Redes - Resumo 2

DNS (DOMAINS NAME SYSTEM)

Sistema hierárquico e distribuído de gestão de nomes para computadores, serviços ou qualquer máquina conectada à Internet ou a uma rede privada.

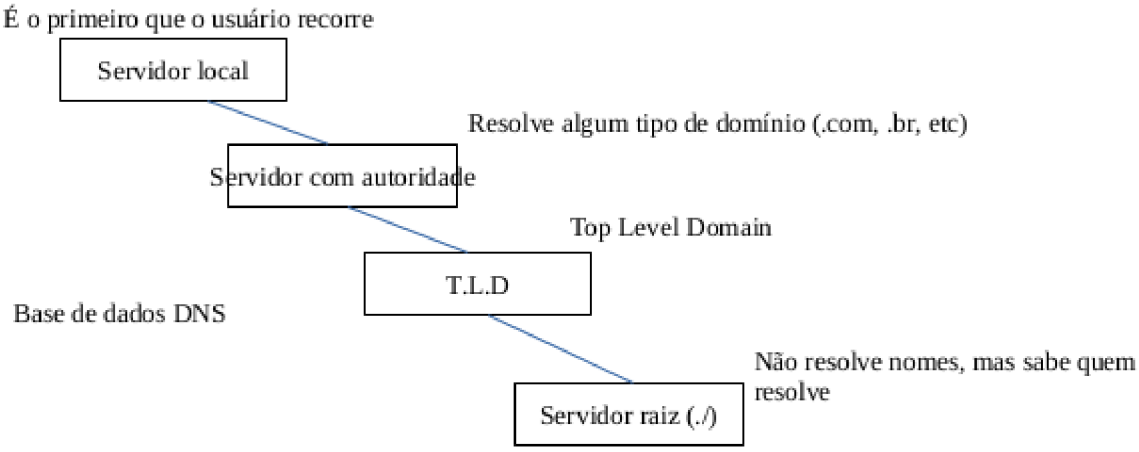
O DNS serve para resolver a questão dos hosts e servidores na Internet.

A principal função do DNS é mapear nomes de domínios em registros de recursos.

Registros de Recursos: DNS deve retornar um pacote com várias informações além do IP de destino. A tarefa do DNS é receber um nome de domínio e responder um registro de recurso.

Se um único servidor tivesse o banco de dados DNS inteiro, o servidor ficaria tão sobrecarregado que seria inútil. Outro problema é que se o servidor caísse, toda a Internet seria atingida.

Servidores DNS



Consulta recursiva: Nessa consulta o endereço será procurado começando pelo servidor local, caso não encontre, a busca é feita no próximo servidor da hierarquia e assim por diante até o servidor raiz, por fim, o servidor raiz encontra quem resolve o domínio, o processo inverso segue a mesma lógica recursiva. **A desvantagem é que se o domínio estiver em um área pertencente a outro servidor raiz, será gasto muito tempo para consultar servidores que não resolvem o domínio.**

Consulta encadeada: A busca começa pelo servidor raiz, é feita uma pergunta diretamente para a raiz referente a qual é o endereço do servidor que resolve aquele domínio, com o endereço, o host pode se conectar ao servidor.

A world wide web

WWW: Estrutura arquitetônica que permite acesso a diversos arquivos espalhadas por milhões de máquinas na Internet.

W3C: Responsável pelo desenvolvimento da Web, padronização de protocolos da Internet e incentivo à interoperabilidade entre sites.

O lado do cliente:

http://www.abcd.com/produtos.html

Uma URL possui três partes:

1. Nome do protocolo, por exemplo, http.
2. Nome DNS da máquina em que a página está localizada, por exemplo, www.abcd.com.
3. (Normalmente) O nome do arquivo que contém a página, por exemplo, produtos.html.

Para poder exibir qualquer página, os navegadores tem de reconhecer seu formato. As páginas da Web são padronizadas utilizando a linguagem HTML. Um navegador é um interpretador de HTML.

O lado do servidor:

Um servidor recebe o nome de um arquivo para pesquisar e retornar. As etapas que um servidor executa em seu loop principal são:

1. Aceitar uma conexão TCP de um cliente (um navegador).
2. Obter o nome do arquivo solicitado.
3. Obter o arquivo (do disco).
4. Retornar o arquivo ao cliente.
5. Encerrar a conexão TCP.

Um servidor deve ser projetado pensando no número de acessos do disco que podem acontecer, algumas soluções seriam colocar na cache os arquivos mais requisitados e tornar o servidor multithreaded (multitarefa).

URLs: As URLs são necessárias para saber qual o nome da página, onde ele está localizada e como pode ser localizada.

HTTP

Protocolo de transferência de hipertexto. Define as mensagens que podem ser enviadas e as respostas que podem ser recebidas entre cliente/servidor.

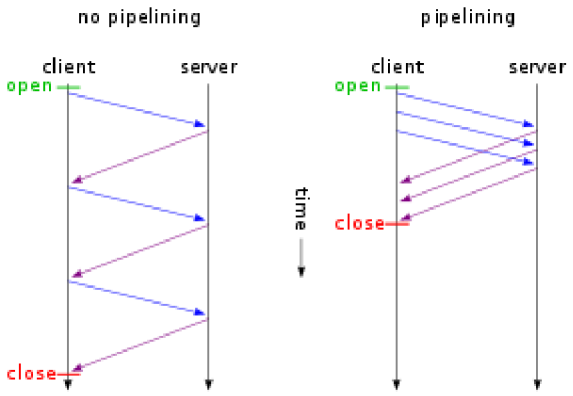
TCP pota 80.

TCP faz uma série de serviços: **controle de fluxo, controle de perda, mensagens duplicadas, mensagens com erros, etc.**

Vantagem que nem o cliente e nem o servidor precisam se preocupar com esses serviços.

HTTP 1.0: Uma requisição por vez

HTTP 1.1: Mais de uma requisição por vez. Pipeline (estabelece uma ou mais requisições além da próxima a ser executada.)



Métodos HTTP

Operações que o protocolo HTTP permite que sejam feitas.

1. **GET** - Requisitar arquivos no servidor;
2. **HEAD** - Requisitar cabeçalho;
3. **PUT** - Enviar arquivos pro servidor;
4. **POST** - Enviar arquivos pro servidor porem não sobrescreve (Controle de Versão);
5. **DELETE** - Remover um arquivo;
6. **TRACE** - Verificar as solicitações que foram recebidas;

7. **OPTION** - Consultar informações do servidor.

Cabeçalhos de solicitação: Linhas adicionais com mais informações.

- **User-Agent:** Informar ao servidor sobre seu navegador, SO e outras propriedades.
- **Accept**: Tipo de páginas que o cliente pode manipular.
- **Host**: O nome DNS do servidor.
- **Authorization**: Provar que o cliente tem direito de ver a página solicitada.

Camada de Transporte

Fornece *comunicação lógica entre processos* de máquinas diferentes.

Camada de Rede - Comunicação lógica entre os hosts

Mensagens são divididas em seguimentos que são remontados no destino.

Cálculo de Bits {

$2^{n}_{valores}$ → n bits => 0 ---- 2^{n} - 1

}

TCP

- Entrega confiável
- Controle de fluxo*
- Garante a entrega
- Controle de Congestionamento*
- Orientado à conexão*

Controle de Fluxo - Evita que o transmissor sobrecarregue o receptor

Controle de Congestionamento - Evita que o transmissor sobrecarregue a rede

Orientado à Conexão - Ocorre uma troca de mensagens de controle entre as partes

Protocolo de ponto a ponto - Estabelece uma conexão entre as partes

Transferência confiável sobre canais confiáveis: Não há erros de bits e não há perdas de pacotes.

Canal com erros de bits: O canal pode trocar os valores dos bits num pacote (o checksum pode detectar esses erros). Para recuperar esses erros, podem ser usados:

- Reconhecimentos (ACKs): O receptor avisa explicitamente ao transmissor que o pacote foi recebido corretamente.
- Reconhecimentos negativos (NAKs): O receptor avisa explicitamente ao transmissor que o pacote tem erros. Quando o transmissor recebe NAK, ele reenvia o pacote.

ACK/NAK corrompido: não pode retransmitir pois pode duplicar os dados.

Para tratar as duplicatas, o transmissor acrescenta um **número de sequência** em cada pacote.

Canais com erros e perdas: Além de trocar os valores dos bits, o canal de transmissão pode perder pacotes (dados aos ACKs). Possível solução seria um temporizador.

Protocolos com paralelismo: O transmissor envia vários pacotes ao mesmo tempo, todos esperando para serem reconhecidos.

A comunicação é bidirecional na mesma conexão

O TCP com números de sequência e ACKs.

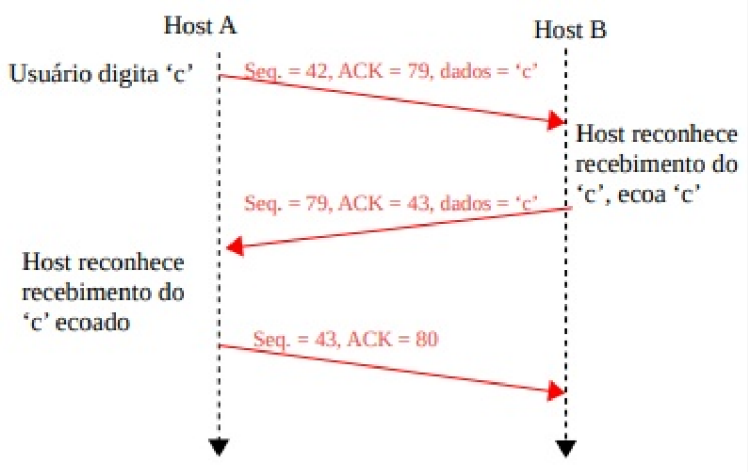
nº de sequência: O nº do 1º byte do segmento

nº de ACK: O nº do próximo byte que se espera do outro lado

O ACK é cumulativo

A ordem de remontagem dos pacotes fica a critério do implementador, o TCP não define.

Exemplo:



Temporização

Esperar um tempo antes de retransmitir um segmento não confirmado.

A divisão do tempo de espera é o mais importante.

Tempo muito curto → Se os pacotes estiverem atrasados, vai ocorrer muita duplicação e isso gera sobrecarga.

Tempo muito longo → A rede se torna lenta para agir em problemas de perda de pacotes.

Pode-se estimar o tempo baseado no envio de um pacote

Pode-se enviar vários pacotes e estimar a média de tempo de envio

Deve existir armazenamento no transmissor e no receptor

Controle de Fluxo

Os processos da aplicação podem ser muito lentos para ler os dados da camada de transporte

Solução - Estimar um tempo que a aplicação leva para ler um segmento.

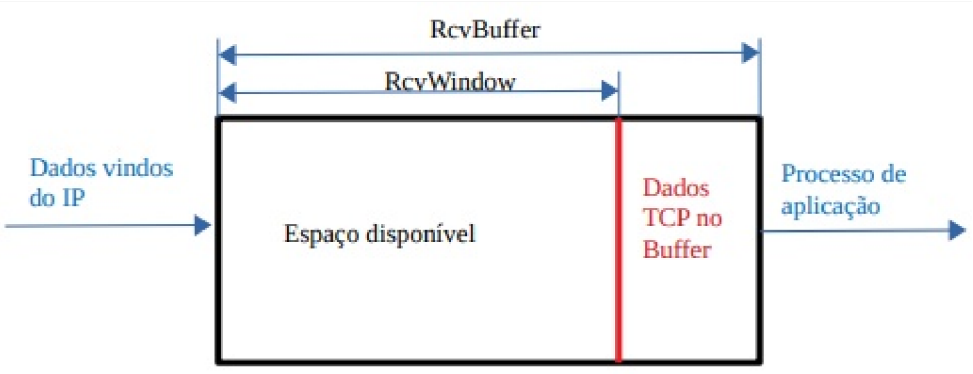
Força a camada de rede a enviar segmentos com o tempo acima do que foi estimado.

O lado receptor possui um buffer de recepção. Os processos de aplicação podem ser lentos para ler o buffer. No controle de fluxo, o transmissor não deve esgotar os buffers de recepção enviando dados rápido demais

O receptor deve informar a área disponível incluindo o valor nos segmentos (RcvWindow).

transmissor limita os dados não confirmados ao RcvWindow, isso garante que o buffer do receptor não vai

transbordar



Gerenciamento de Conexão

É sempre o cliente que inicia a conexão.

O servidor espera pedidos de conexão.

Início da conexão TCP

1. O cliente envia um TCPSYN ao servidor com seu número de sequência inicial
2. O servidor recebe o SYN e responde com o SYNACK
3. O servidor processa o SYN, aloca os buffers e inicializa seu número de sequência
4. O cliente recebe o SYNACK

Fim da conexão TCP

1. O cliente envia um TCPFIN ao servidor
2. O servidor recebe o FIN, retorna o ACK, encerra a conexão e envia o FIN ao cliente

3. O cliente recebe o FIN e retorna o ACK
4. O servidor recebe o ACK, conexão finalizada

UDP

- Não confiável
- Sem controle de fluxo
- Seguimentos podem chegar fora de ordem
- Sem conexão

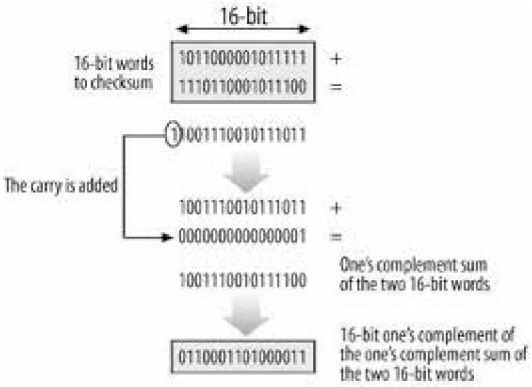
Vantagens

- É mais rápido
- Cabeçalho do segmento é reduzido
- Usado para streaming

O UDP é um protocolo sem conexão, ou seja, não há apresentação entre o UDP transmissor e o receptor.

É possível o transporte confiável, mas isso deve ser implementado na camada de aplicação.

CHECKSUM: Detectar erros no segmento transmitido.



O transmissor insere o checksum no segmento UDP.

O receptor recalcula o checksum e compara o valor obtido com o valor armazenado no segmento.

Se os valores forem diferentes, significa que ocorreu algum erro.

Se os valores forem iguais significa que não foram detectados erros.