1ºcapítulo(Redes de computadores e a Internet)

1.1 O que é Internet?

1.1.1. *Uma descrição detalhada da rede*

-A internet é uma rede que conecta centenas de milhões de dispositivos ao redor do mundo, esses dispositivos são conhecidos como **Hospedeiros(Hosts)** ou **Sistemas Finais**(Ex.: TVs, laptops, geladeiras smart, console de jogo, smartphones...).
Os hosts se conectam através de **Enlaces de Comunicação** e **Comutadores de Pacotes** que realizam a transferência dos dados numa taxa denominada **Taxa de Transmissão** ou **Banda Passante**, e esse é medido em bits/s(bits por segundo).

-**Enlaces(de Comunicação)** são os “cabos” da rede, enquanto os **Comutadores(de Pacotes)** são dispositivos que encaminham o pacote que está chegando em um de seus enlaces de entrada para um de seus enlaces de saída. Os mais comuns são **roteadores** e **comutadores de camada de enlace**.

-A sequência de enlaces e comutadores que um pacote percorre desde o host remetente até o host receptor é conhecida como **Rota** ou **Caminho** através da rede.

-Sistemas finais acessam a Internet por meio de **Provedores de Serviços de Internet** (**ISPs**). Cada ISP é uma rede de comutadores e enlaces.

-Internet é uma “rede das redes” pois é composta de vários ISPs interconectados

- ISPs oferecem aos sistemas finais uma variedade de tipos de acesso à rede. ISPs também fornecem acesso a provedores de conteúdo, conectando sites diretamente à Internet. Esta se interessa pela conexão entre os sistemas finais, portanto os ISPs que fornecem acesso a esses sistemas também devem se interconectar. Esses ISPs de nível mais baixo são interconectados por meio de ISPs de nível mais alto

-Um ISP de nível mais alto consiste em roteadores de alta velocidade interconectados com enlaces de fibra óptica de alta velocidade

- Cada rede ISP, seja de nível mais alto ou mais baixo, é gerenciada de forma independente, executa o protocolo IP e obedece a certas convenções de nomeação e endereço.

-Os sistemas finais, os comutadores e outras peças da Internet executam protocolos que controlam o envio e o recebimento de informações. O **TCP** (Protocolo de Controle

de Transmissão) e o **IP** (Protocolo da Internet) são dois dos mais importantes da Internet

-O protocolo IP especifica o formato dos pacotes que são enviados e recebidos entre roteadores e sistemas finais

-Os padrões da Internet são desenvolvidos pela IETF e os documentos padronizados dela são denominados RFCs(pedido de comentários).

1.1.2 *Uma descrição do serviço*

A internet também pode ser descrita como uma infraestrutura que provê serviços à **Aplicações Distribuídas**, que são aplicações nas quais dois ou mais sistemas finais trocam informações mutuamente(Ex.: WhatsApp, e-mail, navegação web, vídeo em tempo real). Essas aplicações são executadas somente por sistemas finais, e nunca em comutadores.
A **API(Interface de Programação de Aplicação)** da internet é um conjunto de regras que o software emissor de dados deve seguir para que a internet envie corretamente os dados para o software receptor.

1.1.3 *O que é um protocolo?*

-Um **Protocolo** define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento.

1.2 A periferia da Internet

1.2.1 *Redes de acesso*

-Às vezes, sistemas finais são ainda subdivididos em duas categorias: clientes e servidores. Informalmente, clientes costumam ser PCs de mesa ou portáteis, smartphones e assim por diante, ao passo que servidores tendem a ser máquinas mais poderosas, que armazenam e distribuem páginas Web, vídeo em tempo real, retransmissão de e-mails e assim por diante.

-O 1º roteador que conecta um host à internet também é conhecido como **Roteador de Borda**.

-Os dois tipos de acesso residencial banda largas predominantes são a linha digital de assinante (**DSL**) ou a cabo.

-Normalmente uma residência obtém acesso DSL à Internet da mesma empresa que fornece acesso telefônico local com fio.

-o modem DSL de cada cliente utiliza a linha telefônica existente (par de fios de cobre trançado) para trocar dados com um multiplexador digital de acesso à linha do assinante (DSLAM). O modem DSL da casa apanha dados digitais e os traduz para sons de alta frequência, para transmissão pelos fios de telefone.

-A linha telefônica conduz, simultaneamente, dados e sinais telefônicos tradicionais, que são codificados em frequências diferentes:
• um canal downstream de alta velocidade, com uma banda de 50 kHz a 1 MHZ; (banda passante <24Mbps)
• um canal upstream de velocidade média, com uma banda de 4 kHz a 50 kHz; (banda passante <2,5Mbps)
• um canal de telefone bidirecional comum, com uma banda de 0 a 4 kHz.
Essa abordagem faz que a conexão DSL pareça três conexões distintas, de modo que um telefonema e a conexão com a Internet podem compartilhar a DSL ao mesmo tempo.

-O acesso à Internet a cabo utiliza a infraestrutura de TV a cabo da operadora de televisão.

-as fibras óticas conectam o terminal de distribuição às junções da região, sendo o cabo coaxial tradicional utilizado para chegar às casas e apartamentos de maneira individual. Em razão de a fibra e o cabo coaxial fazerem parte desse sistema, a rede é denominada híbrida fibra-coaxial (HFC).

-Os modems a cabo dividem a rede HFC em dois canais, um de transmissão (downstream) e um de recebimento (upstream). Como a tecnologia DSL, o acesso costuma ser assimétrico, com o canal downstream recebendo uma taxa de transmissão maior do que a do canal upstream.(42,8Mbits/s de downstream e 30,7Mbits/s de upstream)

-Uma característica importante do acesso a cabo é o fato de ser um meio de transmissão compartilhado. Em especial, cada pacote enviado pelo terminal viaja pelos enlaces downstream até cada residência e cada pacote enviado por uma residência percorre o canal upstream até o terminal de transmissão. Por essa razão, se diversos usuários estiverem fazendo o download de um arquivo em vídeo ao mesmo tempo no canal downstream, cada um receberá o arquivo a uma taxa bem menor do que a taxa de transmissão a cabo agregada. Por outro lado, se há somente alguns usuários ativos navegando, então cada um poderá receber páginas da Web a uma taxa de downstream máxima, pois esses usuários raramente solicitarão uma página ao mesmo tempo. Como o canal upstream também é compartilhado, é necessário um protocolo de acesso múltiplo distribuído para coordenar as transmissões e evitar colisões.

-A Ethernet é mais tipicamente usada em empresas, universidades, etc...possui uma banda passante de 100Mbits/s para usuários e de 1~10Gbits/s para servidores. atualmente os hosts se conectam a switches de Ethernet

-REDES DE ACESSO SEM FIO(WIP)

1.2.2 *Meios físicos*

-Os meios físicos se enquadram em duas categorias: meios guiados e meios não guiados. Nos meios guiados, as ondas são dirigidas ao longo de um meio sólido, tal como um cabo de fibra ótica, um par de fios de cobre trançado ou um cabo coaxial. Nos meios não guiados, as ondas se propagam na atmosfera e no espaço, como é o caso de uma LAN sem fio ou de um canal digital de satélite.

Par de fios de cobre trançado

-O meio de transmissão guiado mais barato e mais usado é o par de fios de cobre trançado, os fios são trançados para reduzir a interferência elétrica de pares semelhantes que estejam próximos.

-Um par de fios constitui um único enlace de comunicação.

-Hoje, as taxas de transmissão de dados para as LANs de pares trançados estão na faixa de 10 Mbits/s a 10 Gbits/s.

Cabo coaxial

-Como o par trançado, o cabo coaxial é constituído de dois condutores de cobre, porém concêntricos e não paralelos.

-bidirecional

-O cabo coaxial pode ser utilizado como um meio compartilhado guiado. Vários sistemas finais podem ser conectados diretamente ao cabo, e todos eles recebem qualquer sinal que seja enviado pelos outros sistemas finais.

Fibras ópticas

-A fibra ótica é um meio delgado e flexível que conduz pulsos de luz, cada um deles representando um bit.

-Uma única fibra ótica pode suportar taxas de transmissão elevadíssimas, de até dezenas ou mesmo centenas de gigabits por segundo.

-Fibras óticas são imunes à interferência eletromagnética, têm baixíssima atenuação de sinal até cem quilômetros.

-o alto custo de equipamentos ópticos vem impedindo sua utilização para transporte a curta distância.

Canais de rádio terrestres

-Canais de rádio carregam sinais dentro do espectro eletromagnético. São um meio atraente porque sua instalação não requer cabos físicos, podem atravessar paredes, dão conectividade ao usuário móvel e, potencialmente, conseguem transmitir um sinal a longas distâncias.

-sujeito a efeitos de propagação por:

- reflexão
- interferência
- obstáculos que fazem o sinal ficar mais fraco

-Canais de rádio terrestres podem ser classificados, de modo geral, em três grupos: os que operam sobre distâncias muito curtas(1 a 2 metros), os de pequeno alcance(algumas dezenas de metros), e os de longo alcance(dezenas de quilômetros)

Canais de rádio por satélite

-canais de Kbps a 45Mbps (ou múltiplos canais menores)

-270 ms de atraso fim-a-fim

-geoestacionários versus de baixa altitude

1.3 O Núcleo Da Rede

-Há duas abordagens fundamentais para locomoção de dados através de uma rede de enlaces e comutadores: **Comutação de Circuitos** e **Comutação de Pacotes**.

1.3.1 *Comutação de pacotes*

-Em uma aplicação de rede, sistemas finais trocam mensagens entre si. Mensagens podem conter qualquer coisa que o projetista do protocolo queira. Podem desempenhar uma função de controle ou conter dados, tal como um e-mail, uma imagem ou um arquivo de áudio.

-Antes de um sistema final enviar dados, esse segmenta os dados e adiciona Bytes de cabeçalho a cada segmento. Os pacotes de informações(ou apenas chamados

Pacotes) resultantes, são enviados através da rede ao sistema final de destino, onde são remontados para os dados originais.

-Os comutadores demoram um tempo de L/R segundos para transmitir o pacote pro enlace, sendo L o tamanho do pacote em bits e R a taxa de transmissão em bits/s.

Transmissão armazena-e-reenvia

-A maioria dos comutadores utilizam a transmissão **Armazena-e-Reenvia** (store-and-forward) nas entradas dos enlaces. A transmissão armazena-e-reenvia significa que o comutador deve receber o pacote inteiro antes de poder começar a transmitir o primeiro bit para o enlace de saída.

-O roteador irá armazenar os bits do pacote que chegam no enlace de chegada em buffers, para que quando o pacote chegue por completo ele possa ser reenviado pelo enlace de saída.

Atraso de fila e perda de pacote

-Além dos atrasos de armazenagem e reenvio, os pacotes sofrem **Atrasos de Fila**. Se um pacote que está chegando precisa ser transmitido por um enlace, mas o encontra ocupado com a transmissão de outro pacote, deve aguardar no buffer.

-Como o espaço do buffer é finito, um pacote que está chegando pode encontrá-lo lotado de outros que estão esperando transmissão. Nesse caso, ocorrerá uma **Perda de Pacote**(um pacote que está chegando ou um dos que já estão na fila é descartado).

Tabelas de Repasse e Protocolos de Roteamento

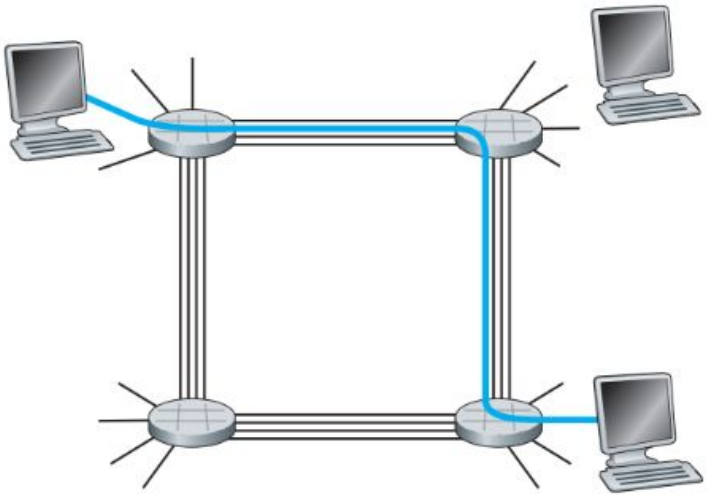
-Na Internet, cada sistema final tem um endereço denominado endereço IP. Quando um sistema final de origem envia um pacote a um destino, a origem inclui o endereço IP do destino no cabeçalho do pacote. Quando um pacote chega a um roteador, este examina o endereço e pesquisa sua **Tabela de Encaminhamento**, utilizando esse endereço de destino para encontrar o enlace de saída apropriado. O roteador, então, direciona o pacote a esse enlace de saída.

-**Tabela de Encaminhamento** são tabelas que o roteador utiliza para decidir para qual enlace de saída o pacote recebido será enviado.

-a Internet possui uma série de protocolos de roteamento especiais, que são utilizados para configurar automaticamente as tabelas de encaminhamento. Um protocolo de roteamento pode, por exemplo, determinar o caminho mais curto de cada roteador a cada destino e utilizar os resultados para configurar as tabelas de encaminhamento nos roteadores.

1.3.2 *Comutação de circuitos*

-Nessas redes, quando dois sistemas finais querem se comunicar, a rede estabelece uma conexão fim a fim dedicada entre os dois hospedeiros, uma conexão forte, na qual os comutadores no caminho entre o remetente e o destinatário mantêm o estado(essa conexão é denominada **Circuito**).



-Quando a rede estabelece o circuito, também reserva uma taxa de transmissão constante nos enlaces da rede durante o período da conexão. Visto que foi reservada largura de banda para essa conexão remetente-destinatário, o remetente pode transferir dados ao destinatário a uma taxa constante garantida.

-Em comutação de pacotes, tais recursos não são reservados; as mensagens de uma sessão usam os recursos por demanda e, como consequência, poderão ter de esperar (isto é, entrar na fila) para conseguir acesso a um enlace de comunicação.

Analogia entre comutação de pacotes e comutação de circuitos:
Como simples analogia, considere dois restaurantes — um que exige e outro que não exige nem aceita reserva. Se quisermos ir ao restaurante que exige reserva, teremos de passar pelo aborrecimento de telefonar antes de sair de casa. Mas, quando chegarmos lá, poderemos, em princípio, ser logo atendidos e servidos. No segundo restaurante, não precisaremos nos dar ao trabalho de reservar mesa, porém, quando lá chegarmos, talvez tenhamos de esperar para sentar.

Multiplexação em Redes de Comutação de Circuitos

-Um circuito é implementado em um enlace por multiplexação por divisão de frequência (**FDM**) ou por multiplexação por divisão de tempo (**TDM**).

-Em FDM, o espectro de frequência de um enlace é compartilhado entre as conexões estabelecidas através desse enlace. Ou seja, o enlace reserva uma **Largura de Banda** de frequência para cada conexão durante o período da ligação.

-Em TDM, o tempo é dividido em quadros de duração fixa, e cada quadro é dividido em um número fixo de compartimentos. Quando estabelece uma conexão por meio de um enlace, a rede dedica à conexão um compartimento de tempo em cada quadro. Esses compartimentos são reservados para o uso exclusivo dessa conexão, e um dos compartimentos de tempo (em cada quadro) fica disponível para transmitir os dados dela.

-Os defensores da comutação de pacotes sempre argumentaram que comutação de circuitos é desperdício, porque os circuitos dedicados ficam ociosos durante períodos de silêncio. Por exemplo, um radiologista que usa uma rede de comutação de circuitos para acessar remotamente uma série de exames. Ele estabelece uma conexão, requisita uma imagem, examina-a e, em seguida, solicita uma nova. Recursos de rede são atribuídos à conexão, mas não utilizados (isto é, são desperdiçados) no período em que o radiologista examina a imagem.

Comutação de Pacotes VERSUS Comutação de Circuitos

-Comutação de Pacotes oferece um atraso fim a fim variável e imprevisível, assim se tornando inadequado para serviços de tempo real. Porém, oferece melhor compartilhamento de banda do que Comutação de Circuitos e possui uma implementação mais simples, eficiente e barata.

-Sabendo que os usuários da rede não ficam 100% do tempo ativos, a comutação de pacotes pode oferecer o mesmo serviço que a comutação de serviços e para mais pessoas, pois a probabilidade de haver usuários ativos suficientes para congestionar a rede é baixa(isso pode ser provado estatisticamente).

EXEMPLO 1:

Suponha que haja dez usuários e que um deles de repente gere mil pacotes de mil bits, enquanto os outros nove permanecem inativos e não geram pacotes. Com comutação de circuitos TDM de dez compartimentos de tempo por quadro, e cada quadro consistindo em mil bits, o usuário ativo poderá usar somente seu único compartimento por quadro para transmitir dados, enquanto os nove compartimentos restantes em cada quadro continuarão ociosos. Dez segundos se passarão antes que todo o milhão de bits de dados do usuário ativo seja transmitido. No caso da comutação de pacotes, o usuário ativo poderá enviá-los continuamente à taxa total de 1 Mbit/s, visto que não haverá outros gerando pacotes que precisem ser multiplexados com os dele. Nesse caso, todos os dados do usuário ativo serão transmitidos dentro de 1 segundo.

EXEMPLO 2:
Suponha que usuários compartilhem um enlace de 1 Mbit/s, e que cada usuário fique ativo apenas durante 10% do tempo(e ocioso nos outros 90%). Com comutação de circuitos, devem ser reservados 100 kbits/s para cada usuário durante todo o tempo. Desse modo, o enlace de comutação de circuitos pode suportar somente 10 (1 Mbit/s/100 kbits/s) usuários simultaneamente. Com a comutação de pacotes, uma rede com 35 usuários, a probabilidade de haver 11 ou mais usuários ativos ao mesmo tempo é aproximadamente 0,0004. Quando houver dez ou menos usuários ativos simultâneos (probabilidade de 0,9996), a taxa agregada de chegada de dados é menor ou igual a 1 Mbit/s, que é a taxa de saída do enlace. Assim, quando houver dez ou menos usuários ativos, pacotes fluirão pelo enlace sem atraso, como é o caso na comutação de circuitos. Quando houver mais de dez usuários ativos ao mesmo tempo, a taxa agregada de chegada de pacotes excederá a capacidade de saída do enlace, e a fila de saída começará a crescer. Como a probabilidade de haver mais de dez usuários ativos é ínfima nesse exemplo, a comutação de pacotes apresenta, o mesmo desempenho da comutação de circuitos, mas o faz para mais de três vezes o número de usuários.

-Por esses e outros motivos, a internet utiliza quase que exclusivamente **Comutação de Pacotes.**

1.3.3. *Uma rede de redes*

-Com o passar dos anos, a rede de redes que forma a Internet evoluiu para uma estrutura bastante complexa. Para entender essa estrutura de rede da Internet de hoje, vamos criar, de modo incremental, uma série de estruturas de rede, com cada nova estrutura sendo uma aproximação melhor da Internet que temos.

-Lembrando que os hosts se conectam à internet através de ISPs.

-Essa parte não estará idêntica ao livro, fiz como achei mais entendível.

-Estrutura de Rede 1:
interconecta todos os **ISPs de acesso** a um único **ISP de trânsito global**. Nosso ISP de trânsito global é uma rede de roteadores e enlaces de comunicação que se espalham pelo planeta, e possuem ao menos um roteador próximo de cada uma das centenas de milhares de ISPs de acesso. Claro, seria muito dispendioso para o ISP de trânsito global montar essa rede tão extensa. Para que seja lucrativo, ele cobraria de cada um dos ISPs de acesso pela conectividade, com o preço refletindo a quantidade de tráfego que um ISP de acesso troca com o ISP global. Como o ISP de acesso paga ao ISP de trânsito global, ele é considerado um cliente, e o ISP de trânsito global é considerado um provedor.

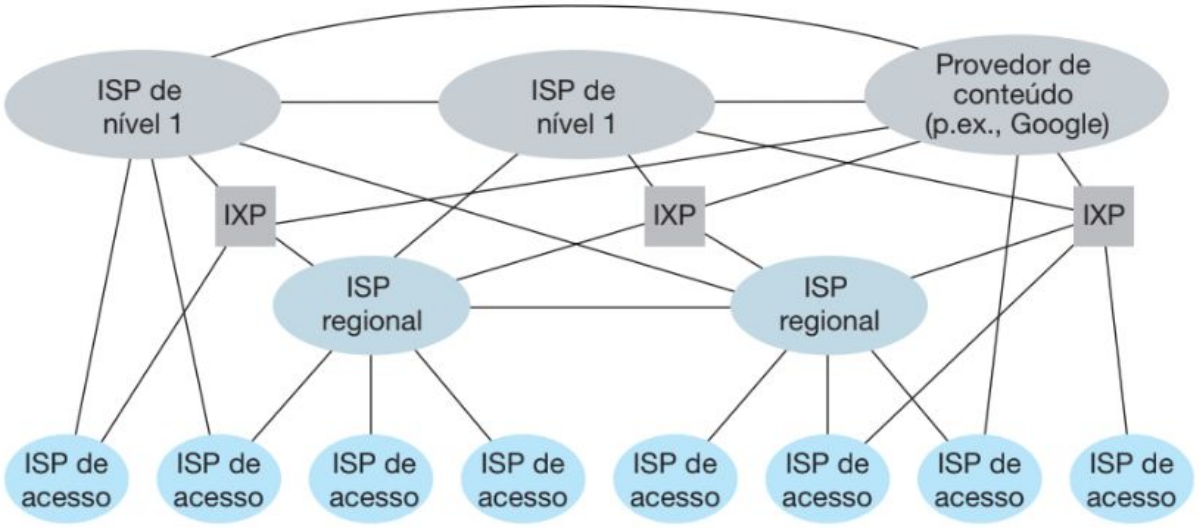
-Se alguma empresa montar e operar um ISP de trânsito global que seja lucrativo, então será natural para outras empresas montarem seus próprios ISPs de trânsito global e competirem com o original.

-Estrutura de Rede 2:
Consiste em centenas de milhares de ISPs de acesso e múltiplos ISPs de trânsito global. Note, porém, que os próprios ISPs de trânsito global precisam se interconectar: caso contrário, os ISPs de acesso conectados a um dos provedores de trânsito global não poderiam se comunicar com os ISPs de acesso conectados aos outros provedores de trânsito global.

-Estrutura de Rede 3:
Embora alguns ISPs tenham uma cobertura global impressionante e se conectem diretamente com muitos ISPs de acesso, nenhum tem presença em toda e qualquer cidade do mundo. Em vez disso, em determinada região, pode haver um **ISP Regional** ao qual os ISPs de acesso na região se conectam. Cada ISP regional, então, se conecta a **ISPs de nível 1**. Estes são semelhantes ao nosso ISP de trânsito global; mas os ISPs de nível 1, que realmente existem, não têm uma presença em cada cidade do mundo.
Também pode haver múltiplos ISPs regionais concorrentes em uma região. Em tal hierarquia, cada ISP de acesso paga ao regional ao qual se conecta, e cada ISP regional paga ao ISP de nível 1 ao qual se interliga. Assim, existe uma relação cliente-provedor em cada nível da hierarquia. Os ISPs de nível 1 não pagam a ninguém, pois estão no topo. Para complicar as coisas ainda mais, em algumas regiões pode haver um ISP regional maior (talvez se espalhando por um país inteiro) ao qual os ISPs regionais menores nessa região se conectam; o ISP regional maior, então, se conecta a um ISP de nível 1.

-Estrutura de Rede 4:
É a mesma estrutura da rede 3, mas com a adição de PoPs, IXPs, multi-home e emparelhamento.
Um **PoP(ponto de presença)** é simplesmente um grupo de um ou mais roteadores (no mesmo local) na rede do provedor, onde os ISPs clientes podem se conectar no ISP provedor.
Multi-Home é quando um ISP cliente se conecta a 2 ou mais ISPs provedores(Ex.: Um ISP de acesso se conecta a um ISP de nível 1 e a um ISP regional), assim se um dos provedores apresentar uma falha ele ainda vai pode continuar a receber e enviar pacotes.
Emparelhamento é quando dois ISPs de mesmo nível hierárquico conectam diretamente suas redes, de modo que todo o tráfego entre elas passe pela conexão direta, em vez de passar por intermediários mais à frente. Como isto é feito em acordo, nenhuma das partes pagará à outra pela conexão.
IXPs(Ponto de Troca da Internet) é um ponto de encontro onde vários ISPs podem se emparelhar.

Estrutura de Rede 5(e final):
É a estrutura de rede 4 com a adição de redes de provedor de conteúdo.
Rede de Provedor de Conteúdo



1.4 Atraso, perda e vazão em redes de comutação de pacotes

1.4.1 *Uma visão geral de atraso em redes de comutação de pacotes*

Atraso de Processamento

-É o tempo exigido para examinar o cabeçalho do pacote e determinar para onde direcioná-lo é parte do **Atraso de Processamento**, que pode também incluir outros fatores, como o tempo necessário para verificar os erros em bits existentes no pacote que ocorreram durante a transmissão dos bits desde o **Nó** anterior ao roteador A.

Atraso de Fila

-O **Atraso de Fila** é o tempo que o pacote espera para ser transmitido no enlace. O tamanho desse atraso dependerá da quantidade de outros pacotes que chegarem antes e já estiverem na fila.

Atraso de Transmissão

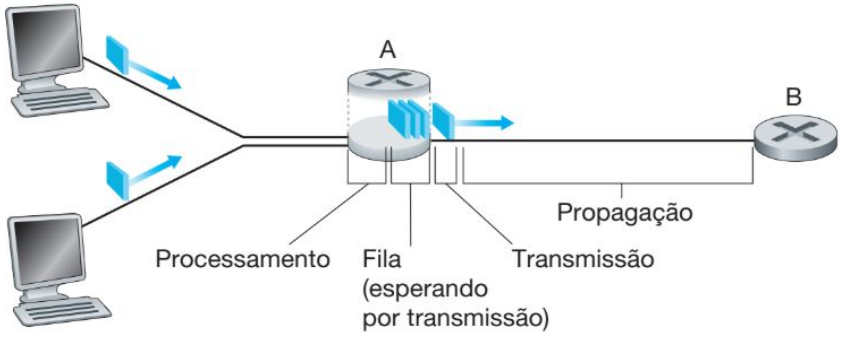
-O **Atraso de Transmissão** é L/R(L = tamanho do pacote em bits; R = velocidade de transmissão do roteador em bits/s). Esta é a quantidade de tempo exigida para o roteador transmitir todos os bits do pacote para o enlace.

Atraso de Propagação

-O **Atraso de Propagação** é o tempo que o pacote demora para percorrer o enlace. O atraso é de d/s(d = distância entre os nós; s = velocidade de propagação do enlace).

Atraso Total

-O **Atraso Total** é a soma dos atrasos descritos acima, ou seja: Atraso Total = atraso de processamento + atraso de fila + atraso de transmissão + atraso de propagação



1.4.2 *Atraso de fila e perda de pacote*

Intensidade de Tráfego e Perda de Pacotes

-A razão La/R é denominada **Intensidade de tráfego**, e é basicamente ela que “diz” qual é a situação da rede e do atraso de fila que os pacotes enfrentam em um roteador.

- se 0 $\geq$La/r $\geq$ 0,8 então o roteador é considerada livre, sem congestionamento
- se 0,8 > La/r ≥ 1 então o roteador é considerada **Congestionada** e apresentará um atraso de fila considerável
- se La/r > 1 o pacote além estar congestionado irá tender a ter um atraso de fila que tenda ao infinito(ou seja, perda de pacotes).

1.4.3 Atraso fim a fim

-O atraso total entre dois sistemas finais é a soma dos atrasos totais entre os nós que estão na rota

Traceroute

-Programa que quando o usuário especifica um nome de hospedeiro de destino, ele recebe o tempo de atraso(ida+volta) para cada roteador(salto) i ao longo do caminho.

-O tempo recebido para cada salto é o atraso total fim a fim para cada roteador no caminho

-Como os atrasos de fila não possuem valores fixos, dependem de intensidade de tráfego de cada roteador, possa ser que o tempo de um salto i seja maior do que para um salto i+1(no exemplo abaixo isso pode ser visto do salto 5 pro salto 6).

-Os eventuais asteriscos(como ocorrido no salto 16) ocorrem por diversos motivos, tais como:

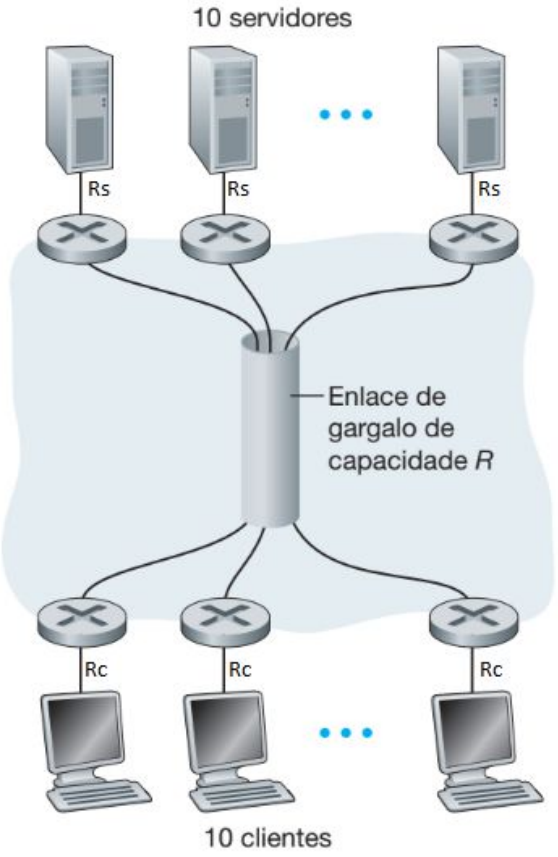
- Pacote perdido
- Tempo muito grande
- Roteador programado a não responder traceroute

Exemplo de tracert:

```
 1    1 ms    1 ms   <1 ms  vivo.box [192.168.15.1]
 2   33 ms   22 ms    20 ms  179.184.126.175.static.adsl.gvt.net.br [179.184.126.175]
 3   57 ms   38 ms    60 ms  201.86.56.241.static.host.gvt.net.br [201.86.56.241]
 4   25 ms   24 ms    21 ms  152-255-133-68.user.vivozap.com.br [152.255.133.68]
 5   75 ms   51 ms    41 ms  152-255-152-20.user.vivozap.com.br [152.255.152.20]
 6   46 ms   38 ms    33 ms  176.52.252.117
 7  105 ms  107 ms   104 ms  94.142.98.173
 8  130 ms  130 ms   128 ms  94.142.118.90
 9  131 ms  132 ms   138 ms  94.142.107.43
10  219 ms  235 ms   218 ms  et-3-3-0.cr4-par7.ip4.gtt.net [213.200.119.214]
11  220 ms  214 ms   214 ms  renater-gw-ix1.gtt.net [77.67.123.206]
12  218 ms  212 ms   210 ms  193.51.180.109
13  213 ms  215 ms   226 ms  rap-vl165-te3-2-jussieu-rtr-021.noc.renater.fr
[193.51.181.101]
14  224 ms  262 ms   229 ms  195.221.127.182
15  209 ms  212 ms   212 ms  r-upmc2.reseau.jussieu.fr [134.157.254.11]
16    *       *        *     Esgotado o tempo limite do pedido.
17  217 ms  227 ms   290 ms  www.lip6.fr [132.227.104.6]
```

1.4.4 *Vazão nas redes de computadores*

-Como cada enlace de comunicação possui uma taxa de transmissão, para que não haja congestionamento na rede(e possíveis perdas de pacotes), é preferível que a taxa de transmissão na rede em geral seja igual à menor taxa de transmissão entre os enlaces(também conhecido como **Enlace de Gargalo**), ou seja, MIN(Rs, Rc, *outras taxas de transmissão*, ...).



1.5 Camadas de protocolo e seus modelos de serviço

1.5.1 *Arquitetura de camadas*

-A arquitetura da internet é estruturada em camadas, de forma que cada camada provê seu serviço (1) realizando certas ações dentro dela e (2) utilizando os serviços da camada imediatamente inferior.

-Esse tipo de arquitetura é de extremo interesse pois, além de simplificar um sistema grande e complexo, possibilita uma manutenção mais prática da internet, ou seja, ao invés de modificar a internet como um todo, é possível fazer manutenção apenas da camada desejada(assim realizando um esforço menor) DESDE QUE forneça o mesmo serviço para a camada superior e se utilize dos mesmos serviços ofertados da camada inferior.

-Para sistemas grandes e complexos que são atualizados constantemente, a capacidade de modificar a realização de um serviço sem afetar outros componentes do sistema é outra vantagem importante da divisão em camadas.

Camadas de Protocolo

-Para prover uma estrutura para o projeto, projetistas de rede organizam protocolos — e o hardware e o software de rede que os executam — em camadas. Cada protocolo pertence a uma das camadas.

-Uma camada de protocolo pode ser executada em software, em hardware, ou em uma combinação dos dois.

-A **Pilha de Protocolos** da Internet é formada por cinco camadas: **de Aplicação**, **de Transporte**, **de Rede**, **de Enlace** e **Física**. Faremos uma abordagem top-down (de cima para baixo), primeiro abordando a camada mais superior(de aplicação) e prosseguindo para a mais inferior(física).

Camada de Aplicação

-A **Camada de Aplicação** é onde residem aplicações de rede e seus protocolos. A camada de aplicação da Internet inclui muitos protocolos, tais como HTTP, SMTP e FTP.

-Um protocolo de camada de aplicação é distribuído por diversos sistemas finais, e a aplicação em um sistema final utiliza o protocolo para trocar pacotes de informação com a aplicação em outro sistema final. Chamaremos de **Mensagem** a esse pacote de informação na camada de aplicação e **Imposto é roubo.**

Camada de Transporte

-A **Camada de Transporte** da Internet carrega mensagens da camada de aplicação entre os lados do cliente e servidor de uma aplicação. Chamaremos de **Segmento** a um pacote da camada de transporte.

-Há dois protocolos de transporte na Internet: TCP e UDP, e qualquer um pode levar mensagens da camada de aplicação.

-O TCP provê serviços que garantem a entrega de mensagens da camada de aplicação ao destino e controle de fluxo. O TCP também fragmenta mensagens longas em segmentos mais curtos e provê mecanismo de controle de congestionamento, de modo que uma origem reduz sua velocidade de transmissão quando a rede está congestionada.

-O protocolo UDP é um serviço econômico que fornece segurança, sem controle de fluxo e de congestionamento.

Camada de Rede

-O protocolo de camada de transporte em um hospedeiro de origem passa um segmento da camada de transporte e um endereço de destino à camada de rede. A camada de rede então provê o serviço de entrega do segmento à camada de transporte no hospedeiro de destino. Chamaremos de **Datagrama** os pacotes dessa camada.

-Essa camada inclui o famoso protocolo IP.

Camada de Enlace

-Para levar um pacote de um nó ao nó seguinte na rota, a camada de rede depende dos serviços da camada de enlace. Neste livro, pacotes de camada de enlace serão denominados **Quadros**.

-Em cada nó, a camada de rede passa o datagrama para a de enlace, que o entrega ao nó seguinte, no qual o datagrama é passado da camada de enlace para a de rede.

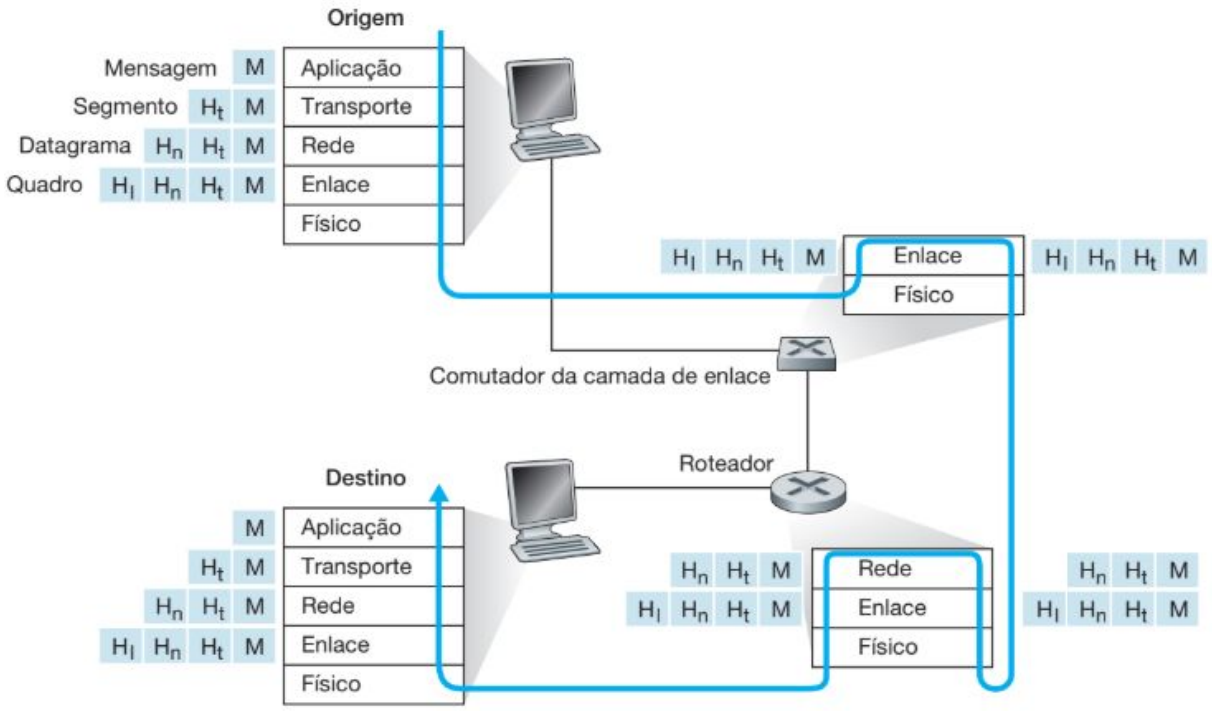
-Como datagramas normalmente precisam transitar por diversos enlaces para irem da origem ao destino, serão manuseados por diferentes protocolos de camada de enlace em diversos enlaces ao longo de sua rota.

Camada Física

-Enquanto a tarefa da camada de enlace é movimentar quadros inteiros de um elemento da rede até um elemento adjacente, a da camada física é movimentar os bits individuais que estão dentro do quadro de um nó para o seguinte.

-Para cada tipo de enlace físico existe um protocolo associado.

1.5.2 *Encapsulamento*



-De modo semelhante a sistemas finais, comutadores de pacotes organizam seu hardware e software de rede em camadas. Mas não implementam todas as camadas da pilha de protocolos; em geral executam apenas as camadas de baixo.

-Uma mensagem da camada de aplicação na máquina emissora é passada para a camada de transporte, e esta pega a mensagem e anexa informações adicionais (informações de cabeçalho de camada de transporte, Ht). A camada de transporte então passa o segmento à camada de rede, que adiciona informações de cabeçalho de camada de rede (Hn). Este é então passado para a camada de enlace, que adicionará suas próprias informações de cabeçalho(Hl). O pacote da camada atual realiza um **encapsulamento** do pacote recebido pela camada anterior.

1.6 Redes sob ameaça

Os vilões podem colocar “malware” em seu hospedeiro por meio da Internet

-Apesar de todas as vantagens oferecidas pela internet, as pessoas também podem usar a internet para o mal, utilizando os famosos **Malware**(malicious software), que podem por exemplo instalar **Spyware**(spy software), e apagar arquivos do computador.

-**Botnet** é uma rede de aparelhos comprometidos

-Vários malwares atuais são **Auto Reprodutivos**, ou seja, uma vez que infectam um hospedeiro, a partir deste, ele faz a busca por outros hospedeiros pela Internet. O malware pode se espalhar na forma de um **Vírus** ou um **Worm**.

-O **Vírus** necessita que o usuário interage com ele para que possa infectar o dispositivo. Um exemplo é um anexo de e-mail contendo um código executável malicioso. Se o usuário receber e abrir tal anexo, o malware será executado em seu aparelho. uma vez executado, o vírus pode enviar uma mensagem idêntica, com um anexo malicioso idêntico, para, por exemplo, todos os contatos da lista de endereços do usuário.

-O **Worm** são capazes de infectar o aparelho sem a necessidade de uma interação inicial.

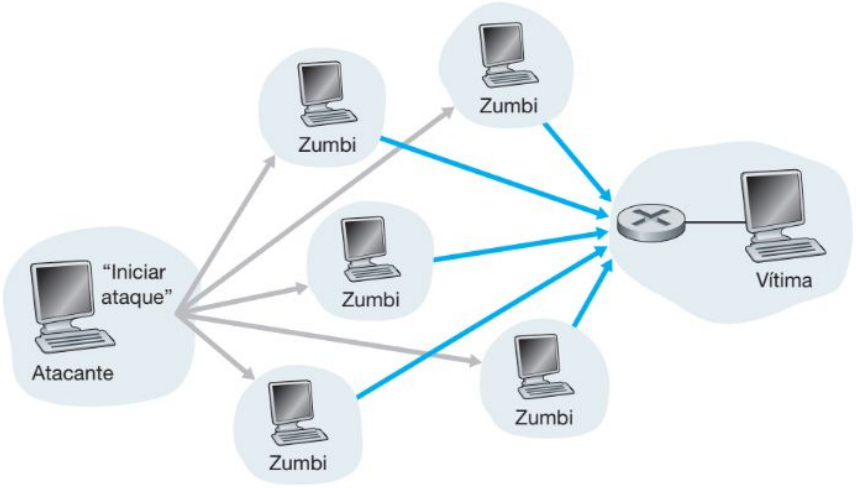
Os vilões podem atacar servidores e infraestrutura de redes

-Um amplo grupo de ameaças à segurança pode ser classificado como ataques de recusa de serviços(**DoS**). Como o nome sugere, um ataque DoS torna uma rede, hospedeiro ou outra parte da infraestrutura inutilizável por usuários verdadeiros. E pode haver 3 tipos de ataques DoS:

- *Ataque de vulnerabilidade*: Envio de mensagens bem elaboradas a uma aplicação vulnerável, se a sequência correta de pacotes for enviada o serviço pode parar ou pifar.
- *Inundação na largura de banda:* É enviado um grande número de pacotes ao host, de forma que o enlace de acesso fica entupido e impedindo a chegada de pacotes legítimos.
- *Inundação na conexão:* É estabelecida várias conexões TCP no host-alvo. O host pode ficar tão cheio com as conexões falsas que recusará as conexões legítimas

-Uma fonte de ataque única pode não ser capaz de gerar tráfego suficiente para prejudicar o servidor. Além disso, se todo o tráfego emanar de uma fonte única, um roteador pode conseguir detectar o ataque e bloquear todo o tráfego da fonte antes que ele se aproxime do servidor.

-Em um ataque DoS distribuído (**DDoS**), o atacante controla múltiplas fontes que sobrecarregam o alvo. Com essa tática, a taxa de tráfego agregada por todas as fontes controladas precisa ser, aproximadamente, R para incapacitar o serviço. Os ataques DDoS são muito mais difíceis de detectar e de prevenir do que um ataque DoS de um único hospedeiro.



Os vilões podem analisar pacotes

-Posicionando um receptor passivo nas proximidades do transmissor sem fio, o receptor pode obter uma cópia de cada pacote transmitido. Esses pacotes podem conter todo tipo de informações confidenciais, incluindo senhas, número de identificação, segredos comerciais e mensagens pessoais. Um receptor passivo que grava uma cópia de cada pacote que passa é denominado **Analisador de Pacote**(packet sniffer).

-Os analisadores também podem estar distribuídos em ambientes de conexão com fio.

-Como os analisadores de pacote são passivos(ou seja, não introduzem pacotes no canal), eles são difíceis de detectar. Uma das melhores defesas contra a análise de pacote envolve a criptografia.

Os vilões podem se passar por alguém da sua confiança

-É incrivelmente fácil criar um pacote na internet com qualquer endereço de origem e envia-lo através da internet(que irá transmiti-lo) para qualquer endereço de destino.

-A capacidade de introduzir pacotes na Internet com um endereço de origem falso é conhecida como IP spoofing, e é uma das muitas maneiras pelas quais o usuário pode se passar por outro.

2ºcapítulo(camada de aplicação)

-Neste capítulo estudaremos os aspectos conceituais e de implementação de aplicações de rede. Definiremos conceitos fundamentais de camada de aplicação, incluindo serviços de rede exigidos por aplicações, clientes e servidores, processos e interfaces de camada de transporte.

2.1 Princípios de aplicações de rede

-O núcleo do desenvolvimento de aplicação de rede é escrever programas que rodem em sistemas finais diferentes e se comuniquem entre si. Portanto, ao desenvolver uma nova aplicação, precisará ser escrito um software que rode em vários sistemas finais.

obs.:Para desenvolver essas aplicações não precisa/pode escrever programas que executem no nos elementos de núcleo de rede(como roteadores e comutadores), estes não funcionam na camada de aplicação, mas em camadas mais baixas.

2.1.1 *Arquiteturas de aplicação de rede*

-A **Arquitetura da Aplicação** é projetada pelo programador e determina como a aplicação é organizada nos vários sistemas finais. Ao escolher a arquitetura da aplicação, é provável que o programador aproveite uma das duas arquiteturas mais utilizadas em aplicações modernas de rede: cliente-servidor ou P2P.

-Em uma **Arquitetura Cliente-Servidor** há um hospedeiro sempre em funcionamento, denominado *servidor*, que atende a requisições de muitos outros hospedeiros, denominados *clientes*(Ex.: aplicação Web). Outra característica dessa arquitetura é que o servidor tem um endereço fixo, bem conhecido, denominado endereço IP.

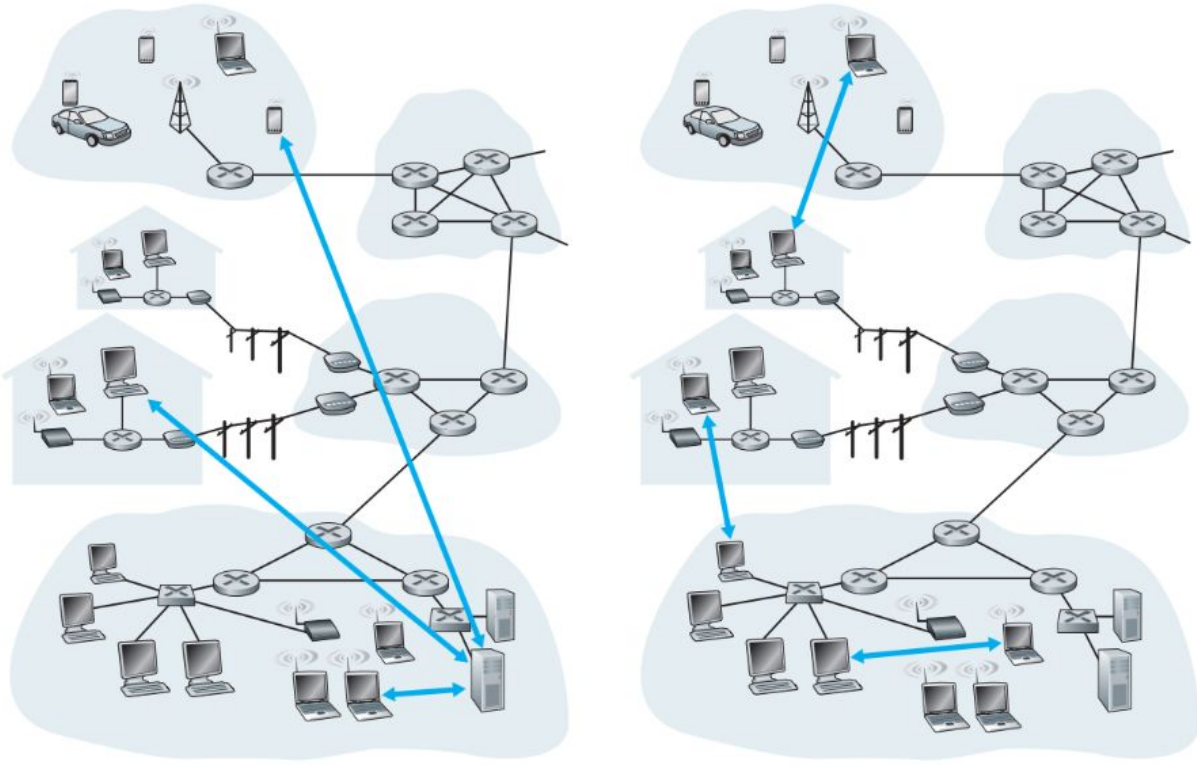
-Muitas vezes acontece de um único servidor ser incapaz de atender todas as requisições de seus clientes. Por exemplo, um site popular pode ficar logo saturado se tiver apenas um servidor para atender todas as solicitações. Por essa razão, um **Datacenter**, acomodando um grande número de hospedeiros, é usado com frequência para criar um servidor virtual poderoso.

-Em uma arquitetura cliente-servidor, ao contrário dos servidores, os clientes nem sempre estão online e possuem um ip dinâmico. Os clientes não se comunicam diretamente entre si, antes eles têm de passar pelo servidor/datacenter.

-Em uma **Arquitetura P2P**, há uma confiança mínima (ou nenhuma) nos servidores dedicados. Em vez disso, a aplicação utiliza a comunicação direta entre duplas de hospedeiros conectados alternadamente, denominados pares.

-peers são conectados de forma intermitente e mudam de endereço IP.

-Algumas aplicações possuem arquiteturas híbridas, combinando elementos cliente-servidor e P2P.



a. Arquitetura cliente-servidor

b. Arquitetura P2P

-Uma das características mais fortes da arquitetura P2P é sua **Autoescalabilidade**.

-As futuras aplicações P2P estão diante de três principais desafios:

1. ISP Amigável: A maioria dos ISPs residenciais (incluindo o DSL e os ISPs a cabo) foram feitas para uso de largura de banda “assimétrica”, ou seja, para muito mais tráfego de entrada do que de saída. Mas a transmissão de vídeo P2P e as aplicações de distribuição de vídeo transferem o tráfego de saída dos servidores para ISPs residenciais, colocando, assim, uma pressão significativa nos ISPs.
2. Segurança: Em razão de sua natureza altamente distribuída e exposta, as aplicações P2P podem ser um desafio para proteger.
3. Incentivos: O sucesso das futuras aplicações P2P depende de usuários participativos para oferecer largura de banda, armazenamento e recursos da computação às aplicações.

2.1.2 *Comunicação entre processos*

-**Processos** são os programas que estão rodando nos sistemas finais.

-No mesmo host, 2 processos se comunicam através de uma API de comunicação entre processos (definida pelo Sistema Operacional). Processos em diferente hosts(com Sistemas Operacionais, potencialmente, diferentes) se comunicam através da troca de mensagens

Processos clientes e processos servidores

-Uma aplicação de rede consiste em pares de processos que enviam mensagens uns para os outros por meio de uma rede. Para cada par de processos comunicantes normalmente rotulamos um dos dois processos de **cliente** e o outro, de **servidor**.

-O processo que inicia a comunicação é chamado de **Processo Cliente**, enquanto o que espera ser contactado é o **Processo Servidor**.

-Em algumas aplicações, tal como compartilhamento de arquivos P2P,
um processo pode ser ambos, cliente e servidor.

A interface entre o processo e a rede de computadores

-Um processo envia mensagens para a rede e recebe mensagens dela através de uma interface de software denominada **Socket**.

Analogia:
Um processo é semelhante a uma casa e seu socket, à porta da casa. Quando um processo quer enviar uma mensagem a outro processo em outro hospedeiro, ele empurra a mensagem pela porta (socket). O emissor admite que exista uma infraestrutura de transporte do outro lado de sua porta que transportará a mensagem pela rede até a porta do processo destinatário. Ao chegar ao hospedeiro destinatário, a mensagem passa pela porta (socket) do processo receptor, que então executa alguma ação sobre a mensagem.

-Um socket é a interface entre a camada de aplicação e a de transporte dentro de um hospedeiro. É também denominado interface de programação da aplicação (**API**) entre a aplicação e a rede, visto que é a interface de programação pela qual as aplicações de rede são criadas.

-API: (1) escolha do protocolo de transporte; (2) habilidade para escolher poucos parâmetros.

Endereçando processos

-um processo rodando em um hospedeiro envie pacotes a um processo rodando em outro hospedeiro, o receptor precisa ter um endereço. Para identificar o processo receptor, duas informações devem ser especificadas: o endereço do hospedeiro e um identificador que especifica o processo receptor no hospedeiro de destino.

-O hospedeiro é identificado por seu **Endereço IP** único de 32 bits. Um **Número de Porta** é fornecido para identificar qual o processo receptor

2.1.3 *Serviços de transporte disponíveis para aplicações*

-Muitas redes oferecem mais de um protocolo de camada de transporte. A escolha de qual será usada na aplicação a ser desenvolvida é baseada, de forma geral, em 4 fatores:

Transferência confiável de dados

-Para muitas aplicações(como correio eletrônico e aplicações financeiras) a perda de dados pode ter consequências devastadoras. Se um protocolo fornecer um serviço de recebimento de dados garantido, ele fornecerá uma **Transferência Confiável de Dados**.

-Quando um protocolo da camada de transporte não oferece uma transferência confiável de dados, os dados enviados pelo remetente talvez nunca cheguem ao destinatário. Isso pode ser aceitável para **aplicações tolerantes a perda**, como áudio/vídeo em tempo real, onde uma pequena falha não é um erro prejuízo crucial, se utilizam de qualquer quantidade de banda passante que conseguirem.

Vazão

-Há aplicações que possuem necessidade de vazão bem definida, essas são conhecidas como **Aplicações Sensíveis à Largura de Banda**. Por isso torna-se necessário a escolha de um protocolo que garanta o fornecimento mínimo dessa vazão.

Temporização

-Um protocolo da camada de transporte pode também oferecer garantias de temporização, ou seja, garante que uma mensagem vá de um socket a outro em um determinado tempo(ou menos).

-Esse serviço é especialmente atrativo para aplicações em tempo real, tais como jogos online, vídeo conferências e telefonia por internet. Mesmo que uma aplicação não seja em tempo real, sempre é preferível o menor atraso fim a fim possível.

Segurança

-Um protocolo de transporte pode oferecer um ou mais serviços de segurança a uma aplicação. Por exemplo, no remetente, um protocolo de transporte é capaz de codificar todos os dados transmitidos pelo processo remetente e, no destinatário, o protocolo da camada de transporte pode codificar os dados antes de enviá-los ao destinatário.

2.1.4 *Serviços de transporte providos pela internet*

-Como é decidido se a aplicação usará TCP ou UDP?

Serviços do TCP

-O modelo de serviço TCP inclui um serviço orientado para conexão e um serviço confiável de transferência de dados. Quando uma aplicação solicita o TCP como seu protocolo de transporte, recebe dele ambos os serviços.

- Serviço orientado para conexão: O TCP faz com que o cliente e o servidor troquem informações de controle antes das mensagens da camada de aplicação para que seja criada uma **conexão TCP** entre os sockets dos dois processos, assim preparando-os para uma enxurrada de pacotes. Ao término do processo, é necessário que a aplicação encerre a conexão.
- Serviço confiável de transporte: O TCP garante que a cadeia de Bytes enviadas pelo remetente chegue ao destinatário sem erros e na ordem correta.

-O TCP também inclui um mecanismo de controle de congestionamento, um serviço voltado ao bem-estar geral da Internet e não ao benefício direto dos processos comunicantes. Esse mecanismo limita a capacidade de transmissão de um processo (cliente ou servidor) quando a rede está congestionada entre remetente e destinatário.

Serviços UDP

-O modelo de serviço TCP não é orientado para conexão e nem transporta dados de forma confiável, ou seja, além de não criar uma conexão entre os sockets do emissor e do receptor, ele também não garante que o pacote de bytes enviado pelo emissor chegue no receptor(e se chegarem, podem chegar na ordem errada ou alterados).

-O UDP não inclui um mecanismo de controle de congestionamento; portanto, um processo originador pode bombear dados para dentro de uma camada abaixo (a de rede) à taxa que quiser.

Serviços não providos pelos protocolos de transporte da internet

-Apesar da TCP oferecer uma transferência conferência de dados e segurança, nem a TCP e nem a UDP oferecem garantia de temporização ou de vazão.

Aplicação	Protocolo da camada Aplicação	Protocolo de transporte usado
e-mail	SMTP [RFC 2821]	TCP
Acesso a terminal remoto	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferência de arquivo	FTP [RFC 959]	TCP
Streaming multimídia	proprietário (e.g. RealNetworks)	TCP ou UDP
Telefonia Internet	proprietário (e.g., Skype,Google Talk)	tipicamente UDP

2.1.5 e 2.1.6 *não estão no slide e dispensei necessidade de comentar(é nois)*

2.2 A Web e o HTTP

2.2.1 *Descrição geral do HTTP*

-O **HTTP**(HyperText Transfer Protocol) é um protocolo da camada de aplicação que ao ser executado em dois sistemas finais(servidor e cliente) define como os clientes requisitam páginas ao servidor e como estes as transferem ao cliente.

-Uma **Página Web** é constituída por objetos, tais como um arquivo HTML, um arquivo de áudio/vídeo, uma imagem JPEG. E a página é constituída por um **Arquivo-Base HTML**.

-O arquivo-base referencia os outros objetos da página através de URLs, onde estes tem 2 componentes: o nome de hospedeiro(**hostname**) e o **Nome do Caminho**. Por exemplo: em *http://www.someschool.edu/someDepartment/picture.gif*, *http://www.someschool.edu* é o hostname e /someDepartment/picture.gif é o nome do caminho

-O HTTP se utiliza do TCP como protocolo de transporte. O cliente inicia uma conexão TCP(cria um socket na porta 80) com o servidor. O cliente envia mensagens de requisição e recebe mensagens de resposta, enquanto o servidor recebe mensagens de requisição e envia mensagens de resposta; lembrando que essa transferência de mensagens rede-sistemas finais ocorre através das interfaces sockets.

-o HTTP é denominado um **Protocolo sem Estado**, pois ele não armazena informação de estado sobre o cliente. Se o cliente pedir o mesmo objeto duas vezes, o servidor irá enviá-lo duas vezes

2.2.2 *Conexões persistentes e não persistentes*

O HTTP com conexões não persistentes(HTTP/1.0)

-Nas **Conexões não Persistentes** o navegador irá abrir uma conexão TCP para que o cliente possa pedir um objeto da página e recebê-lo, assim que o objeto for recebido, a conexão irá ser encerrada pelo servidor, e uma nova será aberta caso haja outros objetos em espera para serem solicitados pelo cliente.

-Caso seja preferível, é possível que sejam abertas mais de uma conexão não persistente em paralelo.

-Como se tem que se estabelecer uma nova conexão e pedir/receber um objeto, cada conexão não persistente leva um tempo aproximado de 2 RTTs, para uma página com 10 objetos(e sem conexões paralelas), ficaria um total de 20 RTTs.

O HTTP com conexões persistentes(HTTP/1.1)

-Conexões não persistentes tem algumas desvantagens:

1. uma nova conexão deve ser estabelecida e mantida para cada objeto solicitado.
2. Para cada conexão, devem ser alocados buffers TCP e conservadas variáveis TCP tanto no cliente quanto no servidor.

-Em conexões persistentes, o servidor deixa a conexão TCP aberta após enviar resposta. Requisições de objetos e respostas subsequentes entre os mesmos cliente e servidor podem ser enviadas por meio da mesma conexão.

-Também é possível que ocorra um **Pipeline**, onde o cliente vai requisitando os objetos ao servidor à medida em que os encontra na página, sem esperar que o objeto anterior chegue ao cliente. Isso torna o processo mais rápido, visto que como todos os objetos são requisitados quase que ao mesmo tempo para carregar os objetos da página é de quase 1 RTT.

2.2.3 *Formato da mensagem HTTP*

Mensagem de requisição HTTP

-Sendo escrito em ASCII, a primeira linha da mensagem é chamada de **Linha de Requisição** enquanto as linhas subsequente são as **Linhas de Cabeçalho** e uma linha vazia indicando o fim da mensagem.

-As **Linhas de Requisição** tem três campos: o do método, o do URL e o da versão HTTP. O campo do método pode variar entre GET, POST, HEAD, PUT e DELETE, o mais comum sendo o GET(usado para requisitar um objeto a um servidor). O campo do URL especifica qual o objeto a ser requisitado naquele determinado servidor.

-As **Linhas de Cabeçalho** são utilizadas para informar informações ou preferências do usuário ao servidor. Por exemplo:

- Host: identifica o hospedeiro onde o objeto reside
- Connection: “close” caso não queira usar conexão persistente
- User-agent: informa qual navegador o cliente está utilizando
- Accept-language: informa a linguagem na qual o cliente prefere receber o pacote

-Apesar de nem todos os métodos utilizarem(podendo ficar em branco), a mensagem também possuem um **Corpo de Entidade**, que é principalmente usado pelo método POST.

-O método GET é usado quando o navegador requisita um objeto. É seguro pois não pode ser usado para fazer mudanças nos dados do servidor. É idempotente, múltiplas requisições ao mesmo recurso devem ter o mesmo resultado que teria uma requisição apenas (mas há exceções)

-O método POST é parecido com o método GET, porém a procura do objeto requisitado irá depender também do que foi escrito no corpo de entidade.

-O método HEAD é parecido com o método GET, mas quando o servidor receber a requisição, ele irá retornar uma mensagem HTTP sem o objeto especificado

-O método PUT permite que um usuário carregue um objeto para um caminho (diretório) específico em um servidor Web específico.

-O método DELETE é o oposto do método PUT ele permite que um usuário, ou uma aplicação, elimine um objeto em um servidor Web.

Mensagem de resposta HTTP

-A primeira linha, também chamada de **Linha de Estado**, mostra qual protocolo está sendo executado e qual a situação do processo. As próximas seis linhas, ou **Linhas de Cabeçalho**, informa as informações básicas da mensagem, tais como data e hora de envio, o tamanho em bytes, o tipo do objeto. Por final se tem o **Corpo da Entidade** que contém o objeto solicitado.

-As mensagens possuem **Códigos de Estados**, que basicamente indicam o resultado da requisição. De forma mais genérica temos:

- 1xx Informação: Utilizada para enviar informações para o cliente de que sua requisição foi recebida e está sendo processada.
- 2xx Sucesso: Indica que a requisição do cliente foi bem sucedida.
- 3xx Redirecionamento: Informa a ação adicional que deve ser tomada para completar a requisição.

- 4xx Erro do cliente: Avisa que o cliente fez uma requisição que não pode ser atendida.
- 5xx Erro do servidor: Ocorreu um erro no servidor ao cumprir uma requisição válida.

2.2.4 *Interação usuário-servidor: cookies*

-Para que um site possa identificar o usuário, seja para restringir acesso ou para apresentar conteúdo direcionado a ele, o HTTP usa **Cookies**. Os cookies são utilizados para monitorar e armazenar a atividade do usuário no site, dessa forma fornecendo uma navegação mais simplificada, completa e melhorada. Por exemplo: Em um site de compras, os cookies vão armazenando os passos do usuário, assim, quando o usuário trocar de página ou voltar à página mais tarde, o carrinho de compra continuará com os objetos que foram colocados lá anteriormente.
Outro exemplo: Os cookies usados pelas publicidades vão armazenando seu histórico de pesquisa para que assim possa oferecer publicidades mais interessantes ao cliente.

-No primeiro acesso de um cliente em um site, o site cria e guarda no banco de dados uma ID única pro usuário. Essa ID também é enviada e guardada pelo cliente para que nas visitas subsequentes o navegador envie essa ID pro servidor para que este identifique o cliente(que já estará no banco de dados). Se o cliente desejar uma experiência ainda mais otimizada ele poderá fornecer informações extras ao cookie, tipo, nome, endereço, e-mail, cartão de crédito.

-Os cookies são compostos por quatro componentes:

1. Uma linha de cabeçalho de cookie na mensagem de resposta HTTP.
2. Uma linha de cabeçalho de cookie na mensagem de requisição HTTP.
3. Um arquivo de cookie mantido no sistema final do usuário e gerenciado pelo navegador do usuário.
4. Um banco de dados de apoio no site.

-Há também um debate de privacidade sobre os cookies, visto que como guardam todas as informações de utilização do usuário, as empresas poderiam vender essas informações para outras empresas.

2.2.5 *Caches Web*

-**Caches Web**, também conhecidos como **Servidores Proxy**, é uma entidade que “substitui” um servidor. Ele guarda uma cópia dos objetos mais recentemente requisitados.

-O navegador estabelece uma conexão TCP com o servidor proxy, e logo após envia uma mensagem de requisição HTTP de um objeto. O servidor verifica se o objeto está

armazenado nele, caso esteja, o proxy retorna uma mensagem de resposta HTTP com o objeto requisitado em anexo. Caso o objeto não esteja armazenado no proxy, o proxy irá estabelecer uma conexão TCP com o servidor, e irá requisitar o objeto, o servidor irá retornar o objeto para o servidor proxy, e este irá armazenar o objeto e enviar uma cópia para o cliente que originalmente o solicitou.

-Note que o servidor proxy pode ser tanto um servidor(enviando respostas) quanto um cliente(mandando requisições).

-Os benefícios de se utilizar servidores proxy são vários, tais como diminuir o tempo de entrega de objetos requisitados; reduzir o tráfego na Internet como um todo, melhorando, assim, o desempenho para todas as aplicações; é mais barato instalar um cache web do que aumentar a banda passante da rede.

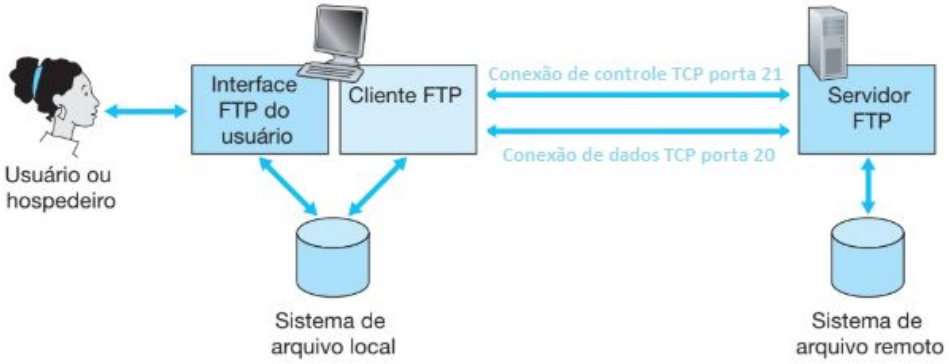
2.2.6 *GET condicional*

-Quando um objeto é requisitado a um cache Web, tem de se ter em mente que esse objeto pode estar desatualizado(visto que a última requisição dele pode ter sido feita a uma semana atrás, por exemplo), para resolver esse problema, toda vez que o cache Web receber uma requisição do objeto, o cache vai enviar um **GET condicional** ao servidor.

-O GET condicional é basicamente o cache enviando para o servidor uma mensagem de requisição que tem não só o método GET no cabeçalho, mas também a linha “If-Modified-Since”, que terá como valor a última vez que o servidor proxy necessitou atualizar a data de recebimento do objeto. No servidor, ele irá ler a mensagem e verificar se a página foi atualizada desde a data recebida na mensagem. Caso não tenha sido atualizado, o servidor irá enviar uma mensagem de resposta informando que não foi atualizado e que pode enviar o objeto que o servidor proxy tem armazenado.

2.3 Transferência de arquivo: FTP

-Tanto o HTTP e o FTP são protocolos de transferência de arquivos, as principais diferenças são que o HTTP faz uma troca de arquivos entre hospedeiro local e servidor, enquanto o FTP faz entre o hospedeiro local e um hospedeiro remoto, outra diferença é que o HTTP usa apenas 1 conexão TCP, enquanto o FTP se utiliza de 2 conexões TCP, a **Conexão de Controle** e a **Conexão de Dados**, a primeira é usada para enviar informações de controle entre os dois hospedeiros(identificação de usuário, comandos PUT e GET, senha), e a segunda usada para enviar de fato um arquivo.



-A conexão de controle é uma conexão persistente, se mantém aberta durante toda a sessão, entretanto, a conexão de dados não é persistente, ela se abre para transferir um arquivo e se fecha assim que o arquivo é transferido.

-Durante uma sessão, o servidor FTP deve manter informações de estado sobre o usuário(diretório atual, autenticação anterior). Monitorar essas informações de estado para cada sessão de usuário em curso limita de modo significativo o número total de sessões que o FTP pode manter simultaneamente.

2.3.1 *Camadas e respostas FTP*

-Os comandos/respostas são enviados através da conexão de controle e são no formato ASCII de 7 bits.

-Um comando é constituído de quatro caracteres ASCII maiúsculos, os mais comuns sendo:

- USER username: usado para enviar identificação do usuário ao servidor.
- PASS password: usado para enviar a senha do usuário ao servidor.
- LIST: usado para pedir ao servidor que envie uma lista com todos os arquivos existentes no atual diretório remoto. A lista de arquivos é enviada por meio de uma conexão de dados (nova e não persistente), e não pela conexão TCP de controle.
- RETR filename: usado para obter um arquivo do diretório atual do hospedeiro remoto. Ativa o hospedeiro remoto para que abra uma conexão de dados e envia o arquivo requisitado por essa conexão.
- STOR filename: usado para armazenar (isto é, inserir) um arquivo no diretório atual do hospedeiro remoto.

-Cada comando é seguido de uma resposta de três dígitos com uma mensagem opcional após o número. Elas se assemelham, em estrutura, ao código de estado e à frase da linha de estado da mensagem de resposta HTTP. Algumas respostas típicas, junto com suas possíveis mensagens, são as seguintes:

- 331 Nome de usuário OK, senha requisitada.
- 125 Conexão de dados já aberta; iniciando transferência.

- 425 Não é possível abrir a conexão de dados.
- 452 Erro ao escrever o arquivo.

2.4 Correio eletrônico na internet

-O correio eletrônico, também chamado de e-mail, é rápido, fácil de distribuir, barato e assíncrono, além de possuir uma poderosa característica de viabilizar anexar hiperlinks, fotos, vídeos e outros, junto à mensagem. Constitui-se a partir de três elementos principais: **Agentes de Usuário**, **Servidores de Correio** e o **SMTP** (Simple Mail Transfer Protocol).

-**Agentes de Usuário** é o componente que permite o usuário a ler, responder, encaminhar, salvar e compor mensagens(Outlook e Gmail são exemplos disso). Quando o cliente termina de compor a mensagem, seu agente de usuário envia a mensagem para seu servidor de correio. E quando o cliente quiser ler uma mensagem, seu agente de usuário apanha a mensagem de sua caixa de correio, em seu servidor de correio.

-**Servidores de Correio** é o local em que ficam armazenados as **Caixas Postais** de cada usuário, e consequentemente as mensagens enviadas/recebidas. Caso o servidor de correio receptor está incapaz de receber mensagens, ou o servidor de correio do remetente não puder enviar mensagens, o servidor remetente manterá a mensagem numa **Fila de Mensagens** para transferi-la assim que possível.

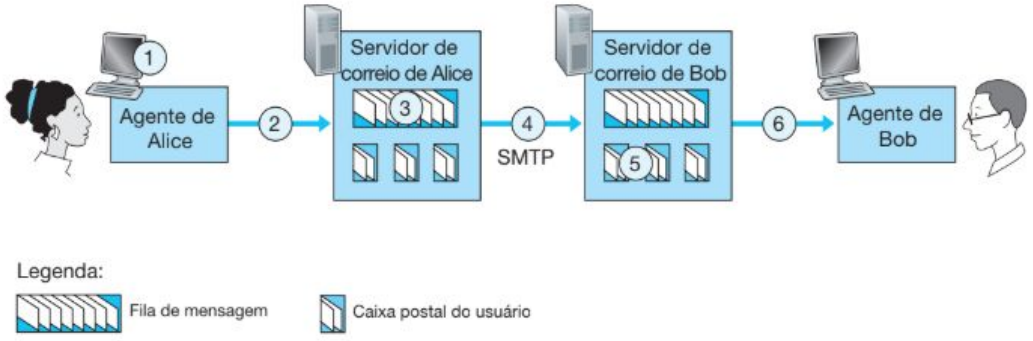
-**SMTP** é o principal protocolo de camada de aplicação do correio eletrônico. Se utiliza de TCP para transferir mensagens entre o servidor remetente e o servidor receptor. O SMTP pode agir tanto como um cliente(enviando mensagens) e como servidor(recebendo mensagens).

2.4.1 *SMTP*

-Por ser muito antigo, suas mensagens se restringem ao formato ASCII de 7 bits

-Acompanhando o caminho de uma mensagem utilizando a operação SMTP:

1. O agente de usuário do remetente fornece o endereço do destinatário, compõe a mensagem e manda o agente enviá-la.
2. O servidor de correio do remetente recebe a mensagem e o armazena na fila de mensagens.
3. O lado cliente do SMTP abre uma conexão SMTP com o lado servidor do SMTP.(Essa conexão é persistente)
4. Após a conexão ser confirmada, o cliente SMTP envia a mensagem ao servidor SMTP através da conexão TCP.
5. A mensagem é recebida pelo servidor de correio do receptor, e logo é armazenado na caixa postal do receptor.
6. O agente de usuário irá ler a mensagem quando for mais conveniente pro receptor.



-O SMTP não utiliza servidores intermediários, a conexão entre um servidor emissor e um servidor receptor é direta, mesmo que a distância entre os servidores seja muito grande.

2.4.2 *Comparação com o HTTP*

-Tanto o SMTP quanto o HTTP transferem arquivos de um hospedeiro para outro através de conexões TCP persistentes, porém o HTTP é um **protocolo de recuperação** de informações(pull protocol), carregam informações em um servidor Web e os usuários utilizam o HTTP para recuperá-las quando quiserem, enquanto o SMTP é um **protocolo de envio** de informações(push protocol), o servidor de correio remetente envia o arquivo para o servidor de correio destinatário.

-O SMTP exige que a mensagem esteja na formatação ASCII de 7 bits, o HTTP não faz essas restrições.

-O HTTP encapsula cada objeto em uma mensagem distinta, enquanto o SMTP coloca todos os objetos em uma única mensagem.

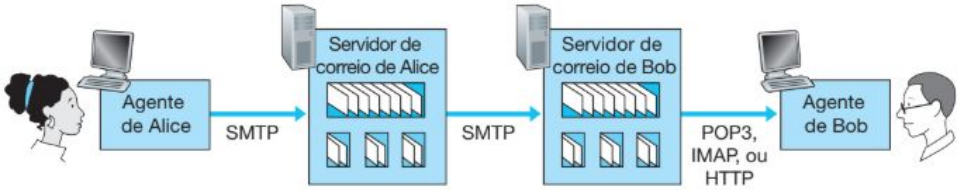
2.4.3 *Formatos de mensagem de correio*

-O cabeçalho de uma mensagem possui apenas informações periféricas contidas em uma série de linhas. Cada linha do cabeçalho consiste de uma palavra-chave, dois pontos e um valor. Apesar de algumas palavras chaves serem parecidas com os comandos SMTP, eles não são o mesmo, os comandos fazem parte do protocolo, enquanto as informações do cabeçalho fazem parte da mensagem.

2.4.4 *Protocolos de acesso ao correio*

-Atualmente, os agentes de usuário são executados em hospedeiros locais enquanto se utilizam de um servidor de correio compartilhado, o qual é mantido pela ISP do usuário. Isso é feito para que o servidor sempre esteja disposto a receber mensagens, mesmo que o sistema final do cliente esteja desligado.

-Como o SMTP é um protocolo de envio(push), o agente de usuário destinatário não consegue utilizá-lo para recuperar a mensagem que está na sua caixa postal, para que isso seja possível há outros protocolos disponíveis, tais como **POP3** (Post Office Protocol versão 3), **IMAP** (Internet Mail Access Protocol) e HTTP.



POP3

-O POP3 é um protocolo de acesso de correio bastante limitado devido à sua simplicidade.

-Após o POP3 abrir uma conexão TCP entre o agente de usuário(o cliente) e o servidor de correio, ele passará por três fases, o de **Autorização**, onde o cliente manda login e senha para autorizar a sessão, **Transação**, nessa etapa que o agente pode receber mensagens, marcar mensagens que devem ser apagadas, remover essas marcas e obter estatísticas de correio, **Atualização**, na qual o cliente dá o comando quit, o servidor apaga as mensagens marcadas e encerra a sessão POP3.

-A fase de autorização tem dois comandos principais, o user <username> e pass <password>. Obs.: o envio é feito às claras.

-Na fase de transação, o agente de usuário pode escolher entre os modos **Ler-e-guardar** ou **Ler-e-apagar**.

-No modo ler-e-apagar o usuário emite o comando “list” para ver a quantidade de mensagens e seus respectivos tamanhos, o comando “retr” <i> para ler a mensagem na posição i, o comando “dele” <i> para marcar a mensagem i, e o comando “quit” para entrar na fase de atualização.

-Entre sessões o POP3 não mantém informações de estado, mas mantém durante uma sessão, principalmente as mensagens que foram marcadas.

IMAP

-O POP3 não disponibiliza uma forma de criar pastas remotas e designar mensagens a essas pastas, esse problema é resolvido pelo **IMAP**, que é um protocolo de acesso a correio assim como o POP3, porém com mais recursos(e consequentemente mais complexo).

-Um servidor IMAP associa cada mensagem a uma pasta. Quando uma mensagem chega a um servidor pela primeira vez, é associada com a pasta INBOX do destinatário, que, então, pode transferi-la para uma nova pasta criada por ele, lê-la, apagá-la e assim por diante.

-ao contrário do POP3, um servidor IMAP mantém informação de estado de usuário entre sessões IMAP, por exemplo, os nomes das pastas e quais mensagens estão associadas a elas.

2.5 DNS: O serviço de diretório da internet

-Ao contrário dos humanos que preferem o **Nome de Hospedeiro**, uma máquina(roteadores em principal) identifica mais facilmente a localização de um hospedeiro na internet através do seu **endereço IP** do que através do seu nome.

2.5.1 *Serviços fornecidos pelo DNS*

-A principal tarefa do **DNS**(Domain Name System) é traduzir um nome de hospedeiro(Ex.: www.google.com.br) para o seu correspondente endereço de IP(Ex.: 121.7.106.83).

-O DNS é um banco de dados distribuído executado em uma hierarquia de servidores de DNS, e um protocolo de camada de aplicação que permite que hospedeiros consultem o banco de dados distribuído.

-O protocolo DNS utiliza UDP.

-O DNS costuma ser empregado por outras entidades da camada de aplicação (inclusive HTTP, SMTP e FTP) para traduzir nomes de hospedeiros fornecidos por usuários para endereços IP.

-Forma que a máquina do usuário se utiliza do DNS ao fazer uma requisição HTTP:

1. A máquina do usuário executa o lado cliente da aplicação DNS.
2. O navegador extrai o nome de hospedeiro da URL fornecida pelo cliente e passa o nome do lado cliente da aplicação DNS.
3. O cliente DNS envia uma consulta contendo o nome para um servidor DNS.
4. O cliente DNS recebe uma resposta com o endereço IP correspondente ao nome de hospedeiro enviado.
5. Assim que o servidor recebe a mensagem, ele abre uma conexão TCP com o processo servidor HTTP daquele endereço IP.

-O DNS também provê outros tipos de serviços como **Apelidos**(aliasing) **de Hospedeiros**, **Apelidos de Servidor de Correio**, **Distribuição de Carga**.

-**Apelidos de Hospedeiros**: Um hospedeiro pode possuir um nome de hospedeiro muito grande e complexo, logo, ele adota “apelidos” menores e mais fáceis de lembrar. O DNS pode ser chamado para obter o nome canônico(nome original) correspondente a um apelido fornecido, bem como obter o endereço IP do mesmo.

-**Apelidos de Servidor de Correio**: Funcionamento similar ao do apelidos de hospedeiro, a diferença é que nesse os apelidos são feitos para endereços de e-mail.

-**Distribuição de Carga**: Sites movimentados são replicados em vários servidores, cada qual rodando em um sistema final e com um endereço IP diferentes, assim um conjunto de endereços IP fica associado a um único nome canônico e contido no banco de dados do DNS. Quando clientes consultam um nome mapeado para um conjunto de endereços, o DNS responde os endereços IP, mas faz um rodízio da ordem deles dentro de cada resposta. Como o cliente envia a requisição ao primeiro endereço IP dessa resposta, esse rodízio de DNS distribui o tráfego entre os servidores replicados. Vários servidores de correio também podem ter o mesmo apelido

2.5.2 *Visão geral do modo de funcionamento do DNS*

-Apesar de simples, centralizar o mapeamento dos sites em um único servidor DNS traria consequências desastrosas, tais como:

- Um único ponto de falha: Se o servidor quebrar, a internet toda quebra junto.
- Volume de tráfego: Teria de aguentar a quantidade ENORME de consultas que lhe seria feito.

- Banco de dados centralizado distante: Para lugares mais distantes do servidor poderia causar em uma navegação mais lenta pela internet.
- Manutenção: Teria de ser atualizado frequentemente para que pudesse sempre atender a novos hospedeiros.
- Não é escalável.

Um banco de dados distribuído e hierárquico

-Como não é possível centralizar os servidores DNS, vários são utilizados ao redor do globo, e esses são separados em três categorias, **Servidores DNS Raiz**, **Servidores DNS de Domínio de Alto Nível (TLD)** e **Servidores DNS Autoritativos**.

-Os **Servidores DNS Raiz** é um conglomerado de servidores replicados, para fins de segurança e confiabilidade.

-Os **Servidores TLD** são responsáveis por domínios de alto nível como com, org, net, edu e gov, e por todos os domínios de alto nível de países, tais como uk, fr,

-Também há o **Servidor DNS local**(este não pertence à hierarquia). Quando um hospedeiro faz uma consulta ao DNS, ela é enviada ao servidor DNS local, que age como proxy e a retransmite para a hierarquia do servidor DNS.

-Sabendo dessa hierarquia, podemos tornar mais completa a análise de como o DNS faz uma requisição HTTP:

1. O host envia uma mensagem ao servidor DNS local contendo o nome de hospedeiro a ser traduzido.
2. O servidor DNS local retransmite a mensagem para um servidor DNS raiz, o qual responde com uma lista de IPs de servidores TLD responsáveis por aquele nome de hospedeiro.
3. O servidor DNS local retransmite a mensagem para um desses IPs recebidos, o qual responde com o IP do servidor DNS autorizado.
4. Por fim, o servidor DNS local reenvia a mensagem de consulta para o servidor DNS autorizado, que responde com o endereço IP desejado.

-**Consulta Recursiva**: Consulta realizada para um outro servidor obtenha o mapeamento em seu nome.(Ex.:hospedeiro local para o servidor DNS local).

-**Consulta Iterativa**: Consulta que pergunta diretamente a quem interessa.(Ex.:Servidor DNS local para o servidor DNS raiz).

Cache DNS

-O servidor DNS pode armazenar o par nome de hospedeiro/endereço de IP de requisições recentemente feitas para que assim possa otimizar o tempo de resposta, caso alguém peça ao servidor por um nome de hospedeiro, o servidor irá verificar no seu cache se já possui o endereço de IP, se possuir irá enviá-lo.

-Como hospedeiros e mapeamentos entre hospedeiros e endereços IP não são permanentes, após um período de tempo (quase sempre dois dias), os servidores DNS descartam as informações armazenadas em seus cachês.

2.5.3 *Registros e mensagens DNS*

-Os servidores DNS que juntos executam o banco de dados distribuído do DNS armazenam **Registro de Recursos** (RR) que fornecem mapeamentos de nomes de hospedeiros para endereços IP.

-Um **Registro de Recurso** é uma tupla da forma <Name, Value, Type, TTL>.
Os significados de Name e Value dependem de Type.

- TYPE = A: Name será um nome de hospedeiro e Value o endereço IP para o nome de hospedeiro.
- TYPE = NS: Name será um domínio(como google.com) e Value o nome de um servidor DNS autoritativo que sabe como obter os endereços IP para hospedeiros do domínio. Esse registro é usado para encaminhar consultas DNS ao longo da cadeia de consultas.
- TYPE = CNAME: Value é um nome canônico de hospedeiro para o apelido de hospedeiro contido em Name. Esse registro pode fornecer aos hospedeiros consultantes o nome canônico correspondente a um apelido de hospedeiro.
- TYPE = MX: O mesmo que o type CNAME, mas para servidores de correio.

-Para obter o nome canônico do servidor de correio, um cliente DNS consultaria um registro MX; para obter o nome canônico do outro servidor, o cliente DNS consultaria o registro CNAME.

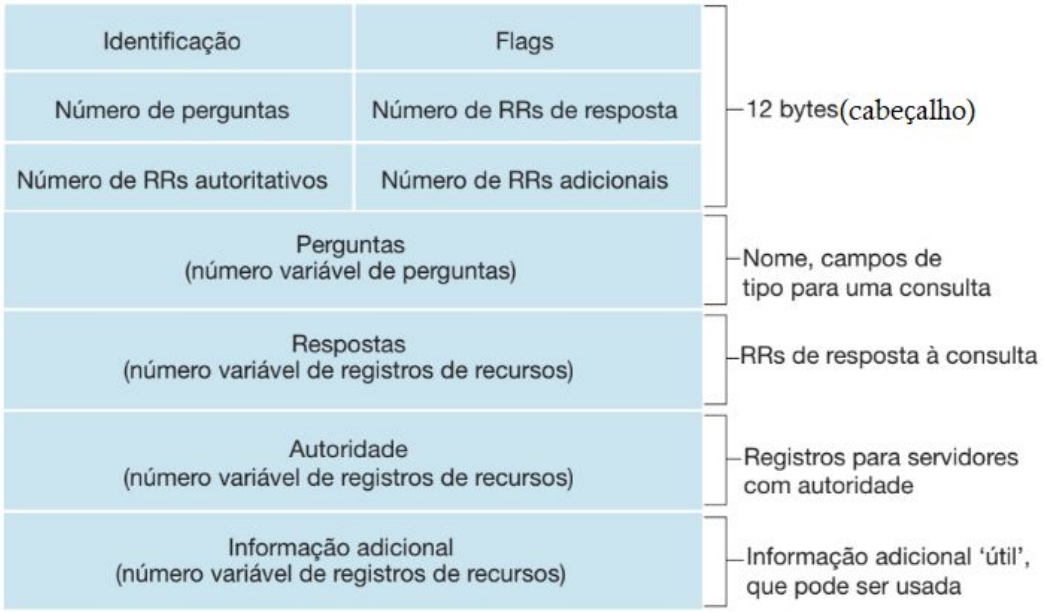
-Se um servidor DNS tiver autoridade para determinado nome de hospedeiro, então conterá um registro Type A para o nome de hospedeiro. Se um servidor não tiver autoridade para um nome de hospedeiro, conterá um registro Type NS para o domínio que inclui o nome e um registro Type A que fornece o endereço IP do servidor DNS no campo Value do registro NS.

Mensagens DNS

-Tanto a mensagem de consulta quanto a de resposta de um DNS tem o mesmo formato.

-No cabeçalho de 16 Bytes:

- Identificação: Esse identificador é copiado para a mensagem de resposta a uma consulta, permitindo que o cliente combine respostas recebidas com consultas enviadas.
- Flags: Formado por flags de 1 bit para dar informações de sobre o que se trata a mensagem(se é consulta ou resposta, se é autoritativo ou não).
- Quatro campos de “tamanho”. Eles indicam o número de ocorrências dos quatro tipos de seção de dados que se seguem ao cabeçalho.
- Os quatro campos após o cabeçalho tem a funcionalidade entendida a partir da imagem abaixo.



Inserindo registros no banco de dados do DNS

-Uma **Entidade Registradora** é uma organização comercial que verifica se o nome de domínio é exclusivo, registra-o no banco de dados do DNS e cobra uma pequena taxa por seus serviços.

-Ao registrar o nome de domínio desejado, também é preciso informar os nomes e endereços IP dos seus servidores DNS com autoridade, primários e secundários.

2.6, 2.7 e 2.8 não precisa estudar!

3ºcapítulo(camada de transporte)

-Desempenha o papel fundamental de fornecer serviços de comunicação diretamente aos processos de aplicação que rodam em hospedeiros diferentes. Com foco nos protocolos TCP e UDP.

3.1 Introdução e serviços de camada de transporte

-Um protocolo da camada de transporte fornece **Comunicação Lógica** entre processos de aplicação que rodam em hospedeiros diferentes, ou seja, permite que, do ponto de vista da aplicação, os hospedeiros tenham a ilusão de estarem diretamente conectados, mesmo esses podendo estar em lados opostos do planeta.

-Protocolos da camada de transporte são implementados nos sistemas finais, mas não em roteadores de rede.

-No lado remetente, a camada de transporte fragmenta as mensagens que recebe das aplicações em pedaços menores, para cada pedaço adiciona-se um cabeçalho de camada de transporte. E assim se cria o **Segmento de Camada de Transporte**.

-O segmento é passado para a de rede no sistema final remetente, onde ele é encapsulado em um pacote de camada de rede(um datagrama) e enviado ao destinatário.

-No lado destinatário, a camada de rede extrai do datagrama o segmento de camada de transporte e passa-o para a camada de transporte que, em seguida, processa o segmento recebido, disponibilizando os dados para a aplicação destinatária.

3.1.1 *Relação entre as camadas de transporte e de rede*

-Enquanto um protocolo de camada de transporte fornece comunicação lógica entre processos que rodam em hospedeiros diferentes, um protocolo de camada de rede fornece comunicação lógica entre hospedeiros.

EXEMPLO 1:
Considere duas casa, cada qual com 12 crianças, e cada uma envia semanalmente uma carta para cada criança da outra casa(144 cartas enviadas por casa). Cada casa

possui uma criança responsável pela coleta das cartas a serem enviadas e distribuição das cartas recebidas, Ana na casa1 e Bruno na casa2.
O serviço postal oferece uma comunicação lógica entre as casas, enquanto Ana e Bruno oferecem uma comunicação lógica entre os primos.
mensagens de aplicação = cartas em envelope
processos = primos
sistemas finais(hosts) = casas
protocolo da camada de transporte = Ana e Bruno
protocolo de camada de rede = serviço postal (incluindo os carteiros)

-No exemplo acima, os serviços oferecidos por Ana e Bruno são limitados pelos serviços do correio. Se o serviço postal não tem prazo máximo de entrega, então nem Ana e nem Bruno podem oferecer garantia de atraso máximo. Dessa forma vemos que alguns serviços ofertados pelos protocolos da camada de transporte são limitados pelos protocolos da camada de rede, tais como atraso máximo ou garantia de largura de banda, mas pode oferecer serviços não garantidos pela camada de rede, como transferência confiável de dados.

3.1.2 *Visão geral da camada de transporte na Internet*

-O protocolo de camada de rede IP(internet protocol), IP é um serviço de entrega de melhor esforço, o que significa que o IP faz o “melhor esforço” para levar segmentos entre hospedeiros comunicantes, mas não dá nenhuma garantia.

-A responsabilidade fundamental do UDP e do TCP é ampliar o serviço de entrega IP entre dois sistemas finais para um serviço de entrega entre dois processos que rodam nos sistemas finais.

-Tanto o UDP quanto o TCP fornecem uma verificação de integridade ao incluir campos de detecção de erros nos cabeçalhos de seus segmentos.

-O UDP oferece apenas esses dois serviços, entrega de dados e verificação de integridade. Assim como o IP, o UDP é um serviço não confiável.

-Por outro lado, o TCP é um serviço confiável, oferecendo uma **Transferência Confiável de Dados**. Se utiliza de controle de fluxo, números de sequência, reconhecimentos e temporizadores para assegurar que os dados sejam entregues do processo remetente ao processo destinatário corretamente e em ordem.

-TCP também oferece um controle de congestionamento. Permite que outras conexões TCP compartilhem igualmente a largura de banda disponível no enlace.

3.2 Multiplexação e demultiplexação

-Ampliação do serviço de entrega hospedeiro a hospedeiro provido pela camada de rede para um serviço de entrega processo a processo para aplicações que rodam nesses hospedeiros.

-**Demultiplexação**: No host de destino, pode haver várias aplicações rodando ao mesmo tempo, e é trabalho da camada de transporte entregar os vários segmentos que está recebendo aos seus respectivos processos.
-A camada de transporte do hospedeiro destinatário na verdade não entrega dados diretamente a um processo, mas a um socket intermediário. Como pode haver mais de um socket, cada um tem um identificador exclusivo.

-Cada segmento possui no seu cabeçalho um conjunto de campos para que o segmento seja entregue no seu socket apropriado. A camada de transporte examina esses dados e encaminha o segmento para ao socket especificado.

-**Multiplexação**: No host remetente, reúne partes de dados de diferentes sockets, encapsula cada parte com informações de cabeçalho(usados na demultiplexação) para criar segmentos, e passar esses dados para a camada de rede.

-Usando o exemplo da 3.1.1 a Ana realiza um serviço de demultiplexação a entregar cartas às crianças da casa, e um serviço de multiplexação ao agrupar para enviar as cartas das crianças.

-Tanto no TCP quanto no UDP o cabeçalho possui “número de porta de origem” e “número de porta de destino”, para indicar de qual número de socket a mensagem foi enviada, e para qual socket deve ser recebida no hospedeiro destinatário. Alguns número de portas são reservados, tais como 80(HTTP) e 21(FTP).

Multiplexação e demultiplexação não orientadas para conexão

-Numa comunicação UDP, a camada de transporte do host remetente cria um segmento que inclui os dados de aplicação, a porta de origem, a porta de destino e outros dois valores, e envia o segmento para a camada de rede, que faz uma tentativa de melhor esforço para entregar o segmento ao host destinatário.

-Se o segmento chega no host destinatário, a camada de transporte desse host verifica na mensagem qual o socket de destino e irá encaminhá-lo para o seu respectivo socket.

-Um socket UDP é totalmente identificado por uma tupla com dois elementos, consistindo em um endereço IP de destino e um número de porta de destino.

-O número de porta de origem serve para caso o destinatário precise enviar uma resposta ao remetente. Nesse caso, a porta de destino do segmento a ser enviado será a porta de origem do segmento que foi recebido.

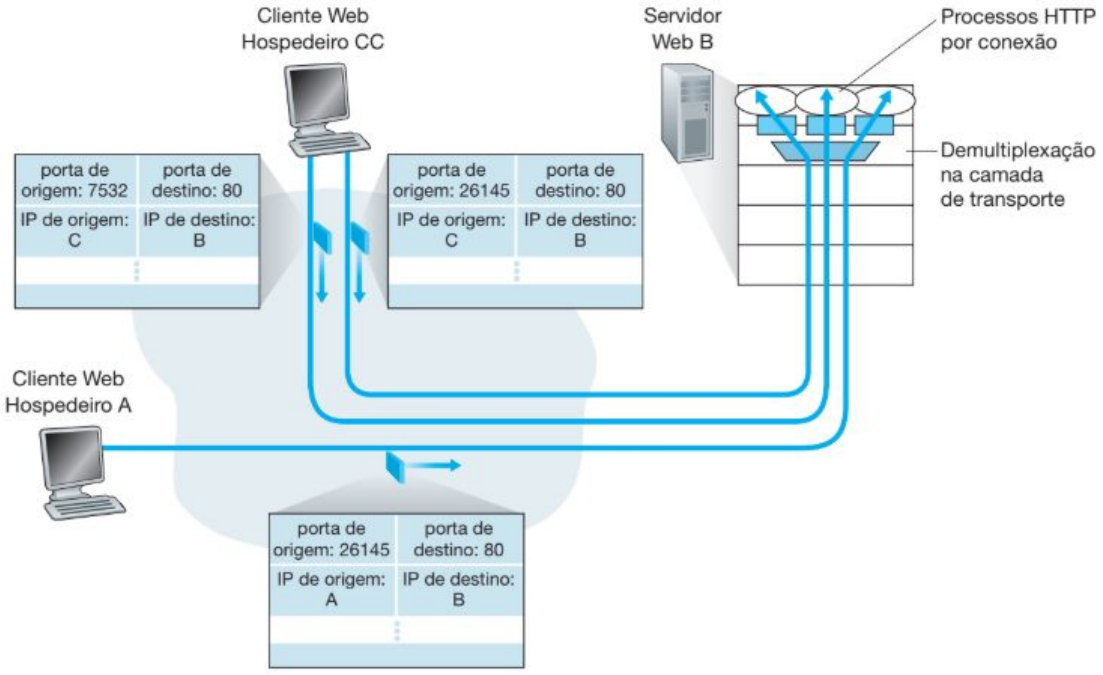
-Caso receba vários segmentos com mesmo número de porta de destino, a conexão UDP irá enviar todos os segmentos para o mesmo socket.

Multiplexação e demultiplexação orientadas para conexão

-Um socket TCP é totalmente identificado por uma tupla de quatro elementos, consistindo em endereço IP de origem, número de porta de origem, endereço IP de destino, número de porta de destino.

-Ao contrário da conexão UDP, na conexão TCP caso haja mais de um segmento requisitando conexão com o mesmo número de porta de destino, a conexão TCP irá criar um novo socket para aquela conexão.

-O hospedeiro servidor pode suportar vários sockets TCP simultâneos. Quando um segmento TCP chega ao hospedeiro, todos os quatro campos de identificação são usados para direcionar (demultiplexar) o segmento para o socket apropriado.



Servidores Web e TCP(achei desnecessário e confuso)

3.3 Transporte não orientado para conexão: UDP

-Como o UDP não faz um “handshake” com o hospedeiro destinatário, dizemos que é um protocolo **Não Orientado para Conexão**.

-Como protocolo de transporte, o UDP faz o mínimo que pode ser feito. Além de multiplexar/demultiplexar as mensagens e incluir uma verificação de erros simples.

-Apesar do TCP parecer melhor que o UDP, algumas aplicações preferem se utilizar do UDP visto que:

- Maior controle no nível de aplicação sobre quais dados são enviados e quando. Não possui um controle de congestionamento, assim “garantindo” uma taxa de envio de dados, e um tempo de entrega menor.
- Não estabelece conexão, assim evitando mais atrasos. O UDP envia as mensagens ao receptor sem se comunicar previamente com este, o que adicionaria mais um atraso.
- Não há estados de conexão. Ou seja, como armazena quase nenhuma informação sobre o estado, os servidores podem suportar mais clientes UDP.
- Os segmentos enviados tem cabeçalhos menores(tamanho de 8 bytes).

-O UDP também é preferível para aplicações que estão em constante modificação (visto que informações perdidas logo serão substituídas) e nas que precisam operar mesmo com redes congestionadas.

-O excesso do uso do UDP pode causar problemas tanto aos clientes UDP quanto aos TCP. Como não possui controle de congestionamento, muitos dados são enviados de uma vez só. Com vários pacotes na rede isso consequentemente congestiona a rede e faz com que pacotes UDP sejam perdidos, e que os clientes TCP reduzam drasticamente suas taxas de envio de dados.

3.3.1 *Estrutura do segmento UDP*

-O cabeçalho UDP possui apenas 4 campos(cada um ocupa 2 bytes), o número de porta de origem e o de destino, comprimento do segmento(quantidade de bytes no segmento, cabeçalho + mensagem), **Checksum** utilizado pelo receptor para verificar se a mensagem contém erros.

-Os dados da aplicação ficam na parte da mensagem no segmento UDP.

3.3.2 *Checksum UDP*

-Serve para verificar se houve alteração de bits no segmento durante sua movimentação da origem ao destino.

-O seu valor é igual ao complemento da soma de todas as palavras de 16 bits que a mensagem contém. No receptor, ele soma todas as palavras de 16 bits, incluindo o checksum, e a soma é pra dar 1111111111111111, se houver algum bit zero, houve mudança de bits no segmento.
-Embora o UDP forneça verificação de erros, ele nada faz para recuperar-se de um erro. Algumas implementações do UDP apenas descartam o segmento danificado; outras passam o segmento errado à aplicação acompanhado de um aviso.

3.4 Princípios da transferência confiável de dados

-É tarefa de um **Protocolo de Transferência Confiável de Dados** transferir dados ao receptor na ordem correta e sem erros. A tarefa pode ser dificultada caso o protocolo da camada abaixo seja não confiável(o que irá ser suposto nessa seção).

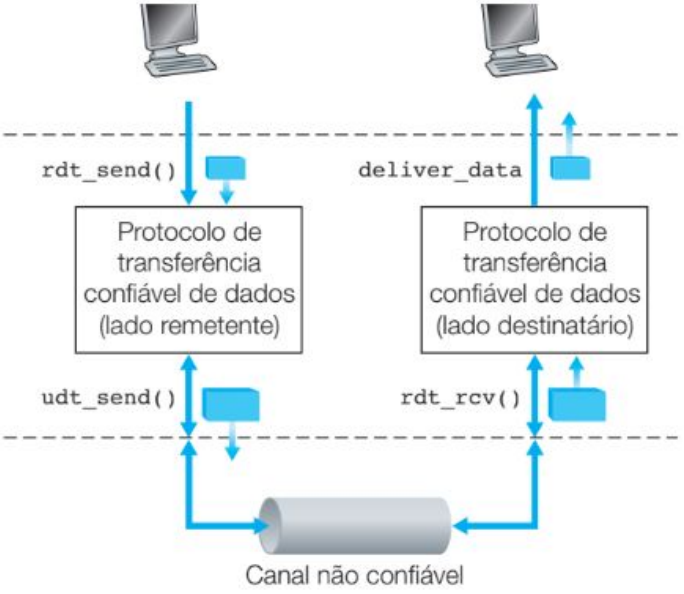
-rdt_send() é invocado pela camada superior e é o envio de data pela camada superior para a camada de transporte. (rdt indica a entrada de algo na camada de transporte).

-udt_send().

-rdt_rcv() é chamado quando um pacote chega no lado destinatário do canal.

-deliver_data() é chamado quando o protocolo rdt quiser entregar dados à camada superior.

-As funções acimas será usadas tanto pelo emissor quanto pelo receptor, visto que ambos enviam e recebem pacotes de informação.

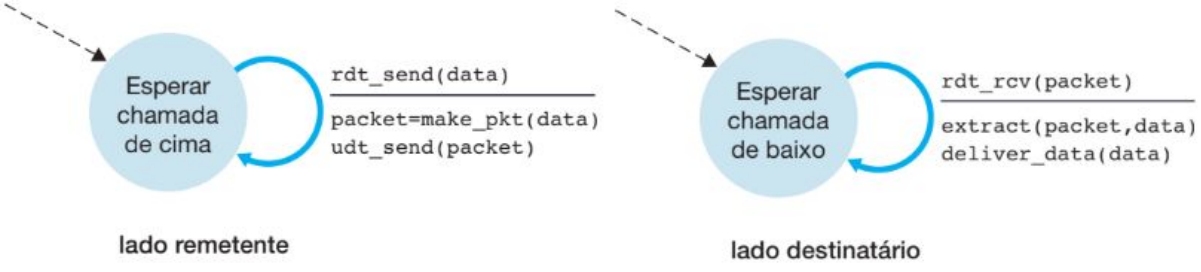


3.4.1 *Construindo um protocolo de transferência confiável de dados*

Transferência confiável de dados sobre um canal perfeitamente confiável: rdt1.0

-Canal subjacente é completamente confiável. Não há perda de pacotes, nem erro nos bits e entrega os pacotes na ordem correta.

-Tanto o remetente quanto o destinatário apresentam **Máquina de Estado Finito**(FSM) distintas. as setas na máquina representam a transição de um estado para outro dentro do protocolo. O evento que causou a transição fica acima da linha, e as ações realizadas ficam abaixo da linha.

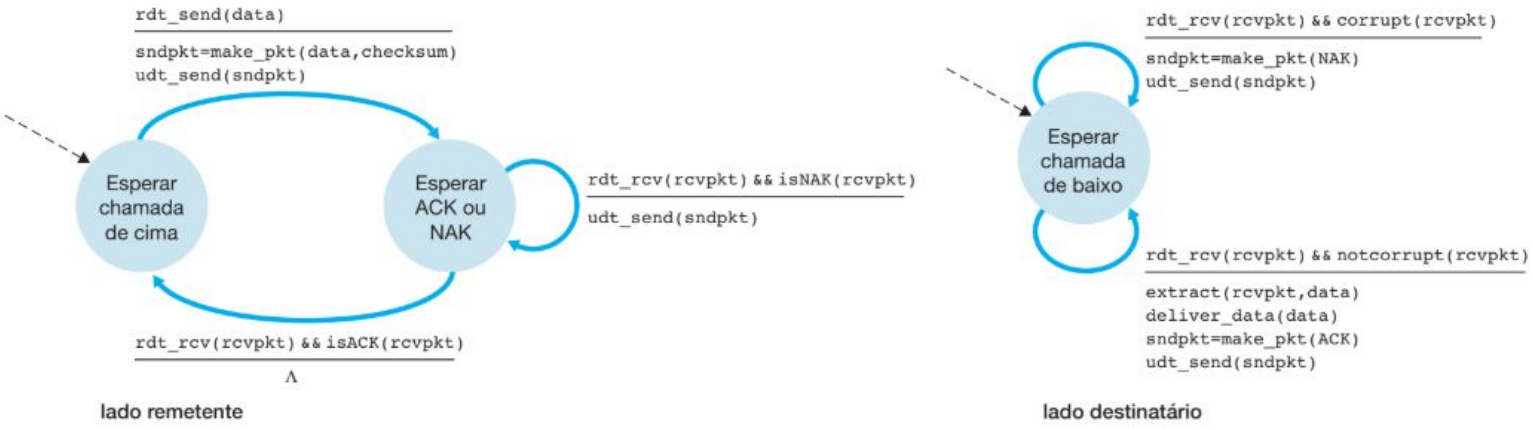


Transferência confiável de dados por um canal com erros de bits: rdt2.0

-O canal subjacente se assemelha ao do rdt1.0, mas esse pode corromper os bits dos pacotes enviados.

-O campo checksum do cabeçalho do pacote serve com a finalidade de verificar se houve troca de bits nos dados do pacote. Para que o remetente(que pode estar longínquo do destinatário) saiba que seu pacote foi recebido com ou sem erros, é necessário que o destinatário mande uma mensagem avisando. Se o remetente receber do destinatário um ACK, a entrega foi bem sucedida, caso contrário receberá um NAK.

-Caso o remetente receba um NAK, o mesmo irá retransmitir o pacote que continha o erro.



-O rdt2.0 é conhecido como um protocolo **Stop-and-Wait** porque ele não pode receber mais dados da camada superior enquanto estiver esperando um ACK/NAK do destinatário.

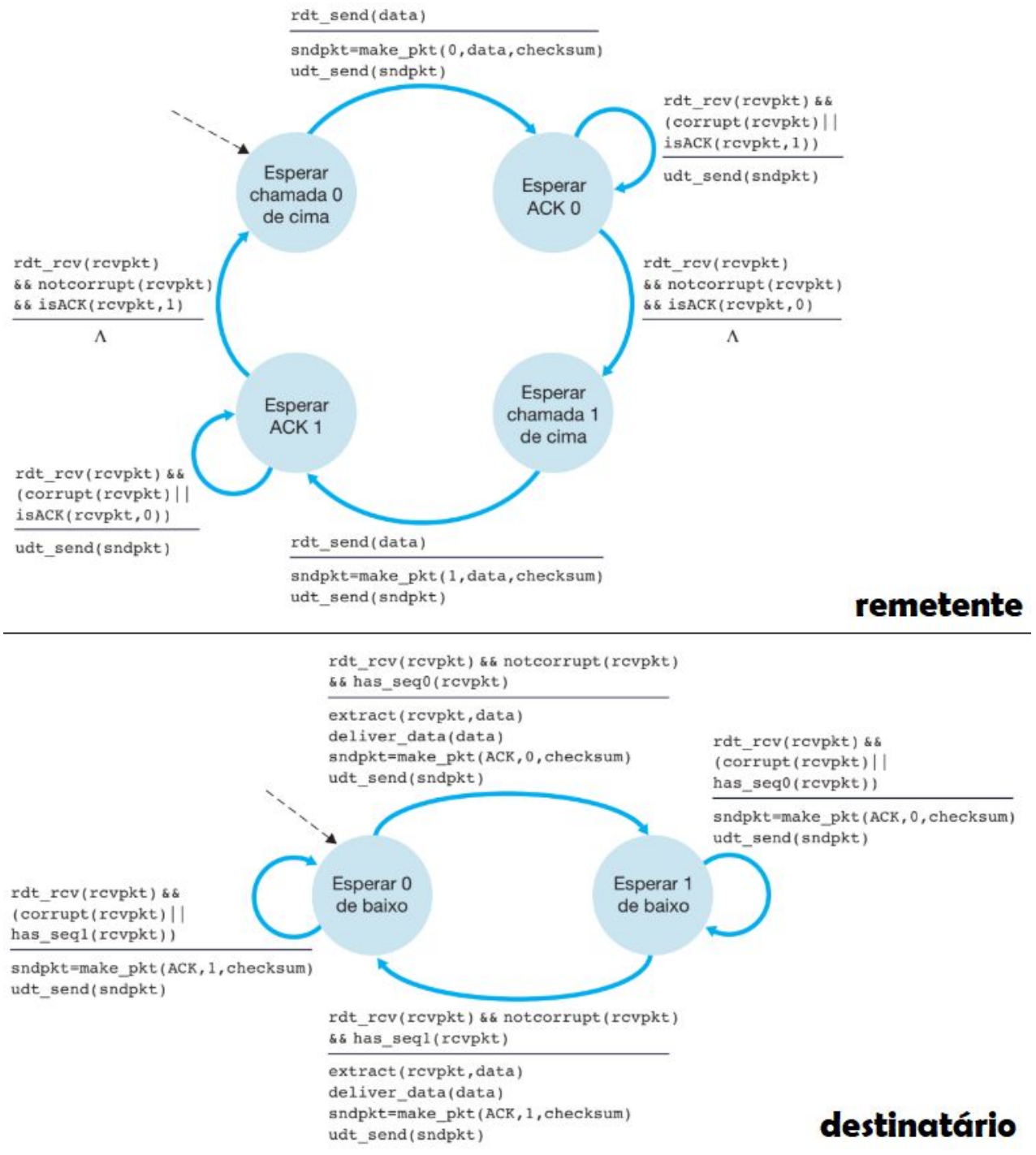
rdt2.1

-Uma falha desse protocolo é que tanto o ACK quanto o NAK também podem ser corrompidos antes de chegar de volta ao remetente. Para resolver esse problema, a solução utilizada é adicionar ao cabeçalho dos pacotes um **Número de Sequência** e retransmitir o pacote caso receba um ACK/NAK corrompido.

-A adição de um número de sequência é necessária para que o destinatário saiba se o pacote recebido se trata de um novo pacote ou de uma retransmissão, assim tratando pacotes duplicados.

rdt2.2

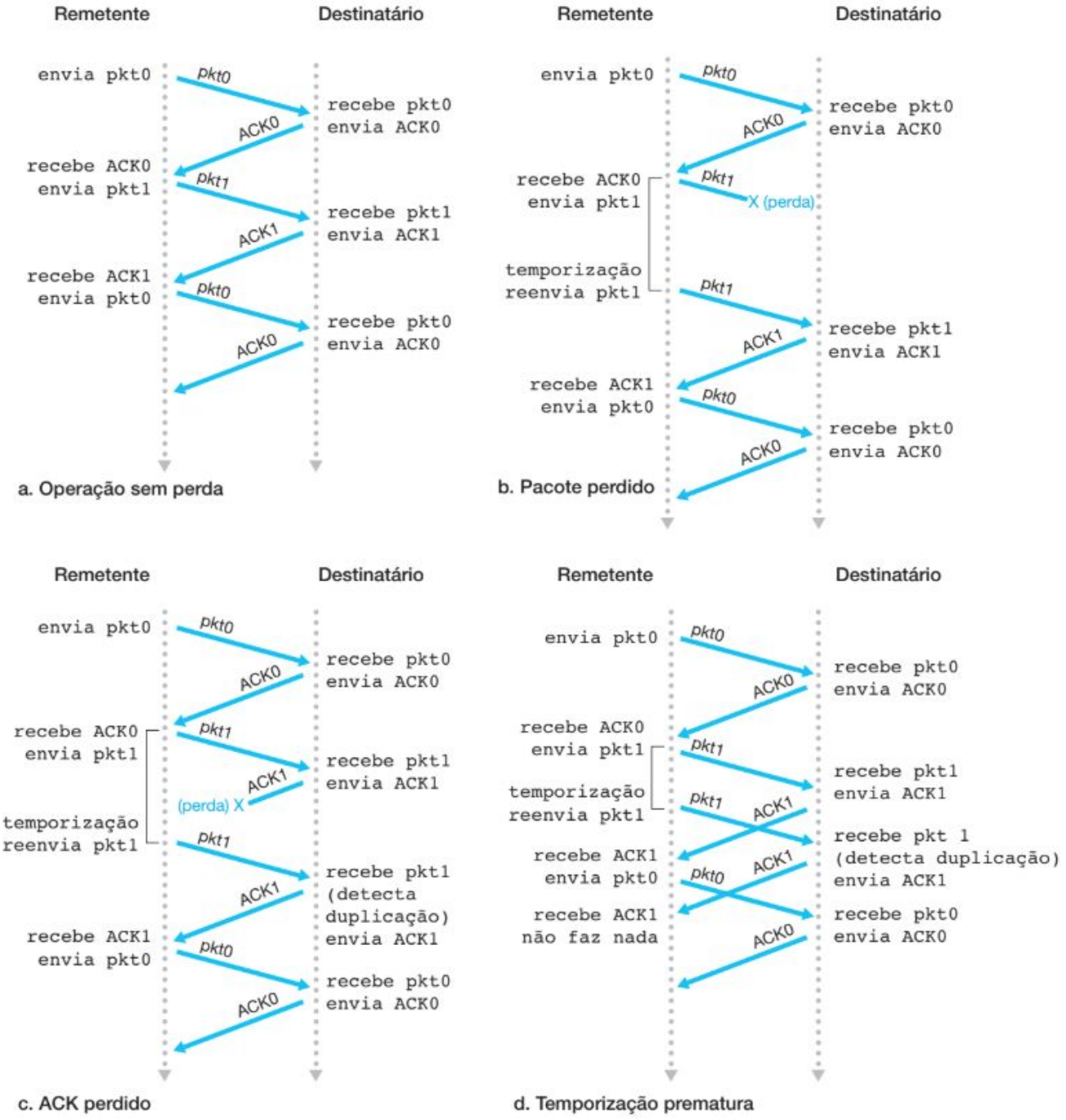
-Semelhante ao rdt2.1, mas não utiliza NAKs, apenas ACKs. Se o remetente recebe dois ACKs seguidos com o mesmo número de sequência, ele identifica que o destinatário não recebeu corretamente o pacote com o número de pacote seguinte. Para que funcione corretamente, a mensagem ACK conterá o número de sequência do pacote que está sendo reconhecido.



Transferência confiável de dados por um canal com perda e com erros de bits: rdt3.0

-O canal subjacente se assemelha ao do rdt2.0, mas esse pode ocorrer a perda de pacotes.

-Após enviar um pacote, o remetente inicia um temporizador para aguardar o ACK do pacote, caso a resposta demore demais(pacote ou ACK perdido), o remetente retransmite o pacote. O temporizador é parado quando acabar o tempo ou receber um ACK. Caso o ACK tenha apenas se atrasado, haverá duplicação de pacotes no destinatário, mas o número de sequência visto no rdt2.2 resolverá esse problema.



3.4.2 *Protocolos de transferência confiável de dados com paralelismo*

-O rdt3.0 apesar de funcional é lento para as redes de hoje devido a ser um protocolo do tipo stop-and-wait. O tempo que o usuário efetivamente passaria ocupado enviando dados é de (L/R)/(RTT + L/R).

-Para melhorar o desempenho, o remetente é autorizado a enviar N pacotes antes de precisar esperar o primeiro ACK.Como há vários pacotes enchendo o canal, essa técnica é chamada de **Pipelining**(paralelismo).

-Visto o aumento no número de pacotes enviados, o paralelismo aumenta a faixa de número de sequências disponíveis para os pacotes.

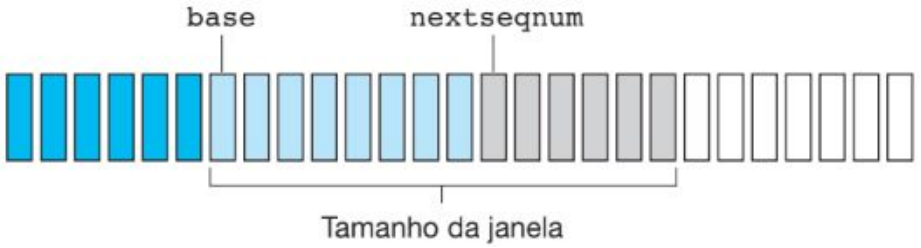
3.4.3 *Go-Back-N (GBN)*

-Nesse tipo de protocolo 4 faixas de números são criadas na faixa de número de sequências [0, …).

- [0, base-1] - pacotes já reconhecidos.
- [base, nextseqnum-1] - pacotes enviados, mas não reconhecidos.
- [nextseqnum, base+N-1] - números de sequência prontos para utilização.
- [base+N, …) - números de sequência que não podem ser utilizados

base é o pacote mais antigo não confirmado, e nextseqnum é o menor número de sequência ainda não utilizado.

-Chama-se de **Janela** os números de sequência que estão na sequência [base, base+N-1], e ele possui N elementos(**tamanho de janela N**). À medida que o protocolo opera a janela anda pra frente.



-No cabeçalho de cada pacote, k bits ficam salvos para armazenar o número de sequência, logo os números de sequência só podem ir de [0, 2^k -1]. Utiliza-se aritmética modular caso se necessite enviar mais de 2^k pacotes.

-O remetente GBN deve saber responder a três tipos de eventos:

1. Chamada de cima: Quando a camada superior chamasse rdt_send(), o remetente primeiro precisa verificar se existe um número de sequência disponível dentro da janela, caso tenha o pacote é enviado e nextseqnum é atualizado, caso contrário, o pacote é devolvido, assim indicando que a janela está cheia.
2. Recebimento de um ACK: GBN trabalha com **reconhecimento cumulativo**, ou seja, um ACK com número n indica que todos os pacote com número de sequência até n(inclusivo) foram corretamente recebidos.
3. Esgotamento(**timeout**): Caso o temporizador do pacote base da janela se esgote, o protocolo GBN irá retransmitir todos os pacotes da janela que foram enviados e não confirmados.

-No lado do destinatário, toda vez que ele recebe um pacote ele enviará um ACK de volta ao remetente. Se o último pacote recebido tinha número n-1 e o atual tem n, o ACK de resposta será de número n e o pacote será entregue à camada superior. Caso contrário(chegue fora de ordem), o pacote é descartado e o ACK de resposta será de número n-1.

3.4.3 *Repetição seletiva (SR)*

-Apesar de funcionar parecido com GBN, este protocolo visa evitar retransmissões desnecessárias, fazendo com que o remetente retransmita apenas pacotes com suspeita de terem sido recebidos com erros.

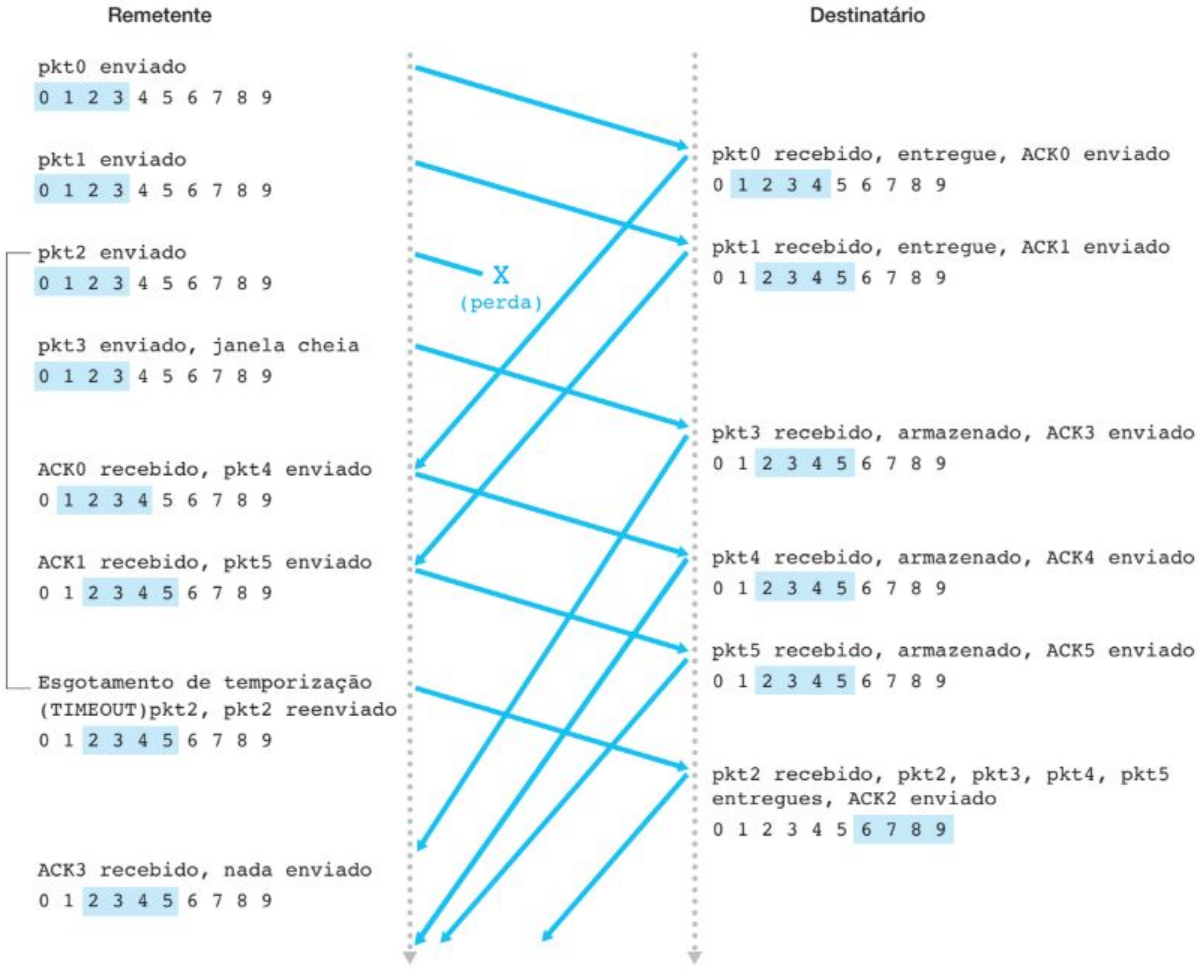
-Agora, tanto no remetente quanto no destinatário terão uma “visão” dos números de sequência. E o tamanho de cada janela será de N.

-No remetente:

- O remetente irá agir da mesma forma do que no GBN ao receber uma chamada da camada superior.
- Cada pacote enviado terá seu próprio temporizador, quando o temporizador se esgotar, apenas aquele único pacote será retransmitido.
- Ao receber um ACK, o remetente irá marcar cada pacote individualmente como recebido. Caso o ACK tenha o número igual a base, a janela do destinatário se moverá para frente até o menor número de sequência que foi enviado mas não reconhecido, e agora poderá enviar pacotes com novos números de sequência.

-No destinatário ao receber um novo pacote:

- Se o número de sequência do pacote estiver entre [rcv_base, rcv_base+N-1], um ACK com o número será enviado ao remetente. Se o pacote não tiver sido recebido anteriormente, ele será armazenado no buffer. Se o número de sequência for igual a rcv_base, ele e os outros pacotes sequenciais armazenados no buffer serão entregues à camada superior, e a janela do destinatário se move de acordo com a quantidade de pacotes entregues.
- Se o número de sequência do pacote estiver entre [rcv_base-N, rcv_base-1], um ACK de mesmo número é devolvido ao remetente.
- qualquer outro caso o pacote é apenas descartado.



-Como o destinatário não sabe o que está acontecendo no destinatário, o tamanho máximo seguro para uma janela é a metade da quantidade de números de sequência, caso contrário, possa ser que o destinatário confunda uma retransmissão com novos dados sendo entregues.

3.5 Transporte orientado para conexão: TCP

3.5.1 *A conexão TCP*

-Diz-se que TCP é orientado para conexão porque antes dos sistemas finais enviarem dados um pro outro, eles precisam antes fazer um **Handshake**, se apresentarem. Enviam segmentos um pro outro para estabelecer alguns parâmetros da transferência.

-O TCP permite que os dados da camada de aplicação fluam de um hospedeiro A para um hospedeiro B, ao mesmo tempo que dados fluam de B para A. Esse tipo de serviço é chamado **Full-Duplex.**

-TCP funciona de **ponto-a-ponto**, ele cria uma conexão entre dois hospedeiros, nunca mais, um remetente e um destinatário.

-Uma vez que a conexão TCP é estabelecida, dados podem ser trocados entre os sistemas finais. Para isso, os sistemas reservam uma porta para o processo e um **Buffer de Envio**(para enviar dados) e um **Buffer de Recepção**(para receber dados), local que serão armazenados os dados até o TCP achar conveniente enviar à camada de rede.

-A quantidade máxima de dados que pode ser retirada e colocada em um segmento é limitada pelo tamanho máximo do segmento (**MSS**).

3.5.2 *Estrutura do segmento TCP*

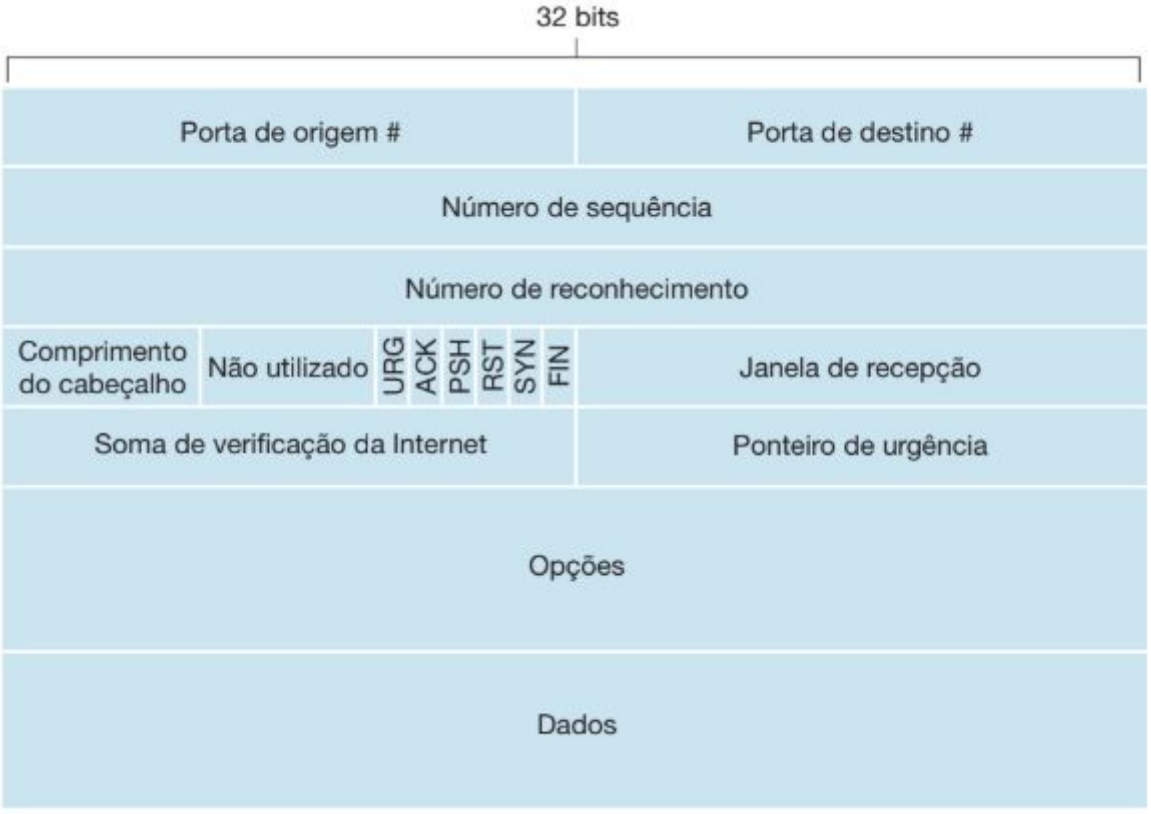
-O segmento é dividido em campo de dados e campo de cabeçalho.

-O campo de dados contêm uma quantidade de dados da aplicação, e seu tamanho é menor ou igual à MMS.

-O campo de cabeçalho de um segmento TCP contém:

- Porta de origem.
- Porta de destino.
- Checksum.
- Número de sequência.
- Janela de recepção(número de bytes que o destinatário está disposto a receber).
- Comprimento de cabeçalho.
- Opções.

- Flags, tamanho de 6 bits. **ACK**: indica se o número de reconhecimento é válido; **RST**, **SYN**, **FIN**: usados para estabelecer e encerrar a conexão; **PSH**: indica que o destinatário deve passar os dados para a camada superior imediatamente; **URG**: usado para mostrar que há dados urgentes nesse segmento.
- Ponteiro de emergência.



Números de sequência e números de reconhecimento

-Número de sequência: número do primeiro byte do segmento.

-Número de reconhecimento: próximo número de sequência que o sistema final local espera receber do sistema final remoto.

-Não há regras para como o TCP deve lidar com segmentos recebidos fora de ordem, essa decisão é tomada pelo programador.

Telnet: um estudo de caso para números de sequência e números de reconhecimento(Não precisa estudar)

3.5.3 *Estimativa do tempo de viagem de ida e volta e de esgotamento de temporização*

-Assim como o protocolo rdt3.0, o TCP também trabalha com temporizadores.

Estimativa do tempo de viagem de ida e volta

-O valor para um timeout tem que ser maior do que 1 RTT, senão haverá várias retransmissões desnecessárias, mas não pode ser muito grande, senão haverá uma reação lenta a perda de pacotes.

-**SampleRTT** é o valor de um RTT para um segmento enviado. Só se mede um sampleRTT para um segmento caso não haja outro sampleRTT sendo contado. Não se calcula sampleRTTs para retransmissões.

-**EstimatedRTT** é uma média ponderada dos sampleRTTs até então calculados. É atualizado para cada novo sampleRTT pela seguinte fórmula:
EstimatedRTT = (1 - α) x EstimatedRTT + α x SampleRTT

-Essa média atribui um peso maior às medidas mais recentes visto que reflete melhor a situação da rede.

-**DevRTT** é o desvio típico entre o sampleRTT e o estimatedRTT
DevRTT = (1 - β) x DevRTT + β x |SampleRTT – EstimatedRTT|

Estabelecimento e gerenciamento da temporização de retransmissão

-O valor do timeout inicial é recomendado ser 1s, mas após a primeira medição de estimatedRTT e sampleRTT, pode ser calculado da seguinte forma:
TimeoutInterval = EstimatedRTT + 4 x DevRTT

3.5.4 *Transferência confiável de dados*

-O TCP cria um **Serviço de Transferência Confiável de Dados** sobre o serviço de melhor esforço do IP. Isso garante que os dados que um processo leia num buffer não estejam corrompido, nem duplicados e em sequência.

-Devido a limitações de hardware, não é possível ter um temporizador para cada segmento enviado. Dessa forma, apenas 1 é ativado por vez. E o intervalo dele é dado pelo timeoutInterval.

-Como o TCP trabalha com ACKs cumulativos, quando é recebido um ACK seu valor é analisado, se for maior do que sendBase, o valor de sendBase é atualizado para o valor do ACK

Alguns cenários interessantes

-Suponha um remetente A e um destinatário B.

1. A envia um segmento de 8 bytes com número de sequência 92. O segmento chega em B sem erros e este envia um ACK 100, porém ele é perdido. Ao acabar o temporizador A irá retransmitir o segmento de 8 bytes com número de sequência 92. B ao receber irá descartar o segmento por já ter recebido e enviará novamente um ACK 100.
2. A inicia o temporizador e envia 2 segmentos, o primeiro de 8 bytes e número de sequência 92, e o segundo de 20 bytes e número de sequência 100. B recebe ambos os segmentos e envia um ACK 100 e um ACK 120 de volta, porém esses chegam no remetente após o timeout, assim houve retransmissão por parte do remetente do primeiro pacote. Ao dar timeout de novo, nenhum pacote será retransmitido, visto que os dois primeiros ACKs chegaram no remetente.
3. Similar ao 2º caso, mas o 1º ACK 100 é perdido e o ACK 120 chega antes do timeout. Como TCP trabalha com ACK cumulativo, ele entende que ao receber o ACK 120, o destinatário recebeu tudo até o byte 119.

Duplicação do tempo de expiração

-Ao dar timeout em um temporizador, o remetente irá retransmitir o segmento com número sendBase e irá duplicar o valor utilizado na última transmissão dele.

-Essa modificação provê uma forma limitada de controle de congestionamento. A causa mais provável da expiração do temporizador é o congestionamento na rede, o que provoca descarte de pacotes e/ou longos atrasos de fila. Se as origens continuarem a retransmitir pacotes de modo persistente durante um congestionamento, ele pode piorar. Em vez disso, o TCP age com mais educação: cada remetente retransmite após intervalos cada vez mais longos.

Retransmissão rápida

-Uma técnica para aumentar a eficiência do tempo de retransmissão é a verificação de **ACK Duplicado**. Como o nome já diz, um ACK duplicado é um ACK que já foi previamente recebido pelo remetente.

-Se o remetente receber 3 ACKs duplicados para o mesmo dado, ele entenderá isso como uma indicação de que aquele segmento foi perdido na rede. Caso isso ocorra, será realizada uma **Retransmissão Rápida**, retransmite o pacote “perdido” antes do timeout do temporizador e reinicia o temporizador.

Go-Back-N ou repetição seletiva?(inútil)

3.5.5 *Controle de fluxo*

-Para que os buffers de recepção não saturem de dados, o TCP oferece um **Serviço de Controle de Fluxo**; Ele iguala a taxa de envio do remetente com a taxa de leitura do destinatário(**Serviço de Casamento de Taxas**).

-O TCP faz com que o remetente mantenha uma variável chamada de **Janela de Recepção** que armazena em si o espaço livre disponível no buffer do destinatário.

-**LastByteRead**: número do último byte na cadeia de dados lido do buffer pelo processo de aplicação em B.

-**LastByteRcvd**: número do último byte na cadeia de dados que chegou da rede e foi colocado no buffer de recepção de B.

-**RcvBuffer**: Espaço do buffer.

-Como o TCP não permite que o buffer estoure, temos:
RcvBuffer ≥ LastByteRcvd - LastByteRead

-A janela de recepção daquele buffer terá um tamanho(**rwnd**).
rwnd = RcvBuffer - (LastByteRcvd - LastByteRead)

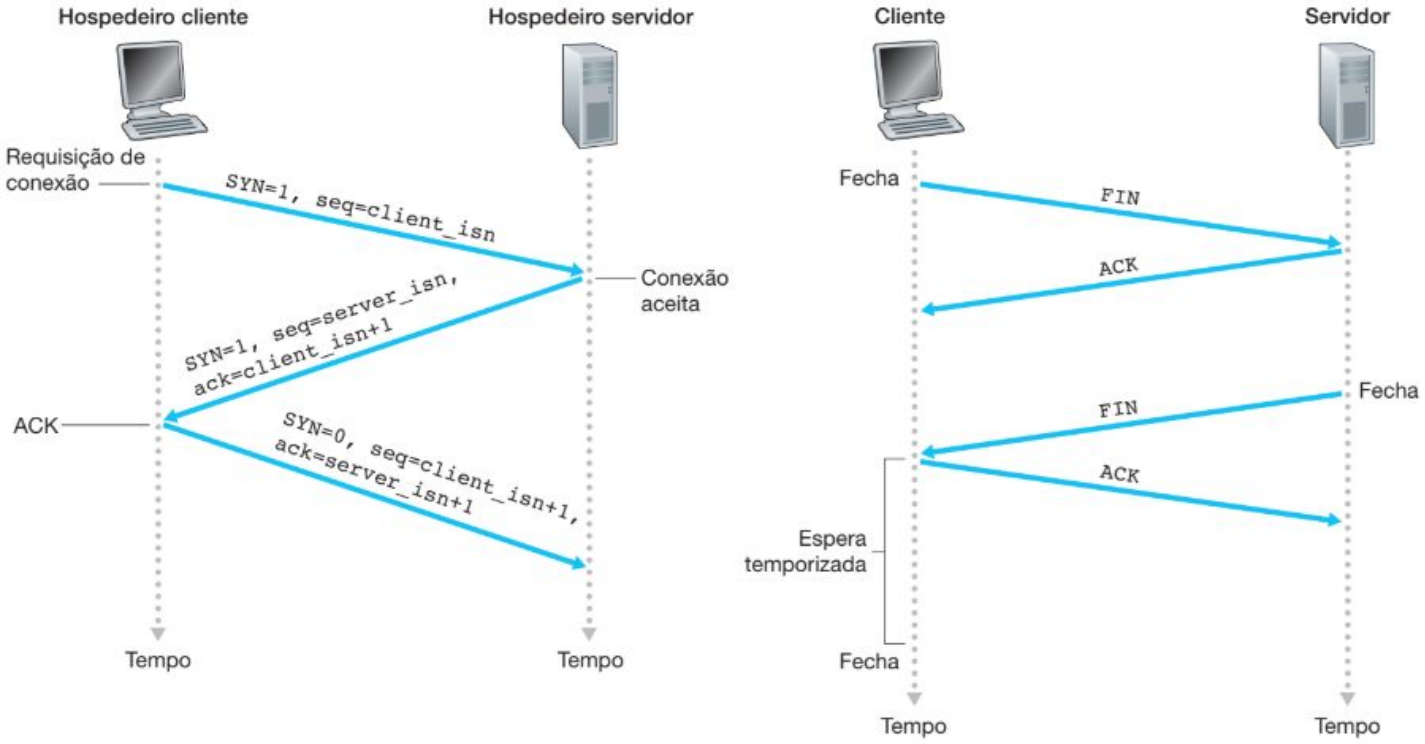
-O remetente envia dados enquanto o tamanho da quantidade de dados enviados mas não reconhecidos for menor do que o tamanho da janela do destinatário.

3.5.6 *Gerenciamento da conexão TCP*

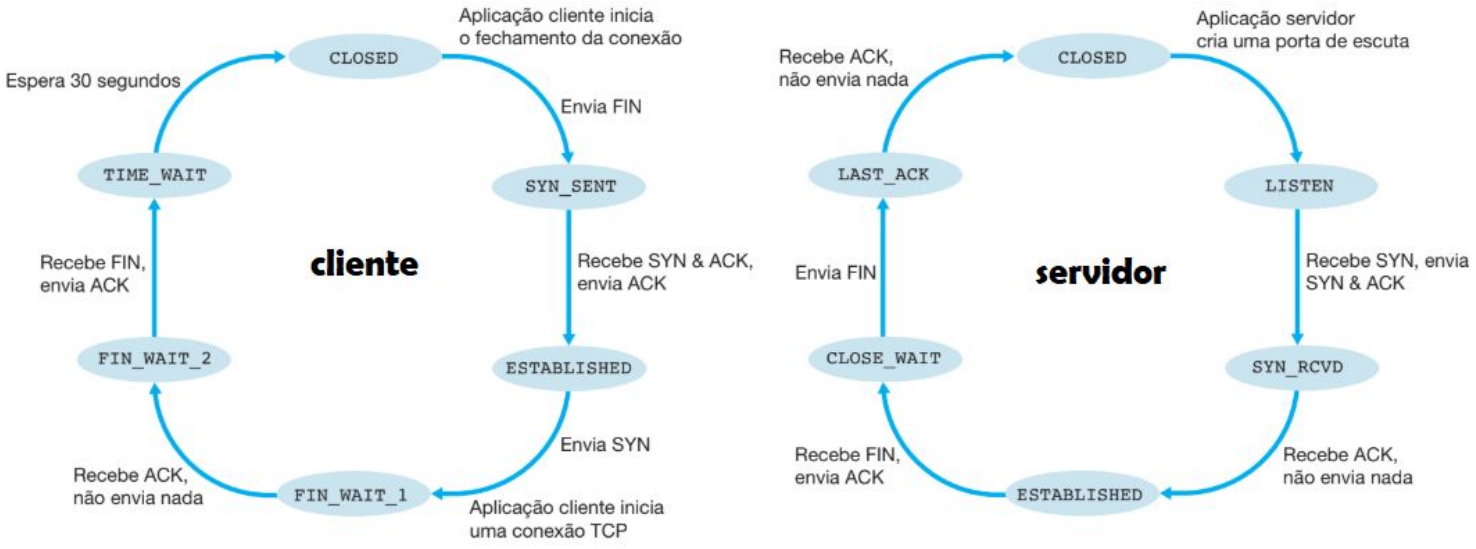
-Para o lado cliente estabelecer uma conexão TCP com o lado servidor é feito uma **Apresentação de Três Vias**(handshaking):

1. O cliente envia um segmento TCP ao servidor sem dados da aplicação, mas a flag SYN do cabeçalho é ajustado para 1 e especifica o número de sequência inicial do cliente(client_isn).
2. Assim que o segmento chega no servidor, aloca buffer e variáveis TCP e envia um SYNACK de volta. O SYNACK não contém dados da aplicação, mas o SYN está setado para 1, o número de reconhecimento do cabeçalho fica pra client_isn+1, e o servidor especifica o número de sequência inicial do servidor(server_isn) no campo número de sequência do cabeçalho.
3. Ao receber o SYNACK, o cliente aloca buffers e variáveis para a conexão. O cliente envia um último segmento reconhecendo o recebimento do SYNACK. O bit SYN fica em 0, server_isn+1 no campo de número de reconhecimento e client_isn+1 no campo de número de sequência.

-Tanto o cliente quanto o servidor podem solicitar o fechamento de uma conexão TCP. Ao fazer isso eles enviam um segmento TCP com a flag FIN setada para 1, o outro lado reconhecerá e enviará uma confirmação, esse então enviará seu próprio segmento TCP com flag FIN setada para 1 e o lado que iniciou a conexão retornará uma confirmação e esperará um tempo antes de efetivamente fechar a conexão. Todos os recursos(buffers e variáveis) são liberados.



-Estados TCP do cliente e do servidor



3.6 Princípios de controle de congestionamento

3.6.1 *As causas e os custos do congestionamento*

Cenário 1: dois remetentes, um roteador com buffers infinitos

-Os hospedeiros transmitem os dados a uma taxa de λ bytes/s e ambos passam pelo mesmo roteador com um enlace de saída de capacidade R.

-À medida que λ cresce, a quantidade de dados novos recebidos no hospedeiro(Θ, **Vazão por conexão**) também cresce. Todavia, dado que os dois remetentes compartilham um enlace de saída de capacidade R, o valor máximo que Θ poderá assumir será R/2, independente de quão alto λ seja. Logo o ideal é fazer $0 < \lambda \leq R/2$.

-Apesar de parecer tentador fazer λ tender a R/2, isso pode causar um grande atraso na entrega dos pacotes.

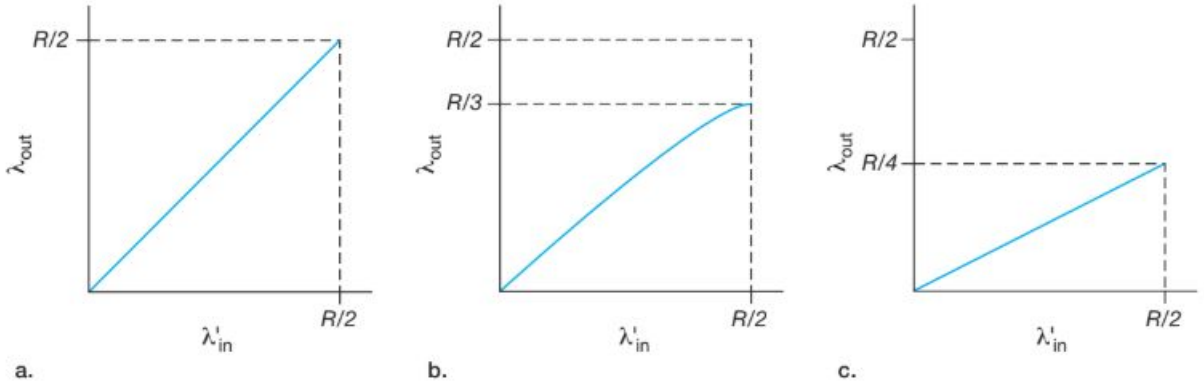
Cenário 2: dois remetentes, um roteador com buffers finitos

-A taxa que o remetente envia dados novos continua sendo λ, e a taxa que envia dados em geral é λ'(**Carga Oferecida**).

-A) Caso o remetente sempre soubesse o espaço livre útil do buffer do roteador, ele nunca enviaria pacotes quando o buffer estivesse cheio. Neste caso não ocorreria perda de pacotes e o cenário 1 se repete.

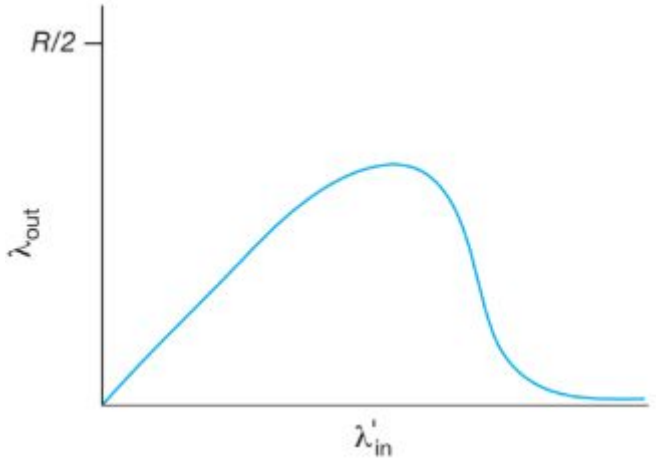
-B) Agora, o remetente só irá realizar retransmissão ao ter certeza de que o pacote foi perdido. À medida que λ' se aproxima de R/2, teremos que Θ tenderá à R/3, ou seja, há uma taxa de R/3 para dados novos e de R/6 para dados retransmitidos.

-C) O temporizador do remetente estoura e retransmite um pacote que ainda está esperando na fila do roteador. O roteador gasta um tempo transmitindo um pacote que não foi perdido, enquanto podia processar um dado novo. Isso faz com que Θ tenda à R/4 à medida que λ' se aproxima de R/2.



Cenário 3: quatro remetentes, roteadores com buffers finitos e trajetos com múltiplos roteadores

-À medida que λ’ aumenta, há uma “briga” entre os remetentes para conseguir espaço nos buffers dos roteadores intermediários. Eventualmente, quando todos os buffers estourarem, pacotes que chegarem ao roteador serão descartados, assim desperdiçando o trabalho realizado por roteadores que processaram o pacote anteriormente.



3.6.2 *Mecanismos de controle de congestionamento*

-**Controle de Congestionamento Fim-a-Fim**: Utilizado quando a camada de rede oferece nenhum suporte para o controle de congestionamento, nem informa aos sistemas finais se a rede está ou não congestionada.

-**Controle de Congestionamento Assistido pela Rede**: Utilizado quando a camada de rede oferece um suporte de controle de congestionamento para o sistema final.

3.6.3 *Exemplo de controle de congestionamento assistido pela rede: controle de congestionamento ATM ABR*

-Fornece um serviço elástico, ou seja, se o caminho emissor-receptor estiver livre, usará toda banda passante disponível, caso contrário irá reduzir sua taxa até o menor valor garantido.

-As **Células de Gerenciamento de Recursos**(células RM) se intercalam com as células de dados e podem ser usadas para portar informações relacionadas ao congestionamento entre hospedeiros e comutadores. Quando a célula RM chega no receptor, esse retornará uma célula RM ao emissor.

-O serviço ABR provê três mecanismos para sinalizar informações relacionadas a congestionamento dos comutadores ao destinatário:

1. **Bits CI e NI**: O bit NI indica se a rede está levemente congestionada, e o bit CI indica se a rede está severamente congestionada. O receptor retorna esses bits inalterados para o emissor.
2. **Bits EFCI**: Se a célula de dados precedendo a célula RM tiver RFCI setado, emissor seta bit CI na célula RM retornada.
3. **Campo ER**: Abaixa a taxa de envio do emissor para a menor permitida.

3.7 Controle de congestionamento no TCP

-O TCP utiliza controle de congestionamento fim-a-fim dado que a camada IP não fornece aos sistemas finais realimentação explícita relativa ao congestionamento da rede.

-**cwnd** impõe um limite ao remetente de quantos dados podem ser enviados à rede. Dessa forma a quantidade de dados enviados e não confirmados deverá ser menor do que o mínimo entre cwnd e rwnd.

-O remetente identifica um congestionamento na rede ao ser realizado um evento de perda de pacote(timeout de um temporizador ou recebimento de três ACKs duplicados).

-O TCP é **autorregulado**. Ele se utiliza dos atrasos dos ACKs para regular o tamanho de cwnd. Se o atraso for alto, a janela será diminuída; Se o atraso for baixo, a janela será aumentada.

-Para que o TCP defina a taxa de envio, ele segue alguns princípios:

- Um segmento perdido implica congestionamento, portanto, a taxa do remetente TCP deve diminuir quando um segmento é perdido.
- A taxa do remetente pode aumentar quando um ACK chegar para um segmento não reconhecido antes.
- Busca por largura de banda. Enquanto ACKs novos estiverem chegando, a taxa de transmissão vai aumentando até que ocorra um evento de perda de pacote, momento no qual a taxa de transmissão diminui.

Partida lenta

-Ao se inicializar uma conexão TCP, é atribuído 1 MSS ao valor de cwnd, e consequentemente a taxa de envio inicial fica MSS/RTT.

-Como a largura de banda disponível pode ser maior do que MSS/RTT, para que a largura de banda aumente rapidamente, o valor de cwnd começa em 1 MSS e é duplicado para cada RTT.

-O processo de partida lenta é finalizado ao ocorrer um evento de perda de pacote(ocasionado por um timeout de temporizador). Estabelece o valor de **ssthresh**(slow start threshold) para cwnd/2, cwnd volta a ser 1 MSS e o processo de partida lenta é reiniciado.

-Também pode ser finalizado quando cwnd ≥ ssthresh. A partir daqui o modo do TCP passa a ser de **Prevenção de Congestionamento**.

-Por fim, quando o remetente receber 3 ACKs duplicados, o TCP realizará uma retransmissão rápida e entrará no modo **Recuperação Rápida**.

Prevenção de congestionamento

-Neste modo, ao invés de duplicar o valor de cwnd a cada RTT, se incrementa o valor de cwnd em 1 MSS a cada RTT.

-Quando ocorre um evento de perda de pacote(ocasionado por timeout de temporizador), o comportamento é exatamente o mesmo do da partida rápida.

-Quando o remetente receber 3 ACKs duplicados, o TCP irá tornar ssthresh = cwnd/2, cwnd = cwnd/2, e entrará no modo de Recuperação rápida.

Recuperação rápida

-O valor de cwnd é aumentado em 1 MSS para cada ACK duplicado que causou o TCP a entrar neste modo.

-Quando um ACK novo chegar, o TCP fará cwnd = ssthresh e alterará para o modo prevenção de congestionamento.

-Caso ocorra um evento de perda de pacote(timeout de temporizador), o TCP fará ssthresh = cwnd/2, cwnd = 1 e alterará para o modo partida lenta.

Controle de congestionamento no TCP: retrospectiva

-O controle de congestionamento do TCP é denominado aumento aditivo, diminuição multiplicativa (**AIMD**), visto que ele aumenta linearmente, e decresce pela metade.

Descrição macroscópica da vazão do TCP

-A taxa média de envio de uma conexão TCP é $\frac{0,75*W}{RTT}$, onde W é o valor da janela quando ocorre um evento de perda de pacotes. Isso ocorre devido à natureza AIMD da conexão TCP, tornando a taxa de transmissão variando entre $\frac{W}{RTT}$ e $\frac{W}{2*RTT}$.

TCP por caminhos com alta largura de banda

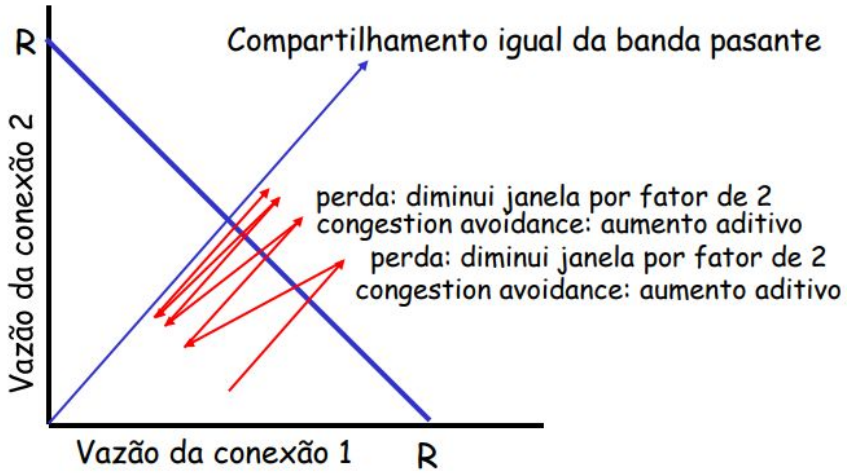
-Sendo L a taxa de perda temos:

vazão média de uma conexão = $\frac{1,22*MSS}{RTT*\sqrt{L}}$

3.7.1 *Equidade*

-A conexão TCP divide igualmente a banda passante R disponível no enlace de gargalo entre as K conexões que passam por ali.

-Independente de qual sistema final iniciou a conexão primeiro, após algum tempo, as taxas de todos as conexões irão se igualar visto que o TCP opera sob um controle de congestionamento do tipo AIMD.



Equidade e UDP

-Do ponto de vista do TCP, o UDP é nada justo, visto que ele não cooperam com as outras conexões nem ajustam suas taxas de transmissão de maneira a “pensar” nos outros usuários.

-Como UDP envia dados a uma taxa constante independente da situação da rede, isso pode reduzir as taxas de transmissão do TCP.

Equidade e conexões TCP paralelas

-Mesmo com a equidade da conexão TCP, nada impede que uma aplicação abra mais de uma conexão TCP paralelamente. Quanto mais conexões uma aplicação tiver, maior será a sua fração de largura de banda.

EXEMPLO:
Em um enlace de taxa R há 10 conexões TCP, cada conexão com R/10 de taxa. Caso uma nova aplicação requisite uma conexão TCP, este receberá R/11 de taxa(e as outras conexões irão atualizar para R/11). Mas caso uma nova aplicação requisite 10 conexões TCP, este receberá R/2, enquanto as outras conexões irão dispor de apenas R/20.

4ºcapítulo(camada de rede)

4.1 Introdução

-Para um hospedeiro H1 enviando informações para um hospedeiro H2, H1 irá encapsular o segmento da camada de transporte em um **Datagrama** e irá enviá-lo ao seu roteador vizinho R1. Em H2, quando o seu roteador vizinho R2 receber o datagrama, a camada de rede irá extrair o segmento do datagrama e irá entregá-lo à camada de transporte de H2.

-O papel primordial dos roteadores é repassar datagramas de enlaces de entrada para enlaces de saída.

4.1.1 *Repasse e roteamento*

-Para seu correto funcionamento, identifica-se duas funções da camada de rede:

- **Repasse**: Quando um pacote chega ao enlace de entrada de um roteador, este deve conduzi-lo até o enlace de saída apropriado.
- **Roteamento**: Os algoritmos que calculam o caminho do pacote de um remetente ao destinatário são denominados **Algoritmos de Roteamento**.

-Cada roteador tem uma **Tabela de Repasse**. Ao chegar um novo pacote no roteador, este irá verificar no cabeçalho do pacote um valor em determinado campo. Ao descobrir o valor, o roteador irá verificar na sua tabela a qual enlace de saída esse valor está associado e irá repassar o pacote para o enlace de saída apropriado.

-As tabelas de repasse de um roteador são configuradas a partir de mensagens recebidas do algoritmo de roteamento.

-**Comutadores de Pacotes** são dispositivos genéricos de transferência de pacotes de um enlace de entrada para um enlace de saída:

- **Comutadores** (de Camada de Enlace): Baseiam a decisão de repasse no valor que está no campo da camada de enlace.
- **Roteadores**: baseiam sua decisão de repasse no valor que está no campo de camada de rede.

Estabelecimento de conexão

-Para algumas arquiteturas da camada de rede é necessário que os host com auxílio do roteadores intermediários estabeleçam entre si um **Circuito Virtual**(**CV**) antes do envio de fluxo de datagramas.

4.1.2 *Modelos de serviço de rede*

-O **Modelo de Serviço de Rede** define as características do transporte de dados fim-a-fim entre hosts remetente e destinatário.

Arquitetura da rede	Modelo de serviço	Garantia de largura de banda	Garantia contra perda	Ordenação	Temporização	Indicação de congestionamento
Internet	Melhor esforço	Nenhuma	Nenhuma	Qualquer ordem possível	Não mantida	Nenhuma
ATM	CBR	Taxa constante garantida	Sim	Na ordem	Mantida	Não haverá congestionamento
ATM	ABR	Mínima garantida	Nenhuma	Na ordem	Não mantida	Indicação de congestionamento

-Garantia de Temporização é se o tempo entre o envio de dois pacotes sucessivos é o mesmo que o tempo entre o recebimento dos mesmos pacotes. A diferença entre esse tempos é chamada de **Jitter**.

4.2 Redes de circuitos virtuais e de datagramas

-Assim como na camada de transporte, a camada de rede pode disponibilizar um serviço **Orientado para Conexão**, e um **Não Orientado para Conexão**.

-Redes que ofereçam um serviço orientado para conexão são denominados **Redes de Circuitos Virtuais**(redes CV), enquanto os que não oferecem são denominados **Redes de Datagramas**.

4.2.1 *Redes de circuitos virtuais*

-Um **Circuito Virtual(CV)** consiste em:

1. Uma rota(série de enlaces e roteadores) entre o remetente e o destinatário.
2. **Números de CV**. Um número para cada enlace ao longo da rota.
3. Registros na tabela de repasse em cada roteador ao longo do caminho.

-Um pacote no circuito portará um número de CV em seu cabeçalho. Como um enlace pode ter um número de CV diferente para cada circuito virtual, cada roteador ao longo da rota deverá atualizar o número de CV no pacote.

Interface de entrada	Nº do CV de entrada	Interface de saída	Nº do CV de saída
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

-Em um CV, os roteadores devem manter **Informação de Estado de Conexão** para as conexões em curso. Cada vez que uma nova conexão é estabelecida, um novo registro de conexão deve ser adicionado à tabela de repasse do roteador. E cada vez que uma conexão é encerrada, um registro deve ser removido da tabela.

-Fases em um CV:

1. Estabelecimento de CV: A camada de transporte emissora contata a camada de rede, especifica o endereço do receptor e espera até a rede estabelecer o CV. A camada de rede determina o caminho pelos quais todos os pacotes do CV trafegarão, determina o número de CV para cada enlace ao longo do caminho e adiciona um registro na tabela de repasse em cada roteador no caminho. A camada de rede pode também reservar recursos no caminho (dependendo do serviço).
2. Transferência de dados: Assim que estabelecido o CV, pacotes podem fluir por ele.
3. Encerramento do CV: O remetente ou destinatário informa à camada de rede que deseja encerrar a conexão CV. A camada de rede informará ao outro lado que deseja encerrar o CV e atualizará as tabelas de repasse dos roteadores intermediários indicando que aquele CV deixou de existir.

-As mensagens que os sistemas finais enviam para iniciar ou encerrar um CV são conhecidas como **Mensagens de Sinalização** e os protocolos para trocá-las como **Protocolos de Sinalização**.

4.2.2 *Redes de datagramas*

-Em uma **Rede de Datagramas**, toda vez que um sistema final quer enviar um pacote, ele marca o pacote com o endereço do sistema final de destino e então o envia para dentro da rede. Roteadores em uma rede de datagramas não mantém informações de estado sobre CV, visto que não se utiliza número de CV.

-Diferente dos circuitos virtuais, quando um pacote chega em um roteador, o roteador examina na sua tabela de repasse qual interface de enlace de saída está associado o endereço do sistema final de destino do pacote, e então o repassa o pacote pro seu devido enlace.

-Há mais de 4 bilhões de endereços disponíveis para endereços de 32 bits, as tabelas de repasse funcionam em faixas de prefixos de endereços. Se houver concordância em mais de 1 prefixo, a prioridade é dada para o prefixo mais longo.

Prefixo do endereço	Interface de enlace
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
senão	3

-Como as tabelas de repasse em uma rede de datagramas podem ser atualizadas a qualquer momento, é possível que uma série de pacotes enviados de um remetente sigam rotas diferentes até o destinatários, assim podendo desordena-los.

4.2.3 *Origens das redes de circuitos virtuais e de datagramas(achei inútil)*

4.3 O que há dentro de um roteador?

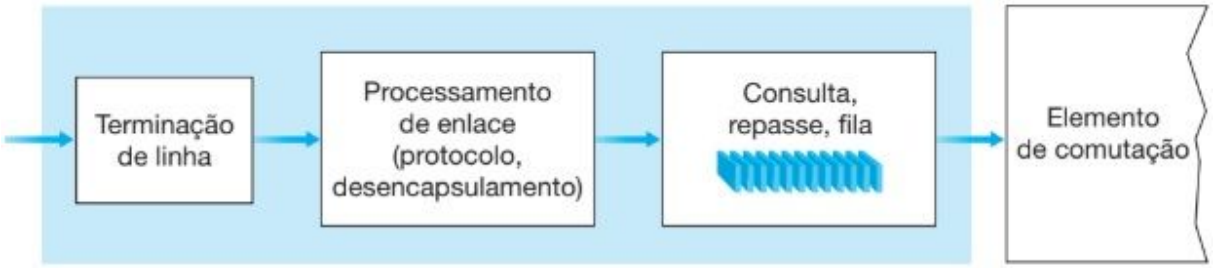
-**Portas de Entrada**: Ela realiza as funções de camada física de terminar um enlace físico de entrada em um roteador. Executa também as de camada de enlace necessárias para interoperar com as funções da camada de enlace do outro lado do enlace de entrada. E a função de exame também é realizada na porta de entrada; Aqui que a tabela de repasse é consultada para redirecionar o pacote para a porta de saída correta. Pacotes de controle são repassados de uma porta de entrada até o processador de roteamento.

-**Elemento de Comutação**: Conecta as portas de entrada do roteador às portas de saída.

-**Portas de Saída**: Armazena os pacotes que foram repassados para ela e os transmite para o enlace de saída após realizar as funções de camada de enlace e camada física. Quando o enlace é bidirecional, a porta de saída pode ser também vista como uma porta de entrada, e vice-versa.

-**Processador de Roteamento**: Executa os protocolos de roteamento, mantém as tabelas de roteamento e as informações de estado do enlace, e calcula e atualiza a tabela de repasse para o roteador.

4.3.1 *Processamento de entrada*



-Tem a função de **Terminação de Linha**, ou seja, realiza o papel de receber os bits(implementação da camada física).

-Tem a função de **Processamento de Enlace**, ou seja, implementa a camada de enlace.

-Tem a função de **Consulta**, **Repasse** e **Fila**. É aqui que o roteador usa a tabela de repasse para determinar a porta de saída para a qual o pacote será encaminhado.

-Cada porta de entrada tem sua própria cópia da tabela de repasse. Como a porta tem sua própria tabela, as decisões de repasse podem ser feitas localmente, sem sobrecarregar o processador de roteamento(**Comutação Descentralizada**).

-Um pacote que já tenha tido sua porta de saída determinado pode ser impedido de entrar no elemento de comutação caso o elemento esteja ocupado transitando outros pacotes. O pacote é enfileirado até que seja liberado o seu envio.

4.3.2 *Elemento de comutação*

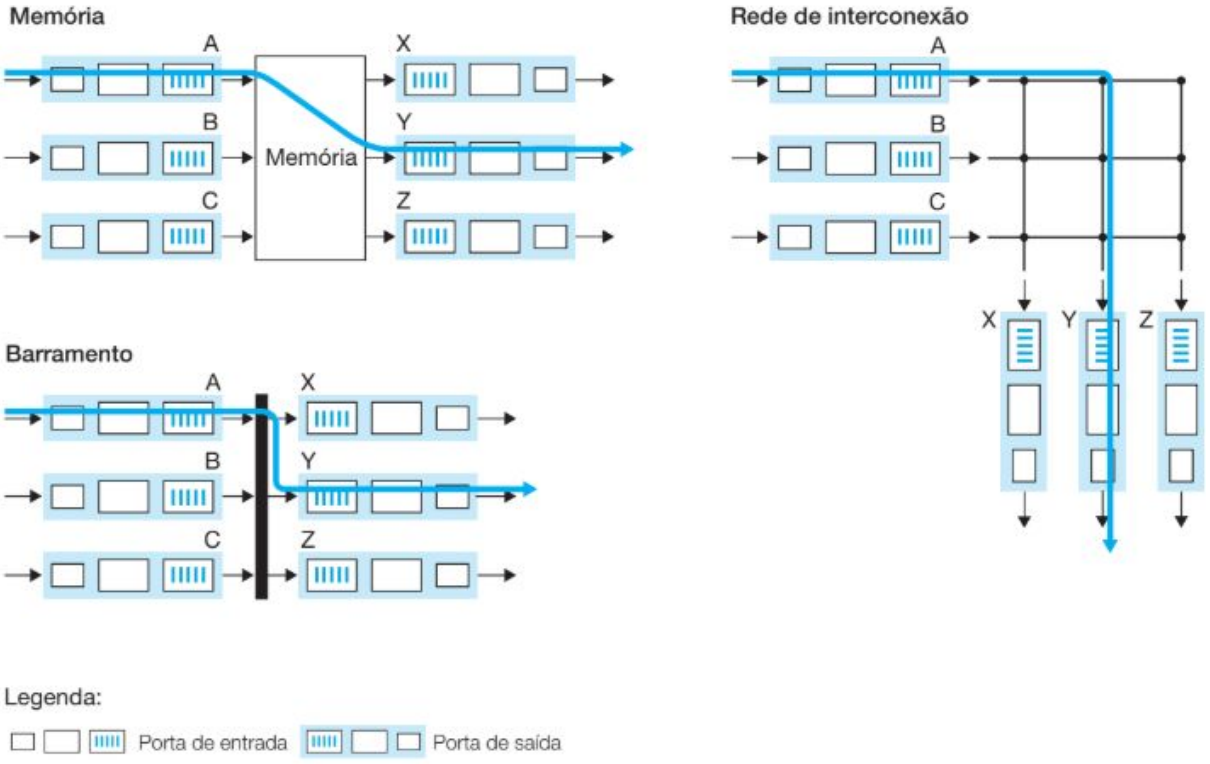
-É o elemento do roteador que comuta(repassa) os pacotes das portas de entrada para as portas de saída.

-Essa comutação pode ser feita de diversas formas, as mais comuns sendo **Comutação por Memória**, **Comutação por um Barramento**, **Comutação por uma Rede de Interconexão**.

-Na **Comutação por Memória** o pacote na porta de entrada é copiado para a memória do processador. O processador de roteamento extraí o endereço de destino do cabeçalho do pacote e então copia o pacote para a porta de saída apropriada. A velocidade é limitada pela banda passante da memória.

-Na **Comutação por um Barramento** a porta de entrada adiciona um rótulo ao pacote para indicar a porta de saída correta. O pacote então é recebido por todas as portas de saída, mas somente a porta que combina com o rótulo manterá o pacote. Na porta de saída o rótulo é removido do pacote. Como só 1 pacote pode atravessar o barramento por vez, a velocidade é limitada pela banda passante do barramento.

-Na **Comutação por uma Rede de Interconexão** consiste em 2^n barramentos que conectam n portas de entrada com n portas de saída. As entradas se conectam com as saídas através da abertura e fechamento de cruzamentos. Se as portas de saída não forem as mesmas, é possível transferir vários pacotes simultaneamentes.



4.3.3 *Processamento de saída*

-Toma os pacotes que foram armazenados na memória da porta de saída e os transmite pelo enlace de saída. Realiza também as funções de transmissão necessárias nas camadas de enlace e física.

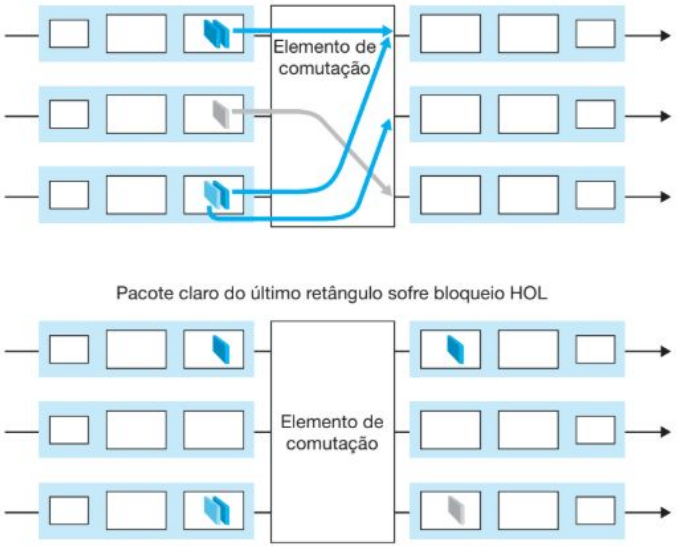
4.3.4 *Onde ocorre formação de fila?*

-Filas de pacote podem se formar tanto na porta de entrada quanto na porta de saída. À medida que a fila aumenta, eventualmente a memória do roteador será estourada e poderá ocorrer **Perda de Pacotes**.

-Na porta de saída haverá um **Escalonador de Pacotes** que irá decidir qual dos pacotes irá ser transmitido por vez. O escalonador fornece **Garantia de Qualidade de Serviço**.

-Se o elemento de comutação não for veloz o suficiente (em relação às taxas da linha de entrada) para transmitir todos os pacotes que chegam através dele, então poderá haver formação de fila também nas portas de entrada.

-Um datagrama na porta de entrada pode prevenir outros datagramas na fila atrás dele de serem enviados. Esse fenômeno é chamado de **Bloqueio de Cabeça de Fila**(**HOL**). Mesmo que sua porta de saída esteja livre, o datagrama na fila tem de esperar o datagrama mais na frente da fila a ser enviado antes.



4.3.5 *O plano de controle de roteamento(julguei desnecessário)*

4.4 O Protocolo da Internet (IP): repasse e endereçamento na Internet

4.4.1 *Formato de datagrama*

-O datagrama possui 14 campos:

- **Número da versão:** Especifica a versão do protocolo IP.
- **Comprimento do cabeçalho:** Indicam onde começam os dados. Tipicamente tem um valor de 20 bytes.
- **Tipo de serviço:** Diferencia os diferentes tipos de datagrama.
- **Comprimento do datagrama:** É o tamanho total do datagrama em bytes.

- **Identificador, flags, deslocamento de fragmentação:** Esses três campos têm a ver com a fragmentação do IP.
- **Tempo de vida:** Usado para não permitir que um datagrama fique circulando na rede por muito tempo. Esse campo é decrementado de uma unidade cada vez que o datagrama é processado por um roteador. Se esse campo chegar a 0, o datagrama deve ser descartado.
- **Protocolo:** Informa ao destinatário qual protocolo de transporte o datagrama deve ser passado.
- **Checksum:** Trata cada 2 bytes **do cabeçalho** como se fosse um número e os soma usando complemento a 1 aritmético. O resultado da soma é armazenado no checksum. Um roteador calculará o checksum para cada datagrama, se o valor diferir do valor no campo checksum será identificado um erro. O campo do Checksum deve ser atualizado para cada roteador, visto que o valor do campo tempo de vida é decrementado em 1, e talvez haja mudança em outros campos.
- **Endereços IP de origem e de destino:** IP do host remetente e IP do host destinatário.
- **Opções:** Permite que um cabeçalho IP seja estendido. Esse campo foi descartado no cabeçalho da versão IPv6.
- **Dados:** Normalmente carrega um segmento da camada de transporte(TCP/UDP).

-O cabeçalho do datagrama tem ao menos 20 bytes.

Fragmentação do datagrama IP

-Diferentes protocolos da camada de rede podem suportar transferir diferentes tamanhos de pacotes. A quantidade máxima de dados que um quadro da camada de enlace pode carregar é denominada unidade máxima de transmissão(**MTU**).

-Quando um datagrama tem tamanho maior do que o MTU, o roteador fragmenta o pacote; divide o datagrama em vários datagramas de tamanhos menores, e cada pedaço é chamado de **Fragmento**.

-O sistema final destinatário fica encarregado de remontar os fragmentos. Quando o destinatário recebe um datagrama fragmento ele tem que determinar se recebeu o último fragmento e como deve reconstruir os fragmentos para retomar o datagrama original.

-Quando um datagrama é criado, o hospedeiro remetente marca o datagrama com um número de identificação, bem como com os endereços de origem e de destino. Quando um roteador precisa fragmentar um datagrama, cada fragmento é marcado com o endereço de origem, o endereço do destino e o número de identificação do datagrama original.

-Para o destinatário saber se recebeu todos os fragmentos do datagrama, o último fragmento tem o bit de flag no cabeçalho ajustado para 0, enquanto os outros têm ajustado para 1.

-Para o destinatário determinar se falta algum fragmento, o campo de deslocamento(**offset**) é utilizado para indicar a posição exata do fragmento no datagrama IP original.

-O campo de offset é calculado a partir do campo **length**. Esse campo informa o tamanho total do datagrama em bytes(cabeçalho+dados). Caso seja o primeiro fragmento, o offset será 0, senão o offset terá seu valor ajustado para o (offset do fragmento anterior) + (valor no campo length - tamanho do cabeçalho em bytes)/8; ou seja:

$$offset(n) = offset(n-1) + \frac{length(n) - cabeçalho(n)}{8}$$

-No destinatário, se um ou mais fragmentos não chegarem, o datagrama incompleto será descartado e não será passado à camada de transporte.

4.4.2 *Endereçamento IPv4*

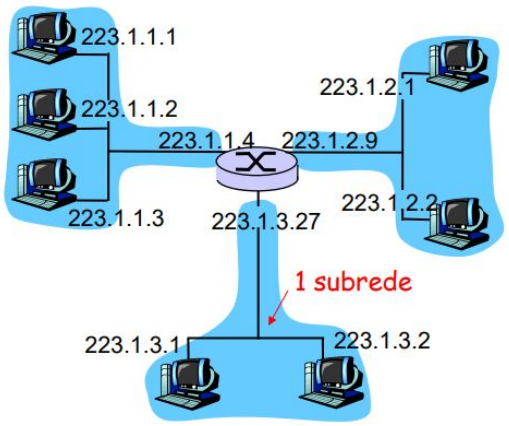
-A fronteira entre um roteador e um enlace, ou de um host e um enlace é denominada **Interface**. Sistemas finais costumam ter apenas 1 interface, enquanto roteadores têm múltiplas(mas poucas) interfaces.

-Cada interface tem seu próprio endereço IP, logo o IP está associado à interface, e não com um hospedeiro ou roteador.

-O endereço IP é composto de 32 bits, e esse é escrito em **Notação Decimal Separada por Pontos**, Na qual cada byte do endereço é escrita em notação decimal e é separado dos outros bytes do endereço através de um ponto.
193.32.216.9 = 11000001 00100000 11011000 00001001

-O endereço IP é “separado” em um prefixo e em um sufixo, onde o prefixo identifica a subrede que na qual o host se encontra, e o sufixo identifica qual o host na subrede.

-Uma subrede é um conjunto de interfaces que se conectam “diretamente”, sem a necessidade de um roteador. Para determinar quantas subredes existem na rede retire todos os roteadores e host da rede, cada “ilha de rede” isolada que sobrar será uma subrede.



-A **Máscara de Subrede** indica quantos bits mais à esquerda do endereço IP fazem parte do prefixo, consequentemente definindo o endereço de subrede.

-A máscara possui duas formas de ser representada:

1. Indica diretamente quantos bits fazem parte. EX: 223.1.3.0/24 está indicando que os primeiros 24 bits determinam o endereço de subrede.
2. Indicando indiretamente. EX: 223.1.3.0/255.255.255.0 nesse endereço após a barra, todos os bits que compõe o endereço IP estão setados para 1, e os que não compõe para 0.

-Para amenizar esse desperdício, surge a estratégia **Roteamento Interdomínio sem Classes**(**CIDR**). O CIDR permite que qualquer número de bits seja utilizado como máscara de subrede. O endereço IP fica então a.b.c.d/x.

-Antes da adoção do CIDR, os tamanhos das parcelas de um endereço IP estavam limitados a 8, 16 ou 24 bits (redes de classe A, B e C), um esquema de endereçamento conhecido como **Endereçamento com Classes**. Essa divisão se mostrou problemática para suportar o rápido crescimento do número de organizações com sub-redes de pequeno e médio portes pois a classe C só suportava $2^8 - 2 = 254$ hosts, mas a B suportava $2^{16} - 2 =$ 65.634. Isso acabava acarretando em muito desperdício de endereços IP.

-O endereço 255.255.255.255 é utilizado para broadcasting. Um datagrama com esse endereço IP de destino será enviado para todos os outros hospedeiros na subrede.

----------BOTEI POR DESENCARGO DE CONSCIÊNCIA----------
-Temos 2 ISPs, a ISP A comprou a subrede 200.23.16.0/20, e a ISP B comprou a subrede 199.31.0.0/16.
A ISP A pede à internet todos os datagramas com endereço que comecem com 200.23.16.0/20 e a ISP B pede à internet todos os datagramas com endereço que comecem com 199.31.0.0/16.

A ISP A tem interesse de vender esses domínios, e pode fazê-lo vendendo-os para 8 empresas, dessa forma se cria uma “subrede” dentro da própria subrede, onde os 3 bits após a máscara determinam para qual empresa o datagrama será enviado. Mas agora a ISP A comprou a ISP B, e deseja ligar uma das empresas à ISP B. Para que a empresa não tenha que configurar todos os roteadores para se adaptar à nova ISP, pode configurar o roteador da ISP B para pedir todos os datagramas com endereço que comecem com 199.31.0.0/16 E que tenham o endereço igual ao da empresa movimentada.
----------FIM DO DESENCARGO----------

Obtenção de um bloco de endereços

-O ICANN é o responsável de gerenciar o espaço de endereços IP e alocar blocos a ISPs e outras organizações. Gerencia DNS raiz. Resolve disputas de nome de domínio.

Obtenção de um endereço de hospedeiro: o Protocolo de Configuração Dinâmica de Hospedeiros (DHCP)

-Ao invés de ter que atribuir manualmente endereços IPs a cada hospedeiro, o **DHCP** permite que um hospedeiro obtenha um endereço IP de forma automática. Pode ser configurado para que um hospedeiro receba sempre o mesmo endereço IP ao se contar à rede, ou pode receber um endereço IP temporário sempre que se conectar. Devido à sua automação, o DHCP é denominado um protocolo **plug and play**.

-Bastante utilizado em redes que os usuários costumam ficar conectados por tempo limitado(bibliotecas, sala de aula, hospitais). Toda vez que um hospedeiro se conecta à rede, o servidor DHCP designa a ele um endereço arbitrário do seu reservatório de endereços disponíveis; toda vez que um hospedeiro sai, o endereço é devolvido ao reservatório.

-Para um hospedeiro recém-chegado, o protocolo DHCP é um processo de quatro etapas:

1. **Descoberta do Servidor DHCP:** O novo hospedeiro precisa descobrir o servidor DHCP da sua subrede, portanto ele envia uma mensagem de descoberta DHCP, com endereço IP de destino 255.255.255.255 para que o datagrama chegue em todos os aparelhos da subrede, e um endereço IP de origem 0.0.0.0
2. **Oferta(s) dos Servidores DHCP:** Um servidor DHCP que receba uma mensagem de descoberta DHCP, responde com uma mensagem de oferta DHCP. Essa mensagem também é enviada usando o endereço IP de destino de difusão.
3. **Solicitação DHCP:** O cliente escolherá uma dentre as várias ofertas recebidas com uma mensagem de solicitação DHCP, repetindo os parâmetros de configuração.
4. **DHCP ACK:** O servidor responde a mensagem de requisição DHCP com uma mensagem DHCP ACK, confirmando os parâmetros requisitados.

-Uma vez recebida o DHCP ACK, o cliente está liberado a utilizar o endereço IP ofertado.

Tradução de endereços na rede (NAT)

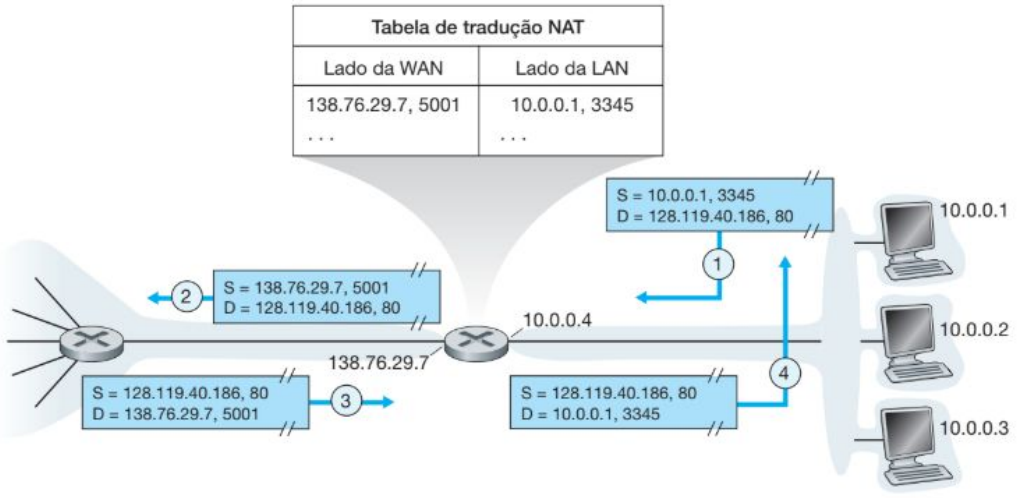
-O roteador que usa NAT não parece um roteador para o mundo externo, pois se comporta como um equipamento único com um único endereço IP.

-O NAT permite que subredes compartilhem o mesmo prefixo de endereço IP, visto que o roteador que distribui os datagramas aos hospedeiros possui seu próprio endereço IP exclusivo dos demais roteadores da rede. Assim expandindo a quantidade de endereços IPs disponíveis na rede.

-Para que o roteador saiba para qual dispositivo da subrede deve encaminhar um pacote vindo da internet, ele se utiliza de uma **Tabela de Tradução Nat**.

-Quando um datagrama chega de um hospedeiro para o roteador, o roteador gera um novo número de porta para aquela conexão, e então adiciona na tabela um novo registro o qual traduz o par <endereço IP NAT, porta roteador> para o par <endereço IP host, porta host>. O roteador então substitui o endereço IP de origem e a porta de origem pela tradução na tabela e manda o datagrama para a internet.

-Quando um datagrama chega da internet para o roteador, o roteador consulta a tabela para saber a qual host o datagrama está direcionado, após descobrir ele substitui os valores do endereço IP de destino e porta de destino e envia o datagrama para dentro da subrede.



-Como o número de porta tem 16 bits, um roteador NAT consegue suportar mais de 60.000 conexões simultâneas.

-O NAT gera controversias entre os puristas da comunidade IETF.

- Roteadores devem processor somente até a 3ª camada

- Viola o argumento fim-a-fim. Hospedeiros devem se comunicar diretamente, sem a interferência de nós que modifiquem os endereços IPs e números de portas.
- Esgotamento de IPs deve ser resolvido com IPv6.

UPnP(desnecessário)

4.4.3 *Protocolo de Mensagens de Controle da Internet (ICMP)*

-O **ICMP** é usado por hospedeiros e roteadores para comunicar informações do nível de rede entre si. Sendo usado principalmente para comunicação de erros.

-Em termos de arquitetura, o ICMP está uma camada acima da camada de rede(mas não faz parte da camada de transporte), visto que mensagens ICMP são encapsuladas em datagramas IP, que nem segmentos TCP/UDP são.

-Mensagens ICMP incluem um campo de tipo, um campo de código e os 8 primeiros bytes do datagrama IP que causou o erro.

-O traceroute é executado com mensagens ICMP. O traceroute de origem envia uma série de datagramas comuns ao destino, cada série constitui de 3 datagramas. A 1ª série tem um TTL de 1, a 2ª série tem um TTL de 2, a 3ª série um TTL de 3 e assim por diante. A origem inicializa um temporizador para cada datagrama enviado.

-No traceroute quando o n-ésimo datagrama chega no n-ésimo roteador, o roteador observa que o TTL do datagrama expirou. Seguindo as regras do protocolo IP, o roteador descarta o datagrama e envia uma mensagem ICMP de aviso à origem. Essa mensagem ICMP inclui o nome do roteador(se houver) e seu endereço IP. Quando a mensagem chega na origem, o temporizador é parado e assim se obtêm o tempo de ida e volta.

-O datagrama do traceroute inclui um segmento UDP com um número de porta improvável de estar sendo utilizado. O destinatário ao eventualmente receber esse datagrama retornará ao remetente uma mensagem ICMP de erro “porta de destino inalcançável”. O remetente ao receber essa mensagem ICMP entenderá que o traceroute foi finalizado.

-Um roteador pode ser programado para não enviar mensagem ICMP, consequentemente também não responderá a traceroutes.

4.4.4 *IPv6*

-Com os endereços de 32 bits começando a se esgotar, houve um interesse crescente em desenvolver o sucessor do IPv4, o IPv6.

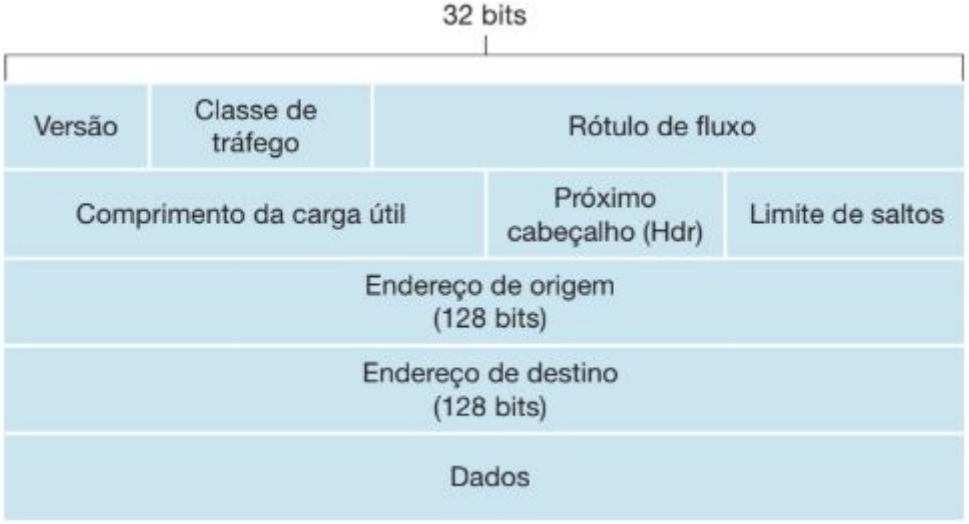
Formato do datagrama IPv6

-As maiores mudanças no introduzidas no IPv6 foram:

- *Capacidade de Endereçamento Expandida*: O endereço IP passou de 32 bits para 128 bits(quantidade suficiente para endereçar cada grão de areia da terra). O IPv6 introduziu um novo tipo de endereço, denominado endereço para qualquer membro do grupo (**anycast**), que permite que um datagrama seja entregue a qualquer hospedeiro de um grupo.
- *Cabeçalho Aprimorado de 40 bytes*: O cabeçalho do datagrama agora tem um tamanho fixo de 40 bytes, resultando num processamento mais veloz do datagrama pelo roteador.
- *Rotulação de Fluxo e Prioridade*: Rotula os pacotes de diferentes tipos de serviços. E oferece prioridade a certos tipos de fluxo em relação a outros.

-Campos no datagrama IPv6:

- **Versão:** Campo de 4 bits utilizado para identificar a versão do protocolo IP.
- **Classe de Tráfego:** Campo de 8 bits que diferencia os diferentes tipos de datagrama.
- **Rótulo de Fluxo:** Campo de 20 bits usado para identificar um fluxo de datagramas.
- **Comprimento da Carga Útil:** (16 bits)Número de bytes no campo de dados.
- **Próximo Cabeçalho:** Identifica o protocolo da camada de transporte que o conteúdo encapsulado deve ser entregue(EX.: TCP/UDP).
- **Limite de Saltos:** Conteúdo decrementado em 1 para cada roteador que repassa o datagrama. Se chegar a 0, o datagrama é descartado.
- **Endereço de Origem/Destino:** Endereço IPv6 do remetente e do destinatário.
- **Dados:** Carga útil do datagrama IPv6.



-Alguns campos do datagrama IPv4 foram removidos:

- **Fragmentação/Remontagem:** Essa operação agora fica exclusiva pros sistemas finais. Caso um roteador receba um datagrama muito grande, ele será descartado e uma mensagem ICMP de erro(“pacote muito grande”) é enviada de volta ao remetente. O remetente pode então reenviar os dados em datagramas menores.
- **Checksum:** Como tanto o TCP quanto o UDP já apresentam essa característica, para poupar tempo de processamento nos roteadores o checksum fica exclusivo pros sistemas finais.
- **Opções:** O campo de opções não faz mais parte do cabeçalho-padrão do IP. Em vez disso, passou a ser um dos possíveis próximos cabeçalhos a ser apontados pelo cabeçalho do IPv6. Isso resulta em um cabeçalho IP de tamanho fixo de 40 bytes, que é mais rápido de se processar.

-O ICMP também atualizou para acompanhar o protocolo IPv6, e agora se chama ICMPv6. Novos códigos foram adicionados, tais como “pacote muito grande” e “opções IPv6 não reconhecidas”.

(EXTRA)Notação do endereço IPv6

-8 grupos, cada grupo de 2 bytes é representado por 4 dígitos hexadecimais.
2001:0db8:85a3:08d3:1319:8a2e:0370:7334

-Um grupo composto de quatro 0 pode ser omitidos e substituídos por ::
2001:0db8:85a3:0000:1319:8a2e:0370:7334 equivale à
2001:0db8:85a3::1319:8a2e:0370:7334

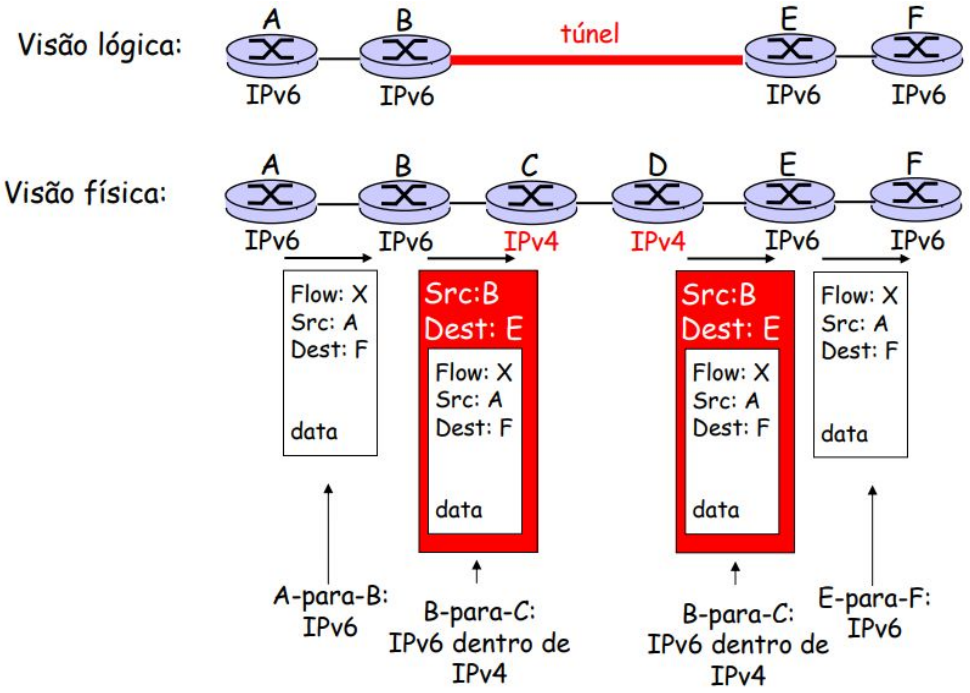
-As simplificações não podem gerar ambiguidades
2001:C00:0:0:5400:0:0:9 → 2001:C00::5400::9 ??

-Em uma url deve-se utilizar colchetes ao escrever o endereço IPv6 visto que normalmente após o colchetes se encontra o número de porta.
http://[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:443/

Transição do IPv4 para o IPv6

-Uma forma é o “dia do mutirão”, onde todas as máquinas da terra seriam desligadas e seriam atualizadas de IPv4 para IPv6. Devido à gigantesca quantidade de máquinas conectadas à rede nos dias atuais, essa ideia se torna impossível.

-A melhor opção é o **Tunelamento**. O tunelamento funciona da seguinte forma; Caso dois roteadores que suportam IPv6 queiram se comunicar, mas entre eles há roteadores intermediários que não suportam, esses roteadores funcionarão como um túnel. O roteador remetente irá encapsular o datagrama IPv6 dentro de um datagrama IPv4 e enviará através dos roteadores intermediários até que chegue no roteador destinatário. O roteador destinatário ao receber o datagrama, determina que ele contém um datagrama IPv6, extrai o datagrama e o roteia normalmente.



5ºcapítulo(camada de enlace: enlaces, redes de acesso e redes locais)

5.1 Introdução à camada de enlace

-Um **Nó** é qualquer dispositivo que rode um protocolo da camada de enlace. Isso inclui hospedeiros, roteadores, comutadores e pontos de acesso Wi-Fi.

-**Enlaces** são os canais de comunicação que conectam nós adjacentes.

-Um pacote dessa camada se chama **Quadro**, e ele encapsula um datagrama.

5.1.1 *Os serviços fornecidos pela camada de enlace*

-O serviço básico da camada de enlace é transferir um datagrama de um nó até outro adjacente, mas diferentes protocolos podem oferecer diferentes tipos de serviços, tais como:

- *Enquadramento de dados*: Encapsular datagramas em quadros.

- *Acesso ao enlace*: Um protocolo de controle de acesso ao meio (MAC) especifica as regras segundo as quais um quadro é transmitido pelo enlace. O caso mais interessante é quando vários nós compartilham um único enlace de difusão. Aqui, o protocolo MAC serve para coordenar as transmissões de quadros dos muitos nós.
- *Entrega confiável*: Garantia que irá transferir sem erro cada datagrama pelo enlace. Mais utilizado por enlaces sem fio onde as taxas de erro costumam ser maiores, para que o erro seja corrigido localmente ao invés de forçar uma retransmissão. Pouco utilizado em enlaces com taxas de erro baixa, visto que é visto como uma sobrecarga desnecessária.
- *Detecção e correção de erros*: Os erros podem ser causados por atenuação de sinal e ruídos eletromagnéticos. O hardware do receptor identifica a presença de erros e descarta o quadro ou envia uma sinalização ao remetente para a retransmissão do quadro. Alguns protocolos fornecem não apenas a detecção do erro como também determina o local exato do erro e o corrige.

5.1.2 *Onde a camada de enlace é implementada?*

-Normalmente a camada de enlace é implementado no adaptador de rede (placa de interface de rede - **NIC**). No núcleo do adaptador de rede está o controlador da camada de enlace, em geral um único chip de sistema especial, que executa vários serviços da camada de enlace. Muito da funcionalidade do controlador da camada de enlace é realizado em hardware.

-No remetente, um datagrama é encapsulado em um quadro, adiciona bits ao cabeçalho do quadro e então o transmite para o enlace de comunicação, seguindo o protocolo de acesso ao enlace.

-No destinatário, ao receber o quadro é realizada uma procura por erros. Se não houver erros, o datagrama anexado é passado para a camada de rede.

5.2 Técnicas de detecção e correção de erros

-No nó remetente os dados, D, são aumentados com bits de detecção e correção, EDC, para que os dados fiquem protegidos contra erros de bits. Tanto D quanto EDC são enviados ao destinatário em um quadro. O nó receptor irá receber D’ e EDC’, que podem ou não serem iguais a D e EDC, respectivamente.

-As técnicas de detecção e correção não são 100% perfeitas, é possível que o receptor não detecte erros na informação recebida e, consequentemente, entregue um datagrama corrompido à camada de rede.

-Técnicas mais sofisticas têm uma chance menor de não detectar erros, mas ficam sujeitas a uma sobrecarga maior.

-Três técnicas para detecção são: **verificação de paridade, métodos de soma de verificação e verificações de redundância cíclica** (CRCs).

5.2.1 *Verificações de paridade*

-Nessa técnica o campo EDC é composto de apenas 1 bit. Em um esquema de paridade par, esse bit é ajustado para que a quantidade de bits 1 de D+EDC seja um número par. Análogo ocorre para o esquema de paridade ímpar, porém o bit é ajustado para que a quantidade seja ímpar.

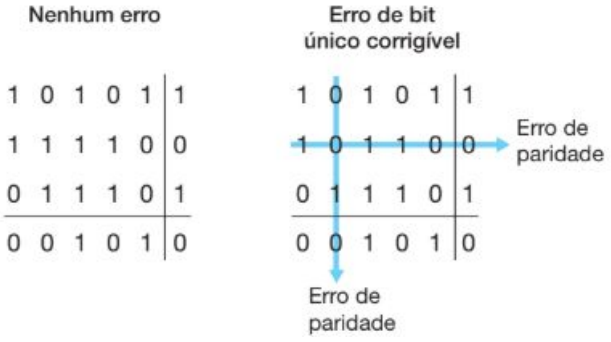
-Se o receptor contar a quantidade de bits 1 e encontrar uma paridade diferente do que o esperado, se sabe que houve uma inversão de uma quantidade ímpar de bits.

-Se uma quantidade par de bits foi invertido isso resultaria em um erro não detectado.

-Essa técnica tem uma eficiência melhor caso a probabilidade de erro em bits seja pequena, e que um bit independa do outro para que haja erro. Na realidade, os erros em geral se aglomeram em “rajadas”, e torna essa técnica não recomendada.

-Para a correção de bits, pode ser utilizado **Paridade Bidimensional**. Os bits de D são separados em i linhas e j colunas, e um valor de paridade é calculado para cada linha e para cada coluna.

-Suponha que ocorreu um único erro de bit em D, com esse esquema, tanto a linha quanto a coluna do bit modificado estarão com a paridade errada. O receptor então pode não apenas detectar que houve erro, como corrigi-lo visto que sabe exatamente o seu local. Também é possível detectar e corrigir um erro de bit ocorrido nos bits de paridade.



-Correção de erros antecipada(**FEC**) é a capacidade do receptor para detectar e corrigir erros. As técnicas FEC são valiosas porque podem reduzir o número exigido de retransmissões do remetente e consequentemente o tempo total visto que não terá o atraso de propagação da viagem de ida e volta.

5.2.2 *Métodos de soma de verificação*

-Também conhecido como o Checksum já estudado na camada de transporte. Funcionamento idêntico ao do protocolo TCP e UDP.

-Os bytes de dados são tratados como inteiros de 16 bits e são somados. O complemento 1 dessa soma é carregada no cabeçalho do quadro.

-O receptor soma todos os bytes do quadro, incluindo os bytes de cabeçalho, e se o valor resultante houver apenas bits 1, nenhum erro é detectado(mas possa ser que erros tenham sido cometidos).

-Essa técnica é utilizada mais na camada de transporte pois esta trabalha em geral com software, o que exige uma verificação simples e rápida. Porém, na camada de enlace a verificação é feita em hardware dedicado nos adaptadores, o que permite rodar operações mais complexas mais rapidamente.

5.2.3 *Verificação de redundância cíclica (CRC)*

-Tanto o remetente quanto o destinatário fazem um acordo de um padrão de r+1 bits, conhecido como **Gerador**, G. O bit mais significativo de G é igual a 1.

-Para a parcela D de dados, o remetente escolherá r bits adicionais, R, e os anexará à D de modo que o padrão de d+r bits resultante(interpretado como um número binário) seja divisível exatamente por G, usando aritmética de módulo 2.

-No destinatário o processo de verificação de erros com CRC é simples, o destinatário divide os d+r bits recebidos por G. Se o resto da divisão for diferente de zero, é detectado um erro, caso contrário, os dados são aceitos como corretos.

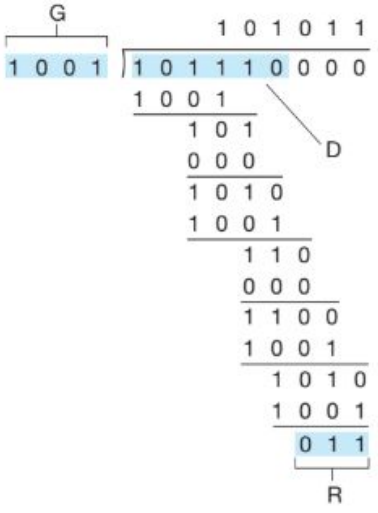
-Todos os cálculos de CRC são feitos por aritmética de módulo 2 sem “vai 1” nas adições nem “empresta 1” nas subtrações. Isso significa que a adição e a subtração são idênticas e ambas são equivalentes à operação ou exclusivo (XOR) bit a bit dos operandos.

-Como na aritmética binária comum, a multiplicação por 2^k desloca um padrão de bits para a esquerda por k casas. Assim, dados D e R, a quantidade D · 2^r XOR R dá como resultado o padrão de bits d + r.

-Desejamos achar um R de forma que D · 2^r XOR R = nG, ou seja, seja múltiplo de G. D · 2^r XOR R = nG equivale à D · 2^r = nG XOR R. Dessa forma vemos que

$$R = resto\left(\frac{D \cdot 2^r}{G}\right)$$

-Segue um exemplo com G = 1001, D = 101110 e r =3.



-Cada padrão CRC pode detectar erros de rajadas de menos de r+1 bits.

5.3 Enlaces e protocolos de acesso múltiplo

-Há dois tipos de enlaces de rede, os enlaces **Ponto-a-Ponto** e de **Difusão(Broadcasat)**.

-O enlace **Ponto-a-Ponto** consiste em apenas um remetente em uma ponta e um único destinatário na outra ponta.(EX.: Ethernet, LAN sem fio)

-O enlace **Broadcast** pode ter vários nós remetentes e vários nós destinatários, todos conectados ao mesmo canal de transmissão único e compartilhado. O termo broadcast/difusão é utilizado porque quando um nó transmite um quadro, todos os outros nós recebem uma cópia do quadro.

-Como cada nó tem a capacidade de transmitir quadros, é possível que dois ou mais transmitam quadros ao mesmo tempo. Quando isso acontece, todos os nós recebem vários quadros ao mesmo tempo, isto é, os quadros transmitidos colidem em todos os receptores.

-De forma geral, quando ocorre uma colisão, o receptor não consegue “entender sentido” nos quadros recebidos, pois os dados são “embaralhados” entre si. Dessa forma, todos os quadros envolvidos são descartados. O canal de difusão é desperdiçado durante o intervalo de colisão.

-Para assegurar que o canal realize trabalho útil mesmo com vários nós, é necessário coordená-los. Essa é a tarefa dos **Protocolos de Acesso Múltiplo**, os quais regulam como os nós irão transmitir pelo canal de difusão compartilhado.

-Um protocolo de acesso múltiplo pode se encaixar em uma de três categorias: **protocolos de divisão de canal, protocolos de acesso aleatório e protocolos de revezamento.**

5.3.1 *Protocolos de divisão de canal*

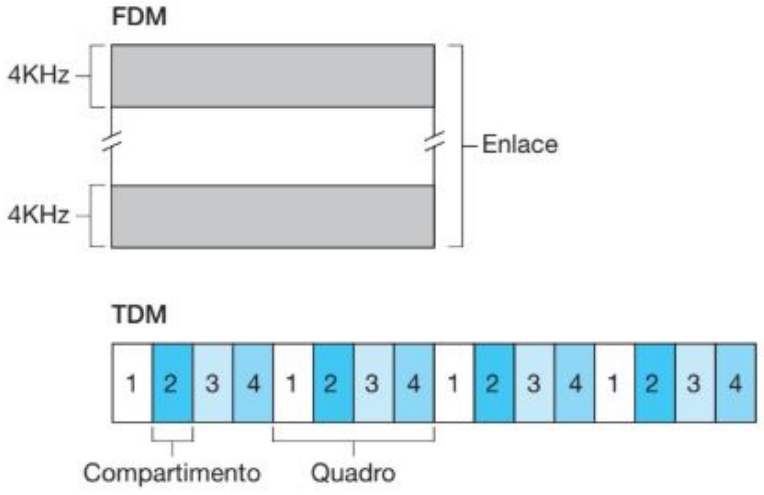
-Suponha a seguinte situação: Um canal de difusão que suporta N nós e velocidade de transmissão de R bits/s. Com esse cenário iremos examinar como o **Protocolo TDM**, o **Protocolo FDM** e o **Protocolo de Acesso Múltiplo por Divisão de Código(CDMA)** agem para dividir a largura de banda do canal.

- O **Protocolo TDM** divide o tempo em **Quadros Temporais**, os quais depois são divididos em N partes, cada uma das partes é associada a um dos nós.

-Sempre que um nó tiver que transmitir um pacote, ele os transmitirá durante à parte atribuída e le no quadro temporal.

-Apesar do protocolo TDM ser justo e eliminar colisões, ele apresenta duas falhas importantes. A primeira é que um nó fica limitado à velocidade de transmissão de R/N bits/s mesmo que seja o único nó com pacotes a enviar. O segundo é que o nó sempre deverá esperar sua vez para transmitir mesmo que seja o único nó pacotes a serem transmitidos.

-O **Protocolo FDM** divide o canal em frequências diferentes e reserva cada uma das frequências para um dos N nós, cada canal com taxa de transmissão de R/N bits/s. Esse protocolo compartilha os benefícios do protocolo TDM, além de poder sempre estar estar transmitindo. Todavia, compartilha a desvantagem de um nó só ter uma taxa de R/N bits/s, mesmo sendo o único a querer transmitir pacotes.



-O **Protocolo CDMA** atribui a cada nó um código. Cada nó utiliza o seu código para codificar os bits que irá transmitir.

-Se os códigos forem escolhidos com cuidado, esse protocolo permite que nós diferentes transmitam informações simultaneamente.

5.3.2 *Protocolos de acesso aleatório*

-Com esse tipo de protocolo o nó transmissor sempre transmite à taxa total do canal, R bits/s.

-Se houver uma colisão, cada nó irá retransmitir seu pacote. Essa retransmissão nem sempre é de imediato, cada nó envolvido na colisão escolhe independentemente um

atraso aleatório. Como os atrasos são escolhidos de forma aleatória e independente, a chance de ocorrer uma nova colisão diminui.

Slotted ALOHA

-Iremos admitir algumas suposições iniciais:

- Todos os quadros tem tamanho L.
- O tempo é dividido em intervalos de tempo L/R segundos.
- Quadros são transmitidos apenas em início de intervalos.
- Os nós estão sincronizados.
- Se dois ou mais nós colidirem, todos os nós estarão cientes da colisão antes do término do intervalo.

-Se não houver colisão, os nós se preparam para enviar o próximo quadro no intervalo seguinte.

-Se uma colisão for detectada o nó terá uma probabilidade p de retransmitir o quadro no próximo intervalo, e uma probabilidade 1-p de pular o próximo intervalo e repetir essa decisão.

-Além de ser simples e descentralizado(apenas necessitando da sincronização dos nós) o slotted ALOHA permite que cada nó transmita continuamente seus quadros na taxa máxima do canal.

-Apesar de sua vantagens, o slotted ALOHA apresenta colisões e desperdício de intervalos, além de exigir sincronia entre os nós e que estes detectem colisões em tempos menores do que o tempo de transmissão de um quadro.

-A eficiência deste protocolo pode ser calculada. Sabendo que $p * (1-p)^{N-1}$ é a chance que 1 nó transmita sozinho em um slot, então $N * p * (1-p)^{N-1}$ é a chance que algum dos N nós consiga. Para a máxima eficiência basta encontrar p que maximize $N * p * (1-p)^{N-1}$. Ao serem realizadas as contas, encontra-se que a eficiência máxima é de 37%, ou seja, o canal só é usado em transmissões úteis 37% do tempo(grande desperdício).

Aloha

-No aloha puro não é necessário que os nós estejam sincronizados. Quando um nó recebe um datagrama, ele o transmitirá imediatamente para o canal.

-Se um quadro transmitido sofrer uma colisão, o nó terá uma chance p de retransmitir de imediato, caso contrário o nó esperará um tempo L/R para que possa repetir esse processo.

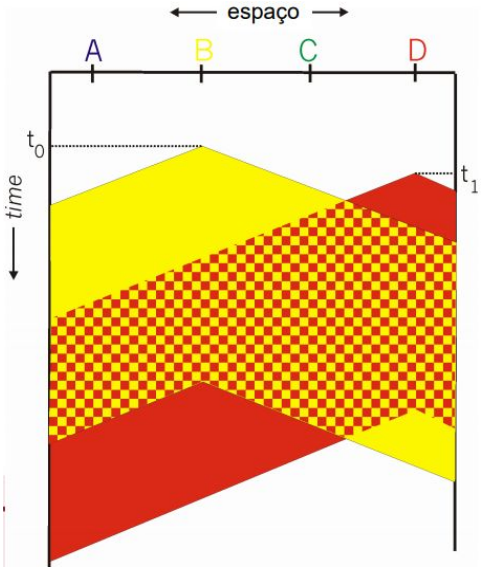
-A eficiência deste protocolo é a metade da do slotted ALOHA já que não basta apenas ter a sorte de ninguém no mesmo intervalo transmitir nada, mas que ninguém no intervalo anterior envie algo também.

CSMA (acesso múltiplo com detecção de portadora)

-Esse protocolo pode ser comparado a uma conversa humana que respeita duas regras:

1. *Ouça Antes de Falar*: O nó ouve o canal de transmissão verificando se há outro nó transmitindo, se houver, ele esperará que este encerre a transmissão para que possa iniciar sua transmissão.
2. *Se alguém começar a falar ao mesmo tempo que você, pare de falar*: Enquanto transmite, o nó também escuta o canal. Se for detectado outro nó transmitindo no canal, ele para de transmitir e espera por algum tempo para repetir o ciclo.

-Mesmo o nó não podendo transmitir caso outro nó esteja transmitindo, é possível que ocorram colisões caso o nó decida transmitir antes que os dados de transmissão de outro nó cheguem nele.



-Note que mesmo detectando a colisão, os nós continuam transmitindo o quadro. Portanto este protocolo não apresenta detecção de colisão.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

-O funcionamento deste protocolo é basicamente o mesmo do CSMA, com a diferença que caso seja detectada uma colisão, o nó irá encerrar instantaneamente a transmissão atual.

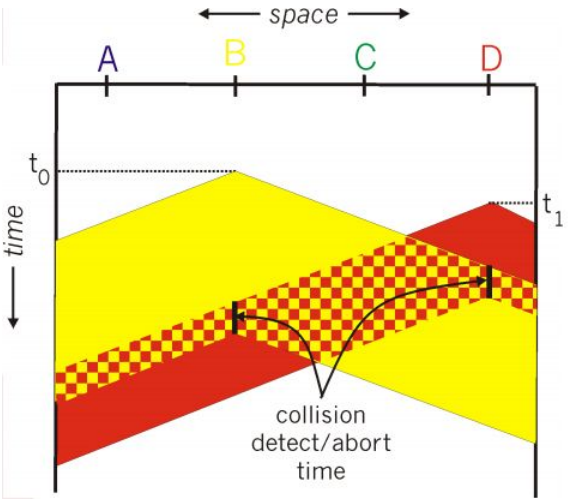
-Para garantir que o tempo de espera seja grande suficiente para não ocorrer uma nova colisão, e pequeno o suficiente para não deixar o canal ocioso por muito tempo, é utilizado o algoritmo de **recuo exponencial binário.** Neste algoritmo, quando um nó for transmitir um quadro que já sofreu N colisões, um valor k será aleatoriamente escolhido dentro do intervalo {0, 1, 2, …, $2^N - 1$}.

-Toda vez que um novo quadro for transmitido o algoritmo retorna do zero, sem considerar colisões passadas recentes.

Eficiência do CSMA/CD

-A eficiência deste protocolo pode ser calculada como $\frac{1}{1+5*\frac{Tprop}{Ttrans}}$, onde Tprop é o tempo de propagação e Ttrans é o tempo de transmissão.

-Pela fórmula vemos que para obter o máximo de eficiência, pode-se tanto aumentar Ttrans quanto abaixar Tprop. Como essa segunda opção é inviável para uma rede real, os trabalhos se concentram em abaixar Tprop.



5.3.3 *Protocolos de revezamento*

-Protocolos de acesso múltiplo ideias devem tanto permitir que quando haja apenas 1 nó ativo este tenha uma vazão de R bits/s, quanto quando haja N nós ativos, cada nó apresente uma vazão de R/N bits/s.

-Os protocolos ALOHA e CSMA apresentam a primeira propriedade, mas não a segunda. Os **Protocolos de Revezamento** foram criados com o intuito de apresentar ambas propriedades.

-Um desses protocolos é o protocolo **Polling**. Neste protocolo um dos nós é designado como um nó “mestre”. Este nó, por alternância circular, envia uma mensagem a cada um dos nós informando que ele está liberado para enviar até um certo número máximo de quadros.

-Contudo, mesmo com esse protocolo evitando colisões e intervalos vazios, ele também introduz um atraso de seleção, ou seja, se apenas 1 nó deseja enviar quadros, enviará a uma taxa menor que R bits/s, pois o nó mestre tem de escolher cada um dos nós ociosos por vez. Além disso, se o nó mestre falhar/precisar de manutenção, o canal inteiro ficará inoperante.

-Outro protocolo é o **Protocolo de Passagem de Permissão(token)**. Este protocolo não apresenta nó mestre, ao invés disso possui um quadro especial denominado **Token,** que é passado entre os nós em uma ordem fixa.

-O nó só é permitido enviar quadros caso possua o token. Quando o nó recebe o token, ele só enviará ao próximo nó da sequência quando não possuir mais quadros para transmitir ou tiver enviado um número máximo de quadros.

-Mesmo com o benefício da descentralização, o protocolo é bem frágil, visto que a falta de um nó derruba o canal inteiro. Ou, se um nó acidentalmente se descuida e não libera a permissão, então é preciso chamar algum procedimento de recuperação para recolocar a permissão em circulação.

5.3.4 *Protocolos de revezamento(Não precisa estudar \o/)*

5.4 Redes locais comutadas

5.4.1 *Endereçamento na camada de enlace e ARP*

Endereços MAC

-Cada interface de rede de um hospedeiro ou roteador possuirá seu próprio endereço da camada de enlace, também chamado de **Endereço MAC**(Media Access Control).

-Comutadores da camada de enlace não possuem endereços MACs associados às interfaces, que se conectam aos hospedeiros e roteadores. Isso porque a função do comutador da camada de enlace é transportar datagramas entre hospedeiros e roteadores; um comutador faz isso de modo transparente, ou seja, sem que o hospedeiro ou roteador tenha que endereçar o quadro explicitamente para o comutador intermediário.

-O endereço MAC é composto por 6 bytes e é representado de forma hexadecimal, com cada byte sendo representado por um par de números hexadecimais.

-Para garantir que não haja dois endereços MAC iguais no mundo, as empresas comprar uma faixa de endereços MAC.

-O endereço MAC é fixo e linear, sempre será o mesmo independente da rede em que se conecte, ao contrário do endereço IP que é variável e hierárquico, ele mudará

dependendo da subrede em que se encontre, um parte representando a subrede na qual se encontra, e outra parte representando o usuário.

-Quando um adaptador que enviar um quadro para outro adaptador, o remetente insere no cabeçalho o endereço MAC do destinatário e o transmite para a rede, mas às vezes esse quadro deverá ser transmitido por difusão. Quando um adaptador recebe um quadro, ele irá verificar o endereço MAC de destino, se for diferente do seu, o quadro é descartado, mas se for o mesmo, o adaptador extrai o datagrama do quadro e o transfere para o protocolo superior.

-Caso se deseje transmitir um quadro para todas as outras interfaces, deve-se utilizar o endereço de difusão MAC, este é FF-FF-FF-FF-FF-FF.

ARP (protocolo de resolução de endereços)

-O protocolo **ARP** realiza a tradução de endereço MAC para endereço IP, e vice-versa.

-No remetente, o protocolo ARP retorna o endereço MAC correspondente ao endereço IP fornecido pelo remetente, desde que esse endereço esteja na mesma subrede. Se o endereço fornecido estiver fora da subrede, o ARP retornará um erro.

-Cada nó(hospedeiro ou roteador) na subrede tem em sua RAM uma **tabela ARP**. Esta realiza o mapeamento de endereço IP para endereço MAC, assim como seus valores de tempo de vida(TTL). Esse tempo indica em quanto tempo o dado mapeamento será apagado.

Endereço IP	Endereço MAC	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

-Nem todo hospedeiro da subrede estará na tabela ARP, visto que ele pode nunca ter sido registrado, ou seu mapeamento pode ter expirado.

-Quando um hospedeiro for enviar um datagrama para outro hospedeiro na subrede e este não estiver endereçado na tabela ARP do remetente, este irá usar o protocolo ARP.

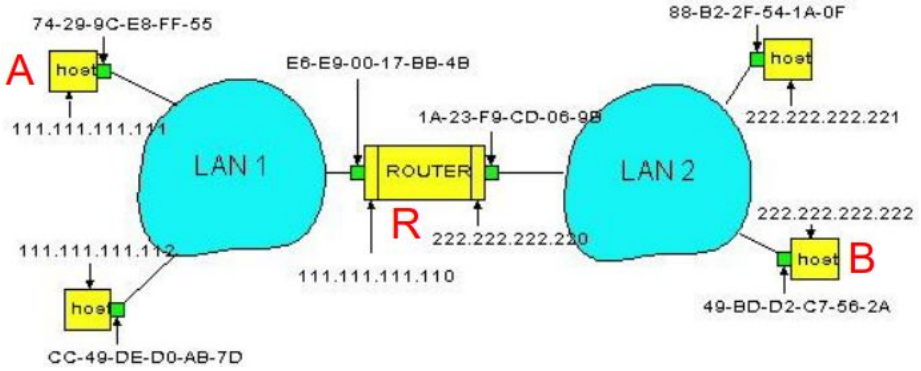
1. O remetente enviará um **Pacote de Consulta ARP** para o adaptador indicando que o endereço MAC de destino é o de difusão(FF-FF-FF-FF-FF-FF).
2. O adaptador encapsula o pacote de consulta ARP em um quadro com o endereço de destino sendo o endereço de difusão e o transmitirá à subrede.
3. O pacote é recebido por todos os outros adaptadores na subrede. O adaptador irá verificar se o endereço IP de destino no pacote é o mesmo que o seu próprio endereço IP. Aquele no qual a igualdade ocorrer devolve ao hospedeiro que fez a consulta um pacote ARP de resposta com o endereçamento desejado.

4. O remetente inicial ao receber o pacote ARP de resposta poderá atualizar sua tabela ARP e transmitir seu datagrama.

-**Observações**: ARP é do tipo plug-and-play. O ARP é talvez mais bem considerado um protocolo que fica em cima do limite entre as camadas de enlace e de rede, não se adequando perfeitamente na simples pilha de protocolos.

Envio de um datagrama para fora da sub-rede

-Suponha a seguinte situação, o hospedeiro A quer enviar um datagrama ao hospedeiro B.



-Como o ARP não consegue agir fora da sua própria subrede, quando A for enviar um quadro, ele irá criá-lo com o endereço IP de destino sendo o de B, mas o endereço MAC de destino será o do roteador R, visto que o pacote necessariamente passará pelo roteador antes de chegar em B.

-O roteador desencapsula o datagrama IP do quadro e utiliza sua tabela de repasse para determinar para qual interface o datagrama será retransmitido.

-Quando o roteador determinar a interface, o adaptador da interface irá encapsular o datagrama em um novo quadro, mas dessa vez o adaptador utilizará sua tabela ARP(que contém o endereçamento de B) para definir o endereço MAC de destino correto, ou seja, o endereço MAC de B.

5.4.2 *Ethernet*

-Tecnologia dominante para LANs cabeadas. O sucesso dela se deve principalmente por ser mais barata do que as outras tecnologias e proporcionar uma banda passante igual ou maior, além de ser mais simples.

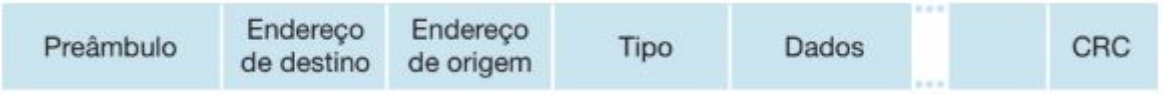
-No início, a Ethernet utilizava uma topologia de barramento, ou seja, todos os quadros transmitidos movem-se para, e são processados por, todos os adaptadores conectados ao barramento.

-Atualmente a Ethernet usa topologia estrela. Essa se utiliza de **Hubs** e **Switches**.

-**Hubs** são dispositivos da camada física que atuam sobre bits individuais, e não sobre quadros. Quando um bit, representado por 0 ou 1, chega de uma interface, o hub recria o bits, aumenta a energia e transmite para todas as outras interfaces.

Estrutura do quadro Ethernet

-O adaptador emissor encapsula o datagrama IP em um quadro Ethernet.



- ***Preâmbulo(8 bytes)***: 7 bytes com padrão 10101010 seguido de 1 byte de padrão 10101011. É utilizado para sincronizar relógios dos adaptadores.
- ***Endereço de Destino(6 bytes)***: Contém o endereço MAC de destino.
- ***Endereço de Origem(6 bytes)***: Contém o endereço MAC de origem.
- ***Campo de Tipo(2 bytes)***: Indica o protocolo a ser usado na camada superior.
- ***Campo de Dados(46 a 1500 bytes)***: Esse campo carrega o datagrama IP.
- ***CRC(4 bytes)***: Verificado no receptor, se erro é detectado, o quadro é descartado.

-As tecnologias Ethernet fornecem um serviço não orientado para conexão. Isto é, o adaptador remetente simplesmente envia o quadro ao adaptador destinatário, sem antes “se apresentar”.

-As tecnologias Ethernet fornecem um serviço não confiável. Ou seja, o destinatário não envia mensagens ACKs ou NACKs ao remetente. Isso pode causar lacunas no fluxo de datagramas passados para a camada de rede. Se estiver usando TCP, essas colunas serão preenchidas, se utilizar UDP, haverá lacunas vazias na aplicação.

Tecnologias Ethernet

-Achei inútil, mas para não falar que não falei das flores, resumão do slide, Ethernet usa protocolo CSMA/CD.

-Veja slide.

5.4.3 *Comutadores da camada de enlace(switch)*

-Um switch é transparente aos hospedeiros e roteadores na subrede. Isso significa que um quadro é endereçado de um nó a outro, que envia o quadro à LAN sem estar ciente da existência dos comutadores.

-Switch tem buffer para armazenar dados caso receba mais informação do que consiga processar.

Repasse e filtragem

-**Filtragem** é a capacidade do switch em determinar se o quadro recebido deve ser repassado ou descartado.

-**Repasse** é a capacidade do switch em determinar para qual interface de saída o quadro deve ser repassado.

-Tanto o repasse quanto a filtragem são feitos a partir de **Tabelas de Comutação**. A tabela contém registros de alguns hospedeiros e roteadores da LAN.

-Um registro na tabela de comutação contém:

- Um endereço MAC.
- A interface do switch que leva em direção a esse endereço.
- Horário o qual o registro foi adicionado à tabela.

Endereço	Interface	Horário
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
....	....	....

-Suponha que um quadro com endereço de destino DD-DD-DD-DD-DD-DD chegue ao switch pela interface x. O switch indexa sua tabela com o endereço recebido e irá realizar o repasse e filtragem, escolhendo uma de três possíveis situações:

1. Não existe nada na tabela para o endereço fornecido. Nesse caso, o switch repassa cópias do quadro para todas as interfaces dele, exceto a interface x.
2. A entrada na tabela associa o endereço dado à interface x. Nesse caso, tanto o remetente quanto o destinatário estão na mesma LAN, e o switch realiza a função de filtragem e descarta o quadro.
3. A entrada na tabela associa o endereço dado à interface y. Nesse caso, o switch realiza sua função de repasse e repassa o quadro para a interface y.

Autoaprendizagem

-Switches têm a capacidade de automaticamente montar suas próprias tabelas de comutação. Essa montagem é feita da seguinte forma:

- A tabela de comutação está inicialmente vazia.
- Para cada quadro recebido de uma interface, o switch armazena em sua tabela o endereço MAC, a interface pela qual o quadro chegou e o tempo atual. Eventualmente, todos os hospedeiros e roteadores irão estar na tabela.
- O switch apagará o registro caso após certo tempo tenha recebido nenhum quadro com aquele endereço como endereço de destino.

-Switches são dispositivos do tipo plug-and-play.

-Switches também são full-duplex, ou seja, qualquer interface do switch pode enviar e receber ao mesmo tempo.

Propriedades de comutação da camada de enlace

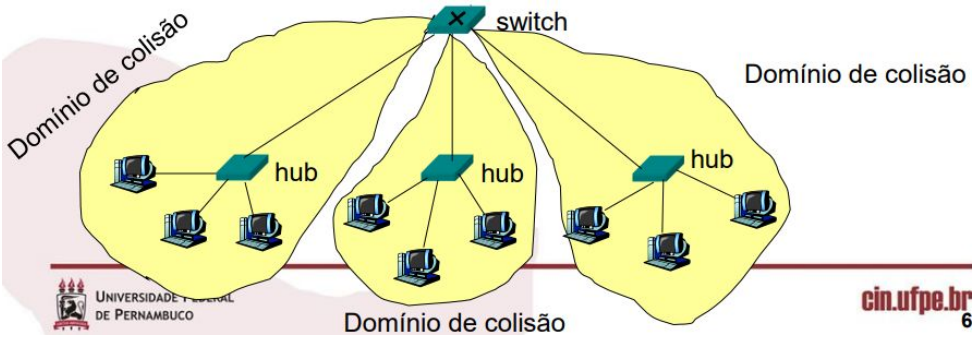
-Há diversas vantagens em se utilizar switch no lugar de enlaces de difusão por barramento ou topologias de estrela baseada em hubs, tais como:

- *Eliminação de Colisões*: Os switches armazenam os quadros e nunca transmitem mais de um quadro em um segmento ao mesmo tempo. Como em um roteador, a vazão máxima agregada de um comutador é a soma da velocidade de todas as interfaces do comutador. Portanto, os switches oferecem uma melhoria de desempenho significativa em relação às LANs com enlaces de difusão.
- *Enlaces heterogêneos*: Consegue operar com vários tipos de enlaces diferentes simultaneamente.
- *Gerenciamento*: O switch é capaz de detectar problemas em interfaces e desconectá-las internamente para que não atrapalhe no seu funcionamento. Se um cabo for cortado, apenas o hospedeiro que o estava usando será desconectado, permitindo com que a rede continue funcionando.

Coisa que está no slide mas não no livro

-Switches criam **Ilhas de Colisão**. São segmentos de LAN da subrede que estão em risco de ocorrer colisão

-Como o switch filtra pacotes, dados de um segmento não são enviados para outros segmentos.



-O switch tem a função **Cut-Through**, que basicamente permite que o quadro seja encaminhado da porta de entrada para a porta de saída antes da recepção completa do quadro(mas só pode após ter recebido o cabeçalho, com o endereço de destino).

Coisa dita em sala

-Como o switch direciona o quadro diretamente para o hospedeiro de destino, caso o endereço de destino esteja na tabela de comutação, alguém que esteja utilizando *wireshark* não conseguirá analisar o quadro, visto que não o receberá.

-Entretanto, ele ainda poderá analisar quadros que sejam enviados por difusão, por exemplo, os quadros que ainda não tem endereço de destino endereçados na tabela.

Comutadores(switches) versus roteadores

-Switches são dispositivos da camada de enlace, enquanto roteadores são dispositivos da camada de rede.

-Como switches processam até a 2ª camada, eles são um pouco mais rápidos do que roteadores, que processam até a 3ª camada.

-Roteadores tem um funcionamento melhor para redes maiores, enquanto comutadores são mais do que suficientes para redes pequenas.

-Switches são plug-and-play, ao contrário de roteadores.

5.4.4 *Redes locais virtuais (VLANs)****(ACABOU CARALHOOOOOOOOOOOOO)***