Relatório T1 - Protocolo Heartbeat com Raw sockets

Bruno Lippert e Victor Putrich

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

1. Introdução

Este relatório tem como objetivo apresentar o processo de implementação de um protocolo, chamada Heartbeat que deve ser encapsulado pelo protocolo Ethernet. Usaremos *raw sockets* para estabelecer o canais de comunicação entre os hosts ativos. O objetivo do trabalho é compreender o funcionamento do protocolo Ethernet e comunicação de forma distribuída via sockets.

Na *Seção 2* explicaremos o funcionamento do protocolo, em seguida, a *Seção 3* trata de explicar o funcionamento do raw sockets e como usá-los. A *Seção 4* traz o detalhamento da implementação e a *Seção 5* apresenta um pequeno cenário de teste que usamos neste projeto.

A implementação do trabalho encontra-se no repositório do Git **[/schererl/HEARTBEAT-Protocol]**. O objetivo do relatório é explicar como foi feito o desenvolvimento do projeto e como foi pensada a elaboração das estruturas necessárias para o protocolo.

2. Protocolo Heartbeat

O protocolo Heartbeat tem como objetivo conectar hosts de uma mesma rede e possibilitar que troquem mensagens textuais entre sí, para isto, de tempos em tempos precisam informar para a rede que estão ativos de tempos em tempos. O protocolo é dividido em três mensagens:

- **START nome**: No momento que um novo host deseja se conectar a rede, é preciso enviar em broadcast a mensagen start junto com o nome de seu dispositivo. Esta mensagem serve para informar aos participantes da rede que existe um novo dispositivo que se conectou.
- **HEARTBEAT nome**: De 5 em 5 segundos, os hosts conectados devem enviar uma mensagem em broadcast HEARTBEAT com o nome do dispositivo. O objetivo dessa mensagem é avisa os membros conectados a rede que o host se mantém ativo. Outro uso do HERTBEAT é que ao receber um pacote START, envia-lo diretamente para a nova máquina conectada.
- **TALK nome dados**: Essa mensagem é enviada como forma de comunicação entre duas máquinas participantes. Para enviar uma mensagem o dispositivo deve colocar o nome da máquina destino e a mensagem que deseja transmitir. Detalhe: só deve ser possível enviar mensagens para hosts ativos na rede, por ativo, entende-se que o dispositivo deve mandou a última mensagem de HEARTBEAT com no máximo 15 segundos de intervalo.

Como já mencionado, utilizamos o Ethernet para encapsular o protocolo Heartbeat, de forma que seja possível transmitir pacotes entre nodos conectados por uma mesma rede. Com o objetivo de reproduzir o comportamento das três mensagens listadas acima, criamos uma *struct* para armazenar as informações necessárias para a transmissão do pacote, são elas.

- **type**: tipo *uint8* que receber valor 1 para mensagens START, 2 para HEARTBEAT e 3 para TALK.
- **hostname**: vetor de 16 caracteres, serve para armazenar o nome do host.
- **talk msg**: vetor de 32 caracteres, serve para armazenar as mensagens de TALK.

2.1. Protocolo Ethernet

A representação do pacote ethernet está dividida no cabeçalho e o frame, sendo o cabeçalho composto pelo endereço mac de origem e destino, ambos vetores de tamanho seis do tipo unsigned int de 8 bytes, também com uma variável de tipo, indicando qual pacote está sendo encapsulado. Para o nosso protocolo, adotamos o valor 0x0666 para representar o pacote Heartbeat. A *Struct* eth frame além de armazenar o cabeçalho Ethernet, armazena a estrutura do Heartbeat, note que aqui seria possível encapsular outros pacotes como IP ou UDP, sinalizados a partir do do atributo de *Type* no cabeçalho Ethernet.

3. Raw Socket

Usamos neste trabalho uma comunicação via raw socket que serve para estabelecer um meio de transmissão entre máquinas host. Raw Sockets não são tratados pelo sistema operacional, eles são processados diretamente na CPU, tornando possíveis que os pacotes possam ser visualizados e manipulados a nível de aplicação sem nenhum tipo de pre-processamento prévio, o caminho inverso também é válido.

Para usarmos a comunicação via raw socket, precisamos das seguintes bibliotecas:

- sys/socket.h
- sys/ioctl.h
- linux/if packet.h
- arpa/inet.h

Para abrirmos o socket atribuímos os seguintes parâmetros:

sockfd = socket(AF PACKET, SOCK RAW, htons(0x0666))

O AF PACKET permite uma aplicação receber o enviar pacotes raw, SOCK RAW é o valor que define o tipo de pacote enviar, por fim, o último parâmetro corresponde ao número de protocolo. A função retorna um inteiro que identifica o número do socket *sockfd* aberto, usamos ela na hora de fazer o envio e recebimento de pacotes. No caso do recebimento, devemos manter o socket aberto para evitar perda de pacotes recebidos.

É importante chamar atenção que para enviar e receber pacotes via socket, é preciso de permissão de administrador do sistema. Além disso, é devemos habilitar a interface de rede para modo promíscuo (tanto no envio, como no recebimento), para que os pacotes recebidos possam ser passados diretos para a CPU, sem que os headers sejam removidos.

A *Figura* 1 mostra o trecho de código que definimos o modo promíscuo. Resumidamente, acionamos o modo promíscuo fazendo uma chamada de sistema, alternado uma flag de dispositivo.

4. Implementação

Para nossa aplicação um host deve ser capaz de enviar e receber mensagens, além de poder escrever via terminal quando desejar enviar um pacote do tipo TALK. A estratégia

```c
/* Set interface to promiscuous mode */
strncpy(ifopts.ifr_name, ifName, IFNAMSIZ - 1);
ioctl(sockfd, SIOCGIFFLAGS, &ifopts);
ifopts.ifr_flags |= IFF_PROMISC;
ioctl(sockfd, SIOCSIFFLAGS, &ifopts);
```

Figura 1. Ativar modo promíscuo em C.

para que possamos fazer esses três ”serviços”simultaneamente, sem que um interrompa o outro é usar *threads*:

- **PRIMEIRA THREAD:** É responsável por abrir um socket e ficar aguardando recebimento e processamento de pacotes, isso inclui, a adição de novos hosts ou atualização dos hosts que ainda estão ativos conforme recebimento de pacotes HEARTBEAT e o envio direcionado de um HEARTBEAT para os pacotes que enviarem START.

- **SEGUNDA THREAD:** Envia pacotes HEARTBEAT em broadcast de 5 em 5 segundos. Esta thread envia o pacote e é colocada para dormir 5s em *loop*.

- **TERCEIRA THREAD:** Fica aguardando entradas no terminal, que podem ser tanto para listar membros ativos de rede com o comando *list*, ou enviar um pacote TALK, pelo comando ”TALK nome dado”.

.

Usamos a biblioteca ”pthread.h”para abrir as threads através do comando *pthread create* que recebe de parâmetro um ponteiro para uma variável de tipo *pthread t* e a função a ser executada pela thread. Importante usar a função pthread join para sincronização de fechamento das threads.

4.1. Estrutura de Controle de Hosts Ativos

Para que o envio e recebimento de pacotes ocorra, o protocolo deve se assegurar que os pares comunicantes estão ativos, para isso, criamos um vetor contendo uma estrutura de controle chamada *host info* que guarda o nome do host, seu endereço mac e uma variável do tipo ”time t”chamada *last beat*. A última é atualizada para cada vez que o dispositivo receber de seus vizinhos um pacote HEARTBEAT. Toda vez que um vizinho enviar um pacote START, este é adicionado no vetor de controle.

Adotamos um tamanho de vetor fixo para receber as máquinas hosts, este método simplifica a gerência deste vetor. Nós não fazemos nenhum tipo de remoção de fila em lista encadeada ou realocação de posições, apenas vamos adicionando novos hosts conforme se conectam. Também não mantemos nenhuma flag de controle de hosts ativos. O que fazemos é armazenar quantos hosts existe neste vetor e ao iterá-los, simplesmente olhar para o parâmetro *last beat* e conferir se a última batida + 15 segundos é maior ou igual ao tempo atual, se for, o host esta ativo e é considerado na hora de transmitir pacotes.

Apesar de não ser ideal em termos de uso de memória, partimos do princípio que para o nosso estudo de caso as redes não são tão grandes, em contrapartida, como não precisamos percorrer e atualizar toda nossa lista de tempos em tempos para alterar flags ou remover hosts, temos um ganho de desempenho e simplificação no fluxo do programa.

4.2. Envio de Pacotes

Os pacotes enviados pelo protocolo são descritos na *Seção 2*. Ao iniciar, o programa envia em broadcast uma mensagem do tipo START, e a cada 5 segundos em broadcast e quando recebe uma mensagem de start, uma mensagem do tipo HEARTBEAT. A mensagem do tipo TALK só é enviada a partir de um input do usuário, que deve forncer um destino válido (presente na lista de dispositivos conhecidos), e uma mensagem textual.

O envio em si dos pacotes é gerenciado por um método chamado sendRaw, que recebe o tipo da mensagem a ser enviada, os dados, caso hajam, e o endereço MAC destino. Ao receber esses parâmetros a função então inicializa um novo raw socket, define as propriedades da interface para o envio de mensagens e preenche os protocolos. Os protocolos são preenchidos em ordem: Ethernet e Heartbeat. No protocolo Ethernet são preenchidos as informações de destino e origem, assim como o ethernet type, que no caso do protocolo Heartbeat foi definido como 0x0666. Para o protocolo Heartbeat são preenchidos os campos de tipo de mensagem, dados e hostname, então o pacote é enviado via Raw Socket.

4.3. Recebimento de Pacotes

Como mencionado anteriormente o recebimento dos pacotes é gerenciado por uma thread. O método responsável pelo recebimento chama-se recvRaw, ele é responsável por monitorar continuamente o recebimento de qualquer pacote que possua o Ethernet type do protocolo Heartbeat. Ao receber um pacote, primeiramente é verificado se o endereço MAC de origem não é o mesmo da máquina atual, caso seja descartamos o pacote, pois não queremos processar dados enviados em broadcast pela própria máquina. Em seguida é feito o gerenciamento do tipo de mensagem recebida, caso seja um TALK os dados e o hostname da origem são exibido na tela para o usuário. Caso seja um HEARTBEAT, então é verificado se o host já está na tabela de hosts conhecidos, caso esteja, é atualizado seu tempo do último heartbeat, caso o contrário ele é adicionado a tabela. Por fim, temos a mensagem de START, onde o novo host é adicionado a tabela de hosts conhecidos e é enviado uma mensagem de HEARTBEAT direcionada para ele.

5. Cenário de teste

Para realizar os testes e validar a solução foi utilizado uma simulação no core emulator. A topologia dos testes pode ser observada na figura 2.

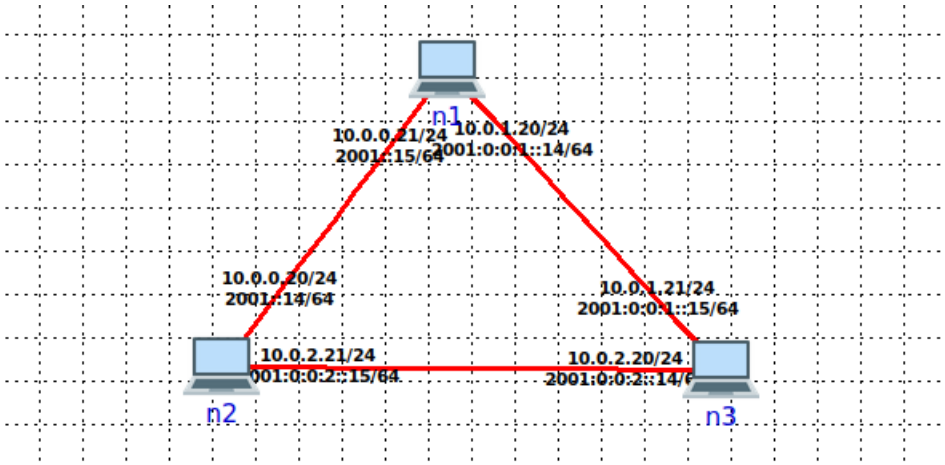

Figura 2. Topologia de testes.

Após iniciar o programa nos 3 nós da rede, listamos os hosts conhecidos de n1, como visto na figura 3.

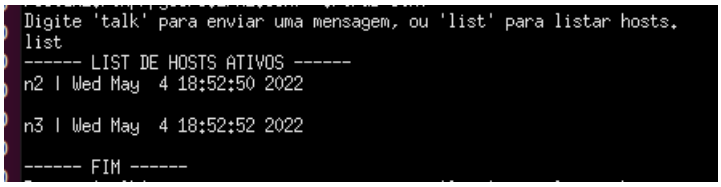

Figura 3. Nós conhecidos de n1.

Em seguida testamos o envio de mensagens de n1 para n2, como visto na Figura 4.

Figura 4. n1 envia mensagem para n2.

No terminal de n2 podemos observar a mensagem chegando, como visto na Figura 5.

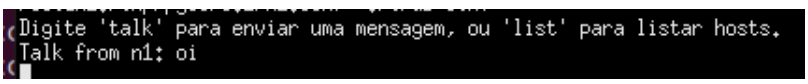

Figura 5. n2 recebe mensagem de n1.

Em seguida, fazemos a queda do nó n3, e listamos os nós conhecidos por n1, como visto na Figura 6.

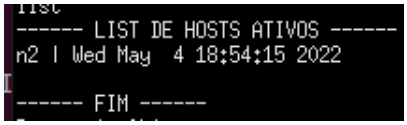

Figura 6. Nós conhecidos de n1.

6. Instrução de Uso

Para rodar a implementação do protocolo, é preciso rodar o executável com o nome da interface que deseja usar. Disponibilizamos também no repositório um arquivo *run.sh* que compila o projeto e roda diretamento, basta trocar o nome de interface.