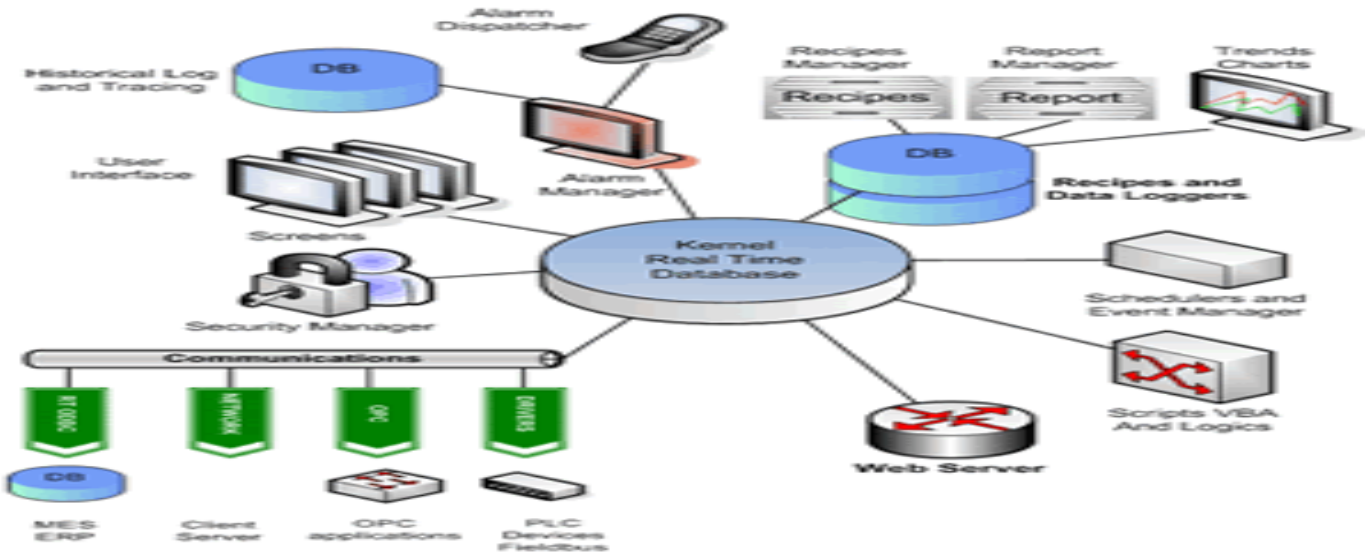
Modelos de Sistemas



MSc. Eugénio Alberto Macumbe

Modelos de Sistemas

- Um modelo de arquitectura de SD está preocupado com a localização das partes e o relacionamento entre elas. Os exemplos incluem os modelos de processos cliente/servidor e processos ponto a ponto.

- Modelos fundamentais estão preocupados com a descrição mais formal das propriedades que são comuns à todas as arquitecturas. Iremos discutir três modelos fundamentais:

  - **O modelo de interacção**, negocia com o desempenho e a dificuldade de ajuste de relógio limitado dos SD. Ex: Entrega de msg;
  - **O modelo de falhas**, dá uma especificação precisa de falhas que podem ser exibidas pelo processo e canais de comunicações;
  - **O modelo de segurança**, discute as possíveis ameaças para os processo e canais de comunicações. É introduzido o conceito de canal seguro.

Modelos de Sistemas

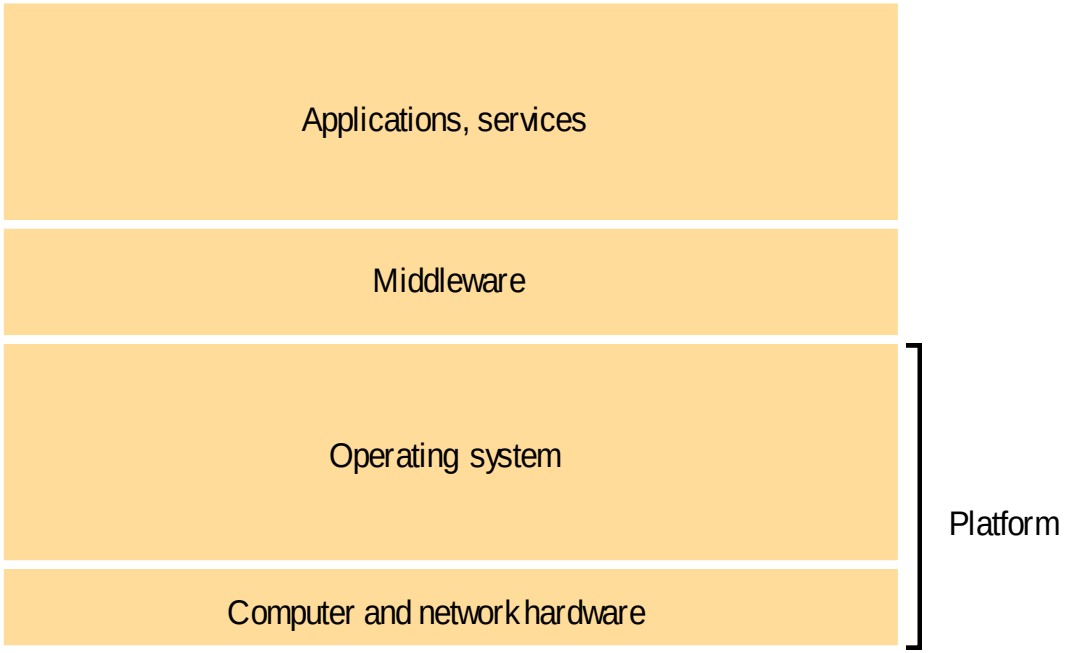
- Sistemas que têm a intenção de serem usados no mundo real devem ser projetados para funcionar corretamente sob uma grande variedade de circunstâncias, problemas e ameaças possíveis.

- Neste capítulo, estudaremos os modelos e suas principais propriedades.

- Os Sistema de arquitectura são estruturados em termos de componentes especificados separadamente. Sempre tendo em vista a preocupação de garantir, confiabilidade, desempenho, segurança, gerenciamento a um custo razoável.

Modelos de Sistemas

- **Camadas de Software**
  - arquitectura de Software, era o termo utilizado para estrutura de camadas ou módulos de software para um único computador. Actualmente é utilizado para definir serviços oferecidos e requisitados entre processos localizados em um mesmo ou diferentes computadores.



- **Camadas de Software**

  - **Conceito de Plataforma**, é o termo utilizado para fazer referência de camadas de baixo nível de hardware e software que dão serviços a camadas superiores.

  - Normalmente, em termos de computadores, uma plataforma está relacionada ao tipo de arquitectura de processador e o sistema operacional utilizado. Ex: Intel/Windows, Intel/Linux, Sun/Sparc/SunOS, Intel/Solaris, PowerPC/MacOS, etc.

  - MIDDLEWARE, pode ser representado por processos ou objectos em um conjunto de computadores que interagem entre si para providenciar suporte para comunicação e compartilhamento de recursos para as aplicações distribuídas.

- **Camadas de software**

  – Em outras palavras, um Middleware em SD se preocupa em oferecer serviços de forma eficiente de comunicação e de compartilhamento de recursos, simplificando seu uso através da abstração da plataforma de baixo nível, oferecendo um modelo de comunicação em grupo, notificação de eventos, a replicação de dados compartilhados e transmissão de dados multimídia e tempo real.

  – RPC - *Remote Procedure Calling*, ou chamada de procedimentos remotos é o termo utilizado para definir o método utilizado por determinado Middleware para fazer chamadas de serviços oferecidos pelos processos ou objectos.

- **Camadas de software**

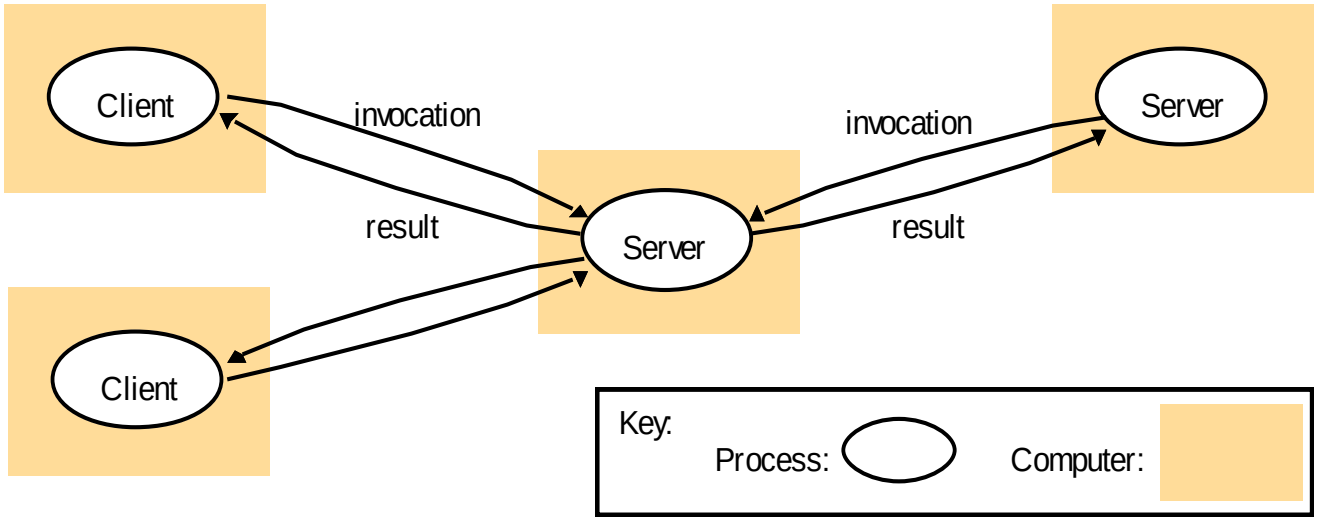
  – Um dos mais conhecidos Middleware que providencia serviços de acesso orientado a objectos é o CORBA - *Common Object Request Broker Archicteture.*

  – Outros:
    - Java-RMI, *Java Remote Method Invocation;*
    - DCOM, *Microsoft's Distributed Component Object Model e*
    - RMODP (ISSO/ITU-T's), *Reference Model for Open Distributed Processing.*

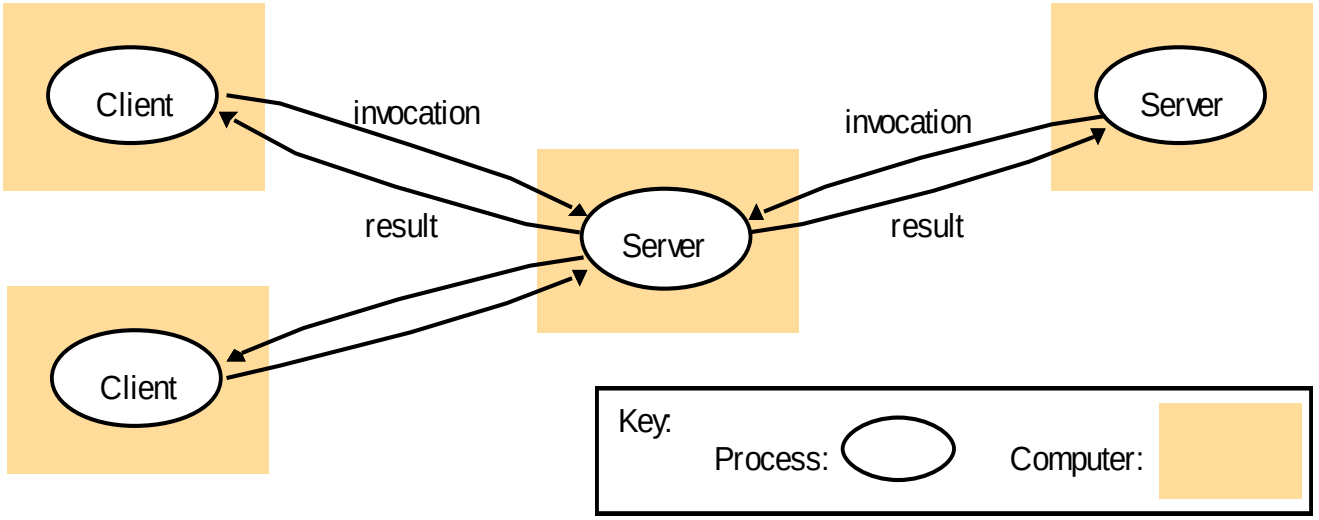
- **Camadas de software**

  – Problemas com o uso de Middleware:
    - Embora os Middleware tentem sempre oferecer transparência para os programadores, nem sempre isso é bom.
    - Algumas situações podem exigir que os problemas de comunicação, por exemplo, devam ser tratados no nível da aplicação. Ex: E-mail.
    - Outra situação é os Middleware o intuito de garantir confiabilidade, traz menos desempenho numa comunicação de rede. Ex: UDP vs TCP.

- **arquitectura de sistemas**
  - **Modelo Cliente-Servidor:**
    - É a arquitectura mais comum e frequentemente utilizada como exemplo de SD;
    - Os processos clientes invocam serviços aos servidores. Neste modelo, um servidor pode se tornar um cliente para solicitar serviços de outro servidor.



- **arquitectura de sistemas**
  - Ex:
    - Um servidor web pode se tornar um cliente para o sistema de arquivos do S.O, para os servidores DNS;

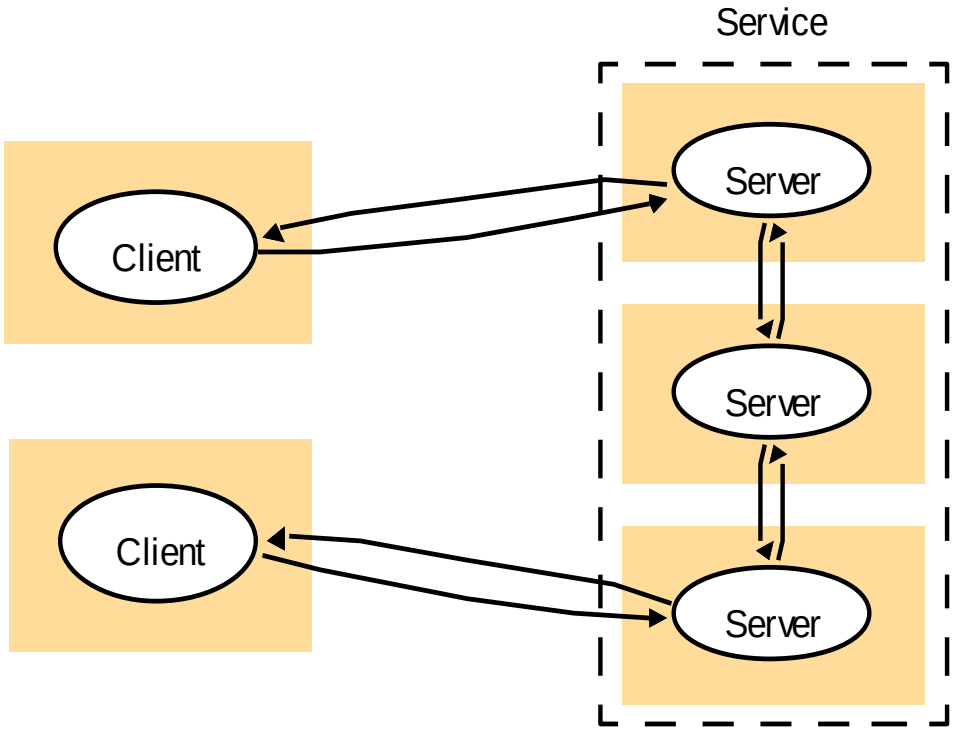


- **arquitectura de sistemas**

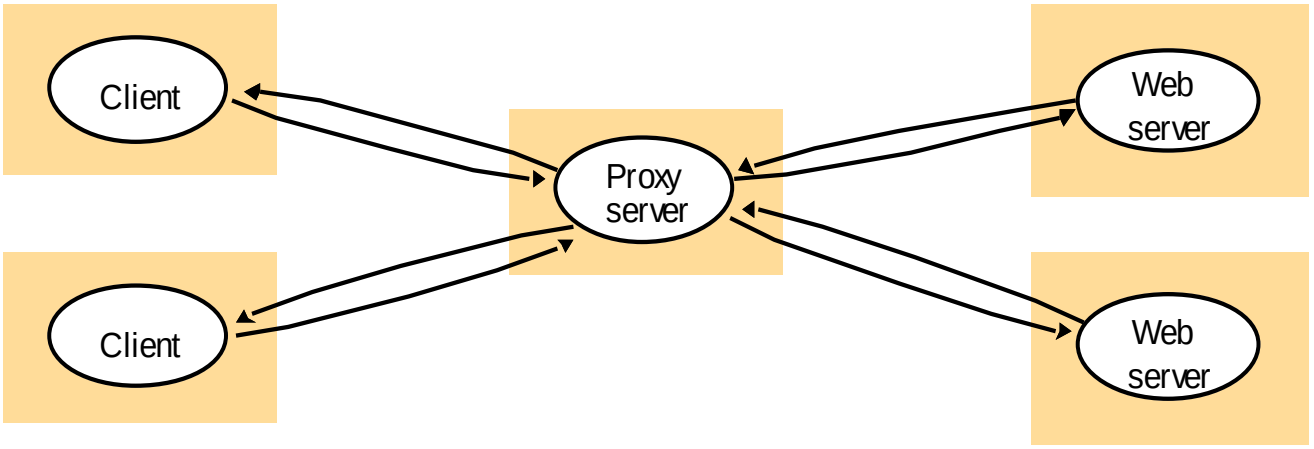
  - **Serviços providenciados por Múltiplos Servidores:**

    - Nesta categoria, os serviços são implementados em vários servidores que interagem entre si para oferecerem os serviços solicitados pelos processos clientes.

    - Ex: Servidores Web, BDs, DNS e etc.



- **arquitectura de sistemas**
  - **Servidores Proxy e de Caches**
    - A técnica de cache é recente, e é utilizada em servidores que mantêm cópias de dados solicitados anteriormente. Quando um cliente faz uma solicitação a um proxy, primeiro ele verifica se os dados estão presente localmente, caso contrário ele busca a informação efetivamente na rede.
    - Ex: Servidores de Proxy Web.



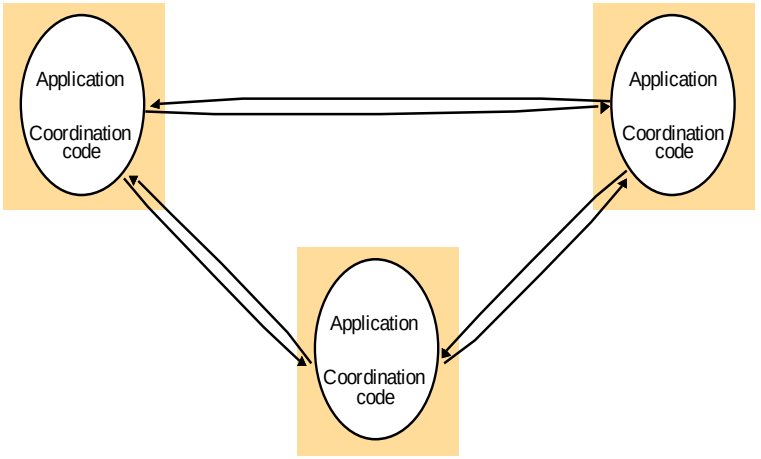
- **arquitectura de sistemas**

  – **Servidores Proxy e de Caches**
    - A técnica de cache permite criar servidores Proxy Web, o que permite que dados em cache possam ser compartilhados para muitos clientes;
    - a técnica de cache pode ser inserida nos clientes, de modo que possa beneficiar as várias aplicações no acesso aos dados.

  – O propósito geral é garantir desempenho e disponibilidade de dados sem aumentar a carga ou tráfego na rede utilizada.

- **arquitectura de sistemas**

  - **Processo Ponto a Ponto**
    - Nesta arquitectura todos os processos interagem com regras semelhantes. Trabalham de forma cooperativa para desempenhar actividades computacionais. Não há processo clientes ou servidores. Cada processo mantêm controle de seus recursos e o modo de interacção com outros processos.
    - Ex: games.



- **Variações no modelo Cliente Servidor**

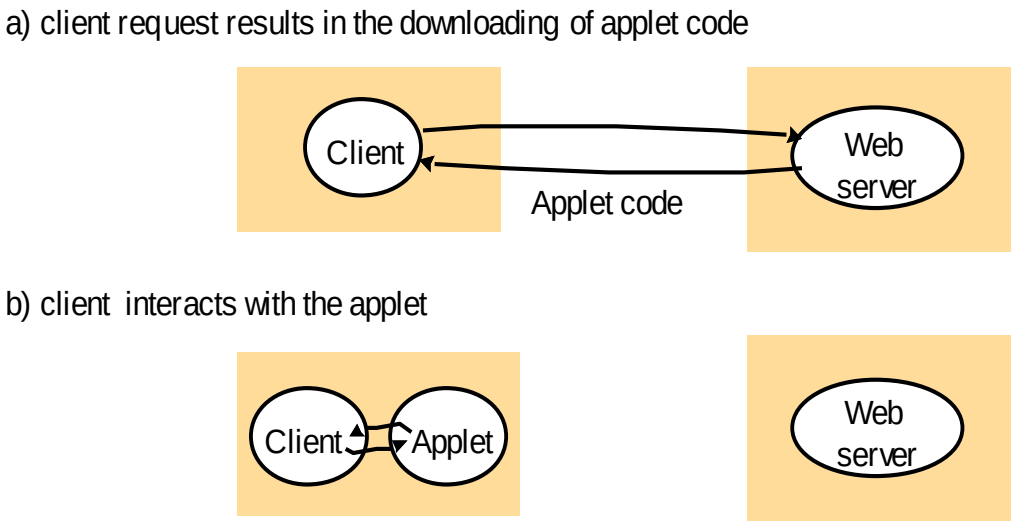
  – Algumas variações deste modelo decorrem de razões listadas abaixo:

    - Uso de Códigos Móveis e Agentes Móveis;
    - Possibilidade de baixar cursos de recursos de hardware;
    - Flexibilidade para adicionar e remover dispositivos móveis.

- **Variações no modelo Cliente Servidor**

  – **Códigos Móveis**
    - Os applets são bom exemplo de códigos móveis. Seu funcionamento é baseado de modo a permitir que os códigos sejam transferidos do servidor Web para os browsers, onde são executados.

  – Vantagens:
    - Sem delay;
    - sem tráfego;



- **Variações no modelo Cliente Servidor**

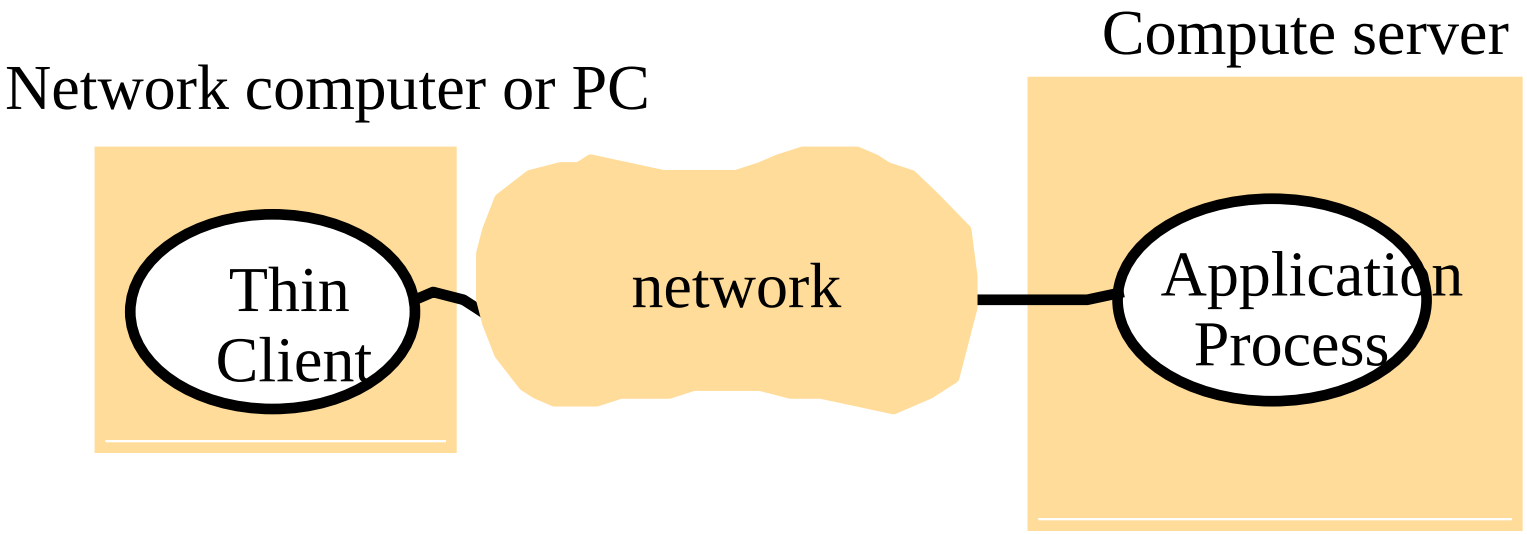
  - Através de códigos móveis é possível oferecer serviços que não podem ser dado normalmente pela Web.
    - Ex:
      - Serviços de mensagens do tipo Push;
      - Actualização do relógio por parte do servidor;
      - interacção automática com outros servidores.

- **Variações no modelo Cliente Servidor**
  - Agentes Móveis
    - São programas executáveis (códigos e dados) que trafegam de um computador ao outro para executar alguma tarefa.
      - Ex: Aplicações que necessitem colectar diversas informações em muitas máquinas. A vantagem é a redução do tráfego de rede, devido a redução de número de chamadas remotas por parte de códigos fixos;
      - Deve ter alta preocupação em relação a segurança. O Ambiente deve preocupar em como e quais recursos ele poderá dar aos códigos móveis. Por outro lado, os agentes também podem sofrer problemas e não completarem duas tarefas, pois não conseguiram ter acesso autorizado aos recursos necessários.

- **Variações no modelo Cliente Servidor**

  – Rede de Computadores
    - "Network Computers" tem arquitectura baseada de modo que o sistema operacional, aplicativos e dados são sempre armazenados em servidores, porém sua execução é realizados nos computadores clientes.
    - A premissa é que o gerenciamento dos recursos é feita sempre pelos servidores, trazendo como vantagem a facilidade de utilização desses computadores por parte dos usuários, pois desta forma eles não tem que se preocupar com espaço em disco, actualização, instalação e remoção de aplicativos.
    - Outra vantagem é que os computadores de rede tem preço muito baixo, pois eles não precisam manter grande espaço em disco.

- **Variações no modelo Cliente Servidor**
  - Thin Clients
    - O termo “thin Client” é o termo utilizado para fazer referência a uma arquitectura que providencia através de camada de software, suporte a uma interface gráfica aos aplicativos que estão sendo executados em um Servidor Remoto.

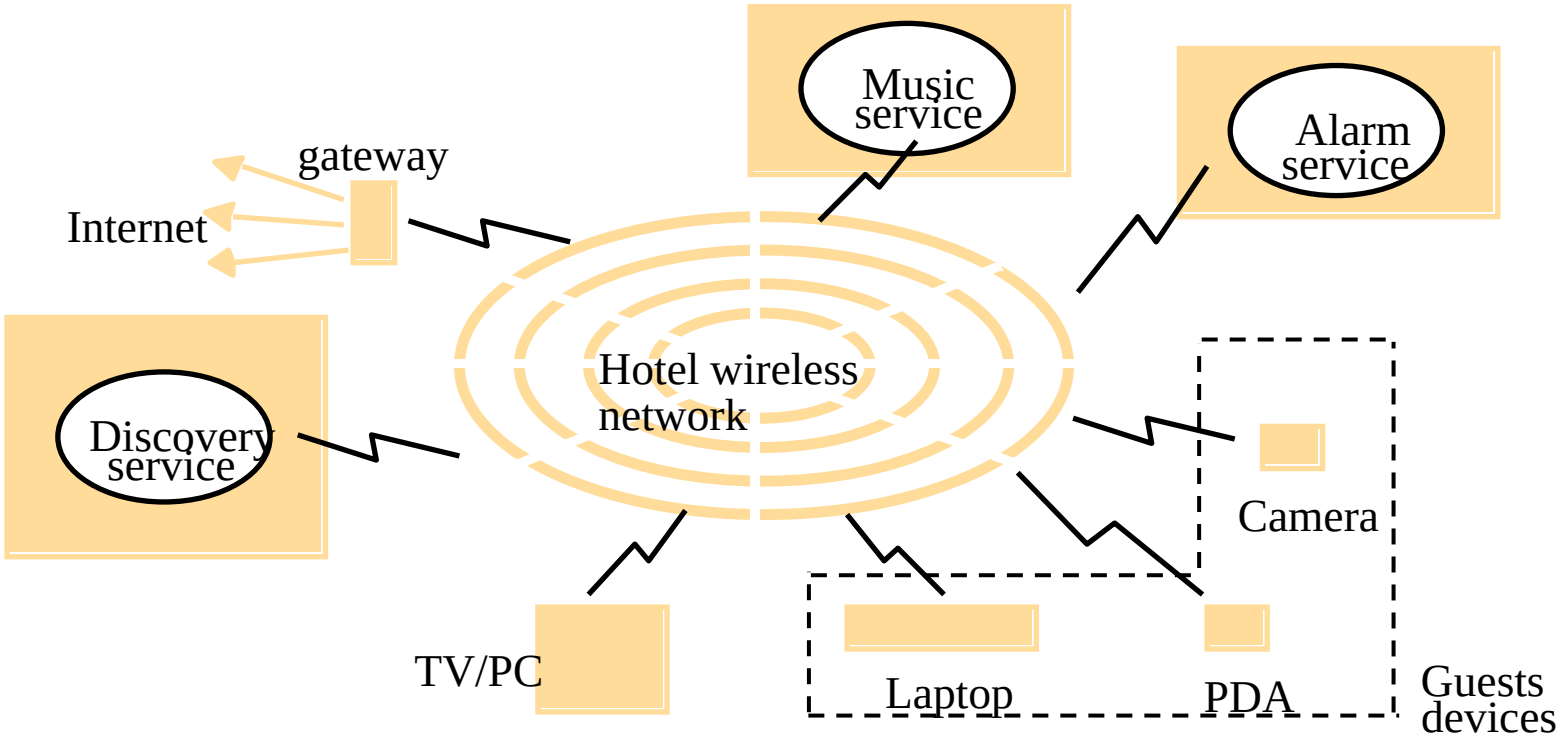


- **Variações no modelo Cliente Servidor**
  - Thin Clients
    - Características:
      - Os servidores devem ser poderosos, capazes de executar diversos processos em paralelo para atender inúmeros clientes;
      - Diferente da arquitectura de computadores de rede, os aplicativos são todos executados nos servidores;
      - Sua principal desvantagem é quando se utiliza aplicativos gráficos. Ex: CADs e processamento de imagens;
      - Exemplo de sistema: X11 (UNIX).

- **Variações no modelo Cliente Servidor**
  - Conexão espontânea
    - Trata de um conceito importante para utilização de dispositivos móveis. Ela se preocupa na forma como devem ser integrados os diversos dispositivos móveis com outros dispositivos não móveis com a infra-estrutura de rede disponível.

    - A idéia básica é oferecer facilidade de conexão de dispositivos para os usuários, de modo que eles não tenham que configurar ou instalar novos softwares para acessar um serviço.

- **Variações no modelo Cliente Servidor**
  - Conexão espontânea



- **Variações no modelo Cliente Servidor**

  - Exemplo de Cenário para Conexão espontânea:
    - Providenciar facilidades no acesso de Serviços, tais como:
      - Serviços de Música;
      - Sistema de alarme;
      - Acesso a Web, via TV/PC;
      - Serviço de Impressão de Fotos via TV ou Impressora;
      - Controle Remoto via PDA, para controlar o ambiente;
      - Fax;
      - etc.

- **Variações no modelo Cliente Servidor**

  – **Desafios para a Conexão espontânea:**

    - **Facilidade para conexão para Rede Local**, deve permitir a capacidade de não ficar limitado a uma localidade;
    - **Facilidade para integração e acesso aos serviços Locais,** deve ter a capacidade de descobrir automaticamente quais serviços um determinado ambiente possui;
    - **Conectividade Limitada,** deve ter a capacidade de continuar trabalhando mesmo que a conexão seja interrompida momentaneamente;
    - **Segurança e Privacidade**, deve evitar que o ambiente e o usuário sejam atacados ou tenham sua privacidade invadida de alguma maneira.