NAT, ICMP e IPv6

Diego Passos

1 NAT

Na aula anterior, no contexto dos protocolos IP e DHCP, discutimos como interfaces de rede conectadas à Internet obtêm seus endereços IP. Em última análise, estes endereços são atribuídos pela IANA.

Ocorre, no entanto, que já há vários anos a IANA não possui mais faixas de endereços *não reservados* para alocar para os Registros Regionais de Internet. Na verdade, quatro dos cinco RIRs do mundo também já não possuem mais faixas disponíveis para novas alocações.

Este fenômeno de escassez de endereços IPv4 no mundo não é recente: há mais de duas décadas já se projetava que isso ocorreria. Quando houve a migração do endereçamento baseado em classes para o endereçamento CIDR, o motivo para a projetada escassez de endereços era simplesmente a forma ineficiente de alocação de sub-redes: faixas excessivamente grandes eram alocadas, gerando desperdício. Hoje, no entanto, a razão é bem mais simples: há mais dispositivos que endereços IPv4. A Cisco estima que, em 2017, aproximadamente 8 bilhões de dispositivos computacionais se conectaram à Internet. Este número, 8 bilhões, é claramente maior que o número máximo de endereços IPv4 distintos que se pode atribuir — de fato, é quase o dobro.

Este enorme número de dispositivos conectados à Internet pode parecer, a princípio, um dado inconsistente. Se há apenas cerca de 4 bilhões de possíveis endereços IPv4, como pode haver 8 bilhões de dispositivos se conectando à Internet? Em primeiro lugar, deve-se ter em mente que muitos destes dispositivos não ficam permanentemente conectados, permitindo um *reuso de endereços*. Em segundo lugar, em qualquer dado momento, **há um grande número de interfaces de rede conectadas à Internet que utilizam endereços IP repetidos.**

Esta afirmação parece contradizer o que estudamos até aqui nesta disciplina. Implicitamente, estivemos assumindo que endereços IP deveriam ser únicos na Internet. E, de fato, isso é *geralmente* verdade, mas há algumas exceções — nem tão raras assim — incluindo um dos tópicos desta aula: a técnica NAT.

O NAT (do inglês, *Network Address Translation*) é uma técnica através da qual um único **endereço IP público** (também chamado às vezes de **endereço roteável**) pode ser *compartilhado* por vários dispositivos computacionais. O termo *compartilhado* significa aqui que datagramas originados por estes dispositivos computacionais são recebidos por outros dispositivos na Internet pública com o campo de endereço de origem contendo o tal endereço público.

Endereços IP privados — em oposição aos endereços públicos — são aqueles compreendidos nas faixas `10.0.0.0/8`, `172.16.0.0/12` e `192.168.0.0/16`. Estas faixas são reservadas justamente para que usuários finais possam construir redes IPv4 privadas, com dispositivos que não se comunicam **diretamente** com o restante da Internet, sem a necessidade de um processo complexo de atribuição de faixas únicas de endereçamento IP. Endereços privados não podem — ou não deveriam — aparecer em tabelas de roteamento de roteadores do núcleo da Internet. Logo, um datagrama destinado a um destes endereços passando pelo núcleo da rede muito provavelmente será descartado por ausência de uma rota adequada.

O fato de uma rede ser privada, no entanto, não impede que seus dispositivos se comuniquem de forma *indireta* com os demais equipamentos na Internet. Isso, na verdade, acontece o tempo todo na Internet. Considere, por exemplo, um cenário de uma rede doméstica, em que um roteador é instalado na residência para "distribuir" o acesso à Internet a vários dispositivos computacionais. Este roteador precisa possuir ao menos duas interfaces de rede: uma interface **externa**, através da qual ele se conecta ao ISP que provê o serviço de acesso à Internet, e uma interface **interna**, através da qual o roteador se comunica com os dispositivos dentro da residência.

A interface externa possui um certo endereço IP possivelmente público. No entanto, é muito comum que a interface interna esteja conectada a uma sub-rede privada (*e.g.*, `10.0.0.0/24`). Se um *host* pertencente a esta sub-rede privada deseja enviar um datagrama a um servidor web na Internet pública, o roteador de borda da rede privada não pode simplesmente executar um encaminhamento, transmitindo o pacote pelo seu enlace externo — qualquer resposta que o servidor enviasse de volta ao *host* se perderia no caminho, já que endereços privados não são roteáveis na Internet pública.

A solução proposta pelo NAT é que, antes de realizar o encaminhamento do pacote para a Internet pública, o roteador na borda da sub-rede privada faça uma tradução de endereços. Em outras palavras, este roteador deve substituir o endereço de origem do datagrama — no qual consta o endereço privado do *host* — pelo seu endereço público. Assim, qualquer resposta gerada pelo servidor seria corretamente encaminhada através da Internet pública de volta até o roteador de borda.

O problema desta abordagem é que, ao receber a resposta vinda do servidor web, o roteador de borda não teria como diferenciar este pacote — que deveria ser entregue ao *host* privado — de um outro pacote qualquer que, de fato, seja destinado ao próprio roteador. Para permitir esta diferenciação, o NAT precisa armazenar algum tipo estado de cada pacote encaminhado pelo roteador da sub-rede privada para a Internet pública. Com este objetivo, o NAT utiliza uma tabela — comumente chamada de *tabela NAT* — em que ele anota o **número de porta de origem** do pacote, associado ao endereço IP privado do *host* que o originou. Ao receber um pacote vindo da Internet, o roteador de borda **verifica a porta de destino** — porque, em uma resposta, a porta de origem do pacote original se torna a porta de destino — e consulta a sua tabela. Caso algum mapeamento seja encontrado, o roteador entende que se trata de um pacote direcionado a um dos *hosts* privados. Caso contrário, assume-se que o pacote é destinado, realmente, ao roteador.

No primeiro caso, o roteador desfaz a tradução de endereços: substitui o endereço de destino no cabeçalho IP pelo endereço privado do *host*, de acordo com a informação encontrada na tabela NAT. Após fazer isso, o roteador continua o processo de encaminhamento, operando sobre o datagrama modificado.

Na prática, há mais alguns detalhes a serem considerados. Em primeiro lugar, os mapeamentos armazenados pelo NAT na sua tabela são usados para conexões ou fluxos de dados, e não para datagramas isolados. Isso significa que, por exemplo, todos os pacotes de uma conexão TCP iniciada por um *host* privado para um servidor na Internet pública utilizarão o mesmo mapeamento. Quando o primeiro pacote da conexão passa pelo roteador de borda da rede privada, ele consulta a tabela procurando por um mapeamento já existente para aquela conexão. Caso não encontre, é criado um novo mapeamento que, então, será aplicado a todos os demais pacotes daquela conexão. Repare que isso também vale para fluxos de pacotes UDP, ou para qualquer outro possível protocolo de transporte.

Em segundo lugar, o NAT *pode* também alterar os números de porta — de origem, quando o datagrama deixa a rede privada, ou de destino, quando ele entra — como parte do processo de tradução de endereços. Isso é necessário porque existe a possibilidade de que dois *hosts* diferentes na rede privada decidam estabelecer conexões com um mesmo número de porta de origem simultaneamente. Neste caso, ao receber um pacote de resposta da Internet pública, o roteador que realiza o NAT precisará discernir a qual conexão — e, portanto, a qual *host* privado — pertence este pacote. Logo, quando o primeiro pacote de um fluxo passa pelo roteador de borda, o roteador cria um mapeamento incluindo, ao menos, as seguintes três informações:

1. **Porta original**. Número de porta conforme conhecido pelo *host* privado.
2. **IP privado**. Endereço IP do *host* privado.
3. **Porta pública**. Novo número de porta atribuído pelo roteador para os pacotes do fluxo correspondente.

Por fim, note que os números de porta do roteador de borda são um recurso importante no NAT: se não houver mais números de porta não utilizados, novas conexões ou fluxos de dados não podem ser criados da rede privada para a Internet pública. Por este motivo, o NAT precisa de algum mecanismo para identificar — ou inferir — que certos mapeamentos podem ser removidos da sua tabela. Para conexões TCP, é comum que o NAT monitore os segmentos de controle — SYN, SYNACK, FIN e FINACK — para descobrir rapidamente quando a conexão é fechada e, portanto, seu mapeamento não é mais necessário. Para fluxos UDP, pode-se usar, por exemplo, um temporizador associado a cada mapeamento: se novos pacotes não utilizarem aquele mapeamento durante um determinado intervalo pré-configurado, o mapeamento é esquecido.

1.1 NAT: Motivações

A escassez de endereços IPv4 na Internet é certamente a principal motivação para a utilização do NAT. Houvesse grande oferta de endereços IP, o NAT provavelmente seria abolido, bastando que ISPs recebessem faixas maiores, repassando-as para as redes dos seus clientes.

Entretanto, há algumas outras potenciais vantagens no emprego do NAT. Uma delas é a criação de uma certa *independência* no gerenciamento dos endereços da rede privada. No caso de uma pequena empresa, por exemplo, que muda de ISP, se cada dispositivo possuísse endereços IP públicos,

possivelmente seria necessária a reconfiguração de todos os dispositivos, já que com o novo ISP, provavelmente mudaria a faixa de endereços utilizada pela empresa. Ao utilizar NAT, tal modificação não é necessária, já que basta a alteração do endereço IP público do roteador de borda.

Há também quem argumente que o NAT serve como um mecanismo de segurança, protegendo os equipamentos da rede privada de ataques iniciados na Internet pública. Como os mapeamentos da tabela NAT são criados apenas quando o primeiro pacote de um fluxo **sai da rede privada**, em geral não é possível que um dispositivo externo *inicie* uma conexão para dentro da rede privada. Isso força que qualquer conexão seja necessariamente iniciada pelos *hosts* da rede privada, o que proveria certo grau de segurança adicional.

1.2 NAT: Análise

O NAT é amplamente empregado na Internet atual: redes domésticas quase sempre o utilizam, e ele também é bastante popular em redes institucionais. No entanto, o uso do NAT é bastante controverso.

Um dos motivos para isso diz respeito à organização em camadas da Internet. **O NAT é uma técnica *cross-layer*** — embora ele atue mais diretamente na camada de rede, ele utiliza, e às vezes modifica, informações da camada de transporte (o NAT, aliás, utiliza o número de porta, um endereço da camada de transporte, como uma extensão do endereço da camada de rede). Mecanismos *cross-layer* são muitas vezes empregados com o propósito de otimizar certos aspectos do funcionamento de uma rede, mas eles constituem uma violação da organização em camadas do modelo TCP/IP. Uma das possíveis consequências negativas desta violação é a criação de uma dependência entre implementações particulares das camadas: o NAT só é bem sucedido se ele conhecer os protocolos de transporte que estão sendo utilizados. Se criarmos um novo protocolo de transporte que, em particular, coloque os campos de número de porta em locais diferentes do seu cabeçalho, termos também que modificar as implementações do NAT.

Outro problema do NAT é a impossibilidade de *hosts* externos iniciarem comunicações com *hosts* privados. Embora isso tenha sido apresentado como uma potencial vantagem, em termos de segurança, na seção anterior, esta limitação muitas vezes atrapalha o funcionamento de certas aplicações. Em particular, o NAT geralmente funciona bem para aplicações cliente-servidor — assumindo que o *host* privado seja o cliente —, mas em aplicações P2P ele se torna problemático, impedindo que certos pares sejam usados como servidores e, com isso, quebrando o balanço entre demanda e oferta tão importante neste modelo.

Por fim, outro argumento contrário ao uso do NAT é o de que sua principal motivação — a escassez de endereços IPv4 — já possui uma solução técnica superior: a adoção do IPv6. O IPv6 é a próxima versão do protocolo IP — sucessora do IPv4 — e será discutida em algum nível de detalhe mais a frente nesta aula. Por agora, basta citar que o IPv6 introduz campos de endereço bem maiores que o IPv4, aumentando, em muito, a quantidade de endereços disponíveis.

De toda forma, o NAT é, hoje, uma realidade na Internet. Em parte, isso se deve ao fato desta técnica ser de fácil implantação — ao contrário do IPv6, como veremos adiante. Basta que se configure o roteador de borda para realizar a tradução de endereços, algo relativamente simples. Em teoria, um único roteador de borda, usando um único endereço IP público, daria suporte a mais de 64 mil conexões simultâneas (já que há 2^{16} portas diferentes disponíveis).

Repare, no entanto, que há também desvantagens práticas no uso do NAT. Uma delas é o fato de que, na prática, o número de conexões simultâneas suportadas por um roteador que realiza NAT tende a ser bem menor que as 2^{16} citadas no parágrafo anterior: restrições de memória, por exemplo, comumente limitam o tamanho máximo da tabela NAT bem abaixo deste máximo teórico em implementações reais.

Além disso, o NAT muitas vezes possui uma interação ruim com conexões TCP ociosas. Se um mapeamento na tabela NAT não é utilizado por algum tempo, ele é removido sob a hipótese de que a conexão associada não existe mais. Note que esse é um critério importante mesmo para conexões TCP, já que é possível que, no meio de uma comunicação, os *hosts* percam conectividade com a rede antes de trocarem os segmentos de fechamento de conexão. Se, por outro lado, a conexão estava apenas sem tráfego durante este período, novos segmentos transmitidos não encontrarão o mapeamento correto ao passarem pelo roteador. Na prática, a conexão, que até então era considerada aberta pelos dois *hosts*, será quebrada.

1.3 NAT *Traversal*

Como já explicado em duas ocasiões nesta aula, um dos problemas — ou vantagens, dependendo do ponto de vista — do NAT é a impossibilidade de um *host* externo iniciar uma comunicação com um *host* privado. Infelizmente, para certas aplicações este tipo de interação é importante. Por isso, e

pela enorme popularidade do NAT hoje em dia, este problema já foi bastante estudado, e as soluções encontradas são genericamente denominadas de técnicas para *NAT Traversal*.

Uma solução simples para *NAT Traversal* é a inserção manual de regras estáticas de mapeamento na tabela NAT. Suponha que uma rede doméstica utiliza NAT para compartilhar um único endereço IP público entre vários dispositivos. Suponha que um destes dispositivos execute um servidor web e que desejemos permitir acesso externo (*i.e.*, vindo da Internet pública).

Para que um *host* externo consiga abrir uma conexão TCP para a porta 80 deste servidor web, inevitavelmente ele terá que gerar um datagrama destinado ao endereço IP público do roteador de borda. Entretanto, quando este datagrama chegar ao roteador de borda, este consultará a sua tabela NAT e precisará encontrar um mapeamento indicando que este datagrama deve ser encaminhado para o endereço privado do servidor web, mantendo-se a porta de destino como sendo a porta 80. Se, previamente, inserirmos uma entrada estática e fixa — *i.e.*, que não expire — com este mapeamento, a comunicação será bem sucedida. Este tipo de solução muitas vezes recebe o nome de *encaminhamento de porta*, ou *port forwarding*, em inglês.

A solução de encaminhamento de porta funciona bem para a inclusão de mapeamentos semi-permanentes — *i.e.*, que raramente mudam — na tabela NAT. Neste caso, é razoável assumir que um administrador de rede realize esta configuração manualmente.

No entanto, em certos cenários, aplicações são executadas nos *hosts* da rede privada de maneira mais dinâmica, potencialmente abrindo *sockets* para escuta em portas nem sempre previsíveis. Para estes casos, uma solução mais apropriada seria o uso de um protocolo de comunicação que permitisse que um *host* privado solicitasse dinamicamente ao seu roteador a inclusão de um mapeamento na tabela NAT. Um protocolo que permite isso é o chamado IGD (do inglês, *Internet Gateway Device Protocol*), empregado, por exemplo, por alguns aplicativos P2P.

Para que esta solução funcione é preciso que tanto a aplicação quanto o roteador suporte o IGD. Aliás, ainda que o roteador dê suporte ao protocolo, é preciso que este suporte esteja habilitado, o que nem sempre é verdade. Por este motivo, uma aplicação deve estar preparada para a possibilidade as duas soluções anteriores falharão. Neste caso, outra alternativa relativamente comum na Internet é o uso da técnica de *relaying*.

Um exemplo de aplicação P2P que utiliza *relaying* — em último caso — é o Skype. Se dois dispositivos desejam estabelecer uma chamada, mas ambos se encontram **atrás de NATs**, eles utilizem um outro par da rede que possua um endereço IP público — que pode, inclusive, ser um servidor dedicado a isso — como um intermediário. Ambos os lados da chamada estabelecerão fluxos de dados com o intermediário. Através destes fluxos, cada lado enviará seus pacotes que chegarão à camada de rede do intermediário que fará o encaminhamento para o outro lado.

Repare que **o encaminhamento citado aqui ocorre na camada de aplicação**. O aplicativo Skype rodando no intermediário terá *sockets* abertos para se comunicar com cada um dos lados da chamada. O que é recebido por um *socket*, é enviado pelo outro e vice-versa.

2 ICMP

Quando falamos de protocolos de comunicação, muitas vezes nos referimos aos chamados *pacotes de controle*. Isto é, pacotes que não transportam dados do usuário, sendo utilizados para a troca de informações importantes para o funcionamento do protocolo. Por exemplo, os segmentos SYN e SYNACK do TCP podem ser compreendidos como pacotes de controle.

O IP, no entanto, não define pacotes de controle. Ao invés disso, para a transmissão de informações de controle no nível da camada de rede, utiliza-se um protocolo específico: o ICMP (do inglês, *Internet Control Message Protocol*).

Tanto *hosts* quanto roteadores podem gerar e receber mensagens ICMP. Estas mensagens são usadas primariamente para informar outros elementos da rede sobre condições de erro, mas há também mensagens do ICMP utilizadas com o objetivo principal de dar suporte a certas ferramentas de teste e diagnóstico da rede, como o `ping`.

Embora o ICMP seja um protocolo de camada de rede, suas mensagens são transmitidas em datagramas IPv4, da mesma maneira que TCP e UDP têm seus segmentos encapsulados. Isso pode levar à conclusão errônea de que o ICMP seria um protocolo de camada de transporte, o que não é verdade por conta da sua utilização estar diretamente atrelada à funcionalidades da camada de rede.

Não entraremos em detalhes nesta disciplina sobre o formato de cabeçalho do ICMP. Para os nossos propósitos, é suficiente saber que cada mensagem ICMP tem um *tipo* que pode ser mais especificado na forma de um *código*. O par tipo-código, portanto, identifica a semântica de uma mensagem ICMP.

Um exemplo de mensagem ICMP é a de tipo 3, código 1, que em termos mais simples significa *Host de destino inalcançável*. Esta mensagem é normalmente gerada pelo roteador de borda de uma

sub-rede ao receber um datagrama que deve ser encaminhado a um endereço IP daquela sub-rede, mas que não parece estar associado a nenhum *host* conectado à rede naquele momento. Esta mensagem ICMP é enviada, então, de volta ao *host* que originou o datagrama que causou o erro. Há também mensagens similares para os casos de *Rede inalcançável* — *i.e.*, o roteador não sabe como encaminhar o pacote para sub-rede a qual o endereço de destino do datagrama pertenc — e *Porta inalcançável* — o *host* de destino não encontrou nenhum *socket* aberto na porta especificada.

Outras duas mensagens ICMP comumente utilizadas na Internet são a *echo request* e a *echo reply*. Ao contrário dos exemplos anteriores, estas mensagens não são geradas em caso de erro. Na verdade, elas dão suporte à ferramenta de diagnóstico de rede `ping`. O `ping` gera uma mensagem do tipo *echo request* endereçada ao *host* especificado pelo usuário. Ao receber um *echo request*, um *host* gera como resposta um *echo reply*. Ao receber o *echo reply*, o `ping` imprime uma mensagem na tela, mostrando — entre outras estatísticas — que *host* especificado pelo usuário está acessível.

Há, também, uma mensagem ICMP destinada a avisar à origem que um de seus datagramas foi descartado pela rede por conta de um TTL expirado. Neste caso, o roteador que realizou o descarte se encarrega de gerar a mensagem ICMP de erro.

Existem, ainda, mensagens ICMP previstas para que roteadores possam disseminar informações de roteamento para outros roteadores. No entanto, como veremos em aulas posteriores, protocolos de roteamento atualmente populares na Internet raramente utilizam o ICMP para este tipo de propósito.

Outra mensagem ICMP prevista, mas raramente utilizada na prática, é a *Source quench*. Seu propósito é avisar proativamente à origem de um fluxo de dados sobre uma situação de congestionamento na rede.

2.1 ICMP e Traceroute

Em aulas anteriores, o utilitário `traceroute` foi utilizado em demonstrações para exibir a rota utilizada para encaminhar pacotes entre dois *hosts*. Assim como a ferramenta `ping`, o `traceroute` utiliza o ICMP para obter informações que auxiliem no diagnóstico da rede — neste caso, da rota utilizada. Ao *contrário* do `ping`, no entanto, a funcionalidade do `traceroute` não têm suporte nativo no ICMP. O **que o `traceroute` faz é provocar situações de erro na rede** que forcem a geração de pacotes ICMP e permitam à ferramenta obter as informações que deseja sobre a rede.

O `traceroute` funciona da seguinte maneira. O programa abre um *socket* UDP e envia vários datagramas destinados ao *host* de destino especificado pelo usuário. Como porta de destino, o `traceroute` utiliza uma porta que, *provavelmente*, não estará associada a nenhum *socket* no destinatário. Antes de enviar os três primeiros datagramas, o `traceroute` configura o *socket* UDP para que um TTL inicial de 1 seja utilizado. A cada três datagramas enviados, o `traceroute` incrementa o valor do TTL em uma unidade. Isso prossegue até que a execução se encerre.

Ao configurar o valor do TTL inicial dos três primeiros datagramas para 1, o `traceroute` faz com que estes datagramas sejam necessariamente descartados no roteador de primeiro salto do caminho até o destinatário. Este descarte é interessante para o `traceroute` porque ele estará atrelado ao envio, pelo roteador de primeiro salto, de uma mensagem ICMP reportando o problema. Ao receber a mensagem ICMP de erro, o `traceroute` extrai o endereço IP do roteador e o exibe na tela. Aumentando gradativamente o TTL dos datagramas e repetindo o processo, o `traceroute` obtém os endereços de todos os saltos intermediários no caminho até o destinatário.

Mas como o `traceroute` sabe quando parar? Repare que quando o TTL é suficientemente alto para que o datagrama chegue ao destinatário, não haverá descarte por TTL expirado e, portanto, a mensagem ICMP de erro correspondente também não será entregue. Por isso, é necessário algum outro critério de parada. Lembre-se que os datagramas enviados pelo `traceroute` são direcionados a uma porta UDP a qual provavelmente não haverá *socket* associado no destinatário. Se isso é verdade, ao receber o datagrama, o *host* de destino ainda gerará uma mensagem ICMP de erro, mas agora para reportar que a *porta de destino é inalcançável*. Ao receber esta mensagem de erro, o `traceroute` sabe que o destinatário foi alcançado e que o caminho completo foi traçado.

3 IPv6

O IPv6, padronizado inicialmente na RFC 2460 de 1998, é a próxima versão do protocolo IP, sucessora do IPv4. Embora a Internet, hoje, funcione com o IPv4, há uma série motivos que justificam a proposta de uma nova versão do protocolo IP.

A principal motivação para o IPv6 já foi discutida nesta aula: a escassez de endereços IPv4 na Internet. Para atacar isso, o IPv6 utiliza endereços de 128 bits, bem maiores que os do IPv4. Mesmo

diante das projeções atuais de crescimento da Internet, ainda não se vislumbra um cenário em que $2^{128} bits$ (um número com quase 40 algarismos decimais) endereços sejam insuficientes.

Além disso, outra motivação era tentar fornecer um suporte mais adequado à diferenciação de tráfego com vistas ao suporte de QoS. Para tanto, o IPv6 aumentou o número de bits de cabeçalho que podem ser utilizados para esse fim, além de permitir uma categorização bem mais flexível dos datagramas em *tipos*.

Uma outra motivação importante era a simplificação do processamento do cabeçalho. O cabeçalho IPv4 tem tamanho variável, por conta do campo de opções — que pode ou não estar presente e, neste último caso, também pode variar de tamanho. O processamento de cabeçalhos de tamanho variável é mais complexo, já que é as tarefas executadas dependem de valores que serão encontrados — ou não — no próprio cabeçalho. Embora esta diferença possa parecer insignificante, para um roteador que precisa processar centenas de milhares ou até milhões de pacotes por segundo, qualquer redução em complexidade se torna representativa. Por isso, o IPv6 adotou um formato de cabeçalho de tamanho fixo — e, portanto, de mais fácil processamento. O IPv6 ainda dá suporte a opções similares às do IPv4, mas para isso utiliza um outro cabeçalho separado, fazendo com que, na prática, estas opções, se existirem, sejam a carga útil do datagrama IPv6.

Outro fator que resulta em alta complexidade nos roteadores IPv4 é a possibilidade de fragmentação. A divisão de um datagrama original em vários pedaços é uma tarefa custosa e, na prática, percebeu-se que ela pode ser evitada. No próprio IPv4, a fragmentação muitas vezes não ocorre por decisão das próprias camadas superiores. Um exemplo disso é o TCP: para que uma possível fragmentação de segmentos não interfira em mecanismos do TCP, como a estimativa do RTT, por exemplo, o TCP instrui o IP a nunca fragmentar seus segmentos. Isso é feito atribuindo o valor 1 a uma *flag* do cabeçalho IPv4 denominada *do not fragment*. Quando esta *flag* é 1 e o datagrama é maior que o MTU do enlace do próximo salto, o roteador simplesmente descarta o datagrama.

Neste ponto, pode-se perguntar porque o TCP preferiria que seu segmento seja descartado a ser simplesmente fragmentado — e, neste caso, entregue. A resposta é que, além de descartar o datagrama grande demais, o roteador também gera uma mensagem ICMP de volta ao *host* de origem avisando sobre o erro — informando, inclusive, o MTU do enlace que causou o descarte. Com isso, o TCP pode reduzir o valor do seu MSS, gerando segmentos menores que agora respeitem o MTU do enlace problemático. Repare que estes segmentos ainda podem ser grandes demais para o MTU de um enlace posterior no caminho: neste caso, novamente haverá descarte, envio da mensagem ICMP e nova redução no MSS. Em certo momento, este processo levará o TCP a encontrar o maior valor possível de MSS tal que os MTUs dos enlaces ao longo do caminho sejam respeitados. Este técnica é conhecida como *Path MTU Discovery* e é amplamente utilizada na Internet atualmente.

Levando em consideração que vários protocolos e aplicações hoje já habilitam a *flag do not fragment* no IPv4 e que há uma técnica que permite a descoberta do MTU de um caminho, o IPv6 simplesmente aboliu a fragmentação. Datagramas IPv6 que são maiores que o MTU do próximo enlace são simplesmente descartados e a origem é avisada do fato. Note que esta mudança segue perfeitamente a filosofia de inteligência nas bordas da Internet: removeu-se uma funcionalidade complexa do núcleo e a responsabilidade de solução foi delegada às bordas.

Outras mudanças do IPv6 em relação ao IPv4 incluem:

1. **Remoção do campo *checksum***. Outra alteração que visa a simplificação do processamento dos datagramas. Boa parte dos protocolos de camada de enlace já incluem algum tipo de mecanismo de verificação de integridade. Desta forma, percebeu-se que o *checksum* do IPv4 é muitas vezes redundante **em relação aos mecanismos de verificação das camadas inferiores**. Por esta razão, este *checksum* quase nunca é útil.

2. **Criação do ICMPv6**. Assim como o ICMP acompanha o IPv4, o ICMPv6 provê funcionalidades para que *hosts* e roteadores reportem erros e enviem outras informações em redes IPv6.

3.1 IPv6: Transição

Como já citado, a primeira RFC do IPv6 data de 1998. Mesmo assim, depois de quase 20 anos, a Internet é majoritariamente uma rede IPv4.

A razão para esta lentidão na migração para o IPv6 está na natureza decentralizada da administração da Internet. Não é possível forçar todos os que equipamentos migrem simultaneamente para o IPv6 porque não há uma única entidade com autoridade administrativa sobre todos os dispositivos conectados à Internet.

A solução, portanto, é uma migração gradual. Entretanto, é preciso levar em conta a incompatibilidade de versões do IP: um roteador IPv4 não é capaz de processar datagramas IPv6.

Para contornar esta dificuldade, a Internet utiliza algumas soluções de coexistência entre IPv4 e IPv6. Uma bastante popular é baseada no conceito de **tunelamento**.

Para entender o conceito de tunelamento, considere o seguinte cenário. Suponha que as redes institucionais de duas empresas, A e B, operem ambas sobre IPv6. Entretanto, o caminho disponível — passando pela Internet pública — para encaminhar pacotes entre os roteadores de borda de A e B inclui apenas roteadores IPv4. Se um *host* da empresa A gera um datagrama IPv6 destinado a um *host* da empresa B, o roteador de borda de A não pode simplesmente enviá-lo para o roteador de próximo salto: o datagrama seria descartado pela falta de suporte ao IPv6.

Ao invés disso, quando o datagrama IPv6 chega ao roteador de borda da empresa A, ele pode encapsular este pacote em um cabeçalho IPv4 endereçado ao endereço IPv4 do roteador de borda da empresa B. Para todos os efeitos, o resultado deste encapsulamento é simplesmente um datagrama IPv4, que pode ser normalmente encaminhado pelos roteadores IPv4 intermediários.

Ao receber o datagrama IPv4 vindo da Internet pública, o roteador de borda da empresa B verifica que a carga útil é um datagrama IPv6 e o desencapsula. O datagrama IPv6 resultante pode, então, ser encaminhado normalmente pela rede institucional da empresa B, até seu destinatário.

De certa forma, a situação descrita neste cenário mostra duas "ilhas" de equipamentos IPv6 interconectadas através de uma rede puramente IPv4. O encapsulamento do datagrama IPv6 em um IPv4 basicamente cria um túnel que esconde a real natureza do datagrama original para a rede IPv4. Ao final deste túnel — *i.e.*, no roteador de borda da empresa B — o datagrama volta a ser puramente IPv6.

Embora haja mecanismos de coexistência entre IPv4 e IPv6, e embora o IPv6 exista há duas décadas, o migração entre versões ainda se encontra nos estágios iniciais. Estatísticas de grandes provedores de conteúdo, como Google e Akamai, sugerem que menos de 20% dos *clientes* possuem suporte ao IPv6.

Este ritmo de migração é surpreendente se considerarmos a enorme evolução que as camadas superiores da pilha de protocolos TCP/IP sofreram no mesmo período, com o advento e popularização de uma série de aplicações outrora impensáveis, como redes sociais e serviços de *streaming* de vídeo de alta qualidade sob-demanda. A explicação para isso está no fato de que as migrações na camada de aplicação são muito mais simples: se um novo protocolo de aplicação surge na Internet, basta que cliente e servidor o implementem: a rede é apenas um canal de comunicação. Para protocolos na camada de rede, a situação é mais difícil porque a utilização muitas vezes depende da colaboração de várias entidades.