Implementação do Protocolo FINLAN

Gabriel Rossi de Oliveira Malva, Emílio Carvalho Dias, Bruno Cezar de Oliveira, João Henrique de Souza Pereira, Pedro Frosi Rosa, Sérgio Takeo Kofuji

***Abstract*—The main protocols of the TCP/IP architecture were designed 30 years ago and because the large use of this architecture it is difficult to implement new ideas in its intermediate layers. This work in progress shows the implementation of the Fast Integration of Network Layers (FINLAN) protocol, that proposes ways and discussions for the possibilities of the new generation Internet. This protocol works at the network and transport layers of the Internet architecture and allows the distributed systems communication without the use of the IP, TCP and UDP. By the FINLAN design it can contribute for the post IP studies and technologies.**

***Index Terms*—Network Layers Optimization, Raw Sockets, Local Networks, Post IP.**

I. Introdução

Sistemas de telecomunicações que utilizam a arquitetura TCP/IP conseguem oferecer serviços com boa confiabilidade e segurança, porém há alguns problemas no uso desta arquitetura, como por exemplo redundância em alguns cabeçalhos que geram *overhead* desnecessário [1].

Também há campos que não são mais utilizados ou que foram remodelados para atender alguma necessidade não especificada no projeto original da arquitetura Internet, como é o caso do TOS, que foi modificado para DSCP [2], para atender a necessidade de transmissão com pedido de qualidade, criando assim improvisações nas especificações para atender as novas necessidades das redes, que os sistemas de telecomunicações trouxeram. Por exemplo, necessidade para comunicação VoIP (*Voice over IP*) [3]-[5].

Os estudos sobre o FINLAN (*Fast Integration of Network Layers*) tem a motivação de propor uma nova estrutura de rede com menor *overhead* a partir de um projeto para nova estrutura da arquitetura TCP/IP, permitindo a comunicação distribuída sem utilizar os protocolos IP, TCP e UDP a partir de uma nova definição para a camada de enlace, para que esta possa atender diretamente os serviços que suportem a camada de aplicação.

Este trabalho tem o objetivo de implementar o protocolo FINLAN e criar uma biblioteca para o sistema operacional Linux. Também é objetivo demonstrar a capacidade do protocolo implementado com a apresentação dos resultados dos testes realizados.

G. R. O. Malva e P. F. Rosa fazem parte do Departmento de Ciência da Computação, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil e-mail: gabrielrossi@comp.ufu.br, pedro@facom.ufu.br.

E. C. Dias e B. C. Oliveira fazem parte do Departamento de Ciência da Computação, Centro Universitário do Triângulo, Uberlândia, MG, Brasil e-mail: emilio@ctbc.com.br, brunoco@ctbc.com.br.

J. H. S. Pereira e S. T. Kofuji fazem parte do Departamento de Engenharia Elétrica, Universidade de São Paulo, São Paulo, SP, Brasil e-mail: joaohs@usp.br, kofuji@pad.lsi.usp.br

Este trabalho está organizado da seguinte maneira: Na segunda seção são apresentados os trabalhos relacionados. Na terceira seção é apresentada a proposta da solução para implementar o protocolo. Na quarta seção é apresentada a implementação e resultados dos testes com o FINLAN. E na quinta seção são apresentadas as conclusões desta implementação e sugestões para os trabalhos futuros.

II. Trabalhos Correlatos

Na camada de transporte o UDP, foi implementado em grande parte dos sistemas operacionais e este protocolo não possui objetivo de garantir a entrega dos pacotes, mas sim a de comunicação com baixo *overhead*, o que é útil em sistemas de comunicação em tempo real, por exemplo, para aplicações de vídeo, VoIP e redes de sensores.

Quanto ao endereçamento na Internet, com sua acelerada expansão houveram problemas relacionados à quantidade de endereços suportados para uso na rede mundial, assim foram desenvolvidas soluções como o NAT e criada uma nova especificação, o IPV6, com maior capacidade de endereçamento (de 32 bits no IPv4 para 128 bits no IPv6) e cabeçalho reduzido, também para diminuir o *overhead* e o processamento dos pacotes [6].

Também foram especificadas alternativas para as redes ethernet, como o Infiniband, que foi desenvolvido para redes de alta velocidade e que suporta serviços com baixa latência e alto *throughput* [7]. Outra alternativa para estas redes é o Myrinet, que possui menor *overhead* e maior *throughput* que as redes ethernet, e possibilita menor interferência na transmissão de dados e baixa latência [8].

A. O Protocolo FINLAN

O protocolo FINLAN implementa uma nova estrutura para atender as novas necessidades tecnológicas modificando o cabeçalho do protocolo ethernet com a adição de 6 campos, conforme apresentado na Fig. 1.

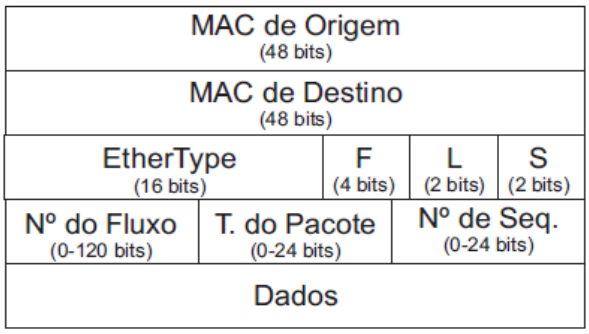

Fig. 1. Estrutura do cabeçalho FINLAN

A identificação do protocolo FINLAN é feita pelo valor 0x0880 no campo *ethertype* que é o campo responsável por

identificar o protocolo a nível de enlace na arquitetura TCP/IP. O valor 0x0880 encontra-se disponível para uso no IANA (*Internet Assigned Numbers Authority*).

O cabeçalho original do protocolo ethernet utiliza dois campos de identificação do MAC (*Medium Access Control*) para identificar as máquinas na rede e estes campos, assim como o *ethertype* são mantidos no FINLAN . Os campos com bits de identificação do FINLAN são “F”, “L” e “S”, respectivos ao “Número de Fluxo”, “Tamanho do pacote” e “Número de Seqüência”. A *flag* identificação “F” tem tamanho de 4 bits e as *flags* “L” e “S” possuem de 2 bits. Pelo uso destas *flags* observa-se que o protocolo suporta o endereçamento de uma grande quantidade de estações através de fluxos e também o envio de pacotes com grandes quantidades de dados, o que permite a expansão dos limites do MTU (*Maximum Transmission Unit*).

As *flags* de identificação permitem que os campos associados a elas, assumam valores de tamanho pequeno até muito elevado, adaptando-se assim à necessidade do sistema de comunicação e minimizando o desperdício em bytes. A *flag* “F” informa o tamanho do campo “Número do Fluxo”, que pode assumir valores com tamanho de 1 a 8 bytes, e as *flags* “L” e “S”, informam o tamanho dos campos “Tamanho do Pacote” e “Número de seqüência”, que podem assumir valores de 1 a 4 bytes.

Além de endereçar os pacotes em redes com conectividade em camada 2, o FINLAN permite endereçar aplicações distintas pelo uso do campo “Número do Fluxo”. Assim, cada aplicação define um fluxo para receber os dados na comunicação distribuída com outras aplicações.

O “Número de Seqüência” adiciona ao protocolo a função de que aplicações distintas utilizem o mesmo fluxo, permitindo assim o fluxo entre múltiplas aplicações na mesma estação ou em estações distintas. Desta forma é atendida a necessidade de localização das aplicações que o UDP e TCP fazem através do uso de portas.

III. Proposta de implementação

O uso do protocolo FINLAN é feito através de uma biblioteca fornecida ao usuário programador, de maneira que ele possa enviar e receber informações utilizando este protocolo de maneira transparente ao desenvolver aplicações.

Para criar esta biblioteca e permitir o endereçamento de estações sem o IP, como também localizar aplicações sem o TCP e UDP, foi utilizado um *socket* chamado *RAW* [9]. Com este foi eliminada a pilha padrão TCP/IP e escritos os dados diretamente na camada de enlace.

A idéia central desta primeira implementação do FINLAN é a eliminação das atuais camadas de Transporte e Rede, eliminando assim campos redundantes.

O ambiente utilizado na implementação desta biblioteca utiliza processador Intel padrão CISC com placa Fast Ethernet.

Em nível de software foi utilizado o Sistema Operacional Ubuntu, versão 9.04, com Kernel 2.6.28-14 e as compilações foram feitas com o compilador GCC 4.3.3, sendo que não foi utilizada nenhuma opção para “*tunning*”.

A. Estrutura da Proposta Desenvolvida

A estrutura da proposta neste trabalho, com a criação de uma biblioteca para tornar o uso do FINLAN transparente por desenvolvedores de aplicações de sistemas de comunicação é resumida em 4 elementos, conforme a Fig. 2, sendo eles a biblioteca criada, que fica entre as aplicações e o sistema operacional para então fazer uso da própria rede.

A aplicação é responsável pela comunicação com a biblioteca FINLAN e faz a interface com o usuário que deseja utilizar este protocolo.

A biblioteca possui as funções necessárias para utilização do FINLAN e esta comunica com o Sistema Operacional que, assim como em outros protocolos, é responsável pela comunicação com os elementos de rede utilizados.

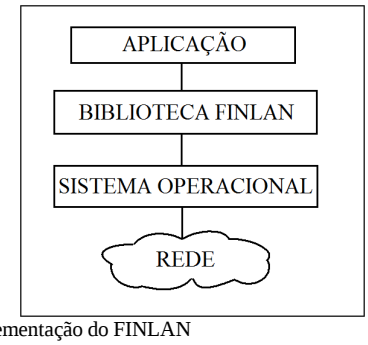

Fig. 2. Implementação do FINLAN

IV. Implementação do finlan

A Implementação do FINLAN teve início em seu projeto seguido da programação em RAW SOCKETS, que permite a construção de sockets sem as camadas padrão do TCP/IP. Desta forma foi possível construir a biblioteca que pode ser utilizada por aplicações através de *include*.

Após o desenvolvimento da biblioteca, para testes, foi criada uma aplicação de transferência de arquivos e feita a comparação de desempenho com os protocolos da arquitetura TCP/IP. Os resultados desta comparação são apresentados ao final desta seção.

A. Biblioteca FINLAN

A biblioteca denominada finlib.c é responsável pelo envio e recebimento das mensagens FINLAN. Entre as funções de envio criadas para esta biblioteca tem-se: create_socket_finlan, send_finlan, get_mac, create_header_eth, create_header_ finlan, add_data e socket_finlan. A função create_socket_ finlan é responsável pela criação do socket retornando-o para a aplicação enviar suas mensagens, deixando o gerenciamento da biblioteca (como a chamada das outras funções) e envio de dados, a cargo da função send_finlan.

A função create_socket_finlan assim como a função send_finlan são as funções necessárias para a utilização do protocolo e a aplicação as utilizará para enviar os dados. Assim os detalhes do protocolo ficam transparentes aos desenvolvedores de aplicações, que não precisarão conhecer os detalhes do FINLAN, nem controlar a construção dos campos deste protocolo.

A função send_finlan tem cinco parâmetros: 1) a interface de rede para envio dos dados; 2) o socket criado pela função create_socket_finlan; 3) o fluxo responsável pelo endereçamento da mensagem na aplicação da máquina de destino; 4) O MAC de destino; e, 5) A própria mensagem.

Inicialmente a função send_finlan chama a função create_header_eth que é responsável pelo cabeçalho ethernet, sendo que estes campos são compostos pelo campo *ethertype* definido com o valor 0x0880 (escolhido para o protocolo FINLAN por estar disponível no IANA) em hexadecimal, o MAC de destino passado pela chamada do usuário, e o MAC de origem obtido pela função implementada get_mac.

Ao finalizar o cabeçalho ethernet, inicia-se a função responsável por criar o cabeçalho FINLAN. É nesta função que possui a implementação necessária para adicionar os dados de fluxo, sequência, tamanho do pacote e as *flags* de acordo com a especificação deste protocolo.

Após a criação do cabeçalho os dados a serem transmitidos são adicionados pela função add_data e posteriormente enviados pela função socket_finlan onde está implementado o algoritmo de envio utilizando a RAW SOCKET.

Entre as funções de recebimento existem as seguintes: create_socket_finlan, received_finlan e get_mac. Sendo que similarmente às funções de envio, as funções received_finlan e create_socket_finlan são as funções necessárias para a utilização do protocolo que a aplicação externa utilizará para receber os dados.

A função received_finlan possui quatro parâmetros, sendo que o primeiro é o da interface de rede responsável por capturar os pacotes, o segundo, o socket criado pela aplicação pela função create_socket_finlan, o terceiro, o fluxo, que de acordo com a especificação do protocolo é passada pela aplicação externa indicando qual fluxo a mesma estará escutando para receber as mensagens e por último uma variável, que será utilizada pela aplicação externa obter a mensagem recebida.

A função received_finlan possui a lógica do recebimento destas mensagens, sendo que primeiro é verificado o *ethertype* da mensagem comparando-o com o valor do *ethertype* FINLAN e depois analisa o endereço MAC de destino da mensagem, comparando-o com o endereço MAC local obtido pela função get_mac. Com estas condições satisfeitas é realizada a cópia da mensagem para a variável responsável por armazenar a mensagem na função received_finlan.

B.Aplicação e Análise dos Resultados

No FINLAN o ganho de *overhead* é significativo comparado com o uso das camadas de rede e transporte da arquitetura TCP/IP. A análise seguinte mostra alguns detalhes destes ganhos.

Nesta análise foi utilizado um arquivo de 10GB (10.737.418.240 bytes)

FINLAN: 18 bytes de cabeçalho para o tamanho de pacote utilizado.
10.737.418.240 + (18 * 7.245.222) = 7.245.222 pacotes
Total = 10.867.832.236 bytes

TCP: 54 bytes de cabeçalho (14 Eth, 20 IP e 20 TCP).
10.737.418.240 + (54 * 7.425.601) = 7.425.601 pacotes
Total = 11.138.400.694 bytes

Pelos detalhes acima verifica-se que o ganho total, para transferência de um arquivo de 10GB foi de 258MB (270.568.458 bytes), com diferença de 180.379 pacotes enviados. O gráfico de desempenho nesta comparação é apresentado na Fig. 3.

Para obter resultados mais precisos com a implementação do FINLAN foi criada uma aplicação de transferência de arquivos para enviar e receber os dados transmitidos. Esta aplicação foi desenvolvida utilizando a biblioteca criada neste trabalho.

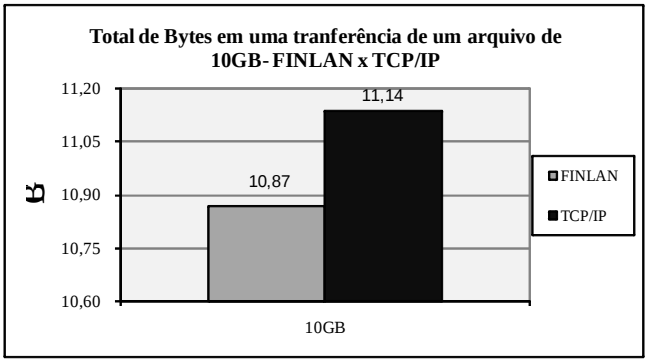

Fig. 3. Total de Bytes em uma transferência de um arquivo de 10GB utilizando FINLAN E TCP/IP

No envio de dados a aplicação utiliza as funções de tratamento de arquivos da biblioteca stdio.h para obter os dados pertencentes ao arquivo que deseja-se enviar. Nela há um controle responsável pelo tamanho de cada mensagem enviada.

Após a formação dos blocos de mensagem é utilizada a biblioteca implementada para a criação do socket e envio das mensagens utilizando a função send_finlan passando os atributos necessários.

Para receber as mensagens, a aplicação através da função received_finlan obtém as mensagens enviadas pela origem e posteriormente assim como no envio utiliza as bibliotecas de tratamento de arquivos para remontar o arquivo transferido.

Para um comparativo foi criada esta mesma aplicação utilizando o protocolo UDP, sendo que para este comparativo tenha resultados precisos em termos de comparação, ambos os testes foram executados em ambientes exatamente iguais.

A Fig. 4 mostra o gráfico comparativo da porcentagem de pacotes transmitidos chegado ao destino dos protocolos utilizados, sendo que para ambos os casos nas transferências ocorridas até 10Mbytes não houve perda de pacotes, e nos casos subseqüentes o FINLAN sobressaiu sobre o UDP, o que garante sua boa performance em sistemas de telecomunicações em redes de dados, como por exemplo, para ser utilizado em redes convergentes como as NGN (*Next Generation Networks*).

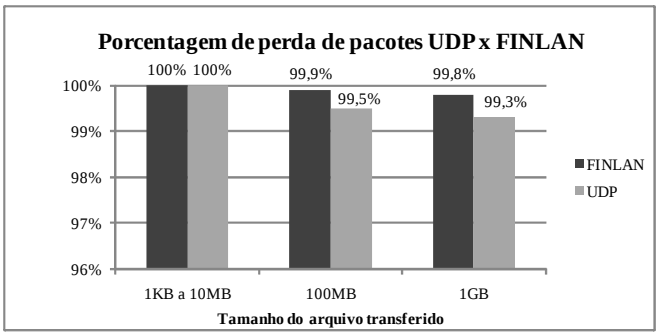

Fig. 4. Comparativo de pacotes transferidos FINLAN x UDP

V. Conclusão e Trabalhos Futuros

Neste trabalho foi apresentada a implementação do protocolo FINLAN e feita a comparação do desempenho desta implementação com os protocolos da arquitetura TCP/IP. Esta comparação comprova a boa eficiência deste protocolo explicada pela redução de *overhead* sobre os protocolos IP, TCP e UDP.

No estágio atual de especificação do FINLAN não há garantia de entrega, assim, para seu uso por aplicações que exigam maior confiabilidade, é necessário que hajam meios de transmissão confiáveis, como fibra ótica. Porém sua especificação e a biblioteca implementada neste trabalho já garantem o uso com eficiência em sistemas que utilizam atualmente o UDP como protocolo de transporte, por exemplo, aplicações de tempo real em redes de sensores e os sistemas de comunicação VoIP utilizados nas atuais NGN.

Para trabalhos futuros é sugerido a implementação de garantia de entrega e o de roteamento em redes mundiais, como também mecanismos para segurança e QoS. É sugerido também a implementação da biblioteca desenvolvida em um nível mais baixo do sistema operacional.

Referências

[1] J. Kay and J. Pasquale. Profiling and reducing processing overheads in tcp/ip. IEEE/ACM Transactions on Networking, 4:817–828, 1996.
[2] A. S. Tanembaum. Computer Networks. 2002.
[3] D. Comer. Internetworking with TCP/IP Volume I – Principles, Protocols and Architecture , 1995.
[4] D. Comer, D. L. Stevens. Internetworking with TCP/IP Volume II – Design, Implementation and Internals ,1999.
[5] D. Comer, D. L. Stevens. Internetworking with TCP/IP Volume III – Client-Server Programming and Applications,1997
[6] RFC2460 - Internet Protocol, Version 6 (IPv6) Specification. 1998.
[7] R. E. Grant, M. J. Rashti, and A. Afsahi. An analysis of qos provisioning for sockets direct protocol vs. ipoib over modern infiniband networks. Proceedings of the 2008 inter-national Conference on Parallel Processing, 2008.
[8] A. Barak, I. Gilderman, and I. Metrik. Performance of the communication layers of tcp/ip with the myrinet gigabit lan. Comput Commun, 22:989–997, 1999.
[9] M. M. Alves. Sockets Linux. 2008.