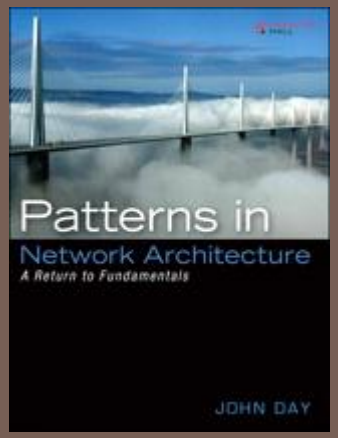


PERSEGUINDO A ARQUITETURA DAS CAMADAS MAIS ALTAS

Capítulo 4

Patterns in Network Architecture

Introdução

Citações

- *Você tem Telnet e FTP. O que mais você precisa?*

  – Alex McKenzie, 1975

- *Se o Terminal Virtual está na camada de apresentação, onde está a fonte de alimentação?*

  – *Al Reska, 1979*

- Estas citações revelam que as camadas superiores sempre foram misteriosas e não muito claras.

Estrutura Geral

- Será que existe alguma estrutura geral para organizar as funções das camadas superiores, como existem para as camadas inferiores?
- As camadas inferiores se organizaram muito mais rapidamente:
  - Em 1975 já era comum ouvir falar nas camadas de transporte, rede, enlace de dados e física.
  - As duas camadas inferiores são dependentes do meio físico e as duas superiores (ou do meio) são independentes do meio físico e fim-a-fim.

Dificuldade em Estruturar as Camadas Superiores

- Aparentemente nenhuma estrutura ou decomposição parece se aplicar.
- Parte disto foi devido à ausência de aplicações e, em um certo sentido, em muitas aplicações!
- No início havia poucas aplicações (3-4) acessadas através de "sockets bem conhecidos". Embora desde cedo tenha sido reconhecida a necessidade de algum tipo de diretório.
- Mas, no geral os protocolos de aplicação eram produtos pontuais, com características únicas para cada aplicação.

O Modelo OSI

- O trabalho para o OSI enfrentou o problema e por um tempo parecia que tivesse feito progresso.
- Mas, embora tenham identificado alguns elementos de uma estrutura geral, alguns passos errados na arquitetura tornaram o projeto das aplicações obscuros e um excesso de generalidade que requeria implementações complexas para as aplicações mais simples.
- Foram divisões internas que mataram o OSI.

Roteiro

- Este capítulo procura juntar o entendimento recente do que torna característico estes protocolos.
- Todos os problemas estão resolvidos?
- Longe disto, mas com o arcabouço a ser apresentado teremos uma visão mais clara e a possibilidade de avançarmos.
- Relacionamento entre redes e sistemas distribuídos.
- Redes não é "tudo" e não pode ser considerado que inclua toda a computação distribuída.
- Temos que entender bem a fronteira e o relacionamento entre elas.

Um Pouco de História

As Camadas Superiores da ARPANET

- Preocupação inicial não era com as aplicações e sim em:
  - Construir a rede e
  - Identificar o que daria para ser feito com computadores com arquiteturas tão diferentes!
  - Como colocar software de rede relativamente complexo em sistemas com os recursos já bastante limitados?

ARPANET: Diversidade das máquinas

- Comprimento das palavras (em bits):
  - 16, 18, 24, 32, 36 (dois tipos), 48, 64, etc.
- Sistemas operacionais:
  - Pelo menos uma dúzia com modelos bem distintos para E/S, processos, sistemas de arquivos, proteção, etc.

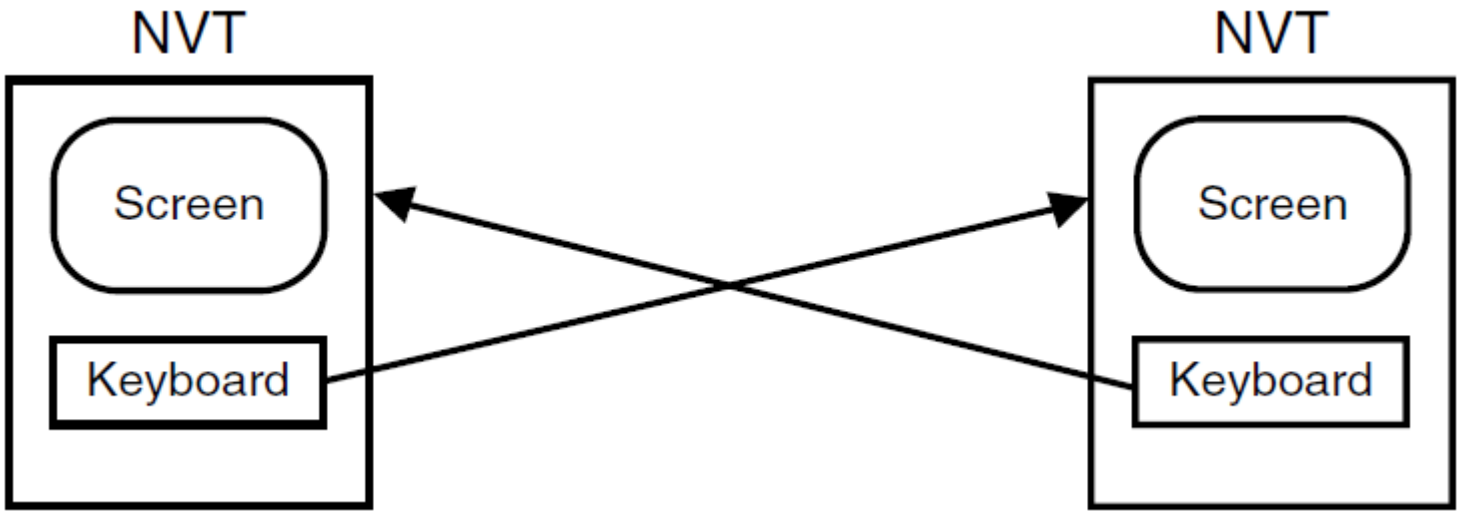
- Diretriz inicial: prover acesso através da rede a cada um destes sistemas como se fosse um usuário local.
  - Primeiras aplicações: acesso a terminal, transferência de arquivos, submissão de jobs para execução.

Telnet

- Foi o primeiro protocolo de terminal virtual.
- Não era uma aplicação de login remoto, mas um protocolo de driver de terminal.
- O login remoto é uma aplicação construída usando o Telnet.
- Os projetistas perceberam que ele não era apenas um protocolo para conectar *hosts* a terminais, mas que poderia ser usado também como um mecanismo de comunicação entre processos orientado a caractere: *midleware*.
- Poderia ser usado para conectar duas aplicações que tivessem sido escritas para interagir com humanos.

Telnet: NVT (*Network Virtual Terminal*)

- Representação canônica de um terminal básico.
- Definia um terminal rudimentar modo "rolagem" com pouquíssimos atributos.
- O terminal e o *host* convertem suas representações locais para a representação usada pelo NVT e vice versa.



Telnet: Mecanismo de Negociação

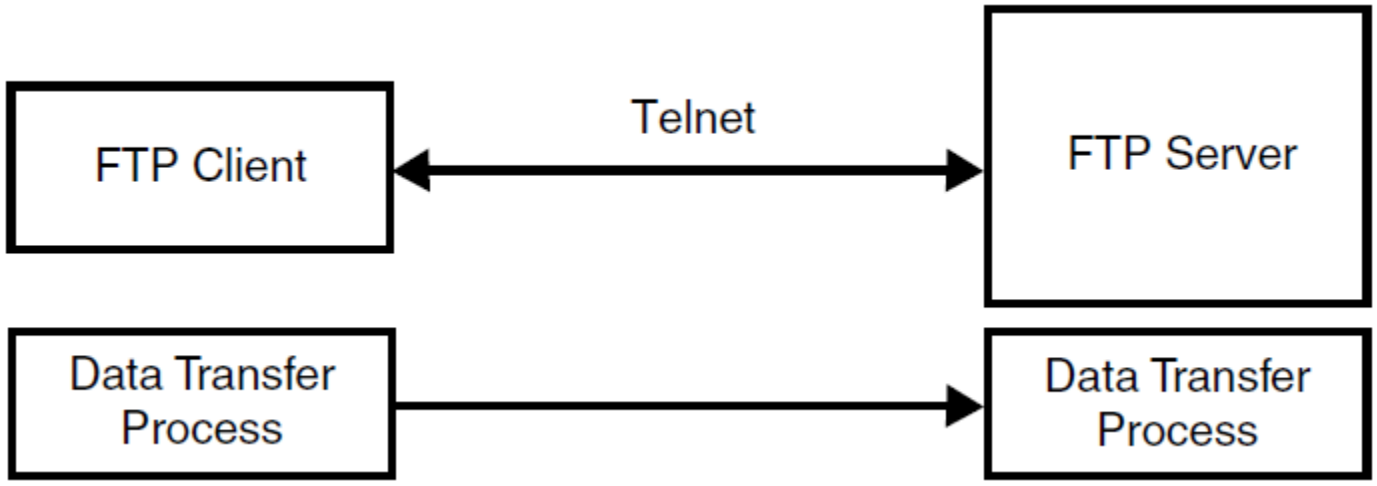
- Mecanismo simétrico: permitia que a solicitação de um fosse a resposta ao outro.
- No estabelecimento da conexão cada lado anuncia o que pretende e o que não pretende fazer (WILL/WONT),
- E aquilo que espera que o outro lado faça ou deixe de fazer (DO/DONT).

Telnet: Modo de Funcionamento

- No caso de terminais *half-duplex*:
  - Apenas a interface entre o protocolo e o usuário remoto seria *half-duplex*.
  - O protocolo operaria como *full-duplex*.
- Modelo fluxo (*stream*):
  - O uso do modelo fluxo implicou na necessidade de análise de cada caractere para identificar se seria um caractere de comando.
  - Se tivesse colocado os comandos Telnet e os dados do terminal em "registros" separados, teriam reduzido grandemente a sobrecarga.

FTP *(File Transfer Protocol)*

- Foi construído sobre o Telnet, parte por razões arquiteturais e parte por razões pragmáticas.
- Usa uma conexão Telnet para enviar os seus comandos de quatro caracteres seguidos normalmente por um único parâmetro terminado por um CRLF.

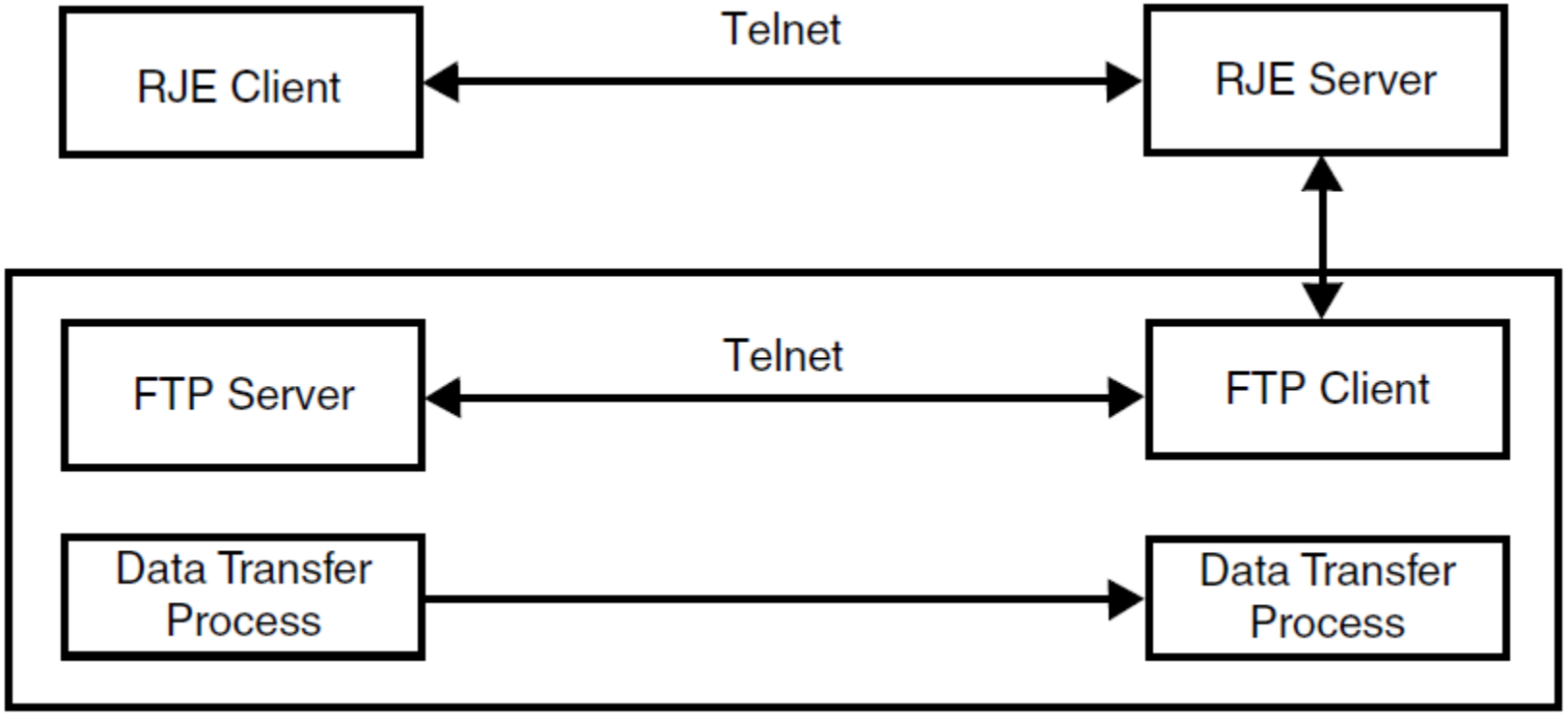


FTP

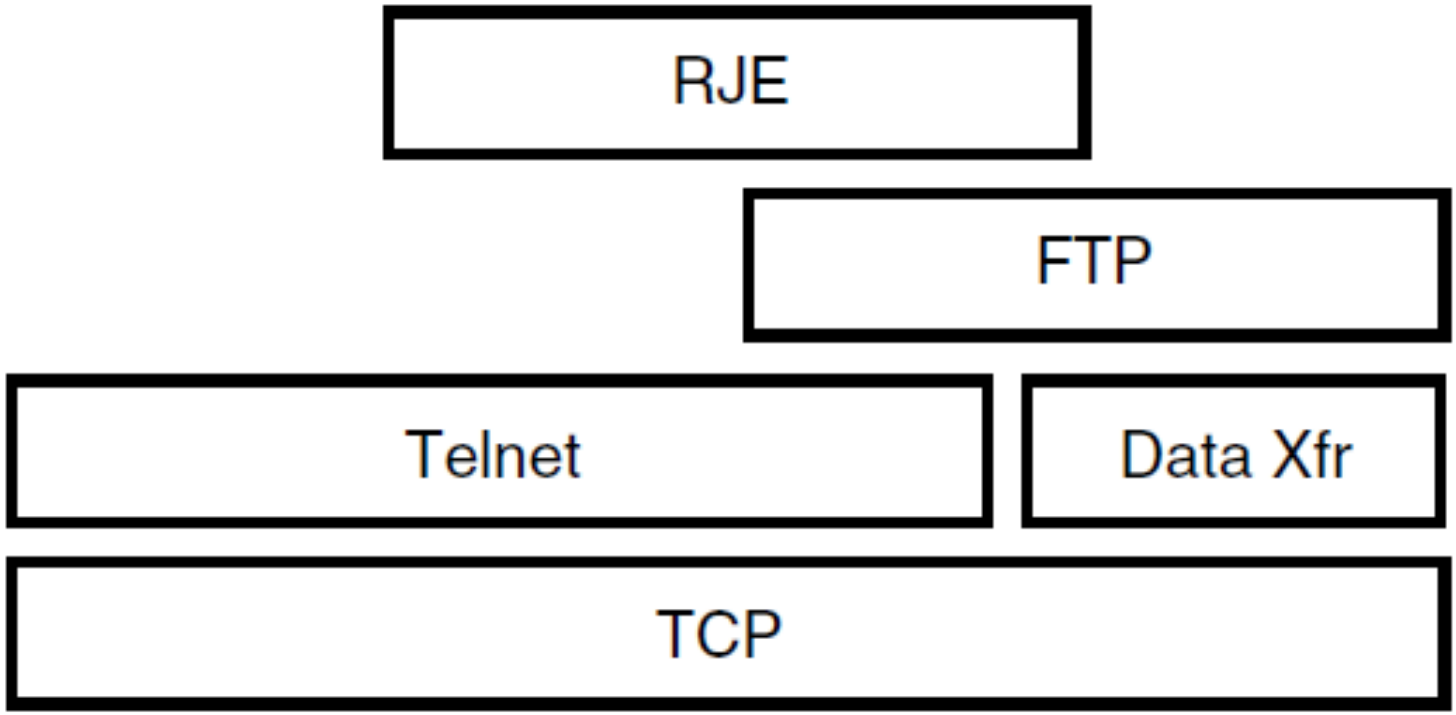
- A razão arquitetural para separar os fluxos de comandos e de dados era para que os comandos, em particular pedidos de interrupção (*abort*) não ficassem presos atrás de uma transferência de arquivo grande.
- O FTP definiu um sistema de arquivos rudimentar chamado de NVFS (*Network Virtual File System*) e os comandos básicos para realizar as transferências de arquivos e consulta ao sistema de arquivo remoto.
- Inicialmente correio eram dois comandos no FTP. Demorou um pouco para ser separado num protocolo distinto.

RJE *(Remote Job Entry)*

- Permitia a submissão de um programa para rodar numa máquina remota e recuperar a saída (normalmente um arquivo de impressão).



ARPANET: Arquitetura das Camadas Superiores



ARPANET: Lições Aprendidas

- Por mais rudimentar que fosse, provou que as aplicações poderiam ser construídas e podiam ser muito úteis.
- Ganhamos experiência valiosa com sistemas distribuídos.
- Aprendemos a dificuldade de lidar com diferenças sutis na semântica dos diversos sistemas para o que pareciam conceitos muito similares.
- Aprendemos a necessidade de encontrar um equilíbrio entre super especificação e manutenção da utilidade.

ARPANET: Lições Aprendidas

- Qual a estrutura necessária para uma rede de compartilhamento de recursos?
- Um grupo de interessados formou um grupo de interesse de usuários (USING) para desenvolver os protocolos necessários:
  - Linguagem de comando comum
  - Editor de rede
  - Protocolo comum de cobrança (para usar os hosts)
  - FTP melhorado
  - Protocolo gráfico.
- Mas a ARPA não gostou muito de perder o controle e cortou o financiamento do grupo...

ARPANET: Lições Aprendidas

- Os protocolos de camada superior da ARPANET deram uma maior contribuição para a nossa compreensão do projeto de protocolos do que para a arquitetura das camadas superiores.
- Mas isto era compreensível, pois o desenvolvimento de aplicações distribuídas não era a motivação principal para o projeto.
- O uso da forma canônica (ex.: NVT) foi uma grande inovação teórica e prática. Permitiu a redução de um problema $O(n^2)$ para um problema $O(n)$ evitando a tradução para cada par de sistemas distintos.

ARPANET: Lições Aprendidas

- As MIBs *(Management Information Bases)* são formas canônicas atuais.
- A experiência da ARPANET mostrou que há vantagens em fazer errado da primeira vez: a nova versão do Telnet ficou muito melhor a partir de uma síntese de mecanismos conflitantes.
- Mas tem que ser muito errado, caso contrário são feitos apenas pequenos remendos, como foi o caso do FTP.

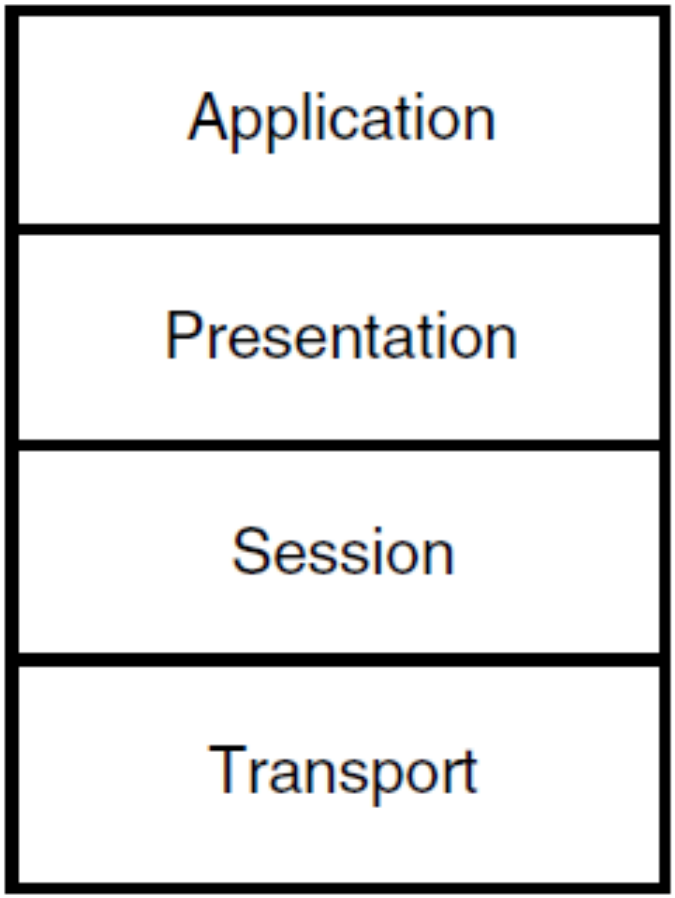
ARPANET: Lições Aprendidas

- A elegância não deve ser levada muito adiante:
  - RJE usando FTP e FTP e RJE usando Telnet levou a uma solução impraticável.

A tentativa do OSI

- Começando em 1978, OSI foi a primeira tentativa dos grupos de padronização de fazer algo rápido e o primeiro a aprender que seria difícil senão impossível chegar a um acordo com um conjunto de interesses tão diferentes.
- Adotaram 7 camadas a partir de uma arquitetura proposta pela Honeywell.

OSI: Arquitetura das Camadas Superiores



- Questões:
  - Quais partes dos protocolos existentes pertenceriam às camadas de sessão e apresentação?
  - Que outras funções pertenceriam a estas camadas?
- Não houve muito consenso e o desacordo correspondia a PTTs x indústria de computadores.

OSI: Pressão das PTTs

- Interesse em colocar como OSI dois produtos que já tinham:
  - Teletexto e
  - Videotexto.

OSI: Camada de Sessão

- Controle de diálogo e
- Primitivas de sincronização

- Ou seja, não tem nada a ver com o estabelecimento de sessões, login, segurança e funções associadas!

OSI: Camada de Aplicação

- A camada de aplicação não contém dados dos usuários.
- Ou melhor, as PDUs contêm apenas informações compreensíveis pelo protocolo que interpreta a PDU. Não contém dados a serem interpretados por uma camada acima...
- Foi emprestado o conceito de "esquema conceitual" do mundo de bancos de dados:
  - Para que duas aplicações troquem informações, precisam de "um esquema conceitual no seu universo de discurso".
  - Generalização do conceito de forma canônica desenvolvido para os protocolos da ARPANET.

OSI: Camadas de Aplicação e de Apresentação

- Se a camada de aplicação cuida da semântica (esquemas conceituais), então a camada de apresentação deve cuidar da sintaxe.
- A camada de apresentação deve apenas negociar a sintaxe utilizada pela aplicação.

OSI: Sintaxes

Programming Language		ASN.1
<integer> ::=INTEGER<identifier>;	**Language Definition**	GeneralizedTime ::= [Universal 24] Implicit VisibleString
INTEGER X;	**Statement Definition**	EventTime ::=Set { [0] IMPLICIT GeneralizedTime Optional [1] IMPLICIT LocalTime Optional}
(32-bit word)	**Encoding**	I | L | GeneralizedTime
$012A_{16}$	Value	0203000142_{16}

OSI: Protocolos de Aplicação

- FTAM – *File Transfer Access Method*
  - Baseado em protocolo Britânico
- JTM – *Job Transfer and Manipulation*
  - Baseado em protocolo Britânico
- VTP – *Virtual Transfer Protocol*
  - Baseado numa combinação de propostas da DEC e pesquisadores europeus

- Apesar de incluir mais funcionalidades que as versões anteriores da ARPANET, não trouxeram nenhuma inovação na arquitetura.

OSI: Novos desenvolvimentos

- CCR – *Commitment, concurrency, and recovery*
- Um protocolo de *commit* em duas fases como um componente reusável
- TP – *Transaction Processing*
- RDA – *Remote Database Access*
- RPC – *Remote Procedure Call*
- X.500 – Diretório
- Protocolo de Gerência: CMIP – *Common Management Information Protocol*

CCITT: Protocolo de Correio (X.400)

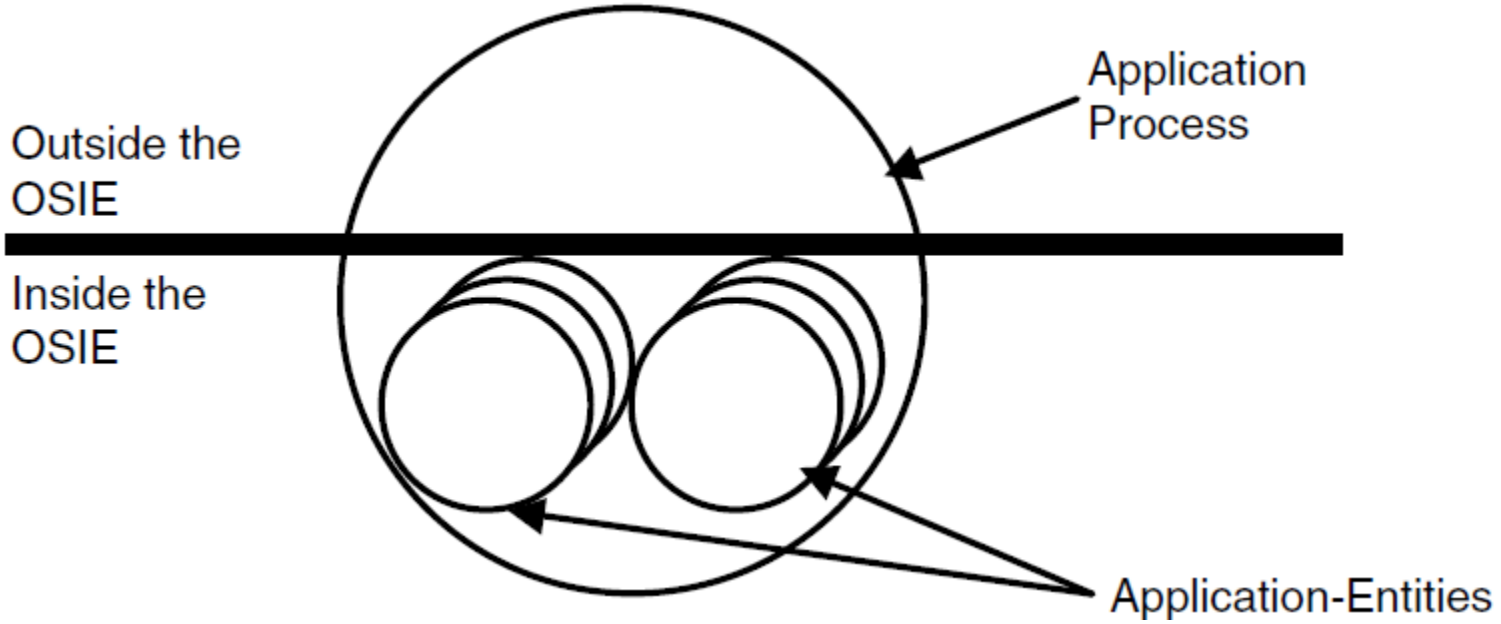
- Na ARPANET as mensagens eram trocadas diretamente entre os computadores.
- À época das discussões que levaram ao X.400, muitos sistemas eram estações de trabalho ou PCs.
  - Isto levou à distinção de servidores que recebiam mensagens em nome de seus usuários
  - Conceitos:
    - MTA – *Message Transfer Agents* e
    - Servidores de mensagens

OSI: Mecanismo Comum de Estabelecimento de Conexão

- ACSE – *Association Control Service Element*
- Provê mecanismos para:
  - Endereçamento na camada de aplicação
  - Negociação de contexto da aplicação
  - Autenticação
- As aplicações definidas pela OSI (CCR, TP, transferência de arquivos, etc.) essencialmente apenas definem o comportamento da fase de transferência de dados.
- O ACSE provê a fase de estabelecimento

OSI: Natureza do Processo de Aplicação

- Grande contribuição do OSI para as camadas superiores.
- Protocolos de aplicação (*application entities*) fazem parte do processo da aplicação, mas há uma parte que fora do ambiente OSI (OSIE).

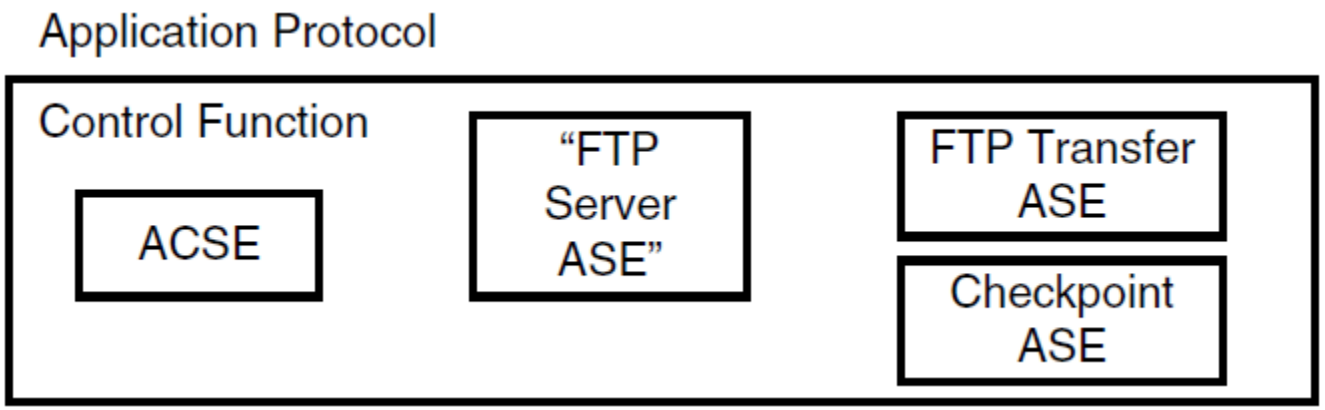


OSI: Natureza do Processo de Aplicação

- Na abordagem da ARPANET não havia praticamente nada do processo de aplicação (AP) fora das entidades de aplicação (AEs).
- Queremos acessar a cnn.com (AP) e não o HTTP (AE).
- OSI:
  - Distinção entre protocolos de aplicação da aplicação
  - Ao mesmo tempo reconhecendo que nomear o protocolo de aplicação requer nomear antes a aplicação.

OSI: Boa Engenharia de Software

- Abordagem OSI permitia que os protocolos de aplicação fossem construídos a partir de módulos reutilizáveis, os ASEs (*Application Service Elements*).
- Composição de um protocolo de aplicação:
  - ACSE e um ou mais ASEs ligados por uma função de controle (CF).



OSI: Problemas com a Estrutura

- Em 1983 já estava claro que cada sessão e conexão de apresentação suportavam uma única conexão de aplicação.
- Não havia multiplexação acima da camada de transporte e não havia necessidade de endereçamento nas camadas de sessão e apresentação.
- Portanto, não havia necessidade para que as camadas de sessão e apresentação estabelecessem conexões de forma serial por conta da sobrecarga.

OSI: Problemas com a Estrutura

- Estava claro que:
  - A implementação da fase de estabelecimento das camadas de sessão, apresentação e aplicação deveriam ser consideradas como uma única máquina de estados.
  - E que as unidades funcionais de sessão (i.e. as primitivas de controle e sincronização) deveriam ser vistas como bibliotecas a serem incluídas caso necessário.

OSI: Problemas mais fundamentais

- Se uma aplicação necessitasse mudar o contexto de apresentação durante o tempo de vida da conexão, ela informaria à camada de apresentação, que efetuaria a mudança renegociando-o.
- Diferentes partes de uma aplicação poderia requerer uma sintaxe diferente a ser usada em tempos diferentes numa conexão.
- Mas, como o protocolo de aplicação poderia invocar primitivas de sincronização de sessão para retornar a um ponto anterior do fluxo, a camada de apresentação deveria ter o registro do pedido do protocolo de aplicação para garantir que a sintaxe correta fosse usada para o fluxo de dados no ponto para o qual se retornou...
- As funções de sincronização deveriam estar acima da apresentação e não embaixo dela!

OSI: Problemas mais fundamentais

- Conflito no uso da camada de sessão:
  - O protocolo de processamento de transações (TP) usou CCR para um *commit* de duas fases, mas também fazia o seu próprio uso de primitivas de sessão necessárias, na mesma sessão de conexão.
  - A sessão não tinha meios de distinguir estes dois usos, e não havia garantias de não interferência entre elas.

OSI: Problemas mais fundamentais

- Repasse de mensagens X.400:
  - Conexões entre aplicações eram a origem e o destino dos dados.
  - Mas, o modelo de referência (RM) permitia explicitamente o repasse na camada de aplicação.
  - Portanto, enquanto a sintaxe do "envelope" tinha que ser entendido por todos os intermediários da camada de aplicação, a sintaxe da "carta" deveria se entendido apenas pelo remetente e pelo destinatário da carta.

OSI: Problemas mais fundamentais

- Repasse de mensagens X.400 (cont.):
  - A arquitetura exigia que o intermediário suportasse todas as sintaxes necessárias não só para o envelope, mas também para a *carta*, mesmo se apenas o remetente e o destinatário precisariam interpretar a sua sintaxe.
  - O SMTP evitou este problema por um acidente histórico, pois a única sintaxe era ASCII, e posteriormente bastou acrescentar o MIME *(Multipurpose Internet Mail Extensions)*

OSI: Lições Aprendidas

- O OSI fez grandes progressos em ampliar o nosso conhecimento das camadas superiores, mas sofreu com problemas causados por interesses conflitantes:
  - Interesses econômicos
  - Modelo “furado”.
- Funções da ACSE na camada de aplicação são conceitos associados com sessões, mas a camada de sessão foi “roubada” pelas PTTs.

OSI: Lições Aprendidas

- O fato de não existir multiplexação nas camadas de sessão e apresentação é um forte indício de que apesar de existirem funções de sessão e apresentação, elas não constituem camadas distintas.
- A negociação da sintaxe de apresentação deveria ser uma função do estabelecimento da conexão da aplicação.
- Precisamos de uma arquitetura de camada de aplicação que suporte a combinação de módulos em protocolos.

OSI: Lições Aprendidas

- No caso de repasses de mensagens (como as de correio) necessitamos de duas camadas para as conexões de aplicação de modo a separar:
  - A sintaxe fim-a-fim da carta e
  - A sintaxe etapa-a-etapa do envelope.
- Devemos esperar que as aplicações possam ser usadas como base para a construção de aplicações mais complexas.

OSI: Lições Aprendidas

- Reconhecimento dos papéis da sintaxe e da semântica nos protocolos de aplicação.
- Distinção entre sintaxes abstrata e concreta.
- Reconhecimento de que os contextos de aplicação e apresentação eram casos específicos de uma propriedade geral dos protocolos (política de negociação).
- Percepção de que o campo nos protocolos inferiores para identificar o protocolo acima era realmente uma forma degenerada de contexto da apresentação e não um elemento de endereçamento.

OSI: Lições Aprendidas

- O OSI ficou preso cedo demais numa estrutura particular de camadas.
- Se a ACSE tivesse sido a camada de sessão, o resultado poderia ainda não ser o mais correto, mas estaria perto o bastante do resto da solução.

Gerenciamento de Redes

- A ARPANET possuía excelentes capacidades de gerenciamento de redes, mas eram internas à BBN.
  - Toda a rede podia ser observada e depurada (na maioria das vezes) a partir do centro de gerenciamento em Cambridge, MA.
- Em 1984 teve início um grande esforço de desenvolvimento de gerenciamento de redes para automação de escritórios e de fábrica: MAP/TOP – GM *Manufacturing Automation Protocol/Technical and Office Protocols*).
- Protocolo de Gerência 802.1 em uso já em 1985 baseado apenas numa estrutura de pedido/resposta (set, get, operações de ação e um evento assíncrono).

Gerenciamento de Redes

- O protocolo do IEEE foi generalizado para o que se tornou o CMIP *(Common Management Information Protocol)* com as seguintes novas características:
  - MIBs orientadas a objetos
  - Inclusão dos conceitos de escopo e filtro.
- Pouco depois mas em paralelo, o IETF começou a trabalhar em gerência de redes.
  - Havia dois esforços: SNMP (*Simple Network Management Protocol)* equivalente ao IEEE 802) e o HEMS *(High-Level Entity Management System)* que era uma abordagem inovadora.
  - Por alguma razão inexplicável o IETF decidiu adotar o SNMP que é *simples* apenas no nome!

Crítica ao SNMP

- As implementações do CMIP e do HEMS são menores.
- O SNMP aderiu ao dogma de que "tudo deve ser sem conexão".
  - Isto limitou a quantidade de informação a ser recuperada complicando o acesso mesmo a pequenas tabelas.
- Limita a natureza de eventos não solicitados de modo que os dispositivos precisam ser consultados (*polled*).
- Com medo com que a padronização tornasse os roteadores intercambiáveis, alguns fabricantes se apressaram em dizer que SNMP era bom para monitoração mas não para controle por que a segurança era fraca.
  - Isto era verdade, mas a conexão de controle Telnet em ASCII com senhas enviadas sem criptografia também não era segura.
  - E todas as máquinas tinham Telnet enquanto pouquíssimas tinham um interpretador ASN.1

Proliferação de MIBs

- Poderiam ter sido criadas apenas uma MIB por camada.
- Mas, no IETF, houve uma proliferação de MIBs uma para cada tecnologia, protocolo e, em alguns casos, até tipos diferentes de dispositivos.
- A chave para simplificar redes deve ser criando algo mais comum e consistente entre as MIBs.
  - Mas isto não é de interesse dos fabricantes.
  - Deveria fazer parte da arquitetura.

O que a Gerência de Redes nos diz sobre as camadas superiores?

- As demais aplicações requerem um protocolo para cada aplicação.
- No gerenciamento de redes a variedade está nos modelos dos objetos.
- Não apenas os protocolos de gerência são aplicáveis a outros sistemas “tipo-rede” (rede elétrica, gás, distribuição de água, empresas aéreas) mas também diversas outras aplicações podem usar o mesmo protocolo básico orientado a objetos para realizar ações a distância.

Nova visão para os protocolos de aplicação

- Muitos protocolos de aplicação podem ser vistos como um número limitado de operações (protocolo) em uma grande variedade de modelos de objeto (operandos ou atributos).
  - Operadores: set/get, create/delete, start/stop, e evento
  - Protocolos na forma:
    - Request/response
    - Notify/confirm (pode ser simétrico)
- Portanto, parece que Alex estava bem próximo do número de protocolos de aplicação; só que eles não eram Telnet e FTP!

Problema de operações demasiado elementares

- Infelizmente não exploramos o que viria com o uso do HEMS, pois poderia ser um meio termo entre os protocolos com estrutura elementares (do tipo máquina de Turing) e as linguagens com uma estrutura de controle completa, como Java, Perl ou XML.
- Esta é uma área que necessita uma maior exploração!

HTTP e a Web

- O protocolo de aplicação que teve o maior impacto na Internet veio de uma outra comunidade!
- A Web foi um sucesso tão grande que para a maior parte das pessoas ela é sinônimo da Internet.
- Novas estruturas e requisitos para a Internet:
  - Necessidade de distinguir a aplicação do protocolo de aplicação
  - Os efeitos de muitas conexões de transporte curtas no tráfego da rede
  - (relacionada à primeira) Novos requisitos para endereçamento provindos da necessidade de lidar com a carga gerada por muitos usuários acessando as mesmas páginas.

Funcionamento da Web

- Digita uma URL *(Universal Resource Locator)* em um cliente HTTP (ou navegador)
- O DNS é usado para traduzir a URL em um endereço IP
- A URL (solicitação) é então enviada usando o protocolo HTTP através do TCP para o servidor HTTP.
- A resposta contém uma página com texto formatado e gráficos e contendo muitas URLs para outros objetos na página.
- Todas estas URLs podem apontar para informações em sistemas diferentes.
- O navegador então acessa cada URL através de uma solicitação sobre uma nova conexão TCP
- Cada resposta provê mais objetos para popular a página Web, assim como URLs, que podem se selecionadas para acessar novas páginas.

A Web

- Um usuário Web acessa uma aplicação Web.
- Podem haver milhares de sites Web no mesmo host todos usando HTTP.
- HTTP é apenas o veículo para que duas partes da aplicação se comuniquem.
  - Primeiro exemplo na Internet da distinção entre AP e AE que o OSI achou necessária.

Falta de um espaço de nomes para a Internet

- Foi preciso criar as URLs
- Parte da URL contém o nome do domínio do *host*.
- Mas, e se quisermos que o servidor Web seja mudado para um outro servidor?
  - Redirecionamento: funciona mas traz uma carga desnecessária e causa atrasos para o usuário.
  - As URLs deveriam ser suportadas diretamente pelo DNS, mas o DNS não foi projetado para este tipo de uso.

Grande Número de Conexões TCP e Volume de Dados

- TCP foi projetado baseado nas hipóteses de que:
  - As conexões seriam curtas (poucos minutos ou segundos) ou muito longas (algumas horas)
  - Os hosts estabeleceriam novas conexões numa média de algumas centenas de conexões por hora.
- Não tinham sido previstas conexões que durassem milissegundos a uma taxa de milhares por minuto!
- O HTTP abre muitas conexões curtas e envia apenas alguns bytes de dados, embora o conteúdo total de dados para uma página seja equivalente a uma transferência de dados de tamanho razoável.

Problema dos Elefantes e dos Ratos

- Cada conexão TCP nunca envia dados suficientes para ser submetido ao controle de congestionamento do TCP, ao contrário de outros tráfegos com conexões mais duradouras.
- Isto faz com que o tráfego HTTP capture uma porção injusta da largura de banda.
- O uso de conexões persistentes (HTTP 1.1) aliviou este problema.

Uso de Caches

- Dado que as páginas Web seguem a lei de Zipf, as mesmas páginas são recuperadas por muitos usuários.
- Isto motiva o uso de caches Web no cliente, na subrede do cliente, no ISP ou por um serviço de hospedagem Web.
- O cliente HTTP é modificado para enviar todos os pedidos para um site de cache independentemente do que está indicado na URL.
  - Caso não encontre o conteúdo desejado será redirecionado para uma outra cache ou para o site indicado pela URL.

Web: Qual a lição?

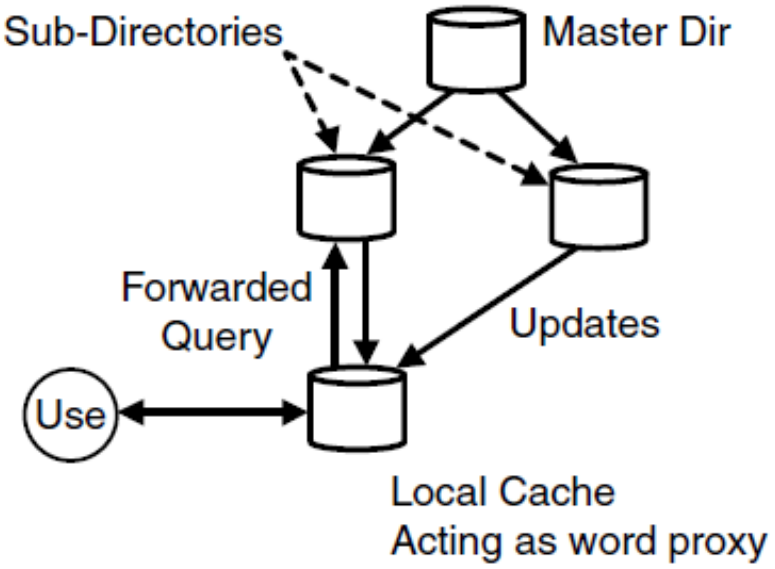
- No caso da Web as estruturas existentes foram remendadas para atender as suas necessidades.
- É preciso encontrar uma ajuda (*band-aid*) para a próxima aplicação?
- Quantas aplicações não estão sendo desenvolvidas porque não há estruturas para suportá-las?
- Quando estes remendos vão interferir uns com os outros?
- Qual é a forma correta de acomodar o balanceamento de carga e migrar aplicações na rede?
- Todos estes problemas encontraremos com outras aplicações.
- Eles foram resolvidos para a Web, mas não os resolvemos para toda a Internet.
- Ou como a imprensa acredita, a Internet seria a Web?

Diretórios ou Protocolos de Resolução de Nomes

- Outra classe interessante de aplicações distribuídas.
- Para a construção de uma rede de compartilhamento de recursos necessitamos um meio para encontrar os recursos!
- Dado que os SOs eram a metáfora do início, a solução era algum tipo de "diretório": um serviço que diria ao usuário onde estão as coisas.
- Exemplos:
  - Sistema Grapevine XNS (final dos anos 70)
  - Estendido em 1983
  - Outros: DNS, X.500, Napster, Gnutella, DHT (*Distributed Hash Table*)
- Todas com os mesmos elementos básicos e a mesma estrutura com pequenas variações.

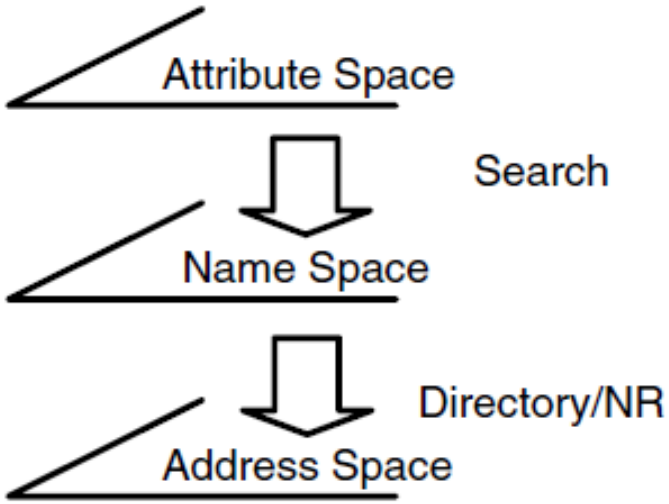
Sistema de Resolução de Nomes

- Um NRS (*Name Resolution System*) consiste de uma base de dados, normalmente distribuída, que é consultada pelo usuário.



Sistema de Resolução de Nomes

- A base de dados mantém o mapeamento entre dois espaços de nomes: um que nomeia o que deve ser localizado e um outro que nomeia onde o objeto se encontra.
- O quê = “nomes”
- Onde = “endereços”



Estruturas de um NRS

- O banco de dados para um NRS pode ser: centralizado, *cached*, hierárquico ou uma combinação do *cached* e do hierárquico (normalmente dependendo do tamanho e do grau de atualização desejado).
  - Centralizado: bases de dados pequenas ou não críticas
  - *Cached*: cache local para melhorar o desempenho e reduzir a sobrecarga ao preço de uma replicação parcial.
  - Grandes bases: explora-se a estrutura hierárquica do espaço de nomes para criar sub-bases de dados responsáveis por subconjuntos da base de dados.

NRS: Protocolos

- Protocolo de consulta/resposta
- Um protocolo para distribuir as atualizações da base de dados, se for distribuída.

Protocolo de Consulta

- O usuário envia uma consulta a um membro do NRS.
- Se o membro puder satisfazer a consulta, ele retorna uma resposta para o usuário.
- Caso contrário um NRS pode ser designado para responder ao usuário passando para um outro banco de dados no sistema ou encaminhando a consulta a outra base de dados do NRS
- A depender do nível de confiabilidade, a base de dados que encaminha a solicitação pode agir como um intermediário *(proxy)* para o usuário, ou a base de dados para a qual foi repassada a consulta pode responder diretamente ao usuário.

Atualização das Cópias Replicadas

- A atualização pode ser iniciada:
  - pela cópia (forma de solicitação)
  - pelo membro quando ocorre uma mudança (forma de notificação)
- A frequência das atualizações variam dependendo:
  - Da frequência de mudanças e
  - Do grau de atualidade necessário.

Características dos NRSs

- Organização da Base de Dados:
  - Centralizada
  - Hierárquica
  - Cache
  - Cache/Hierarquia
- Consulta:
  - Referência
  - Encaminhamento
  - Intermediário (*Proxy*)
- Atualização
  - Periódica
  - Disparada por eventos
  - Combinação

Atualizações

- Quando um novo site entre no ar:
  - Registra sua informação com um serviço local ou quase local
  - Propaga a partir de lá na medida do necessário
- Quando um site sai do ar ou um recurso muda de local:
  - Muda o conteúdo da base de dados
  - Estas mudanças devem ser propagadas.
- Isto é feito das seguintes formas:
  - Por um serviço filho que solicita uma atualização periódica ou
  - Por uma base de dados afetada que avisa aos seus vizinhos das mudanças.

Exemplos

- DNS, X.500 ou Grapevine são estruturadas como descrito, escolhendo políticas específicas para consultar, organização da base de dados e atualizações.
- Napster (centralizado) e Gnutella *(cached)* basicamente fazem a mesma coisa para encontrar arquivos ao invés de aplicações ou *hosts*.

DHTs (*Distributed Hash Tables*)

- Esta abordagem difere da anterior apenas em como é gerado o espaço hierárquico de nomes das aplicações.
- Com DHTs, é aplicada uma função de *hash* ao nome do recurso, normalmente uma URL.
- Isto pode ser interessante em termos de balanceamento de carga, mas destrói qualquer localidade que podia estar embutida no nome original e permitiria aos sites fazer algum esquema de cache inteligente.

O que distingue as Camadas Superiores

Distinção das Camadas Superiores

- Sempre foi difícil encontrar um conjunto de características que fossem melhor do que: “aquilo que está acima da camada de transporte”.

Primeira Característica

- Nas camadas superiores o processamento ocorre em unidades que possuem significado semântico para a aplicação (provocam a menor sobrecarga/esforço de processamento para a aplicação).
  - Nas camadas intermediárias o processamento é realizado em unidades mais adequadas aos requisitos de alocação de recursos; e
  - Nas camadas inferiores, dominam as características do meio de comunicação ou da tecnologia de rede.

Segunda Característica

- Nas camadas superiores o endereçamento é independente da localização.
- Ou melhor, enquanto que o endereçamento das camadas inferiores são baseados na topologia da rede, o endereçamento das camada superiores é normalmente baseado numa espécie de topologia "semântica"
  - Por exemplo, todas as aplicações de acesso à rede, todas as aplicações de desenvolvimento de software, etc.

Significado Semântico

- Nas camadas inferiores os limites da mensagem ou PDU são escolhidos para acomodar as restrições do meio ou da tecnologia de rede.
  - Os requisitos das aplicações são raramente percebidos.
- Isto muda nas camada superiores onde os limites do "registro" ou "transação" da aplicação se tornam importantes.

Independência de Localização

- Desde 1972 se havia reconhecido que seria altamente desejável que as aplicações pudessem migrar de *host* para *host*.
- E, para acomodar esta migração seria necessário que as aplicações fossem nomeadas de modo que os seus nomes fossem independentes de sua localização (i.e., do *host* no qual se encontram).

Endereços e Topologias

- Apesar de um endereço ser um nome, um nome não é necessariamente um endereço.
- Atribuímos endereços a objetos de modo que seja fácil encontrá-los.
- Os algoritmos para atribuir endereços a objetos definem uma topologia (em muitos casos, uma topologia métrica).
- Portanto, endereços sempre representam pontos numa topologia, enquanto que os nomes são apenas rótulos.

Endereços e Localização

- Nas camadas inferiores são usadas características geográficas ou de topologia da rede como princípio para localizar objetos.
- Nas camadas superiores são usadas outras características para localizar aplicações que são raramente relacionadas com a localização.
- Infelizmente, ao contrário das camadas inferiores, ainda não emergiu um conjunto de características claras para organizar o espaço de endereços e respectivos esquemas para as aplicações.

Endereços de Aplicações

- Os endereços de aplicações são organizados mais pelo “que” ou “quem” do que pelo “onde”.
- Em alguns casos houve um retorno a características dependentes de localização. Mas, estes esquemas excluem a possibilidade da migração transparente ao usuário da migração das aplicações.