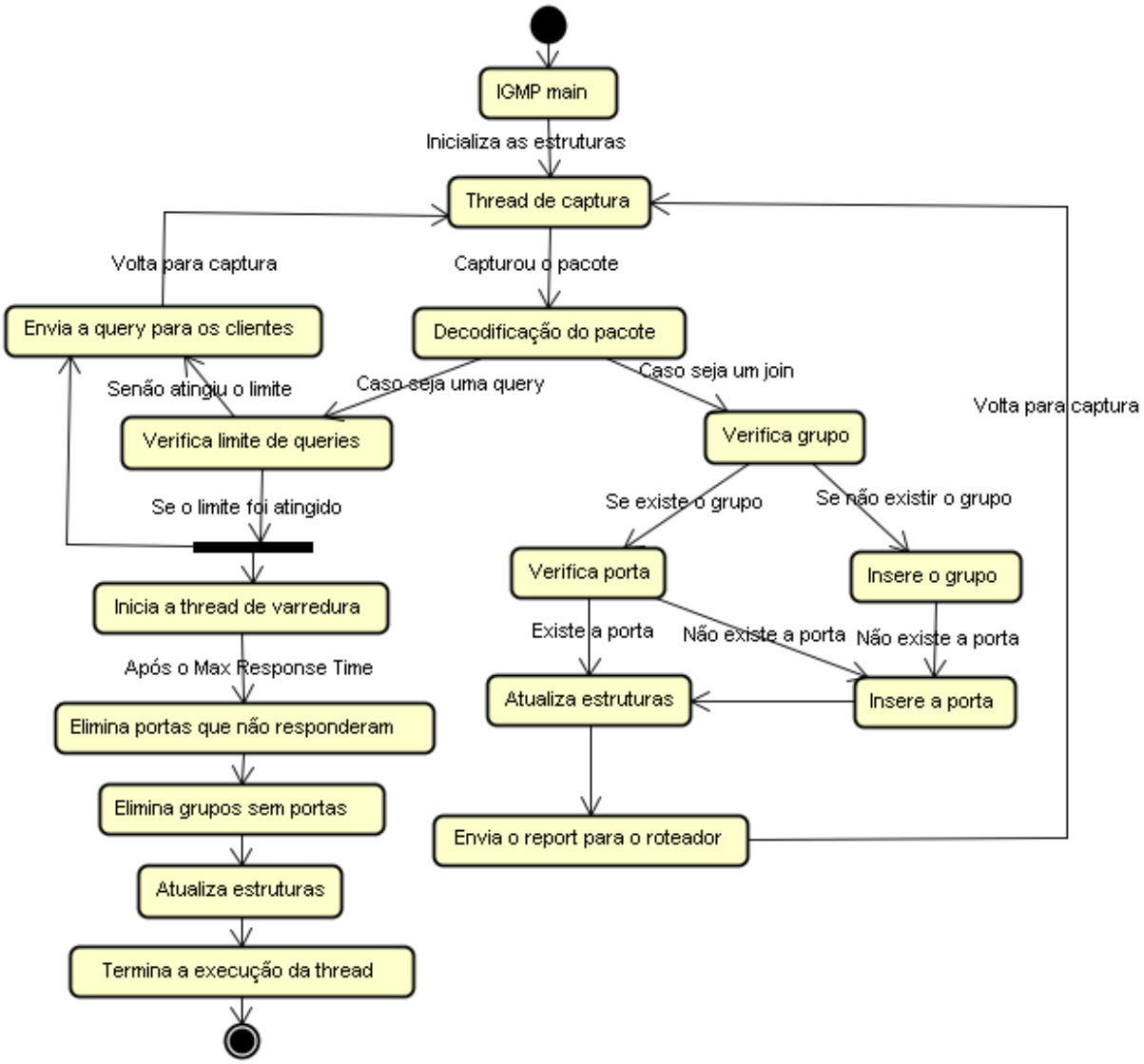

Figura 27 - Fluxo do algoritmo do IGMPV1 desenvolvido

5. IMPLEMENTAÇÃO DO PROTOCOLO IGMPV2

5.1 IGMP Versão 2

IGMPv2 permite a desvinculação de um grupo ser rapidamente comunicada aos roteadores, o que é importante para grupos que utilizam muita banda e sub-redes com grupos altamente voláteis [FEN97].

Roteadores que forem membros de grupos *multicast* devem agir como hospedeiros além de agir como roteadores, e podem responder a suas próprias inquirições (em inglês *queries*). Todas as mensagens IGMPv2 de interesse aos hospedeiros têm o formato definido na Figura 28 e detalhado a seguir.

Figura 28 - Formato das mensagens de protocolo IGMPv2 [RNP98].

- Type: Existem quatro tipos de mensagens para a interação entre roteadores e hospedeiros:
  - 0x11 - Membership Query - Existem dois subtipos de mensagens:
    - General Query, usada para descobrir quais grupos possuem membros em uma rede.
    - Group-Specific Query, usada para descobrir se um grupo particular ainda possui algum membro em uma rede.
  - 0x16 – Version 2 Membership Report
  - 0x17 – Leave Group
  - 0x12 – Version 1 Membership Report
- Max. Resp Time: O campo Max. Resp Time só possui significado nas mensagens Membership Query. Ele especifica o tempo máximo de resposta de um hospedeiro em unidades de décimo de segundo. Variando este parâmetro roteadores IGMPv2 podem ajustar a latência para retirar um grupo que não possui mais membros, além de poder modificar o tráfego IGMP na sub-rede.

Em relação a cada uma das suas redes associadas, um roteador *multicast* pode assumir uma das duas funções denominadas Inquiridor (Querier) ou Não-inquiridor (Non-Querier). Existe normalmente um único Inquiridor por rede física. Todos os roteadores *multicast* iniciam como Inquiridores em cada rede conectada. Se um roteador *multicast* recebe uma inquirição de um roteador com um endereço IP mais baixo, este torna-se um Não-inquiridor nesta rede. Se o roteador não receber uma inquirição do outro roteador em (Other Querier Present Interval), ele retoma o papel de Inquiridor. Roteadores periodicamente (Query Interval) enviam um General Query para cada rede conectada no qual é o Inquiridor.

Quando um hospedeiro deixa um grupo *multicast*, se ele foi o último a responder uma inquirição para o seu grupo, envia uma mensagem Leave Group para o grupo *multicast* all-routers (224.0.0.2). Implementações

antigas podem enviar a mensagem para o grupo ao qual está saindo ao invés do all-routers.

Quando um Inquiridor recebe uma mensagem Leave Group para um grupo que possui membros na interface recebida, este envia (Last member Query Count) Group-Specific Queries a cada (Last Member Query interval) para o grupo em questão [FEN97]. Se nenhum relato for recebido depois que a última inquirição expirar, o roteador assume que não há mais membros naquele grupo. Durante esta operação transições para Não-inquiridor são ignoradas.

5.1.1 Compatibilidade com roteadores IGMPv1

Hospedeiros IGMPv2 podem ser colocados em uma sub-rede no qual o Inquiridor possui apenas o IGMPv1. Para isso é necessário que o hospedeiro interprete o campo da Membership Query Max. Resp Time, que estará zerado, como se fosse 10 segundos, que é o valor padrão para implementações IGMPv1.

O roteador IGMPv1 só vai analisar Version 1 Membership Reports. Então, o hospedeiro deve manter uma variável indicando se um Inquiridor versão 1 está presente, para determinar que tipo de relato será usado. Um temporizador deve ser usado para voltar à versão 2 se o hospedeiro não receber mais General Queries Version 1 depois de um tempo denominado (Version 1 Router Present Timeout).

Caso exista um roteador IGMPv1 na mesma sub-rede de um IGMPv2, o roteador IGMPv2 deve ser administrativamente configurado para usar IGMPv1.

5.1.2 Compatibilidade com Hospedeiros IGMPv1

Um hospedeiro IGMPv2 pode ser colocado em uma sub-rede onde existem máquinas que ainda não foram atualizados para IGMPv2. Para isso, o hospedeiro deve permitir que seus Membership Reports sejam suprimidos tanto por Membership Reports Version 1 tanto como por Membership Reports Version 2.

Um roteador IGMPv2 pode ser colocado em uma sub-rede onde existem máquinas que ainda não foram atualizados para IGMPv2. Para isso, se um roteador receber um Membership Report Version 1, ele deve definir um temporizador para indicar que há hospedeiros versão 1 presentes que são membros do grupo para o qual a mensagem foi enviada. Este temporizador deve usar o mesmo tempo que o (Group Membership Interval).

Se houver um hospedeiro versão 1 presente em um grupo específico, o roteador deve ignorar qualquer mensagem Leave Group para este grupo.

5.1.3 Lista de Temporizadores e os Valores Padrão

Se configurações não-padrão forem utilizadas, elas devem ser consistentes entre todos os roteadores em um único link [FEN97]. Estas configurações são:

- **Robustness Variable**: Permite o ajuste da perda de pacotes IGMP dependendo da sub-rede. Se uma sub-rede tiver muitas perdas, a Robustness Variable pode ser aumentada. O IGMP é robusto a

(Robustness Variable -1) perdas de pacotes. A Robustness Variable não pode ser zero, e não deve ser 1. Valor Padrão: 2.

- **Query Interval**: É o intervalo entre General Queries enviadas pelo Inquiridor. Padrão: 125 segundos. Pela variação do [Query Interval], um administrador pode ajustar o número de mensagens IGMP na sub-rede. Com valores maiores General Queries serão enviadas com menos frequência.
- **Query Response Interval**: É inserido nas General Queries. Padrão: 100 (=10 segundos). Pela variação do (Query Response Interval), um administrador pode ajustar a quantidade de mensagens IGMP na sub-rede. Valores maiores tornam tráfego menos intenso, pois as respostas dos hospedeiros serão espalhadas por um intervalo maior. O número de segundos representado pelo (Query Response Interval) deve ser inferior a (Query Interval).
- **Group Membership Interval**: É a quantidade de tempo que deve passar antes de um roteador multicast decidir que não existem mais membros de um grupo em uma rede. Este valor deve ser:
  - (Robustness Variable) x (Query Interval) + (Query Response Interval).
- **Other Querier Present Interval**: É o tempo que deve passar antes de um roteador multicast decidir que não há mais outro roteador multicast que deve ser o Inquiridor. Este valor deve ser:
  - (Robustness Variable) x (Query Interval)) +1/2x (Query Response Interval).
- **Last Member Query Interval**: É o Max Resp Time inserido nas Group-Specific Queries enviadas em resposta a mensagens Leave Group. Padrão: 10 (=1 segundo). Este valor pode ser ajustado para modificar a velocidade de detecção de grupos sem membros na rede. Um valor pequeno resulta em um tempo reduzido para detectar a perda do último membro de um grupo.
- **Last Member Query Count**: É o número de Group-Specific Queries enviadas antes de o roteador assumir que não há membros no grupo. Padrão: Robustness Variable.
- **Version 1 Router Present Timeout**: É o tempo que um hospedeiro deve aguardar depois de receber uma General Query Version 1 antes de poder enviar qualquer mensagem IGMPv2. Valor: 400 segundos.

5.2 Adaptação das estruturas de dados

A implementação do protocolo IGMPV2 foi realizada adaptando-se o trabalho desenvolvido para o IGMPV1 com as diferentes funções adicionadas no novo protocolo. A grande mudança do protocolo IGMPV1 para o IGMPV2 foi a introdução do Leave, um sistema para agilizar a retirado de um membro de um grupo. Como visto no IGMPV1 só retiramos o membro do grupo se este não responder a 3 general queries do roteador mais o tempo do Max Response Time. No IGMPV2 mantemos este mecanismo para manter a compatibilidade com o IGMPV1, para o caso do Leave do hospedeiro IGMPV2 ser perdido ou não for gerado. O Max Response Time se tornou configurável não sendo mais fixo em 10 segundos. Com este valor configurável o administrador pode ajustar o número de pacotes que serão gerados tornando o tráfego menos intenso ao usar um valor maior ou mais intenso ao usar um número menor para o Max Response Time, ajustando as perdas e latências de acordo com a sua rede.

5.3 Implementação do Join

O Join do IGMPV2 segue o mesmo modelo do IGMPV1 desenvolvido, a diferença está na decodificação do pacote que agora inclui o novo Membership Report do IGMPV2.

5.4 Timeout de hospedeiros parametrizável

As queries IGMPV2 também devem ser decodificadas diferentemente para fazer o timeout dos hospedeiros, pois agora o Max Response Time é configurável e deve ser passado para a thread que faz a varredura da base de dados dos membros multicast. Isto faz com que esta thread espere o tempo usado na última general query gerada pelo roteador.

5.5 Implementação do Leave

Como foi visto no estudo do protocolo, o Leave é o novo mecanismo de retirada de membros de grupos multicast introduzido no IGMPV2. Ao sair de um grupo multicast o hospedeiro envia uma mensagem do tipo Leave para o roteador para indicar que ele não quer mais fazer parte de um grupo multicast. O roteador então gera queries especificas para aquele grupo para garantir que não existem mais hospedeiros com interesse de participar daquele grupo. Esta verificação do roteador se deve ao fato de o hospedeiro poder estar em uma conexão compartilhada com outros hospedeiros através de switches, hubs e outros equipamentos. Se nenhum hospedeiro responder a 3 queries específicas para o grupo mais tempo do Max Response Time, que na query especifica é padrão de 1 segundo podendo também ser alterado pelo administrador, o roteador elimina aquele grupo da sua tabela.

Com o comportamento do protocolo dominado ao colocar o switch com suporte ao IGMP entre o roteador e os hospedeiros, devemos capturar os Leaves dos hospedeiros e repassarmos para o roteador. Criamos um mecanismo semelhante ao feito para o timeout dos hospedeiros para o IGMPV1. Contamos as queries específicas para os grupos e como podem ser disparados diversos Leaves cada grupo possui seu contador de queries especificas. Quando o limite é atingido uma thread é gerada com o Max Response Time modificado de 1 segundo ou dependendo da configuração com outro valor configurado pelo administrador, este valor é retirado do pacote e repassado para a thread. Ao terminar o tempo a thread examina somente o grupo que gerou as queries específicas, portanto o grupo também é passado para a thread. Com isto podemos ter diferentes Leaves para diferentes grupos ao mesmo tempo gerando uma thread distinta para cada grupo, porém a árvore só é acessada por uma das threads do algoritmo por vez através de um mutex para garantir a exclusão mútua na base de dados dos grupos multicast.

A verificação para saber se o grupo deve ser retirado é feita com outro contador para os reports. Cada porta possui um destes contadores, ele só é incrementado após se receber um report depois de ter recebido uma query específica para aquele grupo. Assim não contamos os reports gerados em resposta a general queries, o que seria errado, pois um membro pode ter respondido uma general query e logo após ter saído do grupo e gerado um Leave. Portanto a thread para o Leave só analisa um dos grupos da base de dados e

verifica um contador de reports que só é incrementado após receber uma query específica, além disso, ao receber um Leave tanto o contador de queries específicas quando os contadores de reports das portas são reinicializados. A Figura 29 resume o algoritmo de Leave explicado nesta sessão através de um diagrama.

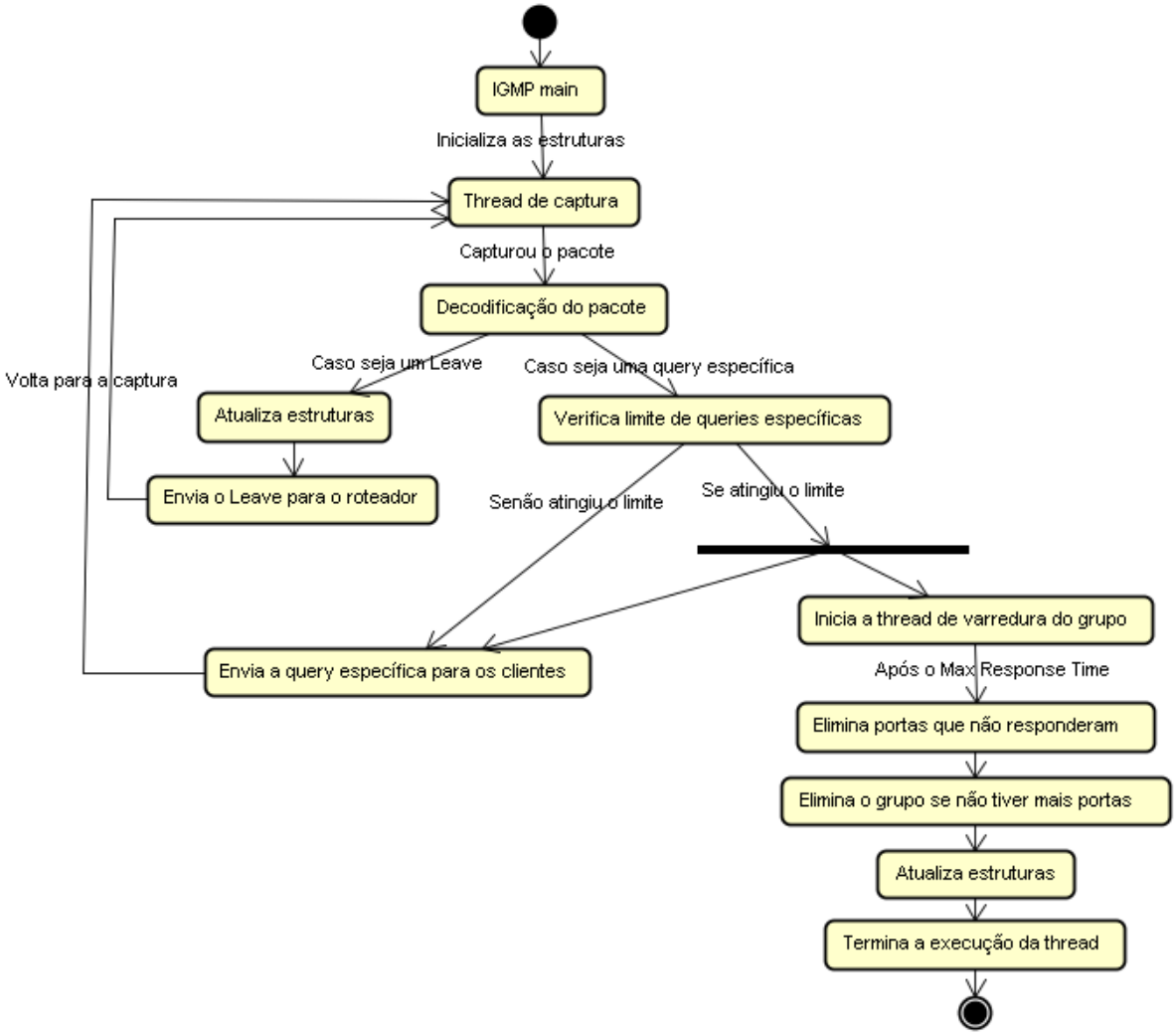

Figura 29 - Fluxo do Leave do IGMPV2

5.6 Compatibilidade com IGMPv1

Com a utilização de hospedeiros ou roteadores que só suportem o IGMPV1, os dispositivos que podem utilizar o IGMPV2 devem interoperar com os dispositivos IGMPV1. Como foi visto no estudo do RFC dependendo se é o roteador ou o hospedeiro IGMPV1 que é colocado em conjunto com outros equipamentos IGMPV2 diferentes ações devem ser realizadas para manter a compatibilidade.

Se a situação for um roteador IGMPV1 com hospedeiros IGMPV2, os hospedeiros devem gerar reports IGMPV1 e não devem gerar Leaves, pois o roteador os descartará. Se um host IGMPV1 é adicionado a uma rede com um roteador IGMPV2, ele não deve aceitar Leaves para os grupos em que o hospedeiro IGMPV1 estiver incluído, para isso o roteador mantém uma variável em cada grupo para indicar que um host IGMPV1 está naquele grupo.

Para a implementação realizada no switch é mantida uma variável para indicar qual é a versão atual do

protocolo usada pelo roteador, dependendo das queries recebidas. Diferenciamos as versões verificando o Max Response Time no pacote, se for zero é IGMPV1 senão é IGMPV2. Ao estar no modo IGMPV1 não são repassados Leaves para o roteador, e não são aceitos reports IGMPV2. Para voltar ao modo IGMPV2 uma thread é criada, esta thread fica esperando por um sinal de um semáforo, este sinal é enviado ao receber uma query IGMPV1. O contador da thread é configurado para 400 segundos, que é o tempo descrito no RFC pra que se volte a operar em modo IGMPV2. Ao receber uma query IGMPV1 o contador é reinicializado, e a thread continua contando o tempo, se nenhuma query IGMPV1 for gerada a thread contará até zero e irá voltar para o modo IGMPV2. A Figura 30 resume o algoritmo de compatibilidade explicado nesta sessão através de um diagrama.

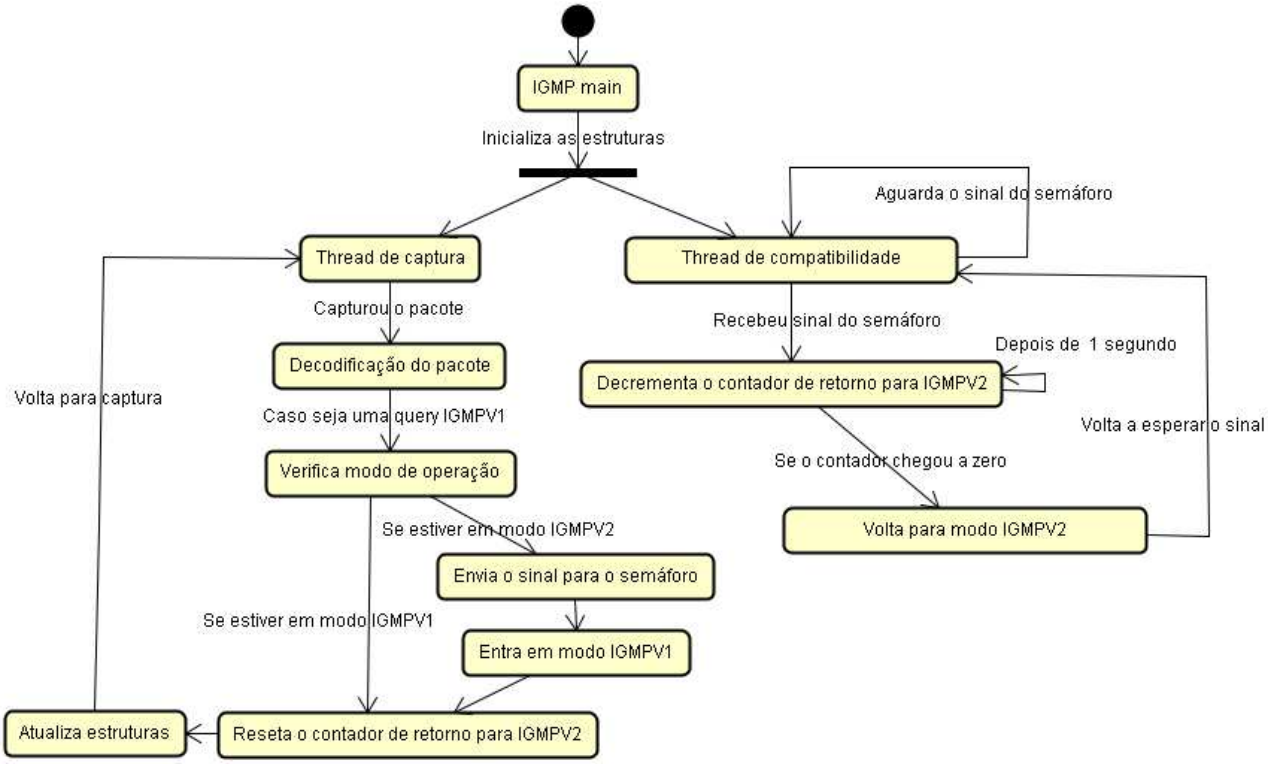

Figura 30 - Fluxo do algoritmo de compatibilidade

5.7 Experimentos e Resultados

Depois da implementação foram realizados testes para verificar o correto funcionamento do protocolo no ambiente de testes criado. Os testes validaram a implementação do protocolo IGMPV1 e IGMPV2 gerando tráfego para diferentes grupos no servidor e gerando diferentes requisições de grupos nos clientes. Também foi mudada a versão usada do protocolo no shell do Xorp para testar a compatibilidade entre o IGMPV1 e o IGMPV2.

Foram criados três grupos multicast gerando streams para eles no servidor de vídeos com o VLC. Cada um dos três clientes usados foi adicionado a um grupo usando o VLC como cliente para cada grupo em cada cliente. Para facilitar os testes o limite de queries foi diminuído para duas queries no algoritmo e o tempo entre as queries foi diminuído para 10 segundos no roteador. O monitoramento do comportamento do

protocolo é feito pelo terminal emulado através da serial com o MINICOM gerando mensagens de depuração no console. O monitoramento dos clientes é feito com uma sessão do VNCVIEWER para cada cliente e o Wireshark para verificar o tráfego dos streams. A conexão ethernet do computador de desenvolvimento é usada para enviar o executável do algoritmo usando o YAWGET. A Figura 31 mostra a configuração da arquitetura usada para realizar os testes do protocolo IGMP Snooping desenvolvido tanto para a versão um quanto para a versão dois do IGMP.

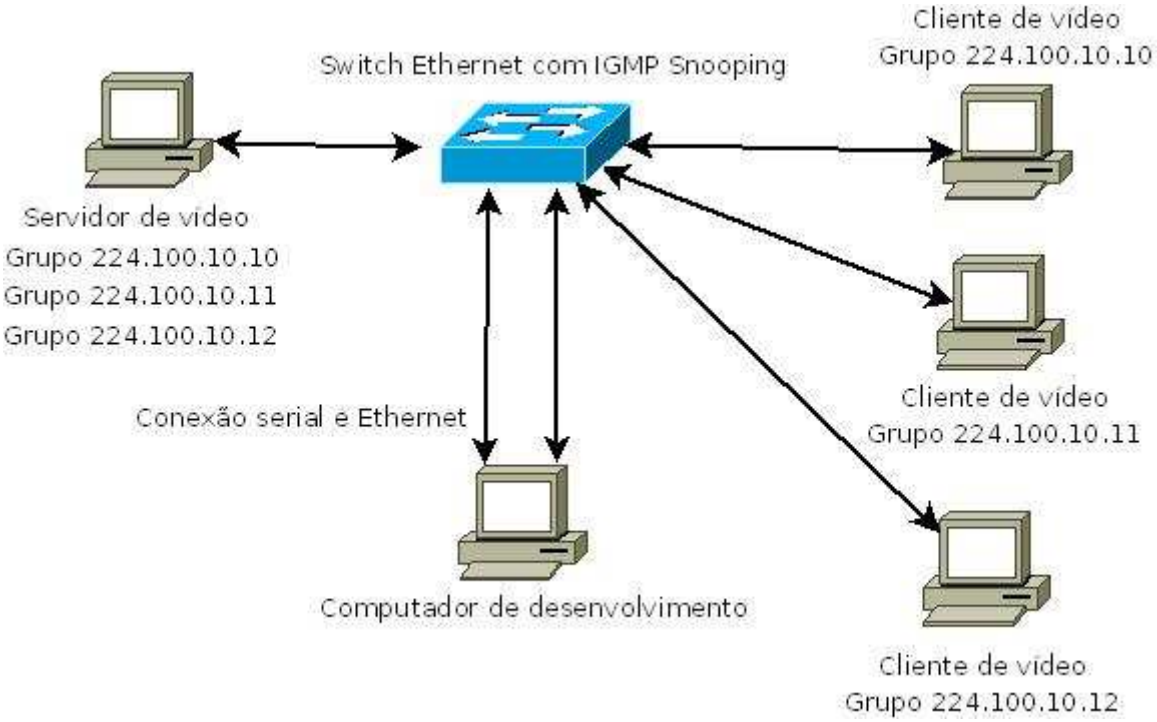

Figura 31 – Configuração da arquitetura para os testes do IGMP Snooping desenvolvido

No primeiro teste o roteador e os clientes estão em modo IGMPV2. No computador de desenvolvimento podemos ver as mensagens de depuração geradas pelo algoritmo. Elas mostram os pacotes IGMP capturados, mostrando se é um Join, Report, query ou query específica. Também são mostrados os grupos de cada mensagem e a estrutura de dados atualizada mostrando os grupos atuais da árvore a cada modificação na estrutura de dados dos grupos multicast. A Figura 32 mostra que foram capturados os reports dos clientes e foi atualizada a estrutura com os três grupos multicast usados no teste.

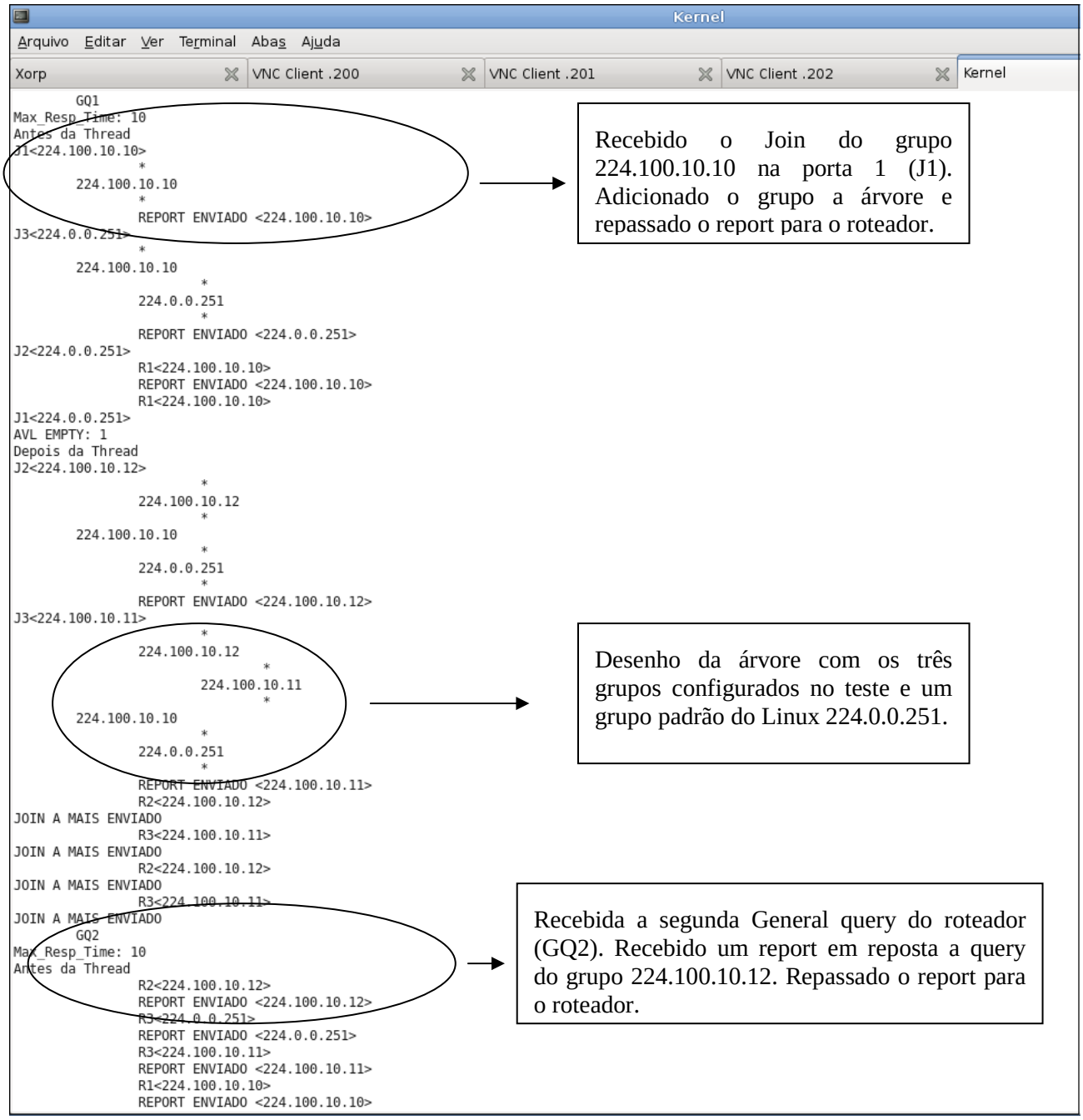

Figura 32 - Mensagens de depuração geradas pelo algoritmo

Para verificar se cada cliente recebeu o fluxo correto de acordo com grupo multicast solicitado usamos o Wireshark em cada um dos clientes, filtrando os pacotes UDP que são utilizados pelo servidor ao gerar os streams de vídeo. Cada janela é uma sessão do VNCVIEWER em um dos clientes.

Podemos verificar que cada cliente recebeu somente o fluxo no grupo no qual ele faz parte verificando a captura dos pacotes em cada janela do Wireshark no campo IP destino em destaque na Figura 33. O VLC renderiza o vídeo de cada grupo em cada um dos clientes, o que em conjunto com a captura do Wireshark mostra que o tráfego está sendo redirecionado corretamente pelo switch Ethernet com o IGMP Snooping desenvolvido.

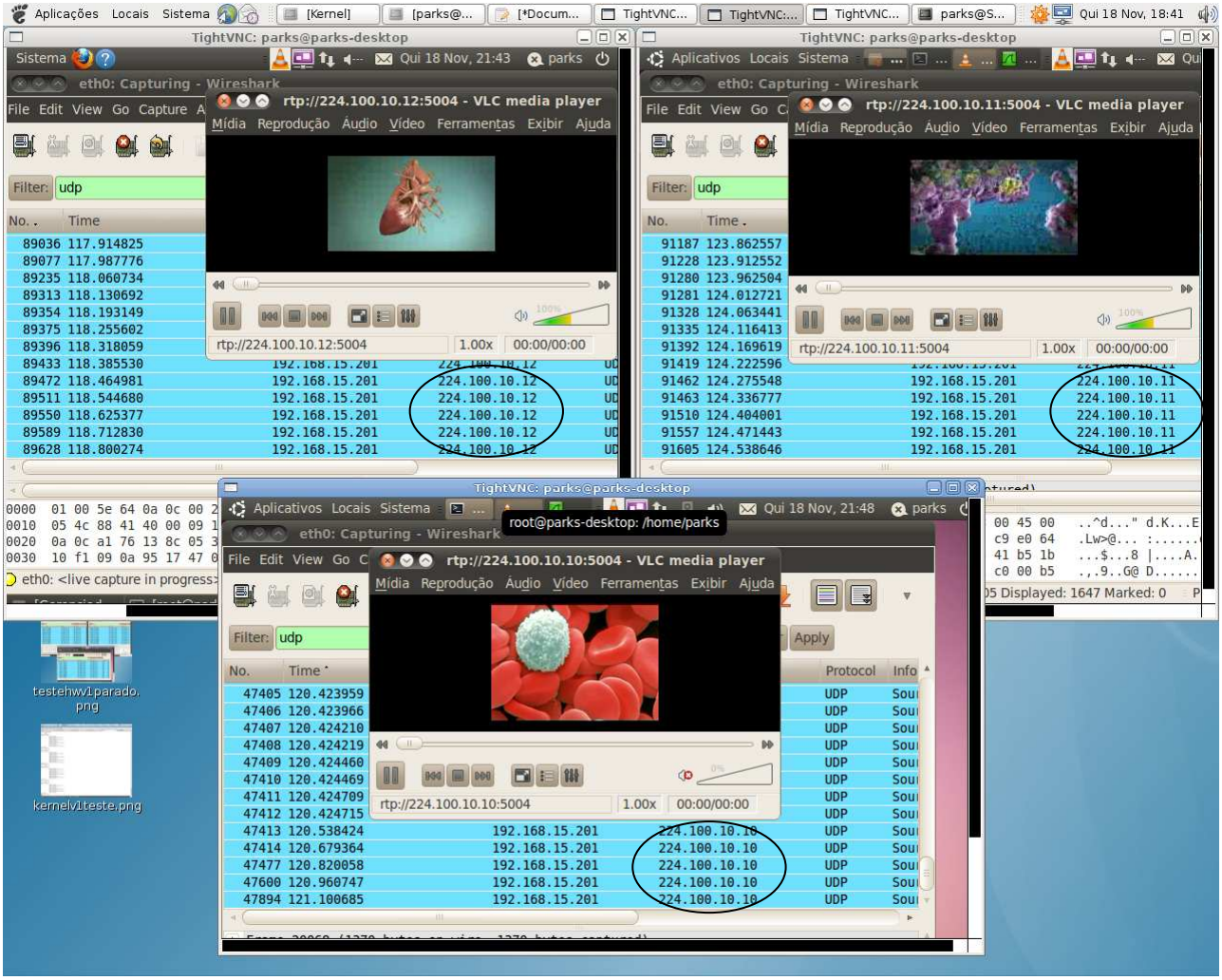

Figura 33 - Clientes recebendo o fluxo de seus grupos multicast

Para testar o Leave a execução do vídeo do cliente que faz parte do grupo 224.100.10.10 foi parada, com isto é gerado um pacote do tipo Leave por este cliente para o grupo. A Figura 34 mostra o recebimento do Leave pelo algoritmo, o recebimento das queries específicas, a retirada do cliente do grupo e conseqüentemente a retirada do grupo da estrutura de dados de grupos multicast, pois somente este cliente está cadastrado no grupo 224.100.10.10.

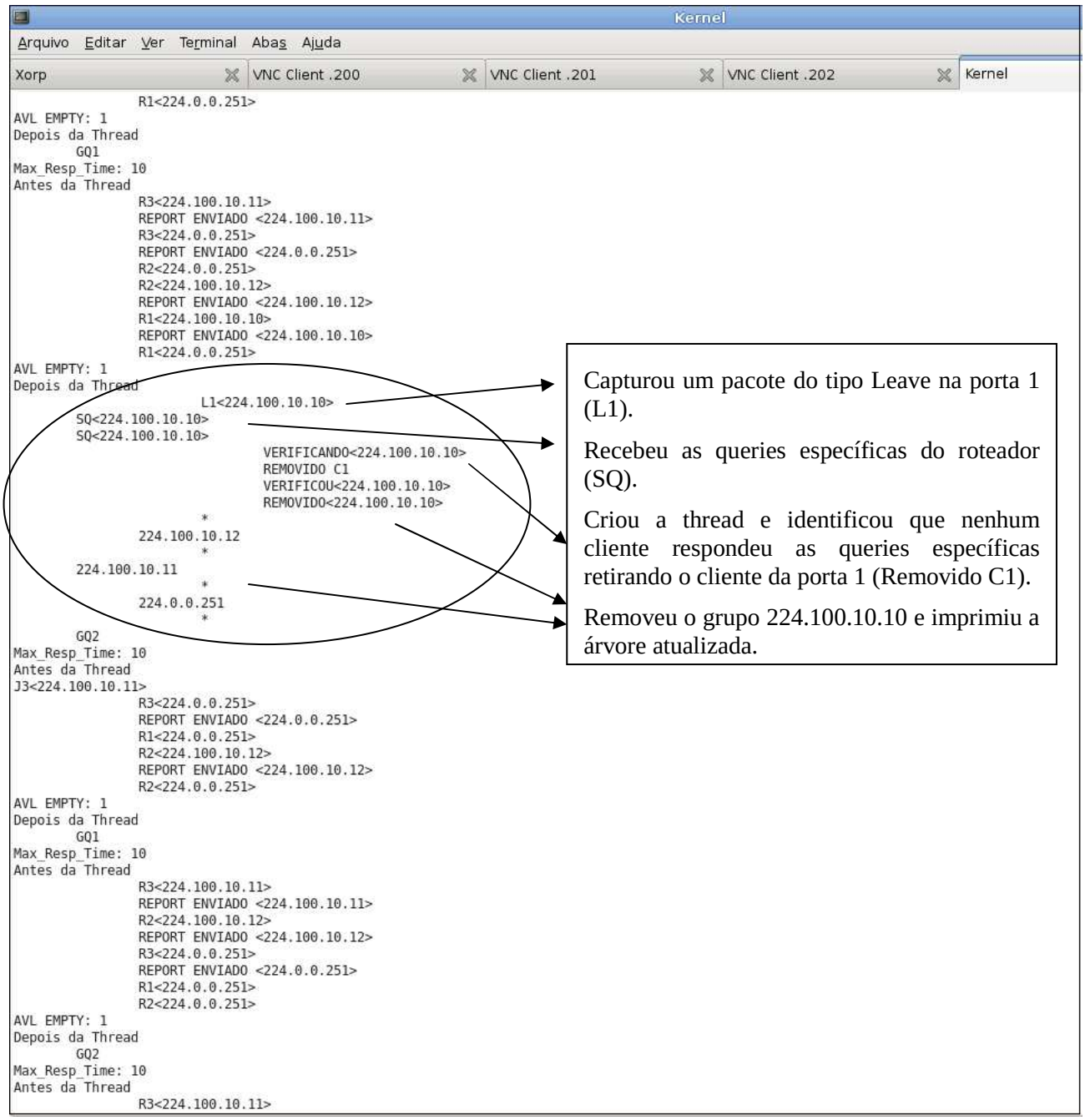

Figura 34 - Depuração do Leave no algoritmo

Nos clientes na janela do Wireshark podemos verificar que o fluxo do grupo retirado foi interrompido no cliente ao parar o vídeo no VLC, mostrando que o Leave está funcionando corretamente. A Figura 35 mostra o resultado nos clientes após a geração do leave.

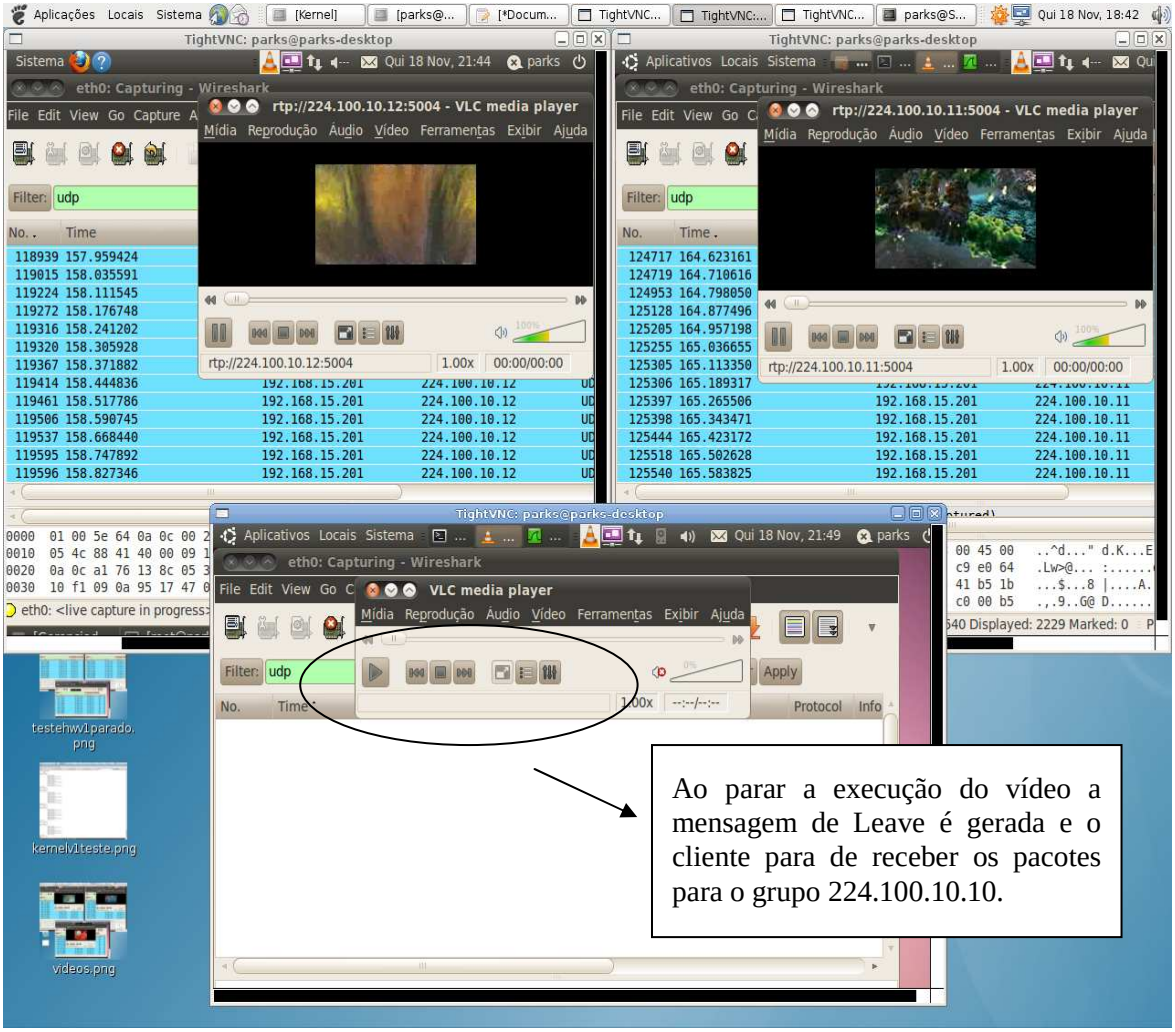

Figura 35 – Fluxo nos clientes após o leave em um dos clientes

A compatibilidade com o IGMPV1 foi testada configurando o Xorp para o modo IGMPV1 nas duas interfaces Ethernet. Na depuração do algoritmo podemos verificar a detecção de queries IGMPV1 causando a mudança do modo de execução do algoritmo. A Figura 36 mostra as mensagens de depuração do algoritmo no momento em que o modo de operação do algoritmo foi modificado para manter a compatibilidade com o IGMPV1.

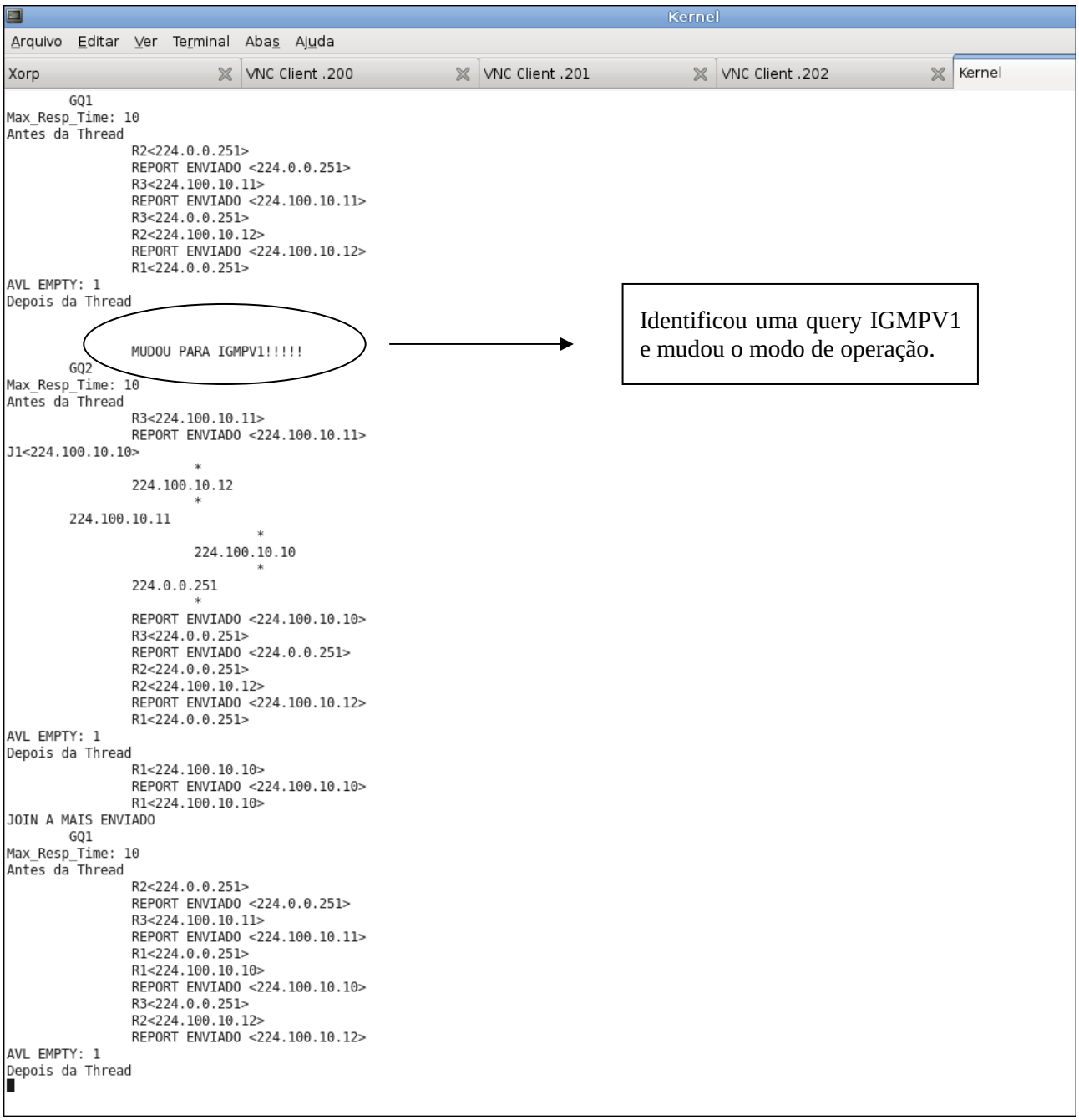

Figura 36 - Mensagens de depuração da compatibilidade com IGMPV1

A execução correta do algoritmo no modo de compatibilidade com IGMPV1 também foi verificada nos clientes, mostrando que o fluxo de cada grupo foi encaminhado para cada cliente corretamente através do Wireshark e do VLC.

A principal diferença entre o protocolo IGMPV2 e IGMPV1 é mostrada ao parar a execução de um dos clientes em modo IGMPV1. Na Figura 37 podemos verificar que o fluxo do grupo continua sendo enviado para o cliente, pois no IGMPV1 não existe o Leave, com isto é necessário esperar várias queries não serem respondidas para que a entrada da tabela seja retirada e o cliente pare de receber o fluxo do grupo multicast.

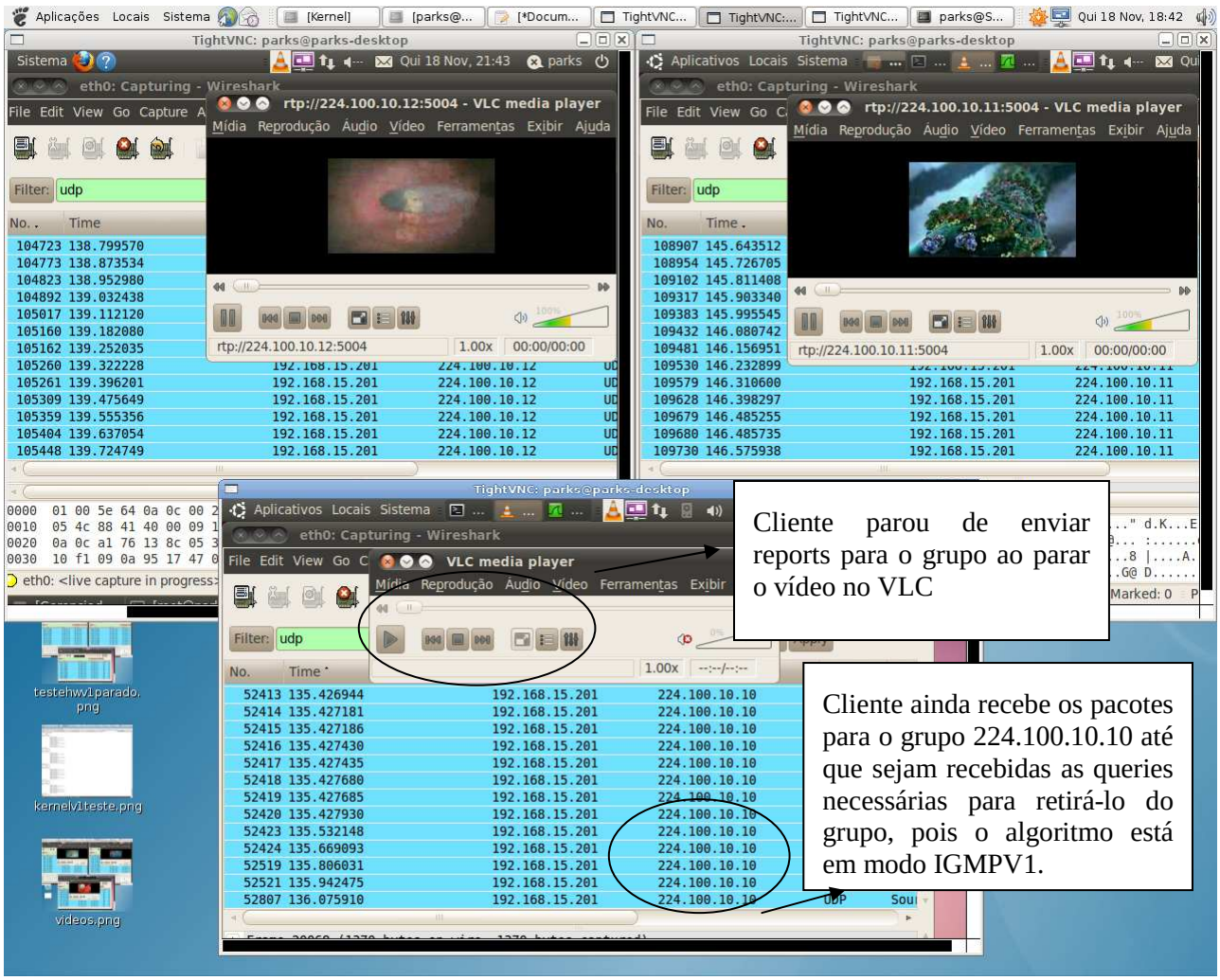

Figura 37 - Fluxo nos clientes em modo IGMPV1 após a saída de um cliente

6. SIMULADOR DE HOSPEDEIROS IGMP E SWITCH MULTICAST

O desenvolvimento do simulador justifica-se pela necessidade de testar o desempenho do algoritmo com um número de hospedeiros e grupos maior do que aquele que a plataforma de *hardware* dá suporte, e também de aumentar as possibilidades de depuração do algoritmo quanto ao seu consumo de memória e funcionalidades. Além do mais, não se encontrou no mercado uma plataforma de hardware com suporte à versão 3 do protocolo IGMP. Assim, a depuração desta versão acontecerá, pelo menos inicialmente, apenas sobre o simulador.

O simulador executa as mesmas tarefas que o *switch Ethernet* quanto ao protocolo IGMP, isto é, repassa os pacotes *multicast* para as portas corretas, de acordo com a tabela de grupos. O simulador utiliza VLANs na interface *lo* do Linux para os clientes, para ser possível a utilização de um grande número de clientes sem utilizar várias placas de rede. Além disso, o comportamento de hospedeiros IGMP também foi desenvolvido de forma que estes utilizem as referidas interfaces virtuais para requisitar conteúdo de algum grupo.

O "módulo multicast" captura o tráfego *multicast* e repassa para os clientes de acordo com a tabela de grupos, além de repassar os pacotes *multicast* da interface *eth0* para a *lo* onde o algoritmo principal está capturando os pacotes IGMP. Pacotes de controle são enviados do algoritmo principal para o "módulo multicast" quando acontece uma atualização na tabela de grupos através de *sockets*. Os clientes geram pacotes IGMP para entrar e sair de grupos de acordo com o conteúdo de um arquivo de configuração. Este arquivo possui as ações e os grupos multicast em hexadecimal para cada uma dessas ações. Também possui o tempo em cada ação deve ser executada pelo cliente com granularidade de 1 segundo. As ações dos clientes são o envio de mensagens IGMP Join e Leave para os grupos correspondentes.

A Figura 38 mostra a arquitetura do simulador.

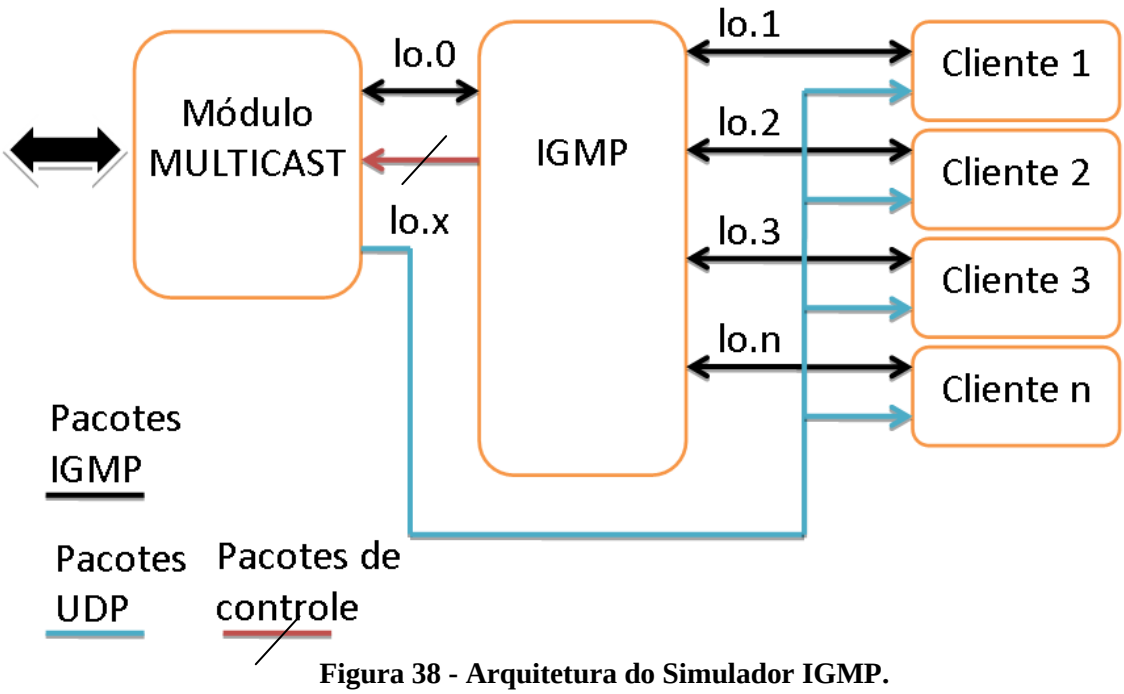

Figura 38 - Arquitetura do Simulador IGMP.

6.1 Utilizando VLANs

As VLANS são interfaces virtuais que podem ser criadas em uma interface padrão, no Linux o seu suporte é ativado em um módulo do kernel, no pacote Ethernet é adicionada uma tag para identificação da VLAN através de um número. Com o suporte do Linux pacotes recebidos em uma interface são repassados para suas interfaces virtuais de acordo com a tag, ao repassá-los para a interface virtual o Linux retira a tag, com isto o pacote capturado na interface virtual é tratato como se tivesse sido capturado em uma interface comum. Ao enviar um pacote em uma VLAN o Linux automaticamente adiciona a tag da interface virtual correspondente. O formato da tag adicionada ao cabeçalho Ethernet é mostrado na Figura 39 a seguir.

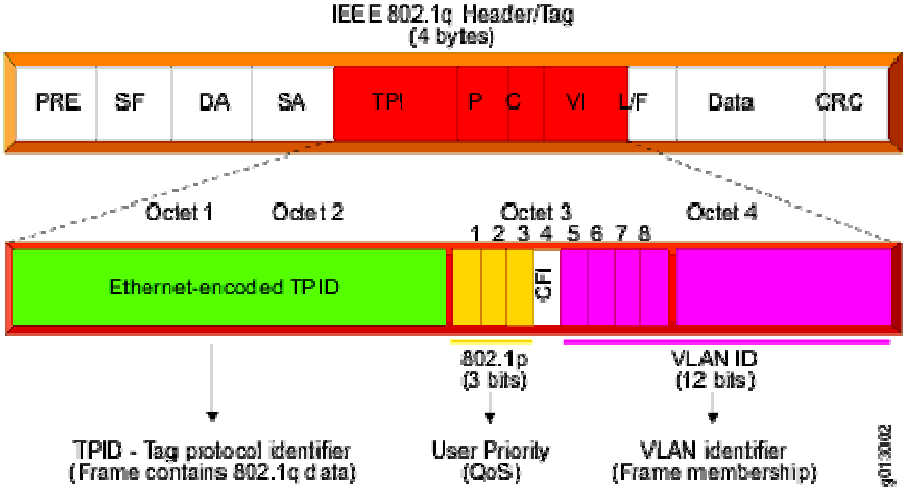

Figura 39 - Formato do cabeçalho Ethernet com a tag de VLAN [JUN10]

No escopo deste trabalho o campo importante é o ID que identifica a VLAN, os outros são mantidos em seus valores padrão. Para configurar VLANs no Linux usamos o comando “vconfig” passando a interface

que queremos adicionar a VLAN e o seu identificador. Para mantermos o mesmo estilo de programação usado no Switch Ethernet da plataforma de desenvolvimento, a captura se dará apenas na interface *lo* do Linux e não em todas as VLANs, com isto devemos gerar as tags de VLAN manualmente para enviar os pacotes em cada interface virtual, assim como foi feito para o switch ao adicionar a tag especial da plataforma. Para o recebimento a tag de VLAN também é decodificada, assim o algoritmo utilizado para executar o IGMP Snooping no switch Ethernet é totalmente utilizado no simulador, somente as funções dependentes de hardware são modificadas e incluídas de acordo com uma macro que seleciona a plataforma que o código deve ser compilado. Deste modo é garantido que o algoritmo principal do IGMP Snooping executado no switch é mesmo usado no simulador. Cada interface virtual corresponde a uma porta do switch, cada cliente é conectado em uma interface virtual.

A Figura 40 mostra as interfaces virtuais para cada cliente configuradas na interface *lo* do Linux através do comando “vconfig”.

Figura 40 - VLANs criadas para cada cliente

6.2 Módulo Multicast

O módulo Multicast é uma emulação do papel do switch do hardware, ele recebe e envia pacotes para o roteador na interface *eth0*, encaminha os pacotes IGMP para o algoritmo na interface virtual "lo.0" e para os clientes na interface virtual "lo.X" correspondente aos clientes. Possui uma tabela interna dos grupos multicast, simulando a tabela de grupos do hardware. Essa tabela é atualizada quando há alguma mudança na base de dados dos grupos multicast do algoritmo do IGMP Snooping através de um socket que faz a comunicação entre os processos. Se um pacote capturado na interface *eth0* é IGMP, esse é repassado para o algoritmo principal na interface "lo.0", esta é a porta de comunicação com o roteador. Se o pacote recebido for UDP multicast é realizada uma pesquisa na tabela para encaminhar o pacote nas interfaces que fazem parte do grupo multicast em suas VLANs correspondentes. Os clientes enviam suas requisições de grupos multicast em suas interfaces virtuais, o algoritmo principal captura os pacotes somente na interface lo, com isto ele recebe os pacotes com a tag de VLAN das portas, se a tag é da VLAN zero esta pertence a conexão com o roteador, se for de outros IDs de VLAN estes pertencem aos clientes. Portanto, o módulo multicast faz toda a conexão dos clientes e do roteador com o algoritmo principal. O algoritmo principal somente captura de uma interface da mesma maneira que foi realizado no switch Ethernet (no simulador captura da interface *lo*, no switch captura da interface *eth1*). Com isto é mantida a compatibilidade das duas implementações podendo assim o algoritmo ser validado com um número maior de clientes.

6.3 Envio de pacotes com a Libnet

Libnet é uma biblioteca que facilita a criação e envio de pacotes [SCH10]. Ela foi utilizada no simulador, pois com a utilização de VLANs do Linux a tag para cada VLAN é automaticamente adicionada ao pacote ao enviá-lo pela interface virtual, no caso do hardware como ele usava uma tag especial não foi possível usar a biblioteca para facilitar o envio dos pacotes, pois ela não gera este tipo de tag. Além disso, na versão de hardware do algoritmo nenhum pacote é criado, todos os pacotes são capturados e encaminhados mudando apenas a tag, no simulador são gerados pacotes para simular os clientes nas interfaces virtuais, pois não foi possível usar o VLC em cada interface virtual. Além do VLC não oferecer uma forma de usar as interfaces virtuais o sistema não iria suportar a carga de trabalho da execução de vários vídeos com um número grande de clientes em um mesmo computador.

A Tabela 7 mostra as funções usadas para a geração e envio dos pacotes. Os protótipos foram omitidos pois são passados muitos parâmetros para as funções. Estes parâmetros são os diversos componentes das camadas que o pacote utiliza como o tipo da mensagem do IGMP, o TTL do pacote entre outros.

Tabela 7 - Funções da biblioteca Libnet utilizadas para geração e o envio de pacotes

Nome da função	Descrição
libnet_init_packet	Usada para inicializar a estrutura da Libnet para o envio do pacote na interface.
libnet_get_hwaddr	Usada para pegar o endereço MAC da interface para o envio do pacote.

libnet_get_ipaddr4	Usada para adquirir o IP da interface.
libnet_build_igmp	Constrói a o cabeçalho IGMP do pacote com os argumentos passados.
libnet_build_ipv4_options	Constrói as opções do protocolo IP do pacote.
libnet_build_ipv4	Constrói a o cabeçalho IP do pacote.
libnet_autobuild_ethernet	Constrói a o cabeçalho Ethernet do pacote, só é necessário passar o MAC destino e o protocolo da camada superior, os outros campos são gerados automaticamente.
libnet_write	Envia o pacote gerado na interface.
libnet_destroy	Libera as estruturas geradas, usado após ter enviado o pacote.

6.4 Simulador de cliente IGMPv1 e IGMPV2

Os clientes desenvolvidos para o simulador são processos distintos que simulam um hospedeiro IGMPV1 ou IGMPV2, gerando requisições de grupos multicast e respondendo as queries do roteador de acordo com a rfc esperando um tempo randômico entre 0 e 10 segundos para enviar a resposta a querie. O envio é feito através da VLAN do cliente com isto a tag é automaticamente inserida pelo Linux. Também é realizada a captura das queries e do stream de vídeo na interface virtual utilizando a Libpcap da mesma maneira que realizada na implementação do algoritmo principal. Os pacotes gerados e recebidos pelo cliente são escritos em um arquivo de log para posterior análise. O cliente lê um arquivo de ações que é gerado por um programa escrito em Java para gerar randomicamente as ações a serem executados pelos clientes, assim não é preciso especificar manualmente o que o cliente deve fazer, facilitando e automatizando os testes no simulador. O tempo de execução do cliente inicia em zero e a cada segundo é verificado se uma ação deve ser executada. A Figura 41 mostra um exemplo deste arquivo de configuração com os tempos em que cada ação deve ser executada na primeira coluna.

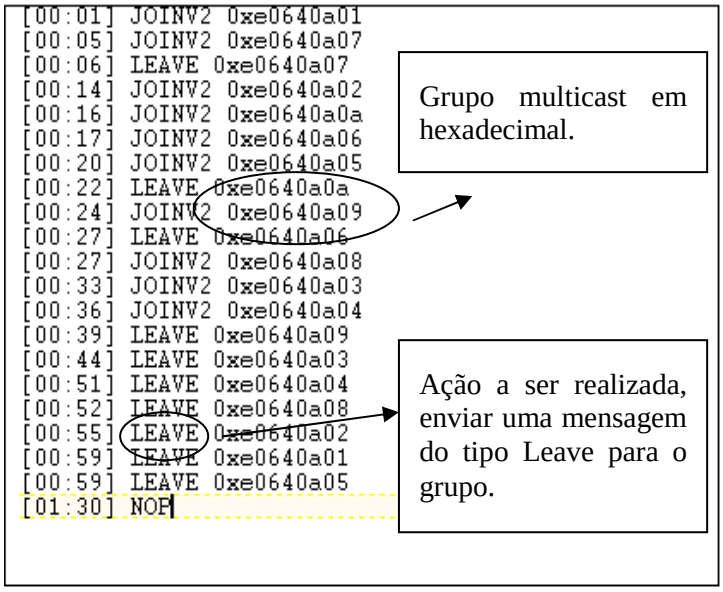

Figura 41 - Exemplo de arquivo de ações gerado automaticamente pelo gerador de ações

Cada cliente gera seu arquivo de log com as ações executadas e os pacotes capturados. A Figura 42 mostra um exemplo de um arquivo de log gerado por um cliente após a sua execução, o cliente faz um Join no grupo 0xe0640a01 que é o grupo 224.100.10.1 e começa a receber o tráfego para o grupo. Este tráfego são os pacotes UDP gerados pelo servidor de vídeo usando o VLC para o grupo multicast. Na primeira coluna do log são mostrados os tempos em que ocorreram os eventos no cliente de acordo com o seu tempo de execução.

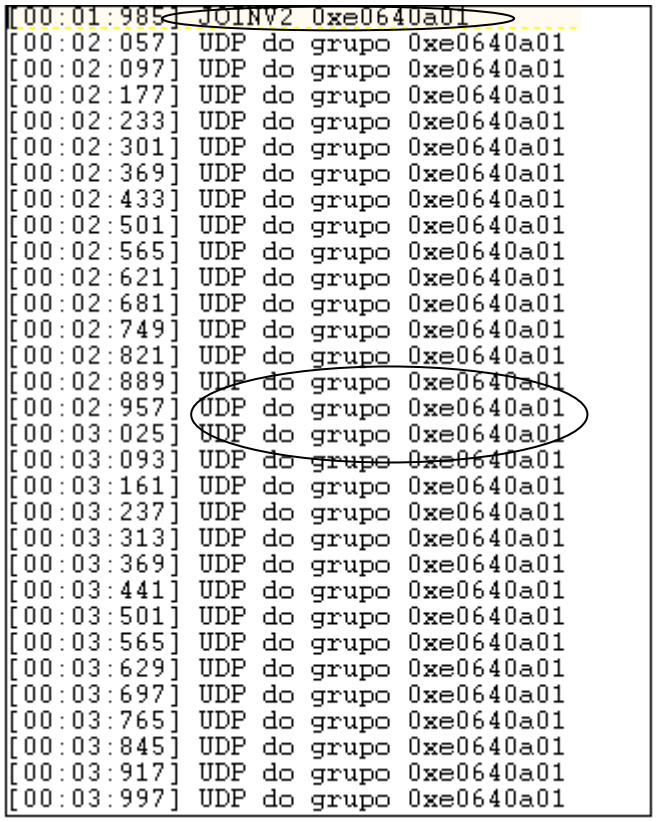

Figura 42 - Exemplo de log gerado por um cliente

6.5 Testes de funcionalidade e desempenho

Foram realizados dois testes no simulador, um com 8 clientes e 10 grupos e outro com 16 clientes e 8 grupos. Para o primeiro teste foram criadas as VLANs com o script que gera automaticamente as VLANs de acordo com o número de clientes requisitado. No servidor de vídeos são gerados os fluxos para cada um dos grupos multicast usados pelos clientes, esta geração de fluxos também é automatizada com o uso de um script. Depois foram geradas as ações para os clientes com o gerador automático de ações. Para fazer a simulação primeiro é executado o módulo multicast, depois o algoritmo do IGMP Snooping compilado para a versão simulador, estes dois processos criam uma comunicação através de sockets para trocar a informação da atualização da tabela de grupos multicast. O módulo multicast também gera mensagens de depuração mostrando os pacotes IGMP recebidos por ele dos clientes e a sua atualização da tabela de grupos de software de acordo com a atualização da tabela do algoritmo do IGMP Snooping. A Figura 43 mostra as mensagens de depuração do módulo multicast com as mensagens trocadas entre ele e o algoritmo principal.

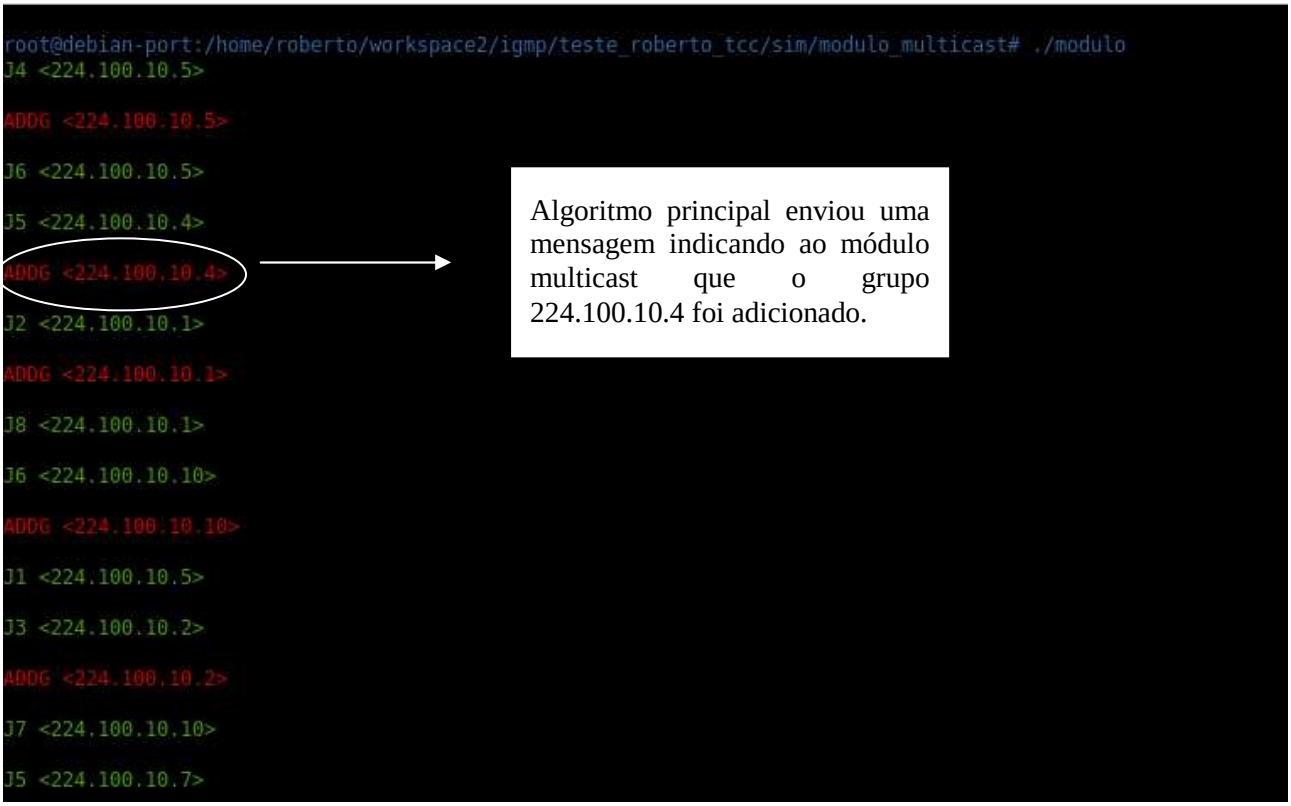

Figura 43 - Mensagens de depuração do módulo multicast

Depois executamos um script para inicialização dos clientes, cada cliente lê seu arquivo de ações e começa a gerar os reports, leaves e a gravar seu log. A Figura 44 mostra os clientes executando em segundo plano no terminal do Linux as ações contidas nos seus arquivos de configuração, são mostrados os nomes das interfaces virtuais que representam cada cliente, as ações e os grupos multicast em hexadecimal usados em cada ação.

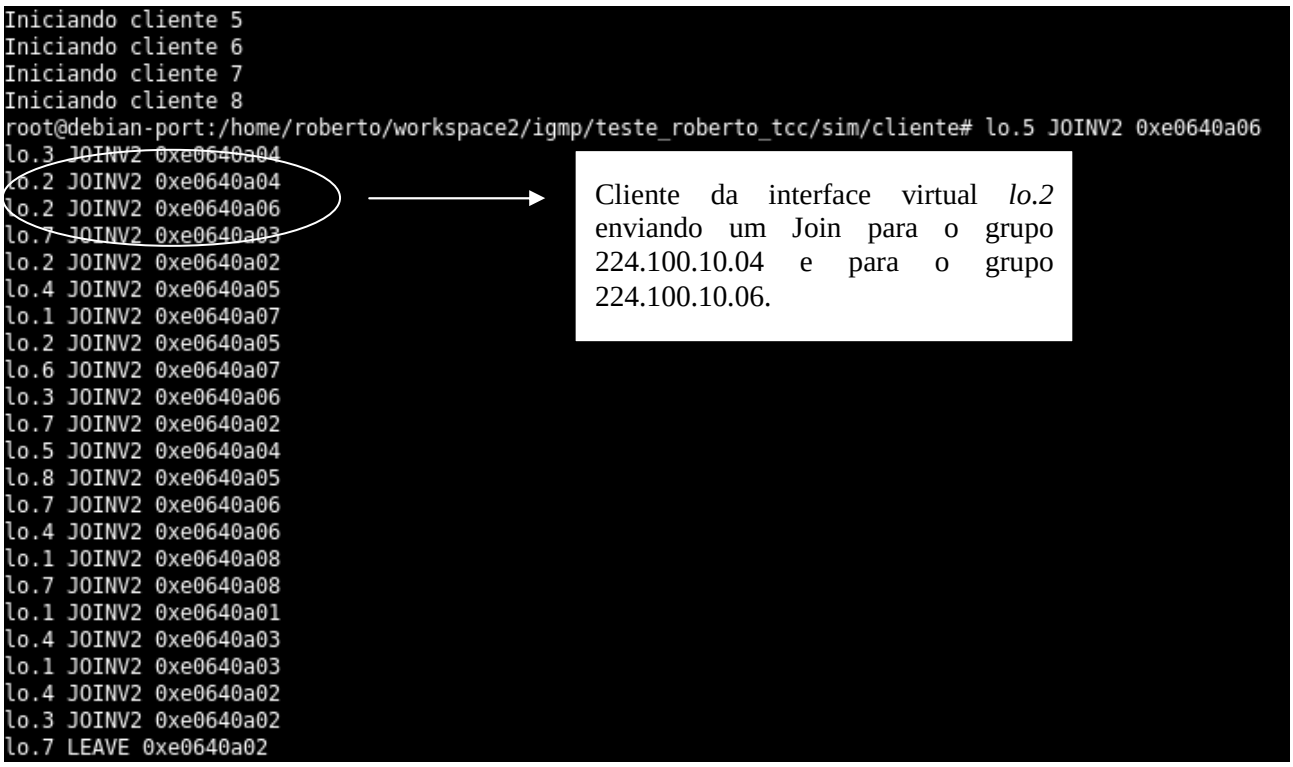

Figura 44 - Clientes executanto as ações descritas no seus arquivos

Como nos testes no switch podemos ver as mensagens de depuração do algoritmo do IGMP Snooping no console. A Figura 45 mostra os Join e reports dos clientes sendo recebidos pelo algoritmo principal e esses pacotes sendo repassados para o roteador. Também mostra a impressão da árvore de grupos a cada inclusão de um novo grupo ao receber um Join de um cliente.

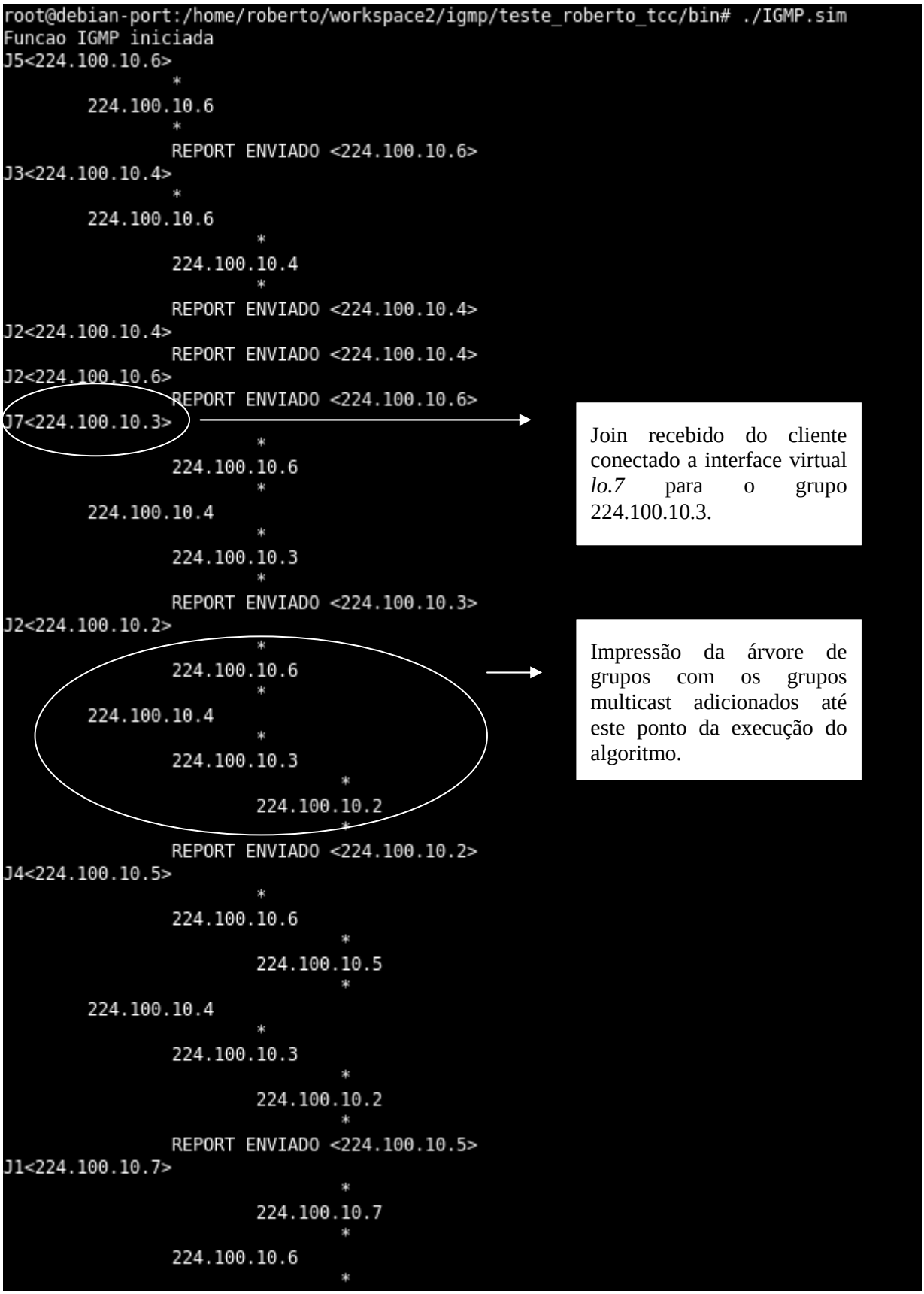

Figura 45 - Mensagens de depuração do IGMP Snooping no simulador

Podemos verificar que os grupos são inseridos na tabela de grupos multicast e os reports e queries são recebidos e encaminhados corretamente analisando cada log gerado pelos clientes, comparando com suas listas de ações verificando se eles receberam o fluxo multicast para os grupos aos quais eles foram incluídos.

Os logs e ações foram verificados para as duas simulações, estes validaram a execução do algoritmo com um maior número de clientes. As listas de ações usadas pelos clientes nos dois testes e os logs gerados por eles podem ser verificados no Apêndice A.

7. EVOLUÇÃO E CONCLUSÕES

7.1 Especificação do IGMP Versão 3

A versão 3 adiciona o suporte para "*source filtering*" [CAI02], isto é, a capacidade de um sistema reportar o interesse em receber pacotes *mulicast* apenas de endereços fonte específicos, necessária para o Source-Specific Multicast SSM [SPR03]. A versão 3 foi projetada para ser interoperável com as versões 1 e 2. Para utilizar todas as funcionalidades do IGMPv3, a interface do serviço IP do sistema deve dar suporte à seguinte operação:

IPMulticastListen (socket, interface, multicast-address, filter-mode, source-list).

- "socket" é um parâmetro específico da implementação usado para distinguir entre diferentes entidades requerentes (por exemplo, programas ou processos) dentro do sistema, o parâmetro socket das chamadas de sistema do BSD Unix é um exemplo específico.
- "interface" é um identificador local da interface de rede em que recepção do endereço multicast especificado deve ser habilitado ou desabilitado. Interfaces podem ser físicas (por exemplo, uma interface Ethernet) ou virtuais (por exemplo, uma VLAN (virtual LAN)). Uma aplicação pode permitir um valor especial não especificado ser passado como parâmetro, para que a interface padrão do sistema possa ser usada. Se a recepção do mesmo endereço de multicast é desejada em mais de uma interface, IPMulticastListen é chamado separadamente para cada interface desejada.
- "multicast-address" é o endereço IP multicast, ou grupo, para que se refere o pedido. Se a recepção de mais de um endereço multicast em uma determinada interface é desejada, IPMulticastListen deve ser chamado separadamente para cada endereço multicast desejado.
- "filter-mode" pode ser INCLUDE ou EXCLUDE. No modo INCLUDE, a recepção de pacotes enviados para o endereço multicast especificado é solicitada somente dos endereços IP de origem constantes no parâmetro "source-list". No modo EXCLUDE, a recepção de pacotes enviados para o endereço multicast especificado é solicitada a todos os endereços IP de origem exceto aqueles listados no parâmetro "source-list".
- "source-list" é uma lista não ordenada de zero ou mais endereços IP unicast de onde a recepção multicast é ou não desejada, dependendo do modo do filtro. Uma implementação pode impor um limite sobre o tamanho da "source-list", mas esse limite não deve ser inferior a 64 endereços por lista. Quando uma operação faz com que a lista de fontes ultrapasse o tamanho limite, a interface de serviço deve retornar um erro.

Para uma dada combinação de "socket", "interface" e "multicast-address", apenas um "filter-mode" e uma única "source-list" pode estar em vigor em qualquer instante de tempo. No entanto, o "filter-mode" ou a "source-list", ou ambos, podem ser alterados por chamadas de IPMulticastListen subsequentes que especificam o mesmo "socket", "interface" e "multicast-address". Cada pedido subsequente substitui completamente qualquer pedido anterior para o dado "socket", "interface" e "multicast-address".

As versões anteriores do IGMP não ofereciam suporte a filtros de origem e tinham uma interface de serviço mais simples que consistia em operações JOIN e LEAVE, que ativavam ou desativavam a recepção para um dado endereço *multicast* de todas as fontes em uma interface. As operações equivalentes nesta nova interface de serviços são:

- IPMulticastListen (“socket”, “interface”, “multicast-address”, EXCLUDE, ()), equivalente ao JOIN.
- IPMulticastListen(“socket”, “interface”, “multicast-address”, INCLUDE ()), equivalente ao LEAVE.

O representação () implica uma lista de fontes vazia. Um exemplo de descrição de uma API para proporcionar as capacidades descritas na interface de serviço pode ser encontrado em [THA04].

Se o modo do filtro solicitado é INCLUDE e a lista de fontes está vazia, então a entrada correspondente ao pedido é eliminada. Se o modo do filtro solicitado é EXCLUDE e a lista de fontes não está vazia, então a entrada correspondente ao pedido é alterada.

7.1.1 Formato das mensagens

As mensagens IGMP são encapsuladas em datagramas IPv4, com um número de protocolo IP de 2. Todas as mensagens IGMP são enviados com um IP Time-to-Live de 1, e com a opção IP ROUTER ALERT [KAT97], em seu cabeçalho IP.

Existem dois tipos de mensagens IGMP relativas ao protocolo IGMPv3, conforme descreve a Tabela 8.

Tabela 8 - Formato das mensagens IGMPv3.

Número do tipo (hex)	Nome da mensagem
0x11	Membership Query
0x22	Version 3 Membership Report

Uma implementação de IGMPv3 também deve dar suporte aos três seguintes tipos de mensagens descritos na Tabela 9, para interoperabilidade com as versões anteriores do IGMP.

Tabela 9 - Mensagens para compatibilidade com versões anteriores do IGMP.

Número do tipo (hex)	Nome da mensagem
0x12	Version 1 Membership Report
0x16	Version 2 Membership Report
0x17	Version 2 Leave Group

A Figura 46 descreve o formato da mensagem Membership Query para o IGMPv3, enquanto a Figura 47 descreve o formato da mensagem Membership Report para esta mesma versão do protocolo.

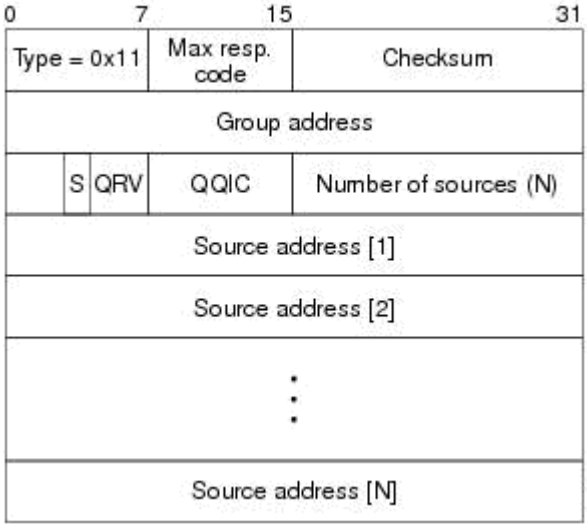

Figura 46 - Formato da mensagem Membership Query do IGMPv3 [CIS02].

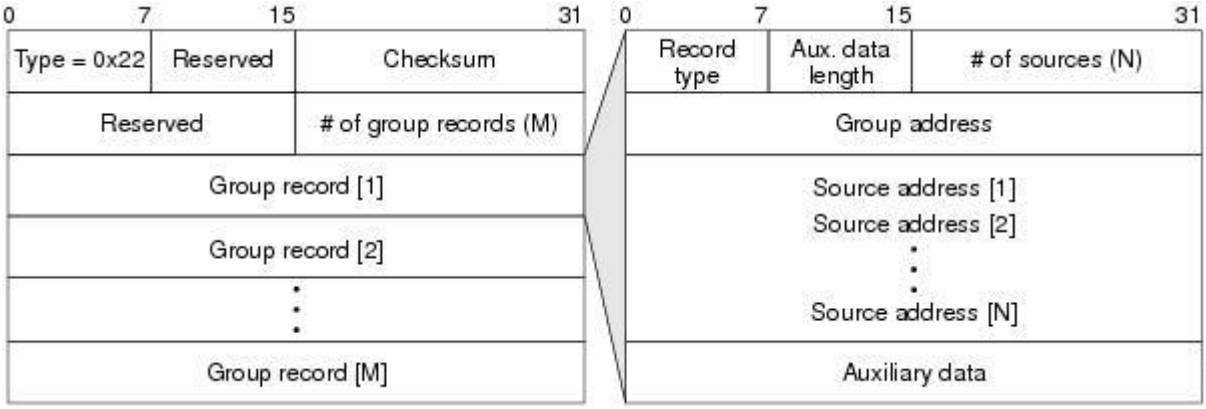

Figura 47 - Formato da mensagem Version 3 Membership Report [CIS02].

7.1.2 Tipos de Group Records

Existem vários tipos de registros de grupo (Group Records) que podem ser incluídos em um Version 3 Membership Report, conforme lista a Tabela 10. A seguir descreve-se cada um destes Group Records.

Tabela 10 - Tipos de Group Records.

Número do Group Record	Nome do Group Record
1	MODE_IS_INCLUDE
2	MODE_IS_EXCLUDE
3	CHANGE_TO_INCLUDE_MODE
4	CHANGE_TO_EXCLUDE_MODE
5	ALLOW_NEW_SOURCES
6	BLOCK_OLD_SOURCES

- MODE_IS_INCLUDE - indica que a interface está com modo de filtro INCLUDE para o endereço *multicast* especificado. Os campos Source Address formam a lista de fontes.

- MODE_IS_EXCLUDE - indica que a interface está com modo de filtro EXCLUDE para o endereço multicast especificado. Os campos Source Address formam a lista de fontes.
- CHANGE_TO_INCLUDE_MODE - indica que a interface mudou seu modo de filtro para INCLUDE para o endereço *multicast* especificado. Os campos Source Address formam a nova lista de fontes.
- CHANGE_TO_EXCLUDE_MODE - indica que a interface mudou seu modo de filtro para EXCLUDE para o endereço *multicast* especificado. Os campos Source Address formam a nova lista de fontes.
- ALLOW_NEW_SOURCES - indica que os campos Source Address formam um a lista de fontes adicionais as quais o sistema deseja receber pacotes *multicast*, para o endereço *multicast* especificado. Se a mudança foi em uma lista de fontes do tipo INCLUDE, estes endereços são adicionados a lista, se for EXCLUDE, estes endereços são removidos da lista.
- BLOCK_OLD_SOURCES - indica que os campos Source Address formam um a lista de fontes as quais o sistema não deseja mais receber pacotes *multicast*, para o endereço *multicast* especificado. Se a mudança foi em uma lista de fontes do tipo INCLUDE, estes endereços são removidos da lista, se for EXCLUDE, estes endereços são adicionados a lista.

7.1.3 Compatibilidade com versões anteriores

Para determinar a versão da mensagem Membership Query utilizamos a seguinte lógica:

- IGMPv1 Query: possui tamanho de 8 octetos e o campo Max Resp Code field é zero.
- IGMPv2 Query: possui tamanho de 8 octetos e o campo Max Resp Code field não é zero.
- IGMPv3 Query: possui tamanho maior ou igual a12 octetos.

Os roteadores podem ser colocados em uma rede onde existem máquinas que ainda não foram atualizadas para IGMPv3. Para ser compatível com os hospedeiros de versão mais antiga, os roteadores IGMPv3 devem funcionar nas versões 1 e 2 em modos de compatibilidade.

Para alternar entre as versões do IGMP, roteadores mantém um temporizador para indicar a presença de hospedeiros IGMPv1 e um temporizador para hospedeiros IGMPv2 por registro de grupo [CAI02]. Quando está em modo de compatibilidade IGMPv2, o roteador trata todas as mensagens IGMPv3 como uma mensagem IGMPv2 sem nenhuma fonte e ignora mensagens do tipo BLOCK e ALLOW. Quando o modo de compatibilidade é IGMPv1, o roteador trata mensagens IGMPv3 como anteriormente, mas adicionalmente ignora mensagens LEAVE IGMPv2.

Para ser compatível com roteadores de versão mais antiga, os hospedeiros IGMPv3 devem funcionar como versões 1 e 2 em modos de compatibilidade. Uma variável é mantida por interface e é dependente da versão das General Queries recebidas nessa interface, além de utilizar um timer para voltar as versões mais recentes, se não receber mais General Queries depois de um determinado período.

7.2 Conclusões

O trabalho realizado consolidou diversos conhecimentos adquiridos na graduação, desde programação até sistemas operacionais e arquitetura de redes e de computadores. Também foi adquirido um grande conhecimento técnico de tecnologias que não foram estudadas na graduação, porém a graduação forneceu os conhecimentos teóricos necessários para o aprendizado destas tecnologias desde configurações de softwares no Linux até manipulação de seus registradores e implementação de protocolos de rede.

Dentre as dificuldades encontradas, as mais importantes foram o acesso direto a memória no Linux com a utilização de drivers. Também foram encontradas dificuldades para a compilação do Linux embarcado, o qual requereu dois tipos de compiladores cruzados e edição de scripts para que o sistema fosse compilado corretamente e fizesse o carregamento correto na plataforma de desenvolvimento.

Dentre os trabalhos futuros estão a implementação do suporte ao IGMPV3 e ao IGMP Proxy. Como foi visto, o IGMPV3 é um protocolo mais complexo e exige um hardware mais especializado que não foi encontrado no mercado. Esta implementação será realizada no simulador emulando as características necessárias para o encaminhamento dos fluxos não somente pelo endereço multicast mais também pelas fontes do fluxo.

A adaptação do trabalho para executar o IGMP Proxy para as versões 1 e 2 do protocolo IGMP, já foi realizada pelo grupo de pesquisa que trabalhou em conjunto com o autor neste trabalho de conclusão. A principal modificação foi adicionar o comportamento de roteador ao protocolo implementado gerando as queries para os hospedeiros. A implementação do IGMPV3 já está em andamento usando a mesma estrutura do IGMPV2 implementado neste trabalho, mas agora incluindo uma terceira árvore para as fontes de dados multicast usadas no protocolo IGMPV3. Como a plataforma da Micrel usada para implementação não possui suporte ao IGMPV3 e como não foi encontrada uma plataforma que possuísse este recurso, o algoritmo está sendo validado usando o simulador adaptado para a versão três do protocolo. Após também será realizada a implementação do IGMP Proxy para a versão três do IGMP.

REFERÊNCIAS BIBLIOGRÁFICAS

[BRA88] Braden, R.; Borman, D.; C, Partridge. “Computing the Internet checksum.” RFC 1071, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc1071, Setembro 1988.

[CAI02] Cain, B.; Deering, S.; Fenner, B.; Kouvelas, I.; Thyagarajan, A. “Internet Group Management Protocol, Version 3.” RFC 3376, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc3376, Setembro 2002.

[CHR06] Christensen, M. “Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches.” RFC 4541, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc4541, Maio 2006.

[CIS02] CISCO. “IP Multicast Technology Overview.” Capturado em: http://www.cisco.com/en/US/docs/ios/solutions_docs/ip_multicast/White_papers/mcst_ovr.pdf, Abril 2002.

[COR05] Corbet, J. “Linux Device Drivers.” Sebastopol:O’Reilly Media, Inc, 2005, 616p.

[CRO85] Croft, B.; Gilmore, J. “Bootstrap Protocol (BOOTP).” RFC 951, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc951, Setembro 1985.

[DEE86] Deering, S. “Host extensions for IP multicasting.” RFC 998, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc988, Julho 1986.

[DEE89] Deering, S. “Host extensions for IP multicasting.” RFC 1112, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc1112, Agosto 1989.

[FEN06] Fenner, B.; He, H.; Haberman, B.; Sandick, H. “Internet Group Management Protocol (IGMP) /Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying").” RFC 4605, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc4605, Agosto 2006.

[FEN97] Fenner, W. “Internet Group Management Protocol, Version 2.” RFC 2236, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc2236, Novembro 1997.

[FER10] Fernandes, L. G. L. “COMUNICAÇÃO ENTRE PROCESSOS.” Notas de aula. Capturado em: http://www.inf.pucrs.br/~gustavo/disciplinas/ppd/material/slides-sockets-novo.pdf, Novembro 2010.

[GAS96] Gaspary, L. P. “Multicast tutorial.” Capturado em: http://penta.ufrgs.br/redes296/multicast/tutorial.html, Dezembro 1996.

[HAB05] Haberman, B.; Martin, J. "Multicast Router Discovery." RFC 4286, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc4286, Dezembro 2005.

[IAN10] IANA. "Internet Multicast Addresses." Capturado em: http://www.iana.org/assignments/multicast-addresses/, Agosto 2010.

[JUN10] JUNIPER NETWORKS, Inc. "physical-link-overview." Capturado em: http://www.juniper.net/techpubs/software/erx/junose52/erx-product-overview/html/physical-link-overview16.html, Novembro 2010.

[KAT97] Katz, D. "IP Router Alert Option." RFC 2113, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc2113, Fevereiro 1997.

[MAL98] Malkin, G. "RIP Version 2." RFC 2453, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc2453, Novembro 1998.

[MAS07] Masters, J.; Blum, R. "Professional Linux Programming." Indianapolis:Wiley Publishing, Inc, 2007, 507p.

[MIC05] MICREL. "Integrated Multi-Port PCI Gateway Solution." Capturado em: http://www.micrel.com/_PDF/Ethernet/datasheets/ks8695px_ds.pdf, Setembro 2005.

[MOY98] Moy, J. "OSPF Version 2." RFC 2328, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc2328, Abril 1998.

[POS81] Postel, J. "Internet Control Message Protocol." RFC 792, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc792, Setembro 1981.

[RNP98] RNP. "Internet Group Management Protocol - Versão 1 e 2 e Real Time Protocol (RTP)." Rede Nacional de Ensino e Pesquisa (RNP). Capturado em: http://www.rnp.br/newsgen/9801/mbone3.html, Janeiro 1998.

[ROC09] Rocha, A. M. A. "Algoritmos e Complexidade - Aulas Práticas." Universidade de Aveiro, Portugal. Capturado em: http://sweet.ua.pt/~f706/algoritmos/index.html, Abril 2009.

[SCH03] Schulzrinne, H.; Casner, S.; Frederick, R.; Jacobson, V. "RTP: A Transport Protocol for Real-Time Applications." RFC 3550, Network Working Group, The Internet Engineering Task Force (IETF). Capturado em: http://www.rfc-editor.org/rfc/rfc3550.txt, Julho 2003.

[SCH08] Schneider, K.; Kocak, T. "Design and implementation of an offload engine for internet group messaging protocol multicast snooping." In: IET Communications, 2008, pp. 484-492.

[SCH10] Schiffman. M. D. "libnet - "libpwrite" Network Routine Library" Linux man page. Capturado em: http://linux.die.net/man/3/libnet, Novembro 2010

[SPR03] SPRINT. "An Overview of Source-Specific Multicast (SSM)." RFC 3569, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc3569, Julho 2003.

[TAN08] Tanenbaum, Andrew S.; Woodhull, Albert S. "Sistemas Operacionais Projeto e Implementação." Porto Alegre:Bookman, 2008, 990p.

[TCP10] TCPDUMP. “TCPDUMP e LIBPCAP.” Capturado em: http://www.tcpdump.org/, Novembro 2010.

[THA04] Thaler, D.; Fenner, B.; Quinn, B. “Socket Interface Extensions for Multicast Source Filters.” RFC 3678, The Internet Engineering Task Force (IETF). Capturado em: http://tools.ietf.org/html/rfc3678, Janeiro 2004.

[VID10] VIDEOLAN. “VLC: open-source multimedia framework, player and server.” http://www.videolan.org/vlc, Setembro 2010.

[WEI08] Wei-Kuo, L.; Ping-Hai, H.; Shu-Kang, T. “An Ethernet Access Architecture for Highly Available IPTV.” In: Global Telecommunications Conference, 2008, pp.1-5.

[WIR10] WIRESHARK. “The World's Foremost Network Protocol Analyzer.” http://www.wireshark.org, Setembro 2010.

[WIT01] Wittmann, R.; Zitterbart, M. “Multicast Communication Protocols and Applications.” San Francisco:Morgan Kaufmann Publishers, 2001, 349p.

[XOR10] Xorp. “Extensible Open Router Platform.” http://www.xorp.org/, Setembro 2010.

[YAG03] Yaghmour.K, “Building Embedded Linux Systems”, Sebastopol:O’Reilly & Associates, Inc, 2003, 391p.

APÊNDICE A – RELATÓRIOS DAS SIMULAÇÕES

Foram realizadas duas simulações nos testes do simulador de hospedeiros e switch multicast. Uma com 10 grupos e 8 clientes e outra com 8 grupos e 16 clientes. Os arquivos de configuração dos clientes com as ações executadas por eles e os logs gerados nestes testes se encontram em:

- http://www.inf.pucrs.br/~ta.parks/wiki/lib/exe/fetch.php?media=simulacoes.zip.