Universidade Federal de Ouro Preto

JSensor: Uma plataforma paralela para simulação de redes de sensores

Danniel Hugo Ribeiro, Joubert de Castro Lima, Ricardo A.R. Oliveira
{dannielhugo,joubertlima,rrabelo}@gmail.com – Universidade Federal de Ouro Preto

Andre L L Aquino
alla.lins@gmail.com – Universidade Federal de Alagoas

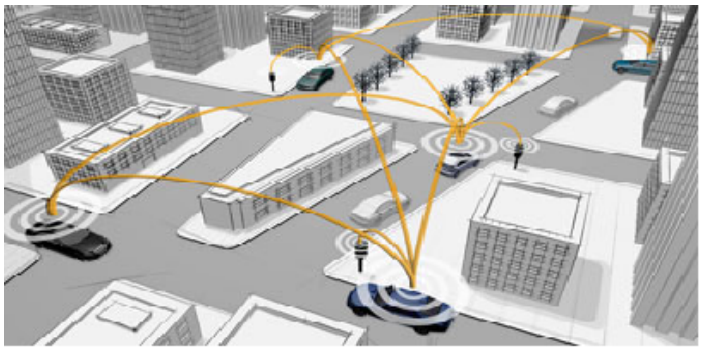
Agenda

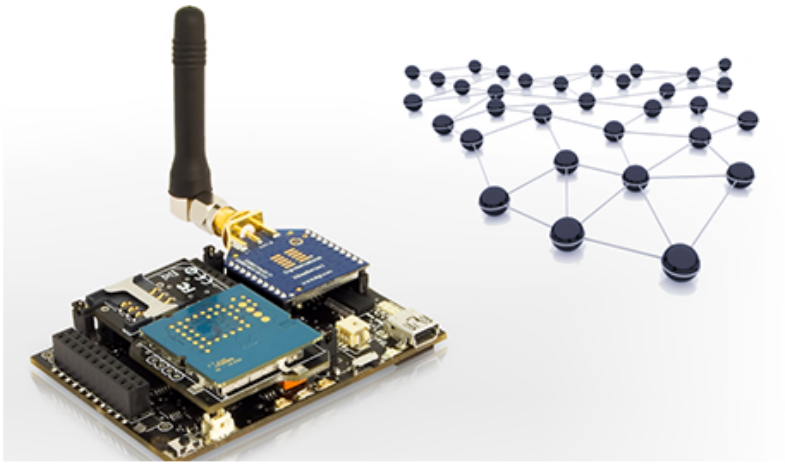
1. O problema
2. O que já foi feito ?
3. Solução JSensor
4. Experimentação
5. Trabalhos Futuros

Agenda

1. O problema
2. O que já foi feito ?
3. Solução JSensor
4. Experimentação
5. Trabalhos Futuros

O problema





Como simular, dezenas, senão centenas de milhões de sensores?

Como garantir simplicidade, escalabilidade e realismo nas simulações massivas?

Impacto para Sociedade

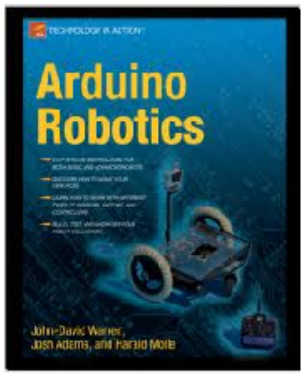
Cidades mais Inteligentes



Serviços de roteamento mais eficientes



Automação residencial





E MUITOS MAIS...

Agenda

1. O problema
2. O que já foi feito ?
3. Solução JSensor
4. Experimentação
5. Trabalhos Futuros

O que já foi feito ?

- Lee et al (2009) e Fernekess et al. (2009)
  Soluções: O primeiro apresenta uma técnica para o processamento paralelo de simulações orientadas a eventos temporais. O segundo introduz um simulador baseado em snapshots temporais da troca de pacotes em uma rede celular.
  Limitações: desconsideram os elementos de redes de Sensores
- Gao et al. (2009)
  Solução:Mostra um framework composto de vários simuladores, com diferentes modelos de redes sem fio.
  Limitação: Não simula muitos nós pois a solução é centralizada em detalhes mais abrangentes da rede, tais como as várias camadas da mesma.

O que já foi feito ?

- Niazi e Hussain (2009)
  Solução:Apresenta um simulador baseado na simulação de agentes em uma rede sem fio.
  Limitação: Não possui escalabilidade para arquiteturas multicore.
- TOSSIM
  Limitações: não possui uma versão paralela (multicore) e nem distribuída (multicomputer).
- OMNET.
  Possui versão de alto desempenho com extensão Horizon
  Muito utilizado pela sociedade, em especial pela comunidade acadêmica, no entanto algumas extensões não são compatíveis entre si.

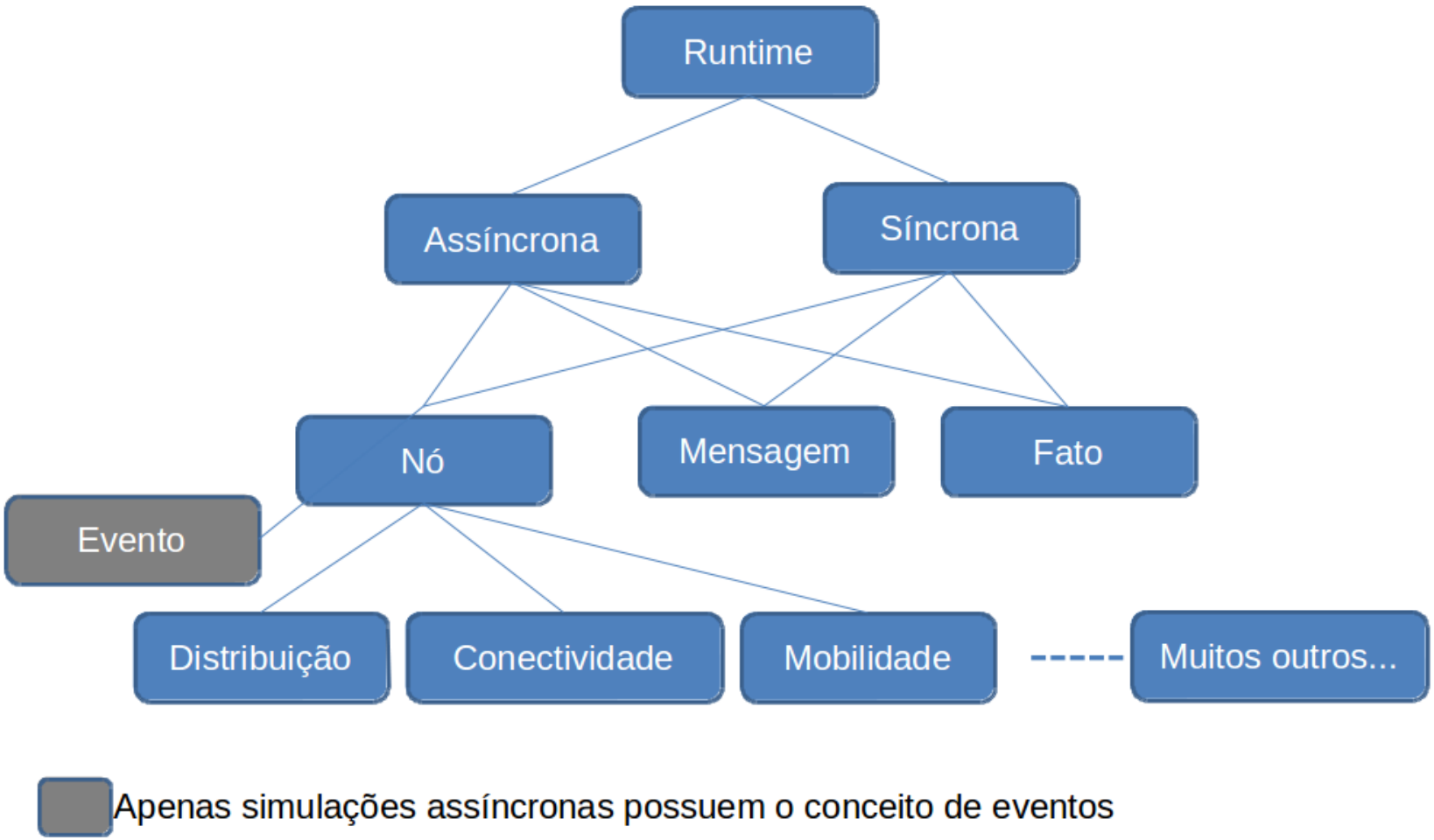
Agenda

1. O problema
2. O que já foi feito ?
3. Solução JSensor
4. Experimentação
5. Trabalhos Futuros

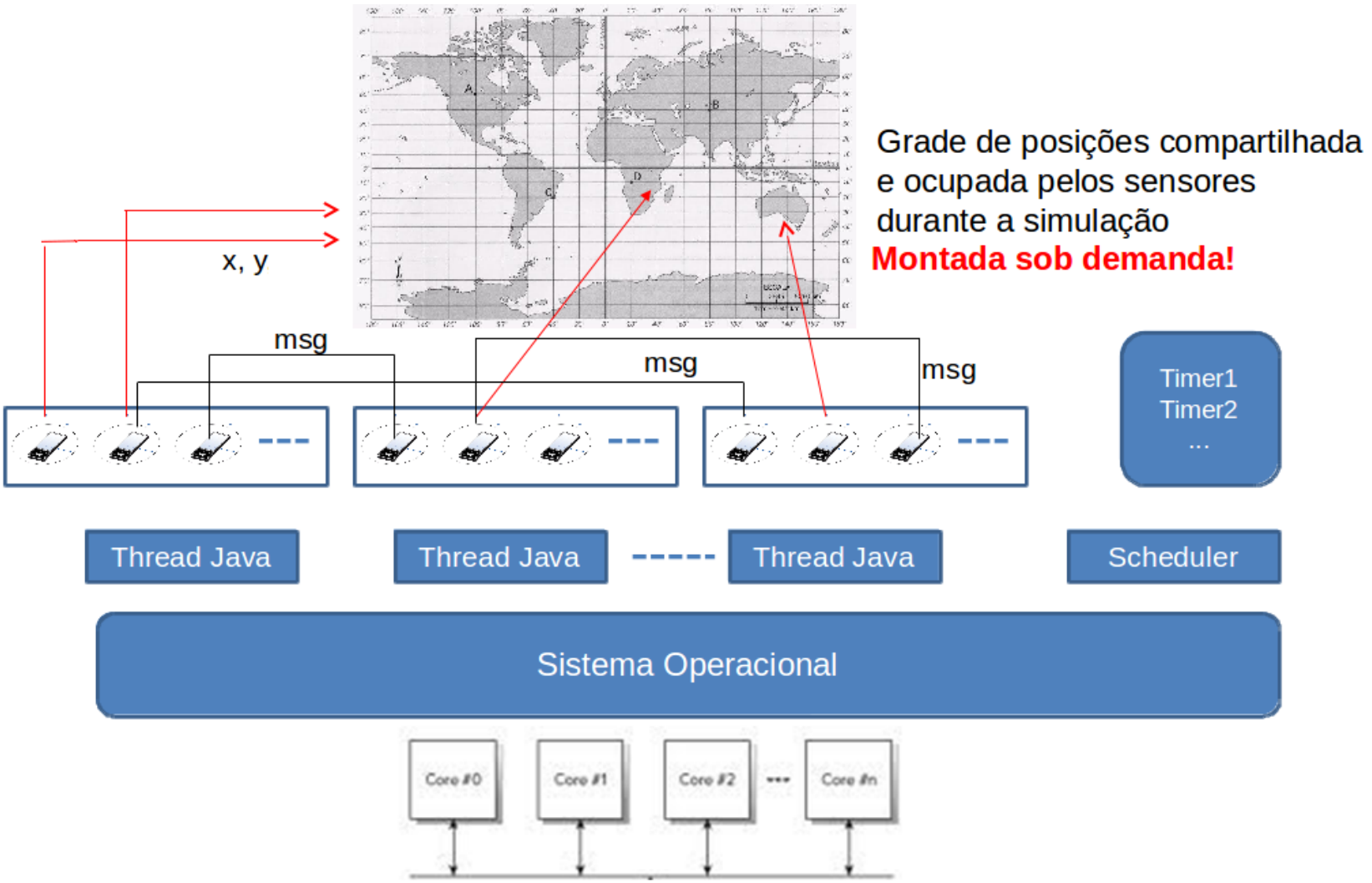
Solução JSensor

- Linguagem Java (portabilidade e simplicidade de codificação)
- Solução atual para arquiteturas de computadores com múltiplos núcleos de processamento
- Simulações Síncronas e Assíncronas
- Simples customizações para uso de quaisquer modelos de conectividade, mobilidade e distribuição
- Livre, Gratuita e disponível em: (www.joubertlima.com.br/jsensor)

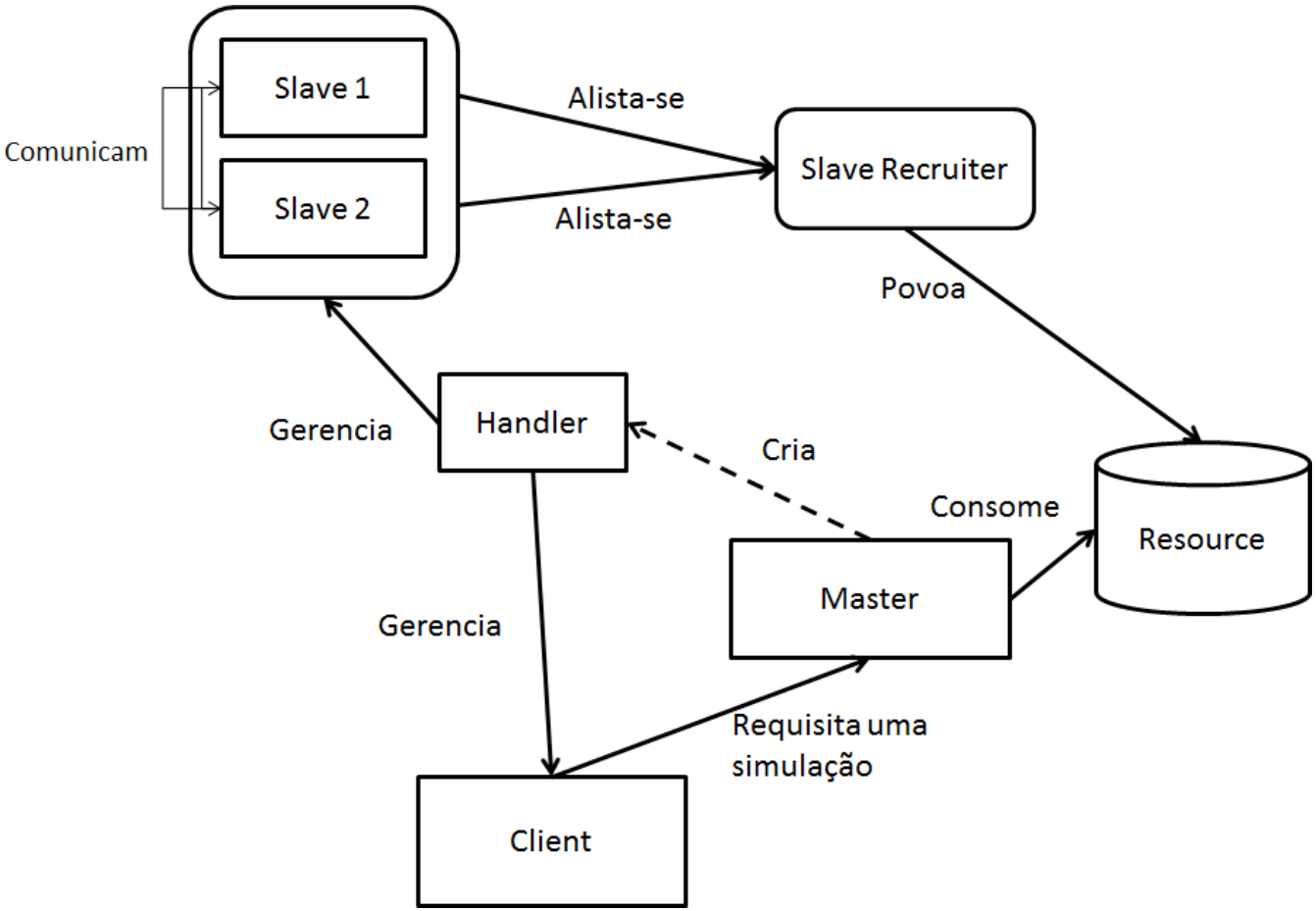
Arquitetura JSensor



Arquitetura JSensor



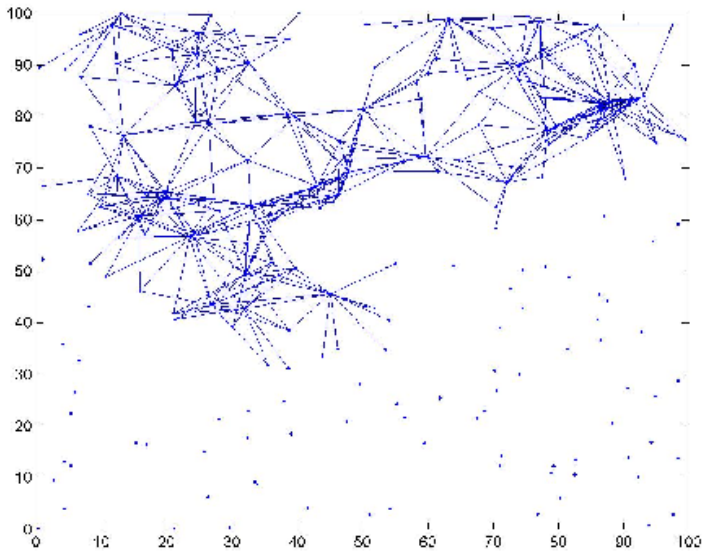
Arquitetura JSensor
Protótipo da versão distribuída



Agenda

1. O problema
2. O que já foi feito ?
3. Solução JSensor
4. Experimentação
5. Trabalhos Futuros

Experimentação

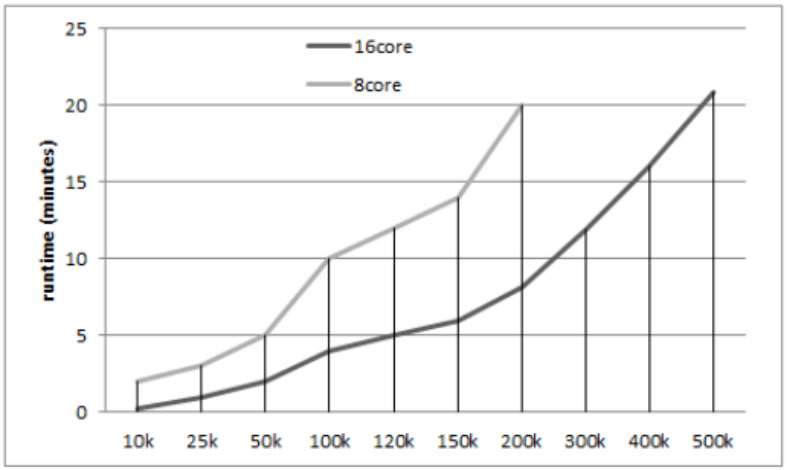


Fooding ou Inundação

- Simulação Assíncrona
- Cada sensor possui um conjunto de vizinhos
- Não possui mobilidade
- O sensor não envia a mesma mensagem a seus vizinhos
- Cada mensagem pode ser enviada em qualquer momento futuro da simulação a qualquer vizinho
- Conectividade, mobilidade e disposição também podem ser escalonados em momentos futuros por cada sensor.

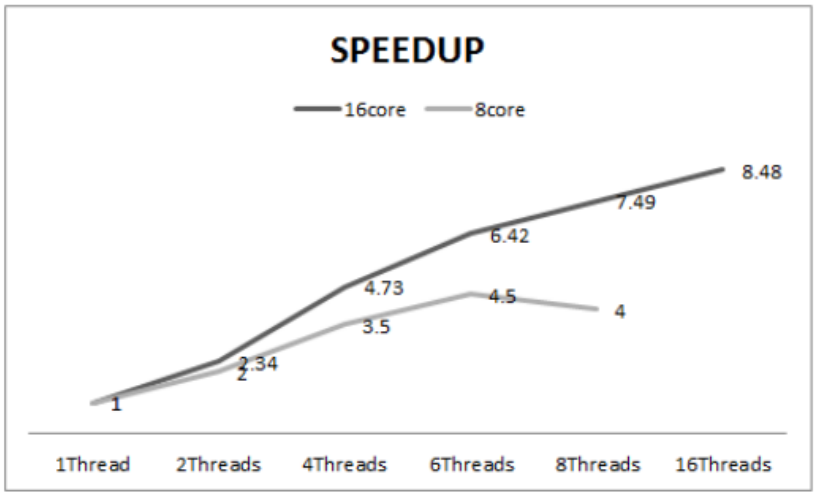
Parâmetros

- Máquinas:
  1. 2 processadores Intel 16 GB de RAM compartilhada; quatro drives SATAII com 7200 rpm; e sistema operacional Windows Server 2003 R2 64-bit
  2. 16 cores de 3 GHz: 8 físicos e 8 lógicos com 12 MB de cache; 48 GB de RAM DDR3 1.03 GHz; dois discos de 500 GB, 10k rpm e um buffer de 64 MB; e sistema operacional Linux Ubuntu 64-bit versão 11.10.
- Número de nós variando de 10k a 500k, cada um com raio de comunicação de 80m.
- Número de threads variando de 1 à 16, dependendo da máquina.
- Média de dez execuções por simulação.



(a) Runtime.

Nessa simulação foi consumido 40-44 GB RAM. Esse tipo de cenário, com milhares de nós, pode facilmente ser encontrado se considerarmos aplicacoes com sensores em celulares, veículos, sistemas de transporte entre outros.



(b) Speedup.

Na máquina 8-core alcançamos 90% de paralelismo, considerando-se a lei de Gustafson. Já na máquina 16-core alcançamos 95% de paralelismo,

Conclusões

- JSensor é eficiente, obtendo entre 90 e 95% de paralelismo.
- JSensor é portável
- Novos algoritmos de mobilidade, conectividade e distribuição podem ser facilmente adicionados ao JSensor

Agenda

1. O problema
2. O que já foi feito ?
3. Solução JSensor
4. Experimentação
5. Trabalhos Futuros

Trabalhos Futuros

- Versão final para memória distribuída
- Implementação de novos tipos de modelos (Ex. energia)
- Testes com o Omnet++ e horizon
- Testes com Sistemas Operacionais que simulam SMPs sobre um grupo de máquinas (Ex. Kerrighed)

Dúvidas



Agradecimentos

LABORATÓRIO iMobilis Computação Móvel

UFOP

UNIVERSIDADE FEDERAL DE ALAGOAS

SensorNet UFAL