

# **Computação Onipresente**

**(UBICOMP - Ubiquitous  
Computing)**

**Manoel C. M. Neto**  
**GSORT- IFBA**

# Paradigma Tradicional

- PC: Teclado, Mouse, Monitor
  - Um computador usado para a manipulação de informações pessoais (mesmo que profissionais)
- Redes: Locais (mesmo que conectadas)
  - Redes cabeadas ou sem fio cujos serviços tem endereço fixo (File Server, Impressoras, etc...)
- Interface de Usuário: Apresentação de metáforas visuais na tela
  - Janelas, manipulação de arquivos, ícones, ponteiro do mouse, menus, etc.

# Ubicomp: Paradigma Completamente Novo

- Sistemas Ubíquos: dispositivos heterogêneos
  - PDAs, Carros, Aviões, Smartphones, laptops, surfaces, etc.
- Infra: heterogênea
  - Diferentes: SO's, interfaces de rede, redes, capacidades (memória, I/O, processamento, etc.), telas, etc.
- Interface de Usuário: Sim e Não
  - Alguns sistemas são feitos para uso direto por usuários (finais): Interface de um smartphone.

# Ubicomp: Paradigma Completamente Novo

- Interface de Usuário: Sim e Não
  - Diferente do PC (um-para-um) em ubicomp muitos-para-muitos
  - Diferente do PC em ubicomp interação implícita, invisível ou mesmo natural: Voz, gestos, presença.
  - Muitos sensores necessários: embarcados em dispositivos ou no próprio ambiente.
  - GPS, Acelerômetro, Câmera, etc (sensores). Podem detectar por exemplo a presença de alguém e deduzir o que está fazendo!!
  - Informação usada para adaptar a aplicação

# Ubicomp: Paradigma Completamente Novo

- Rede: **Normalmente sem fio**
  - Redes sem fio e ad hoc
  - Dispositivos podem se entrar em contato espontaneamente e estabelecer serviços.
  - Quando um dispositivo sai muda a rede (configuração) e o próprio ambiente.
  - Compartilhar fotos via Bluetooth para um servidor conectado a uma rede sem fio.
  - Depois do download para smartphone (Wi-fi) e depois... o mundo é o limite (3G+Facebook) !!!

# Ubicomp: Paradigma Completamente Novo

- Ex.: Hospital do futuro (Bardram et al., 2006)
  - Sensores de presença
  - Médicos recebem informações importantes de seu paciente por proximidade (ECG, RX, tipos de alergia, etc.) em seu próprio dispositivo (tablet ou smartphone).
  - Podem usar alternativamente telas nos quartos ou corredores.
  - Controle do uso correto dos medicamentos (sensor no medicamento + sensor do paciente).

# Ubicomp: Design de Sistemas

- As pesquisas em Ubicomp
  - Usar a infra e tecnologias citadas para criar sistemas
- Algumas questões tratadas (tentativas):
  - Como criar hardware para plataformas de sensores?
  - Como criar SO's para tais plataformas?
  - Como permitir que os dispositivos se encontrem?
  - Como permitir que os dispositivos usem os serviços uns dos outros?
  - Como criar sistemas para rodar em dispositivos "pobres" (Bateria, memória, etc.)
  - Como fazer funcionar em grande infra / sistemas distribuídos, móveis juntos mas invisíveis!!! (no hospital por exemplo)

# Fundamentação de design e processo de criação de "bons" sistemas ubicomp

# Tópicos e Desafios

- São vários !!!
- Começando... **Dispositivos com Recursos Limitados**
  - O primeiro e talvez o desafio menos abstrato.
  - Lei Moore: para PC's MASSA!!!
  - Para dispositivos móveis a história é outra: pouca memória, processamento, conectividade.
- Quando projetar sistemas é importante observar as limitações / heterogeneidade de dispositivos (pode haver incompatibilidades).

# Computação Resource-aware

- Abordagem para desenvolver tecnologias onde as aplicações são notificadas sobre o consumo de recursos.
  - Elas (ou os seus usuários) podem se adaptar!
  - Ex.:
    - Vídeo streaming (é avisado sobre banda disponível - Netflix já faz isso!!)
    - Diminuição do brilho da tela para economizar bateria.

# Computação Resource-aware

- Energia: Um tópico fundamental
  - Fator limitador para a maioria dos dispositivos ubíquos.
  - Normalmente usam baterias.
  - Dispositivos pequenos = baterias pequenas = menos autonomia (tempo de uso)
- Problema ou Oportunidade?
  - Escolher hardware baseado no consumo é o principal problema.
  - Oportunidade de criar sistemas que produzam (ou economizem) energia:
    - Energia cinética de pessoas andando.
    - KERS (Sistema de recuperação de energia cinética) da F1

# Computação Resource-aware

- Criatividade para ajudar na resolução:
  - Aplicações que apenas usam serviços através de servidores:
    - Se um usuário quer imprimir um arquivo através de um PDA.
    - Se o arquivo está em um servidor da rede
    - Ele pede ao servidor que imprima
    - Assim, economiza energia ao usar menos rede para download e depois upload para impressora
- Processamento, acesso a memória, I/O ... tudo consome energia mas o grande vilão é .....

# Computação Resource-aware

- A rede sem fio (de qualquer natureza)
- Tipicamente encontra em dispositivos ubíquos.
- Dessa forma é essencial:
  - Investigar protocolos mais eficientes (perspectiva do consumo de energia) sem perder desempenho na rede.
  - Ex.: Processamento consome menos energia que comunicação (em regra)
  - Redes de sensores (MANET's) tentam realizar o máximo possível de processamento (agregação, filtragem) em um nó (dos nós vizinhos) antes de transmitir.

# Ambientes de execução voláteis

- Uma das questões centrais: Descoberta de serviços
  - Tecnologias ou padrões que permitam que os dispositivos "se encontrem".
  - Ex.: Quando você entra em uma rede sem fio com seu notebook existe um protocolo que te ajuda a localizar a impressora (Bonjour / mDNS)
  - Zero configuration networking: Grupo de tecnologias que incluem descoberta de serviços, endereçamento automático (DHCP), resolução de nomes de máquinas, UPnP (Universal Plug an Play).

# Ambientes de execução voláteis

- Área bastante amadurecida (Descoberta de Serviços)
- Porém ainda existem muitos desafios:
  - Falta de suporte de descoberta multicast em redes locais.
  - Falta de suporte ao emparelhamento entre dispositivos (um-para-um / entre serviços)
  - Métodos complicados/pesados para tal: pedem Id`s, senhas .... tem que ser invisível lembram?

# Ambientes de execução voláteis

- Sistemas Ubicomp são normalmente distribuídos
- Sob esta perspectiva a volatilidade é questão importante:
  - Hardware, software, dispositivos, SO's muito dinâmicos ... "mudam frequentemente".
  - Ex.: Volatilidade de sistemas que conectam e desconectam da rede, criam e destroem links de comunicação.

# Ambientes de execução voláteis

- Sistemas Ubicomp devem tratar esse tipo de volatilidade de forma “graciosa”
- Um dispositivo pode ser desligado ou deixar um ambiente ou ainda ficar sem bateria a qualquer tempo!
- Outro tipo de volatilidade: Mudança na Infra
  - Topologia
  - Largura de banda
  - Mudança de Nomes (resolução de nomes)

# Ambientes de execução voláteis

- Em uma casa / sala inteligente
  - Um novo dispositivo que entra não conhece nada
  - Descoberta de serviços
  - Roteamentos para serviços
- Importante: Volatilidade também existe em sistemas tradicionais
  - Cliente-Servidor (em smartphones, notebooks e servidores) também desconectam!!
  - Nesses sistemas a rede é fixa, o servidor se mantém estável (quando um cliente desconecta) com mesmo nome/endereço.

# Ambientes de execução voláteis

- Importante: Volatilidade também existe em sistemas tradicionais
  - A diferença é que em SD (volatilidade é excepcional) em Ubicomp (é comum ou mais natural)
  - HTTP, CORBA, RMI...premissa de conexões de rede estáveis (socket)
  - Em Ubicomp isso cai completamente !!

# Ambientes de execução Heterogêneos

- Aplicação em Sistemas tradicionais
  - Software que roda em um (ou muitos) nós.
  - Exemplo Aplicação Web (JSP, JSF, etc.)
- Aplicação em Ubicomp
  - Muitos dispositivos (diferentes), conectados (redes diferentes), que trabalham de forma coordenada para atingir um objetivo
  - Ex.: Sala inteligente = Aplicação ubíqua

# Ambientes de execução Heterogêneos

- Tratar heterogeneidade:
  - Não é apenas compilar, construir e implantar uma aplicação para plataformas diferentes (Mac, Windows, Linux, etc.).
  - É uma questão de manter a “continuidade” em tempo de execução nas partes da aplicação que estão em dispositivos com especificações variadas.
  - Ex.: Usuário, entra no GSORT, e quer usar impressora (a partir do seu notebook) para imprimir um documento (servidor Arq. IFBA) e usar seu smartphone para ver e-mail.
  - Protocolo mDNS (descoberta impressora), SO (Windows, Android) Hardwares diferentes

# Ambientes de execução Heterogêneos

- Tratar heterogeneidade:
  - Ainda é complicado pois área muito imatura.
  - A evolução vai evitar a colcha de retalhos em tecnologia.
  - No futuro sistemas vão tratar requisitos de ubiquidade de forma mais homogênea.
  - Por outro lado a evolução dos sistemas Ubicomp vai requerer novas tecnologias, hardwares, etc.
  - Desafios sempre vão existir. Ex.: Nanotecnologia, energias renováveis...etc.
  - Dica: Criar sistemas básicos que pensem em Ambientes heterogêneos

# Ambientes de Uso Flutuantes

- É algo (em um certo grau) que está relacionado com a natureza dos sistemas ubíquos e com a forma que eles são planejados.
- Na computação tradicional:
  - O alvo principal (geralmente) é o gerenciamento de informações (GI).
  - Usuários de PC's fazem GI localmente ou em servidores (Rede)
  - Contexto físico comum é normalmente uma superfície horizontal plana ( Ex.: Mesa no trabalho ou em casa).
  - Quantidade e complexidade de periféricos bem conhecida e limitada (impressoras, HD's externos, câmeras, etc.)

# Ambientes de Uso Flutuantes

- Comparados com a computação tradicional, Sistemas Ubicomp “vivem” em um ambiente de uso muito mais complicado e flutuante.
- Na Ubicomp:
  - O mesmo dispositivo pode ser usado por vários usuários (ex.: Um monitor em uma sala inteligente)
  - O contexto físico é qualquer um (lugar) !!
  - Uso de dispositivos móveis e sistemas embarcados = trabalho em qualquer lugar e uso por qualquer pessoa.
  - A execução de uma tarefa é “distribuída” entre vários dispositivos (heterogêneos como visto).
  - O foco está na tarefa e não na complexidade computacional dos dispositivos e sistemas (emparelhamento, login, rede, conectividade, etc.)

# Ambientes de Uso Flutuantes

- Os desafios para tratar ambientes de uso flutuantes significa construir hardwares, plataformas, SO's, aplicações que permitam este tratamento.
- Eles concentram no tratamento (sentido amplo) de 3 ocorrências:
  - Mudanças na localização do usuário;
  - Mudanças no contexto do computador; e
  - Solicitações de múltiplas tarefas por usuários.
- O primeiro surgiu com a computação móvel
  - Termo : *Location-based computing*
  - Criar sistemas que permitam saber a localização de um usuário e usar esta informação para um determinado fim.
  - GPS é o melhor exemplo...mas existem outros (*GUIDE project*).

# Ambientes de Uso Flutuantes

- O segundo: *Context-aware computing.*
  - Visa adaptar o computador (ou a aplicação) a uma mudança de contexto.
  - Depende ou usa diversas variáveis:
    - Quem está usando o computador
    - Quem ou quais outros dispositivos estão próximos
    - Qual nível de Luz, Temperatura, Som, etc. (Ambiente).
    - Que ferramentas ou materiais (físicos) estão sendo utilizados
- Ex.: Um sistema soaria um alarme ao detectar que um medicamento errado estaria sendo administrado.
  - Reconhecer QUEM é o paciente QUAL é o medicamento entre outras variáveis

# Ambientes de Uso Flutuantes

- Ainda em: *Context-aware computing*.
  - O contexto muda por dois motivos (*senso comum*):
    - O dispositivo foi movido para uma nova localização (novo contexto) - ex.: dispositivos móveis
    - O contexto físico de computadores/ sistemas embarcados mudam por exemplo quando um novo usuário ou novo dispositivo entra em uma casa inteligente.
  - Como tratar?
    - Sentir (sensores), investigar arquiteturas adequadas, modelar, reunir e filtrar informações, e depois usar tudo isso para permitir a adaptação de um sistema/computador.

# Ambientes de Uso Flutuantes

- O terceiro: *Activity-based computing (ABC)*.
  - Objetivo: Tratar a flutuação baseado na necessidade da execução de múltiplas tarefas de múltiplos usuários.
- Em um hospital são muitos pacientes, médicos, enfermeiros, medicamentos, procedimentos...muitas vezes em paralelo = Overhead!!
- Sistemas, dispositivos e aplicações devem ser capazes de ajudar a manter o foco para evitar problemas na execução das atividades.
- Têm que suportar a execução orquestrada e simultânea (dispositivos heterogêneos, redes diferentes, aplicações diferentes, mudanças de contexto...)

# Computação Invisível

- Muito importante para sistemas Ubíquos
- Alcançar a “invisibilidade” ainda é um desafio.
- O computador invisível é na verdade uma metáfora:
  - Podem estar embarcados em prédios, dispositivos médicos etc. São (estão) invisíveis aos olhos humanos.
  - Operam na periferia da atenção humana e neste caso são mentalmente invisíveis (imperceptíveis).
- Representa uma mudança considerável para computação tradicional
  - Antes a computação era baseada na ENORME atenção dos usuários
  - Os usuários para usar um computador precisavam usar (fisicamente, tocar nele, ligar etc.) um computador!!!

# Computação Invisível

- Representa uma mudança considerável para computação tradicional
  - Os softwares eram construídos para mandar/receber mensagem e aguardar a reação do usuário (por exemplo no tratamento de exceções, solicitar que usuário instale um plugin, o velho boot 😊 etc.).
- Na computação invisível todas as premissas anteriores são quebradas!

# Computação Invisível

- Muito tem sido feito para buscar a tal invisibilidade
- Uma dessas áreas é computação autônoma.
  - Visa criar sistemas que possam “se auto gerenciara a se mesmos ☺” ... para superar o rápido crescimento da complexidade do gerenciamento de sistemas.
  - Uma boa definição: “refere-se ao auto gerenciamento de recursos computacionais distribuídos, para permitir possíveis adaptações a mudanças não previstas de forma a ocultar toda a complexidade disso para o usuário final”
  - Toma decisões usando políticas de alto nível e está sempre buscando o status ótimo.
  - (Senso comum) é baseada em uma arquitetura centralizada (ou baseada em clusters).

# Computação Invisível

- Outra área: Sistemas multi-agentes.
  - Ou contrário da computação autonômica não é centralizado.
  - Visa criar agentes que possam migrar pelos diferentes dispositivos conectados a uma rede e trabalhem na execução das tarefas de usuários.
  - São projetados para proteger (tornar invisíveis) as características de baixo nível dos sistemas.

# Computação Invisível

- Gerenciamento de contingência.
  - Visa criar sistemas que evitem o envolvimento de usuários quando falhas/erros ocorram.
  - Nos sistemas tradicionais as falhas são excepcionais
  - Aqui as falhas são contingências naturais de sistemas ubíquos.
  - Usa técnicas para gerenciamento proativo falhas e limitação de recursos.
  - Ex.: “descarregar” um agente de um dispositivo quando a bateria está próxima do esgotamento.

# Computação Invisível

- Degradação tranquila (graciosa).
  - Visa estruturar como os sistemas devem responder a mudanças, e em particular, a falhas do ambiente.
  - Boa parte das tecnologias assumem que a disponibilidade de certos recursos (Ex.: internet) estão permanentemente disponíveis.
  - Quando isso não ocorre um sistema inteiro pode parar!
  - *A graceful degradation* visa criar sistemas que quando um determinado recurso não estiver disponível (ou alguma mudança no ambiente ocorrer), o maior número de funcionalidades possível deve ser mantido (funcionando claro!) de forma mais imperceptível possível para o usuário.

# Segurança e Privacidade

- Um desafio para TODA a computação.
- Em Ubicomp o desafio é maior pois é preciso manter:
  - Heterogeneidade
  - Espontaneidade
  - Invisibilidade
- Confiança é base de qualquer sistema de segurança
  - Em Ubicomp a volatilidade parte do princípio que não é necessário o conhecimento prévio para interação espontânea dos dispositivos
  - Dessa forma a confiança é baixa!!
  - Ex.: Se você entrar em um hospital, é possível usar de forma ubíqua seu smartphone para baixar dados clínicos? Sim mas gera overhead administrativo (é preciso ID, senha, etc. )

# Segurança e Privacidade

- Os protocolos de segurança tradicionais têm premissas que nem sempre são verdadeiras em Ubicomp.
  - Dispositivos móveis são roubados ou adulterados mais facilmente.
  - Dispositivos móveis têm poucos recursos para suportar por exemplo chaves de criptografia assimétricas.
  - Protocolos de segurança não podem contar com um acesso contínuo a um servidor (para solicitar as chaves) como na Web.

# Segurança e Privacidade

- Em Ubicomp é necessário criar um novo tipo de segurança baseado em contexto e localização.
- A autenticação / autorização pode ser baseada na localização e não no usuário.
- Se você quer usar a impressora do GSORT basta estar na Rede do IFBA (ser aluno). Não importa quem é você mas onde você está.

# Segurança e Privacidade

- O uso de sistemas ubíquos pode coletar uma diversidade de dados:
  - Localização
  - Atividades
  - Amigos
  - Vídeos
  - Dados pessoais (Fator RH, CPF, RG, etc.)
- Caso do Goleiro Bruno: “Existem outras formas de provar o desaparecimento de uma pessoa”
  - Movimentação de cartões de crédito/banco.
  - Check-in/out em companhias aéreas.
  - IP

# Segurança e Privacidade

- Se os sistemas são invisíveis, os dados são coletados e os usuários não são notificados.
- O projeto de um sistema ubíquo deve levar em consideração algum mecanismo de proteção contra essas situações.
- O desafio é gerenciar usuários (conscientemente ou não)
  - Usar id's: endereços MAC, Bluetooth e IP, sensores RFID, logins, etc.). Para permitir a mudança de contexto/localização.

# Segurança e Privacidade

- A flutuação dos sistemas ubíquos prevê criação / destruição de associações naturalmente entre dispositivos/usuários.
- Se alguma(s) dessas associações precisar de garantias de segurança isso significa que a autenticação (usuário ou dispositivo) pode ocorrer com frequência.
- Os dispositivos de autenticação comuns (senso comum) são projetados para poucas associações e duram (sessão) na ordem de horas (ex.: Usuário loga em PC e trabalha o dia inteiro).

# Segurança e Privacidade

- Em Ubicomp os cenários preveem que um usuário pode entrar em um ambiente e usar dezenas de dispositivos em apenas alguns minutos.
- Autenticação tradicional nesses cenários são impossíveis.
- Para obter dados do Ar-condicionado (temperatura) seria necessário login e senha!!!