As Arquiteturas Cliente/Servidor na Nova Sociedade da Informação, usando IPv4 e IPv6 em uma rede de protocolos heterogêneos

João Cândido Dovicchi
Núcleo Avançado de Computação Sônica e Multimídia - NACSM
Universidade Federal de Uberlândia
Campus Sta. Mônica - Bloco 1V
Uberlândia - Minas Gerais - Brazil
email: dovicchi@demac.ufu.br

RESUMO:
Este artigo discute a implementação do *Internet Protocol version 6* (IPv6) abordando um ambiente de transição e de "convivência" dos dois tipos de protocolo (IPv4 e IPv6). A compreensão de como estes dois protocolos deverão ser mantidos — ainda por um grande período de tempo — é de fundamental importância para a transparência dos aplicativos nesta rede mista da sociedade da informação. Este trabalho evidencia, também, como os conceitos de arquitetura cliente/servidor terão que ser revistos e, ainda, são apresentadas algumas soluções possíveis para que isto possa acontecer.

KEY WORDS
Redes, Tecnologia da Informação, protocolo IP

1 Introdução

A implementação das redes de alta velocidade pela Rede Nacional de Pesquisa (RNP), hoje denominada RNP2, deverá estar disponível ainda este ano [11]. Diante desta perspectiva, a mudança não é apenas de velocidade mas, também, de um novo conceito de *backbone*. Esta nova rede deverá possibilitar o tráfego de um novo protocolo IP, o IP versão 6 ou, simplesmente, IPv6. A migração de um tipo de protocolo internet (o IPv4) para um novo (IPv6) acarretará algumas diferenças na interconexão entre clientes e servidores, no que diz respeito ao roteamento de pacotes na rede.

Evidentemente, esta mudança está longe de acontecer da noite para o dia. Nas redes de computadores da Sociedade da Informação (SI) ainda deverão trafegar pacotes dos dois tipos de protocolos até que um dia todos terão migrado para a nova versão de IP — se é que isto acontecerá. Diferentemente do início da Internet, quando as redes BITNET e FIDONET possuíam nós roteadores (*gateways*) para a Internet, a nova rede (Internet-2) estará interligada oferecendo os serviços já implementados, principalmente de Web, cujo número de servidores chega à casa dos trilhões e demandará mais de 75 Tbps de capacidade de comunicação [5]. Neste caso, o roteamento de pacotes IPv4 e IPv6 estará coexistindo em uma mesma rede. Durante esta fase de transição, é de suma importância que aplicações IPv4 continuem a funcionar com as novas aplicações IPv6. Na nova Sociedade da Informação, por exemplo, servidores Web não poderão aceitar apenas conexões de clientes IPv6 mas deverão, também, funcionar para clientes IPv4. Em um panorama ideal, teremos que considerar clientes com funcionalidade para IPv4 e IPv6, bem como servidores que operem com os dois tipos de protocolo IP. (veja figura 1)

Entretanto, rotear estes diferentes tipos de pacotes IP requer uma delicada afinação de configurações, uma vez que, conceitualmente, os protocolos possuem diferenças fundamentais. No protocolo IPv4, o tráfego é baseado em *broadcast*, enquanto no IPv6 a ênfase é dada em *multicast* e *unicast* e o roteamento segue regras bastante específicas descritas no documento RFC 2460 [4] e outros[1].

Uma das principais vantagens da nova estrutura é a possibilidade de controle de banda para aplicativos que utilizam multimímida, como é o caso da Educação a Distância que usa videoconferências e a distribuição em fluxo contínuo e tempo real de áudio e vídeo. Entretanto, na nova arquitetura de redes da SI, certamente não haverá a substituição imediata de todos os roteadores e, então, pacotes de aplicações *multicast* terão que ser encapsulados para trafegarem em uma rede *broadcast*. Logo, já se pode prever que, sem o controle de banda, haverá congestionamento da rede, não importando a quantos bits/segundo (bps) o tráfego seja estabelecido.

Embora os protocolos TCP e UDP sejam usados como camadas de transporte nas conexões da Internet, eles, por sua vez, utilizam a camada de rede fornecida pelo protocolo IP (*raw sockets*). A interoperabilidade entre estes protocolos (TCP/UDP e IP) dependem da programação adequada das chamadas e atendimento dos processos entre o cliente e o servidor. O IPv4, usado desde o início da Internet (ARPANET), tem uma capacidade de endereçamento de 32 bits e provê os serviços de pacotes TCP, UDP, ICMP e IGMP. O IPv6, por sua vez, foi criado em 1990 para substituir o IPv4 e tem a sua capacidade de endereçamento ampliada para 128 bits, o que poderá atender às demandas geradas pelo crescimento da Internet. Os serviços providos

[1] Os principais documentos (RFCs) sobre IPv6 podem ser encontrados em http://www.ipv6.org ou em http://www.ietf.org.

pelo IPv6 são TCP, UDP e ICMPv6.

Neste trabalho, consideraremos dois tipos de cenários:

1. Cliente IPv4 → servidor IPv6; e
2. Cliente IPv6 → servidor IPv4.

Por motivos óbvios, não estaremos considerando quando cliente e servidor suportam os mesmos protocolos. No caso dos sistemas IPv6 temos que considerar que suportam ambos os protocolos, caso contrário não existe possibilidade de conexão.

2 Os cenários da nova Internet

Quando consideramos uma rede mista de IP, temos que levar em conta que aplicativos TCP e UDP deverão operar usando chamadas de processos aos *raw sockets* para que se possam estabelecer as conexões. Enquanto os aplicativos IPv4 usam a constante `AF_INET` junto com a estrutura *sockaddr_in*, os aplicativos IPv6 usam `AF_INET6` e a estrutura *sockaddr_in6* que são totalmente diferentes.

A descrição do *socket* de endereçamento para IPv4, é feita da seguinte maneira:

```
struct sockaddr_in {
 sa_family_t sin_family; /* famlia  */
 unsigned short int sin_port; /* porta    */
 struct in_addr sin_addr; /* endereo */
 unsigned char __pad[__SOCK_SIZE__ -\
  sizeof(short int) -\
  sizeof(unsigned short int) -\
  sizeof(struct in_addr)];
};
```

enquanto para IPv6 é:

```
struct sockaddr_in6 {
 unsigned short int sin6_family; /* AF_INET6 */
 __u16 sin6_port; /* porta    */
 __u32 sin6_flowinfo; /* fluxo    */
 struct in6_addr sin6_addr; /* end.IPv6 */
 __u32 sin6_scope_id; /* RFC2553  */
};
```

A principal diferença está na estrutura *in_addr* do IPv4 e *in6_addr* do IPv6 que comportam endereçamento de 32 e 128 bits, respectivamente. Assim, um panorama genérico de aplicativos que usam os dois tipos de protocolos têm que abranger a capacidade de endereçamento de 32 bits (IPv4) e 128 bits (IPv6), por meio dos protocolos TCP, UDP, ICMP e ICMPv6. (veja figura 1)

Uma vez que as conexões em uma rede ethernet dependem da identificação dos endereços *Medium Access Control (MAC address)* vinculados ao IP e nome das extremidades e que deverá ser obtido pelas funções *gethostbyaddr* e *gethostbyname*, o primeiro problema é a configuração dos servidores de nomes que deverão fornecer os registros corretos “IN A” e “IN AAAA”, vinculados aos *sockets* específicos.

Embora aparentemente simples, pois o *Berckeley Internet Name Domain* (BIND) já apresenta suporte para IPv6 [3], o cerne do problema está na diferença entre o

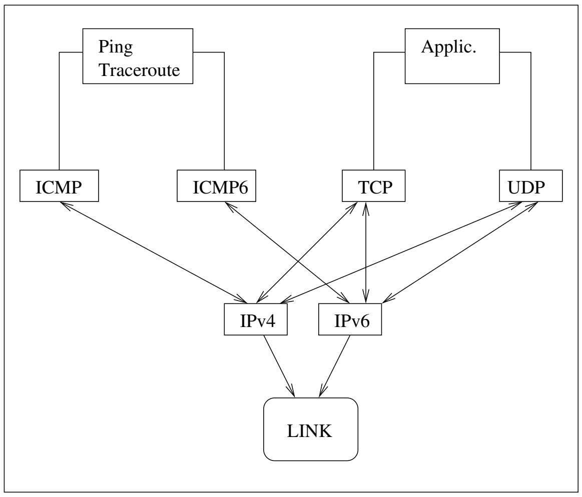

Figure 1. Visão geral de uma rede mista IPv4 e IPv6

endereço MAC nos protocolos IPv4 e IPv6. Uma conexão IPv4 endereça 48 bits de MAC, enquanto o IPv6 endereça 64 bits. Uma vez que as placas de rede atuais suportam 48 bits para seu endereço MAC, existe a necessidade de encapsular valores 48 bits no espaço de 64 bits do cabeçalho do IPv6. Tal procedimento (EUI-64) já se encontra descrito em documento específico [7].

A solução deste problema é fundamental para que cliente e servidor identifiquem seus protocolos. Assim, para que os serviços IPv4 e IPv6 possam funcionar de maneira eficiente e eficaz, é necessário que a programação dos *sockets* de rede seja feita de maneira correta e otimizada.

O primeiro tipo de conexão a ser considerado é de um cliente IPv4 que se conecta a um servidor *dual stack*. Em um cenário deste tipo, um servidor *dual stack* suporta chamadas para conexões IPv4 e IPv6, que deve ser feito mapeando-se endereços IPv4 em pacotes IPv6. (veja figura 2)

No caso do servidor receber uma chamada de um cliente IPv6, a comunicação entre eles se dá normalmente. Entretanto, quando o servidor recebe uma chamada de um cliente IPv4 acontece o seguinte:

1. O cliente IPv4 envia o *gethostbyname* e obtem o registro “IN A” para o servidor. (O servidor tem que ter informações de registro “IN A” para IPv4 e “IN AAAA” para IPv6.)
2. O cliente envia *connect* e o sistema envia um pacote IPv4 `SYN` ao servidor.
3. O servidor recebe o pacote IPv4 `SYN` no *socket* que está ouvindo e liga um apontador que indica que este pacote IPv4 deve ser mapeado para IPv6.
4. O servidor responde ao sistema do cliente com um IPv4 `SYN/ACK`.

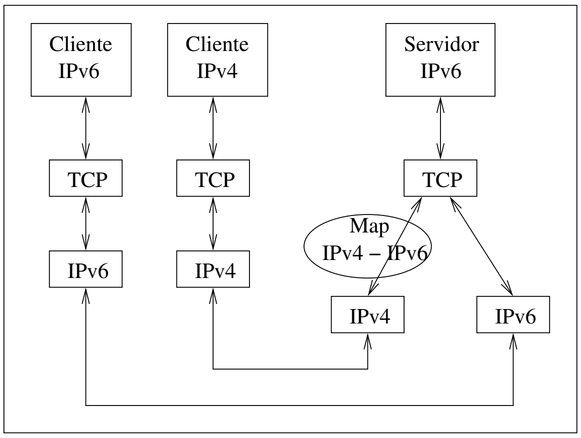

Figure 2. Servidor IPv6, em *dual stack*, atendendo serviços IPv4 e IPv6.

5. Toda a conexão estabelecida é conduzida por meio de IPv4 mapeado em IPv6 e dá-se por meio de datagramas IPv4.

A não ser que o servidor esteja programado para verificar se a conexão é IPv4 ou IPv6, usando uma macro `IN6_IS_ADDR_V4MAPPED`, o servidor não tem como identificar que está se comunicando com um cliente IPv4. Da mesma forma, o cliente jamais identificará que está conectado a um servidor IPv6.

No segundo caso, ou seja, o cliente IPv6 e o servidor IPv4, o cliente já deve identificar que o servidor responde ao *gethostbyname* com o registro "IN A" do DNS, e já dispara a chamada `SYN` no formato IPv4. O mapeamento de IPv4 em IPv6 ocorre dentro do sistema do cliente e a conexão ocorre, normalmente, por meio de datagramas IPv4.

O mapeamento de endereços IPv4 para IPv6 permite que aplicativos em em clientes duais — com suporte a IPv4 e IPv6 — estabeleçam a comunicação com servidores com suporte apenas para IPv4. Estes endereços, criados e divulgados pelos *Domain Name Services* (DNS), são resolvidos com a solicitação feita por um aplicativo IPv6 para um serviço IPv6 em um servidor, mas o servidor tem apenas endereço IPv4.

Assim, quando um sistema com suporte dual, solicita um serviço a um sistema IPv4 ele terá que mapear o endereço IPv4 em um endereço IPv6. O formato de endereço pode ser visto no esquema:

0000 $\cdots$ 0000	FFFF	End. IPv4
80 bits	16 bits	32 bits

Por exemplo, endereço IPv4 `200.131.191.101` mapeado para IPv6 é convertido para:

`0:0:0:0:0:FFFF:200.131.191.101`

ou, de forma mais abreviada, para:

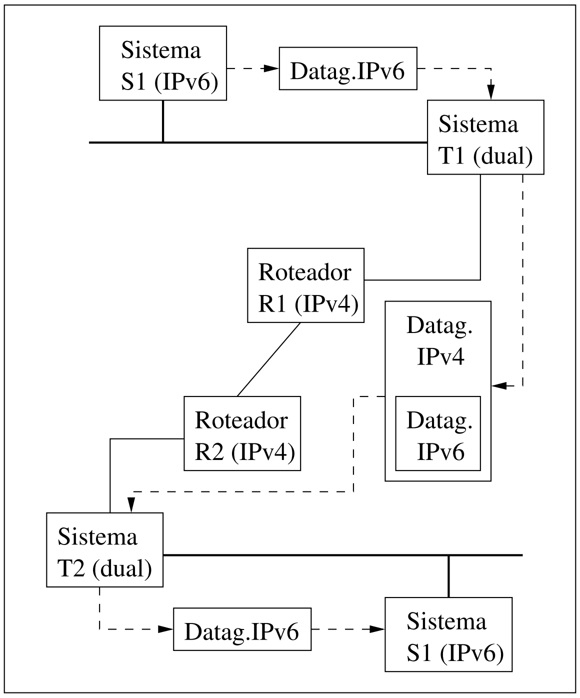

Figure 3. Encapsulamento de IPv6 em datagramas IPv4 para tunelamento em uma rede mista com roteamento IPv4.

`::FFFF:200.131.191.101`.

No caso do servidor *dual stack* em uma rede IPv4, ou seja, cujo roteador não tenha suporte para IPv6, este deve possuir um registro "IN AAAA" de DNS, contendo o endereço IPv6 compatível com IPv4. Todo sistema IPv6 a ser conectado a um servidor com um endereço IPv4 compatível com IPv6, deverá enviar os datagramas IPv6 encapsulados com cabeçalho IPv4, o que é denominado "tunelamento". O formado do IPv4 compatível com IPv6 é representado pelo diagrama:

0000 $\cdots$ 0000	0000	End. IPv4
80 bits	16 bits	32 bits

Por exemplo, o IPv4 `200.131.191.101` compatível com IPv6 é descrito como:

`0:0:0:0:0:0:200.131.191.101`,

ou:

`::200.131.191.101`

de forma abreviada.

Um cenário que ainda será bastante comum durante a migração dos serviços da Internet para a Internet2 será a comunicação entre duas redes IPv6, interligadas por uma rede IPv4. Neste caso, um sistema IPv6 que deve enviar um datagrama IPv6 para outro sistema em outra rede separada por um segmento IPv4, terá que ter um sistema que aja como roteador IPv6 e cria um tunel com outro sistema na rede de destino. (veja figura 3)

Na figura 3, o sistema S1 (IPv6) envia um datagrama

IPv6 para o sistema T1 (IPv4/IPv6), que encapsula o datagrama IPv6 em um datagrama IPv4 e é enviado para T2 (IPv4/IPv6) através de uma rede IPv4 (R1 e R2). O sistema T2 remove, então o cabeçalho IPv4 e envia o datagrama IPv6 para o sistema de destino D1 (IPv6).

Embora o panorama da nova rede possa parecer facilmente contornável, temos que considerar os seguintes problemas:

1. Para que os cenários expostos funcionem, os roteadores da rede terão que suportar e rotear pacotes IPv4 e IPv6.

2. Os pacotes *multicast* vindos de servidores IPv6 terão que estar encapsulados em pacotes IPv4 para trafegar numa rede IPv4, e os roteadores terão que estar programados para suportá-los.

3. A utilização de tunelamento IPv6 sobre IPv4 acarretará uma queda de performance, causando congestionamento do tráfego de pacotes na rede.

No primeiro caso, se todos os roteadores suportarem o tráfego de pacotes de um ou outro protocolo, não existirá problemas em nível de rede, embora tenhamos o problema da correta e eficiente programação dos sistemas e *software*. Nos outros dois casos, o problema é bastante sério, pois depende de como a rede suportará o tráfego e como pode ser feito para que haja um controle da banda utilizada pelos aplicativos, no caso de roteamento de pacotes *multicast*. No caso do tunelamento, tem-se que considerar não apenas o controle de fluxo como, também, a eficiência do tunelamento que, diferentemente de *multicast* que é feito entre servidor e subrede, acontece *peer-to-peer*. (Veja tabela 2)

Cliente	**Rota**	**Servidor**	**Mapa** IPv4/IPv6	**Tunel**
IPv4	IPv4	IPv4	não	não
IPv4	IPv4 IPv6	IPv6 dual	sim	não
IPv6 dual	IPv4 IPv6	IPv4	sim	não
IPv6	IPv4 IPv6	IPv6	não	não
IPv6	IPv4	IPv6	sim	sim

Table 1. Mapeamento e tunelamento de IP em redes mistas

3 Prováveis soluções

Apesar de ainda não definida, a área denominada *flow label* do IPv6 pode ser uma boa opção para solucionar o problema do controle de tráfego em uma rede mista. Mas, antes, devemos considerar alguns aspectos da conexão IPv6 sobre uma rede IPv4 e da conexão entre um servidor IPv6 e um cliente IPv4. Assim teremos as seguintes possibilidades:

1. Para uma conexão de dois sistemas IPv6 que passa, obrigatoriamente, por uma rede com suporte apenas para IPv4, a melhor opção é o tunelamento. Para isso é necessário estabelecer um túnel entre dois pontos.

2. Em uma conexão entre um cliente IPv4 e um servidor IPv6, em uma rede que suporte este último protocolo, a melhor opção é usar a macro `IN6_IS_ADDR_V4MAPPED` no lado do servidor.

Se quisermos ter o controle sobre a banda de *upload* teremos que fazê-lo no próprio pacote IPv6, no primeiro caso, ou antes do mapeamento IPv4, no segundo caso. Isto mostra, pelo menos, que o controle de tráfego na rede poderá ser ditado pelo servidor e controlado pelo cliente, ou seja, nas duas pontas do túnel — diretamente no pacote IPv6 — para o tunelamento ou controlado apenas pelo servidor, antes de mapear IPv6 para IPv4, no outro caso.

Determinados os pontos possíveis de controle, resta saber como fazê-lo. Em ambos os casos teremos que considerar dois modelos propostos pela *Internet Engineering Task Force* (IETF) de *Quality of Service* (QoS) para a Internet: *Integrated Service* ou IntServ [2] e o *Differenciated Service* ou DiffServ [9, 1].

Muitos dos aplicativos para EAD tem sido desenvolvidos com necessidades de tráfego em tempo real pela Internet. O TCP, atualmente utilizado, não é o protocolo ideal para estas aplicações. A melhor solução seria o uso do protocolo *Real Time Transport Protocol* (RTP) que, usualmente, é implementado sobre UDP e mais adequado para aplicações em tempo real. Entretanto, a combinação RTP/UDP não garante a QoS das sessões entre cliente e servidor.

O protocolo denominado *Resource ReSerVation Protocol* (RSVP) pode ser usado através da rede para implementar serviços demandados por aplicações em tempo real. O RSVP permite o controle de fluxo, garantindo uma reserva de recursos para estes aplicativos antes do início da transmissão de dados, mantendo a QoS para o fluxo de dados. Quando o aplicativo invoca o RSVP solicitando uma conexão *peer-to-peer*, o RSVP seleciona a rota e os protocolos de roteamento e, então, reserva recursos solicitados através da conexão. Assim, a utilização deste protocolo depende do controle em nível de rede do fluxo contínuo entre cliente e servidor.

No caso de uma rede com suporte para IntServ, o processamento por fluxo em roteadores com capacidade para IntServ acarreta o problema de excesso de fluxo. No caso de um tunelamento entre duas redes de alta velocidade, a única possibilidade está no controle do fluxo via DiffServ, permitindo a operação de aplicações *peer-to-peer* com controle de fluxo sem perda para a QoS.

Este controle pode ser feito por meio do conjunto de bits do *flow label* que poderá definir o fluxo entre antes do tunelamento (cf. fig 4), limitando a banda de tráfego e

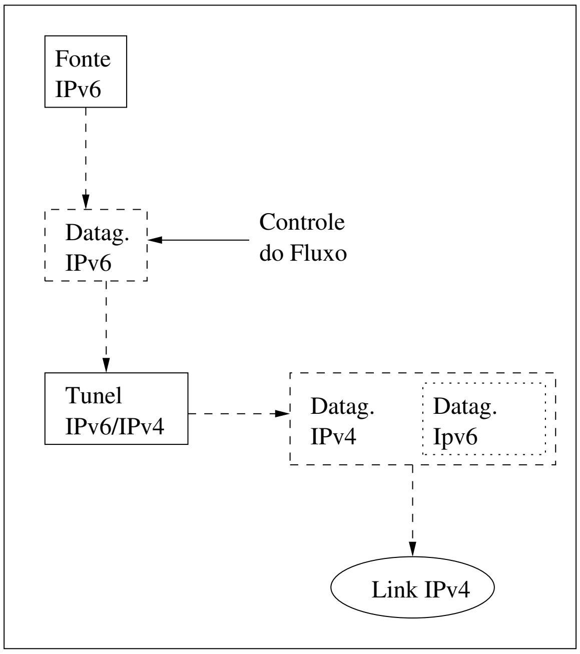

Figure 4. Inserção do controle de fluxo usando o campo *flow information* antes do encapsulamento do tunel.

garantindo um mínimo necessário exigido pelo aplicativo. Evidentemente, a largura da banda será um fator a ser considerado e fortemente limitante, uma vez que os roteadores envolvidos não terão que tomar parte neste controle.

4 Discussão

Embora ainda não existam APIs que utilizem os 28 bits da informação de fluxo (*flow information*) — definidos no cabeçalho dos pacotes IPv6 — para o controle efetivo do fluxo de dados em uma rede IPv6 ou no tunelamento de IPv6 sobre IPv4, existe a possibilidade de se utilizar as informações deste conjunto de bits para que as APIs possam definir o controle de fluxo. A RFC 2460 [4] divide os 28 bits em 8 bits para a classe de tráfego — ou Campo DS — e outros 20 bits de *flow label*. Entretanto, já existe uma tentativa de utilização deste campo do cabeçalho para controlar o tráfego da rede [8].

Entre os problemas a serem resolvidos pelas APIs que utilizam IPv6, o que nos interessa mais é a possibilidade do usuário (ou programa) poder estabelecer os bits de classe de tráfego. Kuznetsov [8] também discute o interesse em identificar, estabelecer e controlar o conjunto de bits do campo *flow label* de pacotes IPv6. Esta possibilidade permitiria o controlede fluxo tanto durante o encapsulamento de IPv6 em pacotes IPv4 quanto durante o mapeamento do pacote IPv4 para IPv6 em uma arquitetura cliente/servidor. Com isto pode-se estabelecer uma reserva de banda para que aplicativos RSVP possam ser controlados em uma rede mista.

Evidentemente, a solução leva a um aumento de ciclos de CPU e gasto de memória mas, em se tratando da necessidade de controle de banda por aplicativos de tempo real, isto será inevitável. Esta solução pode ser considerada "temporária", mas terá que ser padronizada no futuro, caso a migração total para o novo protocolo seja adiada. O controle de fluxo pode ser feito de duas maneiras:

1. A alocação dos bits pode ser feita diretamente pelo usuário (ou programa); e

2. O Kernel pode redirecionar a requisição do controle do *flow label* por meio de um *daemon* do sistema e não o preenche até que tenha a autorização do *daemon*.

O primeiro caso apresenta problemas de segurança, embora o sistema possa bloquear usuários sem privilégio, deixando para o *socket* gerenciar a requisição por meio de controle do SCM_RIGHTS. O segundo caso, pode ser comparado aos tunelamentos VPN (*Virtual Prived Network*) que utiliza o IPsec, podendo, a API, ser implementada com IPsec [8].

Ainda pode-se implementar o *flow label* na área do Kernel, diretamente. O problema, entretanto, é que, neste caso, o sistema ficaria vulnerável aos ataques de DoS (*Deny of Service*). O Kernel teria que manter os marcadores obsoletos até que os mesmos expirassem e um ataque malicioso poderia lotar o espaço do *flow label*, facilmente.

5 Agradecimentos

Este trabalho não teria sido possível sem a colaboração de bolsistas do PIBIC / CNPq e de estagiários do Laboratório de Multimídia do NACSM / UFU.

References

[1] BLAKE, S.; BLACK, D.; CARLSON, M.; DAVIS, E.; WANG, Z. e WEISS, W. "An Architecture for Differentiated Services, RFC 2475, Dec. 1998.
`http://www.ietf.org/rfc/rfc2475.txt`

[2] Braden, R.; CLARK, D. e SHENKER, S. "Integrated Services in the Internet Architecture: an Overview, RFC 1633, Jun. 1994.
`http://www.ietf.org/rfc/rfc1633.txt`

[3] CRAWFORD, M. e HUITEMA, C. "DNS Extensions to Support IPv6 Address Aggregation and Renumbering", RFC 2874, jul. 2000.
`http://www.ipv6.org/specs.html`

[4] DEERING, S. e HINDEN, R. "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, Dec. 1998.
`http://playground.sun.com/pub/ipng/html/specs/standards.html`

[5] DIXOT, S. & YE, Y. 2001."Streamlining the Internet-Fiber Connection", *IEEE Spectrum*, 38 (4), (April), pp.52-57.

[6] GILLIGAN,R; THOMSON, S.; BOUND, J. e STEVENS, W. "Basic Socket Interface Extensions for IPv6", RFC 2133, Apr. 1977. `http://www.ipv6.org/specs.html`

[7] IEEE-1997b "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority". Institute of Electrical and Electronics Engineers, 1997. `http://www.standards.ieee.org/db/oui/tutorials/EUI64.html`

[8] KUZNETSOV, A. N. "IPv6 Flow Labels in Linux-2.2", *Tech. Rep.*, Institute for Nuclear Research, Moscow, apr. 1999. `ftp://ftp.inr.ac.ru/ip-routing/iproute2-current.tar.gz`

[9] NICHOLS, K.; BLAKE, S.; BAKER, F. e BLACK, D. "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC 2474, Dec. 1998. `http://www.ietf.org/rfc/rfc2474.txt`

[10] PARTRIDGE, C., "Using the Flow Label Field in IPv6", RFC 1809, jun. 1995. `http://www.ipv6.org/specs.html`

[11] RNP "RNP2 - Infra-estrutura Internet2 para o Brasil, Rede Nacional de Pesquisa, 2002. `http://www.rnp.br/rnp2`

[12] STEVENS, W. e THOMAS, M. "Advanced Sockets API for IPv6", RFC 2292, Feb. 1998. `http://www.ipv6.org/specs.html`