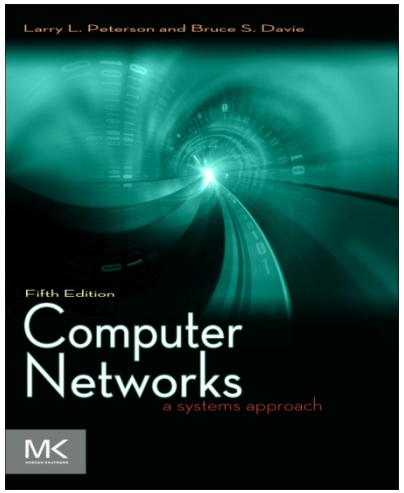


Capítulo 1

Fundamentos

Problemas

- Como construir uma rede escalável que suporte aplicações distintas?
- O que é uma rede de computadores?
- Como uma rede de computadores difere de outros tipos de redes?
- O que é uma arquitetura de rede de computadores?

Visão geral do capítulo

- Aplicações
- Requisitos
- Arquitetura de rede
- Implementação de software de rede
- Desempenho

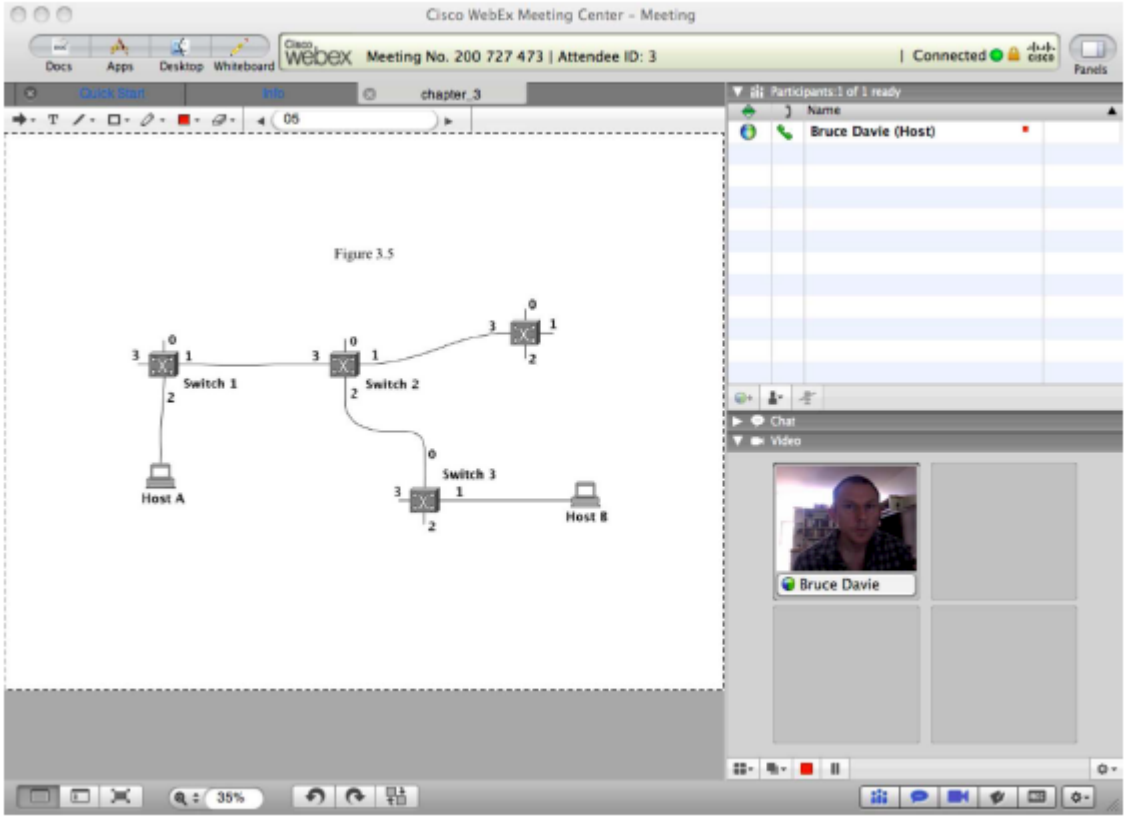
Objetivos do capítulo

- Explorar os requisitos que diferentes aplicações e comunidades impõem a uma rede de computadores
- Introduzir a ideia de arquitetura de rede
- Introduzir alguns elementos chaves na implementação de software de rede
- Definir métricas que serão utilizadas para avaliar o desempenho de uma rede de computadores

Aplicações

- A maioria das pessoas conhecem a Internet (uma rede de computadores) por meio de aplicações
  - World Wide Web
  - Email
  - Redes sociais
  - Streaming de audio e vídeo
  - Compartilhamento de arquivos
  - Mensagens instantâneas
  - …

Exemplo de uma aplicação



Uma aplicação multimídia que inclui vídeo conferência

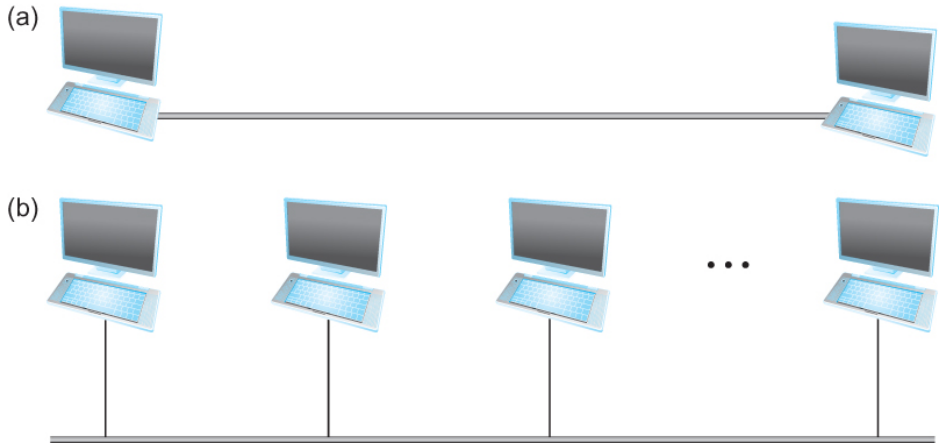
Protocolos deAplicação

- URL
  - Uniform resource locator
  - http://www.cs.princeton.edu/~llp/index.html
- HTTP
  - Hyper Text Transfer Protocol
- TCP
  - Transmission Control Protocol
- 17 mensagens para uma requisição de URL
  - 6 para encontrar o endereço IP (Internet Protocol)
  - 3 para estabelecimento da conexão TCP
  - 4 para a requisição HTTP e confirmação
    - Request: I got your request and I will send the data
    - Reply: Here is the data you requested; I got the data
  - 4 mensagens para terminar a conexão TCP

Requisitos

- Programador de Aplicações
  - Listar os serviços que sua aplicação precisa: entrega de dados com atraso limitado
- Projetista de Rede
  - Projetar uma rede com recursos compartilhados efetiva em termos de custo
- Provedor de Rede
  - Listar as características de um sistema que seja fácil de gerenciar

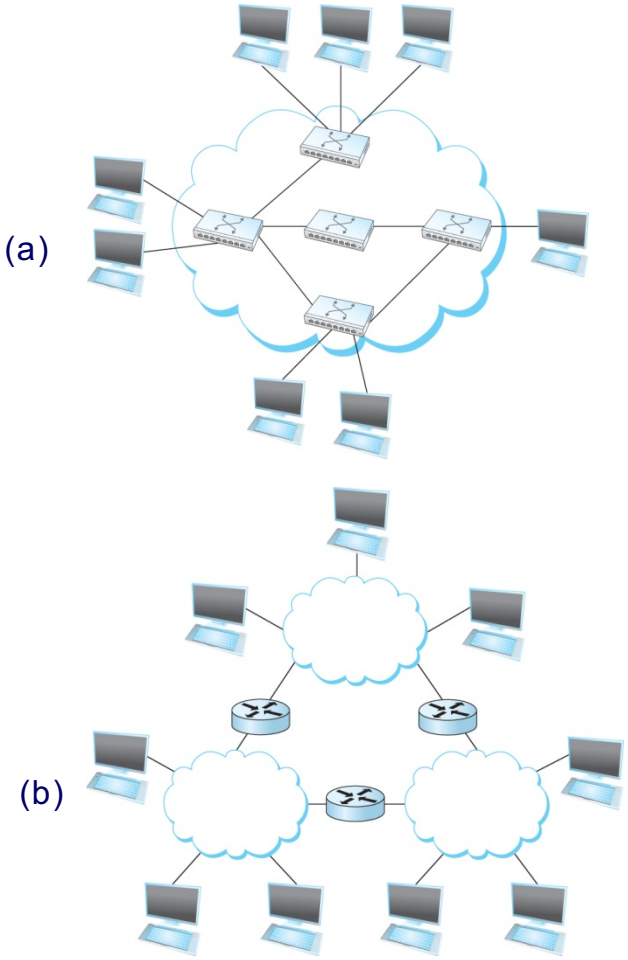
Conectividade



- Precisamos entender as seguintes terminologias
  - Escalável
  - Enlace (link)
  - Nós
  - Ponto-a-ponto
  - Acesso múltiplo
  - Rede comutada
    - Comutação por circuito
    - Comutação por pacote
  - Pacote, mensagem
  - Armazenamento-e-repasse (Store-and-forward).
  - Armazenamento-e-repasse vs Comutação por circuito

(a) Ponto-a-ponto
(b) Acesso múltiplo

Conectividade

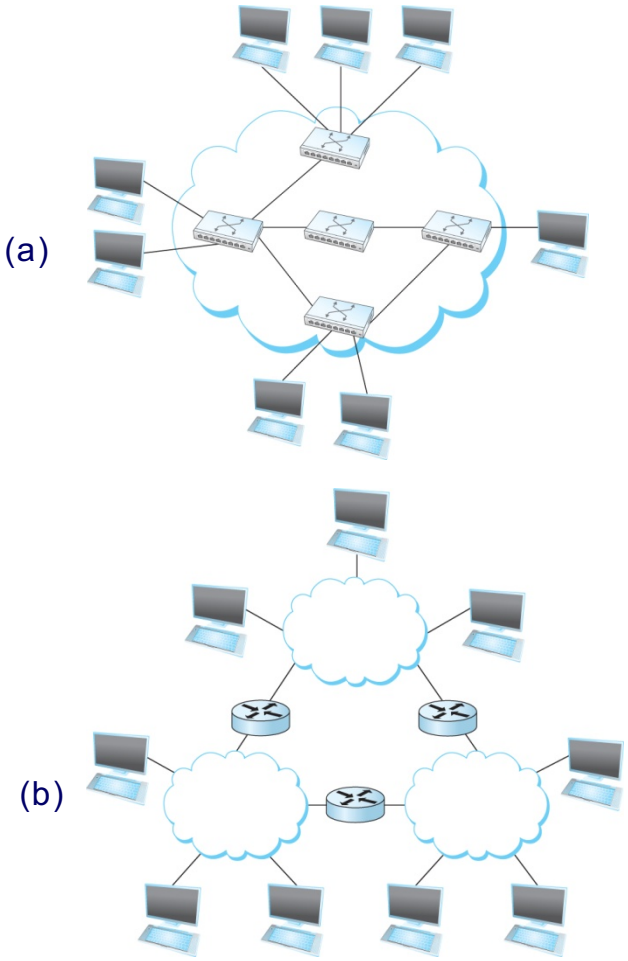


(a) Uma rede comutada

(b) Interconexão de redes

- Terminologias (cont.)
  - Nuvem (Cloud)
  - Hosts
  - Comutadores (Switches)
  - internetwork
  - Roteador/gateway
  - Conectividade Host-to-host
  - Endereço
  - Roteamento
  - Unicast/broadcast/multicast
  - Internet vs internet

Nuvem (nota de esclarecimento)



(a) Uma rede comutada

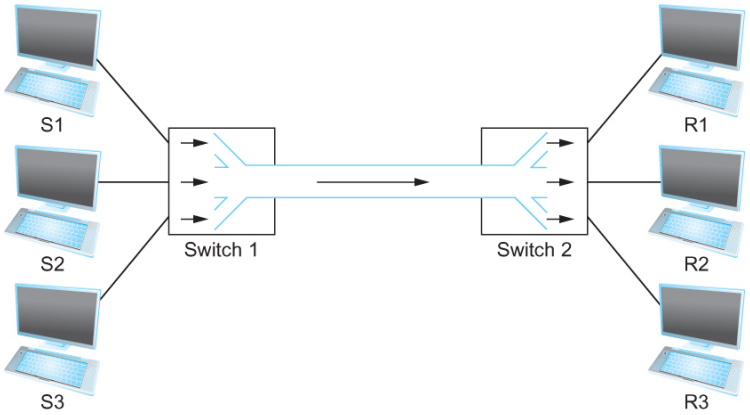
(b) Interconexão de redes

- Terminologias (cont.)
  - O termo nuvem neste curso é utilizado para representar uma rede como uma única entidade, ou seja, para esconder os detalhes de qual tecnologia é utilizada para construir a rede. Não confundir com o termo de computação na nuvem que é utilizado para descrever um conjunto de servidores interconectados em um provedor de serviço e utilizado para hospedar aplicações ou dados.

Main takeaway

- A principal ideia que se pode tirar desses conceitos e terminologias é que podemos definir uma rede recursivamente como consistindo de dois ou mais nós conectados por um enlace físico ou como duas ou mais redes conectadas por um nó.
- Um desafio chave em se prover conectividade é a definição de um endereço para cada nó que seja alcançável na rede (incluindo suporte para broadcast e multicast) e o uso de tal endereço para encaminhar mensagens para o(s) nó(s) destino(s) apropriados.

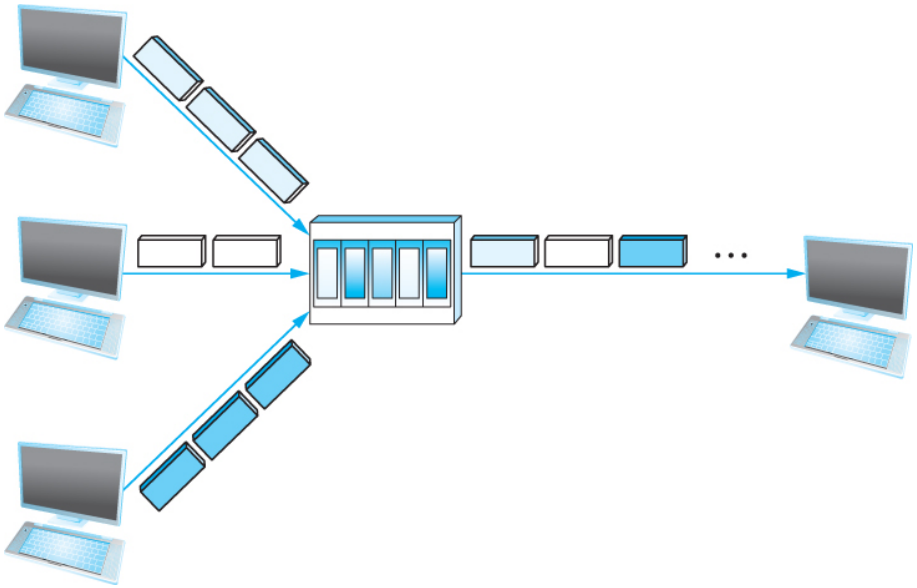
Compartilhamento eficiente de recursos



Multiplexação de múltiplos fluxos lógicos em um único enlace físico

- Recursos: enlaces e nós
- Como compartilhar um enlace?
  - Multiplexação
  - Demultiplexação
  - Multiplexação síncrona por divisão de tempo (STDM)
    - Dados transmitidos em slots prédeterminados

Compartilhamento eficiente de recursos



Um comutador multiplexando pacotes de múltiplas fontes em um enlace compartilhado

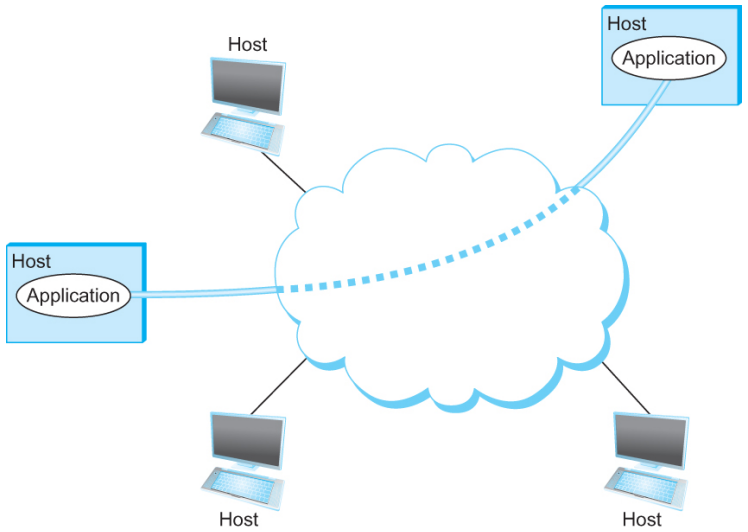
- Multiplexação por divisão de frequência (FDM) – TV a cabo, rádio e TV
- Tanto STDM quanto FDM não utilizam as capacidades dos enlaces de forma eficiente. Limitações: ociosidade do enlace e número máximo de fluxos
- Multiplexação estatística
  - Dados são transmitidos de acordo com a demanda de cada fluxo
  - O que é um fluxo?
  - Pacotes vs. Mensagens
  - FIFO, Round-Robin, Prioridades (Quality-of-Service (QoS))
  - Congestionamento?
- LAN, MAN, WAN, CAN, PAN
- SAN (System Area Networks/Storage Area Network)

Main takeaway

- Multiplexação estatística define uma maneira eficiente para que múltiplos usuários compartilhem recursos (nós e enlaces). Ela define o pacote como a granularidade com a qual os enlaces são alocados para diferentes fluxos.
- Uma alocação justa da capacidade do enlace para fluxos diferentes e a maneira como se lida com congestionamento são alguns dos desafios para a implementação da multiplexação estatística.
- Uma maneira de se caracterizar uma rede de computadores é pelo seu tamanho (LAN, MAN, WAN, CAN, SAN, PAN).

Suporte para Serviços Comuns

- Canais lógicos
  - Caminho de comunicação aplicação-a-aplicação (pipe)



Processos comunicando sobre um canal abstrato

Suporte para Serviços Comuns

- Desafio: identificar o conjunto correto de serviços. O objetivo é esconder a complexidade da rede da aplicação sem restringir muito os projetistas de aplicações.
- Requisitos:
  - Guarantia de entrega ou não?
  - Mensagens entregues na ordem de envio ou não?
  - Privacidade é uma preocupação ou não?

Padrões Comuns de Comunicação

- Cliente/Servidor
- Dois tipos de canais de comunicação
  - Canais Requisição/Reposta (Request/Reply)
  - Canais de streaming de mensagens (Message Stream Channels)

Confiabilidade

- A rede deve esconder erros
- Bits são perdidos
  - Erros de bit (1 vira 0, e vice versa)
  - Erros de rajada (burst errors) – vários erros consecutivos
- Pacotes são perdidos (congestionamento)
- Pacotes sofrem atrasos
- Pacotes são entregues fora de ordem
- Falhas de enlaces e nós

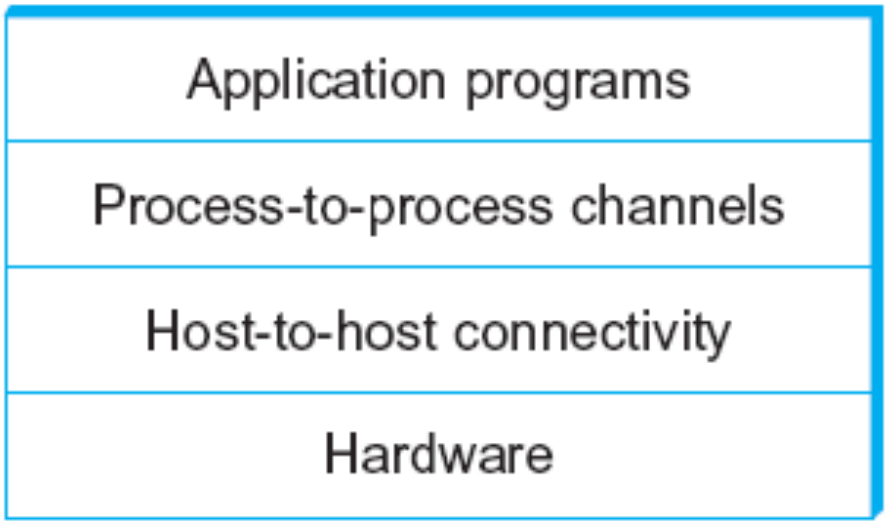
Main takeaway

- A definição de canais lógicos envolve o entendimento dos requisitos de aplicação e o reconhecimento das limitações da tecnologia básica da rede.
- O desafio é encurtar a distância entre o que a aplicação espera e o que a tecnologia pode oferecer

Abstração

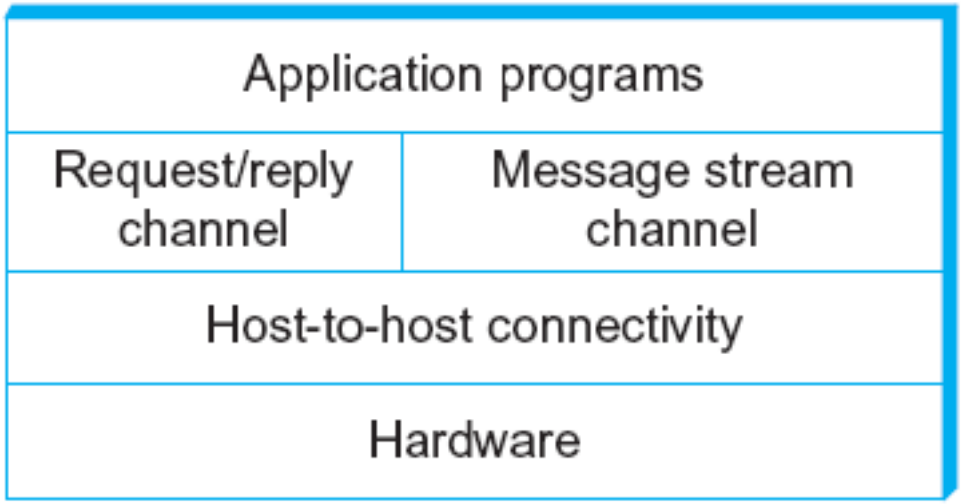
- Abstração, ou ocultamento de detalhes por meio de uma interface bem definida, é a ferramenta fundamental utilizada por projetistas de sistema para gerenciar complexidade
- Abstrações levam naturalmente a camadas, especialmente em sistemas em rede
  - A ideia geral é que você começa com os serviços oferecidos pelo hardware básico de rede e então adiciona uma sequência de camadas, cada uma oferecendo um nível mais alto (mais abstrato) de serviço
  - Os serviços oferecidos nas camadas superiores são implementados em função dos serviços oferecidos pelas camadas inferiores

Arquitetura de rede



Examplo de um sistema de rede em camadas

Arquitetura de rede

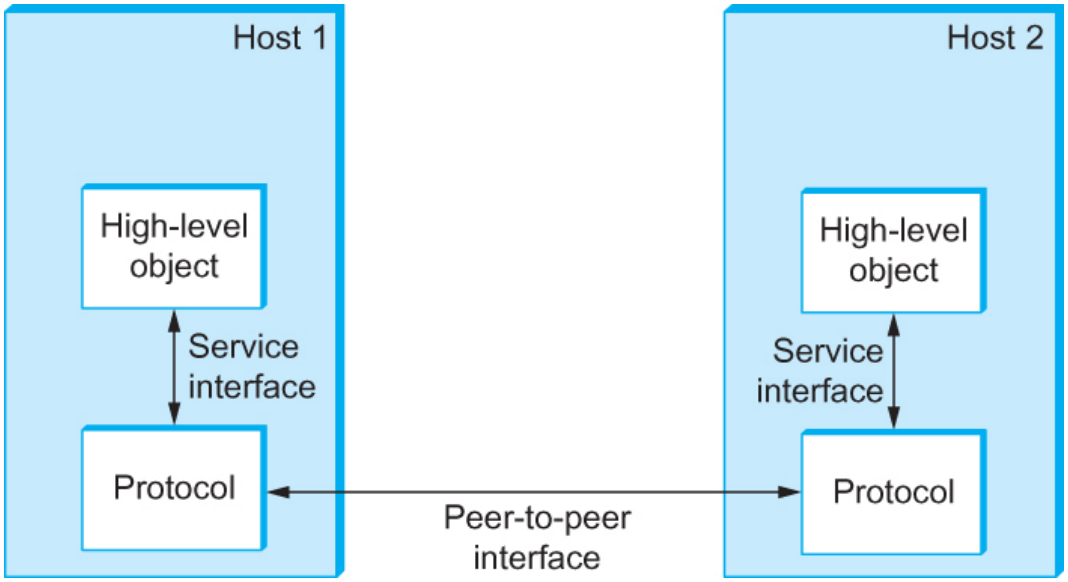


Sistema em camadas com alternativas de abstrações disponíveis em uma determinada camada

Protocolos

- Protocolos definem as interfaces entre as camadas do mesmo sistema e entre as camadas de um sistema par
- Blocos básicos (*building blocks*) de uma arquitetura de rede
- Cada protocolo tem duas interfaces diferentes
  - Interface de serviço: operações no protocolo
  - Interface par-a-par: mensagens trocadas com o par
- O termo “protocolo” é sobrecarregado
  - Especificação da interface par-a-par
  - Módulo que implementa a interface

Interfaces

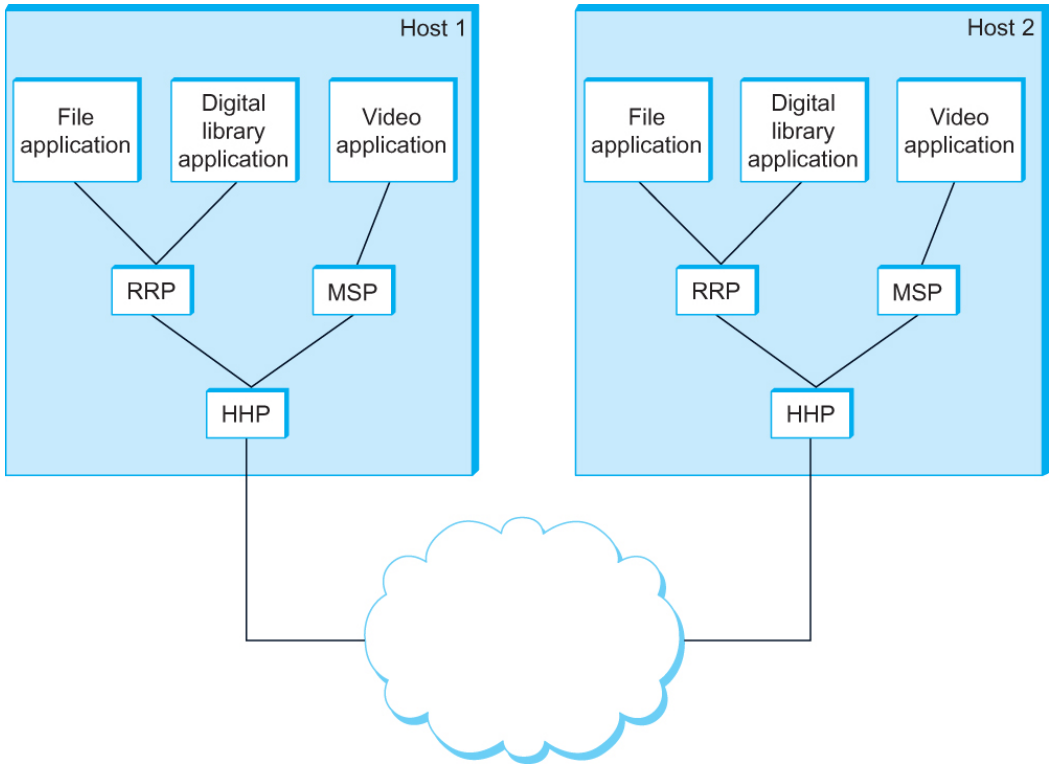


Interfaces de Serviço and Par-a-Par

Protocolos

- Um protocolo define um serviço de comunicação que ele exporta localmente (interface de serviço) junto com um conjunto de regras que governam as mensagens que o protocolo troca com seu(s) par(es) para implementar o serviço de comunicação (interface par).
- Comunicação par-a-par é indireta (exceto na camada física). Cada protocolo se comunica com seu par passando mensagens para protocolos de níveis mais baixos que entregam as mensagens para seus pares.

Grafo de Protocolo

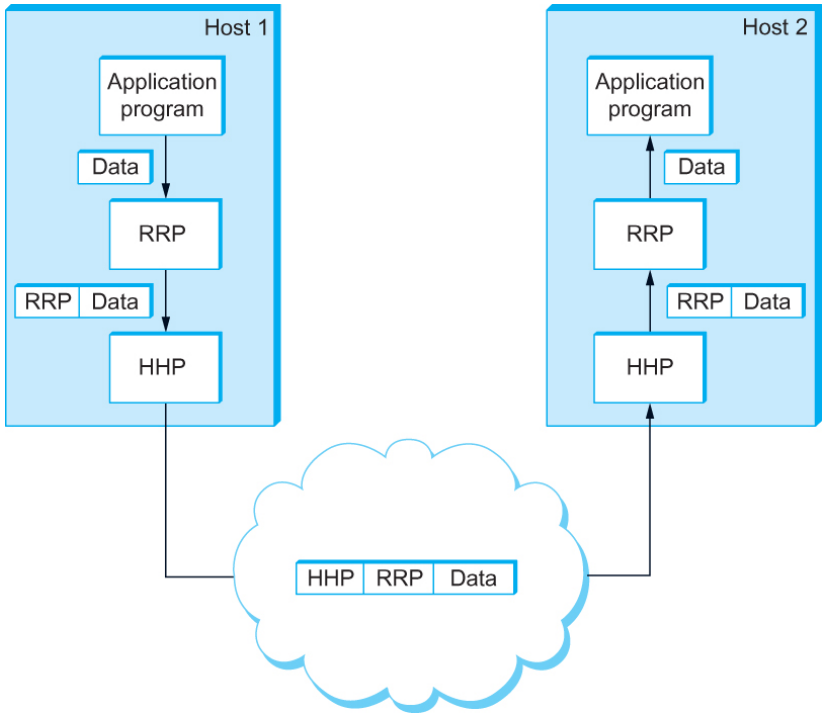


Exemplo de um grafo de protocol. Os vértices são os protocolos e as arestas as relações de dependência. Chamamos de *arquitetura de rede* o conjunto de regras que governam a forma e conteúdo de um grafo de protocolos.

Protocolos

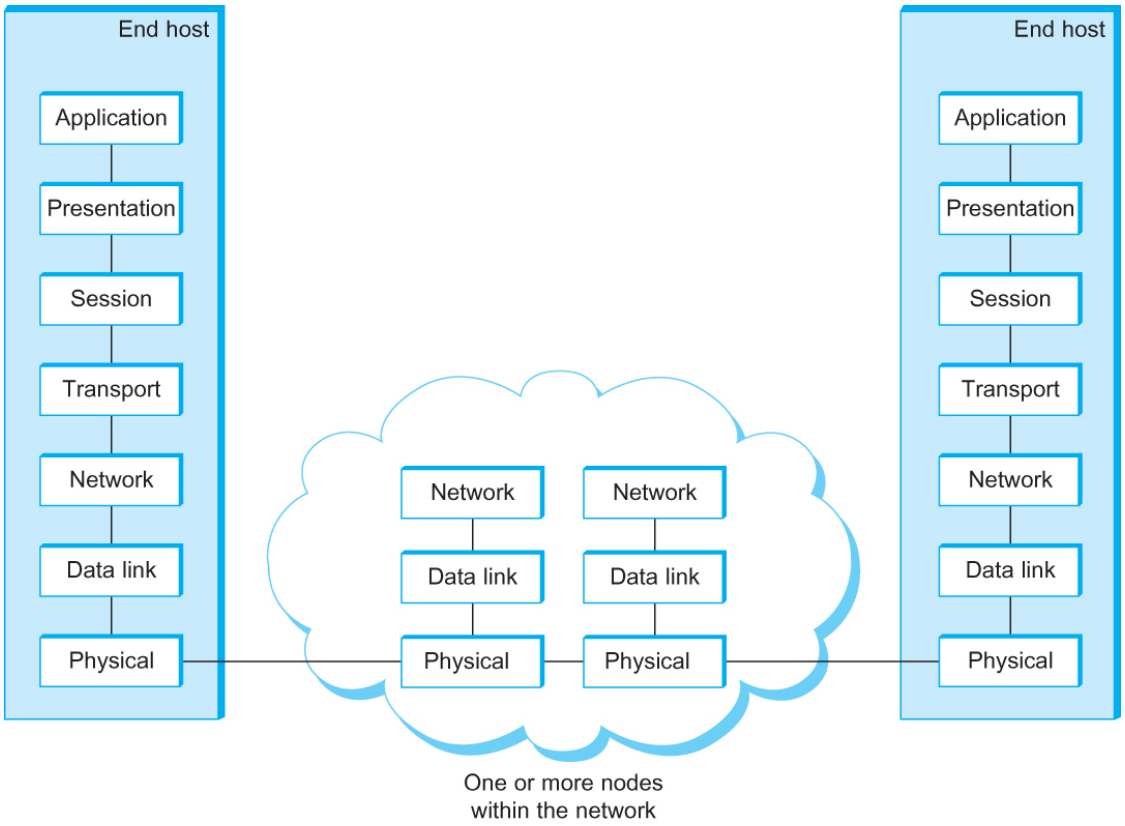
- Especificação de protocolos: prosa, pseudo-código, diagrama de transição de estados
- Interoperável: quando duas ou mais entidades implementam a especificação corretamente
- IETF: Internet Engineering Task Force

Encapsulamento



Mensagens de um nível são encapsuladas dentro de mensagens de níveis mais baixos

Arquitetura OSI da ISO



O modelo de referência OSI de sete camadas da ISO

OSI – Open Systems Interconnection

ISO – International Organization for Standardization

Descrição das camadas

- Camada Física (Physical Layer)
  - Lida com a transmissão de bits brutos em uma enlace de comunicação
- Camada de Enlace (Data Link Layer)
  - Agrega uma cadeia de bits em uma unidade maior chamada quadro (*frame)*
  - O adaptador (placa) de rede junto com o driver de dispositivo do SO implementam o protocolo nesta camada
  - Quadros são entregues nos hosts
- Camada de Rede (Network Layer)
  - Lida com o roteamento entre nós dentro de uma rede comutadoa por pacotes
  - A unidade de dados trocadas entre os nós nesta camada é chamada pacote ou datagrama (*packet/datagram)*

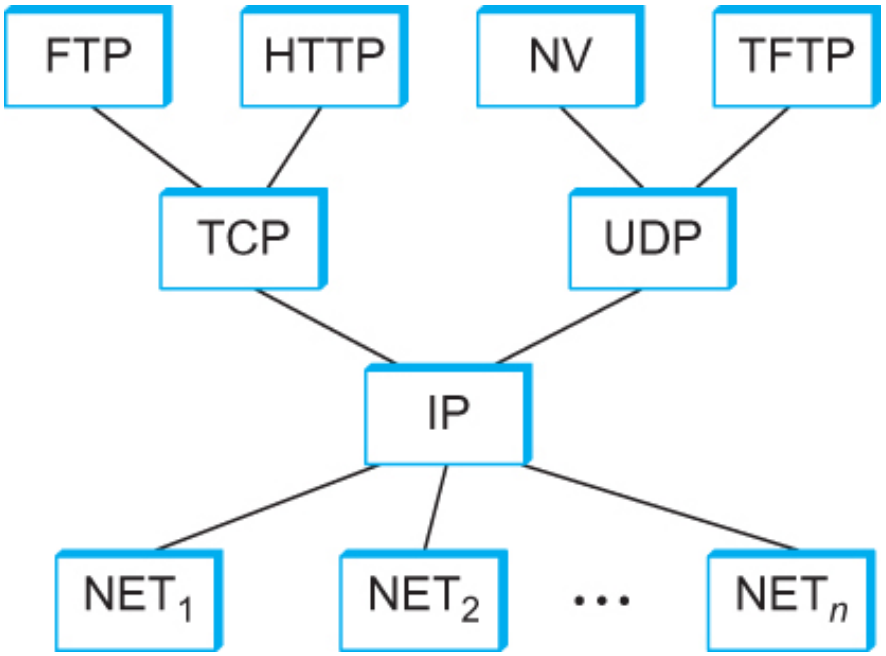
As duas/três camadas são implementadas em todos os nós da rede

Descrição das Camadas

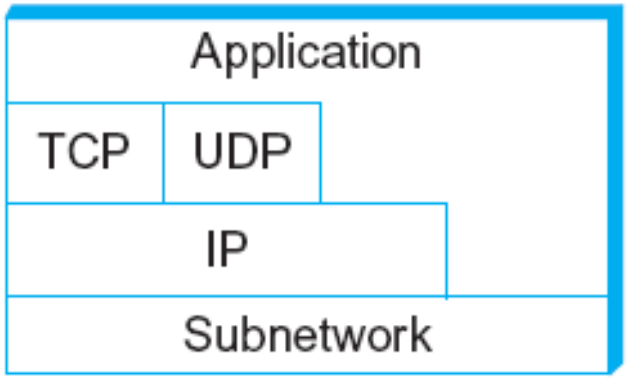
- Camada de Transporte (Transport Layer)
  - Implementa um canal processo-a-processo
  - A unidade de dados trocada nesta camada é chamada de mensagem ou segmento (*message or segment)*
- Camada de Sessão (Session Layer)
  - Oferece um espaço de nomes que é utilizado para unir cadeias de transporte que são partes de uma única aplicação
- Camada de Apresentação (Presentation Layer)
  - Preocupa-se com o formato de dados trocados entres pares
- Camada de Aplicação (Application Layer)
  - Padroniza os tipos de comuns de trocas de mensagens entre aplicações

A camada de transporte e superiores são tipicamente implementadas apenas nos hosts e não nos nós (roteadores e switches) intermediários

Arquitetura Internet



Grafo de Protocolos da Internet



Visão alternativa da arquitetura da Internet. A camada de enlace é muitas vezes referenciada como camada de subrede.

Arquitetura Internet

- Definida pela IETF
- Também conhecida como Arquitetura TCP/IP
- Três características principais
  - Não impõe utilização estrita das camadas. A aplicação pode não utilizar a camada de transporte e enviar dados diretamente para a camada IP
  - Forma de ampulheta – larga no topo, fina na cintura e larga na parte de baixo. A camada IP é o ponto focal da arquitetura
  - Para que um novo protocolo seja oficialmente incluído na arquitetura, é preciso que haja uma especificação do protocolo e pelo menos uma (preferencialmente duas) implementações representativas da especificação
  - Filosofia: *We reject kings, presidents, and voting. We believe in rough consensus and running code. (David Clark)*

Application Programming Interface (API)

- Interface exportada pela rede
- Como a maioria dos protocolos de rede são implementados em software (pelo menos aqueles das camadas superiores) e como parte do sistema operacional (SO), quando falamos em interface exportada pela rede, queremos dizer a interface que o SO oferece para acesso à rede
- A interface é chamada de API de rede (Network Application Programming Interface)

Application Programming Interface (Sockets)

- A interface socket Interface foi originalmente oferecida pela distribuição Unix de Berkeley (BSD)
  - Hoje em dia oferecida em praticamente todos os sistemas operfacionais

- Cada protocolo oferece um certo conjunto de *serviços* e a API oferece a sintaxe pela qual os serviços podem ser invocados em um SO em particular

Socket

- O que é um socket?
  - O ponto onde um processo se conecta à rede
  - Uma interface entre uma aplicação e a rede
  - A aplicação cria o socket
- A interface define operações para
  - Criar um socket
  - Conectar um socket à rede
  - Enviar e receber mensagens pelo socket
  - Fechar o socket

Socket

- Socket Family
  - PF_INET especifica a família Internet
  - PF_UNIX especifica um pipe Unix
  - PF_PACKET especifica acesso direto à interface de rede (i.e., it bypasses the TCP/IP protocol stack)

- Socket Type
  - SOCK_STREAM é usado para especifica uma cadeia de bytes (*byte stream*)
  - SOCK_DGRAM é usado para especificar um serviço orientado a mensagens, tais como o oferecido pelo protocolo UDP

Criando um socket

```
int sockfd = socket(address_family, type, protocol);
```

- O número retornado pela função é o descritor para o socket criado

- `int sockfd = socket (PF_INET, SOCK_STREAM, 0);`
- `int sockfd = socket (PF_INET, SOCK_DGRAM, 0);`

A combinação PF_INET e SOCK_STREAM implica a utilização de TCP

Modelo Cliente-Servidor com TCP

Servidor

- Abertura passiva
- Prepara para aceitar conexões, não estabelece uma conexão de fato

Servidor invoca

```
int bind (int socket, struct sockaddr *address,
                    int addr_len)
int listen (int socket, int backlog)
int accept (int socket, struct sockaddr *address,
                  int *addr_len)
```

Modelo Cliente-Servidor com TCP

Bind

- Associa o socket criado e especificado pelo primeiro parâmetro a um endereço específico do servidor
- O endereço é uma estrutura de dados que combina IP e porta

Listen

- Define quantas conexões podem ficar pendentes no socket especificado

Modelo Cliente-Servidor com TCP

Accept

- Realiza a abertura passiva
- Operação bloqueante
  - Não retorna até que um participante remoto estabeleça uma conexão
  - Quando retorna, a função retorna um novo socket que corresponde à nova conexão estabelecida e o argumento *address* corresponde ao endereço do participante remoto

Modelo Cliente-Servidor com TCP

Cliente

- A aplicação realiza uma abertura ativa
- Ela diz com quem ela quer se comunciar

Cliente invoca

```
int connect (int socket, struct sockaddr *address,
             int addr_len)
```

Connect

- Não retorna até que TCP tenha estabelecido uma conexão na qual a aplicação pode enviar dados (ou retorna em situações de erro)
- O parâmetro address contém o endereço da máquina remota

Modelo Cliente-Servidor com TCP

Na prática

- O cliente geralmente especifica somente o endereço do participante remoto (servidor) e deixa o SO preencher as informações locais
- O servidor geralmente espera (listen) por mensagen em uma porta padronizada (*well-known port*)
- O cliente não se importa com a porta que ele usa para si mesmo, o SO simplesmente seleciona uma porta não utilizada

Modelo Cliente-Servidor com TCP

Depois que a conexão foi estabelecida, a aplicação invoca duas operações

```
int send (int socket, char *msg, int msg_len,
                                        int flags)

int recv (int socket, char *buff, int buff_len,
                                          int flags)
```

Exemplo de Aplicação: Cliente

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 5432
#define MAX_LINE 256

int main(int argc, char * argv[])
{
        FILE *fp;
        struct hostent *hp;
        struct sockaddr_in sin;
        char *host, buf[MAX_LINE];
        int s, len;

        if (argc == 2)
           host = argv[1];
        else {
                fprintf(stderr, "usage: simplex-talk host\n");
                exit(1);
        }
```

Exemplo de Aplicação: Cliente


```
/* translate host name into peer's IP address */
hp = gethostbyname(host);
if (!hp)  {
        fprintf(stderr, "simplex-talk: unknown host: %s\n", host);
        exit(1);
}
/* build address data structure */
bzero((char *)&sin, sizeof(sin));
sin.sin_family = AF_INET;
bcopy(hp->h_addr, (char *)&sin.sin_addr, hp->h_length);
sin.sin_port = htons(SERVER_PORT);
/* active open */
if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        perror("simplex-talk: socket");
        exit(1);
}
if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
        perror("simplex-talk: connect");
        close(s);
        exit(1);
}
/* main loop: get and send lines of text */
while (fgets(buf, sizeof(buf), stdin)) {
        buf[MAX_LINE-1] = '\0';
        len = strlen(buf) + 1;
        send(s, buf, len, 0);
}
}
```

Exemplo de Aplicação: Servidor

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define SERVER_PORT 5432
#define MAX_PENDING 5
#define MAX_LINE 256

int main()
{
        struct sockaddr_in sin;
        char buf[MAX_LINE];
        int len, s, new_s;
        /* build address data structure */
        bzero((char *)&sin, sizeof(sin));
        sin.sin_family = AF_INET;
        sin.sin_addr.s_addr = INADDR_ANY;
        sin.sin_port = htons(SERVER_PORT);
        /* setup passive open */
        if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
                        perror("simplex-talk: socket");
                        exit(1);
        }
```

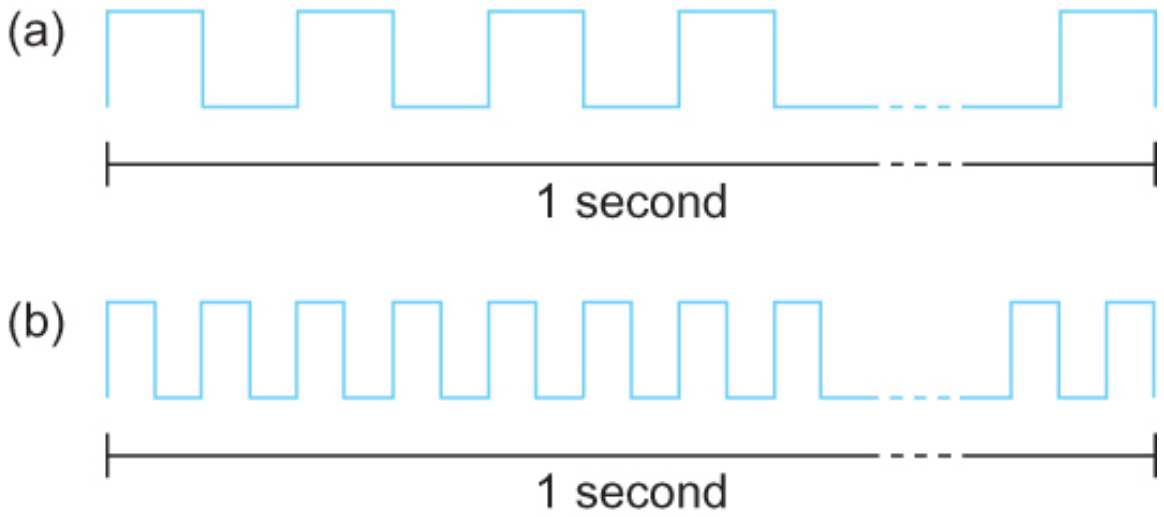
Exemplo de Aplicação: Servidor

```
        if ((bind(s, (struct sockaddr *)&sin, sizeof(sin))) < 0) {
                perror("simplex-talk: bind");
                exit(1);
        }
        listen(s, MAX_PENDING);
        /* wait for connection, then receive and print text */
        while(1) {
                if ((new_s = accept(s, (struct sockaddr *)&sin, &len)) < 0)  {
                        perror("simplex-talk: accept");
                        exit(1);
                }
                while (len = recv(new_s, buf, sizeof(buf), 0))
                        fputs(buf, stdout);
                close(new_s);
        }
}
```

Desempenho

- Largura de banda (Bandwidth)
  - Largura da faixa de frequência
  - Número de bits por segundo que podem ser transmitidos em um enlace de comunicação
- 1 Mbps: 1×10^6 bits/second
- 1×10^{-6} segundos para transmitir cada bit. Em uma linha do tempo, cada bit ocupa o espaço de 1 micro segundo.
- Em um enlace de 2 Mbps a largura de um bit é de 0.5 micro segundo.
- Quanto menor a largura de um bit, maior a taxa de transmissão por unidade de tempo.

Largura de banda - Bandwidth



Os bits transmitidos em uma largura de banda específica podem ser pensados como tendo alguma largura:

(a) bits transmitidos a 1Mbps (cada bit possui largura de 1 μs);

(b) bits transmitidos a 2Mbps (cada bit possui largura de 0.5 μs).

Desempenho

- Latência = Atr. Propagação + Atr. Tansmissão + Atr. fila
- Atr. de Propagação = distância/velocidade da luz
- Atr. Transmissão = tamanho mens./largura de banda

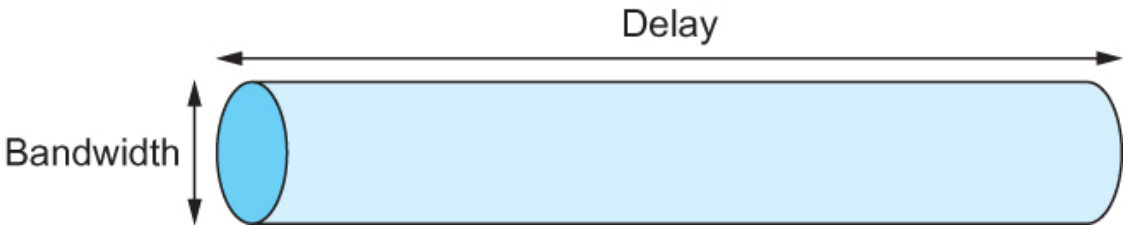
- Transmissão de um bit => propagação é importante
- Transmissão de um número grande de bytes => largura de banda é importante

Atraso X LB (Delay X Bandwidth)

- Podemos imaginar o canal entre um par de processos como um cano
- Latência (delay) é o comprimento e LB é a largura do cano
- Atraso de 50 ms e LB de 45 Mbps

⇒ 50×10^{-3} seconds x 45×10^{6} bits/second

⇒ 2.25×10^{6} bits = 280 KB de dados.



Rede como um cano

Atraso X LB (Delay X Bandwidth)

- A importância relativa de LB e latência depende da aplicação
  - Para transferência de grandes arquivos, LB é o fator crítico
  - Para pequenas mensages (HTTP, NFS, etc.), a latência é o fator crítico
  - A variância da latência (jitter) pode afetar algumas aplicações (*e.g.*, audio/vídeo conferência)

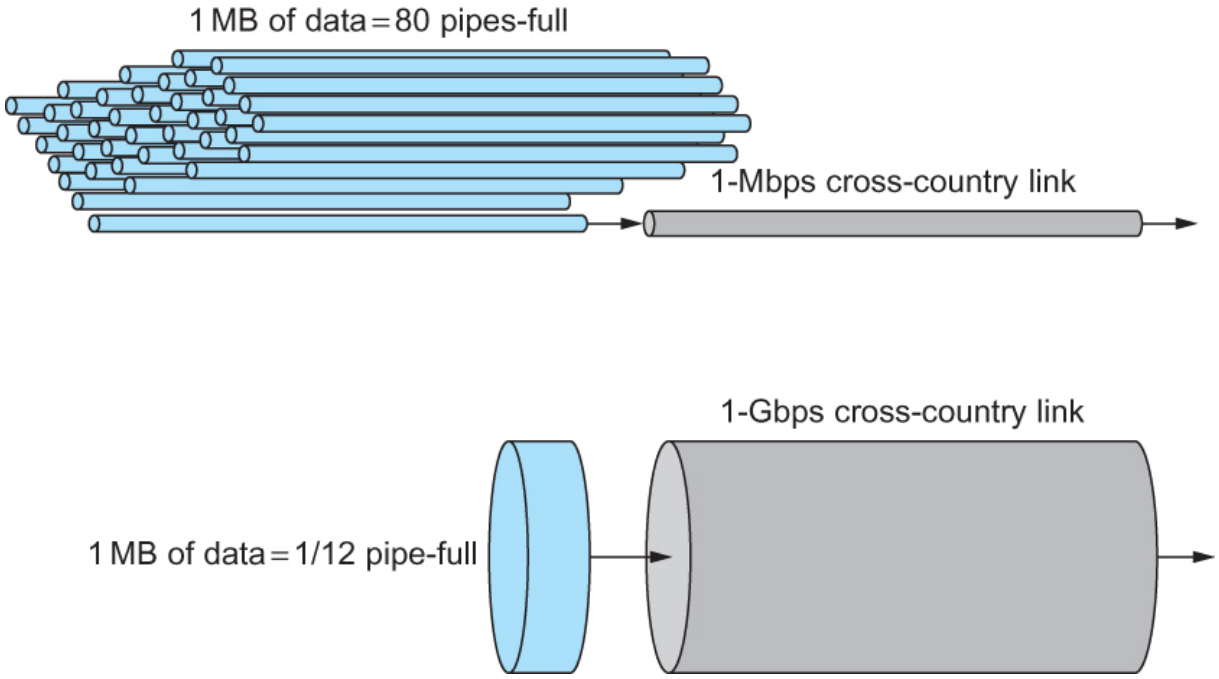
Atraso X LB (Delay X Bandwidth)

- Quantos bits o transmissor deve transmitir até que o primeiro bit chegue no receptor se o transmissor quiser manter o cano cheio
- É necessário esperar uma latência de um sentido (*one-way latency*) para se receber a resposta
- Se o transmissor não mantiver o cano cheio—enviar uma quantidade de dados igual ao produto atraso x LB antes de parar para esperar um sinal—o transmissor não utilizará a capacidade total da rede

Atraso X LB (Delay X Bandwidth)

- LB infinita
  - RTT domina (RTT = Round-trip time)
  - Throughput = TransferSize / TransferTime
  - TransferTime = RTT + TransferSize/Bandwidth
- Tudo é relativo
  - Um arquivo de 1-MB num enlace de 1-Gbps parece um pacote de 1-KB em um enlace de 1-Mbps

Relação entre LB e latência



Um arquivo de 1-MB preencheria um enlace de 1-Mbps 80 vezes, mas preencheria apenas

1/12 de um enlace de 1-Gbps (latência 100 ms)

Resumo

- Identificamos o quê se esperar de uma rede de computadores
- Definimos uma arquitetura em camadas que servirá como padrão para nossos estudos
- Discutimos a interface socket que é utilizada por aplicações para invocar os serviços do subsistema de rede do SO
- Discutimos duas métricas de desempenho que podemos utilizar para analisar o desempenho de redes de computadores