Introdução a Cadeira

Iremos Estudar

1. Redes de Computadores e Internet;
2. Camada de Aplicação;
3. Camada de Transporte;
4. Camada de Rede;
5. Camada de Enlace: Enlace, Redes de Acesso e Redes Locais;
6. Redes sem fio (wireless) e móveis.

Aparelhos com Internet

São os sistemas finais que rodam as aplicações de rede

- Telefone;
- Computadores;
- Geladeira;
- Torradeira;
- Porta Retrato;
- Tablets;
- Televisores e Bluray;
- Consoles;
- Televisão a Cabo wireless;
- Servidor (fornecedor de conteúdo)

Mas como conectá-los?

- Conexão ADSL (Velox);
- Cabo Coaxial(Net);
- 3G/4G;
- Rádio;
- Wi-fi.

Dispositivos Intermediários da Comunicação

Encaminhadores de pacotes de bits, contendo a informação. É o responsável pelo transporte digital da informação.

- Roteadores
- Switches

Links de Comunicação

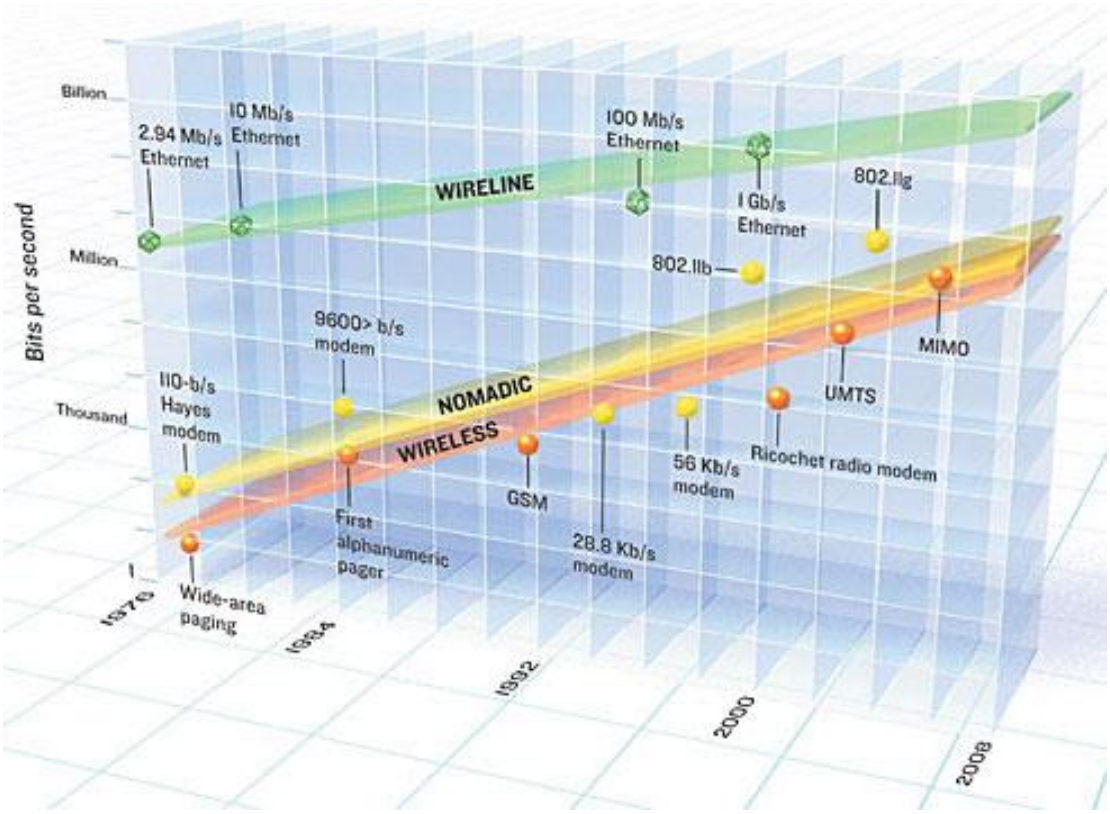
São os responsáveis pelo transporte físico. Possui a taxa de transmissão como propriedade (bandwidth)

- Fibra ótica
- Fibra de Cobre
- Rádio
- Satélite

Lei da Largura de Banda de Edholm

Edhom afirma que a taxa de crescimento da largura da banda das três categorias de banda crescem em termos exponenciais, iguais a Lei de Moore.

"According to Edholm's Law, the three telecommunications categories march almost in lock step: their data rates increase on similar exponential curves, the slower rates trailing the faster ones by a predictable time lag."



Protocolos

Conjunto de regras de comportamento e ações para se obter controle do envio e do recebimento destas informações, definindo o formado, a ordem e ações a tomar na transmissão ou captação.

- TCP;
- IP – especifica o formato entre roteadores e sistemas finais;
- HTTP;
- Skype;
- 802.11

RFC

Documento que descreve padrões de cada protocolo da Internet previamente a serem consideradas um padrão. É estudada, sendo aceita ou rejeitada pelo IESG

- RFC 793 – Protocolo de controle de transmissão
- RFC 2616 – Protocolo de transmissão de Hipertexto
- RFC 2821 – Protocolo de Transmissão de e-mail simples

Internet

"Rede de Redes"

- Livremente hierárquica – mais livre que uma rede telefônica
- ISP's interconectados – provedores de serviço de internet interconectados

Tarefa: Obter as RFC emitidas no 1 de abril de 14 → http://www.rfc-editor.org/info/rfc7168

Periferia da Rede

Quem está na periferia

- Hospedeiro dos programadas que utilizam a Internet(hosts)
    - Computadores de todos os tipos, seja cliente ou servidor.
- Aplicações

Obs: Aplicações distribuídas → Envolvem múltiplos sistemas finais que trocam dados entre si.

Como ocorre a conexão aos roteadores

Através das redes de acesso (redes físicas)

- Redes de acesso residencial
- Redes de acesso coorporativo
- Redes de acesso sem fio

Questões a serem consideradas

- Largura da banda da rede de acesso
- Compartilhada ou dedicada

APIS – Interface de Aplicação

Especifica como um programa que roda em um sistema final pede a infraestrutura da Internet para entregar os dados para outro destino final. Um conjunto de regras que define como os dados devem ser enviados para a Internet de modo que ela possa enviar para o destino.

Acesso Residencial

Os dois tipos prevalecentes de banda larga neste tipo de acesso são:

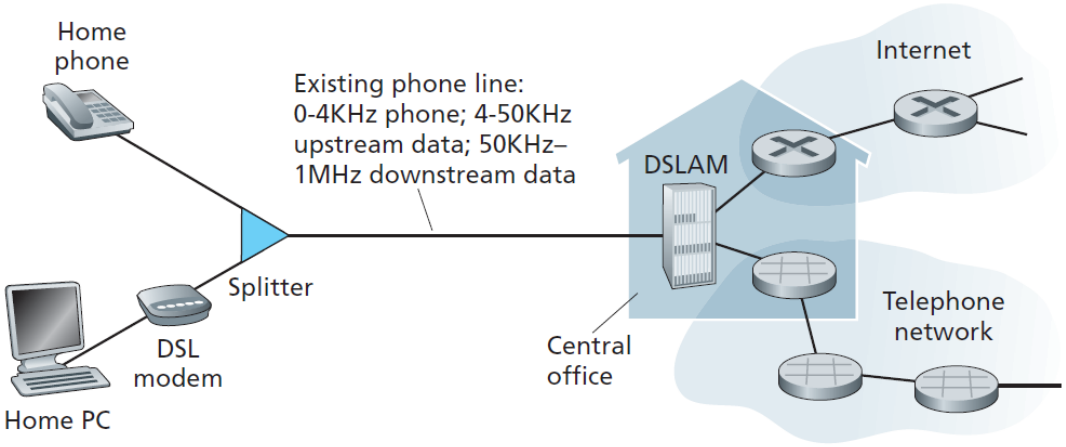
- DSL – Digital Subscriber line

Normalmente a DSL é oferecida pela mesma distribuidora de telefonia, desta forma, a própria companhia telefônica é sua ISP. A informação é passada, através da linha telefônica, para o DSLAM (Digital subscriber line acess multiplexer) na central telefônica, transformando os dados de forma digital para forma analógica (bits para frequência de tons), sendo o DSLAM responsável por transformar este sinal de volta para forma digital.

Para não ocorrer mistura dos sinais telefônicos com o de Internet, ambos são codificados em frequências diferentes e um *Spliter* é responsável pela separação destes sinais quando chegados no domicílio ou passa-los para a DSL, o mesmo ocorre com a DSLAM.

Esta tecnologia serviu para separar o acesso à Internet do acesso a rede telefônica, de modo que agora poderia haver conexão telefônica e à Internet simultaneamente, cada um com suas frequências diferentes, e cada residência tem sua própria rede (rede dedicada).

No Upstream e Downstream de dados, as faixas de frequências são distintas, de modo que a faixa de Downstream é muito maior a que a de Upstream, ambas limitadas por uma faixa suportável pelo cabo.



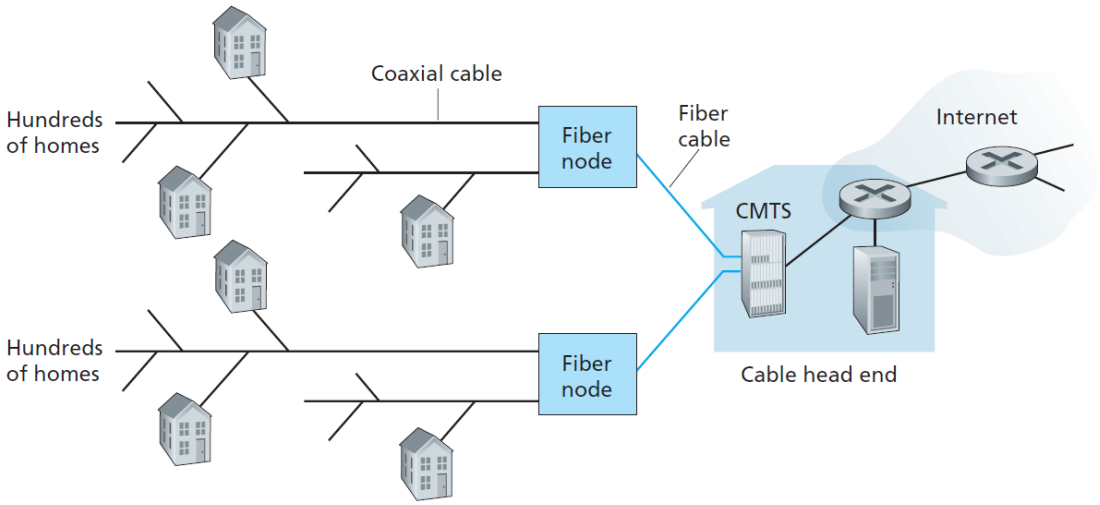
- A Cabo

Enquanto que a redes a DSL dependem da Companhia Telefônica, a acesso via cabo utiliza a infraestrutura das Companhias Televisivas e seus cabos, deste modo, o domicílio obtém acesso à Internet da mesma companhia que sua televisão.

Para se conectar as casas do cliente, as empresas de Televisão utilizam o cabo coaxial (um único cabo com vários ramos – que suporta até 5000 casas). Quando este cabo se interliga com um cabo de fibra ótica, todo o sistema é chamada de Fibra coaxial híbrida (HFC)

Redes de acesso via cabo requerem modems especiais, os modems à cabo. E da mesma forma que para a conexão via DSL existe o DSLAM no CO, ao final da conexão do modem existe o Sistema do Terminal do Modem à Cabo (CMTS), que também transforma o sinal analógica vindo dos cabos para sinal digital, que também tem a característica de ser assimétrica (com frequências baixas sendo transmitidas por um canal e frequências altas por outro canal) – multiplexação por divisão de frequências.

Uma característica muito importante das redes a cabo é que ela divide o seu meio de transmissão, de forma que todas os dados enviados da CMTS para a casas chegam a todas as casas pelo mesmo cabo no canal mais baixo e todos os dados enviados por todas as casa para a CMTS compartilham o mesmo canal mais alto. Por essa razão, se vários usuários fizerem uma requisição da mesma página, vídeo ou imagem para a CMTS, a taxa de envio dos arquivos será muito menor do que a esperada. De forma análoga, se os usuários estiverem apenas surfando na Internet, suas velocidades serão a esLSNperada (rápida), pois será raro que vários usuários façam a requisição do mesmo arquivo ao mesmo tempo. Além disso, porque o canal mais alto é compartilhado, há necessidade de um protocolo para impedir colisões e coordenar as transmissões, o protocolo de acesso múltiplo distribuído.



Apesar de serem prevalecentes, há outros tipos de conexões residenciais, como:

- FTTH – Fiber to the Home

  Como sugere o nome, a FTTH propõe uma ligação direta entre CO e o domicílio através de uma fibra ótica. E há várias tecnológicas que buscam uma distribuição ótima das fibras. Uma delas é a *direct fiber* que é uma ligação direta entre CO e domicílio.

  Outro tipo de distribuição é a qual uma única longa fibra recebe de curtas fibras de cada casa e vai até a CO. Este modelo tem dois tipos de arquitetura, a AON (*active optical networks*) e a PON (*passive optical networks*). A AON é basicamente um Ethernet com switches – será explicado mais tarde.

  Na PON, cada domicílio possui um terminal de rede ótica – ONT – que é conectado ao um *splitter* da vizinhança através de cabos óticos, que é responsável por receber os dados de todas as casas e uni-los em uma única linha ótica, levando até o termina ótico de linha – OLT – na central telefônica. O OLT fica então responsável por converter sinais óticas para digitais e se conecta a Internet pelo roteador da CO. No domicílio, o acesso à Internet é feita via o roteador da casa, que se conecta ao ONT. Ressaltando que todos os dados que vão do OLT ao *splitter* são replicados no *splitter*.

- Via Satélite e Dial-Up

  Em locais em que as outras opções de conexão de Internet não podem ser realizadas, satélites podem ser utilizados como fonte de acesso.

  Já o Dial-up usa a mesma ideia do DSL, utilizando o cabo telefônico para transmitir os dados, com um desvantagem: a conexão é altamente lenta se comparada com os outros tipos de acesso.

Acesso Coorporativo (e atualmente domiciliar)

Em grandes locais em que o acesso à Internet é feito por vários usuários, como campos universitários e corporações – mas também em casas domésticas hoje em dia, há uma necessidade de conectar estes computadores a um roteador final via um rede local, chamada de LAN, tendo como a mais utilizada tecnologia de acesso a Ethernet.

A ideia da Ethernet é bem simples, através de cabos de cobre, sistemas finais se conectam a um *switcher* Ethernet (uma rede *switchers* interconectados) que fica responsável por se conectar à Internet para os sistemas.

Com a utilização da tecnologia sem fio, os sistemas finais se conectam a um ponto de acesso sem utilizar fios – transferindo e recebendo dados deste - e este se conecta a uma rede a cabo baseado na Ethernet.

Nos domicílios, a rede de acesso à Internet ficou mais poderosa, de modo que muitos combinam o acesso via modem a cabo ou DSL com a tecnologia WiFi.

Meios de Propagação

Ainda falando da periferia, temos que discutir um pouco sobre os médios em que as conexões se propagam.

- Meio Físico

  Possui duas categorias:

  - Mídia guiada

    Nas mídias guiadas, as ondas são guiadas num médio sólido, como fibras óticas ou de cobre e cabos coaxiais.
  - Mídia não-guiada

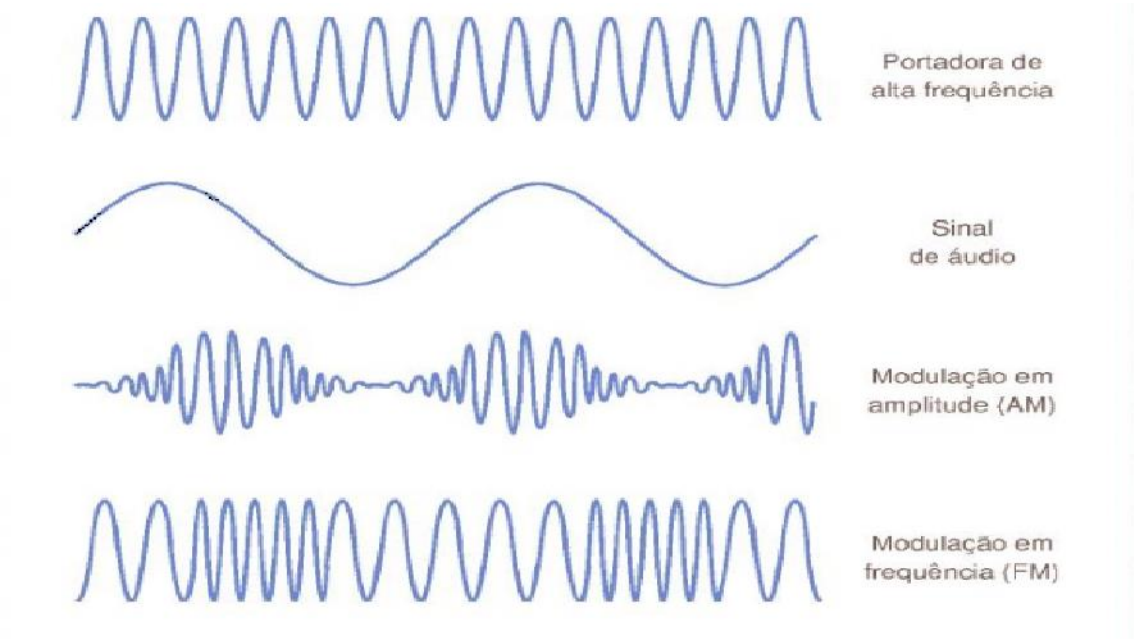
    Nas não-guiadas, temos que as ondas navegam pelo espaço, como ocorre nas redes WiFi e nos satélites digitais.

Como modem funciona (não cobrado)

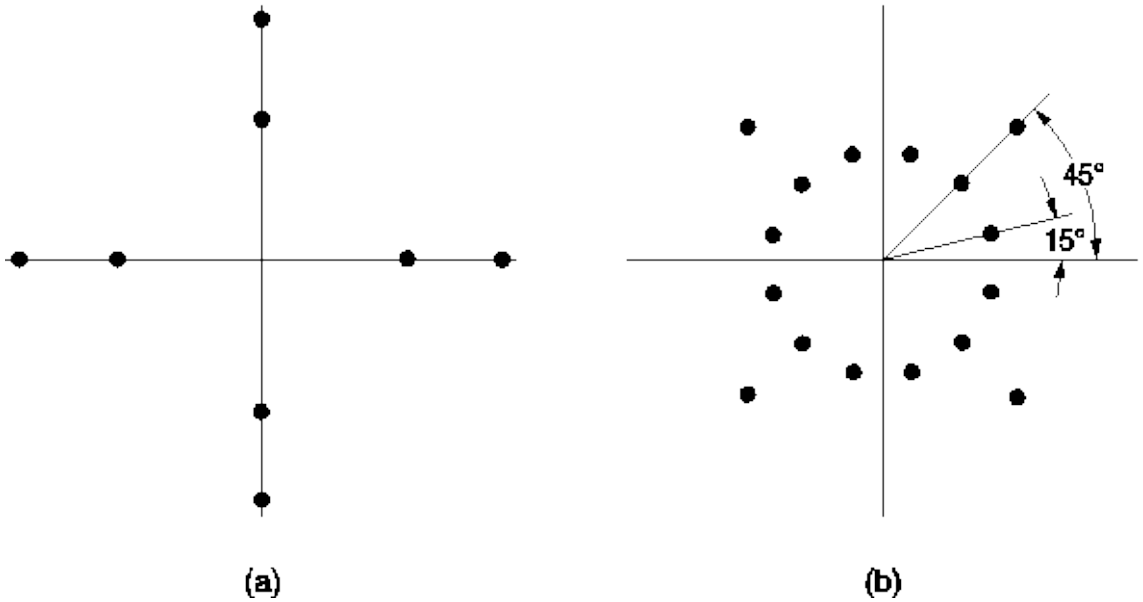
Modulação: Alteração de alguma característica da onda para viabilizar sua compreensão. Existem três tipos de modulação:

- Modulação da amplitude – Alteração da amplitude
- Modulação da fase – Alteração da fase
- Modulação da frequência – Alteração da frequência

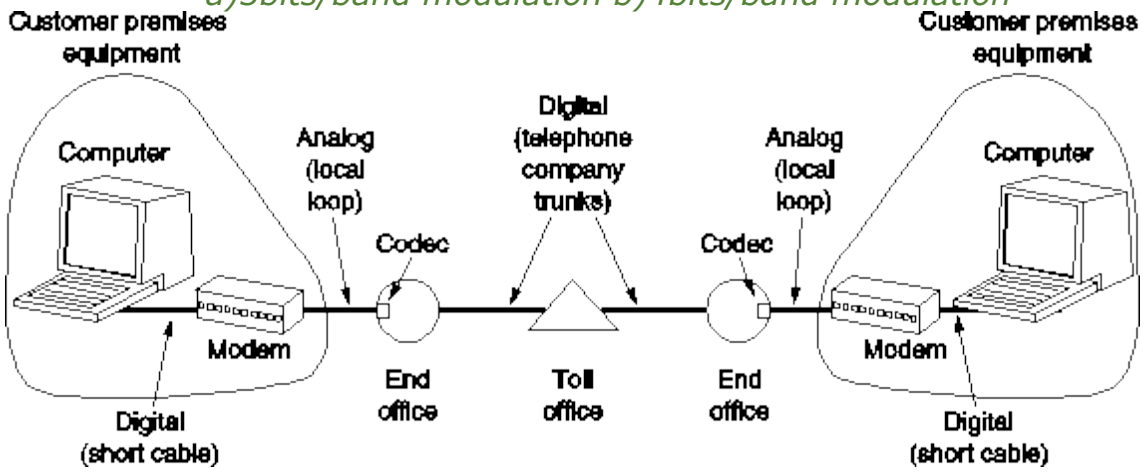
Demodular é o processo de recuperar a informação original a partir da onda modularizada.



Modems utilizam a combinação da modulação de fase com a modulação de amplitude, cada modem com sua especificidade.



a)3bits/band modulation b)4bits/band modulation



Núcleo da Rede

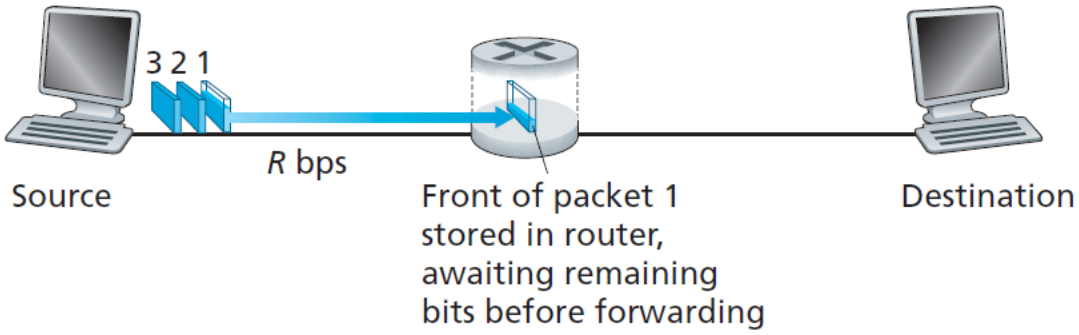
Comutação de Pacotes

Na camada de aplicação, há uma troca de informações entre cada sistema final, contendo qualquer tipo de dado. Para fazer tal procedimento, o responsável por enviar os dados os quebra em pequenos pacotes. No caminho dos pacotes, eles acabam passando por ligações de comunicação (enlaces) e por comutadores (*switches*) de pacotes, para o qual há dois tipos predominantes: **roteadores** e **comutadores de camada de enlace**.

Como os pacotes são transportados pelo canal de comunicação com "velocidade" máxima do canal, então podemos dizer que:

- Seja Rbits/sec a taxa de transferência do canal
- E L o tamanho do pacote percorrendo o canal
- Então o tempo de transmissão T para transferir o pacote é: $T = L/R$.

A maioria dos comutadores de pacote usa a técnica de transmissão *store-and-forward* para enviar os dados. Esta técnica pede para o comutador de pacotes esperar que todos os bits do pacote cheguem a ele para então enviar o primeiro bit do pacote.



Calculemos agora a quantidade de *delay* de transferência que o envio de pacote tem para chegar ao seus destino, partindo de uma fonte qualquer e desconsiderando o *delay* de propagação. Como o tempo de enviar o pacote de uma fonte a outra é L/R segundos, então podemos dizer que o tempo de envio do computador fonte para o *switch* é L/R segundos. De modo análogo, podemos dizer que o envio do *switch* para o computador destino seria L/R segundos, então nosso tempo total é 2L/R segundos para enviar no esquemático acima. Se o *switch* pudesse enviar todos os pacotes assim que o recebesse, então o tempo seria reduzido pela metade.

De forma geral, podemos dizer que:

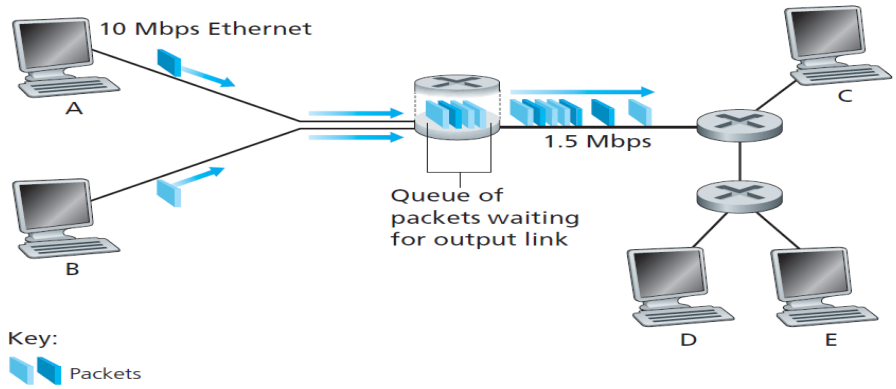
- Dado uma conexão entre dois sistemas finais com N ligações (N-1 roteadores)
- Todas as ligações tem taxa de transferência R
- E seja L o tamanho do pacote
- Então o *delay* de transferência *end-to-end* de um pacote pode ser dito como: $d_{end\text{-}to\text{-}end} = N * (L/R)$.

Para uma transferência de P pacotes em N ligações, podemos deduzir a seguinte fórmula:

- Dado que para passar os P-1 pacotes do primeiro *switch* gastaremos um tempo de (P-1)*(L/R)
- Agora prosseguimos como último pacote até o final, que já sabemos que a fórmula é N*(L/R)
- Logo, o *delay* de transferência de P pacotes por N ligações é: $d_{p\ em\ n} = (P-1+N) * (L/R)$.

Como é de se imaginar, *switches* de pacote podem possuir múltiplas ligações e para cada ligação existe um *buffer* de saída (ou melhor, fila de saída). Este mecanismo é muito importante pois, caso um pacote deseje percorrer uma ligação que está ocupada, o pacote poderá esperar por sua vez na fila de saída. Por causa disso, além do *delay* de transferência, ainda há o *delay* da fila, sendo variável de rede para rede, a depender do tráfego.

Buffers de saída são finitas, por esta razão é capaz de que um pacote que foi enviado por um sistema final encontre o seu respectivo *buffer* cheio. Para este caso, há duas soluções: ou o pacote que foi enviado é perdido na transmissão ou alguém na fila cede o seu lugar e é perdido na transmissão. Em ambos os casos ocorre o que se chama de perde de pacote.



A pergunta que fica é? Como o *switch* sabe qual ligação enviar o pacote? A resposta é bem simples, depende de como está implementado na rede do computador. Vamos iremos focar em como ocorre na Internet.

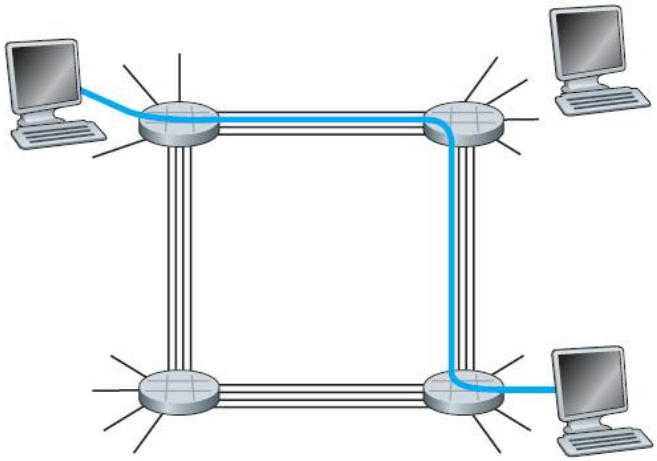
Cada sistema final possui um endereço de IP na internet, então quando um sistema final deseja enviar um pacote para outro sistema destino, o endereço de IP do sistema destino é colocado na frente de cada pacote. Deste modo, cada *switch* sabe para onde deve enviar o pacote. Mais precisamente, cada *switch* possui um tabela de encaminhamento que diz por qual saída o pacote deve seguir. Estas tabelas são configuradas pelo protocolo de roteamento (que existe vários). É neste processo que ocorre o roteamento – determina a porta origem-destino tomada pelos pacotes, e o repasse – move a entrada do roteador para a saída apropriada.

Comutação de Circuitos

A outra maneira de enviar dados através da rede é usando a técnica de comutação de circuitos. Nesta técnica, os recursos que são utilizadas ao longo do

rede (*buffers, ligações, taxa de transmissão*) são reservados, diferentemente dos comutadores de pacote, que não são reservados, ou seja, os recursos são usados quando necessitados e como consequência, pacotes talvez precisem esperar por um canal de comunicação abrir uma vaga. Esta é a ideia de reservas de restaurante, quando reservados, você tem acesso imediato à mesa, assim como, quando reservados, você tem acesso imediato à ligação.

O princípio de comutação de circuitos nasceu da forma como é feita a conexão de dados via telefone. Quando uma pessoa quer enviar um dado para outro via rede telefônica, a rede tem de estabelecer e manter uma conexão a outra fonte (conhecido como *bona fide* ou circuito). Quando esta conexão for estabelecida, uma parte da taxa de transferência é reservado para a conexão de modo a garantir uma taxa de envio constante.



A figura acima explica bem como funciona a comutação de circuitos. Na figura, temos 4 *switches* cada um com 4 circuitos cada. Se considerarmos que cada ligação tem 1Mbps de taxa, então cada circuito terá 250Kbps de taxa dedicada. Além disso, host A reserva o segundo circuito do primeiro *switch* e reserva o primeiro circuito do segundo *switch* para fazer sua transferência. Assim funciona a comutação de circuitos.

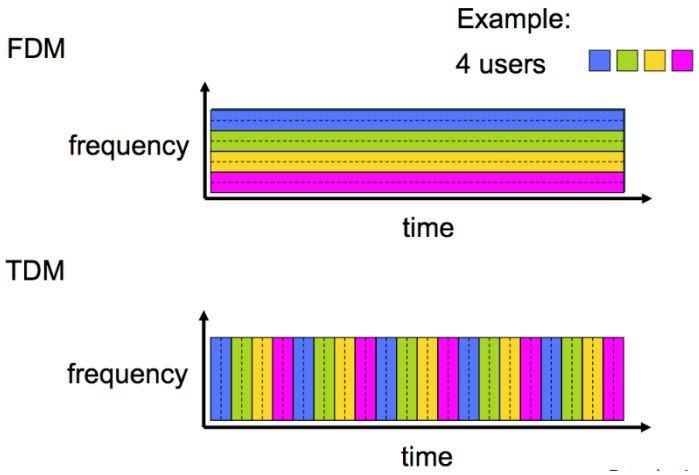
Na Internet, a situação é um pouco mais delicada, pois o pacote não tem como fazer a reserva de um circuito, logo, por mais que ele reserve circuitos da rede interna, ao chegar na Internet, ele poderá sofrer *delay* de *buffer*.

Na implementação de uma comutação de circuitos, existem duas maneiras de implementar os circuitos numa ligação:

- Multiplexação por divisão de frequência (FDM)
  Na divisão de frequência, o espectro de frequência da ligação é dividido entre as conexões, de maneira análogo a como se faz na estações de rádio, em que cada estação recebe uma frequência do espectro de banda.

- Multiplexação por divisão de tempo (TDM)
  Na divisão por tempo, o tempo é divido em quadros de durações fixas e cada quadro é divido em um número fixo de caixas de tempo. Quando é estabelecida uma conexão, uma caixa de tempo em todos os quadros é oferecida ao circuito, e unicamente a ele, para o envio de dados.
  Sua taxa de transmissão é calculada da seguinte forma:

- Seja Q a quantidade de quadros/segundo
- E seja B a quantidade de bits em cada caixa de tempo
- Então a taxa de Transmissão T no circuito é definido por: T = Q*B.



Os defensores da técnica de comutação de pacotes, argumentam que a comutação de circuitos é um desperdício da rede, visto que os circuitos ficam inativos em períodos de silêncio, ou seja, quando um circuito é reservado, mesmo quando ele não está sendo utilizado, outros ficam impossibilitados e utilizá-lo. Além disso, eles também afirmam que é bastante complexo implementá-los.

Mas pensemos, e se quisermos saber quanto tempo levará para transportar um certo dado de uma ponta a outra. Vamos considerar o seguinte exemplo: Um dado de tamanho 640.000 bits será transportado em uma rede com comutação de circuitos. Vamos supor que esta rede usa o sistema TDM com 24 slots (caixas de tempo) e que a taxa de transferência da ligação é de 1,536Mbps e que custa 500msec para estabelecer uma conexão. Vamos então montar os passos:

1. Se para cada caixa de tempo, temos um circuito associado, então teremos 24 circuitos.
2. Nossa banda larga total é de 1,536Mbps, que se dividirmos pela quantidade de circuitos (24), então teremos 64Kbps de banda para cada circuito – Fazemos isso porque temos que dividir a banda para cada slot de tempo.
3. Como foi reservado um certo circuito para a transmissão, então teremos que transmitir os 640.000bits a um taxa de 64kbps.
4. Logo gastaremos 10s para transmitir os dados de uma ponta a outra
5. Mas precisamos acrescentar os 0,5s para estabelecer conexão. Então, ao final, gastaremos 10,5s para fazer todo o processo.

Um fato muito interessante sobre comutação de circuitos é que não importa quantas ligações há entre os dois sistemas finais, o tempo seria o mesmo.

Vamos só citar um exemplo da comutação TDM dos Estados Unidos. Lá, eles usam a Portadora T1, com 1 bit a mais para sincronizar os 24 canais que utilizam para transferência de dados. Como cada canal carrega 8 bits (com 1 para controle e 7 de dados), podemos então dizer que carregamos 193 bits ao total. Além disso, por padrão, carregamos 8.000 amostras por segundo (faixa de 4kHz), então nossa taxa de transferência da banda fica em 1.544Mbps. Um fato curioso sobre esta conexão é

que podemos fazer uma multiplexação com ela de modo a gerar canais com maiores velocidades (combinando 4, 6, 7, ... Portadoras T1 por vez).

Comutação de Circuitos X Comutação de Pacotes

Muitos, contra a comutação de pacotes, argumentam que ela não é eficiente para transmissão ao vivo de dados (como chamadas de telefone e vídeo conferência) por causa do *delay* imprevisível que ele possui e também da questão da fila de pacotes. Já os defensores afirmam que ela tem um melhor compartilhamento de taxa de transmissão e é mais simples, mais eficiente e menos custosa de implementar.

Vamos pegar dois exemplos simples, sem usar muitos dados, para entender melhor a ideia. Imagine que você tem uma X quantidade de pessoas querendo usar a rede para enviar pacotes. Se implementarmos usando a comutação de pacotes pode haver uma fila de espera caso ultrapasse a quantidade de pessoas que enviam pacotes ao mesmo tempo, gerando assim *delay*. Caso contrário, utilizando a comutação de Circuitos, teremos uma quantidade F de quadros (*frames*) por segundo e uma quantidade S de caixas de tempo (*slots*). Sabendo que o usuário gera dados a uma taxa de Tu Mbps, que a maior parte do seu tempo ele não está enviando dados, e que a taxa de transferência de dados da ligação é de T Mbps teremos o seguinte cenário:

- Com a comutação de circuitos, poderemos suportar no máximo T / Tu usuários utilizando a rede.
- Além disso, caso um usuário deseje enviar um arquivo quando ninguém estiver efetivamente utilizando a rede, mas ainda sim conectado, ele terá de espera que seu slot de tempo chegue para poder enviar. De modo que a rede está sendo subutilizada.
- Se utilizarmos a comutação de pacotes, não há número máximo de pessoas comutando, mas se a soma das taxas de transferência dos usuários passar da capacidade da ligação, então pacotes começarão a sofrer *delay* de espera (do *buffer*). No entanto, a probabilidade da quantidade passar, em sua maioria, é bastante pequeno (são feitos alguns cálculos para saber), de modo que a rede é construída para suporta um número esperado e eficiente.
- Concomitantemente, na comutação de pacotes, caso um usuário deseje enviar um arquivo quando outras máquinas não estão utilizando, ele obterá toda a alocação de banda, não tendo de esperar por sua vez.

Como se pode perceber, cada comutação tem suas características, onde podemos destacar a principal de cada, enquanto que a comutação de circuitos ela reserva parte da banda (o circuito) para a conexão – podendo desperdiçar banda com conexões que não estão sendo usadas, a comutação de pacotes se preocupa com a demanda da rede, oferecendo banda unicamente para aqueles que estão pedindo.

Rede de Redes

Como vimos, há vários modos e tecnologias que nos conectam a internet, e existem uma grande gama de ISPs (*Internet Service Provider*). No entanto, o mais importante é tornar essas isoladas reder em uma grande rede que forma o que conhecemos por Internet. A primeira e mais óbvia solução seria ligar todo os ISPs uns com os outros de modo que qualquer sistema final teria acesso a outro sistema final. Mas isto é um método bastante custoso e existe outra solução mais viável.

Para entender melhor a solução, vamos construir ela por partes.

1. No topo da cadeia, temo os fornecedores de informação – como Google e Facebook – que possuem seus próprios servidores de trocam informação entre si sem depender de conexões com camadas intermediárias. Além disso, em alguns regiões, essa camada tem acesso direto a camadas inferiores, sem ter de passar por camadas intermediárias.
2. ISP de Primeira Camada é a camada responsável por fornecer acesso à Internet para camadas mais regionais – que são inferiores a de Primeira Camada, mas também podem oferecer acesso direto para o usuário final sem ter de passar por uma camada regional.
3. ISP de Região é uma camada inferior a de Primeira Camada e faz basicamente a mesma função, exceto que em escala menor e unicamente para os usuários finais.
4. Usuários finais ou acesso ISP são os que dependem de camadas superiores para acessar a rede da Internet, podendo ser de pessoas a empresas.

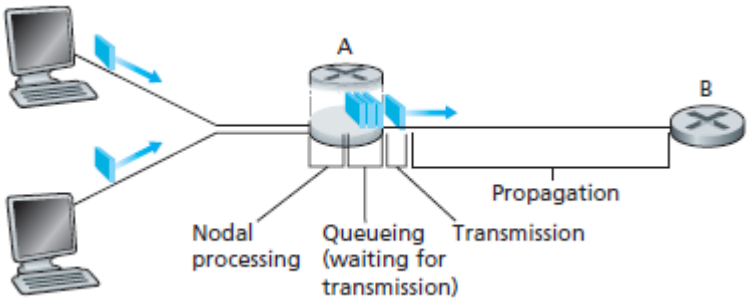
Mas para total hierarquia se manter, é necessário que haja comunicação entre as partes ou compartilhamento de acesso para aumentar a transferência. Com isso surge quatro novos conceitos:

- Pontos de Presença (PoP) → É um conjunto de roteadores, no mesmo espaço, na rede do provedor de acesso onde o ISP cliente pode se conectar.
- Multi-homming → É o ato de se conectar a mais de um provedor de ISP. Ressaltando de isso só pode acontecer com camadas inferiores para superiores, ou seja, uma ISP regional não pode fazer multi-homming com outro ISP regional, mas pode fazer com dois ou mais ISP de Primeira Camada.
- Peering → É o ato em que ISPs de mesma camada de hierarquia se conectam diretamente e o tráfico entre eles ocorre de maneira gratuita, já que não dependem de uma camada superior para fornecer conexão.
- Pontos de troca de Internet (IXP) → É um espaço físico com seus próprios *switches* em que múltiplos ISPs podem fazer peering.

Atrasos de envio de pacotes e Vazão

Existem quatro tipos de atrasos nos nós

- Tempo de transmissão
  É o tempo que leva para a banda transmitir o bit para o enlace, ou seja, é a relação do tamanho do pacote sobre a largura de transmissão da banda.
- Tempo de Propagação
  É o tempo de transporte em meio físico e corresponde literalmente ao comprimento do enlace sobre a velocidade do meio (constante que é 2*10^8 m/seg)
- Atraso de enfileiramento
  É o tempo de espera que o pacote passa na fila de transmissão do roteador quando o enlace do pacote já está ocupado com uma transmissão.
- Tempo de processamento de nó
  É o tempo que leva para o roteador verificar o pacote, seja seus erros, até o enlace que ele deve enviar.



O tempo total do nó (*host* ou roteador) é definido como a soma de todos os tempos parciais. E cada termo deste soma é relativo de acordo com a realidade da transmissão.

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

Atraso de enfileiramento

Pelo fato de termos um atraso de enfileiramento que varia de acordo com a taxa de chegada dos pacotes, então vamos estudar melhor este caso. Vamos definir que A é a taxa de chegada dos pacotes ao enlace, L o tamanho de cada pacote e R a taxa de transmissão do roteador, podemos definir que a intensidade de tráfico é L*A/R, para uma fila de tamanho infinito (não real). Com isso fazemos três análises:

1. Quando LA/R > 1
   Quando este caso ocorre, indica que a chegada de bits está acima da transmissão e a fila tende ao seu limite (ou seja, infinito em teoria) e pode ocorrer perda de dados.
2. Quando LA/R = 1
   O pacote sofre um atraso que depende da taxa de chegada dos pacotes e que varia muito. Se considerarmos que os pacotes chegam a uma taxa

de (L/R)N, sendo N o número do pacote, então o n-ésimo pacote terá *delay* de (N-1)L/R segundos.

3. Quando LA/R → 0
   O pacote não sofre atraso considerável.

Vazão

Podemos definir a vazão como a taxa de bits por tempo na qual os bits são transferidos entre os transmissor e o receptor. Podemos então analisar a vazão média fim-a-fim:

- Quando Rt < Rc → Então a vazão média é restringida pela vazão do transmissor.
- Quando Rt > Rc → Então a vazão média é restringida pela vazão do receptor.

Essa análise é conhecida como gargalo, pois a menor transmissão é que limita o envio de dados. Além disso, quando um transmissor está conectado a mais de um receptor, então sua vazão fica dividida por todos os receptores (Internet).

Divisão de Camada

Bem, podemos dizer muitas das vantagens de separar em camadas o sistema de redes. Além da facilidade de manutenção e atualização, há melhor compreensão de como as partes de relacionam fica mais evidente. Mas vale salientar que é necessário dividir otimamente de modo a não tornar ainda mais complexo o sistema.

No nosso caso, nós temos 5 camadas, cada uma com seus protocolos para realizar as funções:

1. Aplicação
   Dar suporte as aplicações de rede (FTP, SMTP, HTTP);
2. Transporte
   Gerenciar a transferência de dados entre as aplicações (TCP, UDP);
3. Rede
   Repassar datagramas (pacote não orientado a conexão) da origem até o destino (IP, protocolos de roteamento);
4. Enlace
   Transferência de dados entre elementos de rede vizinhos, ou seja, entre dois pontos e não fim-a-fim (PPP, Ethernet, 802.11)
5. Física
   Gerenciamento da transferência dos bits no meio físico

Modelo de Referência ISO

Criou um modelo chamado de OSI (*Open System Interconection*) para facilitar a comunicação, através da padronização. Possui 7 camadas, as 5 iguais ao que usamos, as outras duas são:

6. Apresentação
   Permitir a aplicação interpretar os dados
7. Sessão
   Sincronização, verificação e recuperação da troca de dados

Encapsulamento

Existem dois fatos importantes com relação ao encapsulamento. A diferença básica entre roteador e *switches* é que roteador vai até a camada de rede, propagando a mensagem unicamente para o destino, enquanto que os *switches* vão apenas a camada de enlace, propagando para todos.

O outro fato é o formato como a mensagem é enviada. Além dos bits da mensagem que foi enviada pela camada de aplicação (a mensagem), a camada de transporte coloca o cabeçalho de segmento, a camada de rede coloca a de datagrama e a camada de enlace coloca o quadro (*frame*). Todos eles essenciais para a interpretação e envio dos dados de maneira efetiva e correta.

Segurança em Rede

Bem, o é muito importante a segurança em redes por isso temos de entender como ela pode ser atacada, como pode ser defendida e como projetar barreiras contra ataques. Mas o pior fato de todos é que a rede não foi projetada para ser segura, o que pesa bastante hoje em dia.

Malwares podem atacar um hospedeira de duas maneiras básicas:

- Vírus:
  Infecção autoreplicante através da recepção/execução de um objeto (email, por exemplo).
- Worms:
  Infecção autoreplicante através da recepção passiva de um objeto que se autoexecuta.

  Obs:Uma infecção autoreplicante é aquele com capacidade de procurar novos hospedeiros depois que já infectou um.

Além disso, temos também os *spywares*, que são ataques que visam espionar a máquina do usuário, sabendo tudo o que a pessoa está executando. Máquinas infectadas podem entrar numa *botnet*, que começa a gerar spams e ataques DDoS.

Um DoS é um tipo de ataque que sobrecarrega o servidor através de um ataque múltiplo, utilizando *botnet* para manipular as outras máquinas. Existem três tipos de DoSs:

- Ataque de vulnerabilidade: Envia um pacote específico direto para o servidor responsável que é capaz de da rum *crash* na máquina.
- Sobrecarga de banda: Envia uma imensa quantidade de pacotes, de tal forma que o servidor não consegue dar conta e tem uma sobrecarga, impedindo que usuários legítimos utilizem.
- Sobrecarga de conexão: Estabelece várias conexões (semiabertas ou abertas) com servidor de modo a impedir que outros usuários façam.

Existem também os analisadores (farejadores) de pacotes que todos os pacotes que estão a vir na rede que ele se encontra, ele pode analisa os dados, tendo conhecimento de todas as informações. Concomitantemente, há os imitadores (*spoofing*) de pacotes de IP, que enviam pacotes com endereços origem falsos.

História da Rede e da Internet

- 1961 – 1972 Estreia da comutação de pacotes
  Kleinrock mostra a teoria das filas e mostra a eficiência da comutação de pacotes, aplicando então a comutação em redes militares, permitindo a criação da ARPAnet
- 1972 – 1980 Interconexão de redes novas e proprietárias
  Criação da primeira conexão via satélite no Hawai (ALOHAnet). Além disso Cerf E Kahn criam uma arquitetura para a interconexão de redes (arquitetura atual) com os seguintes princípios:
    - Minimalismo, autonomia – não é necessária nenhuma mudança interna para interconectar redes
    - Modelo de serviço *best effort*
    - Roteadores sem estados (apenas envia o pacote)
    - Controle descentralizado

  Cada empresa/instituição gera sua própria arquitetura, mas Governo Americano pede uma padronização da arquitetura
- 1980 – 1990 novos protocoles, proliferação de redes
  Protocolos como TCP/IP, SMTP e outros são criados. Além disso, DNS é criado para traduzir mnemônicos para endereço de IP. Criado o controle de congestionamento do TCP devido ao crescimento de uso da rede.
- 1990 – 2000 comercialização, a Web e novas aplicações
  A Web surge utilizando o hypertext (1945) com HTML e HTTP e a Internet começa a ser comercializada com aplicações para troca de mensagens instantâneas e compartilhamento de arquivo P2P. Com o crescimento da Internet, a preocupação com sua segurança também aumenta.
- 2000 – diante
  Surgem tablets e smartphones com acesso à Internet, computadores na nuvem, redes sociais, provedores de serviço com suas próprias redes. Crescimento da ubiquidade de acesso à Internet de acesso sem fio.

Princípios da Camada de Aplicação

Alguma aplicações de rede

- Correio Eletrônico
- Netflix
- Skype
- Compartilhamento de arquivo P2P
- Jogos em rede
- *Streaming* de vídeos
- Busca
- ETC.

Aplicação em rede

Programas em rede são caracterizados por executarem em diferentes sistemas finais e comunicarem-se através da rede. Além disso, eles não são relacionados ao núcleo da rede, isto é, não é necessário saber nenhuma informação sobre a mensagem para ser capaz de enviá-la, apenas saber quem é o destino. Com isso, podemos afirmar que o núcleo de rede não executam as aplicações dos usuários.

Arquitetura cliente-servidor

É a arquitetura mais comum na rede e é esperado algumas características importantes de ambas as partes para que esta relação dê certo:

- Servidor
  - Sempre ligado,
  - Endereço de IP permanente,
  - Escalabilidade com *data centers* (conjunto de servidores).
- Clientes
  - Comunicam-se com o servidor,
  - Podem estar conectados intermitentemente,
  - Podem ter endereços de IP dinâmicos,
  - Não se comunicam diretamente com outros clientes.

Arquitetura P2P

Enquanto que a comunicação entre cliente-cliente depende de um servidor, na P2P não há servidor e possui as seguintes características:

- Não há servidor sempre ligado,
- Sistemas finais se comunicam diretamente,
- Pares solicitam serviços de outros pares e, em troca, oferecem serviços para outros parceiros,
- Autoescalabilidade: novos pares trazem nova capacidade de serviço assim como novas demandas por serviço,
- Pares estão conectados intermitentemente e mudam endereços de IP, possuem baixa segurança, sobrecarregam ISPs e necessitam de incentivos para compartilhamento, tendo um gerenciamento complexo.

Comunicação entre Processos

Já sabemos o que é um processo, mas como eles se comunica? Depende da seguinte situação:

- No mesmo sistema final
  Usam comunicação interprocessos, definida pelo sistema operacional.
- Sistemas finais distintos
  Trocam mensagens através da rede, onde o cliente inicia a comunicação e o servidor espera ser contactado (na arquitetura P2P, os processos são comunicados em estilo cliente-servidor, onde o cliente é aquele que pede o arquivo e o servidor, o que oferece).

Sockets

Os processos enviam/recebem mensagens para/dos *sockets*, sendo este análogo a uma porta. Um processo transmissor envia mensagens através da porta e assume a existência de infraestrutura de transporte no outro lado da porta que faz com que a mensagem chegue ao *socket* do processo receptor.

Podemos dizer que o *socket* está entre a aplicação e o SO, interligando os dois, ou seja, faz a ligação entre a camada de aplicação e a camada de transporte.

Endereçamento de Processos

Para que um processo receba mensagens, ele necessita de um identificador, conhecido como endereço IP único de 32 bits. Mas isso não é suficiente para endereçar ao processo certo. É necessário também incluir o número de porta associado ao processo.

Por exemplo:

- Endereço de IP:128.119.245.12
- Número de porta:80

Protocolos da Camada de Aplicação

Temos que agora entender como fazemos para enviar as mensagens das aplicações para a rede e quais padrões elas seguem. Os Protocolos da camada de aplicação definem as seguintes características para envio/recebimento de mensagens:

- Tipo de mensagens trocadas:
  Exemplo: Mensagens de requisição e resposta.
- Sintaxe das mensagens:
  Campos presentes nas mensagens e como são identificados.
- Semântica das mensagens:
  Significado da informação nos campos.
- Regras:
  Determina como e quando os processos enviam e respondem às mensagens.

Além disso, protocolos da camada de aplicação podem ser abertos (definidos em RFCs, permitindo interoperação, HTTP, SMTP) ou proprietários (definidos pelas companhias, Skype).

Necessidade das aplicações

Temos de saber quais são os serviços que as aplicações necessitam para executar em rede e isso será fornecido pelos protocolos de transporte. Temos os seguintes serviços:

- Integridade de dados (sensibilidade a perdas): Necessitam que os dados enviados sejam idênticos para os dados recebidos.
- Temporização (sensibilidade a atrasos):
  Necessitam que os dados sejam enviados rápidos por alguma razão, e acabam sendo tolerantes a perdas por esta razão.
- Vazão (*throughput*):
  Necessidade ou não de uma banda mínima para serem capazes de ser enviadas.
- Segurança
  Proteção e integridade dos dado enviados.

A WEB e o protocolo HTTP

A WEB

É WEB é uma aplicação da rede que consiste basicamente de uma troca de mensagens instantânea e sobre demanda entre o Browser Cliente e o Servidor Web, considerado uma revolução para a época.

Protocolo HTTP (*HyperText Transfer Protocol*)

Protocolo da camada de aplicação da Web, no estilo cliente-servidor, onde o browser é cliente, fazendo uma requisição de um objeto da web para o servidor da web, que responde a esse pedido de alguma maneira. Ou seja, podemos dizer que HTTP é um protocolo de requisições de dados entre aplicações que define como cada uma das partes deve requisitar e responder as mensagens, mas não tem nada haver em como a mensagem será interpretada pelo browser.

Algumas de suas características são:

- Utiliza transporte TCP:
  O protocolo de HTTP não se preocupa com a confiabilidade das mensagens, ele passa essa tarefa ao TCP.
- É sem estado:
  Servidor não mantém informação sobre pedidos do cliente.
- Usa a arquitetura cliente-servidor:
  Isso quer dizer que o servidor fica sempre ligado e tem um endereço de IP permanente.

Além disso, há dois tipos de HTTP

- Persistente
  Múltiplos objetos podem ser enviados sobre uma única conexão TCP entre cliente-servidor.
  Existe também um temporizador para limitar o tempo da conexão.

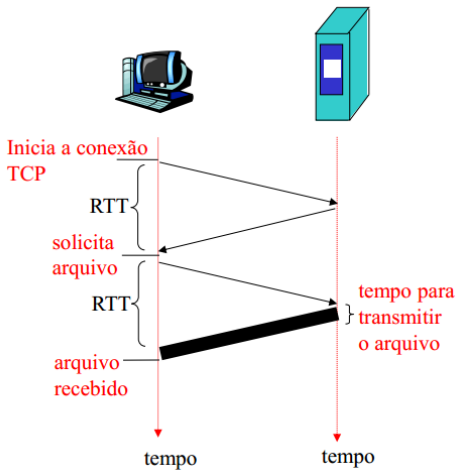
Conexão fica aberta após envio de arquivo, sendo os próximos arquivos pedidos enviados pela mesma conexão, enviando o objeto logo que referenciado, podendo o tempo final ser um único RTT

- Não persistente
  Envia no máximo um objeto na conexão TCP (a conexão é então encerrada).
  Baixar múltiplos objetos requer ter múltiplas conexões.
  Requer 2RTTs para cada objeto, abrindo conexões paralelas para conseguir os objetos referenciados.
  Além disso, o SO aloca recursos do hospedeiro (*overhead*), para cada conexão TCP.

Exemplo de Não persistência:

1. Suponha que um usuário digita uma certa url X,
2. Cliente http inicia conexão TCP a servidor http (processo), sendo porta 80 padrão para http,
3. Servidor http no hospedeiro de x espera pela conexão TCP na porta 80, aceitando conexão do cliente,
4. Cliente http faz pedido ao servidor através do *socket* da conexão, indicando o objeto que o cliente quer receber,
5. Servidor http formula mensagem de resposta, enviado o pedido, se existente, e enviando via *socket*, encerrando a conexão
6. Cliente http recebe a mensagem de resposta contendo o objeto desejado, que contém 10 objetos jpeg,
7. Como é um HTTP não persistente, terá de repetir os passos de 1 a 6 para conseguir os 10 objetos.

Para calcular o tempo ida e volta de um pacote de cliente a servidor em um HTTP não persistente, temos de perceber que há um custo de um RTT (*round trip time*) para iniciar a conexão TCP entre cliente servidor, mais outro RTT para o pedido HTTP e o retorno dos primeiros bytes da reposta HTTP, e agora podemos transferir o arquivo, que tem um certo tempo t', logo, temos como tempo total: total = 2RTT + t'.



Tipos de mensagem HTTP

Há dois tipos de mensagens de tipo HTTP, tipo requisição e tipo resposta:

- Tipo Requisição
  É uma mensagem no formato ASCII
  Possui como formato geral:
  a. Linha de requisição
  b. Linhas de cabeçalho
  c. Linha em branco
  d. Corpo da entidade (usado no método POST)

  Há o método GET e o método POST, sendo o POST o envio de mensagens através de formulários, o conteúdo está no corpo da mensagem, o GET retorna o objeto sem envio de informações no corpo. Existe um GET condicional que envia páginas apenas se elas não tiverem sido modificadas, ou seja, se o cliente possuir no cache a versão atual. Não possuindo, o cache irá procurar o servidor a nova versão.
- Tipo Resposta
  Possui como formato geral:
  a. Código de Status
  b. Linhas de cabeçalho
  c. Linha em branco
  d. Dados a enviar

Cookies: mantendo estado da conexão

Cookies, como já é percebido, mantem o estado da conexão, mas, para isso, há quatro componentes:

- Linha de cabeçalho do cookie na mensagem de resposta HTTP,
- Linha de cabeçalho do cookie na mensagem de pedido HTTP,
- Arquivo cookie mantido no host do usuário e gerenciado pelo browser,
- Banco de dados de retaguarda no site da web.

Deste modo, quando o usuário retornar ao site, no mesmo computador, este saberá quem é o usuário através do identificador que é dado ao usuário e salvo no banco de dados.

Através do uso de cookies, eles conseguem obter algumas informações extras sobre o computador, como autorização de acesso, recomendações, estado da sessão do usuário, recomendações, compras, etc. Isso faz com que cookies aprendam bastante sobre o usuário, como nome e email.

Cache Web (servidor proxy)

A principal ideia de um servidor proxy é agilizar o envio de mensagens respostas e reduzir o tráfego no canal de acesso através do não envolvimento do servidor mestre (origem).

Para isso, é necessário que o usuário configure seu browser para acessar via proxy. No entanto, isso não garante que o acesso será rápido, pois, caso o proxy não tenha o conteúdo, este irá de todo jeito buscar no servidor mestre.

Ou seja, em casos de páginas que são atualizadas constantemente, ter um servidor proxy não garantirá agilidade, porque páginas novas são geradas constantemente e acaba por ter de buscar estas páginas no servidor mestre.

Podemos já perceber então duas características de um cache web, ele pode se comportar como cliente ou como servidor (ele envia respostas para o cliente, mas faz pedido ao servidor, se necessário) e ele é instalado pelo ISP.

Transferindo Arquivos: FTP

O FTP é outro protocolo, talvez um dos mais antigos, utilizado para a transferência de arquivos entre sistemas finais. Podemos dizer que este utiliza o modelo cliente servidor, onde o cliente é aquele que inicia a conexão para transferir ou requisitar um arquivo e o servidor é aquele que atende ao pedido, salvando ou enviando o arquivo requisitado.

Uma característica peculiar do FTP é que ele utiliza duas conexões, uma para controle (porta 21) – todas as informações de controle como identificação e comandos de requisição e inserir – e uma conexão de dados (porta 20) – para o envio de dados/arquivos. Esta característica é chamada de controle fora de faixa, oposta ao HTTP, que envia controle e dados na mesma rede e por isso é um controle em faixa.

Mas assim como o HTTP, ele também pode ser persistente ou não-persistente, mas normalmente o FTP encerra a conexão de dados após enviar o arquivo requisitado, então, se outro arquivo for desejado, outra conexão tem de ser aberta. No entanto, a conexão de controle permanece aberta durante todo o processo. Além disso, o FTP utiliza o TCP como protocolo de transporte. Concomitantemente, o FTP também possui uma lista de comandos e códigos de retorno que pega informações do usuário e retorno um estado sobre o arquivo.

No entanto, diferentemente, o FTP mantém estado no sentido de manter o diretório atual e a autenticação de acesso.

Correio Eletrônico

Existem três grandes componentes no sistema de correio eletrônico:

- Agentes de Usuário:
  Navegadores ou aplicativos que recebem e mostram o email pessoal, ou seja, o leitor de correio.
- Servidores de correio:
  Fora do agente de usuário, este sempre se mantém conectado pronto para enviar ou receber mensagens entre/dos agentes de usuário. É a nossa caixa de correio.
  Uma característica importante é sua fila de mensagens, que são as mensagens que estão a ser enviadas.
- Protocolo SMTP:
  O responsável pelas regras de formatação e comportamento no sistema de envio de correios eletrônicos entre servidores de correios.

SMTP (*Simples Mail Transfer Protocol*)

É mais outro protocolo que requer que as mensagens sejam entregues integras, por isso utilizam o protocolo de transporte TCP, utilizando a porta 25. Outra característica importante a ressaltar é que a transferência entre os servidores de correios é direta, ou seja, não utilizam nenhum sistema intermediário para enviar ou receber as mensagens de outros servidores. SMTP também utiliza o sistema de comandos/respostas em ASCII, só que em 7-bits, o que é uma dor de cabeça atualmente, pois é necessário converter os anexos para ASCII na hora do envio e reconverter a binário, na hora do recebimento.

Para o envio de email se tornar possível, é necessário que a transferência passe por três fases:

1. Handshaking (saudação)
2. Transferência das mensagens
3. Encerramento

Obs: Esta saudação é diferente da saudação do TCP, que confirma apenas que a conexão foi estabelecida. Logo, há duas saudações, a do TCP e a do SMTP.

Um passo-a-passo rápido entre dois agentes de usuários seria o seguinte:

1. Usuário utiliza seu agente para enviar mensagem para um certo endereço;
2. Agente de usuário envia mensagem para o seu servidor de correio, que coloca este mensagem na fila de mensagens;
3. Lado cliente SMTP abre um conexão TCP com o servidor de correio do endereço, e envia a mensagem, se possível;
4. Servidor de correio do outro endereço envia a mensagem para a caixa de entrada;
5. Quando o outro endereço visualizar seu servidor de correio em sua agente de usuário, verá a mensagem mandada.

Informações adicionais sobre o SMTP é que utiliza conexões persistentes, de modo a possibilitar que novas mensagens cheguem na mesma conexão utilizando uma única mensagem com múltiplas partes – mas a conexão é encerrada ao final do envio de todas as mensagens. Além disso, utiliza um código específico (*CRLF.CRLF*) para reconhecer o final da mensagem.

Uma breve comparação entre HTTP e SMTP é que enquanto o HTTP é basicamente um protocolo de retirada, ou seja, HTTP cria uma conexão quando ele requisita um objeto, o SMTP é um protocolo de envio, ou seja, cria uma conexão quando ele deseja enviar um objeto.

A necessidade de que cada usuário se conecta a um servidor de e-mail é porque, além do agente do usuário não saber se o destino não foi alcançado, seria também necessário que o usuário se mantivesse sempre conectado à Internet para ser capaz de receber os novos e-mails.

Como já foi dito, o SMTP é um protocolo de envio, ou seja, ele é usado quando um usuário desejar enviar uma para o seu servidor de e-mail ou quando um servidor de e-mail deseja enviar uma mensagem para outro. No entanto, quando um agente de usuário deseja adquirir uma mensagem de seu servidor de e-mail, ele utiliza outros protocolos para puxar a mensagem: POP3, HTTP ou IMAP.

Protocolos de Acesos ao Servidor POP3 e IMAP

Estes protocolos servem para impedir que o pessoas não autenticadas tenham acesso ao servidor que não lhes foi designados e para capturar as mensagens de servidor.

O Protocolo POP3 utiliza três fases para fazer isto, a de autenticação (autentica do usuário), a de transação (tem acesso as mensagens, podendo deletá-las ou vê-las) e a de atualização (ocorre quando o usuário sai do centro de e-mail e então o servidor atualiza as mensagens que forem para ser deletadas). Mas este protocolo tem um grande defeito, uma vez que o usuário viu o e-mail, este é deletado, de modo que não pode ser acessado de outro cliente. Concomitantemente, o POP3 não mantém estado entre conexões, o que simplifica bastante a sua implementação.

Podemos dizer que o IMAP é um POP3 mais eficiente, capaz de manter as mensagens – mantém no servidor para qualquer cliente autenticado acessar. Além disso, permite que o usuário organize as mensagens em pastas e estados, com o compromisso de que o IMAP irá manter ambos salvos. Por esse fato, o IMAP se torna um protocolo de acesso ao servidor muito mais complexo.

DNS: Domain Name System

Como ficar acessando endereços pelos seus IPs é uma tarefa bastante exaustiva, foi criado uma tecnologia para mapear o IP a um nome, facilitando o acesso a endereços, numa visão humanista. Para possibilitar tal sistema, existe uma base de dados distribuída, onde cada banco de dados cuida de um parcela dos nomes (servidores de nomes), obedecendo uma hierarquia.

No entanto, isto não é feito simplesmente do nada. Para realizar a tradução, foi necessário criar um protocolo na camada de aplicação responsável por traduzir o nome para o respectivo endereço de IP, através da comunicação entre sistemas finais, roteadores e servidores de nomes.

Esta tecnologia foi expandida de tal forma que é permitido ter vários nomes para o mesmo endereço de IP, sendo este endereços secundários chamados de apelidos. Esta ação é chamada de *aliasing*. Também foi criado apelidos para servidores de email. Além disso, foi criado uma distribuição de carga, de modo que pode haver um conjunto de endereços de IP para um mesmo nome.

Para terminar esta introdução, vamos só justificar porque não ser centralizado. Além de ficar dependente de um ponto único de falha, o tráfego para o mesmo local seria exuberante e o acesso poderia ficar lento para localidades longe do banco central. Como último ponto, a manutenção de um banco único se torna muito complexo para uma entidade.

Base de Dados Hierarquica e Distribuída

Imaginemos um sistema hierárquico qualquer. O sistema de DNS obedece a mesma ideia, de cima para baixo, onde os servidores raiz procuram o servidor responsável por uma certa entidade e este servidor fica responsável por encaminhar ao servidor do domínio, que finalmente retorna o seu endereço de IP.

Mas não ficamos só a depender dos servidores raiz. Existe um servidor local intermediário que só passa o comando para um servidor raiz caso este não consiga

localizar / traduzir o nome dado pelo usuário. O servidor raiz, quando requisitado, retorna o endereço para o servidor local para este memorizar o novo nome.

Servidor DNS Local

Podemos dizer que este servidor local não pertence a hierarquia. Esta é gerenciada pelas ISP's. Deste modo, quando um usuário manda um requisição de DNS, este vai para o DNS Local, verifica se em sua memória cache tem a tradução e envia para o usuário. Caso contrário, o DNS pergunta para o servidor raiz, que indica para outro servido, e que indicam para outros servidores, até chegar no destino desejado. Exista também outro método, recursivo, que o servidor que foi repassado a pergunta salva o estado e aguarda a resposta retornar.

Uma vez que o servidor DNS local acha mapeamento, este salva no cache dele, que ficará durante um tempo, até que o seu tempo expire. Apesar disso, pode ser que o mapeamento no cache esteja desatualizado, no entanto, não há solução fixa, o jeito é esperar o tempo em cache expirar para pedir o novo link.

Formato de Mensagens

Diferentemente dos outros protocolos, o formato RR (*resources registers / records*) é igual para mensagens de resposta e pedido, com o mesmo ID para identificar a que pertence a resposta. Além disso, existe *flags* para identificar se deseja-se uma recursão, se é permitido uma recursão e também identificar se a resposta vem do servidor oficial ou de um cache.

Concomitantemente, irá dizer o número de perguntas, número de respostas, número de RRs com autoridade e número de RRs adicionais, juntamente as perguntas, respostas, RRs com autoridade e RRs adicionais, respectivamente.

Ataques Sobre DNS

O DNS pode sofrer vários ataques, até por ser um servidor. Vamos explorar alguns deles:

- Ataque DDoS
  Pode-se atacar ou aos servidores raiz ou aos servidores secundários como TLD. Atacar o servidor raiz nunca se demostrou eficiente, pois o servidores locais são muito mais utilizados, de modo que uma ataque DDoS aos servidores locais serão muito mais suscetíveis.
- Ataques de Redirecionamento
  Existe um ser no meio da transmissão que está enviando respostas erradas e copiando os dados. Isso é feito através de um envenenamento de DNS, onde o servidor DNS começa a enviar a respostas erradas.
- Exploração de DNS para DDos

Aplicações P2P

Já sabemos que P2P não utilizam servidores para realizar a transferência de arquivos, é uma troca de dados entre dois sistemas finais sem intermediário, trocando o endereço de IP dinamicamente.

Vamos comparar os tempos de enviar um arquivo para N sistemas finais na arquitetura cliente-servidor e na arquitetura P2P.

- Cliente-servidor
  Dado que a velocidade de banda do servidor de Us e a do cliente é de Cs, no melhor caso, esta resposta será o máximo entre N*F/Us e F/Cs(min), ou seja, o atraso é, no mínimo, o máximo entre fazer a cópia dos arquivos para os N sistemas finais ou o tempo de download do cliente com menor banda. Isso demostra uma relação linear entre a velocidade e a quantidade de requisições.
- P2P
  Como o servidor precisa enviar apenas um único arquivo, visto que os clientes que o obtém se tornarão "servidores" que também irão oferecer o arquivo. Assim como no cliente-servidor, existe também o atraso do que faz o download com menor banda. No entanto, os novos clientes irão sugar o arquivo para si, mas concomitantemente irão se tornar servidor, ou seja, existe uma crescimento linear tanto na quantidade de pessoas que capturam o arquivo como na quantidade que se tornam servidores.

Com isso, o P2P acaba por ter uma curva crescente de tempo muito menor do que a curva de crescimento do cliente-servidor. Lembrando que isso é apenas a potência, na realidade isso pode mudar.

Estudo sobre os princípios do Bittorrent

Assim como se imagina, o arquivo é dividido em pedaços (de 256kb), onde nem todo mundo tem todo o arquivo, mas o pedaço que ele tem já está disponível para alguém capturar. Existe um servidor aqui que registra os participantes de uma torrente, onde uma torrente é o grupo de pessoas que estão trocando arquivos.

Um novo sistema que acaba de entrar numa torrente irá começar a obter os arquivos, se conectando com a alguns pares dos que estão na lista do *tracker*. Mas, como um o sistema obteve o bloco, este já está apto a distribuir. Além disso, é possível trocar de pares (pois ele podem acabar saindo da conexão).

Para obter um bloco, o sistema que deseja pergunta aos seus vizinhos quais blocos eles tem e começa a pedir, inicialmente, os mais raros. Do mesmo modo, o sistema final, com o bloco disponível, envia para os quatro vizinhos que estão lhe enviando bloco com a taxa mais alta – outros pares ficam sufocados, sendo o sistema reavaliado a cada 10 segundos, assim como a cada 30 segundos seleciona outro par para possivelmente se tornar os top 4.

DHT (*Distributed hash table*)

Além dos torrentes, também existe este estilo de compartilhamento baseado em uma base de dados P2P distribuída. É uma base da dados que possui duplas (chave, valor), distribuindo a duplas entre os sistemas finais. Utiliza a ideia básica de qualquer função de hashing.

Deste modo, ao consultar a DHT com a chave, a DHT retorna valores que batam com a chave. Isto facilita na hora de saber quem está com aquilo que eu preciso. Concomitantemente, sistemas finais também podem inserir pares na DHT, gerando novas possibilidades.

Numa DHT, existem vários modelos, onde o modelo básico é o circular, onde cada sistema final só reconhece o seu antecessor e seu sucessor, de modo que, ao procurar uma chave, este terá de percorrer no máximo toda o círculo para saber quem é responsável por uma chave (busca linear). A busca da chave é feito de uma função hash que assemelha a chave a um identificador (no caso do dado em aula, é um identificador binário).

No entanto, existe outro sistema que é baseado na circular, mas possui atalho, onde cada sistema, além de saber seu sucessor e antecessor, também sabe algum atalho, permitindo projetar atalhos a reduzir a busca para logarítimica.

Além disso, para resolver o problema de algum sistema sair, cada um deles ficam enviando *pings* para verificar se estão vivos, de modo que, caso o sucessor tenha saído, o sucessor imediato vira o novo sucessor.

Skype, estudo de caso

O Skype é considerado um serviço inerentemente P2P, pois, como se sabe, a comunicação é feito entre os usuários finais diretamente, sendo o protocolo proprietário, ou seja, não está sobre domínio público. Para seu estudo, foi feito uma engenharia reversa para ver como se comportava.

Na hierarquia do Skype, existem supernós que ficam responsáveis por manter a comunicação entre os sistemas, que não é um servidor. Apesar disso, existe um servidor de login, mas nada demais.

A existência de supernós é para problemas como a utilização de "NATS", onde viabilizará que dois sistemas que não conseguem se visualizar utilizem um outro sistema intermediário para trocar informações.

Introdução a Camada de Transporte

Podemos dizer que é a camada de Aplicação que fornece a comunicação lógica entre processos fim-a-fim (não confundir com a Camada de Redes, que fornece a comunicação lógica entre sistemas finais e não entre processos) executadas em diferentes hospedeiros, sendo os protocolos executados nos respectivos sistemas finais. Para o envio, o lado transmissor fragmento as mensagens em segmentos, repassando para a camada de rede. Já o lado receptor remonta as mensagens a partir dos segmentos, repassando para a camada de aplicação.

Já sabemos que existem dois protocolos de transporte:

- TCP:
  - Entrega Confiável e ordenada;
  - Controle de Congestionamento;
  - Controle de Fluxo;
  - Estabelecimento de conexão.
- UDP
  - Entrega não confiável e não ordenada;
  - Extensão sem "gorduras" do "melhor esforço" do IP.

No entanto, existem serviços que não são oferecidos por nenhum dos dois: eles não garantem atraso máximo e nem garantia de largura mínima, assim como a camada de rede.

Multiplexação e Demultiplexação

Para o envio de dados, é necessário organizá-los de alguma maneira para que fique fácil de obtê-los posteriormente. Para isso, há estes dois mecanismos que consistem basicamente de:

- Multiplexação no transmissor:
  Reúne os dados de muitos sockets, adicionando cabeçalho de transporte.
- Demultiplexação no receptor:
  Usa a informação do cabeçalho para entregar os segmentos recebidos aos sockets corretos.

Funcionamento da demultiplexação

Quando o hospedeiro recebe o pacote, este examina os datagramas IP para saber o endereço de IP de origem e de destino e também examina o segmento para saber a porta de origem e de destino. Deste modo, o hospedeiro usa ambos os dados para direcionar o segmento ao socket apropriado.

Na demultiplexação não orientada a conexão (UDP), o endereço de IP destino e o número da porta de destino são especificadas no datagrama. Quando o hospedeiro recebe o segmento UDP, este verifica o número de porta de destino no segmento, enviando o segmento UDP para o socket com o número de porta adequado. É o mais simples tanto de implementar.

Já na demultiplexação orientada a conexão (TCP), o socket TCP é identificado pela quádrupla (endereço IP de origem, número de porta origem, endereço IP destino, número da porta destino). É necessário saber estes quatro dados para que o receptor consiga enviar a mensagem para o socket apropriado. Isto é devido ao fato de que o servidor pode dar suporte a muitos sockets TCP simultaneamente, nisto cada socket é identificado pela sua quadrupla. Lembrando que para servidores web, há sockets diferentes para cada conexão de cliente e, em caso de HTTP não persistente, existirá sockets diferentes para cada pedido.

UDP: *User Datagram Protocol*

Vamos estudar agora, cada protocolo da camada de transporte, começado pelo UDP. Vejamos alguns características deste protocolo:

- Protocolo de transporte da Internet mínimo (sem "gorduras");
- Serviço "melhor esforço", segmentos UDP podem ser perdidos ou entregues a aplicação fora de ordem;
- Sem conexão, não existe saudação entre remetente e receptor, tratamento independente para cada segmento UDP;

Apesar de suas nada boas qualidades, ele tem esse *trade off* com uma grande agilidade de envio, muito utilizado por aplicações de *streaming* multimídia (tolerante a perdas e sensível a taxa), também ótimo para DNS e SNMP. A pior característica do

UDP é que a confiabilidade do transferência de dados fica por conta da camada de aplicação, aumentando o trabalho do programador.

O formato do UDP é consistente de 5 campos:

- Porta Origem e Porta Destino de comprimento de 32 bits juntos;
- Comprimento e Checksum de mesmo comprimento
- Dados de aplicação.

Podemos resumir a existência do UDP devido a seus benefícios com a eliminação de conexão, não causando retardo, simples implementação não guardando estado, cabeçalho de segmento resumido e como não há controle de congestionamento, o UDP pode transferir tão rápido quanto desejado (e possível).

Soma de Verificação (CheckSum)

Apesar de não tratar implicitamente os erros de envio, o UDP nos fornece cabeçalhos que ajudam a camada de aplicação a validá-los. O objetivo do checksum é justamente este, detectar erros grosseiros no segmento transmitido.

O checksum é feito no transmissor através da soma (por complemento a 1) do conteúdo do segmento e então coloca o complemento do valor da soma no campo de checksum. O receptor calculo o checksum da mensagem recebida, verifica se o valor bate com o checksum recebido. Se não bater, há erros, caso contrário pode haver erros ainda sim, mas são erros mais sensíveis.

Princípios de Transferência de Dados Confiáveis

Vamos estudar estes princípios pois é um fator importante na camada de enlace e na camada de transporte e é considerado um dos 10 mais importantes tópicos de redes. Vamos ver que a complexidade de implementação de protocolos de transferência confiáveis são proporcionais a não confiabilidade do canal da camada de baixo.

Iremos então, para entender os princípios, desenvolver incrementalmente os lados transmissor e receptor de um protocolo de transferência de dados confiável. Além disso, vamos considerar apenas fluxos unidirecionais (apesar das informações de controle fluírem em ambos os lados). Concomitantemente, usaremos máquinas de estados finita para especificar os protocolos do transmissor e receptor.

Implementação de transferência confiável em canal confiável

Em casos em que o canal de transferência não produz nenhum erro e nem perda de pacotes, a única função do protocolo é então enviar os pacotes, no lado do remetente, e receber os pacotes, no lado do receptor. Com isso teremos uma RMS separada para cada lado (receptor e transmissor) e ficam num único estado de envio e recebimento, respectivamente.

Implementação de transferência confiável em canal com erro de bits

Neste cenário, o canal pode trocar os valores dos bits em um pacote, então temos de recuperar os erros. Primeiro é feito um reconhecimento (*ACK*), onde o receptor avisa explicitamente ao transmissor que o pacote foi recebido corretamente, ou um (*NAK*), onde o receptor avisa explicitamente ao transmissor que o pacote não foi recebido corretamente. Em caso de *NAK*, o transmissor reenvia o pacote.

Com isso, podemos ver que foram criados dois novos mecanismos de correção de erros:

- Detecção de erros;
- Realimentação (*feedback)*: mensagens de controle (*ACK*, *NAK*) do receptor para o transmissor.

Além disso, nossa máquina de estados também cresce, o transmissor, além de ficar enviado pacotes, também fica esperando mensagens de reconhecimento (negativo ou positivo) antes de receber novas mensagens da camada superior. O receptor continua o mesmo, só que envia um sinal para o transmissor dizendo o estado do pacote.

No entanto, e se as mensagens de controle chegarem corrompidas? O transmissor não saberá o que se passou no receptor e retransmitir o pacote é risco de duplicata. Então temos de lidar com duplicatas. Quando o transmissor recebe um sinal corrompido, ele transmite um duplicata, incluindo número de sequência em cada pacote e fica por responsabilidade do receptor não transmitir a duplicata para a aplicação.

Com isso, o Transmissor e o receptor receberão as seguintes características:

- Transmissor:
  - No de sequência no pacote (0,1) – indicando qual pacote eu quero (usamos sistema modulo 2 porque queremos o atual ou o próximo);
  - Verificação ACK/NAK por corrupção;
  - Número de estados duplicados, isto é, estado deve lembrar se pacote esperado deve ter número de sequência 0 ou 1;
- Receptor:
  - Deve verificar se o pacote recebido é uma duplicata (indicado pelo número de sequência).
  - Obs: Receptor não sabe se o último NAK/ACK foi recebido com sucesso.

Bem, sabendo disso, é possível implementar um protocolo sem NAK, apenas utilizando ACKs, o receptor envia ACK para último pacote recebido sem erro e o receptor deve incluir explicitamente número de sequência do pacote reconhecido. ACKs duplicados no transmissor resultam na mesma ação do NAK, ou seja, retransmissão do pacote atual.

Implementação com canais com perda de pacotes

Além de misturar bits, o canal também pode perder pacotes (contendo dados ou ACKs). Apesar de checksum, número de sequência, retransmissões ajudarem, não são suficientes para alguns casos, como pacotes que nunca chegam.

Para estes casos, teremos uma nova abordagem, um temporizador no transmissor que aguarda por um certo tempo para que chegue um ACK. Em caso de estouro de tempo, é retransmitido o pacote. Caso a retransmissão seja apenas duplicata, o número de sequência cuida disto.

Esta implementação funciona, no entanto, seu desempenho é sofrível, pois seu protocolo limita uso dos recursos físicas, pois ele espera para que alguma resposta chegue ou o temporizador estoure.

Por este problema, usa-se normalmente protocolos com paralelismo, onde o transmissor envia vários pacotes em sequência, todos esperando para serem reconhecidos, o que aumenta a faixa de números de sequência e, para poder funcionar, temos de armazenar estes pacotes no transmissor ou/e no receptor. Existem então dois métodos de paralelismo:

- Go-Back-N:
    - Transmissor pode ter até N pacotes não reconhecidos no canal;
    - Receptor envia apenas ACKs acumulativos (não reconhece pacote fora de sequência, mas reconhece todos os pacotes até e inclusive número de sequência n);
    - Transmissor possui um temporizador para o pacote mais antigo ainda não reconhecido (se o temporizador estourar, retransmite todos os pacotes ainda não reconhecidos).
    - Número de sequência de k-bits no cabeçalho do pacote;
    - Admite janela de até N pacotes consecutivos não reconhecidos no Transmissor;
    - Receptor envia ACKs para o pacote recebido corretamente, com o maior número de sequência, lembrando apenas o número de sequência esperado;
    - Pacotes fora de ordem não são recebidos, reconhecendo apenas o pacote com número de sequência mais alto;
    - Receptor ignora ACKs duplicados de modo a não reiniciar o temporizador;
    - Sua principal falha é descartar algo que chegou certo e não tem garantia de chegar certo posteriormente, mas evita reenvio de pacotes desnecessários em ACKs perdidos.
- Retransmissão seletiva:
    - Transmissor pode ter até N pacotes não reconhecidos no canal;
    - Receptor envia ACKs individuais para cada pacote;
    - Receptor armazena os pacotes chegados no buffer, conforme necessário, para posteriormente entregar em ordem a camada de aplicação;
    - Dado que próximo número de sequência disponível está na janela, envia pacote e liga temporizado de ACK. Transmissor possui um temporizador para cada pacote ainda não reconhecido, caso o temporizador estoure, retransmite apenas o pacote correspondente e reinicia temporizador.
    - Se pacote recebido pelo receptor estiver fora de sua janela, então apenas dá um ACK, mas não envia nem armazena, porque já fora enviado.

- Apesar de aparentemente muito efetivo, o método de retransmissão seletiva tem um dilema com relação ao número de janela e números de sequência. Caso o número de sequência não seja menor ou igual que a metade do número de janela, este poderá causar um erro de dados duplicados e passar esses dados como novos (explicado no slide longo). Logo, é necessário que o número de janela seja no máximo o dobro ou maior que o número de sequência.

Transporte Orientado a conexão: TCP

Antes de começar com as propriedades do TCP, vamos revisar suas características básicas para que fique mais fácil o aprendizado posteriormente:

- Ponto a Ponto: um transmissor e um receptor;
- Fluxo de bytes, ordenados e confiável: não estruturado em mensagens;
- Com paralelismo baseado em pipeline: tamanho da janela ajudado (dinamicamente) pelo controle de fluxo e congestionamento;
- Transmissão full duplex: fluxo de dados bi-direcional na mesma conexão e contém um tamanho máximo para o segmento (*Maximum Segment Size* - MSS);
- Orientada a conexão: Inicia o estado do transmissor e receptor antes da troca de dados, a partir do aperto de mãos;
- Fluxo controlado: receptor não será "afogado" pelo transmissor.

Estrutura do Segmento TCP

É importante saber como o Segmento TCP é estruturado para saber como a mensagem é enviado de forma segura: Vamos analisar seus campos que, além de além de incluir porta destino e fonte, incluem:

- Número de reconhecimento e de sequência: é o número que já sabemos e é feita a contagem por bytes de dados (não pela contagem de segmentos), ou seja, é o número daquele byte – exemplo seria: dado que envia o segmento 0 com 100 bytes, logo, o número de sequência do próximo segmento seria 200, não 1; Além disso são ACKs acumulativos, ou seja, se um dado byte foi recebido, dos os outros antes dele também, o que não indica nada sobre o seu método de paralelismo;
- URG: indica que há dados que a camada superior do lado transmissor considera urgentes. Além disso, para indicar onde é o final dos dados urgentes, foi criado um campo de ponteiro de dados urgentes.
- Comprimento do Cabeçalho: como o cabeçalho não tem tamanho fixo, visto que existe um campo Opções antes dos dados, logo é importante definir o comprimento do cabeçalho;
- ACK: indica se o campo de reconhecimento é valido ou não – só não é válido quando pedimos uma conexão – e indica que um segmento foi recebido com sucesso.
- PSH: indica que o dado deve ser entregue o mais rápido possível a camada superior e também pouco utilizado;

- RST, SYN, FIN: quando o SYN está setado, mas o ACK não, é sinal de um pedido de conexão. O contrário é uma resposta a pedido de conexão e apenas o ACK setado é troca de mensagens padrão. FIN encerra a conexão e RST é para reiniciar os dados;
- Janela de recepção: número de bytes receptor está pronto para aceitar
- CheckSum: funcionalidade a mesma do UDP.
- Opções: local onde as negociações de conexão são feitas entre o receptor e o transmissor.

É importante salientar que não é dito na especificação do TCP como é feito o tratamento de segmento fora de ordem (ou seja, quando um segmente é enviado antes de algum de seus anteriores), ficando a responsabilidade por conta do implementador.

Temporizador e RTT

Como escolher um valor de temporizador adequado para o RTT? Sabemos que tem de ser maior que o RTT, mas este varia, logo temos que escolher sabiamente, pois caso seja menor, irá haver temporizador prematuro e nem queremos que a perda de dados tenha resposta demorada.

Para estimar o RTT, é feito uma amostra de RTT baseado no tempo medido entre a transmissão do segmento e o recebimento do ACK correspondente, ignorando retransmissões, pois o RTT é mais curto. Esta estimativa é feita pelo *sampleRTT*, que tenta não ficar variando bruscamente o tempo todo, logo este contém um amortecedor que varia delicadamente com a variação do RTT, utilizando variações recentes juntamente com o valor do último *sampleRTT*. Este amortecedor é a média ponderada móvel exponencial.

Além disso é feito o cálculo de desvio para que se aceite uma margem de erro para o temporizador. Em caso de grandes variações, a margem de erro aumenta, caso contrário, esta diminui.

Transferências Confiáveis no TCP

Vamos agora entender como o TCP faz para realizar transferências de dados confiáveis. Para começo de conversa, o TCP cria um serviço RDT sobre o serviço não confiável de IP, os segmentos da camada de Transporte são transferidos em pipeline, com ACKs acumulativos, utilizando um único temporizador para a retransmissão. Além disso, retransmissões ocorrem, como já sabemos, pelo estouro do temporizador, mas também com o fato de ter ACKs duplicados (na verdade, um trio), pois normalmente o temporizador é bastante longo e, para evitar esperar o estouro, é feito um reenvio do segmento logo.

Para explicar mais simplificadamente o evento de transferência em TCP, iremos inicialmente ignorar os ACKs duplicados e o controle de fluxo e congestionamento, pois eles alteram a ideia geral. Quando formos falar deles, falaremos de como eles alteram a transferência.

Vamos analisar o que acontece no transmissor de dados TCP. Inicialmente os dados recebidos da camada de aplicação é separada em segmentos, cada um com

seu número de sequência – sendo o número, como já falamos, o primeiro byte de dados do segmento. Então é ligado o temporizador, caso ele não tenha sido inicializado, pois consideramos o temporizador do segmento mais antigo ainda não reconhecido. O tempo do temporizador nós já sabemos, é feita o cálculo que mostramos acima.

Toda vez que o temporizador estoura, o segmento relacionado com o temporizador é retransmitido e o temporizador reinicializado com o dobro do tempo anterior, sem fazer o cálculo do *sampleRTT*. Concomitantemente, quando segmentos ainda não reconhecidos recebem um ACK, é feita uma atualização da informação sobre o reconhecimento, religando o temporizador se ainda houver segmentos pendentes.

Já analisamos o transmissor, vamos analisar os eventos que ocorrem no receptor para a geração de ACKs. Existem quatro eventos no receptor e há uma contra ação para cada evento:

- Chegada de segmento em ordem sem lacunas e anteriores já reconhecidos: ACK retardado, esperando um tempo pelo próximo segmento para enviar logo um ACK acumulado. Caso contrário, envia logo o ACK do segmento atual.
- Chegada de segmento em ordem sem lacunas um ACK retardado pendente: envia imediatamente um único ACK cumulativo.
- Chegada de segmento fora de ordem, com número de sequência maior que o esperado (lacuna): envia ACK duplicado, indicando o número de sequência do próximo byte esperado.
- Chegada de segmento que preenche a lacuna parcial ou completamente: ACK imediato se segmento começa no início da lacuna.

Controle de Fluxo

Existe um buffer de mensagens recebidas no socket TCP em que o processo pode pegar os dados desejados de lá. No entanto, sua velocidade de pegar estes itens é inferior a velocidade do receptor TCP de entregar a este buffer. Deste modo é necessário criar um controle de fluxo, onde o receptor controla o transmissor de modo que não haja inundação (esgotamento de espaço), por causa do transmissor estar enviar muitos dados rapidamente.

Para realizar tal mecanismo, o receptor envia no cabeçalho dos segmentos que saem dele um valor *rwnd*, que indica o quanto o espaço livre atualmente há no buffer. O valor máximo do buffer é definido nas configurações ou automaticamente pelo sistema operacional.

Gerenciamento de Conexões

Bem, é fundamental que antes de enviar suas mensagens, o TCP estabeleça uma conexão entre transmissor e receptor. No entanto, o mundo não é perfeito e o estabelecimento de uma conexão TCP pode conter falhas numa rede como a Internet, como atrasos de variáveis, mensagens retransmitidas devido à perda da mensagem, reordenação das mensagens e, para piorar tudo, um lado não encher o outro.

Para que estas falhas não ocorram, é necessário um reconhecimento de interesse em estabelecer conexão em três etapas. Basicamente se faz uma proposta (onde não é feito nenhuma avalição se é interessante ou não fazer), o receptor envia

uma confirmação para saber se o transmissor ainda quer fazer um conexão e então o transmissor envia essa confirmação se interessado.

Mas, além disso, também há a desconexão, pois não é simplesmente desligar um lado e o outro aceita, pois um dos lado ainda pode ter algo a mandar. Só podemos terminar após recebimento e ambas as partes aceitem. Ou seja, no caso de desligamento, quem faz o pedido de fechamento espera pelos dados da outra parte terminarem. Após término, a outra parte indica que terminou e ambas encerram.

Princípios de Controle de congestionamento

Vale começar a dizer que este tipo de controle é diferente do controle de fluxo, enquanto o controle de fluxo é na própria máquina numa relação de receptor e transmissor, o controle de congestionamento é uma relação entre transmissor e toda a rede. Uma definição informal de congestionamento de rede é "muitas fontes enviando dados acima da capacidade da rede de trata-los". Se isto ocorre, sabemos que pode haver perda de dados no buffer dos roteadores lotados ou longas atrasos nas filas do buffer.

Vamos estudar vários cenários e entender suas causas e custos em cada situação:

- Cenário 1: Um roteador com buffers infinitos
    - Nesta situação temos que vários transmissores estão enviando seus dados para o buffer acima da capacidade de envio do buffer. Por conta disso, é criado uma fila muito longa (e provavelmente infinita), gerando um imenso atraso no envio do pacote.
- Cenário 2: Um roteado com buffers finitos
    - O excesso de tráfico gera uma perda do pacote em buffers finitos e que acabam por necessitar de uma retransmissão do pacote. Logo, é feito um novo envio, gerando mais tráfego na rede. Uma idealização deste problema é ter conhecimento do espaço livre do buffer, só enviando quando há espaço (não havendo perda). Outra idealização é ter conhecimento de duplicata, que é então descartada caso não seja necessária, mas ainda assim a banda do roteador é desperdiçada com o envio da duplicada descartável.

Bem, com isso podemos dizer que o custo de congestionamento nos dá muito mais trabalho, para que a vazão sempre a mais próxima do ideal.

- Cenário 3: Múltiplos roteadores com buffers finitos
    - Neste caso, pacotes podem passar por roteadores, mas, antes de conseguir chegar ao destino, o buffer foi ocupado por outro pacote de outra fonte e acaba por ser perdido. Isso nos leva a ver que apesar de tentarmos transmitir mais, acabamos por perder mais dados ainda, logo temos de buscar o ótimo no envio do pacote para poder não desperdiçar o trabalho dos outros roteadores.

Como o controle de congestionamento é um sistema bastante complexo, é feita duas abordagens gerais para o controle de congestionamento:

- Controle de congestionamento fim a fim
    - Não usa realimentação explícita da rede;

    - Congestionamento é inferido a partir das perdas e dos atrasos observados nos sistemas finais;
    - Utilizada pela TCP, onde ele vê que perdeu dados por conta de atraso e busca reduzir sua janela de envio de dados, buscando desobstruir a rede.
- Controle de congestionamento assistido pela camada de rede:
    - Roteadores enviam informações para os sistemas finais, com bit de sinal de congestionamento e taxa explícita para envio pelo transmissor.

Vamos analisar um caso de controle de congestionamento real, os serviços da ATM BR, que possui dois mecanismos para controle de congestionamento:

- ABR (*avaliable bit rate*): é um serviço elástico, onde o transmissor fica a usar da banda a depender da disponibilidade de envio da mesma. Em caso de banda congestionada, a taxa de envio torna-se mínima.
- Célula RM (*resource management*): é enviado entre os dados, e esta célula tem bits que é setada pelo comutadores da rede e cada bit indica o que o transmissor deve fazer.

Controle de Congestionamento em TCP

Diferentemente do exemplo dado no ATM, o controle de congestionamento do TCP é do tipo fim-a-fim, visto que a camada de rede utilizando o protocolo IP não fornece nenhuma retroalimentação com relação ao congestionamento da rede entre os sistemas finais.

Para entender como o mecanismo de controle de congestionamento funciona no TCP, primeiro iremos explicar como ele altera a taxa de envio de dados. Sabemos que existem alguns variáveis que são inicializadas e usadas durantes a conexão TCP, uma delas é a de *congestio window* (cwnd). Esta janela é responsável por minimizar a quantidade de dados a ser enviada na conexão. Se imaginarmos que uma certa quantidade é enviada por RTT, logo temos a relação cwnd/RTT bytes por segundo. Percebemos então claramente que a janela de congestionamento é capaz de regular a quantidade de bytes a ser enviada através da conexão diminuindo a janela em casa de detecção de congestionamento e aumentando caso perceba que esteja livre. Como o TCP utiliza os acks que recebe do receptor para aumentar ou diminuir a sua janela de envio, então ele é dito como *self-clocking*.

Agora, como próximo conhecimento, temos de saber como o TCP detecta o congestionamento da rede. Bem, como não a retroalimentação real da rede com relação a tráfego, o que o TCP faz é assumir verdade quanto ao envio de pacotes. Toda vez que houver estouro de temporizador ou uma trinca de acks duplicados, o TCP considera que houve perda de pacote por conta de congestionamento na rede e, com isso, "detecta" a congestão.

Algoritmo de Controle de Congestionamento do TCP

Assim como todo programa em computação, o controle de congestionamento do TCP também possui seu algoritmo, de modo a decidir em quais momentos e para

quais taxas a banda deve aumentar ou diminuir. É importante salientar que existem três fazes que ocorrem no TCP para o crescimento/encolhimento da taxa de envio:

- Devagarando (*slow start*): Normalmente, para início de envio, o TCP começa com um janela de 1 MMS (*maximum-size segment*) e aumenta o valor em mais 1 para cada ack recebido do receptor. Ou seja, imaginemos que começamos com 1 MMS, ao receber o seu ack, enviamos então 2 MMS, que retornam 2 acks, então enviamos 4 MMS, que retornam 4 acks, que nos permite enviar 8 MMS. E assim segue, dobrando o valor ao retificar que todos os acks foram recebidos, crescendo de maneira exponencial.

Mas, como nem tudo são flores, caso haja algum perda de pacote por estouro de temporizado, o TCP muda de fase e vai para:

- Evitando congestionamento (*congestion avoidance*): Ao estar neste estado, o TCP reinicia o valor do MMS para 1 e faz com que a variável de controle de congestionamento *ssthresh* (limiar de envio) caída pela metade, voltando então a janela a cresce exponencialmente.

Entretanto, também pode ocorrer do TCP estar na fase "devagarando" e enviar 3 acks duplicados. Neste caso também há indicação de congestionamento de rede, mas o TCP muda para outra fase:

- Rápida recuperação (*fast recovery*): Este estado não é implementado por todas as versões de TCP, mas no TCP Reno (usado na prova), ele é utilizado e o que acontece é que, ao invés de resetar o MMS para 1, o MMS é setado para metade do valor que o TCP conseguiu chegar logo antes de ocorrer a falha de envio mais 3 MMS, setando a variável de limiar para o mesmo valor. Então, a partir daí, este cresce linearmente em 1 unidade.

Por esta razão de adicionar linearmente e multiplicar de forma regressiva que o controle de congestionamento do TCP é também chamado por *additive-increase,multiplicative decreace* (AIMD).

Introdução a Camada de Rede

Como já vimos no início do curso, a camada de Rede é responsável pela comunicação host-to-host numa relação de comunicação entre os sistemas finais. Para começar, é importante apresentarmos a distinção entre roteamento e repasse. Enquanto que repasse envolve a passagem dos dados do enlace que recebeu para o enlace destino dentro do mesmo roteador baseado na tabela de roteamento, roteamento é o conjunto de roteadores que interagem entre si através de protocolos de roteamento para determinar os caminhos que o pacote deve seguir dentro da rede.

Vale realçar também que a camada de rede é a responsável por encapsular o segmento da camada de transporte e adicionar o cabeçalho, transformando-o em um datagrama, ou seja, em um pacote da camada de rede.

Modelo de Serviço da Camada de Rede

Assim como a camada de Transporte e da de aplicação oferecem uma gama de serviços, a camada de rede não é diferente. Vamos analisar quais serviços ela é capaz de oferecer:

- Garantia de Entrega com ou sem atraso limitado: Mais cedo ou mais tarde o datagrama irá chega ao seu destino e podemos decidir o atraso máximo que um pacote poderá sofrer dentro da rede.
- Entrega de pacotes na ordem: Os pacotes chegaram ao seu destino na ordem que foram enviados.
- Largura de banda mínima garantida: Somos capaz de definir uma largura mínima de banda para a troca de pacotes entre remetente e destinatário através de uma simulação, mesmo no caminho real haja vários enlaces físicos.
- Jitter máximo garantido: Somos capaz de definir uma limite para o espaço entre a diferença do envio de dois pacotes sucessivos do remetente e o recebimento de dois pacotes sucessivos no destinatário.
- Segurança garantida: É dado um chave aos dois hospedeiros que estão se comunicando para garantir que somente eles são capazes de decodificar as mensagens que estão trocando, além de fornecer integridade de dados e autenticação de fonte.

Enquanto que os dois primeiros itens são especificamente para um único pacote, os outros três são serviços que a camada de rede pode garantir para um fluxo de pacotes. Mas vale avisar que, se considerarmos a Internet, seu modelo de serviço é o de melhor esforço, no caso, ela não garante nada que a camada de rede é capaz de garantir.

Redes de Circuitos Virtuais e Datagramas

Assim como na camada de transporte, a camada de rede também pode oferecer seus serviços orientado ou não a conexão. Apesar de serem bastante semelhantes, vale dizer que enquanto que a camada de transporte realiza a orientação em nível de processo a processo para a camada de aplicação, a camada de rede o faz em nível de hospedeiro a hospedeiro para a camada de transporte.

Além do nível de orientação, podemos dizer que há uma peculiaridade quanto a nome dado as rede de computadores que utilizam orientação, chamadas de redes de circuitos (RC) e as que não são orientadas a conexão, chamada de rede de datagramas. E claro, uma rede deve ser um ou outro, mas não ambos.

Começando nosso estudo mais aprofundado com relação a orientação de conexão, vamos estudar um pouco de circuito virtual. E, assim como na camada de transporte, a camada de rede que é orientada a conexão necessita de um comprimento entre o destinatário e o remetente para configurar todos os roteadores e os próprios sistemas finais para o envio dos dados. Daqui já podemos diferenciar a orientação a conexão na camada de transporte para a camada de rede, enquanto que na camada de transporte os roteadores não enxergam a conexão entre os sistemas finais, na camada de rede, tanto os sistemas finais quanto os roteadores tem noção dos outros dispositivos conectados a eles.

Observando a estrutura de um CV, podemos dizer que este constitui de três principais componentes, o caminho que os pacotes percorreram, os número de CV dos enlaces e o tabela de repasse nos roteadores que participam deste CV. Com isso, existem três fases que um circuito virtual possui:

- Estabelecimento de conexão: é nesta etapa que a rede configura os roteadores e os número de CV dos enlaces para a recepção dos pacotes, além de também poder definir um largura mínima de banda.
- Transferência de dados: é quando definitivamente ocorre a troca de dados entre o remetente e o destinatário.
- Enceramento da CV: assim como na camada de transporte, a conexão entre os sistemas finais é encerrada e as tabelas de repasse são atualizadas para indicar que aquele caminho não existe mais.

Diferentemente do circuito virtual, a o envio de pacotes em uma rede de datagramas é feito sem o estabelecimento da conexão, é apenas anexado no cabeçalho do pacote o endereço de IP do destinatário e é através deste endereço que ocorre o roteamento nos roteadores. Para isto é utilizada a ideia de prefixo, onde uma parte do endereço servirá para indicar a qual enlace de saída deverá ser encaminhado aquele pacote e, em caso de prefixos com encaminhamentos diferentes, então é utilizado o maior prefixo.

Com isso, podemos perceber que por não manter os dados da conexão – mas mantém os dados da tabela de repasse que são atualizados quando necessários – os pacotes enviados pela rede de datagramas não possui garantia de chegada em ordem, visto que dois pacotes consecutivos podem ou não ser enviados pelo mesmo caminho e isto é capaz de acarretar em uma reordenação dos pacotes.

Por dentro de um roteador

Agora que demos uma olhada de falcão no mecanismo da camada de rede, vamos aprofundar mais um pouco e entender como funciona o mecanismo de repasse dentro de um roteador de modo a levar os dados do enlace de entrada para o enlace de saída apropriado. Comecemos então pela arquitetura de um roteador, que consiste de três componentes:

- Porta de entrada: Realiza tanto funções das camadas física e de enlace quanto é responsável por receber o datagrama e repassá-lo ao Elemento de comutação de modo que saída na porta de saída apropriada.

- Elemento de comutação: responsável pela conexão entre as portas de entrada e as portas de saída.
- Porta de Saída: realiza o processo inverso das funções da porta de entrada, recebendo os pacotes do elemento de comutação e os colocando num buffer para ser enviado para o enlace de saída.
- Processador de roteamento: responsável por rodar os protocolos de roteamento, como a tabela de repasse e roteamento e funções de gerenciamento de rede.

Vamos estudar principalmente os três primeiros elementos, começando pela porta de entrada. Como já foi dito, ela é a responsável por receber o pacote e repassar o mesmo para a porta de saída apropriada. Para isto, a porte de entrada necessita ter uma cópia da tabela de repasse do processador de roteamento – que envia cópias atualizadas sempre que necessário – e as consulta para poder saber para que porta enviar. Este mecanismo permite um repasse descentralizado e evita a criação de um gargalo de processamento de repasse dentro de um único ponto centralizado no roteador. Mas claro que, em processadores em que o gargalo se encontra na porta de entrada, é necessário fazer um processamento centralizado para evitar tal situação.

Para alcançar tal meta, é necessário uma super velocidade de processamento na hora de consulta na tabela de repasses, ou seja, uma velocidade a nível de linha, baseado no fato de que o consulta deve demorar no máximo tanto quanto a velocidade de chegar outro pacote para consultar. Para conseguir isto, é feito uma busca binária que pode ser realizada em N, sendo N o número de bits do endereço do destinatário.

Dado que a tabela de repasse já foi consultada, está na hora de passar o pacote para o elemento de comutação. No entanto, caso este já esteja ocupado com outro pacote, o pacote que iria entrar deve esperar numa fila para a chegada de sua vez. Dado que é a vez do pacote ser examinado pelo elemento de comutação, vamos ver o que acontece. Já sabemos que ele é responsável por repassar o pacote da porta de entrada para a porta de saída apropriada, no entanto ele pode fazer isto através de três mecanismos diferentes:

- Comutação por memória: É um dos mais antigos mecanismos para comutação de pacotes e se baseia na memória do processador e de um sistemas de entrada e saída tradicional. Quando um pacote chega na porta de entrada, este interrompe o processador e o pacote é copiado na memória do processador. Já na memória o pacote pode ser analisado para saber para qual porta de saída ele deve ser enviado, sendo então o pacote copiado para o buffer da porta de saída.
- Comutação por barramento: Em vez de um processador para operar sobre os pacotes que chegam, temos um barramento que distribui o pacote para a sua devida porta de saída. No entanto, apenas um pacote por vez pode entrar no barramento, de modo que a velocidade do barramento é um gargalo para a comutação – atualmente é capaz de ser suficientemente poderoso.
- Comutação por interconexão: Para não se limitar a largura de banda do barramento, a comutação de interconexão oferece múltiplos barramentos que são capaz de fazer seus processamentos independentes um do outro e então acelerar o processo de comutação.

Agora que já sabemos como o pacote chega na porta de saída, vamos estudar o que há nessa porta. Já sabemos que ela é a responsável por passar os pacotes para a camada inferior. No entanto, existem momento em que a chegada de pacotes na porta é muito mais que o processamento do enlace, e logo se forma uma fila. Com isso, vamos estudar principalmente o gerenciamento de fila desta porta.

As filas podem ser formar tanto na saída para o comutador de elementos quanto na entrada das portas de saída. Dado que temos n portas de entrada, contanto que nosso processador consiga operar n*pacotes por tempo, então não se formará fila na unidade de entrada. No entanto, uma única unidade de saída pode receber estes pacotes e, se e velocidade do enlace não for mais rápida que a chegada de novos pacotes, logo se formará uma fila e poderá ocorrer perda de dados. Para isto, se fazem vários estudos sobre o tamanho da fila adequado para a transmissão e, o cálculo atualmente feito é: RTT.C/raiz(N) Bits de espaço.

Apesar de ser uma fila, o escalonador de pacotes de fila não necessariamente precisa pegar o primeiro da fila, o pacote escolhido fica a depender do algoritmo de escalonamento, que tem um papel fundamento na garantia de qualidade de serviço da camada de rede. E, assim como existe o escalonador de pacote, também existe um mecanismo para decidir qual pacote deverá ser descartado além de outras funções.

Mas, como dito anteriormente, caso o elemento de comutação não seja rápido o suficiente para processar a chegada dos pacotes, começará a se formar uma fila na porta de entrada do roteador. E existe um problema bastante grave na formação destas filas, chamados de HOL *blocking* (*head-of-line*), que é quando um pacote atrás do primeiro da fila está espera que os que estão na sua frente sejam enviados por uma porta atualmente ocupada, apesar de a sua está livre.

Protocolo da Internet (IP)

Iremos estudar o protocolo mais utilizada na Internet, o IP. Analisemos primeiro o formato do datagrama IP, lembrando que o cabeçalho do datagrama fica acima do segmento da camada de transporte.

Formato do Datagrama

Vamos analisar cada campo do datagrama:

- Versão: indica qual versão do protocolo IP foi utilizada;
- Comprimento do cabeçalho: comprimento do cabeçalho (em bytes);
- Tipo de serviço: tipos dos dados. Ótimo para dar prioridade na rede local;
- Comprimento: comprimento total do datagrama (em bytes);
- Identificação, bits, início de fragmentação: são campos utilizadas para o processo de fragmentação e remontagem;
- Sobrevida: limita o tempo de vida dos datagramas na rede, para não ficarem para sempre na rede;
- Protocolo: indica qual protocolo da camada superior ele deve entregar;
- Checksum Internet: o mesmo do checkSum dos outros protocolos;
- Endereço de Ip de origem e de Destino: cada é um endereço de 32 bits, indicando o destino e a origem;
- Opções: algumas coisas extras, como registrar rota, especificar os roteadores passados, entre outros;

Ao total temos 20 bytes de cabeçalho no datagrama IP, que se soma aos 20 bytes do TCP (se for utilizado TCP – que normalmente é) e mais os bytes da camada de aplicação.

Fragmentação e Remontagem

Alguns datagramas podem ser muito grandes, em tamanho, e, como cada enlace de rede contém um tamanho máximo de transmissão (MTU) que varia de acordo com o enlace, é necessário fragmentar o datagrama em datagramas menores, chamados de segmentos. Para isto, foi repassado ao sistema final a tarefa de resgatar o segmentos e uni-los em um único datagrama antes de entregar a camada de transporte.

Para realizar tal operação é necessário formatar os segmento da seguinte forma:

- Identificação: um ID para identificar a qual datagrama este segmento pertencia originalmente.
- Flag: indica que é o último segmento do conjunto ou se há continuidade.
- Deslocamento: indica a posição do primeiro byte daquele segmento no datagrama original de forma a poder remontar o datagrama original.

Endereço IPv4

O endereço IP não identifica a máquina, mas sim a interface de estação (subrede) e o roteador. Podemos definir a interface como a conexão entre a estação ou o roteador e o enlace físico. Um roteador típico contém várias interfaces, enquanto que uma estação possui uma ou duas interfaces (como Ethernet e WiFi). Podemos definir que uma interface é a fronteira entre o hospedeiro (seja um sistema final ou um roteador) e o enlace físico e o endereço IP está intimamente ligado com a interface.

O endereço de IP com 4 bytes (ou 32 bits) fica então separado da seguinte forma: os bits de mais alta ordem para a parte de subrede e os outros menos significativos para a parte da estação. Com isso, podemos definir uma subrede como interfaces de dispositivos com a mesma parte de subrede no seus endereços de IP, podendo alcançar um ao outros sem passar pelo roteador intermediário.

Existem vários tipos de endereçamento de IP, iremos estudar alguns deles, mas existe um tipo que é baseado em classe, que dar uma certo tamanho fixo de parte de rede para a instituição. O que iremos estudar agora é o *Classless InterDomain Routing* (CIDR). Algumas de suas características são:

- Parte de rede de endereço de comprimento arbitrário.
- Formato do tipo a.b.c.d/x, onde x é um número de bits na parte de subrede do endereço.

Desse modo podemos distribuir blocos de endereços para ISP's de modo que estas possam ser capaz de distribuir os endereços de IP's ou mais blocos para outras organizações.

Agora que temos um bloco com endereços de IP disponíveis, temos de saber como fazer para obter o endereço de IP. Existem duas formas, podemos fazer isto manualmente, o que fica bastante enfadonho, pois o gerente de rede tem de configurar todo o sistema de rede para cada um dos sistemas finais e dos roteadores intermediários, ou através do DHCP (*Dynamic Host Configuration Protocol*), que oferece um endereço de IP ao host dinamicamente através de um servidor. A

vantagem do DHCP é sua capacidade de renovação de "empréstimo" do endereço, além de reutilização do mesmo quando está livre, além do fato de ser *plug and play*, ou seja, o hospedeiro pode utilizar a internet assim que se conectar, visto que é automaticamente. No entanto, para conseguir obter o endereço, é necessário passar por quatro etapas:

Vale salientar que o DHCP não indica apenas o endereço IP, ele também fornece o endereço do roteador que o host deve enviar seus pacotes para outros hosts fora da sua subrede. Fornece também o nome e endereço IP do servidor DNS e a máscara de rede (que identificam a rede e o hospedeiro).

O endereçamento do IP possui um sistema hierárquico, cujo o roteador anuncia a Internet (os outros roteadores) o prefixo do endereço que ele recebe, de modo a facilitar a rota, agregando uma rota para aquele endereço, mecanismo chamado de agregação de rede.

Um outro mecanismo da rede são os endereços NAT, este visa usar um único endereço de IP para uma rede local para conectar-se ao mundo exterior. Isto traz como benefícios a não necessidade de alocar faixas de endereços do ISP – um único endereço de IP é utilizado para todos os dispositivos. Concomitantemente, é possível uma melhor manutenção e escalabilidade da rede local, visto que é possível modificar o endereço dos dispositivos sem notificar o mundo exterior, além de também pode trocar o ISP, sem ter de trocar os endereços dos dispositivos locais. Como última vantagem, existe a segurança da rede local, pois a rede exterior não tem noção do que existe do lado de dentro.

Mas vale salientar que os dispositivos locais não são visíveis ao mundo e, por isto é necessário que o roteador tipo NAT implemente algumas funcionalidades:

- Datagramas que saem: é necessário alterar o par (IP origem, # porta) por (IP NAT, # porta dada pelo NAT), de movo que clientes/servidores remotos vão responder com destino sendo o novo par.
- Tabela de tradução NAT: o roteador NAT deve ser capaz de fazer a tradução de (IP origem, # porta origem) ←→ (IP NAT, # porta dada pelo NAT).
- Datagramas que chegam: é necessário alterar o par (IP NAT, # porta dada pelo NAT) por (IP origem, # porta) correspondente na tabela NAT.

A tradução do NAT é realizada com um campo de número de porta com 16-bits, o que permite no máximo 60.000 conexões simultâneas (o que é bem improvável de acontecer). Além disso, ele também é controverso, pois ele deveria acessar somente até a camada de rede, mas acaba por acessar outras. Também viola o argumento fim-a-fim, pois em aplicações P2P é necessário que o projetista leve em conta os endereços escondidos. Como outra desvantagem é que a escassez de endereço deveria ser feita pelo IPv6, mas tem sendo feita com NAT, que é na verdade um mecanismo de proteção que viola regras de transparência, solucionando um problemas através de métodos não-transparentes.

Para resolver o problema da invisibilidade do NAT, existem alguns soluções, algumas fáceis, outra são mecanismos criados por empresas. Vamos analisar algumas:

- Solução1: Podemos configurar estaticamente o NAT para encaminhar para o cliente/servidor os pacotes diretamente, configurando uma porta específica para aquela máquina.

- Solução2 Protocolo IGD (*Internet Gateway Device*): automatiza a configuração do mapeamento estático de portas através de um mapeamento de portas com tempo de validade.
- Solução3: repasse (utilizado pelo Skype): clientes atrás do NAT se conectam ao relay, assim como um cliente externo. Deste modo o relay serve como intermediário entre pacotes de uma conexão para a outra.

ICMP (*Internet Control Message Protocol*)

É um protocolo utilizado por estações e roteadores para comunicar informação sobre a camada de rede, relatando erros e pedidos e respostas de eco. Fica localizado acima do IP, sendo transportadas nos datagramas IP. Sendo o formato bem pequeno, com tipo, código e os primeiros 8 bytes do datagrama IP causando erro.

O traceroute é um dos mecanismos que utilizam este protocolo. Como, toda vez que um pacote tem seu tempo de vida expirado, é enviado uma mensagem ICMP que inclui nome e endereço de IP do roteador, o traceoute então descarta a mensagem quando a mensagem chega em um roteador. Então a origem calcula o RTT (isto é feito três vezes pelo traceroute). Ele para quando, ou chega no seu destino final ou quando recebe um erro de porta inalcançável.

IPv6

Atualmente temos um grande problema na rede, o espaço de endereçamento de 32 bits em breve será esgotado. Para isto, foi desenvolvido uma nova versão do IP, o IPv6 que além de retirar o problema de esgotamento de espaço, visa também facilitar o processamento e repasse dos datagramas com um formato mais acessível, e também melhorar a qualidade de serviço (QoS). No entanto, temos um aumento do cabeçalho, com tamanho fixo, com 40 bytes - devido ao aumento do número de IP – e não suporte a fragmentação, sendo necessário passar este problema para outra camada. Algumas das alterações mais evidentes foram:

- Capacidade de endereçamento expandida: o tamanho do endereço de IP foi para 128 bits.
- Cabeçalho de tamanho fixo: o cabeçalho tem o tamanho fixo de 40 bytes e não contém a parte de opções, o que agiliza o processamento do datagrama IP.
- Novos campos introduzidos para novas oportunidades, como a ideia de fluxo, capaz de introduzir prioridades dentro de uma rede particular para agilizar transferências.

O cabeçalho IPv6 contém os seguintes novos campos:

- Classe de tráfego: identifica prioridade entre datagramas do fluxo (no entanto não se tem definido como é feito, apenas contém o campo para que outros sistemas implementem);
- Rótulo do fluxo: identifica datagrama no mesmo fluxo (sim, o conceito de fluxo não está claro para o que deve ser utilizado, apenas foi criado para gerar possibilidades);
- Próximo cabeçalho: identifica protocolo da camada superior;

Ainda são incluídos outras campos como o comprimento de carga útil, limite de saltos, endereços de fonte e destino e os dados que são semelhantes aos do IPv4.

Algumas mudanças com relação ao IPv4 forma:

- Checksum: completamente removido para reduzir tempo de processamento a cada roteador;
- Opções: permitidas, porém fora do cabeçalho, indicadas pelo campo "Próximo cabeçalho";
- ICMPv6: uma nova versão do ICMP que permite tipo de mensagens adicionais (como "Pacote muito grande") e funções de gerenciamento de grupo multiponto.

Mas, claro, como é de se imaginar, não é da noite para o dia que irá ocorrer a mudança de IPv4 para IPv6, então fica a pergunta, como a rede funciona com essa mistura de roteadores? Para isto ocorrer, ela utiliza o mecanismo de tunelamento: datagramas IPv6 carregas em datagramas IPv4 entre roteadores IPv4 (como se fosse um encapsulamento), de modo que os datagramas IPv6 se tornam parte dos dados úteis até que se chegue em um sistema que é IPv6.

Camada de Enlace

Vamos aprender um pouco como a camada de enlace funcione, buscando desenvolver um conhecimento sobre da detecção de erros e correção dos mesmos, compartilhamento de canais via *broadcast*, endereçamento da camada de enlace e as redes locais. Para começar, vamos definir primeiro que todos os sistemas finais e os roteadores intermediários serão tratados como nós, visto que não difere o comportamento se um nó é um sistema final ou um roteador. Além disso, como iremos estuda a camada de enlace, temos que entender que enlace é uma ligação que existe entre dois nós. Como última terminologia iremos dizer que os pacotes trocados entre os nós são chamados de quadros, que encapsulam os datagramas que recebem da camada de rede.

Serviços da camada de enlace

O principal objetivo da camada de enlace é transportar os datagramas da camada de rede entre dois nós através de um único enlace. E para isso, utiliza diversos protocolos que estudaremos mais tarde. Para início vamos ver quais serviços a camada de enlace fornece, mas vale dizer que alguns serviços são oferecidos por alguns protocolos e outros não e, além disso, o mesmo enlace pode utilizar um protocolo para um certo datagrama e outro para outro, por isso a camada de rede tem de saber lidar com essa heterogeneidade da camada de enlace. Os serviços possíveis de serem oferecidos são:

- Enquadramento de dados: os datagramas são encapsulados dentro de um quadro que contém um cabeçalho para a troca de mensagens.
- Acesso ao enlace: através dos protocolos MAC (*Medium acess protocol*), são definidas as regras para um bom e justo gerenciamento de acesos ao enlace em situações de *multicast* quando um único enlace é utilizado por mais de dois nós. Em caso de *unicast*, um protocolo de MAC simples é utilizado para a troca de quadros.
- Entrega confiável: Assim como na camada de transporte, a camada de enlace pode realizar entregas confiáveis para evitar que haja necessidade de refazê-la em uma ligação fim-a-fim. Mas, pode ser um processo que

demanda mais processamento, é mais utilizado em tecnologias que apresentam maior taxa de erros, como as sem fio, diferentemente das redes a cabo, que normalmente não apresentam esse protocolo confiável devido a não necessidade.

- Controle de fluxo: Como os quadro são armazenados quando chegam a um nó antes de serem processados, é importante que o remente não inunde o buffer do destinatário e cause perda de dados desnecessária.
- Detecção de erros: Assim como nas camada de transporte e de rede, a camada de enlace pode realizar a detecção de erros, exceto que de um modo mais aprimorado, fazendo-a através de componentes de hardware.
- Correção de erros: Além de detectar, um protocolo pode também ser capaz de corrigir um erro ao percebê-lo, o que agiliza a troca de mensagens entre ambos.
- *Half-duplex* e *full-duplex*: são dois mecanismos exclusivos onde os nós não são capazes de transmitir e enviar dados ao mesmo tempo ou são capazes de fazê-lo, respectivamente.

Mas onde é implementada a camada de enlace? Ela é implementada parte em software e parte em hardware em um adaptador de rede, conhecidos como controla de interface de rede. Dentro do adaptador de rede existe o controlador que implementa a maior das funcionalidades da camada de rede via hardware, o que agiliza o processamento. Do lado do transmissor da interface, é de sua responsabilidade encapsular os datagramas que chegam em quadros além de adicionar ao cabeçalho os bits de controle. Já do lado do receptor, é necessário a implementação das mesmas funcionalidades como detecção de erros, controle de fluxo, entre outros, além de também extrair o datagrama para enviar para a camada de rede.

Técnicas de detecção e correção de erros

Como já sabemos, a detecção de erros não é uma técnica com 100% de chance de ser um sucesso, visto que podem haver erros que passam despercebidos por alguns mecanismos. Faremos uma análise dos três mecanismos mais utilizados pela camada de enlace para detecção e correção.

- Verificação de paridade: com essa técnica é adicionada um bit de paridade para tornar a quantidade de bits 1 presentes nos dados um número par. Apesar de rápida, é um técnica que pode falhar muitos, principalmente porque os erros normalmente são dependentes e acar reta um erro em cascata, o que não é ideal para a verificação de paridade. Também há a verificação em matriz, que além de conseguir detectar o erro, pode corrigi-lo pode saber onde ocorre.
- Soma de verificação: Bastante utilizado pela Internet por se um processamento rápido. No entanto tem um desempenho médio se comparado a outras tecnologias. Segue a mesma ideia do *checksum* já estudado anteriormente.
- Verificação de redundância cíclica (CRC): Mais poderosa que as outras e a mais utilizada na camada de enlace, mas também mais complexa. Através de uma cadeia de bits gerador G, e a concatenação dos bits dos dados com um cadeia de bits R. Se aplica a divisão da concatenação pela cadeira G. Se o resto for zero, então é concluído que não apresenta erro.

Protocolos de acesso múltiplos

Diferentemente dos protocolos *unicast*, os de múltiplos acessos podem causar sérios problemas na transferência de dados, já que se dois ou mais quadro chegarem ao mesmo tempo em um nó, uma colisão de dados ocorrerá e não terá como os dados serem processados. Para gerenciar a transmissão de dados em uma conexão *multicast*, é necessário utilizar protocolos de acesso múltiplos e é o que iremos estudar nesta seção.

Mas antes de começarmos, iremos definir quais características um protocolos de *multicast* deve ter para termos um processamento ótimo dos dados:

- Na existência de um único nó enviando os dados, este nó deve conseguir enviar na velocidade máxima da banda.
- Em caso de M nós, a velocidade média dos nós deve ser a razão de M pela velocidade máxima da banda, de modo a garantir que, em média, todos os nós estão enviando na mesma velocidade.
- Não há nós mestres de modo que não há como um nó falhar e isso prejudicar toda a conexão.
- Protocolo simples de fácil e barata

Começando a analisar os protocolos, temos os protocolos de divisão de canal. Já estudados no início do curso, temos dois métodos para a divisão de canal, a divisão por tempo, TDM e a divisão por frequência, FDM. Dado uma rápida revisa em ambos, na divisão por tempo, cada nó ganha um breve momento no quadro temporal da TDM para enviar seu pacote e deve esperar novamente pela sua vez para enviar outro. Apesar de absolutamente justo e sem ocorrência de colisão, o TDM limita a velocidade de um único nó para 1/N – N número de nós – visto que este deve sempre esperar pela sua vez, sendo uma péssimo escolha para ambientes onde nós ficam muito tempo sem enviar. O FDM tem as mesma vantagens e desvantagens do TDM, exceto que é feita uma divisão das frequências em faixas.No entanto, existe ainda um terceiro protocolo chamado CDMA (*code division multiple access*) que iremos estudar quando formos ver as redes sem fio.

Uma outra classe de protocolos de acesso múltiplo são os protocolos de acesso aleatório em que todos os nós enviam a taxa máximo da banda e, em caso de colisão, os nós envolvidos esperam um tempo aleatório de espera independentes, de modo a garantir que um deles enviará primeiro, não tendo colisão. Um desses protocolos é o slotted ALOHA, onde um nó sempre transmite no início de um intervalo – tempo de transmissão de um quadro – e, caso haja colisão, o nó faz a retransmissão baseado em um probabilidade p onde ele deve decidir que irá retransmitir naquele intervalo ou em outro próximo. Apesar de ser excelente em caso de um único ou poucos nós ativos que vão poder enviar a grandes velocidades, e casos de muitos nós, a quantidade de espaços vazios – devido ao jogo aleatório – e de colisão é maior que a quantidade de envio bem-sucedido, que só ocorrerá quando um único nó enviar o seu quadro naquele intervalo. Antes existia o ALOHA puro, em que o a transmissão do quadro ocorria imediatamente, pois não havia sincronização dos nós e com isso a eficiência era menor.

Um protocolo dessa mesma classe e com melhor desempenho foi desenvolvido e utiliza dois técnicas fundamentais, cada nó sempre busca escutar o canal antes de enviar para saber se está livre – detecção de portadora – e também busca saber se há outro nó transmitindo ao mesmo instante que ele para impedir que colisões ocorram. Este novo protocolo foi o CSMA, e o CSMA/CD (*collision detection*) – *carrier*

sense multiple access. Apesar de utiliza a técnica de escutar antes de transmitir, ainda assim pode ocorrer colisão visto que existe um atraso de propagação fim a fim de canal, então, se um nó X transmitiu seu quadro, mas ainda não chegou ao nó Y e este achou que o canal estava vazio por ainda não ter chegado, Y enviará e causará colisão. Podemos ver claramente que o atraso de propagação é um fator de desempenho muito importante, pois quanto maior, maior é a probabilidade de ocorrer colisão por atraso. Com o CSMA/CD, quando uma colisão é detectada, o nó deve parar sua transmissão para não enviar um pacote que sofreu corrupção de dados e desperdiçar tempo de transmissão.

Ainda existe um terceira classe de protocolos de múltiplos acessos, este são os protocolos de revezamento. Um deles é o protocolo de polling, onde se define um nó mestre que deve escalonar o nó que irá enviar naquele instante, com isso ele tanto obtém velocidade máximo em casos de um único nó, quanto uma velocidade média de 1/N em caso de N nós – um fator que as outras classes de protocolos não possui. Apesar da maior eficiência, os protocolos de polling perdem um pouco de sua grande eficiência visto que gastam tempo para escolher um nó específico antes dele poder transmitir. Além do mais, é um protocolo centralizado, então se o nó mestre falhar, todo o canal falha.

Para suprir a necessidade de um nó mestre foi desenvolvido um protocolo de passagem de permissão que utiliza a existência de um quadro especial, o *token*, que é passada de nó em nó para que o retentor ganhe permissão de transmitir. Apesar disso, se um nó falhar, ele pode levar consigo o *token*, o que quebraria o canal. Logo é necessário desenvolver um mecanismo de recuperação de *token*.

Endereçamento e ARP

Assim como na camada de rede, cada sistema final e roteador tem seu endereço IP – mas que varia a depender da sub-rede em que a máquina se encontra – também existe um endereço na camada de enlace, mas que identifica a interface de rede e é um indicador fixo de 6 bytes em representação hexadecimal – pode ser mudado por software, mas não varia de acordo com a sub-rede. Este endereço é chamado por LAN, endereço físico ou MAC (*media access control*).

Como agora temos dois tipos de endereçamentos, os de protocolo da Internet – IP – e os endereçamentos MAC, é necessário fazer a tradução do endereço de Internet para o endereço físico da máquina e isto é uma responsabilidade do ARP (*address resolution protocol*). É como o ARP foi um DNS, exceto que em vez de funcionar para a Internet como um todo, o ARP funciona apenas nas redes locais, de modo a possibilitar que outros protocolos de rede sejam utilizados em rede privadas no local do protocolo IP.

Para realizar tal tarefa, o nó fonte deve primeiro procurar em sua tabela ARP se ele já sabe a conversão do IP destino para o endereço MAC destino. Caso saiba, ele poderá prosseguir com o envio, caso não, ele enviará um pacote especial chamado de ARP *Query* para o roteador da sub-rede e este enviará uma mensagem de *broadcast*. Quando os nós recebem a mensagem, verificam se ele se encaixa no IP, caso afirmativo – e único – o dito cujo envia uma mensagem de retorno com a tradução de IP para MAC e então o nó que inicial tudo poderá atualizar sua tabela.

Apesar de termos explicado como funciona numa rede loca, o funcionamento não se altera muito quando é necessário enviar um pacote para um computador em

uma rede externa. É necessário que o nó remetente coloque no campo MAC destino o endereço MAC do roteador que irá fazer o repasse para o enlace de saída apropriado e envia com o endereço MAC do próximo nó. Isto ocorre até que o quadro chegue ao nó destino.

Rede Ethernet

Assim como a Internet é uma tecnologia profundamente utilizada no sistema global, a Ethernet é a maior tecnologia no sistema de LAN. Ela utiliza dois possíveis tecnologias, a de barramento – com chance de colisão – e as de estrela com um comutador no centro – muito utilizado atualmente e nós não colide um com o outro visto que existe uma saída para cada nó. Além disso, o sistema da Ethernet é um sistema *broadcast*, pois todos os nós recebem uma cópia dos dados enviados dentro da LAN.

Vamos agora estudar a estrutura do quadro de Ethernet para entender melhor seu funcionamento:

- Campo de dados: são bytes de dados de mínimo de 46 bytes – completados com recheio se for necessário – e máximo de 1500 bytes – e caso o tamanho passe, os dados podem ser fragmentados, como falamos anteriormente na camada de rede.
- Endereço de destino: endereço MAC do destino. Caso o endereço MAC não bata com o endereço do nó, o quadro é descartado.
- Endereço fonte: endereço MAC da fonte.
- Campo de tipo: indica qual o protocolo da camada de rede este datagrama deve ser repassado – lembrando que existem vários protocolos de rede, não somente o IP.
- Verificação de redundância cíclica: como já estudado, verifica se os dados estão corrompidos.
- Preâmbulo: Sincronização do relógio para transmissão na taxa adequada, além de bytes para indicar a urgência.

Outras características da rede Ethernet são sua não orientação a conexão, tão logo um nó deseja enviar um quadro, este o faz sem se preocupar se há uma conexão com o outro nó, e também não é um serviço confiável, quando um quadro é lançado, o destinatário não faz nenhum reconhecimento do seu recebimento, logo qualquer falha no envio será corrigida pela camada superior – ou não.

Além disso, a tecnologia Ethernet usufrui do mecanismo de *broadcast* CSMA/CD, com uma análise através dos picos de tensão antes e durante a transmissão.

Comutadores Ethernet

Como atualmente é utilizado bastante a topologia de forma estrela, é importante que estudemos o elemento central desta, o comutador, um elemento transparente para os nós da rede e com uma função muito importante, repassar os pacotes dentro da rede, com um buffer de armazenamento semelhante aos roteadores da camada de rede.

Para que o comutador opera com um bom desempenho, ele implementa duas grandes funções: filtragem – determina se o quadro deve ser repassado para algum interface ou se deve ser descartado – e repasse – indica para quais interfaces o

comutador deve enviar o quadro que chegou através de uma análise à tabela de comutação – uma tabela semelhante a tabela de roteamento, exceto que utiliza o endereço MAC para descobrir para qual interface deve ser levado o quadro.

Com isso, quando um quadro chega no comutador, existem três operações que este pode realizar:

- Não existe entrada na tabela: neste caso o comutador envia a mensagem em *broadcast* para tentar alcançar seu destino.
- Existe uma entrada na tabela, mas é a mesma a qual chegou: não é necessário nenhum repasse e o quadro pode ser descartado.
- Existe uma entrada na tabela e ela é diferente da interface de entrada: a função de repasse é ativada e o quadro é colocado no buffer da interface de saída.

No entanto, é necessário que haja um preenchimento desta tabela de comutação, sendo isto feito de forma inteligente e de melhor desempenho do que um hub, através da aprendizagem automática. O comutador realiza isso gravando o endereço MAC dos nós que enviam dados por ele, desse modo, se todos os nós da rede tiverem enviado dados a ele, ele saberá o mapeamento de todos os nós. Por isso o comutador é um dispositivo *plug and play full-duplex* – o nó e o comutador podem transmitir sem colisões.

Alguns outros características do comutador são:

- Eliminação de colisões: o comutador só envia um pacote por enlace, sendo a sua velocidade de envio a soma das velocidades das interfaces.
- Enlaces heterogêneos: os enlaces do comutador podem executar em diferentes mídias e com diferentes velocidades.
- Sistema de gerenciamento: além de manter os dados estatísticos para gerenciamento da rede, o comutador tem o poder de desconectar um adaptador que está poluindo a rede devido a alguma falha e com isso poupa a rede de alguma queda.

Comutadores x Roteadores

Apesar de ambos apresentarem semelhantes quase que idênticas, existem alguns vantagens e desvantagens na hora de utiliza-los. Começando pelo comutador, apesar de sua vantagem de *plug and play* e velocidade de processamento alta devido a necessidade de ter apenas 2 camadas, o comutador perde na questão de falha, caso um hospedeiro transmitisse quadro loucamente de forma a quebrar a rede, ele o conseguiria porque o comutador sempre faz o repasse dos quadro em *broadcast*, não fazendo nenhuma verificação, além da necessidade de uma tabela ARP de grande porte em casos de haver muitos nós na rede e a limitação da topologia de comutação para uma árvore geradora.

Os roteadores, apesar de ganharem na questão de limitação de topologia, visto que são capazes de criar uma topologia rica de larga escala e ainda contém *firewall* que impedem ataques a rede, perdem na questão de processamento visto que possuem uma camada a mais e não são *plug and play*, necessitando de uma configuração antes de iniciar.

Redes Sem Fio

Daremos uma breve pincelada no sistema de Redes sem fio para estudar o mecanismo CDMA. Além disso, é importante entender como as redes sem fio funcionam em sua base visto que é uma tecnologia que está sendo utilizada constantemente na atualidade.

Começando da base, os elementos da rede sem fio são bastante semelhantes aos da rede cabeada, onde os hosts sem fio são as nossas máquinas, sejam elas desktops, laptops, celulares, todos as nossas máquinas que rodam as aplicações. Além dos hosts, também existe uma estação de base que pode ser cabeada ou não, que fica responsável por fazer o repasse entre os pacotes enviados pela rede sem fio e a rede cabeada. Concomitantemente, temos o enlace sem fio, responsável por conectar a estação e o hospedeiro, existindo protocolos de acessos múltiplos responsáveis por coordenar o acesso.

Os elementos da rede sem fio podem ser estruturados de dois modos, modo de infraestrutura com a estação base responsável pelo repasse dos pacotes e gerenciamento da rede. Mas também existe uma estrutura não-estruturada chamada de Ad Hoc, onde não existem estações-base, sendo os nós dentro do alcance do enlace responsáveis pelo repasse dos pacotes

Característica da rede sem fio

Vamos estudar algumas características da rede sem fio, as comparando com a redes cabeadas.

Um das mais marcantes é da redução da força de sinal, apesar de redes cabeadas também perderem a força com a distância, esta característica é mais marcante na rede sem fio devido aos empecilhos físicos que existem no meio e impedem a passagem dos sinais. Juntamente temos a interferência de outras fontes, que acabam por "bagunçar" o sinal que está sendo transmitido. Como outra inconveniência há a propagação de múltiplos caminhos, visto que por ser sinais, eles são propagados e podem acabar sendo refletidos por objetos, chegando na destino várias vezes em diferentes tempos, sendo necessário tratar tal inconveniência.

Com isso, existe uma relação chamada SNR (relação sinal-ruído), que nós indica que é mais fácil extrair o sinal do ruído em altas taxas de SNR. Podemos então fazer uma análise entre o SNR e o BER (taxa de erro de bits). De acordo com a camada física, temos uma relação direta entre o aumento da potência, o aumento da SNR e diminuição da BER. No entanto, se nos for dada uma SNR, devemos escolher a camada física que atende o requisito BER e que dar maior vazão, tendo que a SNR pode variar com a mobilidade, com adaptação dinâmica da camada física.

O nosso último problema das rede sem fio é a interferência gerada na comunicação dos hospedeiros, que podem interferir no recebimento de um intermediário, visto que, como é baseado em sinais, os sinais podem acabar se sobrepondo ou nem se quer saber da existência do outro.

IEEE 802.11

A I3E é uma arquitetura de rede sem fio que oferece tanto uma estrutura de modo infraestrutura ou de modo ad hoc, onde um BSS (*Basic Service Set*) é uma "célula" que contém os hosts e o ponto de acesso (estação-base). Além disso utilia

CSMA/CA para acesso múltiplo. Deste modo, o hospedeiro deve se conectar a uma AP (*acess point*) através do DHCP para obter um endereço de IP. Além disso, o AP tem que ter uma frequência dada pelo administrador do AP, de preferência com frequência diferente do vizinho para não gerar interferência.

Para que um host se associe a uma AP ele pode utilizar dois tipos de busca:

- Busca passiva: quadros *beacon* são transmitidos pelos APs, o hospedeiro capita o quadro e responde ao AP selecionado.
- Busca ativa: quadro de solicatação *probe* é enviado em *broadcast* para a rede, esperando uma resposta pelas APs disponíveis, que então é iniciado a associação entre o host e o AP selecionado.

O I3E também oferece suporte a colisões para dois ou mais nós transmitindo ao mesmo tempo através do mecanismo CSMA – escutar antes de transmitir – o que impede a colisão com transmissões de outros nós em curso. Como não é fácil detectar as colisões, o I3E visa evitar colisões através do CSMA/CA (*collision avoidance*). Com o CSMA/CA o transmissor espera o canal estar quieto para enviar o quadro, caso contrário, espera um tempo aleatório para tentar novamente. O receptor fica responsável por enviar um ACK para evitar o problema de terminal oculto.

Para evitar colisões, o CSMA primeiro envia um RTS (*Request to send*) à estação-base para reservar um canal para si e poder enviar pacotes grandes e evitar colisões de tais. Apesar disso, o RTS pode sofrer colisão, mas ainda assim ele é pequeno. Como resposta, o RTS envia uma resposta em *broadcast* indicando que está livre para envio CTS. O CTS é escutado por todos os nós, mas apenas o transmissor envia o quadro dos dados, enquanto que outras estações adiam suas transmissões.