Universidade Federal de Paraíba

Centro de Informática

Programa de Pós-Graduação em Informática

Um Ambiente para Simulação e Testes de Comunicação entre Multi-Robôs através de Cossimulação

Thiago José Silva Oliveira

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Computação Distribuída

Orientador: Prof. Dr. Alisson Vasconcelos de Brito

João Pessoa, Paraíba, Brasil

O48u Oliveira, Thiago José Silva.
Um ambiente para simulação e testes de comunicação entre multi-robôs através de cossimulação / Thiago José Silva Oliveira.- João Pessoa, 2016.
80f. : il.
Orientador: Alisson Vasconcelos de Brito
Dissertação (Mestrado) - UFPB/CI
1. Informática. 2. Computação distribuída. 3. Sistemas Multi-Robô. 4. Cossimulação. 5. OMNeT++. 6. *High Level Architecture*.

UFPB/BC CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de **TIAGO JOSÉ SILVA OLIVEIRA**, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 26 de fevereiro de 2016.

Ao vigésimo sexto dia do mês de fevereiro do ano de dois mil e dezesseis, às nove horas, no Centro de Informática - Universidade Federal da Paraíba (unidade Mangabeira), reuniram-se os membros da Banca Examinadora constituída para julgar o Trabalho Final do **Sr. Tiago José Silva Oliveira** vinculado a esta Universidade sob a matrícula 2013115986, candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa *"Sinais, Sistemas Digitais e Gráficos"*, do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: **Dr Alisson Vasconcelos de Brito (PPGI-UFPB),** Orientador e Presidente da Banca, **Dr Tiago Pereira do Nascimento (PPGI-UFPB),** Examinador Interno e **Dr Francisco Petrônio Alencar de Medeiros (IFPB)**, Examinador Externo à Instituição. Dando início aos trabalhos, o professor Presidente da Banca cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado *"Um Ambiente para Simulação e Testes de Comunicação entre Multi-Robôs através de Cossimulação"*. Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "*aprovado*". Assim sendo, eu, Clauirton de Albuquerque Siebra, Coordenador do Programa de Pós-Graduação em Informática - PPGI, lavrei a presente ata que vai assinada por mim e pelos membros da Banca Examinadora.

João Pessoa, 26 de fevereiro de 2016.

Ruy Alberto Pisani Altafim
Vice-Coordenador do Programa de Pós-Graduação em Informática
SIAPE 1971934

Clauirton de Albuquerque Siebra

Prof Dr Alisson Vasconcelos de Brito
Orientador (PPGI-UFPB)

Prof Dr Tiago Pereira do Nascimento
Examinador Interno (PPGI-UFPB)

Prof Dr Francisco Petrônio Alencar de Medeiros
Examinador Externo à Instituição (IFPB)

“É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória, nem derrota” (Theodore Roosevelt)

Agradecimentos

À minha mãe amada, Regina, que sempre me apoiou em todos os momentos e que sempre foi fundamental em minha vida.

À toda minha família, por todo o suporte e carinho.

Aos meus amigos.

A Deus.

UM AMBIENTE PARA SIMULAÇÃO E TESTES DE SISTEMAS MULTI-ROBÔ ATRAVÉS DE COSSIMULAÇÃO

Resumo

Sistemas Multi-Robôs (SMR) consistem em múltiplos robôs interagindo, cada um executando uma estratégia de controle específica, que não é conduzida de forma centralizada. Alguns desafios surgiram da necessidade de desenvolver produtos que levem o mundo real em consideração. Fazer com que ferramentas, metodologias e equipes de diferentes áreas possam trabalhar juntas não é uma tarefa simples de ser realizada. Cossimulação representa uma técnica para validação de sistemas heterogêneos. Seu princípio fundamental é prover suporte à execução de diferentes simuladores de forma cooperativa. Um dos padrões para tal é conhecido como *High Level Architecture* (HLA), que é um padrão descrito no IEEE 1516 e tem sido desenvolvido para dispor uma arquitetura para modelagem e simulação distribuídos. Utilizando HLA, vários simuladores e aplicações reais podem ser simulados juntos. Sendo assim, este trabalho apresenta um projeto para simulação de Sistemas Multi-Robôs (SMR) utilizando ROS cossimulado com um simulador de redes de computadores, o OMNeT++ através do HLA. Seu principal objetivo é tornar as simulações mais próximas da realidade, onde os dados irão ser trocados através de uma rede simulada, como se tivéssemos robôs reais interagindo através de uma rede convencional. Para tal, foi desenvolvida a interface entre o ambiente ROS e o OMNeT++ com o HLA. Experimentos demonstraram que a perda de pacotes foi simulada corretamente, adicionando ao ambiente mais realismo

Palavras-Chave: Sistemas Multi-Robô, Cossimulação, OMNeT++, *High Level Architecture*

An Environment for Simulation and Tests of Multi-Robot Systems Using Co-Simulation

Abstract

Multi-Robot System (MRS) consisting of multiple interacting robots, each running a specific control strategy, which is not driven centrally. Technical challenges arise from the need to develop complex, *software*-intensive products that take the constraints of the physical world into account. Make tools, methodologies and teams from different fields can work together is not an easy task to accomplish. Co-simulation represents on technique of validation in heterogeneous systems. Its fundamental principle is to provide support to execute different simulators in a cooperative way. A known standard is the High Level Architecture (HLA) that is a pattern described in IEEE 1516 series and has been developed to provide a common architecture to distributed model and simulation. Using HLA, several simulators and real applications could be simulated together. That way, this work presents a project for Multi-Robot Systems (SMR) simulation using ROS co-simulation with a network simulator, the OMNeT++, using HLA. The main goal is make the simulations more realistic, where the data exchange will be performed by using a simulated network, as if we had real robots interacting through a conventional network. To this end, an interface was developed between ROS and OMNeT++ using HLA. Experiments demonstrate that the packet losses were correctly simulated, adding realism to simulations.

Key-Words: Multi-Robot Systems, Co-simulation, OMNeT++, High Level Architecture

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

ASyMTRe – *Automated Synthesis of Multi-Robot Task solutions through software Reconfiguration*

BLE - *Broadcast of Local Eligibility*

HLA - *High Level Architecture*

MRCF – *Multi-Robot Coalition Formation*

MRTA – *Multi-Robot Task Allocation*

SRM – *Sistemas Multi-Robô*

FOM – *Federate Object Model*

CSV – *Comma-Separated Values*

UDP - *User Datagram Protocol*

1. INTRODUÇÃO

1.1. APRESENTAÇÃO

Sistemas Multi-Robô (SMR) consistem em múltiplos robôs interagindo, cada um executando uma estratégia de controle específica, que não é conduzida de forma centralizada (CALISKANELLI et al., 2014) e podem ser distribuídos ou centralizados. Ao passar dos anos, as pesquisas em SMR passaram a dar mais foco na investigação de problemas que envolvem vários robôs, ao invés daqueles que poderiam ser resolvidos por um único robô. Em SMRs distribuídos não existe um mecanismo de controle central. Ao invés disso, cada robô opera independentemente no âmbito de detecção e controle local, com comportamento coordenado em nível de sistema, decorrente de interações locais entre os robôs e entre o ambiente (LERMAN et al., 2006).

Robôs trabalhando em conjunto de forma a resolver um problema de maneira cooperativa, fez com que os sistemas se tornassem mais complexos. Essa complexidade tem duas fontes primárias: times maiores e heterogeneidade dos robôs e tarefas (GERKEY e MATARIC, 2004). Em outras palavras, aumenta à medida que os times de robôs se tornam maiores e são compostos por vários tipos, ou seja, quando os times possuem maior heterogeneidade.

O desenvolvimento de *software* para robôs é uma disciplina exigente e que envolve desafios técnicos que surgem da necessidade de desenvolver *software* complexo que lidam com as restrições do mundo físico (BROENINK e GROOTHUIS, 2010). O desenvolvimento de Sistemas Multi-Robô autônomos de alto desempenho requer intensa experimentação de forma realista, controlada e que possa ser repetida (BUCK, BEETZ e SCHMITT, 2002). Dessa forma, tudo deve ser muito bem modelado e simulado.

Em 2007, um grupo de pesquisadores criou um framework de código aberto para robôs chamado *Robot Operating System* (ROS) (QUIGLEY et al, 2009). É um framework flexível para projetar robôs, provendo uma coleção de ferramentas, bibliotecas e convenções que visam simplificar a criação de robôs com comportamentos complexos e robustos através de uma grande variedade de plataformas. Entretanto, fazer com que ferramentas, metodologias e equipes de diferentes áreas possam trabalhar juntas não é

uma tarefa simples de ser realizada. Comunidades acadêmicas e industriais desenvolveram padrões e ferramentas com esse objetivo.

Para NICOLESCU et al. (2007), a cossimulação representa uma técnica de validação de sistemas heterogêneos. De acordo com Souza et al. (2003), o princípio fundamental da cossimulação é prover suporte à execução de diferentes simuladores de forma cooperativa. Isto permite a união de simuladores heterogêneos com diferentes modelos de execução.

Um padrão conhecido para cossimulação é o *High Level Architecture* (HLA). O padrão HLA tem como objetivo prover uma forma mais flexível para interconexão entre diferentes simuladores. Ele é definido no padrão IEEE 1516 e vem sendo desenvolvido para prover uma arquitetura comum para modelagem e simulação distribuídas.

Até este ponto, SMRs poderiam ser simulados através da utilização de ROS e interconectados com outros simuladores e/ou sistemas reais através da cossimulação (usando HLA, por exemplo). Entretanto, as simulações com ROS não conseguem simular eficientemente a parte de comunicação, essencial para o bom funcionamento desse tipo de sistema. Por outro lado, com a utilização de simuladores de rede (como o NS-2, NS-3, OMNeT++, entre outros), SMRs podem ser simulados levando em consideração os aspectos relacionados a comunicação em rede, tornando as simulações e seus resultados mais realistas.

1.1.1. OBJETIVOS

Esta dissertação apresenta um projeto para simulação de SMRs utilizando ROS, cossimulação e simulação de redes de computadores. Seu principal objetivo é tornar as simulações mais próximas da realidade, onde os dados irão ser trocados através da utilização de um simulador de Redes chamado OMNeT++. Isto irá possibilitar a utilização de latência na comunicação, perda de pacotes, utilização de protocolos de rede e outros aspectos relacionados a redes de computadores. Dessa forma, as simulações serão mais próximas da realidade, como se estivéssemos simulando robôs reais trocando informações em uma rede de computadores.

1.1.2. METODOLOGIA

Como forma de alcançar os objetivos almejados, o desenvolvimento deste trabalho foi dividido em algumas etapas. Inicialmente cada uma das ferramentas que fazem parte do projeto foi estudada separadamente. São elas: o *Robot Operation System* (ROS), para simulação de Sistemas Multi-Robô, o CERTI, como ferramenta para *High Level Architecture* (HLA) e o OMNeT++, como simulador de rede. Essas ferramentas serão mostradas em mais detalhes posteriormente.

Uma vez concluída a primeira etapa, o passo seguinte foi desenvolver o ambiente de simulação e testes, objetivo deste trabalho. Para isso, o CERTI foi utilizado como ferramenta de HLA (em C++), o OMNeT++, como simulador de Rede (em C++) e o Stage (ROS) (em Python), como ambiente de simulação de SMRs. Uma vez que já se conhece as ferramentas de forma isolada, se fez necessário integrá-las para que trabalhassem de forma coordenada. Sem isso, não seria possível criar as simulações que serão mostradas adiante. Nesta etapa, o ROS e o OMNeT++ foram integrados através da utilização do CERTI como plataforma de HLA, possibilitando a troca de informações entre os simuladores.

Uma vez que o ambiente estava em funcionamento, testes foram feitos para provar que a utilização de um simulador de rede torna as simulações de SMRs mais próximas da realidade. Os testes foram feitos utilizando simulações com dois robôs e com tempos de 50, 100, 200 e 400 segundos, totalizando 20 simulações (5 para cada tempo de simulação).

Após validar a arquitetura, testes mais detalhados foram feitos, onde dois robôs foram simulados. Isso foi feito considerando aspectos relacionados a redes, como latência de comunicação e taxa de perda de pacotes. Foram feitos testes também com dois robôs, mas com durações de 300, 600 e 1200 segundos, totalizando 75 simulações, sendo 25 para cada tempo de execução, sendo 5 para cada percentual de perda de pacotes (1%, 2%, 3%, 4%, e 5%) e com latência de 59 milissegundos. O objetivo desta etapa é provar que os elementos provenientes do simulador de redes, o OMNeT++, tornam as simulações mais realistas, uma vez que o comportamento dos robôs simulados no ROS foi alterado devido ao OMNeT++.

Esses testes envolvem robôs se movimentando de forma coordenada, onde existem robôs mestres, que se movem de forma independente partindo de um ponto X_0Y_0 para X_1Y_1

e a repassam sua posição atual a cada ciclo de simulação para guiar os demais robôs, e escravos, que se movimentam de acordo com a movimentação do robô mestre. A cada ciclo de simulação, os robôs escravos utilizam a informação repassada pelo mestre e passam a se mover para a localização atual do robô mestre. Esse processo será mostrado com mais detalhes posteriormente.

1.2. ORGANIZAÇÃO DO TRABALHO

Essa dissertação é composta por seis capítulos. O capítulo 02 traz a fundamentação teórica, introduzindo os principais conceitos que envolvem este trabalho e servem como base para o entendimento do mesmo. O capítulo 03 trata de trabalhos relacionados, mostrando aspectos comuns e divergentes entre eles e o trabalho aqui proposto.

O ambiente para simulação e teste de SMRs através da utilização de cossimulação é mostrado em detalhes no capítulo 04, assim como os testes de validação da arquitetura. O capítulo 05 mostra os experimentos e resultados obtidos com a utilização da arquitetura aqui proposta. Por fim, o capítulo 06 apresenta as considerações finais e perspectivas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1. SISTEMAS MULTI-ROBÔ

2.1.1. VISÃO GERAL

Sistemas Multi-Robô (SMR) consistem em múltiplos robôs interagindo, cada um executando uma estratégia de controle específica, que não é conduzida de forma centralizada (CALISKANELLI et al., 2014).

Para LEVENT et al. (2008), um Sistema de Robôs em Rede (*Network Robot System*) é qualquer sistema distribuído que consista de vários robôs em rede e outros dispositivos que, como um todo, seja capaz de interagir com o ambiente através do uso da percepção e ação para desempenhar tarefas.

Neste trabalho utilizamos o termo Sistema de Robôs em Rede como sendo sinônimo de Sistemas Multi-Robô. Em um Sistema Multi-Robô não existe um mecanismo central de controle. Ao invés disso, cada robô opera independentemente sob sensoriamento local e controle, com comportamento coordenado em nível de sistema decorrente de interações locais entre robôs (KRISTINA et al., 2006).

Essa área de pesquisa surgiu no final da década de 1980, quando vários pesquisadores começaram a investigar seus problemas (ARAI et al., 2002). Esse é um campo altamente interdisciplinar, dividindo terreno com campos de pesquisas como Teoria dos Jogos, Programação Linear, Psicologia, Inteligência Artificial Distribuída, Física, Matemática e Biologia (VIG e ADAMS, 2005).

No início das pesquisas nesta área, fazer com que robôs se movimentassem ao redor de um ambiente desestruturado a uma velocidade razoável era considerado algo inovador. Porém, ao passar dos anos, avanços em mecanismos de *hardware*, arquiteturas de *software* e algoritmos resultaram em boa capacidade móvel para sistemas robóticos e fizeram com que mais atenção fosse dada ao estudo desse tipo de sistema (KRISTINA et al., 2006). Outro fator que contribuiu foi o avanço na infraestrutura de rede e internet, reduzindo-se a latência nas comunicações e a melhora na arquitetura (LEVENT et al., 2008). Além disso, SANTOS (2014), defende que Sistemas Multi-Robô têm se tornado uma das mais importantes áreas de pesquisa em robótica devido ao desafio natural da

pesquisa envolvida e as múltiplas aplicações em áreas como redes de sensor móveis e autônomas, vigilância de edifícios, transporte de grandes objetos, monitoramento da poluição aérea e aquática, detecção de incêndios florestais, sistemas de transporte ou buscas e resgate em incidentes de larga escala, entre outros.

Destacam-se entre as aplicabilidades viáveis de Sistemas Multi-Robô:

- **Aplicações de Segurança e construções inteligentes**: Vigilância de edifícios (SANTOS, 2014), Localização e classificação do comportamento das pessoas, situações perigosas ou atos de ameaça e ações podem ser tomadas para manter a segurança (LEVENT et al., 2008);

- **Arquitetura, alocação de tarefa e controle:** Desenvolvimento de arquiteturas, capacidade de planejamento de tarefas e controle através da investigação de problemas com ações de seleção, delegação de autoridade e controle, estruturas de comunicação, coerência entre ações locais, resolução de conflitos e outros problemas relacionados;

- **Assistência a idosos**: Assistência a idosos em suas casas ou em retiros, com o objetivo de prover suporte físico e cognitivo, para facilitar comunicação e monitoramento com prestadores de cuidados, além de detectar e responder a emergências (LEVENT et al., 2008);

- **Automação flexível e manufatura colaborativa**: Utilização no controle em ambientes industriais (LEVENT et al., 2008);

- **Comportamento biológico:** O paradigma baseado em comportamento influenciou as pesquisas em Sistemas Multi-Robô. Muitas pesquisas examinaram as características sociais de insetos (formigas e abelhas) e animais (aves) para desenvolver comportamento semelhante em SMRs (ARAI et al., 2002);

- **Serviços de Robôs em rede**: Em aplicações como coleta de lixo, entrega e logística (LEVENT et al., 2008);

- **Monitoramento em larga escala**: Desenvolvendo redes de sensores autónomos (SANTOS, 2014, LEVENT et al., 2008) com a habilidade de *auto-deploy*, configuração e reparo, capacidade de monitoramento de grandes áreas sem

supervisão em busca de poluição, ameaças ambientais (CALISKANELLI et al., 2014; LEVENT et al., 2008), detecção de incêndios florestais e outras situações perigosas (LEVENT et al., 2008) através do desenvolvimento de algoritmos fundamentalmente distribuídos (ARAI et al., 2002);

- **Busca e resgate cooperativos**: No transporte ou manipulação de grandes objetos [Santos, 2014; ARAI et al., 2002], procura de pessoas e, eventualmente, prestam suporte no resgate lidando com emergências (LEVENT et al., 2008).

Uma vantagem na utilização de SMRs, ao invés de robôs isolados, é que um time de robôs provê redundância e contribui cooperativamente para resolver determinada tarefa ou pode desempenhá-la de forma mais confiável, rápida ou barata quando comparado com um único robô (ARAI et al., 2002). Para isso, o projetista precisa definir que robô deverá realizar que tarefa e quando (KRISTINA et al., 2006). Cada robô, individualmente falando, deve conhecer e saber desempenhar tarefas que irão contribuir para o funcionamento do sistema. Esse processo de atribuir tarefas a robôs individuais, dada uma tarefa a nível de sistema, é chamado alocação de tarefa (*task allocation*) e é funcionalidade chave necessária para qualquer Sistema Multi-Robô (KRISTINA et al., 2006).

A utilização de SMRs requer que ações sejam tomadas de forma cooperativa e não isoladamente. Por sua vez, atingir o comportamento cooperativo deve contar com uma infraestrutura que engloba conceitos como heterogeneidade/homogeneidade dos robôs, a capacidade de um determinado robô para reconhecer e modelar outros robôs, além de capacidade de comunicação (Santos, 2014). Essa cooperação não implica necessariamente em interação direta entre robôs, desde que haja um ganho geral para o sistema. Isso fica claro na definição de cooperação em robótica, mostrada em SANTOS (2014): “Dada alguma tarefa projetada por um designer, um sistema Multi-Robô mostra comportamento cooperativo se, devido a algum mecanismo, exista um aumento na utilidade total do sistema”. É importante ressaltar que um SMR precisa ser tolerável a falhas, ou seja, caso um ou vários robôs parem de operar, o sistema precisa continuar em funcionamento. Com cooperação, vários robôs podem trocar seus papéis dinamicamente de acordo com sua posição física ou restrições do ambiente ou podem adaptar seu posicionamento de acordo com cada sensor específico e/ou sistemas atuadores (SANTOS, 2014).

Apesar dos avanços nos últimos anos, Sistemas Multi-Robô permanecem complexos, sendo um grande desafio o controle e a coordenação desses sistemas (Santos, 2014). Além disso, com o aumento no tamanho dos Sistemas Multi-Robô, a complexidade do projeto de abordagens cresce devido ao aumento de demanda na largura de banda na comunicação e em habilidades computacionais dos robôs (KRISTINA et al., 2006).

O desenvolvimento de robôs capazes de tomar decisões sofisticadas, tanto individualmente quanto em time, é bastante complexo (SANTOS, 2014), já que, devido ao sensoriamento local de cada robô, nenhum robô tem uma visão completa do estado do ambiente como um todo. Dada essa incerteza, acrescentado de informações com ruídos, cada robô precisa tomar decisões isoladamente sobre que ações desempenhar e quando, sem conhecimento completo do que os outros robôs fizeram, estão fazendo ou irão fazer no futuro (KRISTINA et al., 2006).

Esses problemas tornam o desenvolvimento e manutenção de SMRs algo complexo e passível a erros (SANTOS, 2014). Todos esses fatores, quando em conjunto, mostram a importância de simular a comunicação entre os SMRS, uma vez que ela se mostra fundamental nesse processo e que os problemas decorrentes na comunicação influenciam diretamente o comportamento e resultados nos SRMs. Até então, as simulações em SMRs não levam isso em consideração, fazendo com que seus resultados não sejam fiéis ao que ocorre quando testados em ambiente real.

Em seu trabalho, SANTOS (2014) elícita as principais dificuldades em desenvolver Sistemas Multi-Robô:

- Rápidas mudanças nos sensores, aturadores e tecnologias computacionais aumentam a pressão sobre alcançar novas capacidades robóticas. Novos sensores e atuadores necessitam de algoritmos de processamento cada vez mais complexos;

- Sistemas robóticos são hierarquicamente distribuídos. Sensores e atuadores são distribuídos em subsistemas interconectados e essa distribuição aumenta muito em sistemas Multi-Robô;

- Robôs, componentes e processos precisam interagir de maneira eficiente;

- Cada robô individual requer seu conjunto de sensores, processamento, ações e habilidades;

- O time de robôs precisa continuar a operar de forma cooperativa mesmo com falhas individuais e reinicializações assíncronas.

Apesar das dificuldades mostradas, é esperado avanço significativo nos seguintes aspectos (LEVENT et al., 2008):

- **Alto nível de habilidades cognitivas**: Habilidade de adquirir cooperação em planejamento, tomada de decisão e modelagem de ambiente;

- **Autonomia**: Robôs precisam estar aptos a cumprir tarefas sem supervisão, de forma a obter vantagens sobre capacidades individuais e garantir robustez do sistema, mesmo se tratando de robôs heterogêneos.

- **Representação e interpretação de comportamento**: Tanto a representação do ambiente quanto a interpretação são feitas cooperativamente e de forma distribuída. Além disso, abordagens de aprendizagem podem ser desenvolvidas para aprender coletivamente e aprender comportamentos coletivos.

- **Interação homem/robô**: Melhores interfaces para controlar e interagir com Sistemas de Robôs irão aumentar a usabilidade e trazer aplicações novas e mais abrangentes. Por um lado, aumentada a capacidade de percepção cooperativa tornou possível interações mais eficientes com pessoas, através do entendimento de diferentes tipos de sinais vindos de uma ou várias pessoas; por outro lado, um cenário com múltiplos usuários interagindo com vários robôs traz novos desafios que iram impactar significativamente a interação homem/robô.

- **Avaliação de desempenho e benchmarking**: Métodos adequados precisam ser estabelecidos de forma que avalie adequadamente os recursos dos métodos propostos. Isso necessita de uma metodologia de desenvolvimento para desenvolver procedimentos de validação experimental, configurações experimentais compartilhadas (simulações e robôs reais) e benchmarks.

> Como já dito anteriormente, SMRs precisam ser muito bem modelados e testados. Dessa forma, este trabalho foca na parte de simulação de SMRs, de forma que possam ser validados de maneira adequada, enfatizando a comunicação e os problemas decorrentes dela, tornando seus resultados mais próximos da realidade.

Ao passar dos anos, Sistemas Multi-Robô foram se tornando cada vez mais complexos. Essa complexidade se dá devido ao aumento no tamanho do número de robôs que compõem um SMR e à diversidade de robôs interagindo em um mesmo SMR. Nesse sentido, os problemas conhecidos como alocação de tarefas (*task allocation*) e coordenação (*coordination*) se tornaram problemas fundamentais na área (SHIROMA e CAMPOS, 2009).

O problema de alocação de tarefas é um problema em determinar um mapeamento entre robôs e tarefas (TANG e PARKER, 2007) e é considerado um desafio devido à natureza imprevisível dos ambientes, falhas de sensores, falhas dos robôs e mudanças dinâmicas nos requisitos das tarefas, etc. (VIG e ADAMS, 2006). Esse problema ficou conhecido como *Multi-Robot Task allocation* (MRTA) e segundo (GERKEY e MATARIC, 2004), ele é baseado no seguinte questionamento: uma vez estando em um time, qual robô deve realizar que tarefa de forma a alcançar cooperativamente um objetivo global?

Esse questionamento também serviu como base para este trabalho, uma vez que a comunicação é fator determinante para um time de robôs conseguir agir cooperativamente com o intuito de atingir um objetivo em comum. Problemas na comunicação em SMRs impactam diretamente o comportamento dos robôs quando em conjunto, uma vez que eles precisam trocar informações para realizar as tarefas da forma mais adequada e otimizada possível.

Outra questão motivadora é defendida por TANG e PARKER (2005) e trata de como determinar os comportamentos ideais para cumprir determinada tarefa quando a definição de como resolver uma tarefa é dependente do conjunto de robôs e seus recursos sensoriais, perceptivos e motores. Cada um desses fatores é impactante no que diz respeito ao melhor conjunto de atribuições possível dentro de um universo de possibilidades. Robôs pertencentes a um mesmo time podem compartilhar características e

serem diferenciados em particularidades. Isso faz com que essa escolha não seja algo trivial.

2.1.2. COMUNICAÇÃO EM SMRS

O sucesso no controle e coordenação de robôs em um SMR depende de uma comunicação efetiva. Controladores de robôs devem ser robustos no que diz respeito a incertezas e variações nos sensores e atuadores; controladores devem explorar informações distribuídas em uma rede e também devem lidar com incertezas na comunicação (YE et al. 2001).

A informação que é compartilhada com cada robô em um time de robôs móveis tipicamente inclui propriedades percebidas a respeito de objetos ao redor, adquiridas através de sensores. Integrando essas percepções é possível construir uma percepção global mais aproximada e completa (SANTOS, 2014). Isso faz com que seja possível que robôs tomem decisões baseadas em informações obtidas através de outros robôs. Nesse tipo de sistema, a rede se torna parte do ambiente dos robôs e pode contribuir para seu sucesso ou falha (YE et al., 2001).

A mobilidade existente nos robôs requer que seja utilizada uma comunicação sem fio, o que traz algumas restrições, como seu limite de alcance, taxa de transferência (*throughput*) e disponibilidade. A quantidade de informações compartilhada, o número de robôs pertencentes ao time e sua distribuição no espaço, são alguns dos fatores que influenciam diretamente a escolha do tipo de comunicação sem fio (SANTOS, 2014).

Exemplos de SMRs que requerem compartilhamento de informações sem fio podem ser encontrados nos robôs de escavação e cirurgia, onde tanto o operador quanto um computador podem operar o sistema a partir de uma localização remota (ALHARTHI, 2014). Outros exemplos já foram mostrados anteriormente, como veículos, que podem estar conectados através de uma rede e trocar informações sobre as condições da estrada, tráfego e direções (POOVENDRAN et al., 2012).

Segundo SANTOS (2014), os padrões geralmente utilizados em SMRs são: o IEEE 802.11, IEEE 802.15.1 e o IEEE 802.15.4. A seguir, esses três padrões citados serão mostrados. Neste trabalho, o padrão escolhido para utilização foi o IEEE 802.11, uma vez que o simulador de Redes já possui este protocolo nativamente e a implementação dos

demais padrões, até o momento da publicação, foi inviável. Entretanto, a utilização dos demais protocolos faz parte dos trabalhos futuros, uma vez que é interessante testar a arquitetura proposta utilizando diferentes protocolos de comunicação.

2.1.2.1. IEEE 802.11

O padrão IEEE 802.11 (IEEE 802.11 WORKING GROUP, 2012) consiste de uma série de técnicas de modulação através do ar que utilizam um mesmo protocolo básico. Os mais populares são o IEEE 802.11a, IEEE 802.11b e as variantes da IEEE 802.11g. Para ERGEN (2002), os principais objetivos por trás do desenvolvimento desses padrões foram:

- Prover serviços anteriormente suportados apenas por redes cabeadas;
- Alta taxa de transferência (throughput);
- Alta confiabilidade;
- Comunicação contínua em rede;

Uma vez que a primeira versão do padrão 802.11 foi criada em 1997, o *Working Group* (WG) recebeu feedback e notou que muitos produtos não possuíam um grau adequado de compatibilidade que os consumidores esperavam (HIERTZ et al., 2010). Isso fez com que, em 1999, houvesse a criação de um programa de certificação liderado por um grupo chamado *Wireless Ethernet Compatibility Alliance* (WECA), posteriormente renomeado para *Wi-Fi Alliance* (WFA), em 2003. O WFA foi responsável pela popularização do padrão que passou a ser comumente conhecido como Wi-Fi.

No IEEE 802.11 existem três frequências permitidas, mas as mais utilizadas delas são a *Industrial, Scientific and Medical* (ISM) a 2.4GHz, a *Unlicensed National Information Infrastructure* (U-NII) a 5GHz e a terceira é disponível apenas nos Estados Unidos, operando a uma frequência entre 3.65GHz e 3.70GHz.

Segundo SANTOS (2014), IEEE 802.11b e IEEE 802.11g usam a frequência ISM e podem ocasionalmente sofrer interferências de outros equipamentos que emitem na mesma banda, como micro-ondas, telefones sem fio e dispositivos Bluetooth. Um outro problema diz respeito ao consumo de bateria. De fato, o padrão foi desenvolvido para suportar mobilidade, o que geralmente requer baterias, fazendo com que o consumo de energia na comunicação wireless seja sempre um fator importante.

Com o passar dos anos, foram lançadas diversas versões do padrão 802.11, entre elas destaco a 802.11p, para comunicação entre veículos, a 802.11n, para altas taxas de transferência, e a 802.11u, para redes de grande porte.

2.1.2.2. IEEE 802.15.1

O padrão 802.15.1, também conhecido como Bluetooth, é baseado em um sistema de wireless à radio projetado para distâncias curtas e dispositivos baratos para substituir cabos para componentes periféricos de computadores, como mouses, teclados, impressoras, controles, etc. (LEE, SU e SHEN, 2007). Este conjunto de aplicações é conhecido como *Wireless Personal Area Network* (WPAN).

A tecnologia Bluetooth suporta fluxo assíncrono de dados e streams de áudio com links de até 1MB/s. Ela opera na banda ISM de 2.4 GHz utilizando rádios de baixa frequência e serve como uma interface universal amigável para o usuário e de baixo custo que substitui uma infinidade de cabos proprietários que pessoas precisam carregar e utilizar para conectar seus dispositivos pessoais (BISDIKIAN, 2001).

2.1.2.3. IEEE 802.15.4

O padrão 802.15.4, conhecido como ZigBee, é um padrão unicamente desenvolvido para *Low Rate Personal Area Networks* (LRWPANs) (ZHENG e LEE, 2006). Os principais objetivos desse padrão são a comunicação com eficiência energética para baixo fluxo de dados, geolocalização, baixo custo de rede sem fio e oferecer conectividade wireless a nível de dispositivo. Sua utilização é para dispositivos simples com consumo mínimo de energia e que operem a uma distância máxima de 10 metros.

Alguns campos, como indústria, agricultura, residencial, sensores médicos e outros, são mais flexíveis quanto a vazão de dados (throughput). Entretanto, essas aplicações necessitam substancialmente de um baixo consumo de energia (PARK et. Al, 2005). Daí surgiu a necessidade de um padrão como o 802.15.4.

2.1.3. CYBER PHYSICAL SYSTEMS

Cyber-Physical System (CPS) é o termo utilizado para identificar sistemas que integram processos físicos com computação e comunicação (ALHARTHI, 2014). Esse

termo surgiu por volta de 2008, no trabalho feito por LEE et al., e vem sendo muito comentado desde então.

Segundo WOLF (2009), desde a invenção do microprocessador na década de 1970, computadores embarcados têm sido projetados para vários tipos de sistemas, mas os desenvolvedores vêm desenvolvendo este trabalho de maneira empírica, sem base teórica e os CPSs foram criados como tentativa de corrigir esta deficiência.

Para BAHETI e GILL (2011), oportunidades e desafios de pesquisa incluem o projeto e desenvolvimento da próxima geração de aviões e veículos espaciais, veículos híbridos, condução autônoma de veículos e próteses que permitam enviar sinais ao cérebro com o intuito de controlar objetos físicos. LEE et al. (2008) também inclui temas como controle de tráfego e segurança, sistemas automotivos avançados, conservação de energia, controle de ambientes, controle de infraestrutura crítica, recursos aquáticos, entre outros.

Assim como nos SMRs, confiabilidade é um termo crítico para os CPSs, uma vez que não se espera que um sistema de controle de tráfego, dispositivo médico ou de condução autônoma falhe durante operação. É natural pensar que falhas nesse tipo de sistema causariam danos enormes. O problema é que o mundo físico não é um ambiente totalmente previsível (LEE et al., 2008), o que exige que CPSs estejam aptos a lidar com condições inesperadas e a ser adaptáveis.

A inovação e desenvolvimento de CPSs requer que cientistas de computação e profissionais de redes trabalhem com especialistas em várias áreas, como engenharia de controle, processamento de sinais, engenharia civil, engenharia mecânica e biologia (RAJKUMAR, 2010).

Nesta dissertação, o termo CPS não será utilizado apenas porque não serão feitas simulações utilizando sistemas que sejam capazes de interagir com humanos. Porém, como este termo tem ganhado importância na comunidade acadêmica, não pôde ser negligenciado, uma vez que a tendência é que os SMRs passem a se tornar CPSs, como nos casos de SMRs de assistência médica, resgate e outros exemplos citados por BAHETI e GILL (2011) e LEE et al. (2008).

2.1.4. ROBOT OPERATION SYSTEM

Neste trabalho, o conceito de Sistemas Multi-Robô será utilizado como base, de forma que possamos realizar simulações onde haja comunicação entre os diferentes robôs que fazem parte dos sistemas. Para tal, é utilizado o ROS (*Robot Operation System*), um framework flexível e de código aberto para escrever *software*s para robôs. Segundo QUIGLEY et al. (2009), as principais características do ROS são:

- Ponto a Ponto;
- Baseado em ferramentas;
- Multilinguagem;
- *Software* livre e de código aberto;

O ROS é uma coleção de ferramentas, bibliotecas e convenções que visam simplificar a criação de comportamento complexo e robusto de robôs, através de uma ampla variedade de plataformas robóticas. Ele será utilizado em conjunto com outras ferramentas, como veremos adiante. A seguir, são mostrados alguns simuladores de SMRs e é falado o porquê da escolha pelo Stage (GERKEY, VAUGHAN e HOWARD, 2003).

Um importante elemento do ROS é o Núcleo do ROS (ROS Core). Ele é uma coleção de nós e programas que são pré-requisitos para o funcionamento do ROS como um todo. É necessário que o Núcleo do ROS esteja em funcionamento para que seja possível a comunicação entre os componentes presentes nas simulações. Ele é iniciado através do comando roscore.

O Webots foi proposto por MICHEL em 1998 e se trata de um simulador realista de robôs que permite a comunicação com robôs reais. Robôs reais e simulados podem ser programados usando a linguagem C, através da API Khepera, tornando o código fonte do controlador de um robô compatível entre o simulador e o robô.

O Stage (GERKEY, VAUGHAN e HOWARD, 2003) prove uma população de robôs e sensores simulados operando em um ambiente de duas dimensões. Esses dispositivos são acessados através de um player, como se fosse *hardware* real. Ele oferece uma combinação de transparência, flexibilidade e velocidade, o que, segundo os autores, faz com que o Stage seja o ambiente mais útil existente até então. A figura 2.1 mostra um exemplo de simulação no ROS.

Figura 2.1 - Exemplo de Simulação no Stage

Fonte: http://answers.ros.org

Em 2007, CARPIN et al. apresentou o USARSim, um simulador de SMRs que pode ser usado tanto para pesquisa quanto para educação. Segundo os autores, entre as demais características, ele possui uma *engine* usada para simular competições como a iniciativa RoboCup.

PINCIROLI et al. (2012) apresentou um simulador para Sistemas Multi-Robô chamado ARGoS, projetado para simular experimentos complexos envolvendo grandes enxames de robôs de diferentes tipos. Segundo os autores, ele é, ao mesmo tempo, eficiente e flexível, permitindo alto nível de customização.

Entretanto, um problema comum aos simuladores mostrados é não prover suporte para nenhum modelo oficial de comunicação wireless. No caso do ROS, existe um *patch* que provê um modelo WI-FI simples para versões antigas, como a 3.2.2, porém essas versões não são mais suportadas.

Devido a seu suporte a quatro linguagens de programação: C++, Python, Octave e LISP (QUIGLEY et al., 2009), o Stage foi escolhido para ser utilizado neste trabalho. Além disso, no contexto da Universidade Federal da Paraíba, o Stage vem sendo utilizado em diversos estudos.

2.2. SIMULADORES DE REDE E O OMNET++

A seguir serão mostrados alguns simuladores de rede existentes e será apresentada a justificativa pela escolha do OMNeT++.

O NS-2 (CHEN et al., 2007) é um simulador de rede com foco no protocolo IEEE 802.11 e é utilizado por pesquisadores para simular comunicações wireless. Ele foi originalmente desenvolvido para pesquisas em redes com apoio substancial para a simulação dos protocolos TCP, roteamento e multicast.

O NS-3 (HENDERSON et al., 2008) é uma versão mais atualizada do NS-2 contendo diversas mudanças. Entre elas, um novo núcleo, escrito em C++ e Python, objetos mais realistas, e um suporte a virtualização.

O OMNeT++ é um simulador de eventos discretos escrito em C++ inicialmente desenvolvido para simular redes de computadores e outros sistemas distribuídos, sendo utilizado para modelar redes de comunicação, multiprocessadores e outros sistemas paralelos ou distribuídos (VARGA et al., 2001).

O OMNeT++ está disponível desde 1997. Segundo VARGA et al. (2008), apesar de ter objetivos específicos, como já citado, ele foi projetado para ser o mais genérico possível. Desde então, o OMNeT++ tem sido usado por vários domínios, desde simulações wireless e redes ad hoc, simulações de processos de negócio para redes ponto-a-ponto, interruptor ótico e área de armazenamento em simulações de rede. Ao longo dos anos, várias pesquisas foram feitas utilizando o OMNeT++ base.

As pesquisas envolvendo o OMNeT++ abrangem uma grande quantidade de temas, como: *Wireless Sensor Networks* (WSN), *Wireless Mesh Networks* (WMN), criação de frameworks, implementação de diferentes protocolos, simulação veicular, entre outros. A seguir, iremos falar sobre algumas dessas pesquisas.

Embora o OMNeT++ forneça um poderoso framework de simulação, não há suporte direto para comunicação wireless. MALLANDA et al. (2005), mostrou resultados obtidos a partir de um simulador de Wireless Sensor Networks baseado no OMNeT++. Foi desenvolvida uma plataforma que permitia que desenvolvedores e pesquisadores investigassem problemas de topologia, fenomenológicos, de rede, robustez e escala em

redes de sensores wireless para os protocolos 802.11 e o *Directed Diffusion* (um conhecido protocolo de redes sensoriais).

Nesse mesmo seguimento, COLESCANTI, CROCIANI e VITALLETI (2007) propuseram um método para avaliar a confiabilidade do OMNeT++ ao simular Wireless Sensor Networks. Um conjunto de métricas foi criado com o objetivo de avaliar a qualidade dos resultados obtidos a partir de simulações de Wireless Sensor Networks feitas no OMNeT++.

STAUB, GANTENBEIN e BRAUN (2009) propuseram uma arquitetura para testes e validação de novos protocolos e arquiteturas na área de *Virtual Mesh Networks* (VMN). Ele provê ferramentas para teste de comunicação de *softwares* reais inseridos em ambientes controlados, capturando o tráfego real através de uma interface real nos nós *mesh*. O tráfego real é capturado e redirecionado para a rede pertencente ao OMNeT++.

Posteriormente, KÖPKE et al. (2008), apresentou, em seu trabalho, o MIXIM, responsável por estender diversos frameworks desenvolvidos para simulações móveis e wireless. O MIXIM disponibiliza modelos detalhados de canais wireless, conectividade wireless, modelos móveis, modelos com obstáculos e outros protocolos de comunicação.

De acordo com MAYER e GAMER (2008), o OMNeT++ não provê uma forma de integrar aplicações de rede reais nos modelos de simulações usados pelo simulador. Em seu trabalho, eles propuseram abordagens para tornar isso possível através do encapsulamento de aplicações reais como bibliotecas que podem ser dinamicamente carregadas pelo OMNeT++ em tempo de execução.

Em 2009, KATSAROS et al. apresentaram uma implementação do protocolo BitTorrent para o ambiente de simulação OMNeT++. Segundo os autores, a escolha pelo OMNeT++ se deu pela simplicidade, alto grau de modularidade e pelas diversas implementações de protocolos de rede existentes.

STEINBACH et al. (2011), apresentou uma extensão do framework INET, pertencente ao OMNeT++, para simular real-time Ethernet com alta precisão temporal. O módulo apresentado implementa o protocolo TTEthernet, uma extensão em tempo real para Ethernet padrão que é proposto para normalização.

Na área denominada *Vehicular Ad hoc Network* (VANET), NAGEL e EICHLER (2008) propuseram uma abordagem de modelagem para o OMNeT++ e o framework INET com foco em mobilidade e modelagem de canal. A abordagem permite a geração e retirada de nós durante a execução da simulação e que reduz drasticamente o número de eventos da simulação.

Em 2009, DREIBHOLZ, RATHGEB e ZHOU propuseram um conjunto de ferramentas para as simulações feitas no OMNeT++ denominado SimProcTC. Esse conjunto de ferramentas foi projetado para realizar as tarefas comuns e recorrentes nas simulações. Essas tarefas foram parametrizadas por execuções, seu processo de distribuição e a visualização dos resultados.

BOKOR et al. (2009) propôs um framework, denominado HIPSim++, possibilitando que o OMNeT++/INET simular um protocolo denominado *Host Identity Protocol* (HIP). Esse protocolo desacopla o endereço IP a partir de aplicações em camada alta de Internet propondo um novo e criptografado *namespace* para hosts. A validação do framework foi feita através da comparação de uma aplicação real (feita para servir como base de comparação) com os resultados obtidos através das arquiteturas HIP simuladas.

No presente trabalho, o OMNeT++ foi escolhido por conter frameworks para a grande maioria dos protocolos existentes, com ou sem fio, de forma que no futuro, testes possam ser feitos utilizando esses diferentes protocolos. Ele é responsável por trafegar as informações e mensagens que são trocadas entre os robôs pertencentes ao ROS. No capítulo sobre as simulações, será mostrado em maiores detalhes o funcionamento das redes em conjunto com o simulador de SMRs. Em resumo, para cada robô existente no ROS, existe um correspondente no ambiente OMNeT, de forma que toda a informação passa pela rede antes de chegar ao seu destino. Isso será possível através da utilização de Cossimulação, tema do tópico a seguir.

2.3. COSSIMULAÇÃO

2.3.1. VISÃO GERAL

Sistemas em chip se tornaram complexos não apenas em termos de densidade de componentes, mas também em heterogeneidade. Esses sistemas podem ser encontrados em vários domínios, como defesa, médico, eletrônico, comunicação e automotivo

(NICOLESCU et al. 2007). Além disso, o desenvolvimento de *software* para robôs é uma disciplina exigente devido a necessidade de desenvolver *software*s complexos que lidam com as restrições do mundo físico (BROENINK e GROOTHUIS, 2010).

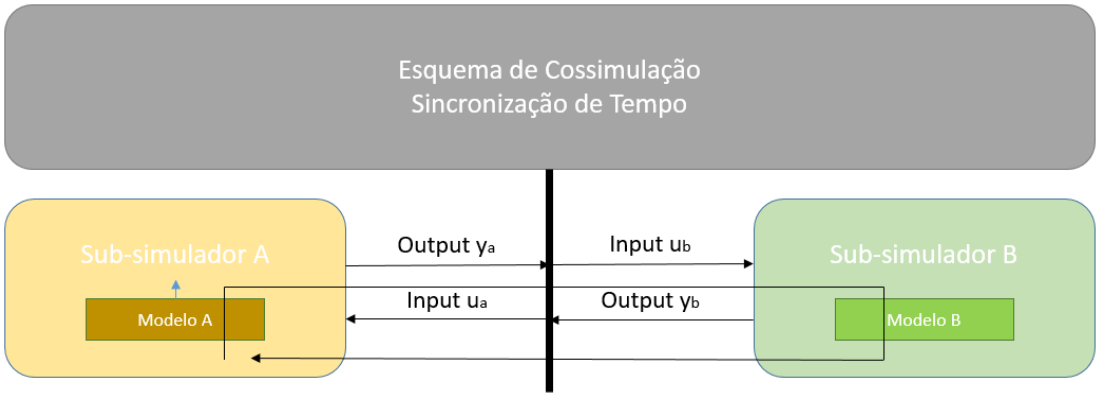
Para NICOLESCU et al. (2007), a cossimulação representa uma das mais populares técnicas de validação para sistemas heterogêneos. Segundo SOUZA et al. (2003), o princípio fundamental da cossimulação é prover suporte a execução cooperativa de diferentes simuladores. Ela permite a junção de simulações de componentes heterogêneos existentes com diferentes modelos de execução. De acordo com BROENINK e GROOTHUIS (2010), durante o processo de projetar sistemas robóticos, a interação entre modelos controladores e modelos dinâmicos de mecanismos robóticos é estudada utilizando cossimulação.

De acordo com ALHARTHI (2014), cossimulação permite integrar modelos heterogêneos de diferentes simuladores, tipicamente de diferentes domínios, como sistemas mecânicos e em rede. Modelos combinados podem interagir com outros durante a simulação através de um mecanismo de interface.

Cossimulação envolve acoplar simulações de Eventos Discretos (*Discrete-Event* – DE) e de Tempo Contínuo (*Continuous-Time* – CT) rodando em ferramentas distintas (FITZGERALD et al., 2013) onde a variável Tempo deve ser sincronizada (BROENINK e GROOTHUIS, 2010) (Figura 2.2). Em simulações de evento discreto, são representados apenas os pontos no tempo onde há mudanças no sistema. Essas mudanças são chamadas de Eventos. Em uma simulação de tempo contínuo, o estado do sistema muda continuamente através do tempo e, de forma a modelar esse comportamento no computador, o simulador aproxima continuamente mudanças por pequenos passos em tempo discreto.

Para BROENINK e GROOTHUIS (2010), o modelo de Tempo Contínuo (CT) é utilizado para mecanismos robóticos dinâmicos e controlados por algoritmos e o modelo de Eventos Discretos (DE) para controle de decisões e lógica implementada em sistemas embarcados.

Figura 2.2 - Esquema de Cossimulação



2.3.2. SINCRONIZAÇÃO DE TEMPO

Na cossimulação, o tempo é fator determinante para sucesso da execução, onde se faz necessário que haja um meio de manter um sincronismo entre todos os componentes participantes das simulações. Para isso, existe um mecanismo de execução global que governa a execução da cossimulação e alterna entre os simuladores. A alternância de execução permite que cada simulador tenha seu turno de execução de forma a desempenhar suas próprias tarefas. Essa alternância pode ser agendada de acordo com passos predefinidos ou pode ser dinamicamente ajustada dependendo dos passos que precisam ser desempenhados ou do estado dos eventos em um modelo contínuo, dependendo de como cada simulador opera (ALHARTHI, 2014).

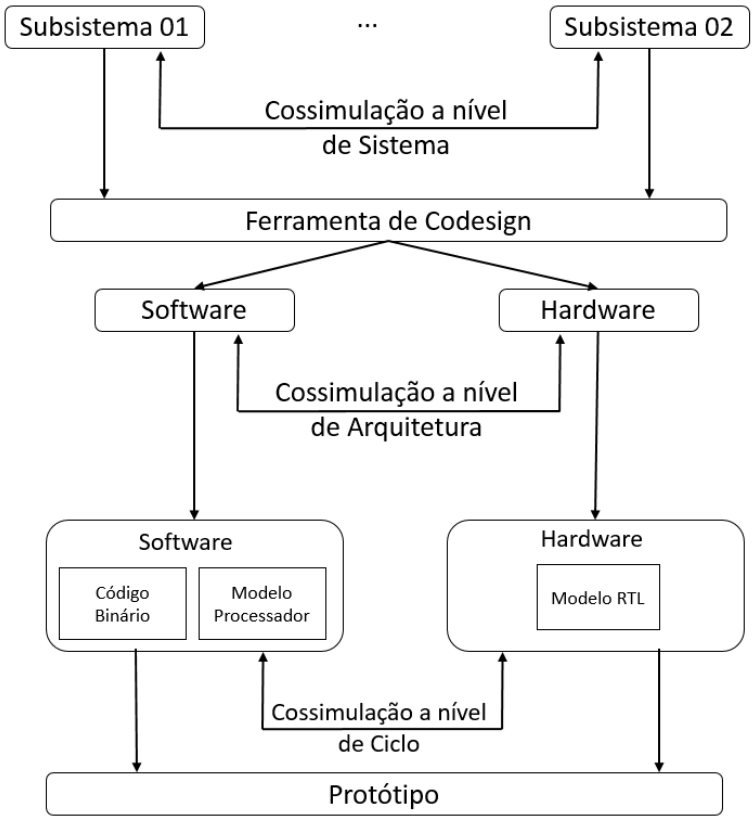
Como vai ser visto mais adiante, no ambiente proposto neste trabalho, quem é responsável por este papel é o próprio CERTI, onde, através de um método chamado advanceTime, garante que todos os participantes tenham seus turnos e que não avancem sem que os demais estejam preparados para isso. Uma vez que há uma requisição para avanço de tempo, ou seja, um dos integrantes termina seu turno, o CERTI recebe uma mensagem avisando sobre o fim do turno e passa a aguardar pelo fim dos demais turnos. O CERTI, por sua vez, também aguarda que todos terminem seus turnos para poder autorizar que o tempo seja avançado e o mesmo ciclo seja iniciado novamente. Dessa forma, há garantia de que todos os envolvidos nas simulações estejam sempre operando no mesmo tempo global de execução.

2.3.3. NÍVEIS DE ABSTRAÇÃO

Para AMORY et al. (2002), existem três níveis de abstração onde a cossimulação pode ser utilizada: Sistema, arquitetura e nível de ciclo. Ao nível de sistema, o objetivo principal é caracterizar a funcionalidade do sistema. Nesse nível, a parte de *software* do sistema é descrita utilizando uma linguagem de programação, como C ou C++. A parte de *hardware* é descrita utilizando uma linguagem de descrição de *hardware*, como VHDL ou Verilog.

Ao nível arquitetural, a comunicação *hardware*/*software* é levada em consideração junto com o dispositivo alvo. Ao nível de simulação de ciclo, os componentes de *software* são simulados usando código binário executando em um simulador de nível de ciclo no processador do host alvo. Esses conceitos são representados na Figura 2.2.

Figura 2.3 - Cossimulação em diferentes níveis de abstração



2.3.4. CLASSIFICAÇÃO

AMORY et al. (2002) ainda classifica cossimulação de acordo com três critérios: número de simuladores, modelo temporal de execução e distribuição.

De acordo com o número de simuladores, uma cossimulação pode ser homogênea ou heterogênea. Um modelo de cossimulação homogêneo traduz a linguagem de descrição dos modelos que compõem o sistema em uma única linguagem, chamada formato intermediário, capaz de descrever todo o sistema. Nessa abordagem apenas um simulador é necessário. Um modelo de cossimulação heterogêneo emprega um simulador dedicado para cada linguagem utilizada. Nessa abordagem um *backplane* é necessário para coordenar a comunicação entre os simuladores.

De acordo com o modelo temporal de execução, uma cossimulação pode ser funcional ou temporal. Um modelo funcional é utilizado para validar o sistema em níveis mais altos de abstração, onde detalhes temporais e de comunicação são negligenciados. Um modelo temporal permite uma validação mais precisa, já que informações de tempo, arquitetura do processador e modelo de barramento podem ser levados em consideração quando os simuladores interagem.

De acordo com a distribuição, uma cossimulação pode ser distribuída ou local. Uma cossimulação distribuída permite execução paralela de simuladores em máquinas geograficamente distribuídas através de redes do tipo *Local Area Network* (LAN) ou *Wide Area Network* (WAN). Os benefícios dessa abordagem são: (i) descentralização de projeto; (ii) projeto e validação do sistema em desenvolvimento em times geograficamente distribuídos; (iii) gerenciamento de propriedade intelectual – simulações podem ser feitas sem o repasse de códigos-fonte; (iv) simuladores podem ser instalados em algumas máquinas apenas; (v) compartilhamento de recursos. Uma cossimulação local não possui overhead de comunicação, já que não é utilizada uma rede para troca de informações. Porém, não possui as vantagens presentes nas co-simulações distribuídas.

Vários trabalhos envolvendo Cossimulação foram propostos ao longo dos anos, como a de NICOLESCU et al. (2006). Nela, é apresentada o CODIS, um framework de cossimulação para sistemas contínuos e discretos. Baseado em um modelo bem definido de sincronização e uma arquitetura genérica para modelos de simulação discretos e contínuos. Os simuladores suportados são o Simulink para componentes contínuos e o SystemC para componentes discretos. Os experimentos mostraram um overhead de sincronização de menos de 30% do tempo de simulação.

AMORY (2002) apresenta a implementação e avaliação de uma ferramenta de cossimulação de *hardware* e *software*, composto por diferentes simuladores, que podem estar distribuídos geograficamente. Nela, a comunicação entre os simuladores é feita utilizando um *backplane* de cossimulação. Ele lê um arquivo que descreve como os módulos estão conectados, automaticamente carregando os simuladores e controlando o processo de simulação.

Em seu trabalho, SUNG e KIM (2011) mostraram a interoperabilidade entre dois Modelos de Computação (Eventos Discretos e Tempo Contínuo) para simular um sistema que controla o nível de água em um tanque através da utilização de HLA como plataforma de simulação distribuída e o MATLAB para simular o sistema de controle.

Uma abordagem para o desenvolvimento de um modelo combinado onde um modelo de evento discreto pode incluir aproximações de comportamentos de tempo contínuo que pode subsequentemente ser substituído por modelos de tempo contínuo acoplados é proposta por FITZGERALD et al. (2013). Uma semântica operacional de cossimulação permite os modelos de eventos discretos e tempo contínuo executarem em seus respectivos simuladores e serem gerenciados por um mecanismo de cossimulação.

Uma arquitetura de *backplane* cliente/servidor chamada SimConnect/SimTalk que implementa a dinâmica do *Kahn Process Network* (KPN) foi proposta por PFEIFER e VALVANO (2011). Com sua coordenação, simuladores são sincronizados através de um fluxo de dados, ao invés do passo de controle explicito de tempo, diminuindo a complexidade da API do *backplane*. Adicionalmente, simuladores possuem apenas o conhecimento das suas filas de entrada e saída, o que oferece um gerenciamento mais fácil, pois aumenta o número de simuladores.

2.3.5. HIGH LEVEL ARCHITECTURE

High Level Architecture (HLA) foi desenvolvida com o objetivo de prover uma maneira mais simples de prover interconexão entre simuladores. Ela provê uma especificação de uma arquitetura comum para uso das diversas ferramentas de simulação no Departamento de Defesa dos Estados Unidos (DAHMANN et al., 1997), é um padrão descrito no IEEE 1516-series e vem sendo desenvolvida para prover uma arquitetura comum para modelagem e simulação distribuídos (IEEE, 2010).

A base do HLA é a premissa de que um único simulador não pode satisfazer completamente todos os usos e usuários (DAHMANN et al., 1997) e seu principal objetivo é tornar possível a interoperabilidade entre modelos distintos e usá-los quando necessário para prover um ambiente distribuído de simulação para sistemas que precisam de processamento em larga escala (BRITO et al., 2013).

Ela provê a estrutura base para interoperabilidade de simulações e sua definição básica inclui (1) regras, (2) interface de especificação e o *template* de modelo de objeto (DAHMANN et al., 1997).

A ideia principal do HLA é separar funcionalidades específicas de cada simulador utilizando uma infraestrutura de propósito geral (BRITO et al., 2013). Como forma de se comunicar com a HLA e outros simuladores, cada simulador precisa usar a *Runtime Infrastructure* (RTI).

A RTI é responsável por especificar a estrutura de cada simulador em interface com a estrutura global da HLA. Uma vez conectado a um RTI, um simulador é chamado de Federação (*Federation*). Após a criação da Federação, os chamados Federados (unidades menores, como os robôs) se juntam à federação e, a partir daí, estão aptos a trocarem informações através da RTI e do HLA. Na literatura, existem trabalhos que utilizam de HLA para prover testes e verificação de sistemas embarcados.

No presente trabalho, a *High Level Architecture* será parte fundamental, provendo uma maneira de utilizarmos o simulador de Sistemas Multi-Robô, o ROS, em conjunto com o simulador de Rede, o OMNeT++. Como falado anteriormente, ela irá prover a base para a interoperabilidade entre os diferentes simuladores, tornando possível a execução simultânea e coordenada durante as simulações, tornando possível explorar o potencial de ambos os simuladores.

3 TRABALHOS RELACIONADOS

Neste tópico, nós discutimos alguns tópicos relevantes para o trabalho apresentado nesta dissertação. Em especial, os trabalhos que tratam de simulações de Sistemas Multi-Robô, Cossimulação e simulações de redes.

Através da utilização de *High Level Architecture* (HLA), STRAßBURGER et al. (1998) propôs um ambiente de simulação distribuída de tráfego, onde um conjunto de federações interoperáveis cooperam e se comunicam através da RTI da HLA. Apesar de utilizar o HLA como integrador, não há preocupação quanto a comunicação no ambiente proposto. O principal diferencial entre este trabalho e o apresentado na dissertação é a preocupação no que diz respeito a comunicação fazer parte das simulações em SMRs. Esse diferencial poderá ser visto com mais clareza na Tabela 3.1, onde há um comparativo entre o trabalho proposto e os trabalhos relacionados.

Em seu trabalho, BRITO et al. (2013) propôs o desenvolvimento e avaliação de uma solução para modelagem e simulação de Modelos de Computação (MoCs) heterogêneos de forma distribuída através da integração de Ptolemy II e *High Level Architecture* (HLA), um middleware para simulação de eventos discretos, de forma a criar um ambiente para execução em alto desempenho de modelos heterogêneos de larga escala. Este trabalho tem foco em cossimulação e SMRs, mas não possui os aspectos relacionados com a comunicação, como no trabalho de STRAßBURGER et al. (1998).

Um framework que utiliza de cossimulação para avaliação do sistema e considera a interação de componentes de *cyber-physical systems* (CPS) para projetar *time-trigged* (TT) CPS foi desenvolvido por ZHANG et al. (2013). A validação do framework é feita em comparação com os resultados de um simulador automotivo do tipo *hardware-in-the-loop*. Assim como os trabalhos anteriores, apesar de utilizar os conceitos de SMRs e Cossimulação, não existe preocupação com a comunicação, parte fundamental do trabalho apresentado nesta dissertação.

Um ambiente chamado AcumenNS3 que permite modelos físicos contínuos ou discretos serem incorporados a uma rede foi proposto por ALHARTHI (2014). Esse ambiente permite que fenômenos físicos possam ser modelados em um ambiente de modelagem física sem comprometer a precisão do fenômeno físico no simulador de rede. O simulador de rede utilizado foi o Network Simulator 3 (NS-3) e a linguagem de

modelagem física utilizada foi a Acumen. A integração dessas plataformas foi feita através da utilização de cossimulação.

Apesar de utilizar os conceitos de cossimulação, simulações de rede e cossimulação, este se trata de um trabalho mais geral, com foco em qualquer tipo de modelo físico contínuos ou discretos. O trabalho presente nesta dissertação tem foco mais específico nos SMRs.

Um ambiente que visa a integração virtual de componentes de modelos de uma cossimulação distribuída foi proposto por SOUZA et al. (2003). A cossimulação é baseada em uma versão modificada da HLA, chamada *Distributed Co-Simulation Backbone* (DCB). Ela impõe padrões proprietários para troca de dados e requer chamadas explícitas a funções da *Runtime Infrastructure* (RTI).

SOUZA et al (2003), assim como ALHARTHI (2014), mostra um trabalho com foco mais geral e não utiliza de um simulador de rede para comunicação, mas de padrões proprietários. Nesta dissertação, uma das razões pela escolha do OMNeT++ foi o fato de ser código-aberto.

Em seu trabalho, POHJOLA, NETHI e JÄNTTI (2008) estenderam o PiccSIM (NETHI et al., 2007), uma ferramenta de simulação e redes conjuntas combinando o MATLAB e Simulink com o NS-2 (um simulador de rede), para dar suporte à mobilidade de forma a criar simulações mais realistas de grupos de robôs. Sua pesquisa tem foco em aplicar o PiccSIM para comparar, em um mesmo cenário, rotas de protocolos de caminho único com rotas de múltiplos caminhos.

O trabalho proposto por POHJOLA, NETHI e JÄNTTI (2008) combina os três principais fatores encontrados na arquitetura proposta na dissertação: SMRs, cossimulação e simulação de redes. Entretanto, seu intuito se resumiu a testar diferentes protocolos de caminho. Nesta dissertação o principal foco é observar como a comunicação irá afetar o comportamento dos robôs.

Um framework para integração de simuladores foi proposto por ROTH et al. (2014). Uma aplicação para comunicação carro-a-carro foi desenvolvida, onde o SystemC foi utilizado como modelo de controlador eletrônico dos carros, o OMNeT++ é responsável pela comunicação e o Sumo, por simular o tráfego de carros.

Em seu trabalho, ROTH et al. (2014) mostra preocupação quanto a comunicação quando utiliza o OMNeT++ como meio para testá-la, apesar de ter sido feito para um contexto bem específico (comunicação carro-a-carro).

Neste trabalho, queremos introduzir o conceito de robôs em rede à simulação, onde seja possível simularmos diferentes protocolos de comunicação, latências, níveis de interferência e outros aspectos que envolvem a comunicação em rede. Isso fará com que as simulações de Sistemas Multi-Robô possam se aproximar ainda mais do que aconteceria se estivéssemos no mundo real.

A seguir, é mostrada uma tabela comparativa entre os trabalhos apresentados em relação aos três principais aspectos tratados nesta dissertação: SMRs, cossimulação e simulação de redes.

Tabela 3.1- Comparativo entre os Trabalhos Relacionados e o Proposto

	Sistemas Multi-Robô	Cossimulação	Simulação de Redes	Abrangência
STRAßBURGER et al. (1998)	X	X		Simulação de Tráfego
SOUZA et al. (2003)	X	X		MoCs
POHJOLA, NETHI e JÄNTTI (2008)	X	X	X	Rotas de Protocolos de Caminho Único X Múltiplos Caminhos
BRITO et al. (2013)	X	X		MoCs
ZHANG et al. (2013)	X	X		SMRs

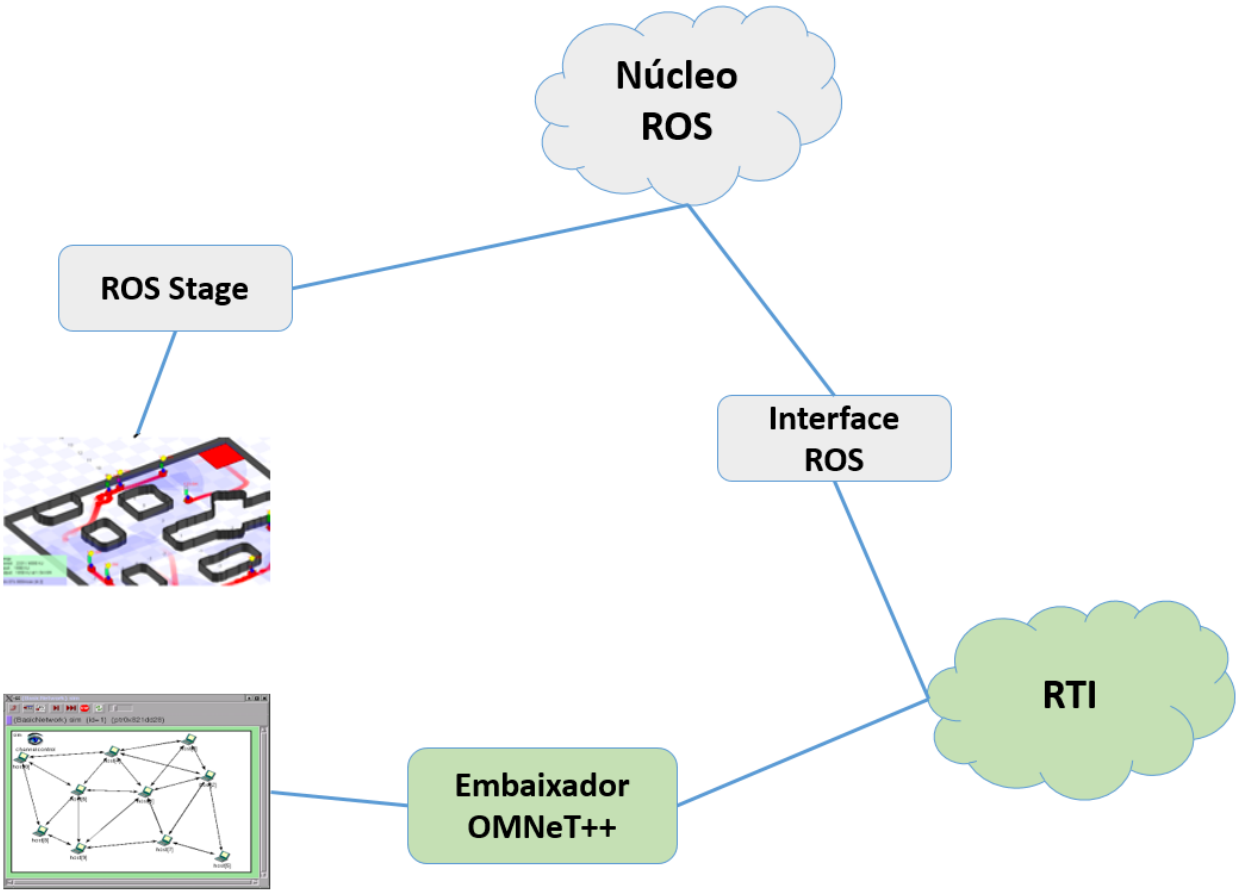
ALHARTHI (2014)	X	X	X	MoCs
ROTH et al. (2014)	X	X	X	Comunicação carro-a-carro
Trabalho Proposto	X	X	X	SMRs

O capítulo seguinte irá tratar da arquitetura proposta neste trabalho. A arquitetura proposta pode ser dividida em duas partes. A primeira compõe todo o ambiente do simulador de SMRs, o ROS. A segunda parte é responsável pela parte de Cossimulação e Rede, utilizando o CERTI como ferramenta de HLA e o OMNeT++, como simulador de Redes.

4 ARQUITETURA PROPOSTA

A seguir, será mostrado o ambiente para o projeto de simulação e testes de Sistemas Multi-Robô através da utilização de Cossimulação, objetivo deste trabalho. A arquitetura pode ser vista na Figura 4.1.

Figura 4.1 - Arquitetura Proposta



O ambiente proposto pode ser dividido em duas partes. A primeira compõe todo o ambiente ROS com seus Robôs, sua interface (Interface ROS) e seu núcleo. A segunda parte é o ambiente HLA, contendo o *Runtime Infrastructure* (RTI), o Embaixador OMNeT++, e a simulação no OMNeT++.

A primeira parte da arquitetura é composta pelo ROS Stage, o Núcleo ROS e a interface. O ROS Stage é responsável por simular o ambiente onde será feita a simulação bem como os robôs e suas características. A figura 4.2 mostra o arquivo de configuração utilizado nas simulações feitas neste trabalho.

Figura 4.2 - Trecho do arquivo de Configuração do ROS Stage

```
window
(
  size [ 600.0 700.0 ]
  center [ 0.0 0.0 ]
  rotate [ 0.0 0.0 ]
  scale 60
)

turtlebot
(
  pose [ 0.0 0.0 0.0 0.0 ]
  name "turtlebot"
  color "black"
)

turtlebot
(
  pose [ 2.0 2.0 0.0 0.0 ]
  name "turtlebot"
  color "blue"
)
```

No arquivo de configuração, todo o ambiente é montado. O primeiro bloco (*window*) é responsável pelo ambiente onde os robôs estarão sendo simulados. Ela possui tamanho (*size*), posição inicial (*center*), rotação (*rotate*) e escala (*scale*).

Após a especificação do ambiente, robôs são inseridos nos dois blocos seguintes. Nesse caso, os dois robôs são do tipo turtlebot e começam em uma posição (eixos x, y, z, w), possuem nome (*name*) e cor (*color*).

O Núcleo ROS é responsável por coordenar o funcionamento interno ao ROS e faz parte da estrutura do ROS Stage. A interface ROS é responsável por fazer com que o ROS Stage possa se comunicar com HLA. Basicamente ela checa as variáveis ROS de todos os Robôs (como, velocidade, posição, etc.) e envia essas informações para o ambiente HLA. Então, é possível haver vários robôs compartilhando seus dados com o OMNeT++, no nosso caso, ou qualquer outro simulador capaz de se comunicar com o HLA.

Antes de abordarmos especificamente a segunda parte da arquitetura, é importante falar sobre como são representados os robôs para o HLA, definindo e padronizando como

será feita a troca de informações entre os simuladores que farão parte da arquitetura proposta neste trabalho. O HLA utiliza do paradigma orientado a objeto para descrever os dados que irão ser trocados entre os federados em um arquivo chamado *Federate Object Model* (FOM). Através desse arquivo podemos descrever classes, objetos, atributos e hierarquias de classes. Uma vez mapeado, todas as interfaces envolvidas precisam implementar sua comunicação baseada nesse arquivo. Os atributos da classe utilizada em nossas simulações são descritos abaixo:

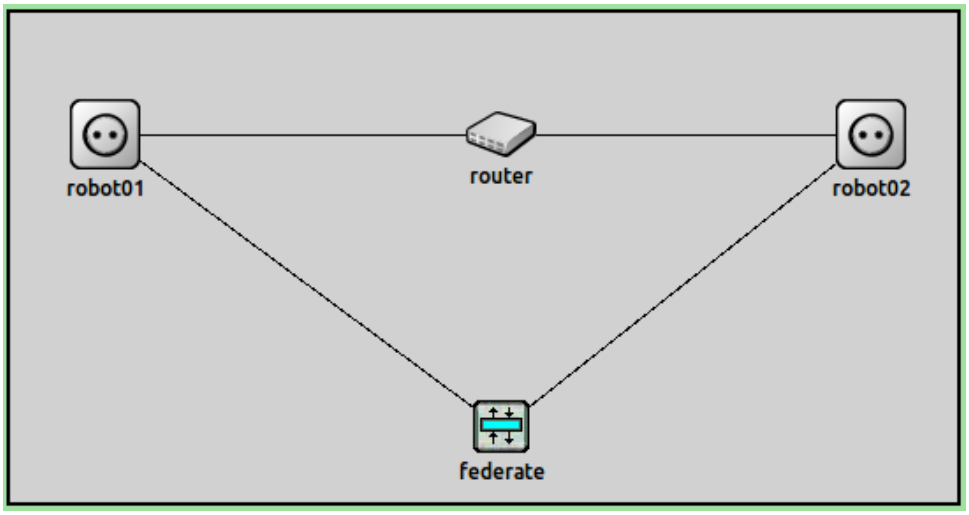
- ID: Identifica um Robô, uma vez que um ou mais robôs podem ser simulados ao mesmo tempo;
- Bateria (*Battery*): Responsável por mostrar o nível de bateria de um robô;
- Temperatura (*Temperature*): Responsável por mostrar a temperatura de um robô. Importante para simular caso onde os robôs sejam sensíveis à temperatura, como em componentes elétricos;
- Sensores (Sensor1, Sensor2, Sensor3): Permite o monitoramento de sensores presentes em um robô;
- GPS: Controla a posição atual do robô;
- Bússola (*Compass*): A bússola do robô;
- GoTo: Controla a posição para qual o robô precisa se mover;
- Rotação (*Rotate*): É um comando específico utilizado para controlar a rotação do robô (em graus);
- Ativação (*Activate*): Ativa ou desativa os sensores ou atuadores de um robô;

A segunda parte da arquitetura é composta pelo *Runtime Infrastructure* (RTI), o Embaixador OMNeT++, e a simulação no OMNeT++. O RTI é responsável pela troca de mensagens entre os simuladores, captando e repassando as mensagens enviadas sempre que necessário. Tanto a interface ROS quanto o Embaixador OMNeT++ se comunicam com o RTI.

O embaixador OMNeT++, chamado de *Federate* (Figura 4.3), é um componente que, para um melhor entendimento, foi representado como um componente separado (Figura 4.1), fisicamente faz parte da implementação da simulação OMNeT++. Sua função é permitir a troca de informações entre o simulador de rede e o HLA sem interferir na comunicação entre os robôs. Em outras palavras, o *Federate* não interfere na comunicação

interna do OMNeT++, ou seja, não há latência de comunicação, perda de pacote ou qualquer outro aspecto relacionado à rede. A Simulação no OMNeT++ pode ser vista na Figura 4.3:

Figura 4.3 - Simulação no OMNeT++



A rede utilizada nos experimentos possui dois robôs, um roteador e um *federate*. Os robôs robot01 e robot02 representam os robôs no ROS e toda a informação destinada a eles deve passar através dos robôs correspondentes no OMNeT++. A comunicação entre os robôs e o roteador possui latência (em milissegundos) e uma taxa de perda de pacotes (em percentual) propositalmente inseridas para aumentar o realismo das comunicações. No OMNeT++, toda simulação é representada por um arquivo de extensão NED (*Network Description*). O arquivo NED utilizado nos experimentos pode ser visto na Figura 4.4.

Figura 4.4 - Arquivo NED utilizado nos experimentos

```
network Mestrado
{
    @display("bgb=538,278");

        types:
                channel Channel extends ned.DelayChannel {
                delay = 59ms;
        }
        submodules:
        router: Router {
                @display("p=279,69");
        }
        robot01: Robot {
                        @display("p=54,69");
                        master = true;
        }
        robot02: Robot {
                        @display("p=489,69");
                        master = false;
        }
        federate: Federate {
                        @display("p=279,235");
        }

    connections:

        robot01.gate[0] <--> {  delay = 0ms; } <--> federate.gate[0];
        robot02.gate[0] <--> {  delay = 0ms; } <--> federate.gate[1];

        router.gate[0] <--> Channel <--> robot01.gate[1];
        router.gate[1] <--> Channel <--> robot02.gate[1];
}
```

O arquivo mostrado na Figura 4.4 é explicado em detalhes a seguir. Nele é definida uma rede, chamada Mestrado, que está posicionada na posição 538, 278, especificada no parâmetro @display. A especificação da rede é dividida em três partes principais: Tipos (*types*), Submódulos (*submodules*) e Conexões (*connections*).

Tipos são como estruturas que poderão ser utilizadas em quaisquer submódulos da rede. No caso mostrado, existe um canal (*channel*) chamado Channel, que possui um *delay* padrão de 59ms. Atributos de um tipo podem ser sobrescritos, como aconteceu nas primeiras duas conexões.

Submódulos representam os componentes existentes na rede (como roteadores, computadores, antenas, etc.). No arquivo mostrado, existem dois robôs (robot01 e robot02)

e um componente chamado Federate, já explicado anteriormente. Os robôs possuem um atributo chamado *master*, que especifica se o robô é do tipo mestre (*master*) ou escravo (*slave*). Além disso, possuem duas portas (*gates*) de saída, uma para comunicação entre os robôs e a segunda para permitir a comunicação entre os robôs e o Federate.

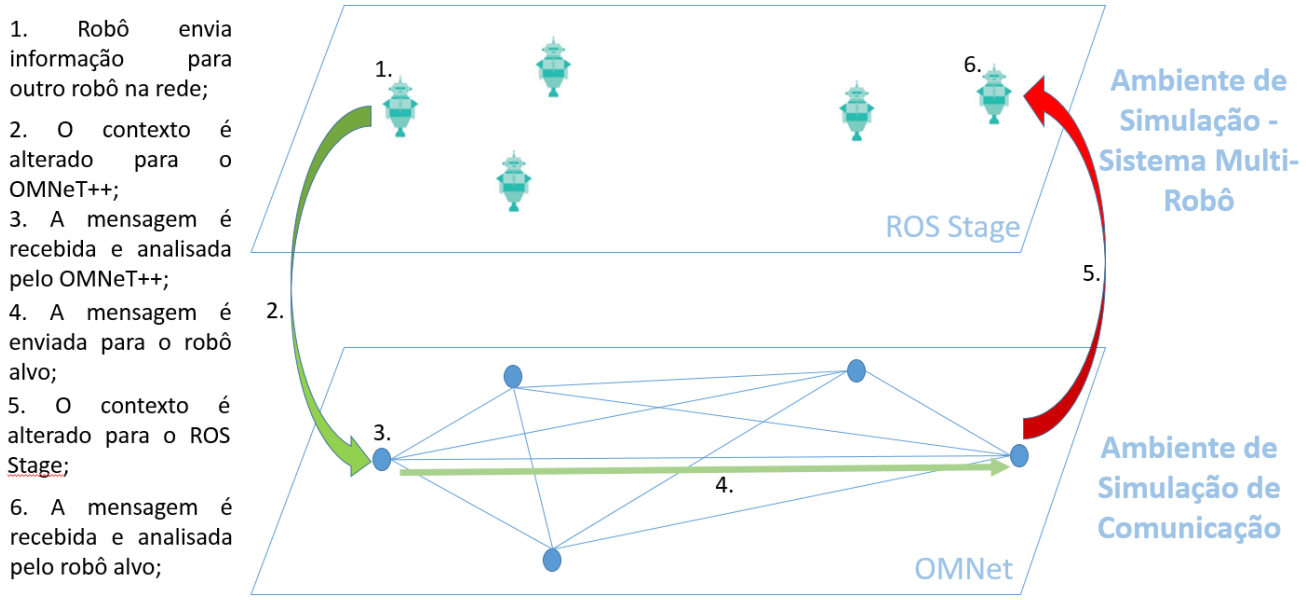
Conexões é o bloco responsável por definir como os componentes da rede irão se comunicar. No caso da figura 4.4, os robôs robot01 e robot02 comunicam-se de acordo com o submódulo Channel e com o componente Federate a um delay de 0ms. O delay na comunicação entre os robôs e o Federate foi propositalmente definido como zero para que o Federate não tenha qualquer impacto na comunicação interna ao OMNeT++, já que seu papel é fazer a comunicação do OMNeT++ para o HLA.

A arquitetura mostrada na figura 4.3 pode ser vista por uma outra visão, a qual chamamos de Ambiente de Cossimulação. Esse ambiente é mostrado na Figura 4.5 e pode ser dividido em duas partes. A primeira, chamada ambiente de Sistema Multi-Robô *(Multi-Robot System Environment)* e a segunda, Comunicação (*Communication)*.

O ambiente de Sistema Multi-Robô, fica responsável por simular os diferentes robôs atuando no mesmo ambiente e utiliza o ROS, como dito anteriormente. Nele são instanciados os *Robot Nodes*, que irão transmitir dados para a segunda camada do ambiente de simulação, a camada de Simulação de Comunicação.

A camada de Simulação de Comunicação utiliza o OMNeT++, um simulador feito em C++, construído inicialmente para simulações de rede. Esse simulador é responsável pela comunicação entre os robôs que fazem parte do ROS, de forma que seja possível a troca de informações. Com isso, será possível simular robôs interagindo em rede e testar as mais diversas situações envolvidas nesse tipo de comunicação (protocolos de rede, perdas de pacotes, interferências, entre outros), tornando as simulações mais próximas da realidade.

Figura 4.5 - Ambiente de Cossimulação



Um robô pertencente ao ROS envia uma informação para outro robô dentro da rede. Essa informação, necessariamente, deve passar pelo OMNeT++, que será responsável por enviá-la ao robô destino através de sua rede. Dessa forma, a mensagem é recebida pelo robô correspondente (no OMNeT++) que a repassa através da rede para o robô destino. Assim que a informação é recebida, é repassada para o robô destino correspondente no ROS, finalizando o ciclo.

Para o desenvolvimento e teste do ambiente proposto nesta dissertação, utilizou-se das seguintes ferramentas: Máquina Virtual (VirtualBox 5.0.10) Ubuntu 14.10 (64 bits) com 5259 MB de RAM, Processador AMD FX™ 6400 *Six-Core processor* (3 cores) e 40GB de disco rígido; Simulador de Rede OMNeT++ (em C++); Ambiente de Simulação distribuída CERTI (HLA) (em C++); Ambiente de Simulação para os Robôs (ROS – *Robot Operation System*); Python para desenvolvimento dos controladores dos robôs simulados pelo ROS.

4.1. SINCRONIZAÇÃO DE TEMPO

Um fator crucial para a correta execução da arquitetura proposta é a sincronização de tempo. Sem ela, cada simulador executaria de forma paralela e as informações trafegariam em tempos aleatórios e descoordenados, o que invalidaria qualquer resultado obtido. Para os simuladores e para o CERTI, as simulações possuem 1 segundo de ciclo e para evitar que o tempo esteja descoordenado, o CERTI dispõe de um método chamado advanceTime para assegurar que todos os simuladores estejam no mesmo tempo de execução.

Uma vez que algum dos federados executa o primeiro advanceTime, ele e o CERTI ficam aguardando até que os demais federados também emitam o mesmo sinal. Ou seja, em uma simulação onde há N simuladores, assim que o primeiro simulador executa o método advanceTime, ele irá ficar aguardando até que os outros N-1 simuladores também façam o mesmo, garantindo que todos estejam no mesmo tempo de execução.

Assim que o método é executado por todos os federados, o CERTI pode enfim avançar o tempo da simulação como um todo, liberando para que todos os simuladores sigam sua execução até o próximo advanceTime. Esse ciclo persiste enquanto todos os elementos da arquitetura estejam em execução.

No ambiente do ROS, enquanto a arquitetura está em funcionamento, existe um processo que controla seu avanço de tempo, executando o advanceTime a cada segundo. Uma vez que isso acontece, os robôs no ROS param sua movimentação e o ROS fica esperando pela “autorização” do CERTI para poder prosseguir sua execução.

Após testes envolvendo toda a arquitetura, foi verificado que o tempo de um segundo era ideal para este cenário, uma vez que com tempo menor, os ciclos eram muito curtos, fazendo com que a simulações terminassem em um tempo muito curto e não fosse possível obter resultados válidos; e com o tempo maior, o HLA e o Stage ficavam muito tempo esperando pelo OMNeT++, fazendo com que os robôs passassem muito tempo parados.

No OMNeT++ a sincronização de tempo é feita por um nó chamado Federate, como mostrado anteriormente. Suas únicas funções são (1) fazer com que o simulador seja

capaz de se conectar ao CERTI e (2) executar o método advanceTime a cada segundo, como no ROS.

Dessa forma, os simuladores avançam seu tempo em um segundo e, após isso, a simulação como um todo pode continuar.

4.2. VALIDAÇÃO DA ARQUITETURA

A ideia é simular cenários no contexto do RoboCup utilizando ROS, cossimulação e simulação de redes. O *Robot World Cup Initiative* (RoboCup) é uma tentativa de melhorar a pesquisa em Inteligência Artificial e Inteligência Robótica provendo um problema padrão onde uma grande quantidade de tecnologias pode ser integrada e examinada (KITANO et al., 1997).

Para avaliar o impacto causado pela adição do OMNeT++, dois tipos de simulações foram criados. As simulações possuem N robôs, onde um deles é chamado de mestre, responsável por coordenar os outros robôs, e N-1 escravos. O robô mestre se move a partir de um ponto X_0Y_0 para X_1Y_1 e, a cada ciclo de simulação, envia sua posição atual para todos os escravos, que seguem o mestre a partir de suas posições até próximo da posição do robô mestre. Em outras palavras, a cada ciclo, o robô mestre envia sua posição atual para os robôs escravos. Uma vez que os robôs escravos recebem essa informação, começam a mover-se de suas posições atuais para a posição repassada pelo robô mestre, seguindo-o.

Como forma de validação, as primeiras simulações foram executadas com dois robôs (mestre e escravo) e com tempos de 50, 100, 200 e 400 segundos, totalizando 20 simulações (5 para cada tempo de simulação).

Como dito anteriormente, como forma de atestar o impacto causado pela adição do OMNeT++, foram feitos dois tipos de simulação. O primeiro tipo de simulação executa em um “ambiente ideal”, uma vez que não existe nenhuma interferência na comunicação entre os robôs pertencentes ao ROS. A simulação servirá de base para comparar o que será feito utilizando o segundo tipo de simulação.

No segundo tipo de simulação, o OMNeT++ é responsável pela troca de mensagens entre os robôs presentes no *Stage* ROS. A utilização do OMNeT++ permite adicionar latências de comunicação, perda de pacotes e outras coisas à rede de forma que permita

que a simulação esteja mais próxima da realidade. Dessa forma, nós comparamos a execução dos dois tipos de simulações e observamos os impactos que a rede simulada pelo OMNeT causa no comportamento dos robôs simulados no *Stage* ROS.

Neste tipo de simulação, além dos dois robôs, existe um roteador que se comunica com os robôs com uma latência de 59 milissegundos e pode ter uma chance de perda de pacotes entre 1 e 5 por cento. Uma vez que um robô envia uma mensagem para outro, um contador é iniciado e 300 milissegundos depois, caso uma mensagem de confirmação de recebimento não seja recebida, a mesma mensagem é reenviada. As simulações seguem o fluxo de comunicação como mostrado anteriormente na Figura 4.5.

4.2.1. Resultados da Validação

Até este ponto, dois aspectos haviam sido observados quando comparados os dois tipos de simulação: Tempo de execução e a quantidade de dados gerada pelas simulações. A Tabela 4.1 mostra de forma geral os dados coletados nas simulações. A partir dela, pode-se concluir que três aspectos tiveram um aumento considerável após a inclusão do OMNeT++. Esses aspectos serão discutidos separadamente.

Tabela 4.1 - Visão Geral dos Dados.

Tempo Simulado	Tempo de Execução (Milissegundos)			Bytes Enviados			Bytes Recebidos		
50	50590	58269	7679	10713	32111	21398	25614	87179	61565
100	100379	116160	15781	20436	62671	42235	48943	169036	120093
200	200199	231969	31770	40117	123158	83041	95796	331845	236049
400	399876	463993	64117	79755	245153	174398	189816	659524	469708

Colunas laranjas representam os dados para as simulações que utilizaram apenas o ROS. Colunas verdes representam as simulações que tiveram o acréscimo do OMNeT++. As colunas azuis mostram as diferenças entre os valores apresentados nas colunas laranjas e verdes.

Em relação ao Tempo de Execução com o OMNeT++, o tempo é calculado levando em consideração o tempo de simulação, a latência existente entre as comunicações e o tempo gerado pela perda de pacotes. A partir dos dados levantados na Tabela 4.1,

percebe-se que a diferença nos tempos de execução pouco mais que dobra à medida que o tempo de simulação também dobra, variando de 7679, para simulações de 50 segundos, até 64117 milissegundos, para simulações de 400 segundos.

Em relação aos Bytes enviados, o aumento foi de 21398 Bytes para simulações de 50 segundos, chegando até 174398 Bytes nas simulações mais longas, o que representa 218% da quantidade de Bytes enviados quando a simulação não possui o OMNeT++ como parte integrante. Similarmente, no que diz respeito aos Bytes Recebidos, para simulações de 50 segundos o aumento foi de 340%, chegando aos 347% para as simulações de 400 segundos.

No que diz respeito às mensagens trocadas durante a fase de validação, mensagens eram disparadas aleatoriamente para o robô mestre ou para o escravo, sem nenhum critério específico para isso. Uma vez que uma mensagem é recebida por um robô, duas novas mensagens são geradas de forma que sua posição é enviada para seu robô correspondente no HLA e a outra segue para o outro robô no OMNeT++. Dessa forma, o número de mensagens enviadas será sempre o dobro das recebidas.

A Tabela 4.2 mostra os dados coletados pelo OMNeT++ em termos de média mensagens enviadas e reenviadas pelos robôs durante os experimentos com 400 segundos de duração.

Tabela 4.2 - Mensagens Enviadas, Recebidas e Reenviadas pelos Robôs nas simulações de 400 segundos.

Simulação	Robô	Enviadas	Recebidas	Reenviadas
1	1	422	211	9
	2	376	188	3
2	1	422	211	4
	2	376	188	5
3	1	422	211	7
	2	378	189	7
4	1	422	211	3
	2	378	189	7
5	1	422	211	10
	2	378	189	11
Médias		**399,7**	**199,8**	**6,6**

De acordo com a tabela 4.2, há um total de 3996 mensagens enviadas (4 perdidas pelo CERTI), 1998 mensagens recebidas pelos robôs (a outra metade era encaminhada para o Federate, como dito anteriormente) e 66 mensagens precisaram ser reenviadas. O quadro completo dos dados coletados durante os experimentos de validação da arquitetura proposta pode ser visto em Apêndice A.

É importante destacar que com o passar dos experimentos, constatou-se a perda de alguns pacotes, como pode-se observar no Apêndice A. Ao final das 20 simulações, 7500 mensagens deveriam ter sido disparadas, porém 16 mensagens foram “perdidas” pelo ambiente de simulação distribuída, o CERTI, o que leva a cerca de 0,2% de mensagens perdidas independente dos demais simuladores (nos dados sobre as mensagens enviadas pelos robôs, a soma dos valores da coluna Enviadas deveria ser de 7500, ao invés de 7484).

Com o fim dos testes de validação, maior ênfase foi dada à parte ROS da simulação, especialmente no que diz respeito a como a movimentação dos robôs foi influenciada pelo OMNeT++.

5 EXPERIMENTOS REALIZADOS

Após a validação da arquitetura proposta, foi criado um cenário para que os experimentos pudessem ser efetuados e os dados, coletados. Assim como nos testes de validação, existem N robôs, onde há um robô chamado de mestre e N-1 robôs escravos. A principal diferença agora é que existe uma preocupação quanto à movimentação dos robôs e como ela é afetada pela inclusão do OMNeT++.

Dessa forma, foram utilizadas duas configurações de simulação: a primeira contendo apenas o ROS e o HLA, chamada Tipo 01; e a segunda contando com o ROS, o HLA e o OMNeT++, a Tipo 02. Em ambas as configurações o comportamento dos robôs é o mesmo: o robô mestre segue de uma trajetória X_0Y_0 para X_1Y_1 de forma autônoma e independente dos demais simuladores. À medida que o robô mestre avança, ele repassa sua posição atual para os robôs escravos. Uma vez que os robôs escravos recebem essa informação, passam a se locomover para a posição do robô mestre. Na primeira configuração isso é feito diretamente. Já na segunda, a mensagem segue o fluxo já explicado na Figura 4.5 no capítulo anterior.

A comunicação entre os robôs foi feita utilizando o protocolo UDP (*User Datagram Protocol*) em uma rede baseada no protocolo IEEE 802.15.4, que segundo MOLISCH et al. (2004), cobre pequenas áreas, como residências, escritórios, ambientes ao ar livre, etc.

Com UDP não há controle de erros, ou seja, não há orientação para a conexão. Uma vez que há perda de pacotes durante a comunicação, este é descartado e os robôs envolvidos na simulação ficam esperando por uma próxima mensagem.

Em ambas as configurações, o robô mestre parte do ponto X_0Y_0, representado pelo ponto (4.0,4.0) e se movimenta até o ponto X_1Y_1, de valor (145,120). A movimentação dos robôs foi capturada a cada ciclo de simulação, gerando arquivos CSV (*comma-separated values*) contendo o tempo de execução (hh:mm:ss:mmmmmm), ponto X, Y que representa sua localização atual e outro ponto X, Y que representa a posição para a qual deve se mover naquele instante.

A Tabela 5.1 mostra a movimentação do robô mestre para as simulações de 300, 600 e 1200 segundos de duração. É importante frisar que o robô mestre se move de maneira autônoma, ou seja, independe de qualquer fator externo. Dessa forma, os valores

de movimentação para o robô mestre obtidos nos experimentos do Tipo 01 são os mesmos para os de Tipo 02, uma vez que apenas a movimentação de robôs escravos é influenciada pelo OMNeT++.

Tabela 5.1 - Movimentação do Robô Mestre para as simulações de 300, 600 e 1200 segundos

Tempo de Execução	Posição Inicial (X;Y)	Distância Percorrida
300	4.0;4.0	38.98
600	4.0;4.0	85.29
1200	4.0;4.0	172.97

5.1. Dados Coletados para os Experimentos do Tipo 01

A primeira parte dos experimentos consistiu em executar as simulações do ROS para um robô mestre e um escravo com durações de 300, 600 e 1200 segundos. A Tabela 03 apresenta a movimentação do robô mestre e do escravo para cada tempo de simulação. Uma vez que a primeira parte foi concluída, o mesmo foi feito acrescentando-se o OMNeT++ (Experimentos do Tipo 02). A Tabela 5.2 mostra a movimentação dos robôs escravos no que diz respeito a distância percorrida nos experimentos do Tipo 01.

O objetivo desses experimentos é fazer com que os robôs escravos saiam de suas posições iniciais e se desloquem até próximo da posição do robô mestre. Como dito anteriormente, a cada ciclo, o robô mestre irá informar sua posição atual para que os robôs escravos sejam informados do ponto para o qual devem seguir.

Para os experimentos do tipo 01, uma vez que o simulador de Redes não foi utilizado, não há a necessidade de realização de diversas simulações, uma vez que não há fatores externos afetando a realização do experimento. Aqui, é simulado o ambiente ideal, o que não ocorre nos experimentos do tipo 02, como será visto posteriormente.

Tabela 5.2 – Distância Total percorrida pelo Robô Escravo ao final das simulações que não utilizaram o OMNeT++ (Tipo 01).

Tempo de Execução	Posição Inicial (X;Y)		Distância Percorrida
300	0.0	0.0	44.43
600	0.0	0.0	89.15
1200	0.0	0.0	178.52

As Figuras 5.1, 5.2 e 5.3 mostram o trajeto percorrido pelos robôs durante os experimentos do Tipo 01 para as simulações de 300, 600 e 1200 segundos, respectivamente. Nelas, as linhas pretas representam a movimentação do robô mestre e as verdes, do escravo. Tanto as Figuras 5.1, 5.2 e 5.3, quanto a Tabela 5.2 serão utilizadas na comparação com os dados referentes às simulações do Tipo 02.

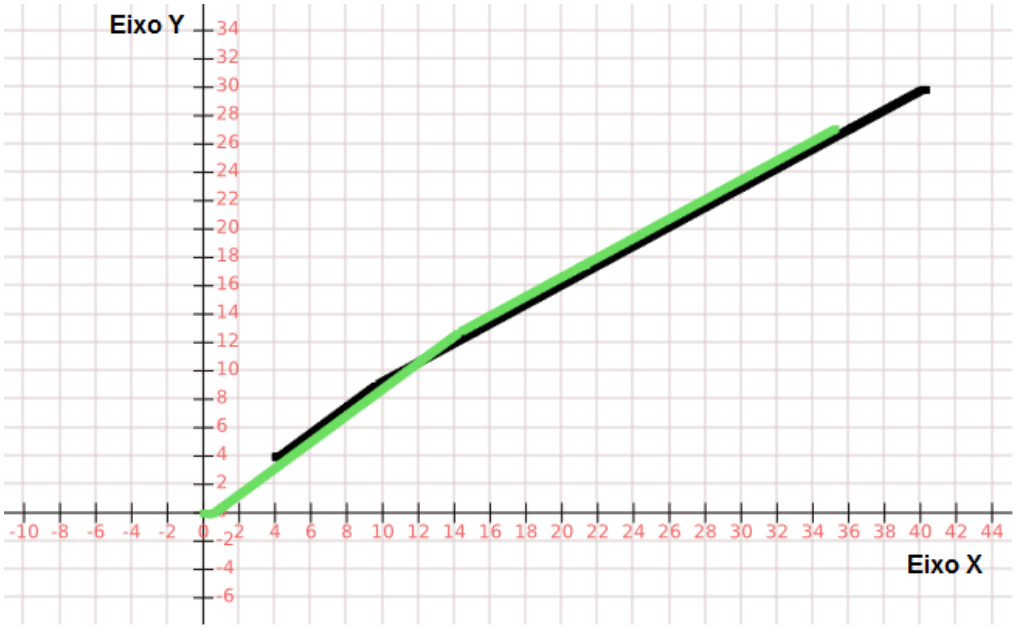


Figura 5.1 - Movimentação dos Robôs para o experimento do Tipo 01 com 300 segundos de duração. As linhas pretas representam a movimentação do robô mestre e as verdes, do escravo.

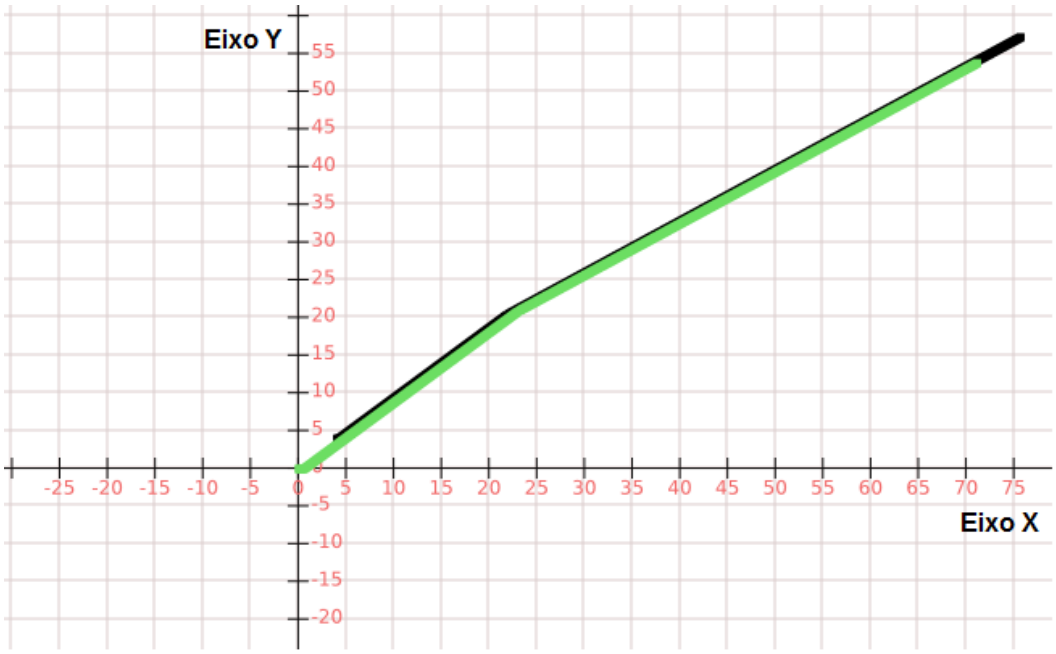


Figura 5.2 - Movimentação dos Robôs para o experimento do Tipo 01 com 600 segundos de duração. As linhas pretas representam a movimentação do robô mestre e as verdes, do escravo.

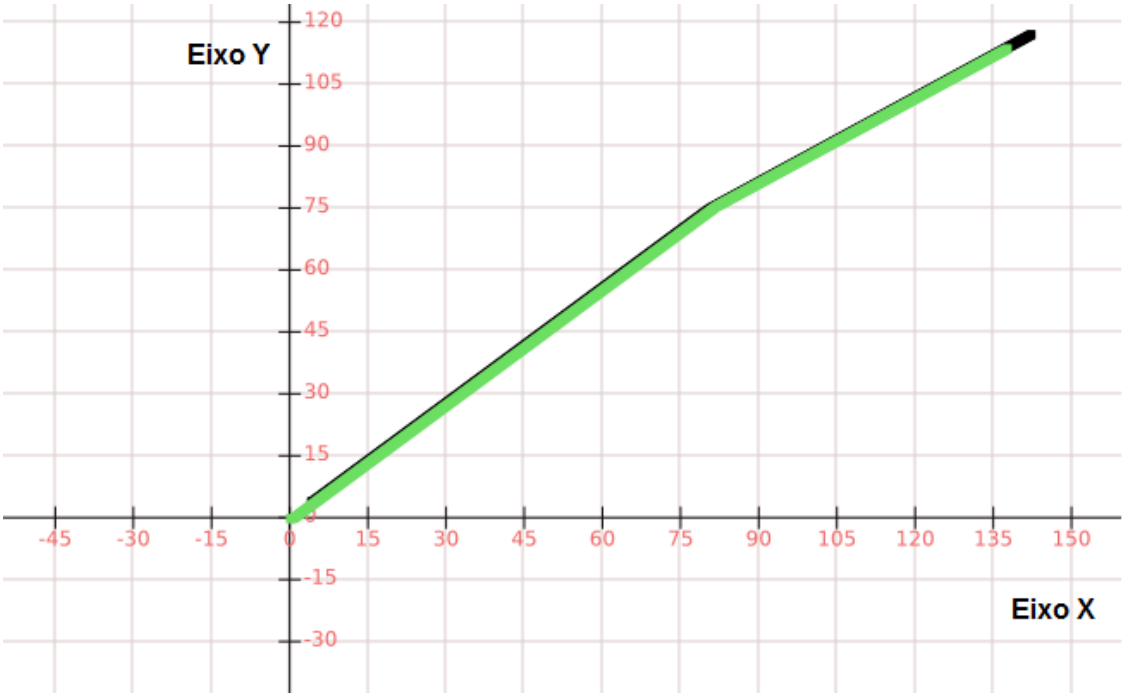


Figura 5.3 - Movimentação dos Robôs para o experimento do Tipo 01 com 1200 segundos de duração. As linhas pretas representam a movimentação do robô mestre e as verdes, do escravo.

Uma vez concluído o levantamento dos dados para a primeira configuração dos experimentos, o mesmo processo foi realizado para os experimentos da segunda configuração, de modo que torne possível realizar a comparação e a constatação do impacto do OMNeT++ no resultado dos experimentos.

5.2. Dados Coletados para os Experimentos do Tipo 02

Como já dito anteriormente, a segunda parte dos experimentos consistiu em executar as simulações utilizando o ROS, o HLA e o OMNeT++ para um robô mestre e um escravo com durações de 300, 600 e 1200 segundos. No total, foram feitas 75 simulações, sendo 25 para cada tempo de execução, sendo 5 para cada percentual de perda de pacotes (1%, 2%, 3%, 4%, e 5%).

A Tabela 5.3 mostra a distância média percorrida pelos robôs para cada tempo de execução (não sendo considerados os percentuais de perda de pacote). A escolha pela média foi necessária devido ao número de simulações realizadas para o Tipo 02, optando pela média e não por resultados isolados.

Tabela 5.3 - Distância final média percorrida pelo Robô Escravo ao final das simulações do Tipo 02.

Tempo de Execução	Posição Inicial (X;Y)		Distância Percorrida	Perdas
300	0.0	0.0	43.18	09
600	0.0	0.0	86.76	17
1200	0.0	0.0	172.88	39

Os dados obtidos pela Tabela 5.3 serão utilizados para comparar a diferença entre as distâncias percorridas para os Robôs Escravos pertencentes aos dois tipos de simulação.

A perda de Pacotes, existente nas simulações do Tipo 02, influenciou diretamente a distância percorrida pelo robô escravo, uma vez que a distância média percorrida pelo Robô Escravo diminui à medida que cresce o número de pacotes perdidos, como pode ser visto nas Tabelas 5.4, 5.5 e 5.6. Elas contêm os dados para as simulações com taxa de perda de pacote de 3% para cada um dos tempos simulados. As simulações com 3% de taxa de perda de pacotes foram escolhidas para compor os gráficos 5.4, 5.5 e 5.6 por se tratarem de um valor médio, uma vez que esses percentuais variam entre 1 e 5 por cento. Os dados dos Robôs Escravos para todas as simulações podem ser vistos no Apêndice B.

Tabela 5.4 - Influência da Perda de Pacotes na Movimentação do Robô Escravo para simulações com 3% de perda de pacote e 300 segundos de duração

Simulação	Pacotes Perdidos	Distância Percorrida
11	11	43.08
12	4	43.67
13	4	43.67
14	11	42.96
15	15	42.19

Tabela 5.5 - Influência da Perda de Pacotes na Movimentação do Robô escravo para simulações com 3% de perda de pacote e 600 segundos de duração

Simulação	Pacotes Perdidos	Distância Percorrida
36	19	86.45
37	18	86.61
38	19	86.30
39	25	85.56
40	12	87.50

Tabela 5.6 - Influência da Perda de Pacotes na Movimentação do Robô escravo para simulações com 3% de perda de pacote e 1200 segundos de duração

Simulação	Pacotes Perdidos	Distância Percorrida
61	32	173.91
62	48	171.54
63	43	172.27
64	38	173.02
65	35	173.47

A partir das Tabelas 5.4, 5.5 e 5.6 percebe-se a direta influência da perda de pacotes em relação à distância percorrida pelo robô escravo. A partir delas é possível notar que as menores distâncias percorridas aconteceram nas simulações com maior perda de pacotes. Isso pode ser notado ao observarmos as simulações de número 15, com 15 perdas e distância percorrida de 42.19, 39, com 25 perdas e distância percorrida de 85.56 e 62, com 48 perdas e distância percorrida de 171.54.

A Tabela 5.7 mostra, para cada tempo simulado, o total de pacotes perdidos, as menores e as maiores distâncias percorridas.

Tabela 5.7 - Perda de Pacotes, maiores e menores distâncias percorridas para cada tempo simulado

Tempo de Simulação	Total de Pacotes Perdidos	Maior Distância Percorrida		Menor Distância Percorrida	
		Perdas	Distância Percorrida	Perdas	Distância Percorrida
300	234	2	44.28	21	41.44
600	422	4	88.69	38	83.62
1200	972	5	177.77	82	166.48

A partir da Tabela 5.7 pôde-se constatar um total de 1628 pacotes perdidos (soma da coluna “Total de Pacotes Perdidos”), com média de 9.36, 16.88 e 38 para 300, 600 e 1200 segundos, respectivamente. A média de perda de pacotes foi obtida dividindo o número de pacotes perdidos por 25 (número de simulações feitas para cada tempo). A Tabela contendo todos os dados coletados durante os experimentos do Tipo 02 está no Apêndice B desta dissertação.

Para simulações de 300 segundos, a menor perda identificada foi na simulação de número 6, onde apenas 2 pacotes foram perdidos, o que resultou na maior distância percorrida: 44.28. Já a maior perda foi identificada na simulação de número 24, onde 21 pacotes foram perdidos, o que resultou na menor distância percorrida: 41.44.

Para simulações de 600 segundos, a menor perda identificada foi nas simulações 27 e 28, onde 4 pacotes foram perdidos, o que resultou na maior distância percorrida: 88.69. Já a maior perda foi identificada na simulação de número 49, onde 38 pacotes foram perdidos, o que resultou na menor distância percorrida: 83.62.

A simulação com maior perda de pacotes entre todas as simulações foi a de número 71, onde 82 pacotes foram perdidos, o que resultou na menor distância percorrida para as simulações de 1200 segundos: 166.48. A menor perda identificada mas simulações de 1200 segundos foi na simulação 55, onde 5 pacotes foram perdidos, o que resultou na maior distância percorrida: 177.77.

A Tabela 5.8 mostra a diferença entre a distância percorrida entre as simulações com maiores e menores perdas de pacote para cada tempo simulado. Nela, fica claro o impacto da perda de pacote na distância percorrida pelo Robô Escravo, uma vez que a distância percorrida diminui à medida que perdem-se mais pacotes.

Tabela 5.8 - Diferença entre a distância percorrida pelo Robô Escravo para as simulações com maiores e menores perdas de Pacotes

Tempo de Simulação	Maior Distância Percorrida		Maior Distância Percorrida		Diferença
	Perdas	Distância Percorrida	Perdas	Distância Percorrida	
300	2	44.28	21	41.44	2.84
600	4	88.69	38	83.62	5.07
1200	5	177.77	82	166.48	11.29

A Tabela 5.8 mostra a diferença de movimentação causada pela perda de pacotes, uma vez que a diferença de 19 pacotes, nas simulações de 300 segundos, resultou em uma diferença na movimentação de 2.84. O mesmo vale para as simulações de 600 segundos, onde a diferença de pacotes perdidos foi de 34, resultando em uma diferença de movimentação de 5.07. Por fim, para as simulações de 1200 segundos, a diferença de pacotes perdidos chegou a 77, com diferença de movimentação de 11.29.

Como forma de mostrar o impacto da perda de pacotes na distância percorrida pelo robô escravo, as figuras 11, 12 e 13 mostram, respectivamente, o trajeto percorrido pelos robôs durante os experimentos do Tipo 02 para as simulações de 300, 600 e 1200 segundos onde ocorreu maior perda de pacotes. Os experimentos utilizados nas figuras são o 24 (21), o 49 (38) e o 71 (82). Nelas, as linhas pretas representam a movimentação do robô mestre e as verdes, do escravo.

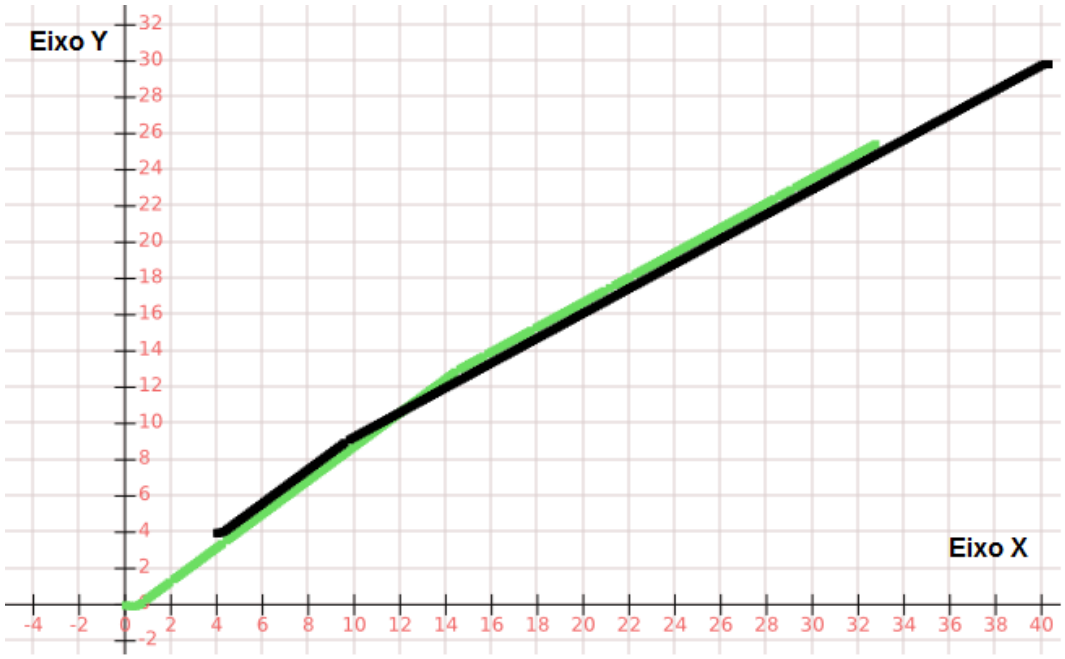


Figura 5.4 - Movimentação dos Robôs para o experimento do Tipo 02, número 24, com 300 segundos de duração. As linhas pretas representam a movimentação do robô mestre e as verdes, do escravo.

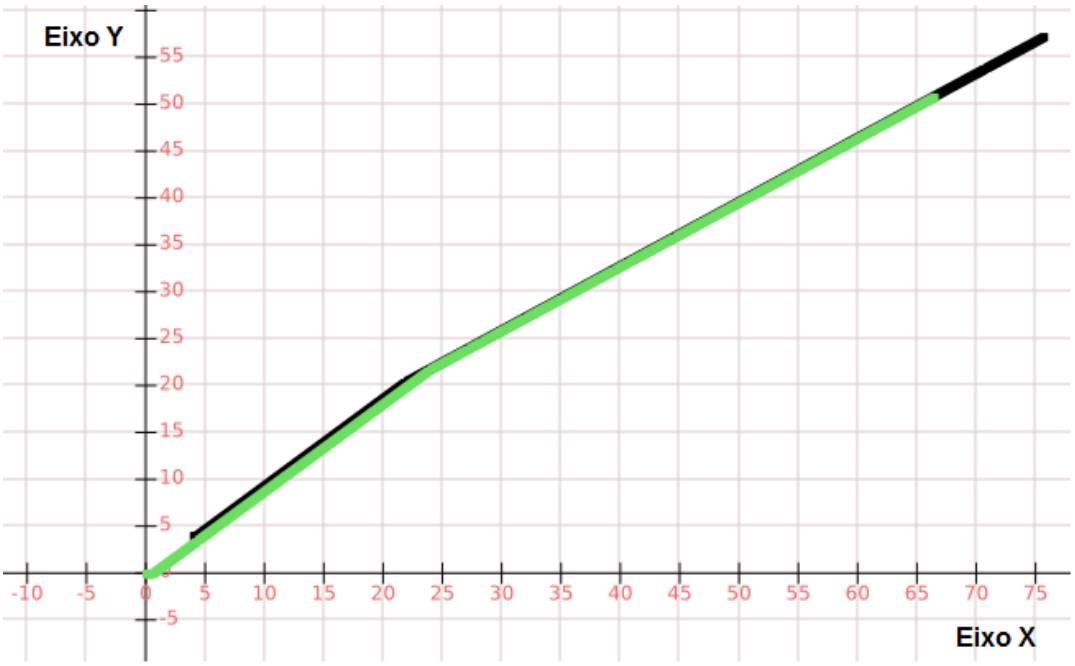


Figura 5.5 - Movimentação dos Robôs para o experimento do Tipo 02, número 49, com 600 segundos de duração. As linhas pretas representam a movimentação do robô mestre e as verdes, do escravo.

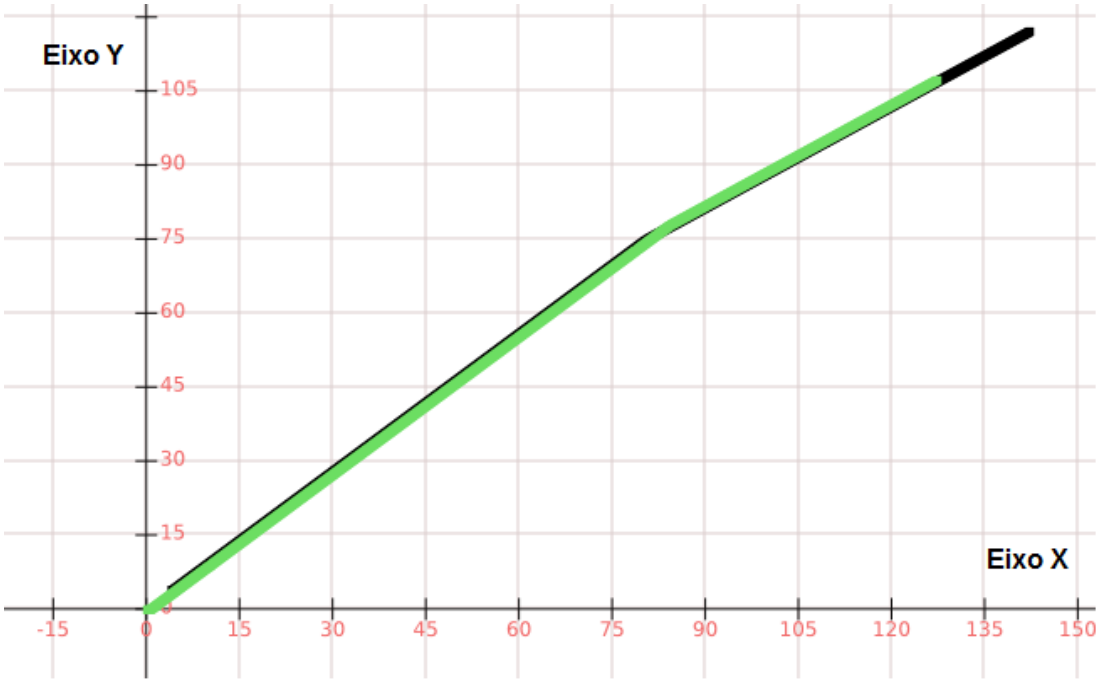


Figura 5.6 - Movimentação dos Robôs para o experimento do Tipo 02, número 71, com 1200 segundos de duração. As linhas pretas representam a movimentação do robô mestre e as verdes, do escravo.

As figuras 5.5, 5.6 e 5.7 servem para mostrar de forma gráfica a distância final entre o robô mestre e escravo. Quando comparadas às figuras 5.2, 5.3 e 5.4, percebe-se que os robôs escravos sempre terminaram as simulações em uma posição mais distante quando comparado com os robôs escravos nas simulações de Tipo 01.

A seguir, serão feitos os comparativos entre os dois tipos de experimentos apresentados anteriormente. Esse comparativo servirá para comprovar a alteração do comportamento dos robôs após a inclusão do simulador de redes, o OMNeT++, objetivo deste trabalho.

5.3. COMPARANDO AS DUAS ABORDAGENS

A Tabela 5.9 compara os dados mostrados pelas Tabelas 5.2 e 5.3, para que se torne mais clara a diferença entre as distâncias percorridas. A partir dela, podemos perceber que para todos os casos, o Robô Escravo sempre se movimentou menos nas simulações do Tipo 02, uma vez que nelas ocorreram perda de pacotes.

Tabela 5.9 - Diferença entre a distância média percorrida (no eixo XY) do Robô Escravo para as simulações do Tipo 01 em relação ao do Tipo 02

Tempo de Execução	Tipo 01	Tipo 02		Diferença
	Distância Percorrida	Distância Percorrida	Perdas	
300	44.43	43.18	09	1.25
600	89.15	86.76	17	2,39
1200	178.52	172.88	39	5,64

Para dar mais ênfase ao impacto da perda de pacotes na movimentação do Robô Escravo, vejamos os casos onde houve menor perda de pacotes. A Tabela 5.10 mostra a diferença na distância percorrida para as simulações com as menores perdas de pacotes. A partir dela, pode-se notar que a diferença na distância percorrida foi menor do que a distância mostrada na Tabela 5.9, que utilizou dados obtidos a partir da média de perda de pacotes e distância percorrida.

Tabela 5.10 - Diferença entre a distância percorrida (no eixo XY) do Robô Escravo para as simulações do Tipo 01 com menor perda de pacotes em relação ao do Tipo 02

Tempo de Execução	Tipo 01	Tipo 02		Diferença
	Distância Percorrida	Distância Percorrida	Perdas	
300	44.43	44.28	02	0.15
600	89.15	88.69	04	0.46
1200	178.52	177.77	05	0.75

A partir dos dados presentes na Tabela 5.10, é possível notar a diferença mínima entre as distâncias percorridas nos casos onde houve menor perda de pacotes, provando novamente o impacto da perda de pacotes na movimentação do Robô Escravo.

Comparando os dados das Tabelas 5.9 e 5.10, pode-se perceber que para as simulações de 300 segundos, a diferença na distância percorrida entre o Robô Escravo do Tipo 01 para o do Tipo 02 era de 1.25 e nos casos de menor perda de pacote chegou a ser 0.15. De forma semelhante, essa diferença baixou de 2.39 para 0.46, nas simulações de 600 segundos e de 5.64 para 0.75, nas simulações de 1200 segundos.

As Figuras 5.8, 5.9 e 5.10 mostram um comparativo entre a movimentação dos robôs escravos com base nas informações apresentadas nas figuras correspondentes, comparando os dados das Figuras 5.2 e 5.5, 5.3 e 5.6; e 5.4 e 5.7, de forma a percebermos essas diferenças de movimentação para cada tempo de simulação. As linhas pretas representam a movimentação dos robôs mestres; as verdes, os robôs escravos para as simulações do Tipo 01; e, por fim, as azuis representam a movimentação dos robôs escravos para as simulações do Tipo 02.

Figura 5.7 - Comparativo relativo à movimentação do robô mestre e dos robôs escravos para simulações com duração de 300 segundos. As linhas pretas representam a movimentação dos robôs mestres; as verdes, os robôs escravos para as simulações do Tipo 01; e as azuis, os robôs escravos para as simulações do Tipo 02.

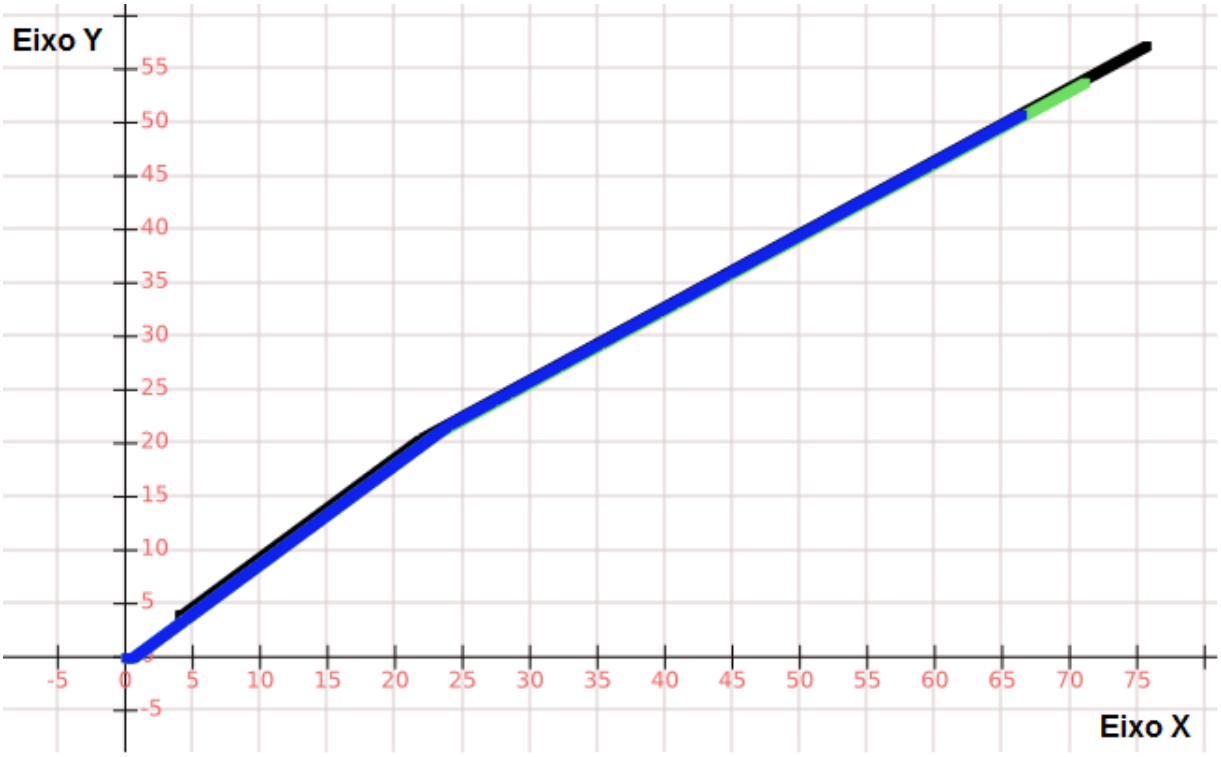


Figura 5.8 - Comparativo relativo à movimentação do robô mestre e dos robôs escravos para simulações com duração de 600 segundos. As linhas pretas representam a movimentação dos robôs mestres; as verdes, os robôs escravos para as simulações do Tipo 01; e as azuis, os robôs escravos para as simulações do Tipo 02.

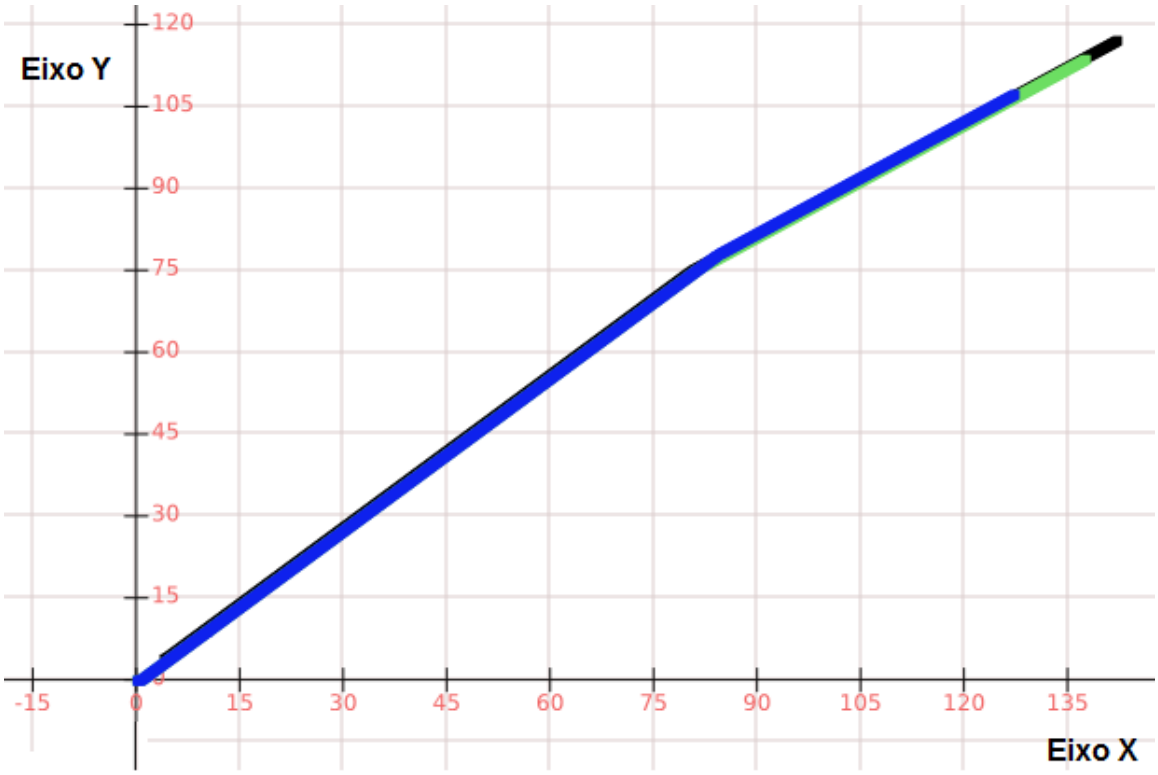


Figura 5.9 - Comparativo relativo à movimentação do robô mestre e dos robôs escravos para simulações com duração de 1200 segundos. As linhas pretas representam a movimentação dos robôs

mestres; as verdes, os robôs escravos para as simulações do Tipo 01; e as azuis, os robôs escravos para as simulações do Tipo 02.

A partir das Figuras 5.8, 5.9 e 5.10, percebe-se que os robôs escravos do Tipo 02 sempre tiveram menor distância percorrida que os do Tipo 01, influência direta da perda de pacotes e do delay da comunicação no OMNeT++.

Como dito anteriormente, o objetivo principal desta dissertação era tornar as simulações de SMRs mais realistas, uma vez que no mundo real, os robôs trocam informações através de uma rede.

Os experimentos realizados mostraram que uma vez que as simulações sairam do seu “ambiente ideal”, ocorreu uma mudança significativa no comportamento dos robôs. Isso foi feito através de dois pontos cruciais que foram possíveis através da inclusão do simulador de rede: a perda de pacotes, que simula uma “falha” na troca de informações entre os robôs, e a latência na troca de informações, que faz com que haja um atraso no envio/recebimento de informações.

Esses são dois fatores importantes para SMRs, como dito anteriormente, uma vez que os robôs irão atuar em tempo real e falhas e/ou atrasos na troca de informações nem sempre são tolerados.

O capítulo seguinte irá falar sobre alguns desafios encontrados, considerações finais e trabalhos futuros.

6 DISCUSSÕES FINAIS E TRABALHOS FUTUROS

Simulações exercem um papel importante na hora de validar *Cyber-Physical* Systems (CPSs) uma vez que reduzem os custos e riscos durante a fase de projeto e testes. Para que elas sejam confiáveis, é necessário uma modelagem realista para os aspectos físicos e cibernéticos (ALHARTHI, 2014). O mesmo acontece com simulações de SMRs. Para que simulações sejam efetivamente adequadas, aspectos do mundo físico precisam ser levados em consideração. No contexto deste trabalho, a troca de informações foi o ponto chave a ser explorado.

Até então, simulações feitas em SMRs utilizavam, entre outras ferramentas, o *Robot-Operation System* (ROS), como falado anteriormente. Porém, sem que aspectos externos sejam levados em consideração, não há garantias de fidedignidade quanto aos resultados obtidos, uma vez que alguns aspectos foram negligênciados, como a comunicação através de uma rede de computadores.

O objetivo desta dissertação foi apresentar um ambiente onde SMRs possam ser testados levando em consideração também a troca de informações em redes de computadores, uma vez que sistemas compostos por várias entidades distribuídas em um espaço podem interagir umas com as outras para enviar informações ou cumprir um objetivo (ALHARTHI, 2014). Isso foi feito através da utilização de cossimulação, onde o ROS, o CERTI e o OMNeT++ puderam estar trabalhando de forma sincronizada e trocando informações. Portanto, o primeiro desafio foi fazer com que essas três soluções pudessem trocar informações, criando um grande ambiente de simulação, uma vez se tratam de dois simuladores com propósitos distintos e uma ferramenta que se propõe a integrá-los.

Durante os experimentos, pôde-se provar efetivamente o impacto que o OMNeT++ obteve no comportamento dos robôs. Entretanto, isto não quer dizer que não existam mais fatores que também podem exercer tal influência. O que não impede que isso seja feito em trabalhos futuros.

Dois fatores principais observados a partir do ambiente de simulação proposto nesta dissertação, como dito, são a perda de pacote e a latência na transmissão de informação. Esses dois fatores influenciaram diretamente o comportamento dos robôs simulados, como mostrado no capítulo anterior. Sendo assim, podemos concluir que as simulações feitas

durante os experimentos são mais próximas da realidade quando comparadas com as simulações que só utilizam o ROS como plataforma de simulação.

Com o sucesso na utilização do OMNeT++ em conjunto com o ROS, outros fatores poderão ser explorados, como outros protocolos de comunicação, por exemplo. Uma vez que o OMNeT++ dispõe de frameworks que simulam diversos protocolos de redes (TCP, UDP, IPv4, IPv6, OSPF, BGP, entre outros), redes com ou sem fio (Ethernet, PPP, IEEE 802.11, IEEE 802.15.1, IEEE 802.15.4, entre outros), testes mais profundos podem ser feitos utilizando o ambiente desenvolvido durante esta dissertação, uma vez que esteja integrado com algum dos frameworks disponibilizados pelo OMNeT++.

Além dos resultados apresentados, este trabalho rendeu a publicação do artigo entitulado "*Simulation and Test of Communication in Multi-Robot Systems using Co-Simulation*" no WorldCist'16 - 4th World Conference on Information Systems and Technologies, realizado entre os dias 22 e 24 de Março, em Recife – Pernambuco – Brasil. O WorldCist é um fórum global para pesquisadores e profissionais para apresentar e discutir as mais recentes inovações, tendências, resultados, experiências e preocupações nas diversas perspectivas de Sistemas e Tecnologias de Informação.

REFERÊNCIAS

ALHARTHI, Mohannad. **Co-simulation Environment for Modeling Networked Cyber-Physical Systems**. Dissertação de Mestrado - Queen's University, Kingston, Ontario, Canada. Abril, 2014.

AMORY, A. et al. **A heterogeneous and distributed co-simulation environment [*hardware*/*software*]**. In: Integrated Circuits and Systems Design, 2002. Proceedings. 15th Symposium on. IEEE, 2002. p. 115-120.

AKIN, H. Levent et al. **Two "Hot Issues" in Cooperative Robotics: Network Robot Systems, and Formal Models and Methods for Cooperation**. EURON Special Interest Group on Cooperative Robotics, 2008.L.

ARAI, T.; PAGELLO, E.; PARKER, L. E. **Editorial: Advances in Multi-Robot Systems**. IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, Outubro 2002, pp. 655-661.

BAHETI, R., GILL, H., **Cyber-physical systems**. The impact of control technology, v. 12, p. 161-166, 2011.

BOTELHO.S, ALAMI.R. **M+: A Scheme for Multi-Robot Cooperation through Negotiated Task Allocation and Achievement**. Proc of the 1999 IEEE International Conf on Robotics and Automation, Detroit, Michigan, pp: 1234-1239, 1999.

BRITO, Alisson V. et al. **Development and Evaluation of Distributed Simulation of Embedded Systems Using Ptolemy and HLA**. In: Distributed Simulation and Real Time Applications (DS-RT), 2013 IEEE/ACM 17th International Symposium on. IEEE, 2013. p. 189-196.

BROENINK, J. F., NI, Y., GROOTHUIS, M. **A. On Model-Driven Design of Robot *Software* Using Co-Simulation**. Proceedings of SIMPAR Workshops Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS. Darmstadt (Germany) November 15-16, 2010.

BUCK, Sebastian; BEETZ, Michael; SCHMITT, Thorsten. M-ROSE: **A Multi Robot Simulation Environment for Learning Cooperative Behavior**. In: Distributed Autonomous Robotic Systems 5. Springer Japão, 2002. p. 197-206.

CALISKANELLI, I., BROECKER, B., TUYLS, K. **Multi-Robot Coverage: A Bee Pheromone Signalling Approach**. Artificial Life and Intelligent Agent Research Symposium. Bangor University - United Kingdom. 2014.

CAO, H. et al. **Complex Tasks Allocation for Multi Robot Teams under Communication Constraints**. In: Proc. of the 5th National Conference on Control Architecture of Robots. 2010.

CARPIN, Stefano et al. **USARSim: a robot simulator for research and education**. In: Robotics and Automation, 2007 IEEE International Conference on. IEEE, 2007. p. 1400-1405.

CHEN, Qi et al. **Overhaul of IEEE 802.11 Modeling and Simulation in NS-2**. In: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems. ACM, 2007. p. 159-168.

DAHMANN, Judith S.; FUJIMOTO, Richard M.; WEAkTHERLY, Richard M. **The department of defense high level architecture**. In: Proceedings of the 29th conference on Winter simulation. IEEE Computer Society, 1997. p. 142-149.

DE, Pradipta et al. **MiNT-m: an autonomous mobile wireless experimentation platform**. In: Proceedings of the 4th international conference on Mobile systems, applications and services. ACM, 2006. p. 124-137.

DREIBHOLZ, Thomas; RATHGEB, Erwin P.; ZHOU, Xing. **SimProcTC: the design and realization of a powerful tool-chain for OMNeT++ simulations**. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009. p. 75.

ERGEN, Mustafa. **IEEE 802.11 Tutorial**. University of California Berkeley, v. 70, 2002.

FITZGERALD, John S. et al. **A formal Approach to Collaborative Modelling and Co-simulation for Embedded Systems**. Mathematical Structures in Computer Science, v. 23, n. 04, p. 726-750, 2013.

GERKEY, B. e MATARIC, M. **A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems**. International Journal of Robotics Research, 23(9):939–954, 2004.

GERKEY, B. P.; MATARIC, M. J. **A Framework for Studying Multi-Robot Task Allocation**. 2003.

GERKEY, Brian; VAUGHAN, Richard T.; HOWARD, Andrew. **The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems**. In: Proceedings of the 11th international conference on advanced robotics. 2003. p. 317-323.

HENDERSON, Thomas R. et al. **Network Simulations with the Ns-3 Simulator**. SIGCOMM demonstration, v. 14, 2008.

HIERTZ, Guido R. et al. **The IEEE 802.11 universe**. Communications Magazine, IEEE, v. 48, n. 1, p. 62-70, 2010.

IEEE. "**IEEE Standard for Modeling and Simulation – High Level Architecture (HLA) – Federate Interface Specification**," IEEE Std 1516.1-2010 (Revision of IEEE Std 1516.1-2000), pp. 1–378, 2010.

IEEE 802.11 Working Group. **IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirement**. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical report, IEEE, Inc, February 2012.

IEEE 802.15.4, **Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)**, IEEE, October 1 2003.

JIANG, Daniel; DELGROSSI, Luca. IEEE 802.11p: **Towards an international standard for wireless access in vehicular environments**. In: Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE. IEEE, 2008. p. 2036-2040.

KITANO, Hiroaki et al. Robocup: The robot world cup initiative. Proceedings of the first international conference on Autonomous agents. ACM, 1997. pp. 340-347.

KIESS, Wolfgang; MAUVE, Martin. **A survey on real-world implementations of mobile ad-hoc networks**. Ad Hoc Networks, v. 5, n. 3, p. 324-339, 2007.

KÖPKE, Andreas et al. **Simulating wireless and mobile networks in OMNeT++ the MiXiM vision**. In: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. p. 71.

KOTZ, David et al. **Experimental evaluation of wireless simulation assumptions**. In: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems. ACM, 2004. p. 78-82.

KUDELSKI, Michal; GAMBARDELLA, Luca M.; DI CARO, Gianni A. **RoboNetSim: An integrated framework for multi-robot and network simulation**. Robotics and Autonomous Systems, v. 61, n. 5, p. 483-496, 2013.

LEE, Edward et al. **Cyber physical systems: Design challenges**. In: Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on. IEEE, 2008. p. 363-369.

LERMAN, K. et al. **Analysis of Dynamic Task Allocation in Multi-Robot Systems**. The International Journal of Robotics Research, v. 25, n. 3, p. 225-241, 2006.

MALLANDA, C. et al. **Simulating wireless sensor networks with omnet++**. Submitted to IEEE Computer, 2005.

MAYER, Christoph P.; GAMER, Thomas. **Integrating real world applications into OMNeT++**. Institute of Telematics, University of Karlsruhe, Karlsruhe, Germany, Tech. Rep. TM-2008-2, 2008.

MICHEL, Olivier. **Webots: Symbiosis between virtual and real mobile robots**. In: Virtual Worlds. Springer Berlin Heidelberg, 1998. p. 254-263.

MOLISCH, Andreas F. et al**. IEEE 802.15. 4a channel model-final report**. IEEE P802, v. 15, n. 04, p. 0662, 2004.

NAGEL, Robert; EICHLER, Stephan. **Efficient and realistic mobility and channel modeling for VANET scenarios using OMNeT++ and INET-framework**. In: Proceedings of the 1st international conference on Simulation tools and techniques for communications,

networks and systems & workshops. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. p. 89.

NETHI, Shekar et al. **Platform for emulating networked control systems in laboratory environments**. In: World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a. IEEE, 2007. p. 1-8.

NICOLESCU, G. et al. **CODIS–A Framework for Continuous/Discrete Systems Co-Simulation**. Analysis and Design of Hybrid Systems, p. 274-275, 2006.

NICOLESCU, G. et al. **Methodology for efficient design of continuous/discrete-events co-simulation tools**. High Level Simulation Languages and Applications-HLSLA. SCS, San Diego, CA, p. 172-179, 2007.

PARK, Tae Rim et al. **Throughput and energy consumption analysis of IEEE 802.15. 4 slotted CSMA/CA**. Electronics Letters, v. 41, n. 18, p. 1017-1019, 2005.

PARKER, L. E. **ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation**. Robotics and Automation, IEEE Transactions on, v. 14, n. 2, p. 220-240, 1998.

PINCIROLI, C. et al. **ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm intelligence**. v. 6, n. 4, p. 271-295, 2012.

PFEIFER, D., VALVANO, J. **Kahn Process Networks Applied to Distributed Heterogeneous HW/SW Cosimulation**. In: Electronic System Level Synthesis Conference (ESLsyn), 2011. IEEE, 2011. p. 1-6.

POHJOLA, Mikael; NETHI, Shekar; JÄNTTI, Riku. **Wireless control of mobile robot squad with link failure**. In: Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on. IEEE, 2008. p. 648-656.

POOVENDRAN, Radha et al. **Special Issue on Cyber–Physical Systems [Scanning the Issue]**. Proceedings of the IEEE, v. 1, n. 100, p. 6-12, 2012.

QUIGLEY, Morgan et al. **ROS: an open-source Robot Operating System**. In: ICRA workshop on open source *software*. 2009. p. 5.

RAJKUMAR, R. et al. **Cyber-physical systems: the next computing revolution**. In: Proceedings of the 47th Design Automation Conference. ACM, 2010. p. 731-736.

ROTH, Christoph et al. **A Simulation Tool Chain for Investigating Future V2X-based Automotive E/E Architectures**. Embedded Real Time *Software* and Systems (ERTS), v. 2, p. 1739-1748, 2014.

SANTOS F.: **Architecture for Real-Time Coordination of Multiple Autonomous Mobile Units** – PhD Thesis - Universidade de Aveiro – 2014.

SHIROMA, P. M.; CAMPOS, M. **CoMutaR: A Framework for Multi-Robot Coordination and Task Allocation**. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. IEEE, 2009. p. 4817-4824.

STENTZ, A.; DIAS, M. **A Free Market Architecture for Coordinating Multiple Robots**. Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 1999.

SUNG, C. et al. **Interoperation of DEVS models and differential equation models using HLA/RTI: hybrid simulation of engineering and engagement level models.** In: Proceedings of the 2009 Spring Simulation MultiConference. Society for Computer Simulation International, 2009. p. 150.

SUNG, C., KIM, T. **Framework for simulation of hybrid systems: Interoperation of discrete event and continuous simulators using HLA/RTI**. In: Proceedings of the 2011 IEEE Workshop on Principles of Advanced and Distributed Simulation. IEEE Computer Society, 2011. p. 1-8.

SOUZA, U. R. F et al. **Tangram—Virtual Integration of Heterogeneous IP Components in a Distributed Co-Simulation Environment**. In: Integrated Circuit Design and System Design, Symposium on. IEEE Computer Society, 2003. p. 125-125.

STAUB, Thomas; GANTENBEIN, Reto; BRAUN, Torsten. **VirtualMesh: an emulation framework for wireless mesh networks in OMNeT++**. In: Proceedings of the 2nd

International Conference on Simulation Tools and Techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009. p. 64.

STEINBACH, Till et al. **An extension of the OMNeT++ INET framework for simulating real-time ethernet with high accuracy**. In: Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011. p. 375-382.

STRAßBURGER, Steffen et al. **Internet-based Simulation using off-the-shelf Simulation Tools and HLA**. In: Proceedings of the 30th conference on Winter simulation. IEEE Computer Society Press, 1998. p. 1669-1676.

TANG, F., PARKER, L. **A Complete Methodology for Generating Multi-Robot Task Solutions using ASyMTRe-D and Market-Based Task Allocation**. Proc. of IEEE International Conference on Robotics and Automation, Rome, Italy, 2007.

TANG, F., PARKER, L. E. **ASyMTRe: Automated Synthesis of Multi-Robot Task Solutions through *Software* Reconfiguration**. In: Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. IEEE, 2005. p. 1501-1508.

TIMMONS, Nicholas F.; SCANLON, William G. **Analysis of the performance of IEEE 802.15. 4 for medical sensor body area networking**. In: Sensor and ad hoc communications and networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on. IEEE, 2004. p. 16-24.

VARGA, András et al. **The OMNeT++ discrete event simulation system**. In: Proceedings of the European simulation multiconference (ESM'2001). sn, 2001. p. 65.

VERHOEF, M. et al. **Co-simulation of distributed embedded real-time control systems**. In: Integrated Formal Methods. Springer Berlin Heidelberg, 2007. p. 639-658.

Vig, L., Adams, J. A.: **Issues in Multi-Robot Coalition Formation.** In: Proceedings of Multi-Robot Systems. From Swarms to Intelligent Automata. Volume 3. (2005) pp. 15-26.

VIG, L.; ADAMS, J. A. **Market-Based Multi-Robot Coalition Formation**. In: Distributed Autonomous Robotic Systems 7. Springer Japan, 2006. p. 227-236.

WANG, J.; GU, Y.; LI, X.. **Multi-Robot Task Allocation Based on Ant Colony Algorithm**. Journal of Computers, v. 07, n. 09, p. 2160-2167, 2012.

WOLF W. **Cyber-physical Systems**. IEEE Computer, 2009, 42(3): 88-89

YE, Wei et al. **Evaluating control strategies for wireless-networked robots using an integrated robot and network simulation**. In: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on. IEEE, 2001. p. 2941-2947.

ZHANG, Z.et al. **Co-Simulation Framework for Design Of Time-Triggered Cyber Physical Systems**. In: Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems. ACM, 2013. p. 119-128.

ZHENG, M.J. LEE, **A comprehensive performance study of IEEE 802.15.4, in: Sensor Network Operations**. IEEE Press, Wiley Interscience, New York, 2006, pp. 218–237, Chapter 4.

APÊNDICES

APÊNDICE A – DADOS COLETADOS DURANTE A VALIDAÇÃO DA ARQUITETURA

Tempo de Simulação	Perda de Pacote (%)	Tempo de Execução	Dados do Federado		Mensagens – Robôs				Mensagens – Roteador		
			Bytes Enviados	Bytes Recebidos	ID	Enviadas	Recebidas	Reenviadas	Enviadas	Recebidas	Perdidas
50	1	56782	20762	40612	1	46	23	0	98	49	0
					2	52	26	0			
50	2	57982	20762	40771	1	46	23	0	98	49	1
					2	52	26	1			
50	3	56900	21140	40659	1	46	23	0	100	50	0
					2	54	27	0			
50	4	62900	21140	40305	1	46	23	1	100	50	5
					2	54	27	4			
50	5	56782	20762	40612	1	46	23	0	98	49	0
					2	52	26	0			
100	1	114000	41640	77994	1	104	52	0	200	100	1
					2	96	48	1			
100	2	116400	41640	78216	1	104	52	2	200	100	3
					2	96	48	1			
100	3	117600	41640	78718	1	104	52	2	200	100	4
					2	96	48	2			
100	4	116400	41640	78188	1	104	52	1	200	100	3
					2	96	48	2			
100	5	116400	41640	78217	1	104	52	1	200	100	3
					2	96	48	2			
200	1	228200	82640	153590	1	188	94	2	400	2	3
					2	212	106	1			
200	2	229282	82262	152860	1	188	94	3	398	199	4
					2	210	105	1			
200	3	230600	82460	153683	1	188	94	1	400	200	5
					2	212	106	4			
200	4	232882	82262	153363	1	188	94	1	398	199	7
					2	210	105	6			
200	5	238882	82262	152864	1	188	94	7	398	199	12
					2	210	105	5			
400	1	462482	164262	304883	1	422	211	9	798	399	12
					2	376	188	3			
400	2	458882	164262	304417	1	422	211	4	798	399	9
					2	376	188	5			
400	3	465000	164640	304558	1	422	211	7	800	400	14

					2	378	189	7			
400	4	460200	164640	304888	1	422	211	3	800	400	10
					2	378	189	7			
400	5	473400	164640	304593	1	422	211	10	800	400	21
					2	378	189	11			

APÊNDICE B – DADOS COLETADOS DOS EXPERIMENTOS DO TIPO 02

Execução	Tempo Simulado	Taxa de Perda de Pacote (%)	Pacotes Perdidos	Posição Final (X; Y)	
1	300	1	4	34,84	26,85
2			5	34,71	26,77
3			3	34,99	26,9
4			4	34,84	26,85
5			3	34,96	26,94
6		2	2	35,09	27,02
7			7	34,59	26,68
8			6	34,59	26,68
9			5	34,71	26,77
10			11	33,97	26,26
11		3	10	34,1	26,34
12			4	34,84	26,85
13			4	34,84	26,85
14			11	33,97	26,3
15			15	33,33	25,87
16		4	13	33,7	26,13
17			11	33,97	26,26
18			6	34,71	26,77
19			19	32,96	25,62
20			12	33,85	26,17
21		5	16	33,33	25,87
22			15	33,48	25,92
23			11	33,95	26,3
24			21	32,71	25,45
25			16	33,33	25,87
26	600	1	9	69,94	53,88
27			4	70,71	53,55
28			4	70,71	53,55
29			6	70,46	53,38
30			7	70,34	53,3
31		2	7	70,34	53,3
32			17	69,08	52,49
33			12	69,72	52,88
34			10	69,84	52,97
35			14	69,45	52,74
36		3	19	68,8	52,36
37			18	68,96	52,41
38			19	68,66	52,29
39			25	68,09	51,82

40			12	69,72	52,88
41		4	19	68,83	52,33
42			27	67,82	51,69
43			18	68,96	52,41
44			19	68,83	52,33
45			20	68,68	52,27
46		5	31	67,32	51,35
47			31	67,32	51,35
48			21	68,56	52,19
49			38	66,41	50,82
50			15	69,35	52,63
51	1200	1	13	136,02	112,85
52			11	136,27	113,02
53			8	136,7	113,2
54			16	135,66	112,59
55			5	136,94	113,36
56		2	35	133,25	111,06
57			32	133,63	111,31
58			27	134,24	111,74
59			24	134,67	111,92
60			31	133,78	111,36
61		3	32	133,6	111,35
62			48	131,63	110
63			43	132,24	110,42
64			38	132,89	110,8
65			35	133,23	111,1
66		4	49	131,5	109,92
67			50	131,38	109,83
68			51	131,2	109,82
69			53	130,98	109,61
70			53	130,98	109,61
71		5	82	127,24	107,37
72			67	129,17	108,54
73			53	130,95	109,65
74			60	130,06	109,09
75			56	130,64	109,32