Resumo Ragnarok: Camada de Aplicação!

A camada de aplicação é uma camada de abstração que engloba protocolos que realizam a comunicação fim-a-fim entre aplicações. É responsável por prover serviços para aplicações de modo a separar a existência de comunicação em rede entre processos de diferentes computadores.

Ela, basicamente, cria as especificações de como será a comunicação entre duas aplicações entre hosts.

**O protocolo da camada de aplicação define:**

* Tipo das mensagens trocadas, mensagens de requisição e resposta;
* Sintaxe dos tipos de mensagem: os campos nas mensagens e como são delineados ;
* Semântica dos campos, ou seja, significado da informação nos campos;
* Regras para quando e como os processos enviam e respondem às mensagens.

**De qual serviço de transporte uma aplicação necessita?**

* **Perda de dados** 
  + Algumas aplicações (ex.: áudio) podem tolerar alguma perda
  + Outras aplicações (ex.: transferência de arquivos, sessão remota) exigem transferência de dados 100% confiável
* **Temporização**
  + Algumas aplicações (ex.: telefonia IP, jogos interativos) exigem baixos atrasos para serem “efetivos”
* **Banda passante**
  + Algumas aplicações (ex.: multimídia) exigem uma banda mínima para serem “efetivas”
  + Outras aplicações (“aplicações elásticas”) melhoram quando a banda disponível aumenta

Arquiteturas de Aplicação:

* **Cliente-Servidor.**
  + **Servidor:** 
    - Hospedeiro sempre ativo
    - Endereço IP permanente
    - Fornece serviços solicitados pelo cliente
  + **Clientes:** 
    - Comunicam-se com o servidor
    - Pode ser conectado intermitentemente
    - Pode ter endereço IP dinâmico
    - Não se comunicam diretamente uns com os outros
* **P2P**
  + Nem sempre no servidor
  + Sistemas finais arbitrários comunicam-se diretamente
  + Pares são intermitentemente conectados e trocam endereços IP
    - *Ex.: Gnutella*
  + Altamente escaláveis mas difíceis de gerenciar
* **Híbrida**
  + Napster
    - Transferência de arquivo P2P
    - Busca centralizada de arquivos:
      * Conteúdo de registro dos pares no servidor central
      * Consulta de pares no mesmo servidor central para localizar o conteúdo
  + Instant messaging
    - Bate-papo entre dois usuários é P2P
    - Detecção/localização de presença é centralizada:
      * Usuário registra seu endereço IP com o servidor central quando fica on-line.
      * Usuário contata o servidor central para encontrar endereços IP dos “amigos”.

**Serviços dos protocolos de transporte da Internet**

**Serviço TCP:**

* Orientado à conexão: conexão requerida entre processos cliente e servidor
* Transporte confiável entre os processos de envio e recepção
* Controle de fluxo: o transmissor não sobrecarrega o receptor
* Controle de congestionamento: protege a rede do excesso de tráfego
* **Não oferece:** garantias de temporização e de banda mínima

**Serviço UDP:**

* + Transferência de dados não confiável entre os processos transmissor e receptor
  + **Não oferece:** estabelecimento de conexão, controle de fluxo e de congestionamento, garantia de temporização e de banda mínima.

**HTTP - Hypertext Transfer Protocol**

**Visão geral:**

* Protocolo da camada de aplicação da Web
* Modelo cliente/servidor
  + Cliente: navegador que solicita, recebe e apresenta objetos da Web
  + Servidor: envia objetos em resposta a pedidos
* HTTP 1.0: RFC 1945
* HTTP 1.1: RFC 2068
* Utiliza TCP:
  + Cliente inicia conexão TCP (cria socket) para o servidor na porta 80
  + Servidor aceita uma conexão TCP do cliente
  + Mensagens HTTP (mensagens do protocolo de camada de aplicação) são trocadas entre o browser (cliente HTTP) e o servidor Web (servidor HTTP)
  + A conexão TCP é fechada
* HTTP é “stateless”
  + Por default, o servidor não mantém informação sobre os pedidos passados pelos clientes
  + Protocolos que mantêm informações de “estado” são complexos!
  + Histórico do passado (estado) deve ser mantido
  + Se o servidor/cliente quebra, suas visões de “estado” podem ser inconsistentes, devendo ser reconciliadas

Modelagem do Tempo de Resposta

* **Definição de RTT:** tempo para enviar um pequeno pacote que vai do cliente para o servidor e retorna.
* **Tempo de resposta:** 
  + Um RTT para iniciar a conexão TCP
  + Um RTT para requisição HTTP e primeiros bytes da resposta HTTP para retorno
  + Tempo de transmissão de arquivo
* **Total = 2 RTT+ tempo de transmissão**

Conexão HTTP

* **HTTP não persistente**
  + No máximo, um objeto é enviado sobre uma conexão TCP
  + O HTTP/1.0 utiliza HTTP não persistente
  + Características do HTTP não persistente: Requer 2 RTTs por objeto
  + Sistema Operacional deve manipular e alocar recursos do hospedeiro
  + para cada conexão TCP
  + (Mas os browsers freqüentemente abrem conexões TCP paralelas
  + para buscar objetos referenciados)
* **HTTP persistente**
  + Múltiplos objetos podem ser enviados sobre uma conexão
    - TCP entre o cliente e o servidor
  + O HTTP/1.1 utiliza conexões persistentes em seu modo padrão
  + **HTTP persistente**
  + **Servidor deixa a conexão aberta após enviar uma resposta**
  + **Mensagens HTTP subseqüentes entre o mesmo cliente/servidor são**
  + **enviadas pela conexão**
  + **Persistente sem pipelining: O cliente emite novas requisições apenas quando a resposta anterior for recebida**
  + **Um RTT para cada objeto referenciado**
  + **Persistente com pipelining: Padrão no HTTP/1.1**
  + **O cliente envia requisições assim que encontra um objeto**
  + **referenciado**
  + **Tão curto quanto um RTT para todos os objetos referenciados**