UniFil
EDUCAÇÃO A DISTÂNCIA

Unidade 3

As camadas de Rede e Transporte

Abertura

Apresentação

Esta unidade apresenta os meios físicos e, de maneira resumida, um breve histórico das redes de computadores até os dias de hoje, bem como as classificações, tipos de redes, topologias e outros objetos de estudo da disciplina.

Aprenderemos também sobre os modelos e padrões que consistem numa parte fundamental da disciplina, formalizando determinados tópicos para que as redes funcionem como a vemos e usamos. Com isto, faremos um breve passeio pelas camadas dos modelos, e a proposta de cada uma.

Ainda nesta unidade, destaco a importância de compreendê-los e através de um exemplo de uma requisição que fazemos nos nossos navegadores, algo rotineiro e que muitas vezes nos esquecemos da complexidade por trás, dada a transparência e facilidade do ato.

Objetivos

- Conhecer os serviços da camada de rede;
- Conhecer os serviços da camada de transporte;
- Compreender a divisão entre camadas e as responsabilidades de cada camada;

Conteúdo programático

Aula 01 – A Camada de Rede

Aula 02 – A Camada de Transporte

Referências

COMER, Douglas E. **Redes de computadores e internet:** abrange transmissão de dados, ligações inter-redes, web e aplicações. 4. ed. Porto Alegre, RS: Bookman, 2008. 632 p.

FOROUZAN, Behrouz A. **Comunicação de Dados e Redes de Computadores.** 4 ed. [*S.l.*]: AMGH Editora, 2009.

KUROSE, James F.; ROSS, Keith W. **Redes de computadores e a internet:** uma abordagem top-down. 3. ed. São Paulo: Addison-Wesley, 2006. 634 p.

TANENBAUM, Andrew S.; WHETERALL, David J. **Redes de Computadores.** [*S.l.*]: Pearson Prentice Hall, 2011. p. 582.

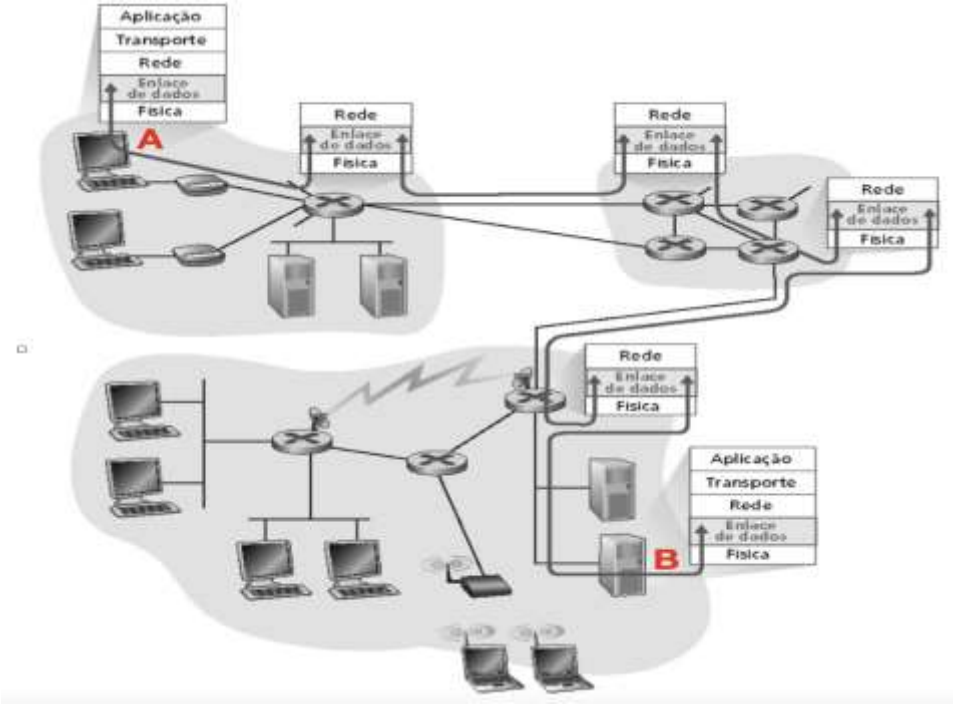
A Camada de Rede

Prosseguindo os estudos, já vimos que cada camada provê serviços para a camada superior e utiliza-se dos serviços fornecidos pelas camadas inferiores para enviar os dados de um lado para o outro. A camada de rede é responsável por:

- Endereçamento
- Roteamento
- Interconexão
- Encapsulamento
- Fragmentação

A camada física é responsável por transportar **bits**, a camada de enlace transporta **quadros (frames)** entre os **nós (nodes)** e a camada de rede transporta **datagramas** entre **hosts**. Utilizando-se da camada de enlace, agora conseguimos enviar dados entre hosts que estão separados entre si, mas conectados através de muitos outros dispositivos e tecnologias diferentes no caminho. A preocupação na camada de enlace é a ligação entre nós, agora na camada de rede, utilizando a conexão entre diversos nós, é possível criarmos caminhos. Na Figura 1, temos o exemplo da comunicação entre A e B.

Figura 1: Ligação da camada de enlace



Fonte: Adaptado de KUROSE, 2006.

Como observamos na Figura 1, para nós usuários, o caminho percorrido dos dados é transparente quando saem do ponto A ao ponto B. Mas os datagramas podem percorrer diversos caminhos, logo, não podemos garantir que o caminho será igual para todos. Os caminhos podem ser definidos de forma estática ou automaticamente. Isto é exatamente o comportamento quando acessamos alguma página web, ela não está na nossa rede

localmente, muitas vezes, não está nem no nosso país, tendo seus arquivos hospedados em algum servidor fora. Só conseguimos acessar, pois existe uma conexão de diversas redes.

A camada de rede recebe o pacote da camada de transporte, encapsula o pacote em um datagrama da camada de enlace e repassa para a camada abaixo. Os nós intermediários na comunicação, desencapsulam os frames, os datagramas e analisam os pacotes no nível de rede, para então encaminhar os dados para outro ponto, ou enviam o *payload* para a camada superior continuar o tratamento da informação.

É comum definirmos e confundirmos a camada de rede com o protocolo **Internet Protocol (IP)**, mas veremos que existem outros protocolos, como ARP, ICMP, IPSec e outros.

Serviços da camada de rede

Como já discutimos, cada camada provê serviços para a camada superior, e no caso da camada de rede os serviços são: Empacotamento, Endereçamento, Roteamento, Interconexão e Fragmentação.

Empacotamento

Empacotamento ou **Encapsulamento** é considerado por alguns autores a principal tarefa da camada de redes, e consiste no encapsulamento da carga útil (***payload***) em um pacote da camada de rede na origem e o desencapsulamento no destino, não havendo a utilização o *payload*, nem a alteração do seu conteúdo. A única alteração possível é a fragmentação do pacote. Caso o pacote seja fragmentado, a camada de rede é a encarregada de aguardar o recebimento de todos, reorganizá-los e entregar para a camada superior.

O host de origem recebe os dados da camada de cima, adiciona no cabeçalho endereço de origem e destino, e outras informações (discutidas adiante) e o *payload*, encapsula e entrega para a camada de enlace. Nenhum host ou roteador durante o percurso possui permissão para alterar qualquer campo; apenas acessar os campos para efetuar a comparação do pacote com o endereço de destino, para encaminhá-lo ou passar para a camada superior.

Roteamento

O **Roteamento** é um serviço considerado tão importante quanto o Empacotamento. É a tarefa de encontrar a melhor rota entre a origem e o destino, pois como estudamos, existem diversas rotas de diferentes tipos de tecnologias entre a origem e o destino. O roteamento utiliza-se de alguns protocolos de roteamento, que mapeia os vizinhos e possui algumas políticas para manter a lista sempre atualizada.

Encaminhamento

É a ação propriamente dita que o cada roteador toma baseado na **tabela de roteamento** ou **tabela de encaminhamento**. Quando um datagrama é recebido, ele deve encaminhar para a rede (*unicast*) ou as redes (*multicast*) as quais ele está conectado. O

roteador acessa o cabeçalho para buscar o endereço de destino ou alguma outra informação, para determinar a interface de saída correspondente na tabela de roteamento.

Controle de erros

Como o datagrama pode ser fragmentado em cada roteador, a verificação de erros se torna ineficiente na camada de redes. Mas, existe um *checksum* do cabeçalho, para controle e prevenção de erros.

Controle de fluxo

Não existe um controle de fluxo na camada de redes. Uma vez que o datagrama está pronto, ele é enviado sem preocupação se o destino terá capacidade de receber. Uma das razões é a simplicidade que esta camada foi projetada. A camada de enlace possui um controle de acesso ao meio, como forma de controle do fluxo também, mas em um nível abaixo, logo é um serviço que a camada de rede não precisa se preocupar.

Outro motivo, é a utilização de *buffers* na camada de aplicação, que não consomem os dados simultaneamente à sua recepção. Desta forma, o receptor armazena dados suficientes para sua utilização posterior.

Controle de congestionamento

O controle de congestionamento ocorre quando um número muito grande de datagramas fica concentrado em algum ponto. Tal efeito pode ser causado pelo envio elevado de datagramas à capacidade da rede ou dos roteadores, ocasionando na perda ou descarte dos datagramas, podendo surtir um efeito de colapso, dependendo da situação.

Qualidade de Serviço - QoS

Com os diversos tipos de serviços e mídias trocadas, a qualidade de serviço passou a ser discutida, especialmente por causa das aplicações de comunicação em tempo real de áudio e vídeo. Para manter a simplicidade da camada de rede, tal funcionalidade é na maioria das vezes implementada em camadas superiores.

Segurança

Nunca foi uma preocupação, pois quando a Internet foi concebida, era utilizada por um pequeno grupo de usuários, na sua maioria do meio acadêmico e, portanto, foi projetada sem funcionalidades para segurança. No entanto, com a dimensão alcançada atualmente e a sua importância, a segurança é um ***big deal***. Para isto, na camada de redes, o IPSec foi proposto.

Videoaula 1

Utilize o QR Code para assistir!

Agora, assista ao vídeo em que o professor aborda todo esse conteúdo introdutório.

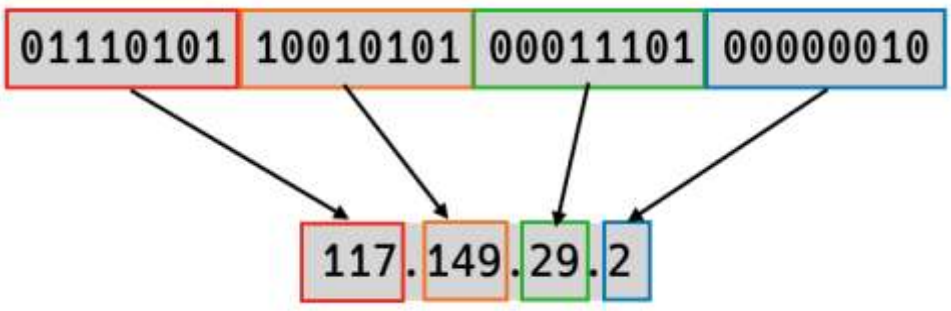
Endereçamento lógico

Como toda camada, na camada de rede também acrescentamos alguns dados, no caso, incluímos um cabeçalho com os endereços lógicos do transmissor e do receptor do pacote proveniente da camada superior. É essa informação que nos ajudará a encontrar o caminho, e por estarmos lidando com algo mundial, nós padronizamos esse endereço, que é o que conhecemos como endereço IP.

Atualmente, trabalhamos com endereçamento com 32 bits, que é o chamado IPv4. Com o crescimento exponencial que tomou a Internet, logo, a quantidade possível de endereços se mostrou insuficiente, e assim, surgiu o IPv6, com o endereçamento de 128 bits. Mas o que é esse tal de endereçamento? Por que a necessidade de 32 ou 128 bits?

A ideia de endereçamento é que ele seja exclusivo dentro da rede, para que seja possível saber para onde o pacote deve ser encaminhado. E como já mencionado, é um endereço global. Assim, se eu quiser acessar a Internet, eu tenho que ter um endereço único, que identifique a minha máquina única e exclusivamente. Assim, o IPv4 surgiu com endereçamentos de 32 bits, ou 4.294.967.296 (2^{32}) endereços possíveis. A formatação segue um padrão binário, mas geralmente usamos valores decimais para escrever os endereços, conforme podemos ver na Figura 2.

Figura 2: Endereçamento IPv4 com 32 bits



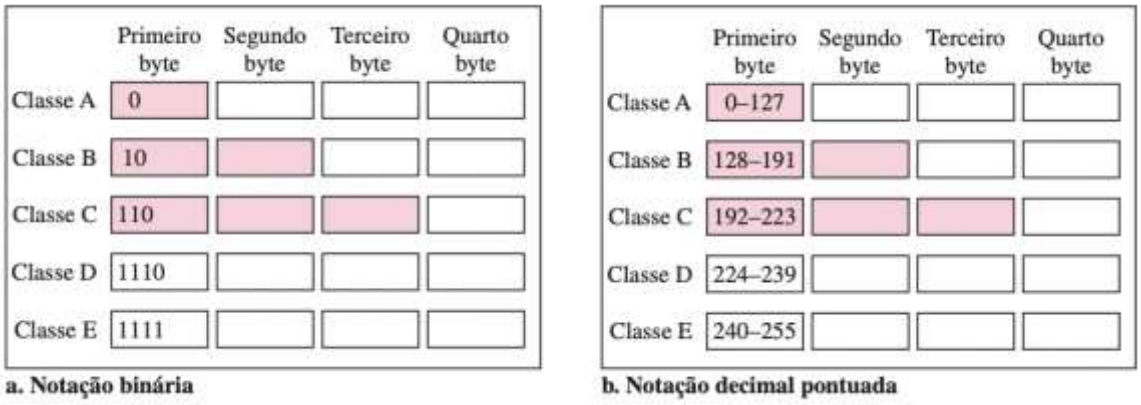
Fonte: o autor.

A Figura 2 ilustra um endereço IPv4 tanto na notação binária como na notação binária e decimal pontuada. Observe que, pelo fato de cada byte (octeto) ser composto por 8 bits, cada número na notação decimal pontuada compreende um valor que vai de 0 a 255. Claro

que para nós lermos, é muito mais fácil a notação decimal pontuada, mas para o computador ele entende apenas a notação hexadecimal.

Um dos problemas que gerou a escassez dos endereços IP foi que no seu início, usávamos a ideia de **endereçamento com classes**. Consiste em dividir o espaço de endereçamento em cinco classes: A, B, C, D e E. Cada classe uma parte do espaço dos bits iniciais, como mostrado na Figura 3.

Figura 3: Endereçamento com Classes



Fonte: FOROUZAN, 2009.

Os endereços das classes D e E são especiais, mas também fazem parte do desperdício, uma vez que na classe D, previram erroneamente uma necessidade de grupos, e os endereços da classe E, como geralmente ocorre na padronização de protocolos, reservaram para uso futuro, sendo que uma pequena quantidade foi de fato utilizada. Mas, o maior problema foi na distribuição dos endereços de classe A, B e C. Os blocos de endereços de classe A eram muito grandes, e os blocos de classe C, muito pequenos, mas os de classe B também eram grandes. Em um bloco da classe A, era possível endereçar 16.777.216 hosts, na classe B, 65.536 hosts e por fim, na classe C apenas 256 hosts.

Os endereços nas classes A, B e C são divididos em netid e hostid, que dependendo da classe tem comprimento variável. Para encontrar estes valores, usamos o conceito de máscara. Uma máscara é composta de 1s contíguos, seguidos por 0s. Na Figura 4, temos as máscaras das classes A, B e C.

Figura 4: Máscaras-padrão para endereçamento com classes

Classe	*Binária*	*Decimal Pontuada*
A	11111111 00000000 00000000 00000000	255.0.0.0
B	11111111 11111111 00000000 00000000	255.255.0.0
C	11111111 11111111 11111111 00000000	255.255.255.0

Fonte: FOROUZAN, 2009.

Para diminuir um pouco o desperdício, foi introduzido as **sub-redes**, assim, se uma organização recebesse bloco de endereços de classe A ou B, poderia dividir em vários grupos menores. Faz-se necessário a utilização de 1 bit na máscara também. Em determinado momento, os endereços de classe A e B se esgotaram, então foi necessário o agrupamento de blocos de classe C, o que formou as chamadas **super-redes**, neste caso, diminuímos 1 bit na máscara.

Indicação de Leitura

Para quem tiver curiosidade, acesse o link abaixo para ter acesso ao relatório da distribuição de endereços IPv4. Disponível em: https://ipv4.potaroo.net/. Acesso em: 31 mai. 2022.

Com a explosão do crescimento da Internet e as falhas no método e distribuição de endereçamento, os endereços se esgotaram, e a solução adotada para mitigar o problema foi utilizar o endereçamento sem classes. Para tanto, algumas restrições foram necessárias: os endereços em um bloco devem ser contíguos; o número de endereços em um bloco deve ser uma potência de 2 e o primeiro endereço tem de ser igualmente divisível pelo número de endereços. No endereçamento sem classes, a máscara para um bloco pode ter qualquer valor entre 0 e 32.

Para escrever um endereço, geralmente usamos a **notação barra** ou **CIDR (Classless Interdomain Routing)**, que mostra a máscara na forma de **/n**, definindo o primeiro e o último endereço, bem como o número de endereços disponíveis. Para um melhor entendimento, segue o exemplo, dado o endereço 175.123.221.0/28. Como calculamos o primeiro e o último endereço, número de endereços disponíveis e a máscara?

O número de endereços é obtido através de 2^{32-n} , nosso n é 28, então temos que $2^{32-28} \Rightarrow 2^4 = 16$. Temos no total, 16 endereços. O primeiro endereço é calculado, colocando 0 os 4 bits mais à direita. Teremos então: 10101111 1111011 11011101 00000000, o último endereço é calculado colocando 1 nos 4 bits mais à direita, ficando: 10101111 1111011 11011101 00001111. Com isto, temos um intervalo de endereços de: 175.123.221.0 - 175.123.221.15. A máscara é formada por 28 bits 1s contiguos, resultando em: 11111111 1111111 11111111 11110000, resultando na máscara 255.255.255.240/28.

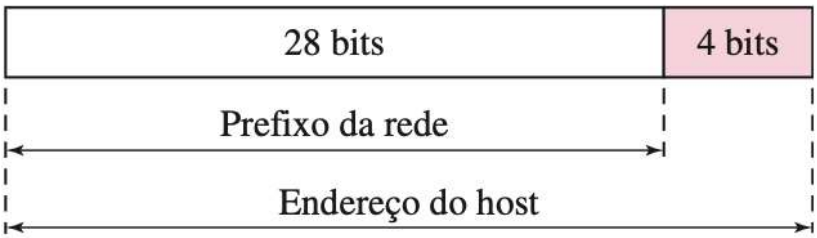
Indicação de Leitura

Para quem tiver curiosidade, acesse o link abaixo para ter acesso ao relatório da distribuição de endereços IPv4. Disponível em: https://datatracker.ietf.org/doc/html/rfc1518. Acesso em: 31 mai. 2022.

O primeiro endereço tem um tratamento especial, é chamado de endereço de rede pois define a rede da organização. O último endereço também é especial, e é chamado de endereço de multicast, utilizado para enviar mensagem para todos os outros endereços dentro da faixa. Logo, temos na verdade apenas 14 endereços para alocar.

Podemos utilizar essa ideia, e criar níveis de hierarquia dentro de uma rede, construindo subredes e assim por diante. Utilizando a mesma rede 175.123.221.0/28, temos a ideia da Figura 5.

Figura 5: Dois níveis de hierarquia num endereço IPv4



Fonte: FOROUZAN, 2009.

Mas podemos utilizar os 4 bits restantes, e quebrar em sub-redes. Se usarmos 1 bit, teríamos duas sub-redes com 8 endereços cada. Ou segmentar a rede em sub-redes menores ainda, usando 2 bits, teríamos 4 sub-redes de 4 endereços cada. Claro, neste exemplo, não faz muito sentido segmentar tanto uma rede assim, mas para os ISPs (Internet Service Provider — Provedor de Acesso à Internet), esta é uma função crucial para organizar as redes, podendo distritbuir o endereçamento regionalmente e logicamente. A alocação de endereços é responsabilidade da ICANN (*Internet Corporation for Assigned Names and Addresses*).

Outra ferramenta para contornar a falta de endereços é a utilização de NAT (*Network Address Translation*). Assim, internamente, as organizações podem usar um grande número de endereços, sem precisar ter os registros ou avisar os ISPs, e externamente, eles usam o endereço global provido pelo ISP. Esse conjunto de endereços utilizados internamente são denominados **endereços privados**. Provavelmente, você deve estar conectado a uma rede usando endereço privado. O intervalo pode ser visto na Tabela 1.

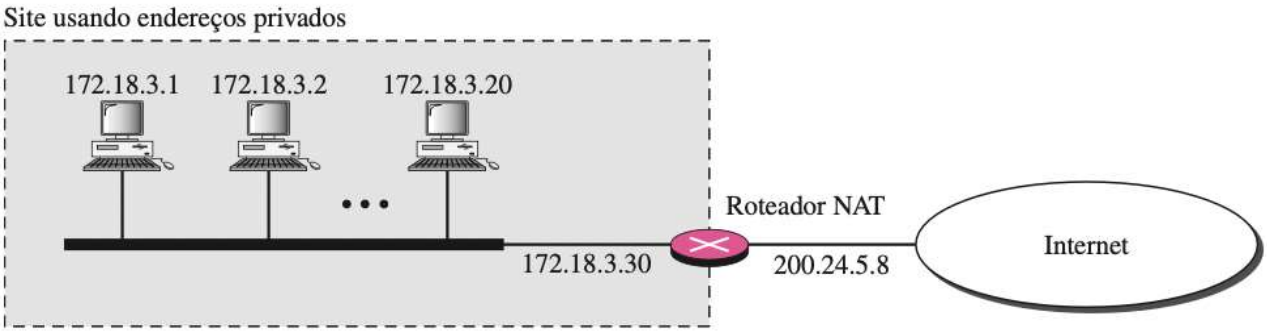
Tabela 1: Endereços para redes privadas

Intervalo			*Total*
10.0.0.0	a	10.255.255.255	2^{24}
172.16.0.0	a	172.31.255.255	2^{20}
192.168.0.0	a	192.168.255.255	2^{16}

Fonte: FOROUZAN, 2009.

O roteador que interliga a rede ao endereço global usa um endereço privado e um endereço global. A rede privada é transparente para o restante da Internet; o restante dela enxerga apenas o roteador NAT com o endereço 200.24.5.8.

Figura 6: Implementação do NAT



Fonte: FOROUZAN, 2009.

Indicação de Leitura

Acessem a RFC que apresenta o NAT.

Disponível em: https://datatracker.ietf.org/doc/html/rfc2663. Acesso em: 31 mai. 2022.

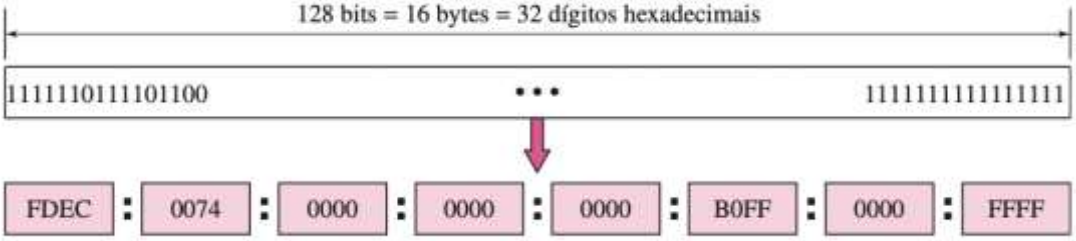
Além do NAT, temos também o DHCP que automatiza o processo de endereçamento IP unicast, adicionando e removendo os endereços e centralizando o gerenciamento. Um servidor DHCP mantém um conjunto de endereços IP e quando um cliente se conecta na rede, inicia o processo de concessão do endereço. Quando o cliente se desconectar, o endereço volta a ficar disponível para que outro dispositivo possa utilizá-lo.

Indicação de Leitura

Acessem a RFC que apresenta o DHCP. Disponível em: https://datatracker.ietf.org/doc/html/rfc2131. Acesso em: 31 mai. 2022.

Apesar de todos os esforços, o endereçamento IPv4 possui um limite e, portanto, um novo formato já foi proposto e estamos no momento de transição. O novo formato é conhecido como IPv6, constituído de 16 bytes ou 128 bits de comprimento. Por ser mais extenso, dificilmente vemos a notação binária sendo utilizada, e o formato decimal também foi descartado. A nova forma de endereçamento utiliza uma notação hexadecimal separada por dois pontos, a Figura 7 ilustra a notação do IPv6.

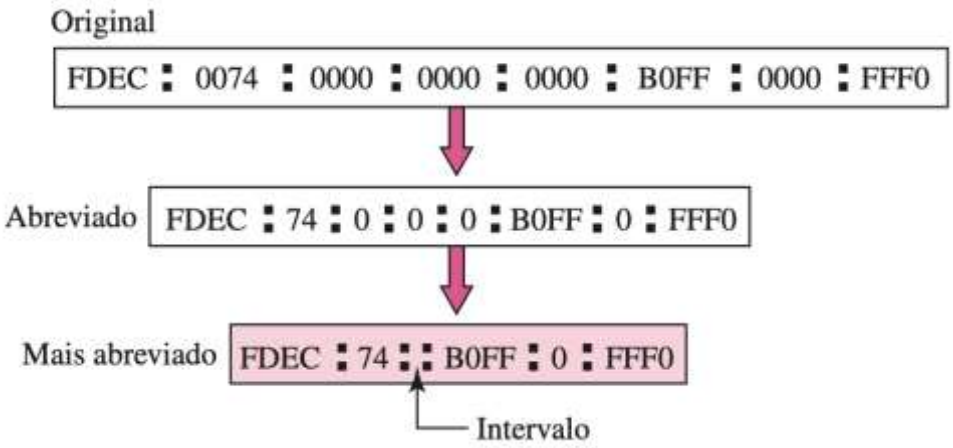
Figura 7: Endereçamento IPv6



Fonte: FOROUZAN, 2009.

Por ser muito longo, e ainda termos muitos dígitos 0 não significativos, podemos omitir, ou melhor, abreviar a forma de escrever o endereço.

Figura 8: Abreviação do endereço IPv6



Fonte: FOROUZAN, 2009.

Dado que usamos 128 bits, nós temos 2^{128} endereços possíveis, o que dá um número tão grande, 340 undecilhões de endereços. Alguns exemplos para ilustrar como esse número é tão grande, se cada endereço IPv6 fosse convertido em 1 cm^2, seria possível criar 7 camadas em volta da Terra, ou se fossemos registrar cada átomo existente na Terra, seria possível registrar em 100 outros planetas semelhantes também. Apesar de todo o espaço, os projetistas já definiram algumas categorias, que podem ser vistas na RFC 2460 e outras utilizadas para definir o funcionamento do protocolo.

Indicação de Leitura

Acessem a RFC que apresenta o IPv6. Disponível em: https://datatracker.ietf.org/doc/html/rfc2460. Acesso em: 31 mai. 2022.

Videoaula 2

Utilize o QR Code para assistir!

Agora, assista ao vídeo em que o professor aborda todo esse conteúdo do endereçamento IPv4 e IPv6.

Sabemos que as camadas física e de enlace de dados de uma rede operam localmente, sendo responsáveis pela entrega de dados na rede de um nó para o seguinte, ligando os *links*. Em cada link estão envolvidas duas interfaces de camadas físicas e duas camadas de enlace de dados. Mas quando os dados chegam na interface, como saber qual caminho percorrer até o destino? Não há nenhuma informação nestas camadas que auxiliem nessa tomada de decisão. É na camada de rede que introduzimos um cabeçalho que contém os endereços lógicos de origem e destino, e nesta camada temos os roteadores e switches como atores principais.

A camada de rede é responsável por consultar sua tabela de rotas para encontrar informações de roteamento (como a interface de saída do pacote ou o endereço físico do nó seguinte) e se o pacote for muito grande, ele será fragmentado. A comutação na camada de rede na Internet usa a abordagem de datagramas para a comutação de pacotes.

Comutação é uma expressão herdada de circuitos, é o simples ato de trocar a direção do sinal elétrico, no nosso caso, trocar a direção dos datagramas. O roteador cria uma ligação entre uma porta de entrada e uma(s) porta(s) de saída(s). Podemos dividir a comutação em duas abordagens: **comutação de pacotes** e **comutação de circuitos**.

Na comutação de pacotes, o dispositivo de origem envia uma mensagem que é fragmentada em diversos pacotes, e o destino recebe um a um, e aguarda todos os pacotes de uma mesma mensagem para repassar para a camada superior. Além disso, a entrega de um pacote pode ser realizada usando-se um **serviço de rede orientado a conexões** ou um **sem conexão**.

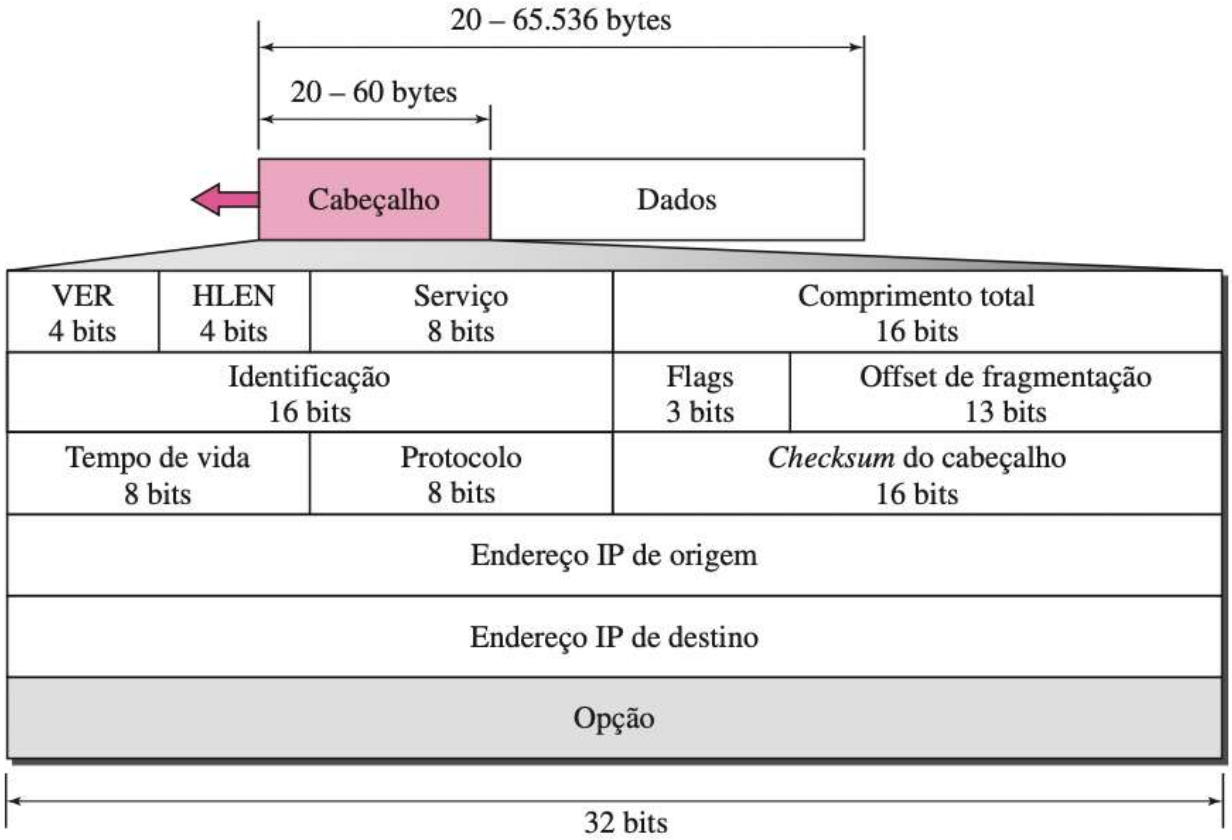
Em um serviço orientado a conexões, a origem estabelece primeiro uma conexão com o destino antes de iniciar o envio de um pacote. Considera-se que exista relação direta entre os pacotes, eles usam uma mesma rota e seguem uma ordem. A conexão é encerrada quando todos os pacotes forem entregues. Por utilizar a mesma rota, não existe a necessidade de ficar recalculando a rota. Esse tipo de serviço é usado em uma metodologia de circuitos virtuais para a comutação de pacotes, como no Frame Relay e ATM.

Já num serviço sem conexão, trata os pacotes de forma independente, isso significa que eles não têm relação direta um com o outro. A rota utilizada pode ou não ser a mesma também. Esta é a abordagem que a Internet adotou para a camada de rede, pois existem tantas redes diferentes interconectadas.

O IPv4 é um protocolo de datagramas sem conexão e não confiável, ou seja, ele não provê mecanismos de controle de erros e fluxo, pois admite que as camadas inferiores tem

baixa confiabilidade, assim, tenta ao máximo entregar o pacote até o destino, mas sem garantias, o que podemos chamar de **serviço de entrega best-effort**. Os pacotes na camada de rede são denominados **datagramas**. O formato de um datagrama do IPv4 é mostrado na Figura 9.

Figura 9: Formato do datagrama IPv4



Fonte: FOROUZAN, 2009.

Indicação de Leitura

Para maiores detalhes dos campos do datagrama do IPv4, acessem a RFC 791.

RFC791 - INTERNET PROTOCOL - DARPA INTERNET PROGRAM - PROTOCOL SPECIFICATION. Disponível em: https://datatracker.ietf.org/doc/html/rfc791. Acesso em: 31 mai. 2022.

Um datagrama pode trafegar por várias redes diferentes. O formato e o tamanho do frame enviado dependem do protocolo usado pela camada física pela qual o frame vai trafegar. Se, por exemplo, um roteador interliga uma LAN a uma WAN, ele recebe um frame no formato da LAN e transmite um frame no formato da WAN, e isto ocorre em todos os pontos. Um dos campos definidos no formato é o tamanho máximo do campo de dados, para determinar o tamanho total que um frame consegue encapsular e enviar, isto é o que chamamos de **Unidade de Transferência Máxima (MTU).** Cada protocolo da camada de enlace

de dados tem seu próprio formato de frame, abaixo alguns valores de MTU para alguns protocolos: HyperChannel (65.535), FDDI (4.352), Ethernet (1.500) e PPP (296).

O comprimento máximo de um datagrama IPv4 é de 65.535 bytes, entretanto, muitas vezes é necessário dividir o datagrama, isso é denominado fragmentação. Quando um datagrama é fragmentado, cada fragmento possui seu próprio cabeçalho com a maioria dos campos repetidos, mas com alguns deles modificados. Um datagrama já fragmentado pode ser fragmentado outra vez, caso ele encontre uma rede com um MTU ainda menor. Em outras palavras, um datagrama pode ser fragmentado várias vezes antes de atingir seu destino final.

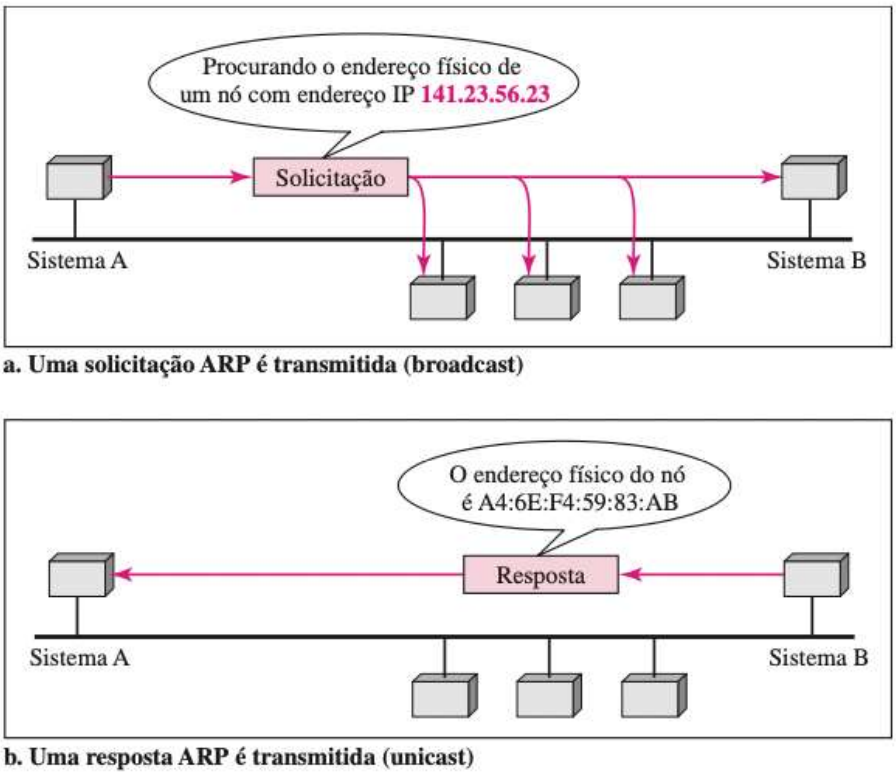
O host ou roteador que fragmenta um datagrama tem de alterar os valores de três campos: flags, offset de fragmentação e comprimento total. O restante dos campos deve ser copiado. Obviamente, o valor do *checksum* deve ser recalculado independentemente da fragmentação.

A ideia geral do checksum e como ele é calculado é a mesma que discutimos na Aula 3, e a o checksum no datagrama IPv4 segue os mesmos princípios. Primeiro, o valor do campo checksum é configurado em 0. Em seguida, o cabeçalho inteiro é dividido em seções de 16 bits e adicionados entre si. O resultado (soma) é complementado e inserido no campo checksum. O checksum no datagrama IPv4 cobre apenas o cabeçalho, e não todos os dados. Como o cabeçalho é alterado a cada roteador visitado, faz sentido ter um checksum, e o checksum dos dados são garantidos por camadas superiores.

Um ponto importante é o mapeamento, ou a construção da tabela de roteamento. Temos o **mapeamento estático**, que envolve a criação de uma tabela que associa o endereço lógico a um endereço físico. Essa tabela é armazenada em cada máquina da rede. Claro que toda vez que houvesse alteração nos dispositivos da rede seria necessário uma modificação da tabela, o que significa retrabalho e um potencial ponto de erro. Por sua vez, no mapeamento dinâmico, cada vez que uma máquina identifica um dos dois endereços, ela pede auxílio a um protocolo auxiliar: o protocolo **ARP (Address Resolution Protocol)**.

Todos os hosts ou roteadores da rede recebem e processam o pacote ARP Request, mas apenas o receptor pretendido reconhece seu endereço IP e responde com um pacote ARP Reply (Resposta ARP). O pacote de resposta contém os endereços IP e físico do receptor. O pacote é transmitido (em unicast) diretamente ao solicitante, usando o endereço físico recebido no pacote de solicitação.

Figura 10: Operação ARP



Fonte: FOROUZAN, 2009.

Indicação de Leitura

Para maiores detalhes dos campos do datagrama do IPv4, acessem a RFC 791. Disponível em: https://datatracker.ietf.org/doc/html/rfc6747. Acesso em: 31 mai. 2022.

Indicação de Vídeo

Veja uma explicação do funcionamento do protocolo ARP. Disponível em: https://www.youtube.com/watch?v=cn8Zxh9bPio. Acesso em: 31 mai. 2022.

Outro aspecto da camada de rede, que precisou de um protocolo auxiliar, é o controle de notificações e consultas. Para isto, foi desenvolvido o **ICMP (*Internet Control Message Protocol*)**. As mensagens ICMP dividem-se em duas grandes categorias: mensagens de notificação de erro e mensagens de consulta.

As mensagens de notificação de erros informam problemas que um roteador ou host (destino) podem vir a encontrar ao processar um pacote IP. Já as mensagens de consulta, que ocorrem aos pares, ajudam os administradores de redes ou de hosts a obterem informações específicas sobre um roteador ou outro host. Um exemplo é a descoberta dos vizinhos, ou descobrir informações sobre o roteador.

Uma das principais responsabilidades do ICMP é notificar erros ocorridos durante o processo de roteamento de datagramas IP. São tratados cinco tipos de erros: *destination unreachable* (destino inalcançável), *source quench* (contenção da fonte), *time-exceeded* (tempo esgotado), *parameter problems* (problemas de parâmetros) e *redirection* (redirecionamento).

Indicação de Leitura

Para maiores detalhes do funcionamento do protocolo ICMP acesse a RFC 792. Disponível em: https://datatracker.ietf.org/doc/html/rfc792. Acesso em: 31 mai. 2022.

Por fim, outro protocolo auxiliar na camada de rede é o **IGMP (*Internet Group Management Protocol*)**, responsável por tratar da comunicação multicast. A comunicação multicast é quando queremos enviar uma mesma mensagem, para mais de um receptor, ou seja, queremos criar grupos. E é exatamente esse o papel do IGMP, ele auxilia no envio de informações para que os roteadores criem tabelas de roteamento capazes de associar grupos.

Indicação de Leitura

Para maiores detalhes do funcionamento do protocolo IGMPv3 acesse a RFC 3376. Disponível em: https://datatracker.ietf.org/doc/html/rfc3376. Acesso em: 31 mai. 2022.

Com isto, encerramos a aula 05. Nesta aula, vimos as funções da camada de rede como o endereçamento lógico, fragmentação dos pacotes, roteamento e alguns protocolos auxiliares. Estudamos também que o protocolo IPv4 gradativamente está migrando para o IPv6, que possui diversas vantagens. Inclusive, em relação aos protocolos acessórios, as funcionalidades foram incorporadas ao ICMPv6. Ele será responsável pelas funções do ARP e do IGMP também. A camada de redes é responsável também pelo roteamento, mas não é escopo do curso passar pelos algoritmos de roteamento, se você tiver curiosidade, existe muita informação e como sempre, a maioria está padronizada nas RFCs.

Videoaula 3

Utilize o QR Code para assistir!

Agora, assista ao vídeo em que o professor explica com mais detalhes os conceitos finais da camada de redes.

A Camada de Transporte

Se a camada de rede é responsável pelo endereçamento e roteamento, mas não garante alguns aspectos relacionados à confiabilidade do canal, a camada de transporte é responsável pela comunicação entre processos finais de uma mensagem inteira. Na camada de transporte da Internet, temos três protocolos: UDP (*User Datagram Protocol*), TCP (*Transmission Control Protocol*) e SCTP (*//*). O UDP é mais simples, não confiável e não orientado a conexão, o TCP por sua vez é o contrário. Já o SCTP é novo, e tenta ser uma união dos pontos positivos dos dois protocolos.

Mas afinal, o que é um processo? Tratamos processo como uma aplicação em execução em um host. Assim, a camada de transporte é responsável pela comunicação entre processos finais de uma mensagem inteira. Como a camada de rede não pressupõe o relacionamento entre os pacotes, a camada responsável por garantir a integridade e a ordem de entrega dos pacotes de uma mensagem inteira, controlando erros na transmissão, bem como o fluxo de dados. O processo de comunicação funciona numa relação **cliente/servidor.**

O funcionamento de uma comunicação **cliente/servidor** é basicamente um processo em execução no cliente que solicita acesso a algum recurso ou serviço de um outro processo que está executando no **servidor.** Quando estamos navegando na web, estamos usando esse paradigma, pois nosso navegador é o cliente que solicita acesso a um recurso, no caso alguma página, que está hospedado no servidor. Para fazer o mapeamento entre os processos, a camada de transporte utiliza portas. Os números de porta são inteiros de 16 bits, com valores entre 0 e 65.535.

Um programa cliente define para si mesmo um número de porta, escolhido de forma aleatória pelo software da camada de transporte, em execução no host cliente. Este é denominado número de **porta efêmero**. Do lado do servidor, não podemos fazer o mesmo, senão seria muito difícil e trabalhoso saber a porta que queremos acessar. Desta forma, no servidor, a aplicação será executada em portas específicas, possibilitando que os clientes utilizem sempre o mesmo valor de porta. No exemplo acessar uma página web, o servidor web geralmente executa na porta 80, por isto, não importa muito a porta do cliente ter o mesmo valor, mas a porta do serviço que queremos consumir, este sim é fundamental sempre ser o mesmo. E claro, a **IANA (*Internet Assigned Number Authority*)** padronizou esses valores.

Indicação de Leitura

Acesse a listagem dos valores das portas e seus serviços. Disponível em: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. Acesso em: 01 jun. 2022.

Basicamente temos três grupos:

- **Portas conhecidas** - As portas na faixa de 0 a 1023 são atribuídas e controladas pela IANA. Estas são as portas conhecidas (*well-known port numbers*).

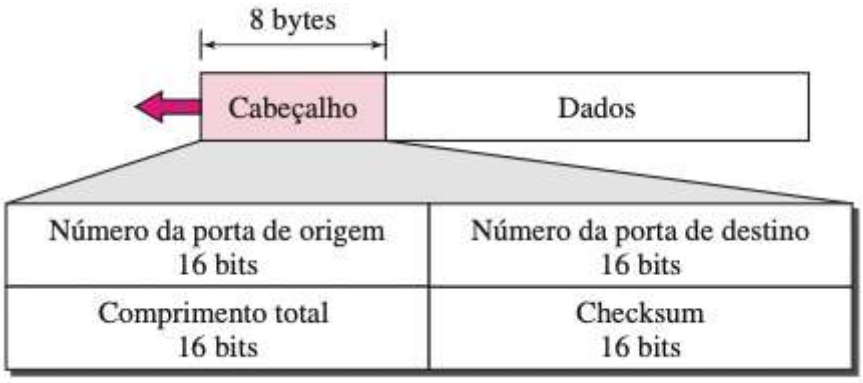
- **Portas registradas** - As portas na faixa de 1024 a 49151 não são atribuídas ou controladas pela IANA. Elas podem ser registradas na IANA para impedir duplicação.
- **Portas dinâmicas** - As portas na faixa de 49152 a 65535 não são controladas nem registradas. Elas podem ser usadas por qualquer processo. Estas são denominadas portas efêmeras.

O endereço socket é formado por um **endereço IP** e **um número de porta**, e o protocolo na camada de transporte precisa do endereço socket do cliente e do servidor.

UDP (User Datagram Protocol)

O primeiro protocolo que vamos discutir é o **UDP (*User Datagram Protocol*)** é um protocolo de transporte **sem conexão** e **não-confiável**, tornando ele simples pois não temos um controle host a host e uma verificação de erros limitada. Por causa da sua simplicidade, ele é mais rápido que o TCP, justamente por não possuir o overhead de controle e confiabilidade. O cabeçalho do datagrama de um UDP é apresentado na Figura 11.

Figura 11: Formato de um datagrama UDP.



Fonte: FOROUZAN, 2009.

Os pacotes UDP possuem um cabeçalho de 8 bytes, formado pelos campos:

- **Porta de origem:** Esse campo especifica o número da porta usada pelo processo em execução no host de origem.
- **Porta de destino:** Esse campo especifica o número da porta usado pelo processo em execução no host de destino.
- **Comprimento:** Esse campo de 16 bits define o comprimento total de um datagrama UDP, compreendendo cabeçalho mais dados. Deve ser menor do que o tamanho total, uma vez que vamos encapsular ele no datagrama IP, que já aceita um tamanho total de 65.535 bytes.
- **Checksum:** Esse campo de 16 bits é usado para detectar erros na transmissão de datagrama UDP (cabeçalho mais dados).

Como já mencionado, o UDP trabalha com serviços de transporte sem conexão, mas o que isso significa na prática? Significa que os datagramas enviados não possuem relação nenhuma entre eles, podem ser oriundos de um mesmo processo e irem para um mesmo destino, mas para o UDP, eles são independentes. Além disso, o UDP não estabelece um caminho virtual, com isto, cada datagrama pode percorrer caminhos diferentes até o destino.

Outro aspecto é que o UDP não é confiável, no sentido que ele não faz o controle do fluxo e erros, com isso, ele pode inundar o receptor com um número enorme de mensagens. E o único controle de erros é o checksum, tirando isso, ele não garante se a mensagem chegou ao destino, se ela é duplicada, não tem nenhum meio de controle de fluxo e de erros. Basicamente, ele delega isso para que a aplicação trate dessas questões.

Sua utilização é adequada para comunicações mais simples, por exemplo, o protocolo **SNMP (*Simple Network Management Protocol*)** utiliza o UDP para o gerenciamento de redes, assim como o **RIP (*Routing Information Protocol*), DNS (*Domain Name Service*)**. Quando existe a necessidade de um fluxo de dados em tempo real, admitindo uma perda, o UDP é a escolha certa. Aplicações VoIP e jogos de computadores fazem uso do UDP, pois é melhor receber o vídeo e voz, garantindo a comunicação, mesmo que isso custe perder um pouco a qualidade.

Videoaula 1

Utilize o QR Code para assistir!

Agora, assista ao vídeo em que o professor aborda o protocolo UDP.

TCP (*Transport Control Protocol*)

Se o UDP não possui controle de erros e fluxos, temos a sua antítese através do TCP (*Transmission Control Protocol*). Assim como o UDP, também é um protocolo de comunicação entre processos, usando as portas e os endereços IP para criar conexões sockets, a grande diferença é que o TCP implementa mecanismos de controle de fluxo e de erros, portanto, é um protocolo **orientado à conexão confiável.**

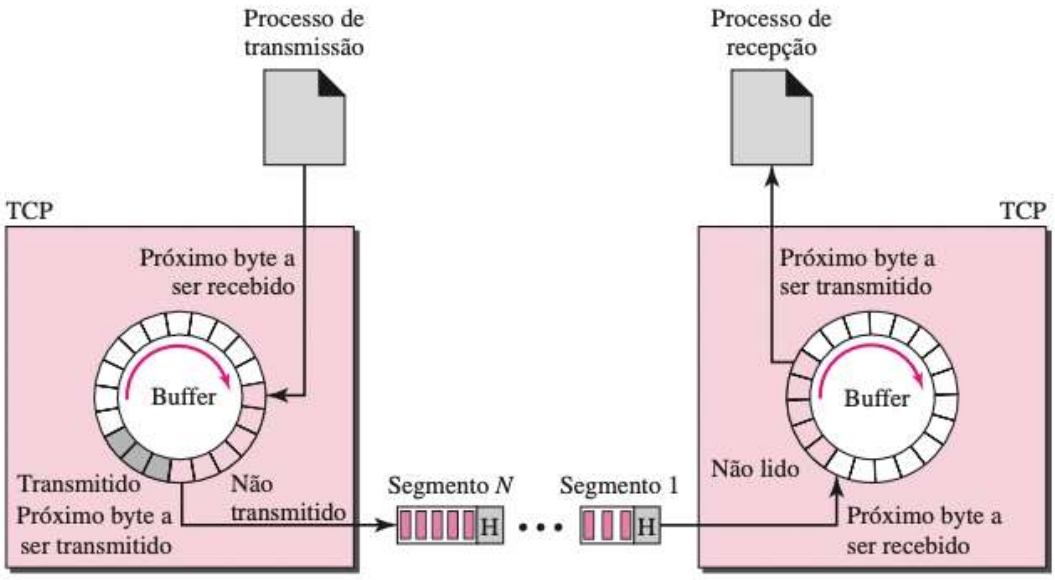
O conceito de ter um protocolo orientado à conexão, é como se o TCP conectasse transmissor e receptor através de um canal (lógico, e não físico), possibilitando a passagem de um fluxo de bytes. Diferentemente do UDP, que não estabelece relação entre os datagramas, o TCP cria esse ambiente, como ilustrado na Figura 12. O canal oferecido é **full-duplex**, o que significa que os dados podem fluir em ambas as direções.

Figura 12: Fluxo de dados no TCP.

Fonte: FOROUZAN, 2009.

Como a velocidade do envio dos dados pelo transmissor e a velocidade de consumo desses dados pelo receptor não é a mesma na maioria dos casos, o TCP trabalha com um buffer. Na verdade, utiliza dois buffers, um em cada ponta da transmissão. Inclusive, o tamanho do buffer pode variar entre o transmissor e receptor. Como o IP agrupa os dados em pacotes e não trabalha com um fluxo contínuo de bytes, o TCP faz esse trabalho de **segmentar** os bytes em **segmentos** antes de passá-los para a camada de rede. Com isto, temos a base para o controle de fluxo e erros. Agora é possível começar a controlar se os segmentos estão chegando, se perdermos algum segmento no caminho, e etc. A Figura 13 ilustra esse processo. Dois campos genéricos auxiliam nesse processo: **número de sequência** e **número de confirmação**.

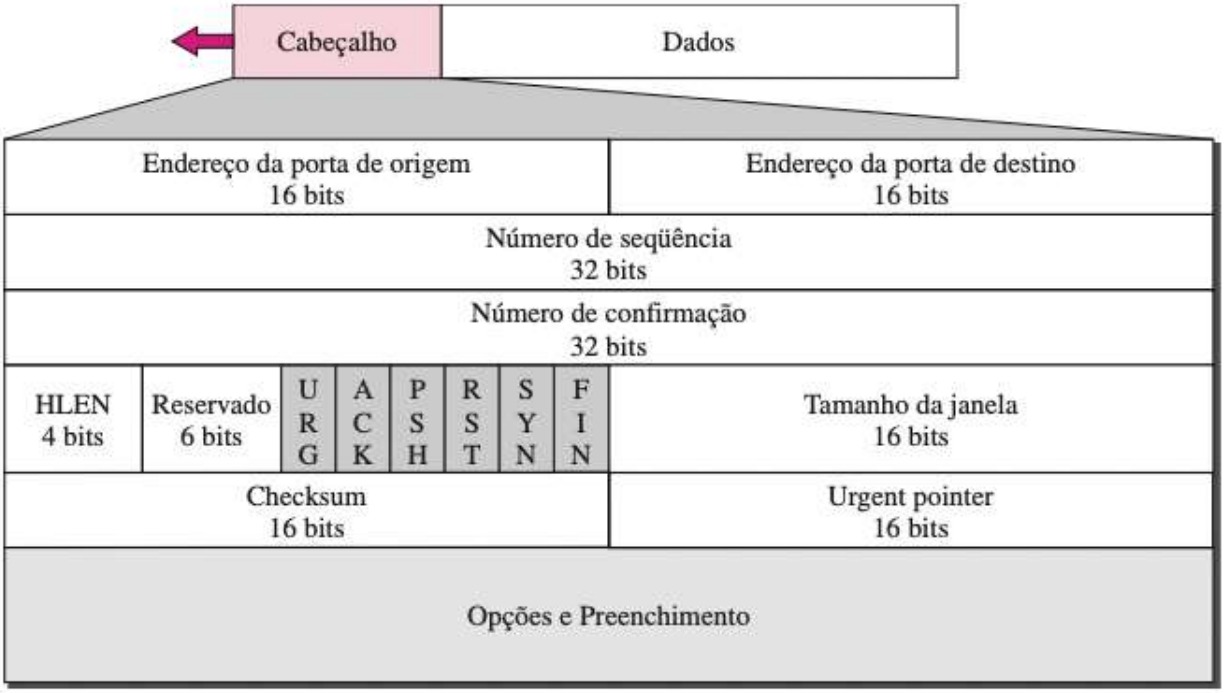
Figura 13: Fluxo de dados no TCP com buffer e segmentação



Fonte: FOROUZAN, 2009.

O TCP pode ser resumido em três passos importantes: estabelecer o canal; trocar os dados; encerrar a conexão. O TCP assume a responsabilidade de entregar na ordem correta os bytes para o outro lado da comunicação. A Figura 14 mostra o formato de um protocolo TCP.

Figura 14: Datagrama TCP



Fonte: FOROUZAN, 2009.

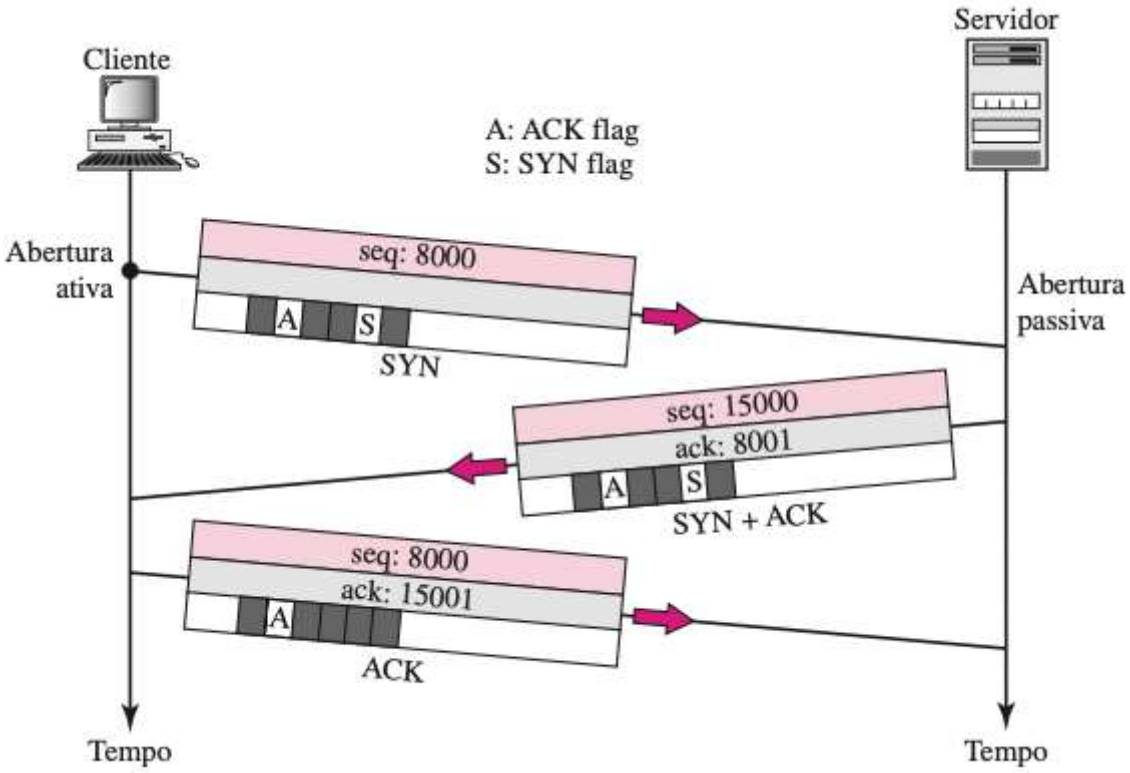
Um segmento TCP é formado por um cabeçalho de 20 a 60 bytes, seguido pelos dados. Os campos do segmento são:

- **Endereço da porta de origem:** Campo de 16 bits que identifica o número da porta do programa de aplicação no host que está enviando o segmento (origem);
- **Endereço da porta de destino:** Campo de 16 bits que identifica o número da porta do programa de aplicação no host que está recebendo o segmento (destino);
- **Número de sequência:** Campo de 32 bits identifica o número atribuído ao primeiro byte de dados de um segmento;
- **Número de confirmação:** Campo de 32 bits identifica o número de bytes que o receptor espera receber da outra parte. Se o receptor tiver recebido corretamente o número de bytes x da outra parte, ele define $x + 1$ como o número de confirmação;
- **Comprimento do cabeçalho (HLEN):** Esse campo de 4 bits identifica a quantidade de palavras de 4 bytes de um cabeçalho TCP. O comprimento do cabeçalho tem entre 20 e 60 bytes;
- **Reservado:** Trata-se de um campo de 6 bits reservado para uso futuro;
- **Controle:** São campos de controle (flags) distintos: URG, ACK, PSH, RST, SYN e FIN. Cada um é um bit, ou seja, o campo tem no total 6 bits;

- **Tamanho da janela:** Esse campo define o tamanho de uma janela TCP, em bytes, que a outra parte deve manter;
- **Checksum:** Campo de 16 bits contém o valor calculado do checksum;
- **Urgent Pointer:** Campo de 16 bits, que é válido apenas se o flag URG estiver ativo;
- **Opções**: Campo pode conter um total de até 40 bytes de informações opcionais que serão incluídas ao cabeçalho TCP.

Como o TCP estabelece uma conexão, o primeiro processo é o que chamamos de ***handshaking***, no caso é o ***three-way handshaking***. A Figura 15 ilustra esse processo.

Figura 15: *Three-way handshaking*



Fonte: FOROUZAN, 2009.

Tudo começa com o servidor informando ao TCP que está aberto a aceitar conexões, uma abertura passiva, mas ele não é capaz de estabelecer sozinho uma conexão. Para isto, o cliente envia uma solicitação de abertura, informando que deseja se conectar e isso inicia o processo do *three-way handshaking*. A ilustração da Figura 15 apresenta esse processo, e o datagrama está "resumido" por questões pedagógicas. As etapas são, segundo Forouzan, 2009:

1. O cliente transmite o primeiro segmento, um segmento SYN, no qual apenas o flag SYN é ativado. Esse segmento destina-se à sincronização dos números de sequência. Ele consome um número de sequência. Quando for iniciada a transferência de dados, o número de sequência será incrementado em 1. Podemos dizer que um segmento SYN

não transporta dados reais, porém podemos imaginá-lo como contendo 1 byte imaginário.

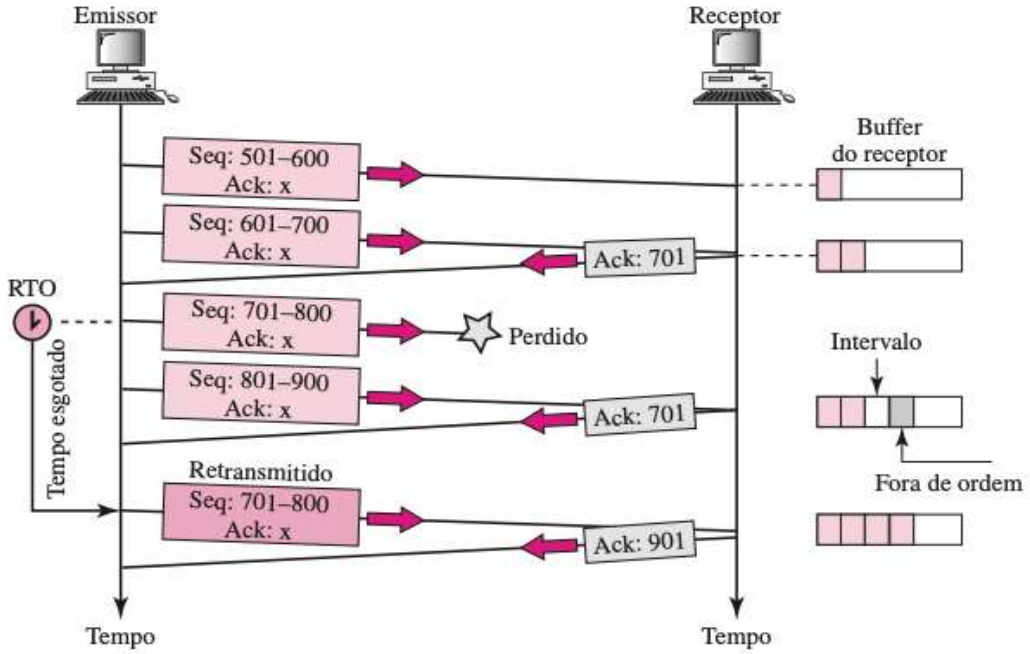
2. O servidor transmite o segundo segmento, um segmento SYN + ACK, com 2 bits de flags ativos: SYN e ACK. Esse segmento tem dupla finalidade. Ele é um segmento SYN para iniciar a comunicação na outra direção e serve como confirmação para o segmento SYN anterior do cliente. Ele consome um número de sequência.
3. O cliente transmite o terceiro segmento. Trata-se apenas de um segmento ACK. Ele confirma o recebimento do segundo segmento com o flag ACK e o campo número de confirmação. Note que o número de sequência nesse segmento é o mesmo que aquele no segmento SYN; o segmento ACK não consome um número de sequência.

Assim como ele faz o *three-way handshaking* para estabelecer a conexão, também acontece quando a conexão é encerrada. Geralmente parte do cliente uma mensagem com o FIN, o servidor retorna com um FIN + ACK e por fim, o cliente responde com um ACK, e assim a conexão é fechada. No TCP, também é possível um lado interromper a transmissão de dados enquanto ainda recebe dados, isso é denominado semiencerramento.

Para o controle de fluxo, o TCP utiliza a técnica de janela deslizante (*sliding window*), como discutido na camada de enlace, e possui algumas funcionalidades similares ao Go-Back-N e à Repetição Seletiva. Mas importante lembrar que cada camada trata coisas diferentes, na camada de enlace tratamos quadros, enquanto que na camada de transporte, tratamos de datagramas. Por isso mesmo, no TCP as janelas deslizantes são de tamanho variável; enquanto que na camada de enlace de dados são de tamanho fixo.

Além do controle de fluxo, o TCP é um protocolo confiável, isso significa que ele implementa controle para detecção de segmentos corrompidos, perdidos ou fora de ordem e segmentos duplicados. Para isto, o utiliza: checksum, confirmação de recebimento e *time-out*. Como já discutido em outras aulas, o checksum é usado para validar o segmento, e é um campo obrigatório em todos os segmentos. A confirmação é outro ponto importante, apenas um segmento ACK não necessita de confirmação. Por fim, o *time-out* que é quando o tempo de retransmissão se esgota ou quando o emissor recebe três ACKs duplicados, ele é retransmitido. A Figura 16 mostra a retransmissão de um segmento após o *time-out*.

Figura 16: Retransmissão após segmento perdido



Fonte: FOROUZAN, 2009.

Videoaula 2

Utilize o QR Code para assistir!

Agora, assista ao vídeo em que o professor aborda o protocolo TCP.

SCTP (Stream Control Transmission Protocol)

Com o avanço da tecnologia, novas aplicações que antes eram improváveis foram possíveis de ser implementadas como a telefonia via IP, serviços de streaming e outros. Isto fez surgir novos protocolos na camada de aplicação que precisavam do melhor dos dois mundos na camada de transporte. Ou seja, um protocolo confiável e orientado a mensagens. O UDP é um protocolo orientado a mensagens, e cada mensagem é independente da outra, esta é uma característica interessante para aplicações em tempo real. Por outro lado, ele precisa que seja confiável com controle de fluxo e erros também. O SCTP possui alguns recursos inovadores não existentes no UDP e TCP.

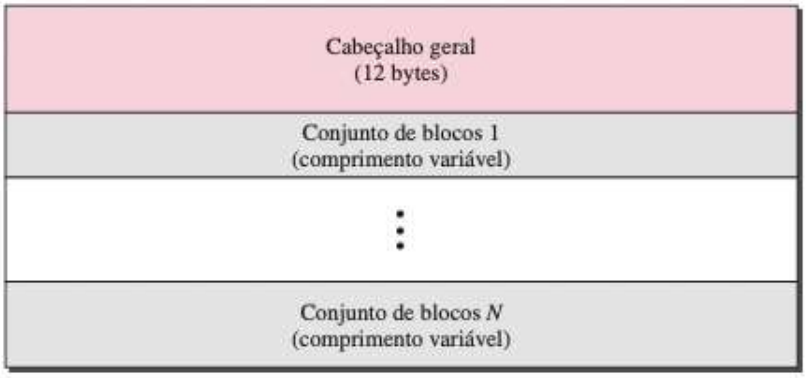
O SCTP também utiliza das portas para identificar a conexão entre os processos, algumas portas são: 9990 (IUA), 1718, 1719, 1720 e 11720 (H.323). Estas são algumas portas

utilizadas em telefonia IP. Temos a porta 2945 (H.298) relacionada à *Media gateway control* em conjunto com outros protocolos.

O SCTP trabalha com a ideia de ter múltiplos fluxos de dados em cada conexão, desta forma, quando um dos fluxos estiver bloqueado, ainda é possível continuar enviando os dados pelos outros fluxos. O conjunto de fluxos é chamado de **associação**, e ele trafega **pacotes**. Uma analogia à associação do SCTP são as pistas em uma rodovia, geralmente, temos a pista da esquerda para o tráfego de veículos mais rápidos, e as pistas à direita para veículos mais lentos. Se uma das pistas acabar interditada, as outras continuam funcionando do mesmo jeito. O SCTP implementa serviços de transporte *full-duplex*, no qual os dados podem trafegar em ambas as direções.

O SCTP possui dois cabeçalhos: um geral (obrigatório) e ***chunk***, e dentro dos *chunks* temos dois tipos também: controle e dados. Um bloco de controle controla e mantém a associação de blocos; um bloco de dados transporta dados de usuário. Em um pacote, conjuntos de blocos de controle precedem conjuntos de blocos de dados. O formato do pacote SCTP pode ser visualizado na Figura 17.

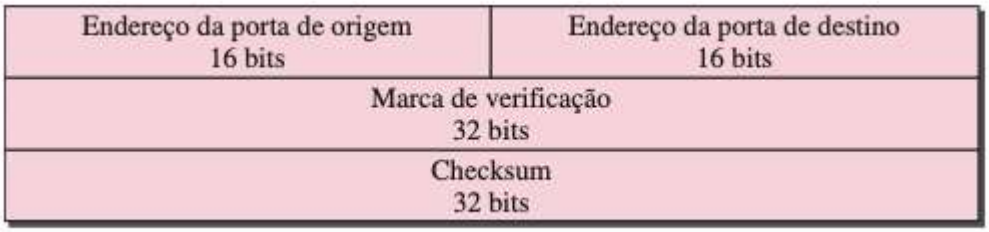
Figura 17: Formato do pacote SCTP



Fonte: FOROUZAN, 2009.

O cabeçalho geral, visto na Figura 18, define as extremidades (endpoints) de cada associação que um pacote pertence, garantindo que o pacote pertence a uma associação específica, preservando a integridade de seu conteúdo, inclusive do próprio cabeçalho.

Figura 18: Cabeçalho geral do SCTP



Fonte: FOROUZAN, 2009.

Existem quatro campos no cabeçalho geral:

- **Endereço da porta de origem:** o número da porta do processo que está enviado o pacote.
- **Endereço da porta de destino:** o número da porta do processo que receberá o pacote.
- **Marca de verificação (*Verification Tag*):** este é um número que correlaciona um pacote a uma associação, é repetido em todos os pacotes de uma associação. Existe um processo de verificação distinto para cada direção da associação.
- **Checksum**. Esse campo de 32 bits armazena o valor calculado do checksum CRC-32. Este tamanho é maior que o do UDP e TCP.

Informações de controle ou dados de usuário são transportados em conjuntos de blocos e, infelizmente, o detalhamento está fora do escopo deste material, mas a leitura suplementar abaixo é indicada para um aprofundamento mais técnico no protocolo.

Indicação de Leitura

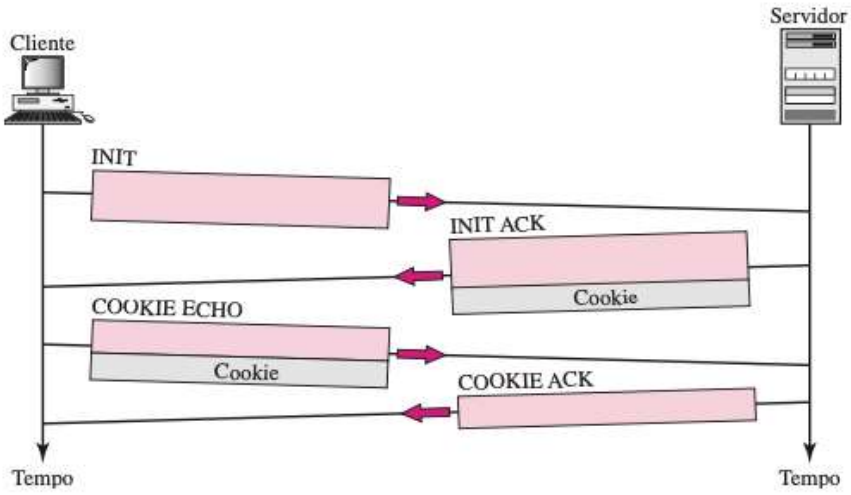
Acesse as seguintes RFCs para maior detalhamento do protocolo e seus serviços.

RFC2960 - Stream Control Transmission Protocol. Disponível em: https://datatracker.ietf.org/doc/html/rfc2960. Acesso em: 01 jun. 2022.

RFC3286 - An Introduction to the Stream Control Transmission Protocol (SCTP). Disponível em: https://datatracker.ietf.org/doc/html/rfc3286. Acesso em: 01 jun. 2022.

Para estabelecer uma associação no SCTP também precisa estabelecer uma conexão, igual o TCP, mas aqui ele é realizado em quatro vias, denominado ***four-way handshaking***. O cliente inicia o estabelecimento da associação e segue os passos da ilustração na Figura 19.

Figura 19: *Four-way handshaking*



Fonte: FOROUZAN, 2009.

Videoaula 3

Utilize o QR Code para assistir!

Agora, assista ao vídeo em que o professor aborda o protocolo SCTP e encerra essa unidade.

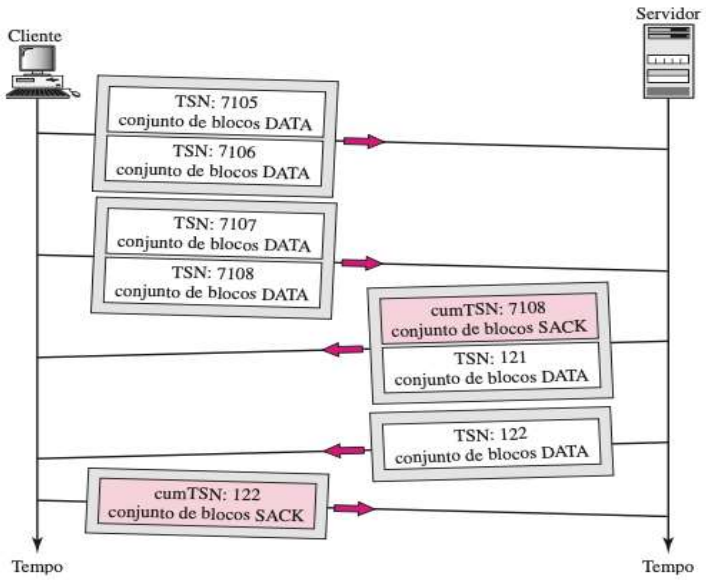
Os passos para o estabelecimento da conexão são (FOROUZAN, 2009):

1. O cliente transmite o primeiro pacote, contendo um conjunto de blocos INIT.
2. O servidor transmite o segundo pacote com sua resposta, contendo um conjunto de blocos INIT ACK.
3. O cliente transmite o terceiro pacote, incluindo um conjunto de blocos COOKIE ECHO. Este é um conjunto de blocos muito simples que ecoa, sem modificações, o cookie enviado pelo servidor. O SCTP possibilita a inclusão de conjuntos de blocos de dados nesse pacote.
4. O servidor transmite o quarto pacote, que inclui o conjunto de blocos COOKIE ACK, confirmando o recebimento do conjunto de blocos COOKIE ECHO. O SCTP permite a inclusão de conjuntos de blocos de dados simultaneamente com este pacote.

Uma vez estabelecida a associação, é possível realizar a transferência de dados entre bidirecional de dados entre os processos finais. O SCTP também suporta o *piggybacking*. A grande diferença é que na transmissão de dados, o SCTP funciona mais parecido com o UDP do

que com o TCP. Cada mensagem é única e inserida em um conjunto de blocos DATA, mas ao contrário do UDP, no SCTP existe um relacionamento entre os conjuntos de blocos.

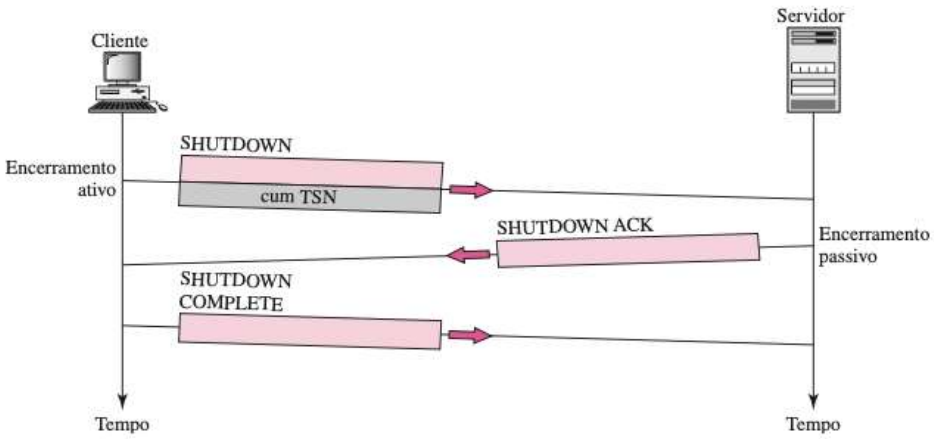
Figura 20: *Four-way handshaking*



Fonte: FOROUZAN, 2009.

O encerramento da conexão também é necessária, envolvendo a troca de mensagens do tipo SHUTDOWN, no caso três passos para encerrar uma associação. Se um lado encerrar a associação, o outro deve parar de transmitir imediatamente novos dados. Se ainda permanecerem dados na fila do receptor após o recebimento de uma solicitação de término, estes serão enviados e a associação será encerrada. Os passos para encerrar são vistos na Figura 21.

Figura 21: Encerramento de uma associação

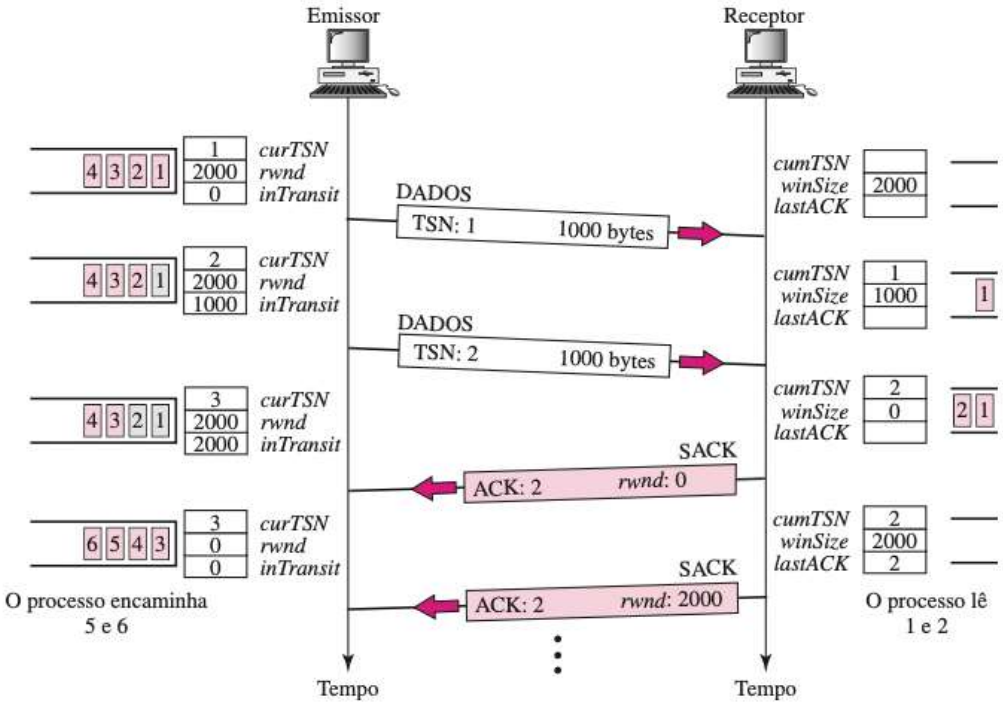


Fonte: FOROUZAN, 2009.

Assim como no TCP, o SCTP também possui um controle de fluxo utilizando a ideia de janelas deslizantes. Neste caso, a janela tem tamanho variável também, e para manter o

controle do fluxo, o SCTP faz uso das variáveis curTSN, rwnd e inTransit no emissor e cumTSN, winSize e lastACK no receptor. Como podemos perceber, é um trabalho mais complexo manter o fluxo das associações. A Figura 22 é mostra um ideia do controle.

Figura 22: Controle de fluxo no SCTP



Fonte: FOROUZAN, 2009.

O trecho retirado Forouzan de explica a Figura 22.

> Originalmente, existem quatro mensagens na fila do emissor. O emissor envia um bloco de dados e adiciona o número de bytes (1.000) à variável *inTransit*. Após alguns instantes, o emissor verifica a diferença entre *rwnd* e *inTransit*, que é de 1.000 bytes, de modo que ele seja capaz de enviar outro bloco de dados. Agora a diferença entre as duas variáveis é 0 e nenhum outro bloco de dados pode ser enviado. Após alguns instantes, chega um SACK que confirma os conjuntos de dados 1 e 2. Os dois conjuntos de blocos são eliminados da fila de transmissão. O valor de *inTransit* agora é 0. O SACK, entretanto, anunciou a janela de recepção com valor 0, que faz que o emissor atualize a variável *rwnd* para 0. Nesse caso, o emissor permanecerá bloqueado; ele não pode enviar outros conjuntos de dados (com uma exceção, que será explicada posteriormente).
>
> No lado do receptor, inicialmente, a fila de recepção está vazia. Após receber o primeiro bloco de dados, uma mensagem é armazenada na fila e o valor de cumTSN será igual a 1. O valor de winSize será subtraído de 1.000, pois a primeira mensagem ocupa 1.000 bytes. Após receber o segundo bloco de dados, o valor do tamanho da janela será 0 e cumTSN igual a 2. Agora, como veremos, o receptor envia um SACK com TSN acumulado igual a 2. Após o primeiro SACK ter sido enviado, o processo lê as duas mensagens, removendo-as da fila de recepção; o receptor anuncia essa situação com um SACK, permitindo ao emissor transmitir outros conjuntos de dados (FOROUZAN, 2009, p. 750-751).

O SCTP utiliza um conjunto diferente de comando e *timers* para manter o controle do protocolo e mantê-lo confiável. A retransmissão funciona de maneira similar ao TCP, usando timers ou a recepção de quatro SACKs. Ainda para o controle de erros, as regras também são similares às implementadas para o TCP.

Com isto, encerramos nossa discussão dos protocolos da camada de transporte. Vimos que temos serviços orientados e não orientados a conexão, bem como um serviço confiável e

não confiável. A camada de transporte é responsável pela ligação entre os processos, através de sockets, ou seja, o conjunto entre os endereços IP e as portas.

Encerramento da Unidade

Nesta unidade você aprendeu quais são os objetivos da camada de redes e transporte. Vimos que o IPv4 é um protocolo não confiável e sem conexão responsável pela entrega da origem até o destino. Os pacotes na camada IPv4 são denominados datagramas. Um datagrama é formado por um cabeçalho com um comprimento máximo de um datagrama é de 65.535 bytes. O MTU é o número máximo de bytes que um protocolo de enlace de dados pode encapsular e variam de protocolo para protocolo. Aprendemos que fragmentação é a divisão de um datagrama em unidades menores para suportar a compatibilização com o MTU de um protocolo de enlace de dados. Vimos que existem protocolos acessórios como o ICMP, ARP e IGMP para auxiliar na camada de rede, e que no IPv6, todos são substituídos pelo ICMPv6.

Além da camada de redes, estudamos a camada de transporte, responsável pela comunicação *end-to-end* e adicionamos o controle de fluxo e erros, por isso, geralmente chamamos de TCP/IP. Mas aprendemos também que temos os protocolos UDP e SCTP. O primeiro não tem controle de erros e trabalha com independência entre as mensagens, enquanto o segundo cria fluxos para trocas de informações. Mas todos provêm serviço para a camada superior, que veremos na próxima unidade, a camada de aplicação.

Espero vocês na próxima unidade!

Bom trabalho!


UniFil
EDUCAÇÃO A DISTÂNCIA
UNIFIL.BR