TODO: Title

Matheus B. Nascimento[1], **Wisllay Vitrio**[1]

[1] Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – CEP 74.001-970 – Goiânia – GO – Brasil

Abstract. *Abstract. Abstract. Abstract. Abstract. Abstract. Abstract. Abstract. Abstract. Abstract. Abstract.*

Resumo. *Resumo. Resumo. Resumo. Resumo. Resumo. Resumo. Resumo. Resumo. Resumo. Resumo. Resumo.*

1. Introdução

Espaço de tuplas é um conceito de memória associativa para computação distribuída/-paralela. Desenolvido por David Gelernter na Universidade de Yale, teve sua primeira implementação na linguagem de coordenação "Linda" (homenagem a uma atriz pornô de nome Linda Lovelace, assim como o nome da linguagem "Ada" é uma homenagem à Ada Lovelace [**?**]).

Provê um repositório de tuplas que pode ser acessado concorrentemente por zero ou mais processos. Por ser baseado em memória associativa, as tuplas são acessadas por seu conteúdo e não por endereços. As tuplas não estão ligadas à nenhum processo, e qualquer um deles pode inserir, ler ou remover tuplas. Este desacoplamento total entre as partes integrantes do sistema provido pelo espaço de tuplas é sua principal vantagem.

Várias linguagens têm implementações de espaço de tuplas, sendo a especificação para Java, JavaSpaces [**?**], a mais famosa. Como parte da tecnologia Jini, JavaSpaces é utilizado em serviços de finanças e telecomunicações para alcançar escalabilidade utilizando processamento paralelo. Por outro lado, a tecnologia Jini como um todo não é sucesso comercial, e o JavaSpaces não é amplamente utilizado.

2. Proposta

Proposta

3. Implementação

Implementação

```
// Stops retransmission attempts on remote station manager (RTS/
    CTS and Data)
Config::SetDefault("ns3::WifiRemoteStationManager::MaxSsrc",
    UintegerValue(0));
Config::SetDefault("ns3::WifiRemoteStationManager::MaxSlrc",
    UintegerValue(0));
```

```
// Create default PHY and Channel
YansWifiChannelHelper chan = YansWifiChannelHelper::Default();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default();
// Set channel
phy.SetChannel(chan.Create());
```

```
// Set Reception Gain to 0
phy.Set("RxGain", DoubleValue(0));
// Disable signal detection so the sending devices don't backoff
phy.Set("EnergyDetectionThreshold", DoubleValue(0));
// Stop the PHY layer from declaring 'CCA_BUSY'
phy.Set("CcaMode1Threshold", DoubleValue(0));
```

```
// Create and setup mobility
MobilityHelper mob;
mob.SetPositionAllocator("ns3::GridPositionAllocator",
        "MinX", DoubleValue(0),
        "MinY", DoubleValue(0),
        "DeltaX", DoubleValue(gridDeltaX),
        "DeltaY", DoubleValue(gridDeltaY),
        "GridWidth", UintegerValue(gridWidth),
        "LayoutType", StringValue("RowFirst"));
mob.SetMobilityModel("ns3::RandomWalk2dMobilityModel",
        "Bounds", RectangleValue(Rectangle(-walkX, walkX, -walkY,
            walkY)));
```

4. Caso de teste

5. Resultados

Resultados

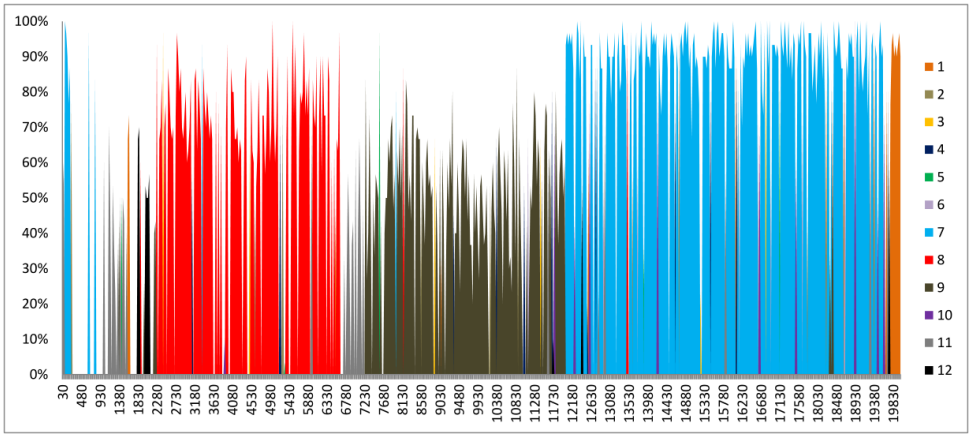

Figura 1. Variação de recebimento durante a execução do algoritmo QL

6. Conclusão

Conclusão

Referências

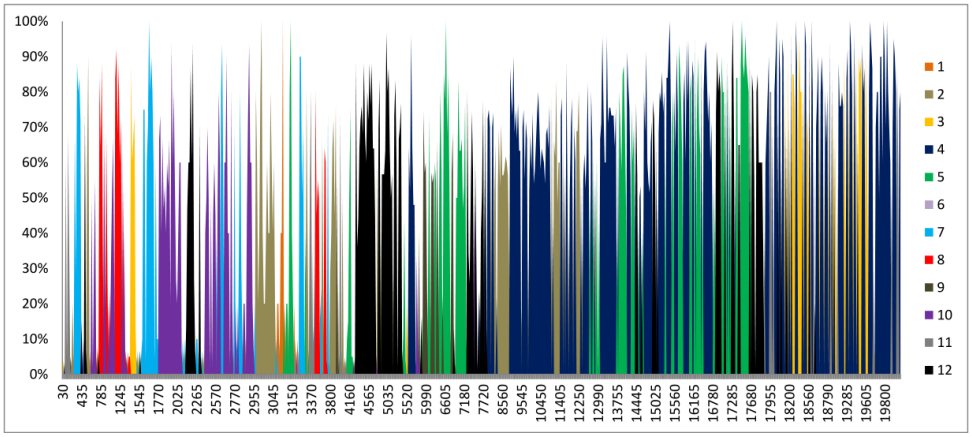

Figura 2. Variação de recebimento durante a execução do algoritmo ES