

# SISTEMAS DISTRIBUÍDOS

## Introdução

Prof. Guilherme C. Kurtz



# Conceitos básicos

- ▶ O que é um Sistema Distribuído?
  - “Um sistema distribuído é um **conjunto de computadores independentes** entre si que se **apresenta** a seus usuários como um **sistema único e coerente**” (Tanenbaum);
  - “Um sistema em que **componentes de hardware e softwares** localizados em computadores em rede se **comunicam e coordenam** suas **ações** por **troca de mensagens**” (Coulouris et al).


# Conceitos básicos

- ▶ O que é um Sistema Distribuído?
  - “Você sabe que existe um sistema distribuído quando a falha de um computador que você nunca ouviu falar impede que você faça qualquer trabalho” (Leslie Lamport);


# Conceitos básicos

- ▶ O que é um Sistema Distribuído?
  - Um sistema distribuído é um conjunto de **componentes hardware e software, interligados através de uma infra-estrutura de comunicações, que cooperam e se coordenam entre si apenas pela troca de mensagens, para a execução de aplicações distribuídas.**

# Aplicações

- ▶ Difusão de informações pela internet;
  - ▶ Comunicação:
    - E-Mail, Messengers, Jogos, etc.;
  - ▶ Controle de processos industriais e equipamentos:
    - Controle/domínio/fiscalização exercida sobre atividades de pessoas ou produtos, para que os mesmos não se desviem de normas pré-estabelecidas;
    - Aumento da **produtividade, flexibilidade, qualidade e confiabilidade.**
  - ▶ Automação de processos dentro de empresas e entre empresas e órgãos públicos;
  - ▶ Sistemas financeiros e bancários:
    - Transações entre bancos, caixas eletrônicos, etc.
- 


# Porque utilizar SDs?

- ▶ Compartilhamento de recursos:
    - Impressoras, arquivos, fax, outros periféricos.
  - ▶ Distribuição da carga;
  - ▶ Tolerância a falhas:
    - Maior disponibilidade de serviços e recursos;
  - ▶ Flexibilidade e Adaptabilidade:
    - Decompor um sistema complexo em vários subsistemas mais simples;
  - ▶ Acesso a serviços sem restrição de localização;
- 

# Características

- ▶ Execução concorrente de componentes:
  - Atendendo requisições e realizando processamento;
  - Necessidade de coordenação;
  - Memória compartilhada: exclusão mútua, semáforos e monitores;
  - Troca de mensagens: mais simples que memória compartilhada;
- ▶ Paralelismo
- ▶ Falhas independentes dos componentes de comunicação:
  - A falha de um componente deve ser isolada e não deve afetar o sistema inteiro;
- ▶ Ausência de um relógio global:
  - Envolve a sincronização dos processos.

# Desafios

- ▶ Heterogeneidade
  - ▶ Abertura
  - ▶ Escalabilidade
  - ▶ Segurança
  - ▶ Tratamento de falhas
  - ▶ Concorrência
  - ▶ Transparência
- 

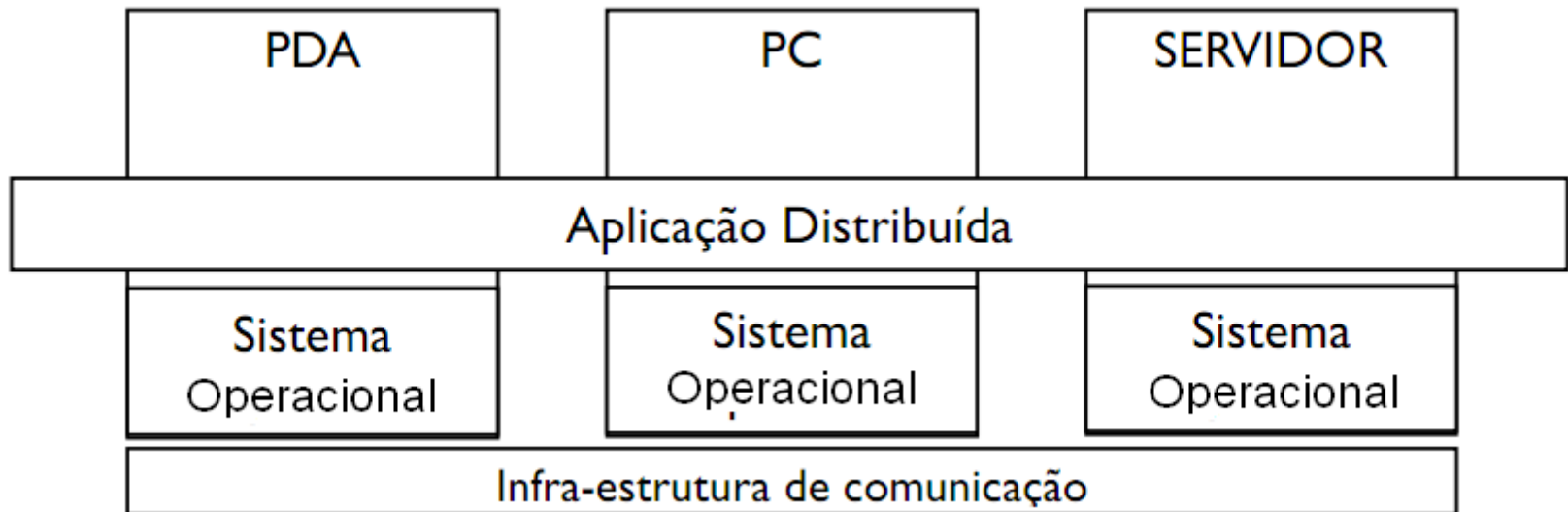


# Heterogeneidade

- ▶ Em Sistemas Distribuídos, tudo pode ser heterogêneo:
  - Redes:
    - Redes móveis (GSM, UMTS, CDMA, 3G), WiMAX, WLANs, wired-LANs, WANs, TCP/IP, etc.
  - Hardware dos computadores:
    - Celulares, PDAs, notebooks, desktops, servidores, clusters, etc.
  - Sistemas Operacionais:
    - Windows, Linux, Mac, Symbian, Android, etc.
  - Linguagens de programação;
  - Diferentes implementações por diferentes programadores ou diferentes versões;
- ▶ Soluções para heterogeneidade:
  - **Middleware**
  - Máquinas virtuais (JVM)

# Heterogeneidade

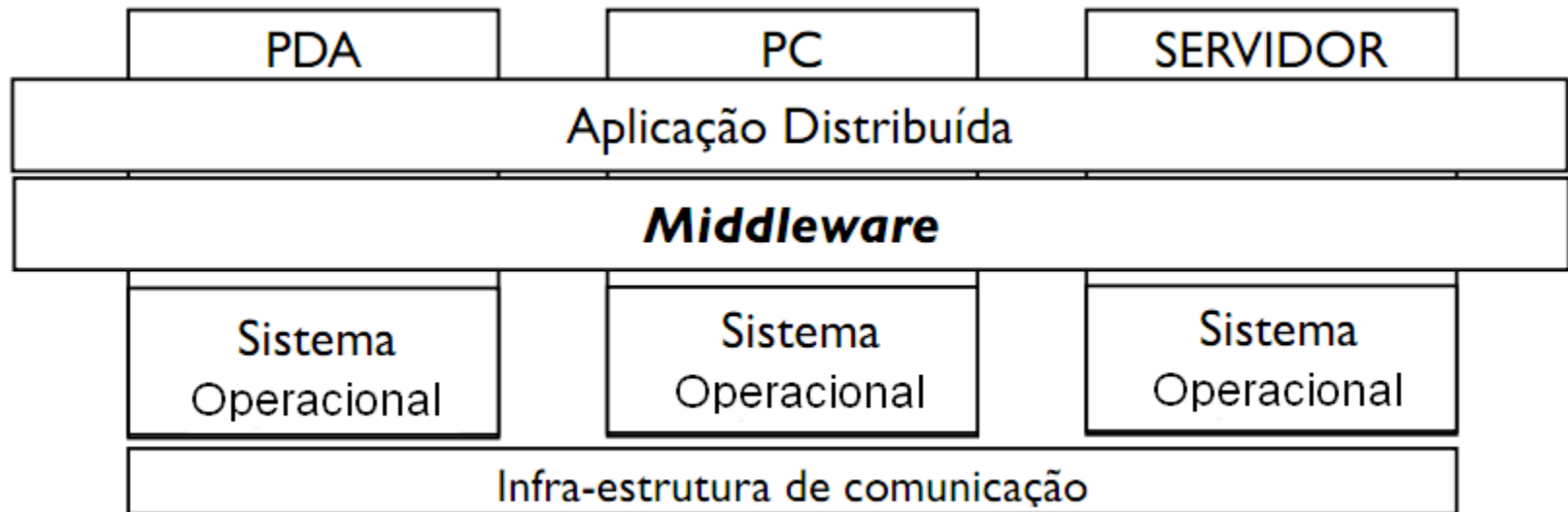
- ▶ Solução sem um **middleware**:



- ▶ Um serviço distribuído diretamente sobre os serviços/recursos do sistema operacional irá se deparar com interfaces heterogêneas.

# Heterogeneidade

- ▶ Solução com um **middleware**:



- ▶ O middleware irá fornecer interfaces homogêneas, **simplificando** o desenvolvimento de aplicações e melhorando a **interoperabilidade**.

# Heterogeneidade

- ▶ Solução com máquinas virtuais:
  - A utilização de máquinas virtuais possibilita a **execução de um mesmo programa em qualquer máquina**;
    - Ex: Máquina Virtual Java;
  - A execução **independe** do hardware ou do sistema operacional;
  - Suporte a uma única linguagem (JavaVM) ou múltiplas linguagens (Microsoft CLR com .NET, C#, etc.);
  - **Simplifica a implantação** dos sistemas distribuídos, a **portabilidade** e a **migração** de códigos;


# Abertura

- ▶ Determina se o **sistema** pode **ser estendido** ou **ser reimplementado** de diversas maneiras;
- ▶ Um sistema distribuído **aberto** é um sistema que oferece serviços de acordo com **regras padronizadas** que descrevem a **sintaxe** e a **semântica** desses **serviços**;
  - Ex: em redes de computadores, há regras que governam o formato, o conteúdo e o significado de mensagens enviadas e recebidas. Tais regras são formalizadas em protocolos;

# Abertura

- ▶ Como alcançar a abertura?
  - Em SDs, em geral os serviços são especificados por meio de **interfaces**;
    - Isolar detalhes da implementação de um certo componente;
      - Ex: utilização de recursos do SO (memória, CPU, etc.)
    - Permitir a interoperabilidade de componentes;
    - Geralmente descritas em uma **linguagem de definição de interface** (*Interface Definition Language* – IDL);
  - Documentação e especificação;
  - Código aberto.

# Abertura

- ▶ Definições em IDL capturam apenas a **sintaxe de serviços**;
  - ▶ Em outras palavras, as **interfaces** especificam com precisão os **nomes das funções** que estão disponíveis em um servidor, junto com os **tipos de parâmetros**, **valores de retorno** e possíveis execuções que podem surgir e assim por diante;
  - ▶ A aplicação cliente sabe que serviços o servidor disponibiliza, mas não precisa saber como ele foi implementado!
  - ▶ Logo, uma interface pode ser vista como um **protocolo de comportamento**;
  - ▶ Permite também que **duas partes independentes** construam **implementações** completamente diferentes dessas interfaces.
- 

# Interfaces em Java

```
//Arquivo TesteInterface.java
public interface Eletronico{

    public void ligar();
    public void desligar();

}
```

```
//Arquivo Televisao.java
public class Televisao
    implements Eletronico {

    public void ligar() {
        System.out.println(
            "Liguei a televisão");
    }
    public void desligar() {
        System.out.println(
            "Desliguei a televisão");
    }
}
```


```
Arquivo Impressora.java
public class Impressora
    implements Eletronico {

    private boolean ligada = false;
    public void ligar() {
        this.ligada = true;
        System.out.println(
            "Impressora ligada!");
    }
    public void desligar() {
        this.ligada = false;
        System.out.println(
            "Impressora desligada!");
    }
}
```



# Abertura

## ▶ Analogia:

- Imagine que vários grupo de programadores devem concordar em um “contrato” que descreve como um software deve interagir;
  - Cada grupo poderá escrever seu próprio código sem nenhum conhecimento de como os outros grupos estão escrevendo;
  - Desta forma, podemos enxergar interfaces como **contratos** que definem como um sistema deve interagir, independente de sua implementação.
- 

# Abertura

- ▶ Imagine também uma sociedade futurista com carros controlados por computador;
- ▶ Fabricantes de automóveis devem escrever um software que seja capaz de operar o veículo (parar, acelerar, virar a esquerda, virar a direita);
- ▶ Um outro grupo industrial, voltado a instrumentos de orientação, implementam sistemas dirigem os carros baseados na sua posição (GPS) e dados sobre as condições do tráfego;
- ▶ Tais fabricantes de automóveis devem entrar num acordo e publicar um **interface padrão** da indústria que explique em detalhes quais métodos devem ser invocados para dirigir o veículo (para qualquer veículo de qualquer fabricante).
- ▶ Os fabricantes de instrumentos de orientação então devem escrever softwares que invoquem tais métodos descritos na interface para comandar o carro

# Interfaces em Java

```
public interface OperarCarro{  
    int virarEsquerda(...);  
    int virarDireita(...);  
    int acelerar(...);  
    int freiar(...);  
}
```

```
public class OperateBMW760i implements OperarCarro{  
    int virarEsquerda(...) {  
        /* codigo com a implementação de virar o  
        BMW para esquerda */  
    }  
    int virarDireita(...) {  
        /* codigo com a implementação de virar o  
        BMW para direita*/  
    }  
    int acelerar(...) {  
        /* código com a implementação de acelerar o  
        BMW */  
    }  
    int freiar(...) {  
        /* código com a implementação de freiar o  
        BMW */  
    }  
}
```

Fabricantes de instrumentos de orientação não precisam se preocupar em como os métodos foram implementados, e nem como são chamados e quais parâmetros devem ser passados, pois todos seguirão um **padrão**.

# Escalabilidade

- ▶ Suportar o aumento dos recursos e usuários mantendo um desempenho satisfatório;
- ▶ Um sistema **escalável** é aquele que **continua eficiente** quando há um **aumento na quantidade de recursos e usuários**, sem a necessidade de se realizar alterações nas implementações dos serviços;
- ▶ Tipos de escalabilidade:
  - De recursos e usuários;
  - Escalabilidade geográfica (rede local, país, mundo,...);
  - Escalabilidade administrativa (uma organização, inter-organizações);

# Escalabilidade

## ▶ Tipos de sistemas:

- **Pequena escala:** sistema confinado a uma pequena organização, envolvendo poucos componentes e usuários;
- **Grande escala:** sistema envolvendo muitos recursos e usuários, cobrindo uma área geográfica significativa e podendo abranger várias organizações.

## ▶ Alguns desafios:

- Controle dos **custos** dos recursos físicos;
- Controlar a **perda de performance**;
  - Distribuição;
- Prevenir o esgotamento de recursos;
- Evitar gargalos de performance;
  - Centralização;

# Escalabilidade

## WORLD INTERNET USAGE AND POPULATION STATISTICS JUNE 30, 2014 - Mid-Year Update

World Regions	Population ( 2014 Est.)	Internet Users Dec. 31, 2000	Internet Users Latest Data	Penetration (% Population)	Growth 2000- 2014	Users % of Table
<u><a href="#">Africa</a></u>	1,125,721,038	4,514,400	<b>297,885,898</b>	26.5 %	6,498.6 %	9.8 %
<u><a href="#">Asia</a></u>	3,996,408,007	114,304,000	<b>1,386,188,112</b>	34.7 %	1,112.7 %	45.7 %
<u><a href="#">Europe</a></u>	825,824,883	105,096,093	<b>582,441,059</b>	70.5 %	454.2 %	19.2 %
<u><a href="#">Middle East</a></u>	231,588,580	3,284,800	<b>111,809,510</b>	48.3 %	3,303.8 %	3.7 %
<u><a href="#">North America</a></u>	353,860,227	108,096,800	<b>310,322,257</b>	87.7 %	187.1 %	10.2 %
<u><a href="#">Latin America / Caribbean</a></u>	612,279,181	18,068,919	<b>320,312,562</b>	52.3 %	1,672.7 %	10.5 %
<u><a href="#">Oceania / Australia</a></u>	36,724,649	7,620,480	<b>26,789,942</b>	72.9 %	251.6 %	0.9 %
<u><a href="#">WORLD TOTAL</a></u>	<b>7,182,406,565</b>	<b>360,985,492</b>	<b>3,035,749,340</b>	<b>42.3 %</b>	<b>741.0 %</b>	<b>100.0 %</b>

Fonte: <http://www.internetworldstats.com/stats.htm>

# Segurança

## ▶ Principais características:

- Confidencialidade:
  - Indivíduos não autorizados não podem obter informação;
- Integridade:
  - Os dados não podem ser alterados ou corrompidos;
- Disponibilidade:
  - O acesso aos recursos e serviços deve continuar disponível.

## ▶ Alguns desafios:

- Autenticação dos parceiros;
  - Utilização de canais de comunicação seguros;
  - Prevenção de ataques DoS;
- 

# Tratamento de falhas

- ▶ Os **componentes** de um sistema distribuídos **podem falhar**, comportando-se de **maneira não prevista** ou não de acordo com o que foi especificado:
  - Problemas na rede;
  - Defeitos em componentes;
- ▶ Geralmente as **falhas** em sistemas distribuídos são **isoladas e independentes**, desta forma:
  - A **falha** de um componente **não deve induzir** a uma mudança incorreta em outro componente, e levando assim a uma **falha generalizada**;



# Tratamento de falhas

- ▶ Tipos de falhas:

- Físicas;
- Software;
- Humanas;

- ▶ Utilizar técnicas de:

- Detecção de falhas;
- Mascaramento de falhas:
  - Após detectar a falha, não torná-la visível para os usuários;
- Replicação de componentes:
  - Ex: RAIDs para sistemas de armazenamento, replicação de serviços e recursos de forma a estar sempre disponível, etc.
- Recuperação após falhas:
  - Recuperar o estado de um serviço, pois algumas operações podem ter sido parcialmente executadas → tratar resultados corrompidos;

# Concorrência

- ▶ Recursos compartilhados possam ser utilizados por diversos processos;
- ▶ Desafios:
  - Sincronização;
  - Disponibilidade;
  - Segurança.

# Transparência

- ▶ **Ocultar o fato de que seus processos e recursos estão fisicamente distribuídos por vários computadores;**
- ▶ **Tipos de transparência:**
  - Acesso;
  - Localização;
  - Migração;
  - Relocação;
  - Replicação;
  - Concorrência;
  - Falha;

# Transparência

Transparência	Descrição
Acesso	Ocultas as diferenças na representação de dados e no modo de acesso a um recurso;
Localização	Ocultas o lugar em que um recurso está localizado
Migração	Ocultas o fato de que um recurso pode ser movido de um local para outro
Relocação	Ocultas o fato de que um recurso pode ser movido enquanto está em uso
Replicação	Ocultas a replicação de um recurso
Concorrência	Ocultas que um recurso está sendo compartilhado por diversos processos diferentes
Falha	Ocultas a falha e recuperação da mesma por um recurso

# SISTEMAS DISTRIBUÍDOS

## Tipos de Sistemas Distribuídos

Prof. Guilherme C. Kurtz




# Tipos de Sistemas Distribuidos


- ▶ Sistemas de Computação
- ▶ Sistemas de Informação
- ▶ Sistemas Pervasivos



# Sistemas de Computação

- ▶ Uma classe importante de sistemas distribuídos é a utilizada em tarefas que necessitam de **computação de alto desempenho**;
  - ▶ Desta maneira, pode-se dividir esta classe em dois grupos;
  - ▶ No caso da computação em **cluster**, o hardware consiste em várias estações de trabalho conectados pela rede local, sendo que cada nó executa o mesmo sistema operacional.
- 

# Sistemas de Computação

- ▶ O segundo grupo é a computação em **grade**. Estes sistemas distribuídos geralmente são montados como uma **federação de computadores**, em que cada sistema pode cair em um **domínio administrativo diferente**, podendo ser muito diferentes em relação a **hardware, software, rede e sistema operacional**.
- 



# Clusters

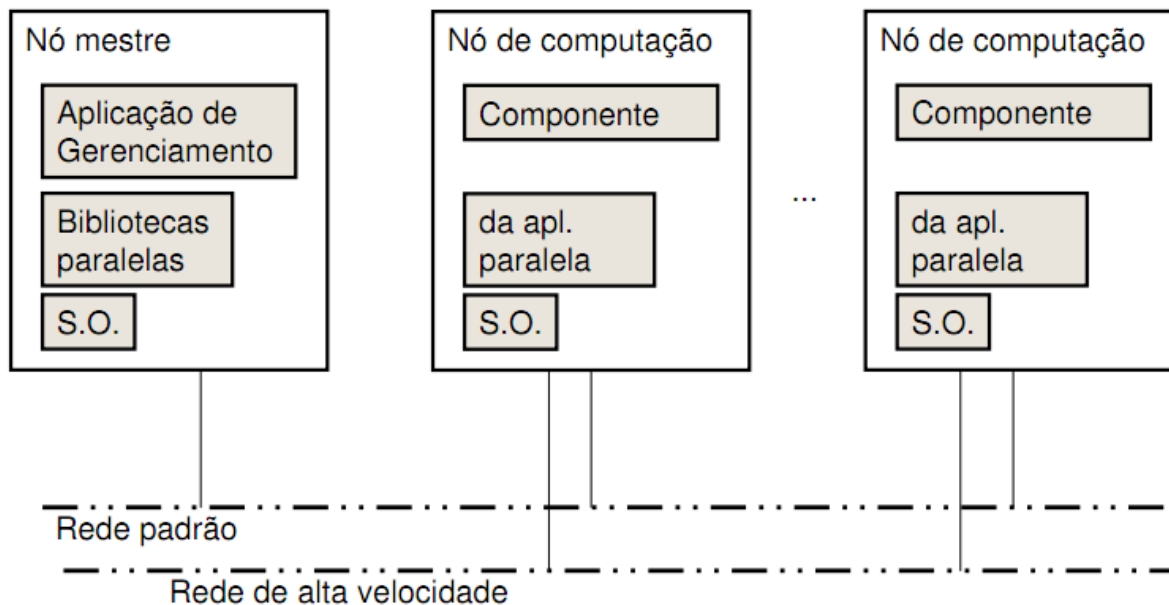
- ▶ Conjunto de **estações de trabalho/PCs** de **configuração semelhante**;
- ▶ A **conexão** entre as estações é feita através da **rede local**;
- ▶ Na maioria dos casos a computação em cluster é utilizada na **programação paralela**, ou seja, uma única aplicação é executada em paralelo;

# Clusters

- ▶ Passaram a se tornar **populares** na medida que o **preço e o desempenho** das estações de trabalho/computadores pessoais **melhorou**;
- ▶ A idéia de se utilizar **computadores de prateleira conectados em uma rede de alta velocidade** para se construir supercomputadores passou a ser atraente;

# Clusters

## ► Clusters Beowulf baseados em Linux




- **Nó mestre:** aloca nós de computação a uma determinada aplicação paralela, gerenciando o cluster
- **Nó de computação:** executam sua parte na aplicação paralela;


# Grade/Grid

- ▶ **Federação de computadores:**
  - **Recursos de diferentes organizações** são reunidos para permitir a **colaboração** de um grupo de pessoas ou instituições;
- ▶ **Heterogeneidade:**
  - Hardware;
  - Software;
  - Rede;
  - Domínios administrativos;
  - Políticas de segurança;
  - Sistema Operacional.
- ▶ **Exemplos:**
  - SETI@home: análise de sinais de rádio vindos do espaço;
  - LHC@home: processamento dos dados coletados no LHC;

# Sistemas de Informação

- ▶ Exemplo clássico: **servidor de banco de dados**;
  - ▶ Basicamente este tipo de aplicação consiste em um **servidor** que executa uma **aplicação** que inclui um **banco de dados**, e programas remotos nos **clientes**;
  - ▶ Tais clientes enviam **requisições** para o servidor executar alguma operação específica e então enviar uma **resposta** para o cliente;
  - ▶ A medida que as aplicações se tornaram mais sofisticadas, **seus componentes passaram a se separar gradualmente**, distinguindo entre componentes de bancos de dados e de processamento, e também permitindo a comunicação de uma aplicação com outra.
- 

# Sistemas Pervasivos

- ▶ Dispositivos de computação móvel e embutidos:
    - Aparelhos de pequeno porte;
    - Mobilidade;
    - Alimentação geralmente por bateria;
    - Conexão sem fio;
  - ▶ Ausência de controle humano:
    - No máximo configurados por seu proprietários;
  - ▶ Dispositivos devem estar continuamente cientes do fato de que seu ambiente pode mudar o tempo todo:
    - Ex: descobrir que uma rede não está mais acessível devido a movimentação do usuário, e então procurar outra rede.
- 

# Sistemas Pervasivos

- ▶ Requisitos para aplicações pervasivas:
  - Adotar mudanças do contexto;
  - Incentivar composição ad-hoc;
  - Reconhecer compartilhamento como padrão;
- ▶ Um aspecto muito importante de sistemas pervasivos é que, em geral, os dispositivos se juntam ao sistema para acessar e possivelmente fornecer informações. Isso requer meios para **ler, armazenar, gerenciar e compartilhar informações** com facilidade;
- ▶ Exemplos de sistemas:
  - Sistemas domésticos;
  - Redes de sensores;
  - Sistemas Eletrônicos para Tratamento de Saúde;

# Sistemas Pervasivos

## ▶ Sistemas domésticos:

- Sistemas compostos por um ou mais computadores pessoais;
- **Integração de eletrônicos** tal como aparelhos de TV, equipamentos de áudio e vídeo, videogames, smartphones, PDAs, entre outros, todos **em um único sistema**;
- Desafios:
  - **Autoconfiguráveis e autogerenciáveis**, pois não se pode esperar que usuários finais estejam dispostos ou sejam capazes de manter um sistema ligado e em funcionamento caso ocorram erros;
  - Atualizações de software e firmware automatizadas;
  - Espaço pessoal: sempre acessíveis, mas com restrições/privacidade



# Sistemas Pervasivos

- ▶ **Sistemas eletrônicos para tratamento de saúde:**
  - Dispositivos para **monitorar** o bem-estar de indivíduos e entrar **automaticamente** em **contato com médicos** quando necessário;
  - Uma meta importante é **evitar** que pessoas sejam **hospitalizadas**;
  - Costumam ser equipados com **vários sensores sem fio**;
  - A rede deve **incomodar** uma pessoa o **mínimo possível**;
  - **Desafios:**
    - Como os dados monitorados serão armazenados? (servidor local? Hospital?)
    - Como evitar perda de dados? (deslocamento, perda de sinal...)
    - Qual a infra-estrutura necessária para gerenciar este tipo de sistema?
    - Como os médicos podem dar um retorno online?
    - Questões de segurança.

# Exercícios

1. De acordo com as várias definições, como você definiria um Sistema Distribuído?
  2. Quais as vantagens de se utilizar um SD?
  3. Em quais aspectos um SD pode ser heterogêneo?
  4. Que tipos de soluções temos em relação a heterogeneidade?
  5. O que você entende por Abertura e Interfaces em SD?
  6. Quais são os tipos de problemas de escalabilidade existentes em SD?
  7. Cite e explique os principais tipos de Sistemas Distribuídos.
- 