UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE GRADUAÇÃO E EDUCAÇÃO PROFISSIONAL
CURSO SUPERIOR DE TECNOLOGIA EM AUTOMAÇÃO INDUSTRIAL

GIANCRISTIAN FRANZOSO

ANÁLISE DA VIABILIDADE DE UTILIZAÇÃO DO *NETWORK SIMULATOR* VERSÃO 2 (*NS*-2) PARA SIMULAÇÃO DE TOPOLOGIAS BÁSICAS DE REDES INDUSTRIAIS.

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
06/2016

GIANCRISTIAN FRANZOSO

ANÁLISE DA VIABILIDADE DE UTILIZAÇÃO DO *NETWORK SIMULATOR* VERSÃO 2 (*NS*-2) PARA SIMULAÇÃO DE TOPOLOGIAS BÁSICAS DE REDES INDUSTRIAIS.

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Automação Industrial, da Universidade Tecnológica Federal do Paraná - UTFPR.

Orientador: Prof. Me. Clovis Ronaldo da Costa Bento

CORNÉLIO PROCÓPIO
06/2016

GIANCRISTIAN FRANZOSO

ANÁLISE DA VIABILIDADE DE UTILIZAÇÃO DO *NETWORK SIMULATOR* VERSÃO 2 (*NS*-2) PARA SIMULAÇÃO DE TOPOLOGIAS BÁSICAS DE REDES INDUSTRIAIS

Trabalho de conclusão de curso apresentado às 13:30 h do dia 15 de junho de 2016 como requisito parcial para a obtenção do título de Tecnólogo em Automação Industrial da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado**.**

COMISSÃO EXAMINADORA

Orientador: Prof. Me. Clóvis Ronaldo da Costa Bento
Universidade Tecnológica Federal do Paraná – UTFPR
/Câmpus Cornélio Procópio

Membro: Prof. Me. Ângelo Feracin Neto
Universidade Tecnológica Federal do Paraná – UTFPR
/Câmpus Cornélio Procópio

Membro: Prof. Dr. Rafael Abrantes Penchel
Universidade Tecnológica Federal do Paraná – UTFPR
/Câmpus Cornélio Procópio

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

Dedico este trabalho, todo empenho e circunstância a pessoa majestosa do Espirito Santo, amigo fiel, verdadeiro sempre presente, melhor companheiro para aqueles que o amam e o melhor amigo daqueles que não O viram, propiciador de todas as coisas, libertador das adversidades e realizador de todo sonho, detentor de toda sabedoria, a Ele a glória, a força, a honra e o poder. “O temor do Senhor é o princípio da sabedoria, e o conhecimento do Santo é prudência.” Provérbios 9: 10.

AGRADECIMENTOS

Agradeço primeiramente a Deus Pai, Filho e Espírito Santo pela oportunidade de, com suas forças, ferramentas e sabedoria, conduzir-me a esta conquista em minha vida.

Agradeço especialmente a meu pai Giancarlo Franzoso (in memoriam), que além de me gerar, me exortava, mesmo sendo eu muito pequeno, da importância e de seu desejo em me ver formado e com uma profissão. Apesar da extrema tristeza pela sua ausência nesta conquista, dedico à emoção deste feito a educação, amor, carinho, atenção, afeto, reciprocidade e respeito que me foi transmitido.

Agradeço a minha mãe, Diná Pereira dos Santos, que sempre me alentou, fazendo muitas vezes, mais que o papel de uma excelente mãe e pai.

Ao Senhor José Maciel de Camargo, homem do qual me orgulho de ter conhecido, exemplo de ser humano, de honestidade, de responsabilidade, de ética, de respeito e temor a Deus e que em todos esses anos me ajudou de todas as maneiras possíveis com sua preocupação exacerbada, revelando um amor incondicional de pai para filho, mesmo não tendo sequer algum vínculo sanguíneo.

À minha irmã, Viviane Pereira dos Santos, minha cúmplice de todos os momentos, minha base, exemplo de vida, esforço e dedicação, que sempre me levanta, me ajuda, me exorta e me abraça. Genitora das minhas duas maiores riquezas, Alisson Henrique e Lívia Maria. Ao meu cunhado, Alexandre dos Santos Silva, pelo papel desempenhado em nosso meio em um dos momentos mais difíceis da minha vida. Amo todos vocês.

Ao meu orientador, Clóvis Ronaldo da Costa Bento, ser humano de caráter, conhecimento, paciência e dedicação excepcional, que desprendeu um tempo extremamente precioso, sempre disposto e presente no desenvolvimento deste trabalho e de outras disciplinas. Agradeço também aos membros da banca de avaliação pela confiança e o respeito em mim depositado. A todos os professores, servidores e coordenadores, meu muito obrigado à instituição UTFPR-CP.

RESUMO

O presente trabalho se propõe a testar a viabilidade do *Network Simulator (NS-2)* como ferramenta para a simulação de redes industriais. O *NS*-2 é um simulador de redes de código fonte aberto amplamente utilizado no meio acadêmico, no entanto, muito pouco é encontrado na literatura a respeito de seu uso na simulação de redes Industriais, especificamente a Profibus.

Pesquisas foram realizadas para conhecer as funcionalidades básicas do *NS*-2 para simulação de redes convencionais que poderiam ser empregadas na simulação de redes industriais sem a necessidade de criação ou alteração dos códigos-fonte escritos em C++. Foram realizadas simulações utilizando-se topologias simples para observar o comportamento da ferramenta e comparar os resultados com valores esperados.

Para efetuar as simulações, foram desenvolvidos *scripts* em linguagem TCL, que tornaram possível a construção das topologias e configuração dos parâmetros da rede, assim como, a representação do protocolo de comunicação, do tráfego, tamanho dos pacotes de dados e comportamento das filas.

A visualização e comparação dos resultados são representadas em tabelas e gráficos.

Palavras-chave**:** *Network Simulator*, Redes Profibus*,* Simulação, Redes Industriais.

ABSTRACT

The present study proposes to test the feasibility of the Network Simulator (*NS*-2) as a tool for the simulation of industrial networks. The *NS*-2 is an open source network simulator widely used in academia, however, very little is found in the literature about its use in simulation Industrial networks, specifically the Profibus.

Searches were performed to know the basic features of the *NS*-2 to simulate conventional networks that could be used in the simulation of industrial networks without the need to create or change the source code written in C++. Simulations using simple topologies were achieved to observe the behavior of the tool and compare the results to expected values.

To make the simulations were developed scripts in TCL language, which made possible the construction of topologies and configuration of network parameters, as well as the representation of the communication protocol, traffic, size of data packets and behavior of the queues.

The viewing and comparison of the results are shown on tables and graphs.

Keywords: Network Simulator, Profibus, Simulation, Fieldbus Networks.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

ACK – *Acknowledge* (Reconhecimento)
ASI – *Actuador Sensor Interface* (Interface Atuador/ Sensor)
AT&T – *American Telephone and Telegraph*
CBR – *Constant Bit Rate* (Taxa Constante de Bits)
CEFET-PR - Centro Federal de Educação Tecnológica do Paraná
CLP – Controlador Lógico Programável
DAINF – Departamento Acadêmico de informática
DP – Decentralized Peripheral (Periféricos Descentralizados)
FMS – *Fieldbus Message Specification*
FTP – *File Transfer Protocol*
GCC – *GNU Compiler Collection* (Coleção de Compiladores GNU)
GPLv2 – *General Public License Version* 2 (Licensa Pública Geral Versão 2)
GT-ITM – *Georgia Tech Internetwork Topology Models* (Modelos de Topologias de Redes Geogia Tech)
GUI – *Grafical User Interface* (Interface Gráfica do Usuário)
IEC – *International Electrotechnology Commission* (Comissão Eletrotécnica Internacional
KDE – *K Desktop Environment* (Ambiente de Área de Trabalho K)
MS - Microsoft
NAM – *Network Animator* (Animador de Redes)
NS – 2 – *Network Simulator 2* (Simulador de Redes Versão 2)
OSI – *Open Systems Interconnection* (Interconexão de Sistemas Abertos)
OTCL – *Oriented-Object Tool Comand Language* (linguagem Ferramental de Comando á Objeto Orientado)
PA – Process Automation (Automação de Processo)
PROFIBUS – *Process Field Bus*
PROFINET - *Process Field Network*
RTT – *Round Trip Time* (Tempo de retorno/volta de mensagens)
SGB – *Stanford Graph Base* (Base de Gráficos Stanford)
TCLCL – *Tool Command Language With Classes* (Linguagem Ferramental de Comando com Classes)
TCP – *Transfer Control Protocol* (Protocolo de Controle de Transferência)
UDP – *User Datagram Protocol* (Protocolo de Datagrama do Usuário)

SUMÁRIO

1 INTRODUÇÃO

No universo contemporâneo, tornou-se requisito indispensável à sobrevivência de uma empresa, indiferente ao seu porte, a melhoria contínua de seus processos e produtos (NETO, 2001). Sendo assim, grande parte do lucro dessas empresas acaba sendo oriundo das estratégias de produção aprimoradas, a exemplo da automação industrial.

Nesse contexto, existem as redes industriais, responsáveis pela comunicação entre servidores, dispositivos de campo, processos e outros dispositivos, locais ou remotos, destinados a executar alguma função específica em uma planta industrial, e para total êxito em um processo produtivo, são necessários sincronismos perfeitos entre todos os equipamentos que compõem o sistema em si.

Porém, prever as circunstâncias ideais e exatas de trabalho de uma rede nem sempre é possível. Muitos problemas podem surgir de modo inesperado, após a implantação do sistema, causando perda de tempo e recursos. Desta forma, a simulação de uma rede industrial, com seus componentes de *hardware*, protocolos e aplicações, muito próximos de um modelo real, pode facilitar o projeto e a instalação de uma planta industrial automatizada.

De um modo geral, é tecnologicamente mais prático, utilizar *softwares* de simulações específicos para cada aplicação, o que não é complicado com o cenário tecnológico em que convivemos no mundo contemporâneo, mas no tocante em relação á acessibilidade comum aos mesmos, isto já é outro assunto.

Como ferramenta de simulação de redes genéricas, dentre tantas há o *NS-2 (Network Simulator version 2)*. É um *software* livre que, apesar de o ser bastante utilizado em diversos protocolos de comunicação, pouco se sabe quanto à utilização em protocolos de comunicações industriais (COUTINHO,2003)

A base de funcionamento do *NS-2* está na criação de *scripts* que representam topologias de redes, rodam e simulam o comportamento da comunicação de pacotes de dados entre os dispositivos componentes do sistema.

Sendo assim, o intuito foi pesquisar a viabilidade da criação de *scripts* para os protocolos de comunicações industriais de padrões abertos, com uma topologia básica da rede mais utilizada, no caso a rede PROFIBUS *(Process Field Bus),* em topologias mestre-escravo (ASSOCIAÇÃO PROFIBUS BRASIL, 2003).

2 PROBLEMA

O projeto e concepção de uma rede industrial, de acordo com sua dimensão, não é uma tarefa simples. A decisão sobre a topologia, equipamentos e aplicações a serem utilizados requer conhecimento e experiência na área, a fim de reduzir o risco de prejuízo em tempo e dinheiro.

Uma vez finalizada, uma planta industrial não permite fácil reestruturação ou qualquer alteração no projeto original sem algum prejuízo não previsto para a organização.

Desta maneira, fazer uma simulação do comportamento da rede industrial, mais próxima possível da realidade encontrada no campo, pode resultar em economia de tempo e capital, antecedendo situações prejudiciais ao planejamento e execução dos processos de comunicação de uma planta industrial com o máximo de desempenho e economia.

Porém, há necessidade de se conhecer o comportamento padrão de uma rede de comunicação para poder utilizá-la em um ambiente industrial e obter um desempenho esperado, ou seja, seria possível utilizar o *NS*-2 para facilitar o processo de desenvolvimento de uma rede industrial?

3 JUSTIFICATIVA

Atualmente no mercado, existem diversos *softwares* comerciais que realizam a simulação de redes industriais. Apesar de serem extremamente eficientes, sabe-se que, por se tratarem de "*closed source*", possui um custo muito elevado de aquisição e manutenção.

O programa utilizado foi o *NS*-2, que é gratuito, na categoria de *software* livre, disponível publicamente sob a licença GNU GPLv2*,* para pesquisa e desenvolvimento. É conhecido e amplamente utilizado no meio acadêmico e possui vasta documentação de suporte e exemplos de aplicação.

O *NS*-3 também foi considerado neste trabalho, mas por ser um software mais recente, ainda não possui farta documentação e exemplos de aplicação.

A simulação tem um importante papel no projeto de qualquer sistema. De modo que, pode-se citar algumas vantagens da simulação:

- Definição da topologia, componentes e aplicações da rede antes da concepção física do sistema;
- Maximização do desempenho;
- Redução do tempo do projeto;
- Redução de custos;
- Antecipação de problemas de projeto;
- Treinamento do pessoal;
- Previsão de instalações futuras.

4 OBJETIVOS

Para o cumprimento do objetivo deste trabalho, as atividades foram guiadas de acordo com um objetivo geral e os respectivos objetivos específicos.

4.1 OBJETIVO GERAL

Testar a viabilidade do uso do *NS-2* como ferramenta de *software* aberto para a simulação de redes industriais.

4.2 OBJETIVOS ESPECÍFICOS

São os objetivos específicos, na criação e desenvolvimento do *script* do programa:

- Conhecer o *NS-2* como ferramenta de simulação de redes em geral;
- Desenvolver *scripts* em TCL para simulação de redes industriais;
- Simular topologias básicas mais utilizadas em automação industrial;
- Estudo e análise dos protocolos de comunicação de redes industriais;
- Revisão dos conceitos de automação industrial e redes industriais;
- Simular redes com o protocolo PROFIBUS.

5 MÉTODO DE PESQUISA

O método necessita de uma estratégia de pesquisa, um cronograma e execução. Do dicionário “Pesquisa”: Ação ou efeito de pesquisar, indagação, inquirição, investigação (FERREIRA, 2002), com intenção de agregar ao estudo um conceito científico, ou seja, conjunto metódico de conhecimentos obtidos mediante observação e experiência, saber e habilidades que se adquire para o bom desempenho de certas atividades, informação conhecimento. (FERREIRA, 2002).

Neste trabalho foi utilizado o tipo de pesquisa aplicada, cuja estruturação:

- Fundamentação teórica;
  - Pesquisa bibliográfica;
- Metodologia de pesquisa;
  - Objeto: programa *NS-2* para simulação de protocolos de redes Industriais;
  - Técnicas: simulação por *software*;
  - Recursos: computador, *Linux Opensuse*;
  - Análise de resultados: comparação;
- Análise e discussão dos dados;

5.1 PESQUISA BIBLIOGRÁFICA

Trata-se da busca incessante sobre a delimitação do assunto ou tema, do material já existente, do que já foi estudado e/ou publicado, e a partir deste material, geralmente livros, revistas reconhecidas, publicações, artigos, periódicos e até mesmo com material da *internet* de sites confiáveis permitidos sem manipulação externa. Sendo utilizada para tal, pesquisa em livros relacionados à Automação Industrial e protocolos de comunicação em redes Industriais. Também pesquisas relativas ao sistema operacional *Opensuse Leap 42.1,* programa *NS-2*.

Preferencialmente arquivos publicados por pessoal conceituado, bem como tutoriais e respectivos métodos práticos de aquisição de conhecimento necessário à elaboração do trabalho.

5.2 PESQUISA APLICADA

De acordo com o livro “Metodologia Científica” (Barros; Lehfeld 2000, p.78), a pesquisa aplicada tem como motivação a necessidade de produzir conhecimento para aplicação de seus resultados, com o objetivo de “contribuir para fins práticos, visando à solução mais ou menos imediata do problema encontrado na realidade”.

A pesquisa aplicada utiliza a fundamentação teórica como referência para a análise de dados obtidos, neste caso, pelos experimentos de laboratório baseados na simulação de protocolos pelo *NS-2*.

6 REVISÃO BIBLIOGRÁFICA

6.1 CONCEITOS SOBRE SIMULAÇÃO

Partindo do fato que o *NS-2* trata-se de um simulador de comportamento de redes de comunicação, pode-se conceituar simulação como:

O estudo e aprimoramento de técnicas de concepção utilizando os experimentos, ou seja, o método de observar ou fazer alguma coisa sobre determinada “condição” (FERREIRA, 2002), vem sendo estudada com afinco há décadas (ARISTÓTELES, 1979).

A conclusão de um experimento, quer seja o mesmo bem-sucedido ou não, depende única e exclusivamente da forma em que o conhecimento para a abordagem do mesmo será aplicado, e tanto o método para realização quanto o método para verificação e correção dos resultados são essenciais à realização do projeto.

Será relatada no tópico a seguir, a metodologia de ensaio básica, porém, necessária para esta pesquisa.

6.2 MÉTODO DE SIMULAÇÃO POR *SOFTWARE*

Sem dúvida, um dos mais variados fins, dos quais o computador é mais utilizado atualmente, é a simulação ou ensaio. Por ser uma excelente ferramenta de interface gráfica e operacional, permite simular quase todo tipo de circunstância, seja ela física ou não, inclusive nas mais diversas áreas científicas. Um grande avanço nesse termo talvez seja a simulação através de impressão 3D.

Mas isso só foi possível porque o *software* operacional (conjunto de linguagens de programação) inserida no *hardware* (parte física) propicia um método de interação homem-máquina através de outros *softwares* específicos, em um determinado ambiente simulado, ou virtual, sem riscos desnecessários, alcançando

resultados muito próximos dos reais, através dos cômputos e comandos designados nas linguagens de programação (JAIN, 1991).

A exemplo, no projeto, como já designado, será utilizado o programa *NS-2*, que através de sua linguagem irá simular o comportamento de uma rede industrial. Isso será possível mediante á todas as técnicas que serão empregadas na aplicação para uso com método de simulação por *software*.

6.3 SIMULADOR DE REDES *NS-2*

6.3.1 Introdução

O *NS-2* é um simulador de redes orientado a eventos desenvolvido na Universidade de *Berkeley* que simula variedades de redes IP. Ele compila os protocolos de rede, como TCP e UPD, o comportamento fonte de tráfego, como FTP, Telnet, Web, CBR e VBR, mecanismo de gestão de fila de roteadores, como *DropTail*, RED e CBQ, algoritmos de roteamento, como *Dijkstra*, e muito mais (GREIS, 2011).

Ele também compila *multicast* e alguns dos protocolos da camada MAC para simulações de LANs. O projeto *NS-2* faz parte do projeto VINT que desenvolve ferramentas para a exibição de resultados de simulação, análise e conversores que transformam topologias de rede geradas por simuladores bem conhecidos para formatos *NS-2*. Atualmente, o *NS-2* é escrito em C++ e OTcl (*Object-oriented tool comand language*), uma linguagem de *script* Tcl com extensões orientadas a objetos desenvolvida no MIT.

Ainda sobre o *NS-2*, O Otcl consiste basicamente na linguagem que interage com o usuário, fazendo também a conversão para a máquina em C++ para a melhor manipulação de *bytes* (*Otcl linkage*). Ressaltando a importância do escalonador de eventos, que organiza e aciona e trata os eventos de acordo com o tempo especificado (COUTINHO, 2003).

6.3.2 Funcionamento

O *NS-2* possui muitas características particulares quanto às simulações de redes, comportando protocolos *wireless* (sem fio), TCP, UDP, por exemplo. Para uma melhor compreensão, observa-se a figura 1, a seguir, sua estrutura básica e a interface para comunicação.

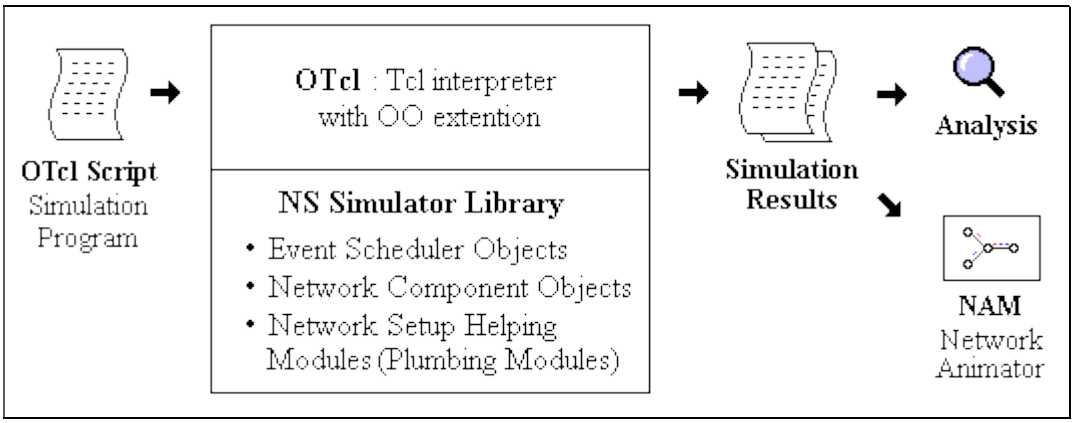

Figura 1 - Visão simplificada do *NS-2*.
Fonte: GREIS, 2011.

Como mostrado na figura 1, do ponto de vista de um usuário o *NS-2* é um interpretador de *scripts* TCL orientado a objetos que tem um escalonador de eventos de simulação, de componentes de rede, bibliotecas de objetos em C++, de configuração de rede (canalização) e bibliotecas de módulos (na verdade, os módulos de canalização são implementados como funções de membro do objeto simulador de base).

Em outras palavras, para usar o *NS-2* é preciso programar em Otcl, a linguagem do *script* da rede a ser simulada. Para configurar e executar uma simulação, o usuário deve escrever um *script* “Otcl” que inicia um agendador de eventos (*scheduler*), configura a topologia de rede usando os objetos de rede e as funções de “encanamento” na biblioteca e diz para as fontes de tráfego quando começar e parar de transmitir pacotes através do escalonador de eventos.

O termo "encanamento" é usado para uma configuração de rede, porque a criação de uma rede de caminhos de dados vem do encanamento possível entre os

objetos da rede, definindo o ponteiro "próximo" de um objeto para o endereço de um objeto apropriado.

Quando um usuário deseja fazer um novo objeto de rede, ele pode facilmente fazê-lo, seja escrevendo um novo objeto ou fazendo um objeto composto a partir da biblioteca de objetos, e sondar o caminho de dados através do objeto. Isto pode soar como trabalho complicado, mas o encanamento de módulos OTcl torna o trabalho muito fácil. O poder da *NS-2* vem deste encanamento.

A figura 2 mostra a arquitetura geral do *NS-2*. Nesta figura, um usuário comum (e não um desenvolvedor *NS-2*) pode utilizar a ferramenta a partir do TCL, no canto inferior esquerdo, para eventual concepção e execução de simulações em Tcl, usando os objetos do simulador na biblioteca OTcl.

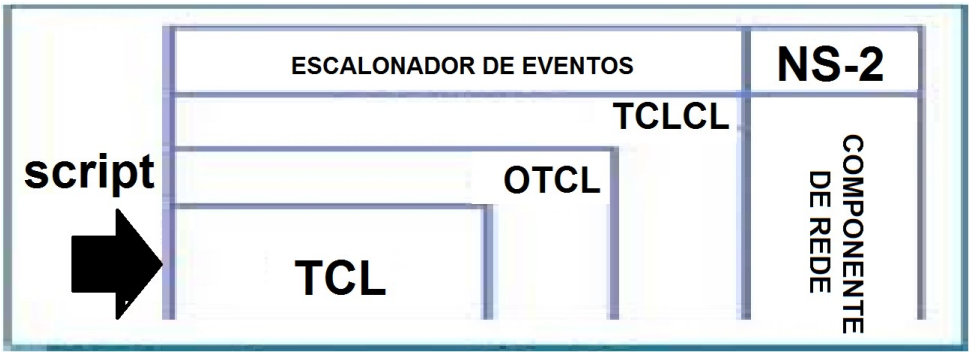

Figura 2 - Arquitetura do *NS-2*.
Fonte: COUTINHO, 2003.

Os escalonadores de eventos e a maior parte dos componentes de rede são implementadas em C++ e disponível para a linguagem Otcl, através de uma ligação ou "*Linkage" em OTcl* que é implementado utilizando a linguagem Tclcl. A junção de todos esses componentes cria o *NS-2*, que é um *software* "OO" (*Object Oriented*), estendido com um interpretador Tcl e bibliotecas para simulação de redes.

Outro componente importante, ao lado dos objetos de rede, é o escalonador de eventos. No *NS-2*, o escalonador de eventos mantém o controle do tempo de simulação e dispara todos os eventos na fila, programando para o tempo atual e invocando componentes de rede apropriados, que geralmente são os que emitiram os eventos, e ao deixá-los realizar a ação apropriada associada ao pacote demarcado pelo evento, os componentes de rede se comunicam uns com os outros,

no entanto, esta ação por parte do *NS-2*, não consome tempo a mais de simulação, ou seja, são contabilizados apenas os atrasos realísticos da rede.

Por exemplo, uma componente chave de rede que simula um *switch* com 20 microssegundos de atraso, na comutação emite um evento para um pacote a ser transferido para o programador como um evento de 20 microssegundos, porém mais tarde. O programador, 20 microssegundos depois, dispara para o componente de comutação, o qual, em seguida, passa o pacote para um componente de ligação de saída mais apropriado sem computar o tempo de atraso na geração da simulação.

Outro uso de um planejador de evento é o "*timer*". Por exemplo, um protocolo TCP precisa de um *timer* para manter o controle de um tempo de transmissão de pacotes para fora da retransmissão (transmissão de um pacote com o mesmo número de pacotes TCP, mas diferentes pacotes ID).

Temporizadores usam programadores de eventos de uma forma semelhante. A única diferença é que o temporizador mede um valor de tempo associado com um pacote e faz uma ação adequada relacionada com esse pacote depois de um determinado tempo que passa, e não simula um atraso.

O *NS-2* está escrito, não só em OTcl mas em C++ também. Por razões de eficiência, o *NS-2* separa a implementação do caminho de dados a partir das implementações de caminho de controle.

A fim de reduzir o tempo de processamento de pacotes e eventos (não tempo de simulação), o planejador de eventos e os objetos de componente de rede básicas, no caminho de dados, são escritos e compilados usando C++.

Esses objetos compilados são disponibilizados para o intérprete OTcl através de uma ligação OTcl que cria uma correspondência Otcl/objeto para cada um dos C++/objetos e faz as funções de controle e as variáveis configuráveis, especificados pelo C++, ato/objeto como membro funções e variáveis de membro do correspondente objeto / OTcl.

Desta forma, os controles de objetos C++ são dadas para OTcl. Também é possível adicionar funções de membro e variáveis para um objeto OTcl / C++ ligados.

Os objetos em C++ que não precisam ser controlados dentro de uma simulação ou internamente utilizado por outro objeto, não requer ligação a OTcl. Da mesma forma, um objeto (não no caminho de dados) pode ser totalmente implementado em OTcl.

A figura 3 mostra um exemplo da correspondência de objetos em C++ e OTcl. Observando-se a figura, nota-se que, para objetos C++ que têm uma ligação OTcl, que formam uma hierarquia, e que há uma outra hierarquia de objetos OTcl correspondente muito semelhante àquela do C++, ou seja, observa-se uma dualidade.

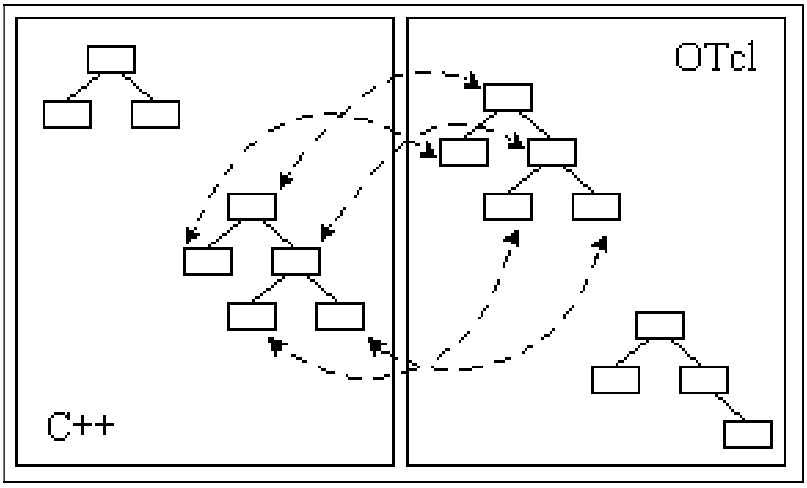

Figura 3 - C++ e OTcl: A Dualidade.
Fonte: GREIS, 2011.

Quando a simulação é terminada, o *NS-2* produz um ou mais arquivos de saída com base em dados de texto que contêm a simulação detalhada.

Os dados podem ser usados para análise de simulação, (exemplos de análise de resultado de duas simulações são apresentados em seções posteriores) ou como uma entrada para uma ferramenta de simulação gráfica animada chamada "*Network Animator" (NAM)* que é desenvolvida como parte do projeto VINT.

O *NAM* tem uma interface gráfica agradável para o usuário, semelhante ao de um "CD *player"* (reproduzir, avançar, retroceder, pausar e assim por diante), e também possui um controlador de velocidade de exibição. Além disso, pode gerar informações graficamente presentes, tais como: o rendimento e o número de pacotes e *"drops"* em cada ligação; para uma melhor visualização da topologia da rede e/ou ensaio, associando para a interpretação dos resultados.

A figura 4 apresenta a hierarquia parcial de classes no *NS-2*. Os componentes de rede são criados a partir destas classes.

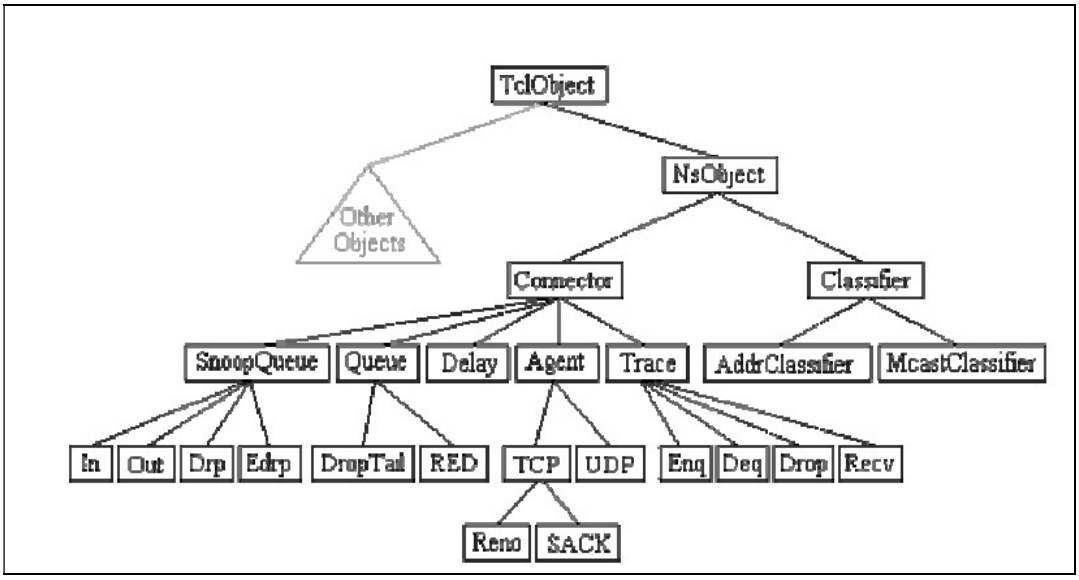

Figura 4 - Hierarquia parcial de classes no *NS-2*.
Fonte: A Ferramenta de simulação *NS-2*, 2015.

A raiz da hierarquia é a classe "*TclObject*", que é a superclasse de todos os objetos da biblioteca OTcl (programador, componentes de rede, temporizadores e outros objetos, incluindo os *NAM* relacionados).

Como uma classe ancestral de *TclObject*, a classe *NSObject* é a superclasse de todos os objetos de componente de rede básicas que lidam com pacotes, que podem compor objetos de rede composto, tal como nós e *links*.

Os componentes básicos de rede são divididos em duas subclasses, conector e classificador, com base no número dos possíveis caminhos de dados de saída.

Os objetos básicos de rede que têm apenas um caminho de dados de saída estão sob a classe "*Connector*", e mudar objetos que têm vários caminhos possíveis de dados de saída estão sob a classe classificadora. Um nó é um objeto composto por um objeto de entrada de nó e classificadores como mostrado na figura 5.

Existem dois tipos de nós no *NS-2*: *unicast* e *multicast*

Um nó *unicast,* que tem um classificador de endereço que faz o roteamento *unicast* e um classificador de chegada. Um nó *multicast*, que promove o encaminhamento de pacotes para diversos destinos, tem um classificador que classificará pacotes em *multicast* e *unicast.*

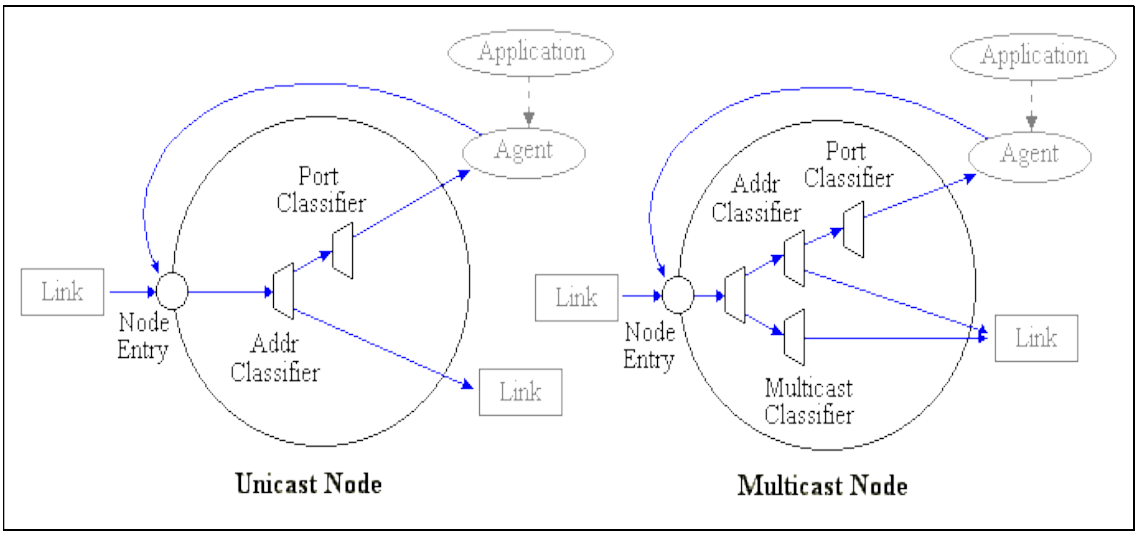

Figura 5 - Nós (unicast e multicast).
Fonte: Greis, 2011.

No *NS-2*, nós *unicast* são os nós padrão. Para criar *multicast*, os nós criados pelo usuário devem ser notificados explicitamente na entrada OTcl do roteiro, logo após a criação de um objeto escalonador, já que todos os nós que serão criados são nós de *multicast.* Depois de especificar o tipo de nó, o usuário também pode selecionar um protocolo de roteamento diferente de um padrão. A seguir, exemplos de declaração dos nós:

- *Unicast :*

  - `$ Type ns rtproto`
  - `Tipo: Static, Session, DV, custo, multi-caminho`

- *Multicast :*

  - `Ns $ multicast (logo após definir $ ns [novo Scheduler])`
  - `$ Type ns mrtproto`
  - `Tipo: CtrMcast, DM, ST, BST`

No *Network Simulator*, Guia básico para iniciantes (COUTINHO, 2003, p 9), Antes de efetivamente começar a programar em *NS-2*, é importante um planejamento de prancheta. Esse esboço do que se quer, em uma folha de papel, ajuda o usuário a ter uma visão macro das simulações pretendidas.

Como auxílio, são apresentados cinco passos essenciais ao desenvolvimento básico do *script*:

- Definir os nós
- Definir a ligação entre os nós (topologia)
- Definir o tráfego que será injetado na rede (tipo de aplicação)
- Definir parâmetros de pacotes, filas, atrasos, velocidades, entre outros.
- Escalonamento dos eventos
- Analisar os resultados

Para se escrever a simulação, qualquer editor baseado em texto pode ser utilizado, como o *emacs*, *"VI"* ou o *"Mcedit",* até os editores gráficos como o "*kedit"*, "*Gedit"* ou o *"Kate"*, de acordo com a interface gráfica instalada. Os arquivos devem ser gravados com a extensão ".tcl".

Um modelo é detalhado a seguir, no tópico exemplos de simulação. Toda a estrutura da simulação, desde os agentes de transporte, passando pelas aplicações e todo o mapeamento físico da topologia está fielmente representado. A noção de tempo no *NS-2* é obtida através de unidades de simulação que podem ser associadas, para efeitos didáticos, a segundos.

6.3.3 Exemplos de simulação

No planejamento mostrado adiante, a aplicação de vídeo (CBR) na simulação é iniciada no momento 0.1 e encerrada no momento 4.5, enquanto que a aplicação de transferência de arquivo (FTP) é iniciada no momento 1.0 e encerrada no momento 4.0. No intervalo entre os momentos 1.0 e 4.0 ambas as aplicações estão sendo transmitidas e é nesse intervalo onde, provavelmente, os congestionamentos irão surgir. No *NS-2* os agentes precisam de um repositório que receberá seus pacotes.

Para representar a construção de uma topologia, tem-se a seguir, um exemplo de *script* para a simulação de uma rede com dois nós e um enlace. Na linha a seguir, o comando *set* cria um objeto chamado ns, da classe simuladora. O objeto ns pode ser utilizado para acessar os métodos e os atributos desta classe.

```
set ns [new Simulator]
```

A próxima linha abre um arquivo chamado *out.nam* com permissão de escrita.

```
set nf [open out.nam w]
```

Utiliza-se a variável ns para configurar uma saída de resultado compatível com o aplicativo *NAM* para o arquivo *out.nam* (variável nf).

```
$ns namtrace-all $nf
```

Cria-se dois nós de rede, variáveis n0 e n1.

```
set n0 [$ns node]
set n1 [$ns node]
```

Cria-se e configura-se um enlace "*full duplex"* entre os nós n0 e n1. Todos os elementos criados possuem capacidade de roteamento e a taxa de transmissão configurada é de 1.5 Mbps com o atraso de propagação do enlace de 10 ms.

O algoritmo de gerência de fila é o *DropTail* (descarte de fim de fila). O tamanho *default* para a fila é de 30 pacotes. Existem outras opções de algoritmos para o gerenciamento de descartes de filas como o RED e o SFQ.

```
$ns duplex-link $n0 $n1 1.5Mb 10ms DropTail
```

A linha a seguir cria uma variável chamada udp0 do tipo *Agent/UDP*, que está programada para imitar o comportamento do protocolo UDP de nível 4.

```
set udp0 [new Agent/UDP]
```

Conecta-se o *udp0* a um nó transmissor, neste caso, nó 0:

```
$ns attach-agent $n0 $udp0
```

Na sequência cria-se uma aplicação do tipo CBR (*Constant Bit Rate*), ou de taxa constante, que gera pacotes separados por intervalos de tempo fixo, (no caso, com 0,01 segundo entre pacotes) e tamanho de pacote fixo (no caso, com 1000 *bytes)*. Para o exemplo, a taxa de geração de informação será dada por (8*1000)/0.01 = 800 kbps:

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1000
$cbr0 set interval_ 0.01
$cbr0 attach-agent $udp0
```

Para realizar o ciclo de transmissão-recepção, é necessário definir o elemento receptor, neste caso o agente *null* é criado e conectado ao nó n1:

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

Configura-se o tráfego gerado pelo agente *udp0* para o *null0* (do nó n0 para o nó n1):

```
$ns connect $udp0 $null0
```

Configura-se o escalonador de eventos, no exemplo, para indicar quais os momentos de início e fim da transmissão:

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Configura-se, ainda, o escalonador de eventos para acionar o procedimento "*finish"* no momento 5.0 de simulação.

```
$ns at 5.0 "finish"
```

Invoca-se o método *run* da classe *Simulator,* que irá executar as funções agendadas no escalonador de eventos.

```
$ns run
```

O procedimento *finish* está sendo chamado no tempo “5.0” (ms) para finalizar a simulação. Neste procedimento serão usadas as variáveis globais ns e nf (se não fosse feita a definição, teriam que ser variáveis locais deste procedimento). A linha com *flush-trace* descarrega os buffers para atualizar o arquivo. A execução do *NAM* é ativada, indicando que a entrada será o arquivo *out.nam* (que é á saída deste script). O “&” indica para o sistema operacional (ex.: Linux) que o programa *NAM* deve rodar em segundo plano (*background*).

```
proc finish {} {

global ns nf

$ns flush-trace

close $nf

exec nam out.nam &

exit 0
}
# eof
```

Na figura 6 está representado um exemplo que mostra uma topologia com quatro nós e operando com o FTP/TCP para transmissão de arquivos.

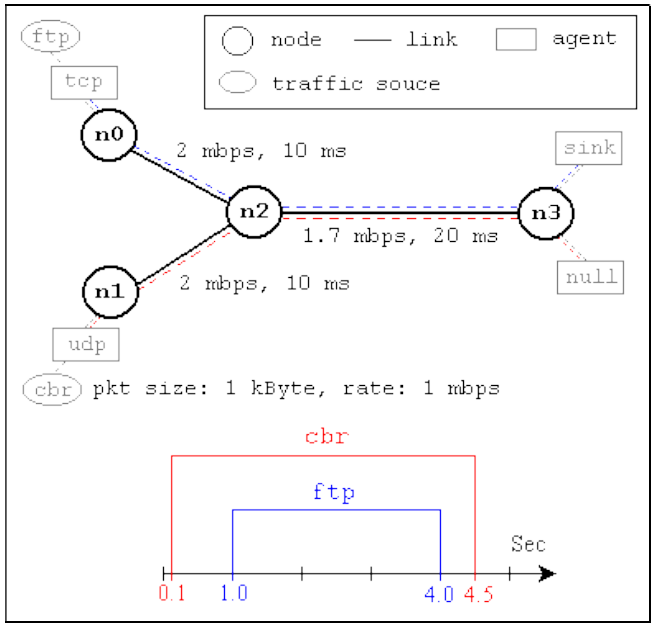

Figura 6 - Exemplo de topologia com o protocolo FTP/TCP.
Fonte: GREIS, 2011.

Esta rede consiste de quatro nós (n0, n1, n2, n3). Os enlaces duplex entre o n0 e n2 e n1 e n2 têm 2 Mbps de largura de banda e 10 ms de atraso. O enlace *duplex* entre n2 e n3 tem 1,7 Mbps de largura de banda e 20 ms de atraso. Cada nó usa uma fila *DropTail*, das quais o tamanho máximo é 10 bytes.

Um agente "tcp" é anexado ao n0, e é estabelecida uma ligação a um agente tcp "*sink*" ligados a n3. Como padrão, o tamanho máximo de um pacote que um agente "tcp" pode gerar é 1 kByte. O agente tcp "*sink*" gera e envia pacotes ACK ao remetente (agente "tcp") e libera os pacotes recebidos.

Um agente "UDP" que está ligado a n1, está ligado a um agente de "*null*" ligado a n3. Um agente "*null*" não responde, apenas libera os pacotes recebidos. O "ftp" e um gerador de tráfego "cbr" estão ligados a "TCP" e agentes "UDP", respectivamente, e os "cbr" está configurado para gerar pacotes de 1 kbyte, à taxa de 1 Mbps.

O "cbr" está definido para começar em 0.1 s e parar em 4,5 s, e "ftp" está definido para iniciar em 1,0 segundos e parar em 4,0 s. No quadro 1 está apresentado o código gerador desta topologia (arquivo *ns-simple.tcl*).

De um modo geral, um script *NS-2* começa com a tomada de uma instância para iniciar uma simulação. O conjunto *ns [new Simulator]*, na primeira linha, gera uma instância para o simulador de objeto *NS-2*, e o atribuí para as variáveis *ns*. Este conjunto faz o seguinte:

- Inicializa o formato do pacote (ignore isso por enquanto)
- Cria um escalonador (o calendário planejador)
- Seleciona o formato de endereço padrão (ignore isso por enquanto)

O objeto "*Simulator*" tem funções de membro e faz o seguinte:

- Cria objetos compostos, tais como nós e links (descritas mais adiante)
- Conecta objetos componentes de rede criados (ex. *attach-agent*)
- Parametriza os componentes conjuntos de rede (principalmente para objetos compostos)
- Cria conexões entre os agentes (ex. Fazer a conexão entre um "TCP" e *"sink*")
- Especifica opções de exibição *NAM*
- Entre outros.

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5


#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false


#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run
```

Quadro 1 - Exemplo de *script*/topologia com o protocolo FTP/TCP.
Fonte: GREIS, 2011.

Para criar uma simulação voltada a ensaios da rede Profibus, é preciso escrever um script *OTCL*, que será lido por um interpretador e gerará uma saída específica. Como exemplo, uma simulação da rede da figura 7 pode gerar uma representação gráfica semelhante à da figura 8.

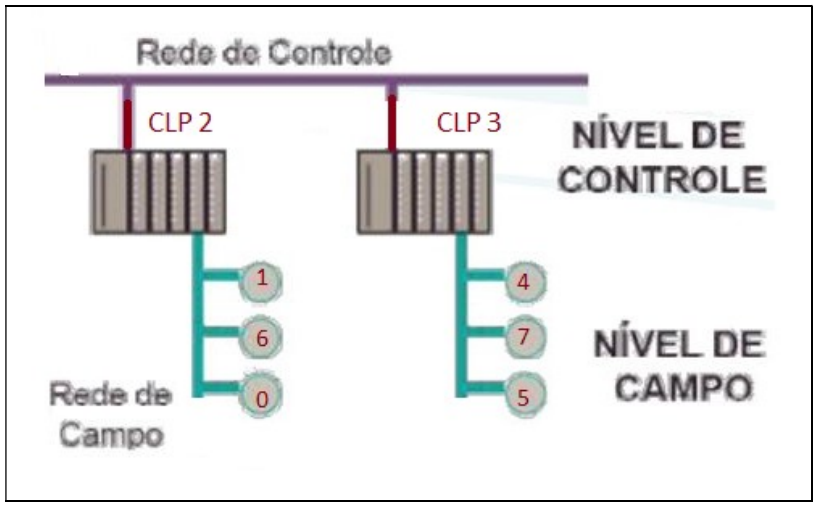

Figura 7 - Topologias básicas propostas para simulações.
Fonte: MECATRÔNICA ATUAL, 2007.

Executando a simulação representada graficamente através do *NAM* (*Network Animator*) verifica-se o tempo de envio e respostas de dados de uma forma mais didática. Na figura 8 tem-se uma representação de como o *NAM* apresenta uma simulação:

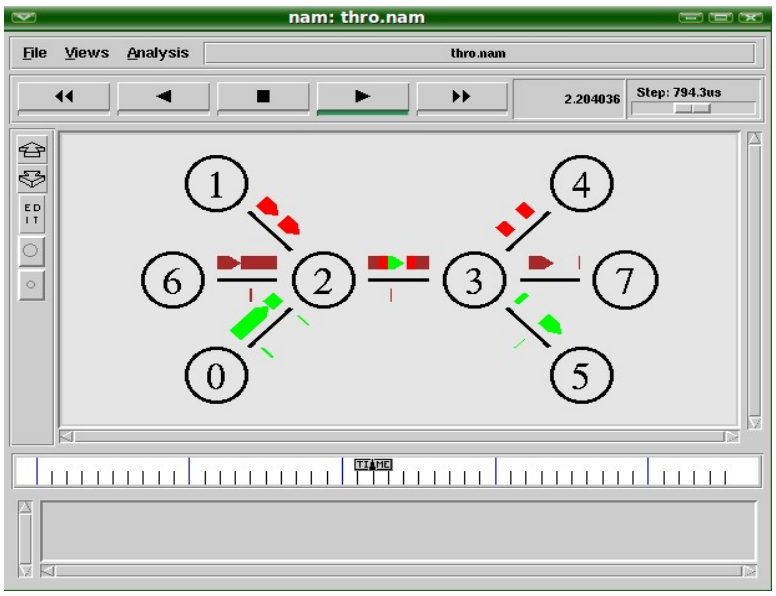

Figura 8 - Saída gerada pelo *NAM*.
Fonte: MECATRÔNICA ATUAL, 2007.

A ferramenta *NAM (Network Animator)* é muito importante para se obter uma ideia gráfica, baseada em animações, do andamento da simulação. Com o *NAM* pode-se observar a transmissão dos fluxos, a formação de filas, o descarte de pacotes, etc. A execução do *NAM* requer a instalação de uma interface gráfica no computador.

Ele possui uma interface gráfica de fácil manuseio e interpretação de dados, com recursos limitados, porém, com muita utilidade e praticidade, por ser uma ferramenta de animação das simulações de rede, para visualização e rastros de pacotes, suporta *layout* de topologia, animação nível de pacote, e vários "*Kits*" de inspeção de dados, além de fornecer informações como o número de pacotes perdidos (*droped*) em cada rede.

6.3.4 Analisando o *Trace* e gerando gráfico

O *NS-2* gera o *log* de todos os eventos ocorridos durante o processo de simulação em um arquivo de texto chamado *trace file*. O *trace* pode chegar a vários *megabytes* de tamanho (ex. 400 MB) e exige cuidados em sua análise. O formato padrão do arquivo *trace* é mostrado no quadro 2.

O primeiro campo diz respeito ao evento ocorrido. Pode ser uma entrada em fila (+), uma saída de fila (-), um descarte de pacote (d), um recebimento de pacote (r), etc. O campo seguinte é o momento da simulação onde o evento ocorreu. Os dois campos seguintes são referentes ao intervalo de nós onde o evento ocorreu. Os dois próximos campos dizem respeito ao tipo (TCP, UDP, entre outros) e tamanho do pacote (em *bytes*), respectivamente.

Uma série de *flags* relacionados a notificação antecipada de congestionamento vêm a seguir, mas normalmente não são utilizados. Depois têm-se a identificação do fluxo, os endereços do transmissor e do destinatário, o número de sequência do pacote e, finalmente, um número que identifica de forma única o pacote na rede.

evento	tempo	nó origem	destino	tipo pacote	tamanho pacote	flags	id fluxo	endereço fonte	endereço destino	num seq	id pacote

```
r : pacote recebido   (no destino)
+ : entrada de pacote (na fila)      endereço fonte   : nó.porta(3.0)
- : saída de pacote   (da fila)      endereço destino : nó.porta(0.0)
d : pacote descartado (da fila)

          r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
          + 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
          - 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
          r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
          + 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
          d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
          + 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
          - 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
```

Quadro 2 - Formato do arquivo de *trace*.
Fonte: COUTINHO, 2003.

Existem algumas ferramentas que podem ser obtidas gratuitamente na Internet para análise do *trace file*. Um exemplo de uma dessas é o *Tracegraph*.

Entretanto, para se usar o *Tracegraph*, existe a exigência do software *Matlab*, que não é livre, para versão *Windows* e de algumas bibliotecas do *Matlab* para a versão *Linux*, mas estas são disponibilizadas juntamente com o *Tracegraph*.

Outra possibilidade é a construção, já no código *Tcl*, de um *trace* personalizado. Um exemplo, (GREIS, 2011), grava duas colunas (tempo de simulação e vazão) em intervalos predefinidos em um arquivo de *trace* que, posteriormente, será usado como parâmetro de entrada no utilitário *XGraph*.

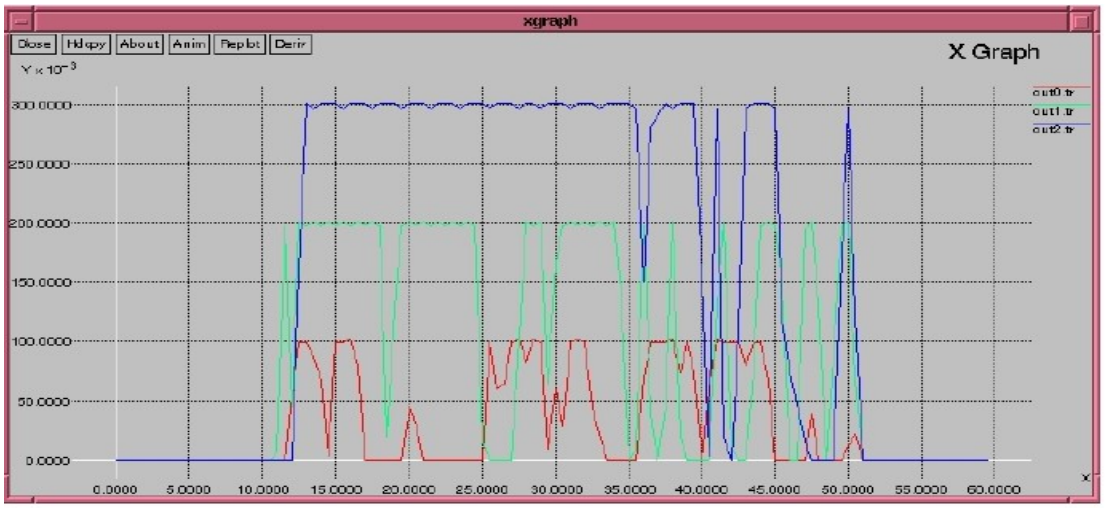

Figura 9 - Exemplo de gráfico da vazão.
Fonte: COUTINHO (2003).

6.4 PROFIBUS

O Profibus trata-se de um protocolo de comunicação para redes de comunicação industrial (LUGLI, 2010). É um protocolo aberto, criado em 1987, por uma associação de autoridades públicas alemãs. O principal objetivo era desenvolver uma tecnologia que possibilitasse que vários equipamentos dos mais distintos fabricantes pudessem desenvolver uma comunicação entre si, aliando custo de instalação e manutenção à segurança, flexibilidade e a garantia de normas e padrões EN 50170 e EN 50254, implementada a partir de janeiro de 2000, seguindo a IEC 61158–1 que vai até 61158–6, seguindo o padrão OSI, conforme comparação representada na figura 10. (ASSOCIAÇÃO PROFIBUS BRASIL, 2009, p.4), resultando em maior interoperabilidade.

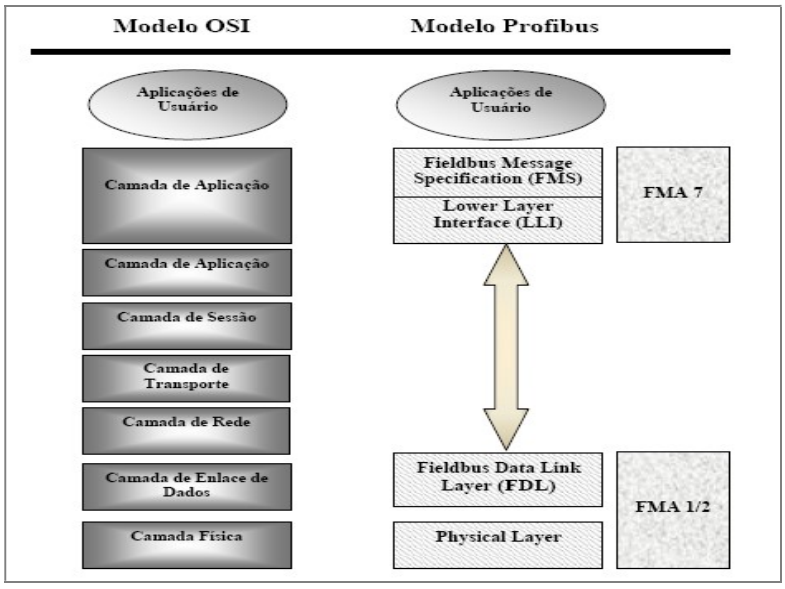

Figura 10 - Modelo Profibus / Modelo Osi.
Fonte: SMAR (2010).

A rede Profibus é dividida em quatro topologias:

1 – Profibus *PA (Process Automation)*
2 – Profibus *DP (Distributed Peripherals)*
3 – Profibus *FMS (Fieldbus Message Specification)*
4 – *Profinet (*Profibus *on Ethernet)*

Dados os fatos relativos à arquitetura e o respectivo custo de instalação, é comum a associação do Profibus com os diversos tipos de tecnologias de redes disponíveis no mercado. A escolha das tecnologias e topologias das redes dependerão das aplicações envolvidas, e o projeto deve visar o retorno financeiro e operacional. Embora havendo a diversidade de protocolos de redes, neste trabalho o foco foi a simulação da rede Profibus, nas modalidades DP e PA.

O formato geral dos *frames* utilizados para comunicação na rede Profibus está representado na figura 11.

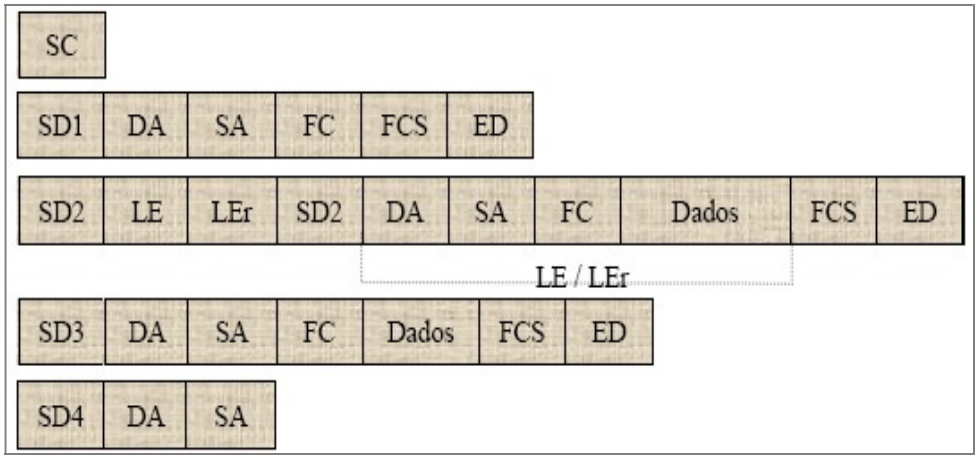

Figura 11 - Os cinco tipos de quadros da rede Profibus.
Fonte: SMAR (2010).

Onde:

- SDn: Tipo do quadro.
- DA: Endereço da estação de destino.
- SA: Endereço da estação que emitiu o quadro.
- FC: Quadro de controle (quadro control).
- FCS: *Checksum* dos campos DA, SA, FC e dados se houver.
- LE/Ler: Tamanho do campo DA, AS, FC e dados do uadro SD2.
- ED: Delimitador final (SMAR, 2010).

Por se tratar de uma rede que pode ser configurada com multimestres, ela ainda diferencia dois tipos de dispositivos no barramento, segundo a (ASSOCIAÇÃO PROFIBUS BRASIL, 2009, p 8) são:

1 – Dispositivos **mestre** ou estações ativas são dispositivos capazes de enviar mensagens a outros dispositivos sem que a mesma seja solicitada, mas, somente quando a mesma estiver na posse do *Token*, e claro, também recebem mensagens.

2 – Dispositivos **escravos** ou estações passivas não possuem direito ao acesso ao barramento, são dispositivos que somente podem confirmar o recebimento de mensagens do mestre ou responder á mensagem solicitada pelo mestre.

Como relatado em (ASSOCIAÇÃO PROFIBUS BRASIL, 2009, p. 7), a rede Profibus admite três tipos de meio físicos:

1 – **RS – 485** para aplicações gerais de automação de manufatura

2 – **IEC 1158 – 2** utilizado na automação de processos.

3 – Fibra óptica para uma maior imunidade ao ruído e maior distância.

A área de aplicação e meio físico é ilustrado na figura 12, onde *RS 485/FO* (Fibra Óptica) é utilizada na Profibus DP e IEC – 61158-2 na Profibus PA.

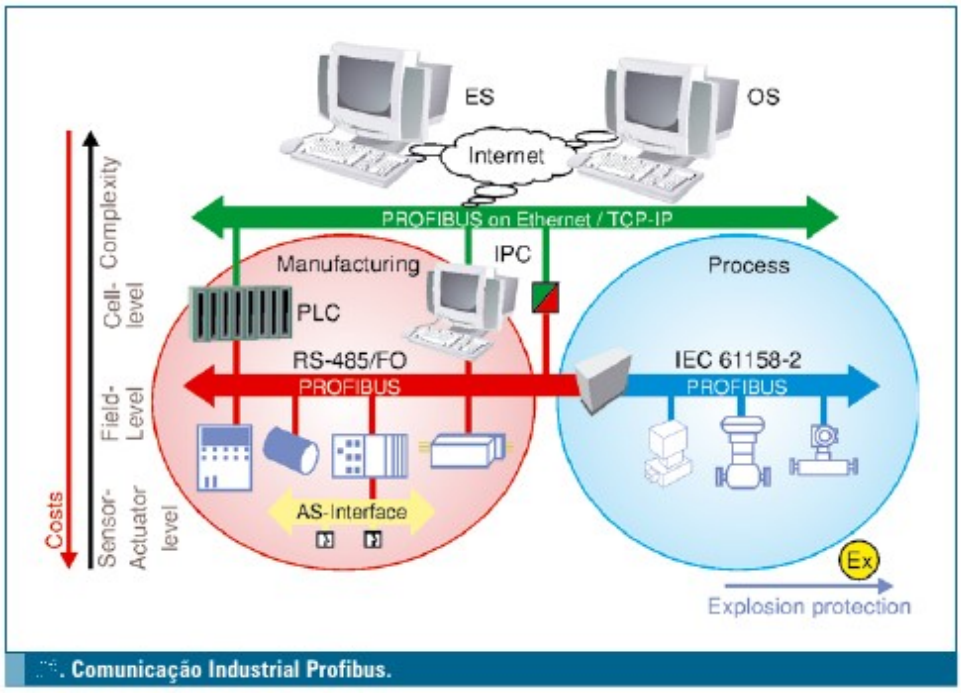

Figura 12 - Áreas e comunicação na Profibus.
Fonte: ASSOCIAÇÃO PROFIBUS BRASIL (2009, p.3).

Sabendo-se que um dos maiores benefícios da rede Profibus consiste no fato de que é necessário apenas um único meio físico para interligar vários dispositivos, o resultado é a diminuição dos custos e fácil expansão dos projetos, porém, os cuidados no projeto, instalação e manutenção são bem maiores, uma vez que simples erros de projeto, configuração e ensaio, podem gerar desde a comunicação intermitente com todos os dispositivos de da rede até a perda total da comunicação (MERLIN, 2011).

6.4.1 Profibus DP

O Profibus DP é um protocolo de comunicação industrial com a capacidade de transmissão de dados em alta velocidade no nível do dispositivo, pois o “CLP” (LUGLI; SANTOS, 2010), por exemplo, consegue se comunicar com outros dispositivos de controle variados (sensores, posicionadores, entre outros.).

Devido à alta taxa de velocidade de comunicação de maneira cíclica e as funções requeridas para estas comunicações são especificadas pelas funções básicas deste protocolo em acordo com a norma EM 50170.

Para realizar as funções cíclicas, as comunicações são solicitadas por equipamentos inteligentes (ex. CLPs) haja vista que os tais são capazes de possuir do tipo: diagnósticos, alarmes e configurações, entre outras, e fornecer poderosas funções para diagnóstico e monitoração. A figura 13 mostra um exemplo de rede Profibus DP.

A seguir serão enumeradas algumas das características do Profibus *DP*:

- Interface RS–485, cabo de par trançado de dois fios ou fibra óptica;
- Taxas de transmissão de 9,6 kbit/seg a 12 Mbps;
- Procedimento de passagem simbólico, mestre-mestre e mestre-escravo;
- Possibilidade de sistema monomestre ou multimestre;
- Máximo de 126 estações mestre-escravo com um único barramento;
- Comunicações ponto-a-ponto (usuário) ou *multicast* (controle);
- Transmissão de dados do usuário em mestre-escravo cíclica e transmissão mestre-mestre acíclica;

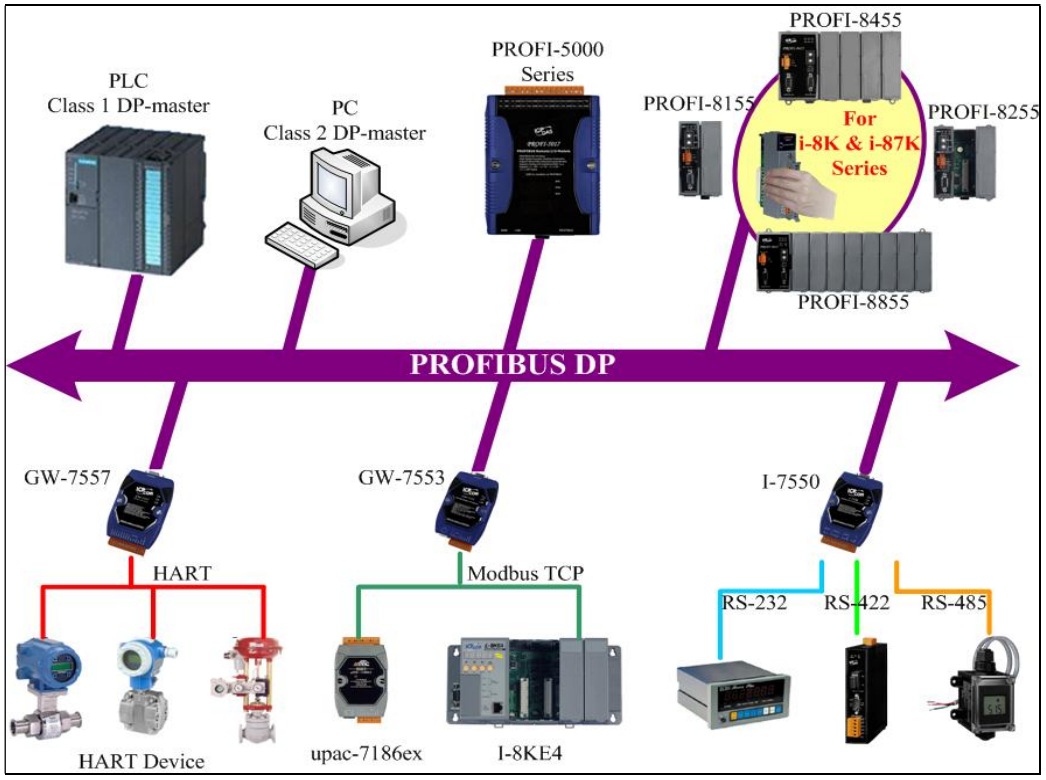

Figura 13 - Topologias DP.
Fonte: ASSOCIAÇÃO PROFIBUS BRASIL (2009, p.3).

Modos de operação do Profibus *DP*:

- *Operate*: Transmissão cíclica de dados de entrada e saída;
- *Clear:* As entradas são lidas e as saídas são colocadas em um *status* sem falhas
- *Stop*: Somente transmissões mestre-mestre são permitidas.

Sincronização:

- Os comandos de controle permitem a sincronização nas entradas e saídas;
- Modo síncrono: As saídas são sincronizadas;
- *Freeze mode*: As entradas são sincronizadas.

Uma representação do desempenho da rede Profibus DP considerando as velocidades de 500 kbps, 1.5 Mbps e 12 Mbps está mostrada no gráfico da figura 14:

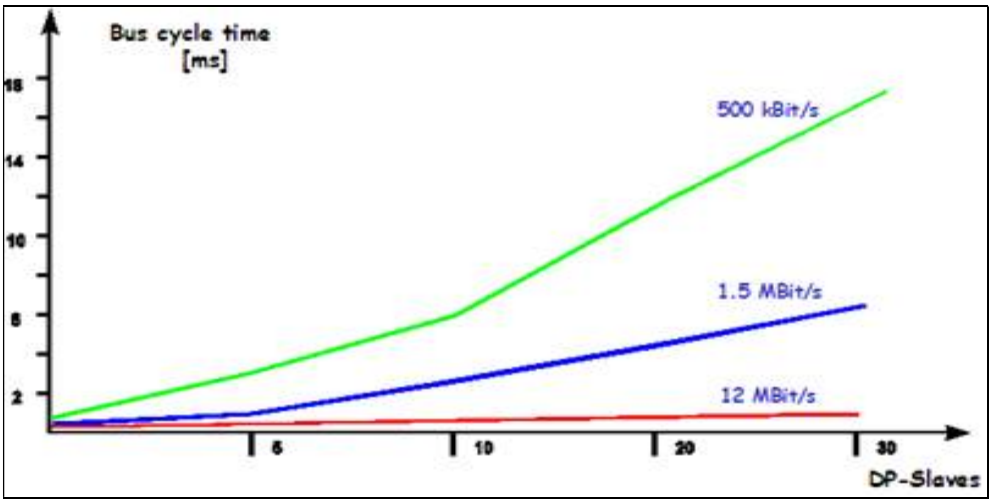

Figura 14 - Gráfico dos tempos de ciclos para diversas velocidades, Profibus DP.
Fonte: SMAR (2010)

6.4.2 Profibus PA

Segundo (ASSOCIAÇÃO PROFIBUS BRASIL, 2009, p. 19–24), os dispositivos instalados na rede PA, por serem escravos, dependem de um mestre para distribuir o controle onde, a rede, interliga 32 dispositivos a cada segmento, sem proteção intrínseca, com a proteção intrínseca até nove dispositivos, ou com proteção encapsulada até 27 dispositivos podendo ainda multiplicar esses números adicionando repetidores (no máximo quatro), criando novos segmentos. Os dispositivos podem ser conectados ou desconectados a rede em operação sem prejudicar seu funcionamento, mesmo que este dispositivo esteja em uma área classificada, o que facilita a manutenção.

Nesse contexto, a Profibus *PA*, utiliza como meio físico para a comunicação o padrão *IEC 61158-2*, recomendando um par trançado blindado, muito embora seja altamente recomendável o uso do cabo tipo A par trançado com blindagem de 0,8 mm^2 de área do condutor (AWG 18), resistência (*loop*) 44 ohms/km, impedância a 31,25 KHz de 100 ohms +/- 20%, atenuação a 39 KHz de 3 db/Km e carga capacitiva de 2 nF.

O meio físico mais utilizado na rede Profibus *PA* segue a norma *IEC 61158 – 2,* dispondo de um cabo 2x1 blindado, utiliza transmissão síncrona, taxa de comunicação (*baud rate*) de 31,25 Kbits/s, comprimento máximo de cabo, 1,9 Km por segmento e 30m por *spur* (*spur* – derivações do barramento principal) sem segurança intrínseca, ou com o uso da segurança intrínseca a distância máxima do cabo de um km por segmento e 30 m por *spur* (ASSOCIAÇÃO PROFIBUS BRASIL, 2009, p. 19).

Quanto as características construtivas, principalmente as especificações dos cabos, no que diz respeito a simulação da rede Profibus *PA*, é adotado como padrão o cabo tipo “A”, que tem as seguintes características técnicas:

- Zo@ 31,25 kHz = 100 Ohms +/- 20%
- Máxima Atenuação @ 39 kHz = 3 dB/ km
- Máximo desbalanceamento da capacitância para blindagem = 2 nF / km
- Máxima resistência DC por condutor = 22 Ohms / km
- Máxima mudança do atraso de propagação 7.8 – 39 KHz = 1.7 μs / km
- Área transversal do condutor 0.75 mm².

Ou seja, como os frames em uma rede precisam ir e vir pelo mesmo cabo, adota-se um décimo do atraso medido em um km de cabo multiplicado por 2 (ida e volta), haja vista que a distância máxima do cabo sem repetidores na Profibus PA é de 100 metros. Para as simulações deste trabalho e na rede em questão, será utilizado o tempo de 0,0034 ms conforme especificação.

Os princípios fundamentais do sistema de transmissão respeitam os seguintes aspectos, é admitida uma única fonte de alimentação a cada segmento, onde os dispositivos são alimentados ao mínimo com 9 Vdc, enquanto uma estação envia dados a outras permanecem inertes consumindo uma corrente fixa, com a função de consumidores passivas de corrente, onde é necessária a terminação passiva aos dois extremos do barramento, podendo este assumir qualquer topologia.

Sendo assim, o consumo de corrente acima citado permanece em 10 mA, onde o dispositivo que envia dados sobrepõe +9 mA ou -9 mA à corrente em regime e, mesmo que o sinal seja transmitido por corrente, ele é lido por tensão.

A utilização do Profibus em dispositivos típicos e aplicações em controle de processos estão definidas segundo o perfil Profibus-*PA*.

Este perfil define os parâmetros dos equipamentos de campo e seu comportamento típico independente do fabricante e se aplica à transmissores de pressão, temperatura, posicionadores, etc. É baseado no conceito de blocos funcionais que são padronizados de tal forma a garantir a interoperabilidade entre os equipamentos de campo.

Os valores e status da medição, assim como os valores de "*setpoint*" recebido pelos equipamentos de campo no Profibus-*PA* são transmitidos ciclicamente com mais alta prioridade via mestre Classe um (DPM1). Já os parâmetros para visualização, operação, manutenção e diagnose são transmitidos por ferramentas de engenharia (mestre Classe dois, DPM2) com baixa prioridade através dos serviços acíclicos pelo DP via conexão C2 (GARCIA, 2011).

Ciclicamente também se transmite uma sequência de *bytes* de diagnósticos. A descrição dos bits destes bytes está no arquivo GSD do equipamento e dependem do fabricante (GARCIA, 2011).

O tempo de ciclo (Tc) aproximado pode ser calculado, como segue, quando se tem o *link device* na rede como mestre da rede PA:

Tc ≥ 10ms x número de equipamento + 10ms (serviços acíclicos mestre Classe 2) + 1.3ms (para cada conjunto de cinco *bytes* de valores cíclicos).

Imaginando a situação onde se tem cinco malhas de controle com cinco transmissores de pressão e cinco posicionadores de válvula. Teríamos um tempo de ciclo de aproximadamente 110 ms.

A transmissão de dados sobre o meio físico utiliza o código *MBP (**M**anchester Coding and **B**us **P**owered*), que significa codificação *Manchester* e energizada pelo barramento (MERLIN, 2011), pois nesta modalidade de rede os dados trafegam junto com a alimentação dos dispositivos pela própria linha de comunicação.

Na figura 15 temos uma representação do *Manchester* utilizado na rede Profibus *PA*.

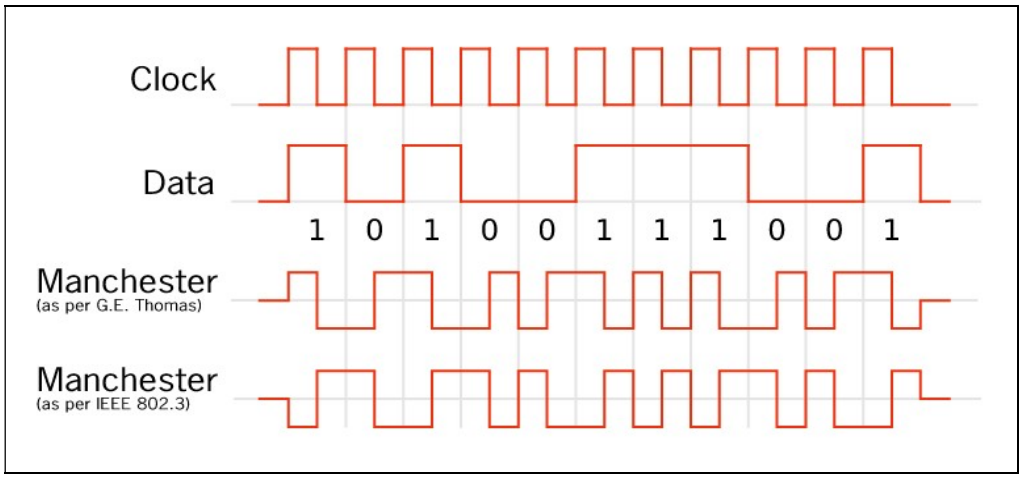

Figura 15 - Codificação Manchester.
Fonte*:* SMAR (2010)

Dentre as topologias mais comuns permitidas à rede Profibus PA, estão:

- Estrela
- Barramento

A figura 16 representa uma topologia estrela, onde um terminador concentra as conexões (*hub*) dos dispositivos e promove a difusão das mensagens.

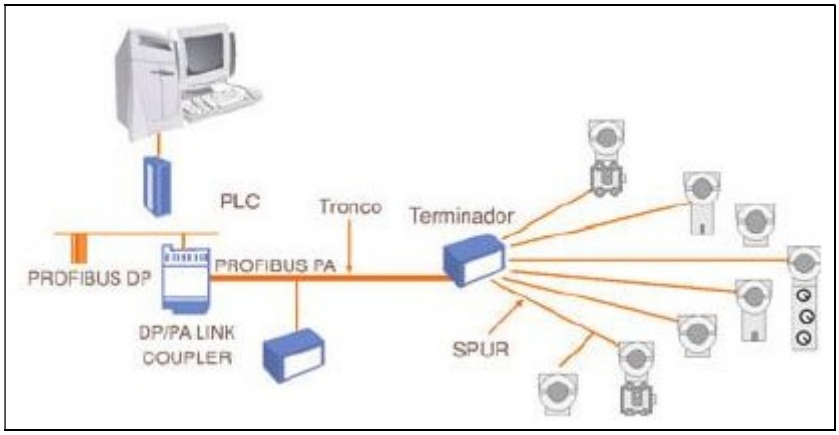

Figura 16 - Topologia em estrela.
Fonte: SMAR (2010).

E um exemplo de topologia em barramento (tronco) está representado na figura 17.

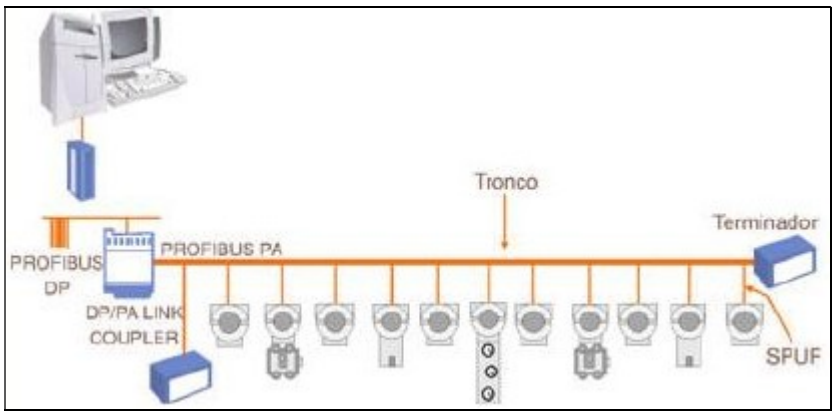

Figura 17 - Topologia Barramento.
Fonte: SMAR (2010).

7. RESULTADOS DAS SIMULAÇÕES

Como já dito antes, este trabalho buscou testar a viabilidade do *NS-2* para o desenvolvimento de um simulador para redes industriais;

A base do projeto é a construção de *scripts* em linguagem TCL para rodarem no *NS-2* e simularem os protocolos de redes industriais de modo gráfico utilizando um *software* auxiliar do *NS-2* chamado *NAM* (*Network Animator*);

Com os *scripts* funcionando, na topologia mestre-escravo, fez-se o levantamento dos dados da simulação tendo como foco nos seguintes resultados:

- Tempo de resposta dos pacotes ou *RTT*;
- Ciclos de operação;

Após a simulação, os dados foram analisados e comparados com valores teóricos para fins de validação. A topologia escolhida para a pesquisa foi do tipo básico e comumente utilizada na prática, tal como mestre-escravo utilizando o *polling* cíclico para consultas dos escravos, para diversos tamanhos de redes.

Devido às várias topologias da rede Profibus que foram simuladas, foi necessário, para um embasamento teórico, realizar inúmeros outros testes computacionais para inibição de prováveis falhas em diferentes aspectos antes adotar um *script* base, criado com uma topologia alterável para análise de comportamento futuro das simulações em Profibus, ou seja, para análise dos resultados. Foram simulados, através deste *script* base, uma topologia da rede Profibus com um mestre e cinco escravos e, em seguida, alterado para um mestre e 30 escravos, com saídas de amostragem para analisar, debater e compreender os vários aspectos das redes da Profibus em si.

Concluídos os procedimentos de instalação, pesquisa análises, testes e validação, iniciou-se uma das fases mais importantes: a análise dos resultados. Afinal, eles foram utilizados na elaboração dos gráficos que serviram de suporte a este trabalho e também para serem submetidos a próximos eventos científicos. Foi fundamental buscar consistência e coerência nesses resultados antes de apresentá-los.

A dificuldade maior estava em encontrar o procedimento mais apropriado para fazer com que o *script* executasse a função de simular o comportamento de uma rede da maneira mais realista possível e demonstrasse isso sistemicamente, ao mesmo tempo em que sua estrutura fosse voltada para a rede Profibus.

Como resultado, o método escolhido disponível no *NS*-2, que foi considerado adequado para criar o *script* mais adequado às tarefas de medir o *RTT* e o ciclo de operação, foi o Agente *Ping* (*Agent Ping*) cujas funcionalidades permitem enviar e receber pacotes de tamanhos variáveis e ainda capturar o *RTT* através de procedimentos internos.

O *script* a seguir foi utilizado para simular uma rede e registar o *RTT* para diversas quantidades de dispositivos, utilizando o método de acesso por consulta cíclica (*polling*), onde o mestre consulta cada escravo sequencialmente. Para facilitar o entendimento, o *script* é descrito passo-a-passo.

Aqui, primeiramente, é declarada a saída para o arquivo que irá plotar na tela a simulação através do *NAM*:

```
#Open a trace file
set nf [open out.nam w]
$ns namtrace-all $nf
```

O procedimento a seguir, parametriza o simulador com os dados de rede e dos dispositivos básicos de maneira a se comportar como uma rede Profibus (SMAR, 2010). Isto torna o *script* muito prático e funcional, pois os parâmetros podem ser facilmente alterados conforme a configuração exigida:

```
#Parâmetros
set Ndisp 10
set NCORE [expr 2+$Ndisp]
set Tpclp 1                ;#Tempo (ms) de processamento do CLP
set Tpd   3.4              ;#Tempo (ms) de proc. do dispositivo
set Tph   3                ;#Tempo (ms) de processamento do hub
set Tsim [expr 0.3*$Ndisp] ;#Tempo de simulação em função do tamanho da
rede
puts "NCORE=$NCORE"
puts "Origem  Destino    Dados(Bytes)     RTT (ms)    RTTacumulado (ms)"
```

Na sequência, podemos observar a criação dos nós, a declaração de cores aos pacotes assim definidos bem como a velocidade da rede e o atraso nominal do *link*:

```
#Criando três nós
set n(0) [$ns node]
set n(1) [$ns node]
$n(0) color "red"
$n(1) color "blue"
$n(0) shape "square"
$n(1) shape "square"
$ns duplex-link $n(0) $n(1) 1.5Mb 1ms DropTail
```

A estrutura, a seguir, trata-se de um laço de repetição, com o comando *for*, para criação dos enlaces entre os nós e associação dos agentes de tráfego:

```
for {set i 2} {$i < $NCORE} {incr i} {

    set Delay($i) [uniform 1 3]
    set n($i) [$ns node]
    #Conectando os nós a dois links
    $ns duplex-link $n(1) $n($i) 1.5Mb $Delay($i)ms DropTail

}

#Criando dois agentes ping e atachando eles ao nó 0 e nó 2.
set p(0) [new Agent/Ping]
$p(0) set packetSize_ 36    ;#definição do tamanho do pacote (request frame
Profibus)
$ns attach-agent $n(0) $p(0)
```

Também foram criados laços de repetição para outras instâncias, de maneira que também fosse possível realizar a simulação com outras configurações padrão da rede Profibus, à exemplo dos frames de resposta com dados que podem ter o tamanho máximo de 246 *Bytes*:

```
for {set i 2} {$i < $NCORE} {incr i} {
    set Pkt($i) [expr int([uniform 36 246])]
    set p($i) [new Agent/Ping
 Agent/Ping set packetSize_ $Pkt($i)
    #$p($i) set packetSize_ 246
    set p($i) [new Agent/Ping]
    $ns attach-agent $n($i) $p($i)
}
```

A seguir, o código seleciona, define e efetua os cômputos de *RTT* unitário, plotando na tela do terminal os caminhos do pacote, *RTT* e *RTT* acumulado:

```
#Define uma função 'recv'  para a classe agente/ping
set Trtt 0
Agent/Ping instproc recv { from rtt } {
        global  ns j NCORE Ndisp Trtt Tpclp Tpd Tph Tcons Pkt
  $self instvar node_
  set dest $j
  set Trtt [expr $rtt + $Trtt]
  set Tcons [expr $Trtt + $Ndisp*$Tpd + $Tpclp + $Tph]
  puts "[$node_ id]       $dest       $Pkt($dest)       $rtt       $Trtt"
}
```

Procedimento que lê, salva e executa os cômputos dos tempos em cada dispositivo:

```
set cont 1
proc dump { interval } {
  global  ns j p NCORE Ndisp cont

  #puts " Ndisp=$Ndisp    $interval  [$ns now]."
  $ns at [expr [$ns now]+$interval] "dump $interval"
  #Connect the two agents

        set j [expr 1+$cont]
    if {$j < $NCORE} {
        $ns connect $p(0) $p($j)
        $ns at [expr [$ns now]] "$p(0) send"
        set cont  $j
        #puts "Destino=$j"
    }
    return
}
```

Ao final, um procedimento faz a chamada ao *NAM*, fornecendo como parâmetros os arquivos gravados em *out.nam* e finaliza os processos:

```
$ns at 0.1  "dump 0.1"

$ns at [expr $Tsim]  "finish"

#Define a 'finish' procedure
proc finish {} {
        global ns nf Trtt Tcons NCORE Tpd
        $ns flush-trace
        close $nf

        puts "  "
        puts "  Trtt = $Trtt ms   Ciclo = $Tcons ms "
        puts "  "
        #puts " $Trtt   $Tcons    [expr $Tcons-$Trtt]" >> RTT.txt
```

```
        puts "running nam..."
        exec nam out.nam &
        exit 0
}
#Run the simulation
$ns run
```

O script completo está representado no Anexo A.

Na simulação do *script* foram obtidas as saídas de diversos tipos de informação sobre a rede, a saber:

- *RTT*, considerando o tamanho do pacote enviado por cada dispositivo da rede e o tempo de propagação do meio físico (enlace);
- Tcons, tempo total da consulta a cada dispositivo, considerando o *RTT* acrescido dos tempos de processamento de cada um dos nós no caminho da mensagem, tempo do mestre (Tclp, normalmente é um CLP), tempo do dispositivo consultado (Tpd) e tempo do hub (Tph). Todos são valores médios fornecidos pelo fabricante.
- Duração do ciclo, tempo do ciclo de consultas efetuadas pelo mestre.

Nos Quadros 3 e 4, a seguir, estão representadas as saídas para cinco e 30 dispositivos, lembrando que ainda se deve sempre acrescentar o mestre e o *hub*, ou seja, para rede com cinco dispositivos temos um total de sete nós.

Mencionados no canto superior esquerdo, o número de dispositivos e os nós e origem, destino. Observando os dados do *script,* é notável que ele também faz uma variação aleatória ou randômica do tamanho dos pacotes em *bytes* entre o valor mínimo e máximo o que aproxima ainda mais de uma simulação real da rede Profibus.

RTT unitário em milissegundos de cada pacote, *RTT* acumulado a cada envio e, ao final, o Trtt, que nada mais é do que a soma total, antes da duração total do ciclo.

Obviamente este projeto destina-se a simular os *links* nó á nó de uma rede industrial com a configuração mais próxima possível do Profibus. O fato de poder simular o *RTT* específico possibilita um enorme avanço com o *NS*-2 nesse aspecto, permitindo a criação e a variação desse *script* em outros aspectos e configurações e até mesmo em outros modelos de redes industriais.

O procedimento pode ser repetido e alterando facilmente alguns dados já especificados, a simulação poder tomar as mais variadas formas, respeitando-se as topologias permitidas e adotadas aos ensaios.

```
Ndisp=5
NCORE=7
Dados= pacotes de retorno gerados estatisticamente a partir de uma
distribuição uniforme.

Origem  Destino    Dados(Bytes)     RTT (ms)    RTTacumulado (ms)

  0       2           203             6.6         6.599999999999996
  0       3           101             7.7         14.300000000000001
  0       4           207             6.6         20.899999999999999
  0       5            85             5.6         26.5
  0       6           170             8.0         34.5

  Trtt = 34.5 ms   Duração do Ciclo = 55.5 ms

running nam...
```

Quadro 3 - Saída do *script* para rede Profibus com 5 dispositivos.
Fonte: o próprio autor.

Para efeito de comparação, foram realizadas simulações para redes Profibus do tipo DP e PA para velocidades de 1,5 Mbps e 12 Mbps. Com os dados dos *RTT*'s de cada enlace e os tempos de consulta totais, foi possível observar as diferenças de desempenho entre elas. Foram construídos gráficos com auxílio do *Gnuplot* (descrito na seção 7). Para leitura das diferenças, que por vias de análise apresentaram-se bem sutis, mas esperadas, estão representados os gráficos do *RTT*, ver figura 18, Ciclo de operação, ver figura 19 e Processamento dos nós, ver figura 20.

```
Ndisp=30
NCORE=32
Dados= pacotes de retorno gerados estatisticamente a partir de
um distribuiçãoo uniforme.


Origem  Destino     Dados(Bytes)      RTT (ms)       RTTacumulado
(ms)

  0         2             86             5.3           5.3
  0         3            214             8.9          14.2
  0         4            178             6.3          20.5
  0         5            196             6.8          27.3
  0         6            226             8.6          35.9
  0         7            121             8.9          44.8
  0         8             58             6.3          51.1
  0         9            120             8.9          60.0
  0        10             75             6.1          66.1
  0        11            222             9.3          75.4
  0        12            234             8.1          83.5
  0        13            223            10.1          93.6
  0        14             59             5.7          99.3
  0        15             48             5.1         104.4
  0        16             38             7.3         111.7
  0        17            160             6.5         118.2
  0        18            106             9.0         127.2
  0        19             61             6.7         133.9
  0        20            176             7.0         140.9
  0        21             89             5.7         146.6
  0        22            182             9.6         156.2
  0        23            150             7.4         163.6
  0        24            220             7.8         171.4
  0        25             90             7.0         178.4
  0        26            152             8.1         186.5
  0        27            125             7.6         194.1
  0        28             41             7.6         201.7
  0        29            104             8.9         210.6
  0        30             75             8.5         219.1
  0        31            200             7.4         226.5

  Trtt = 226.5 ms   Duração do Ciclo = 332.5 ms


running nam...
```

Quadro 4 - Saída do *script* para rede Profibus **com 30 dispositivos.**
Fonte: o próprio autor.

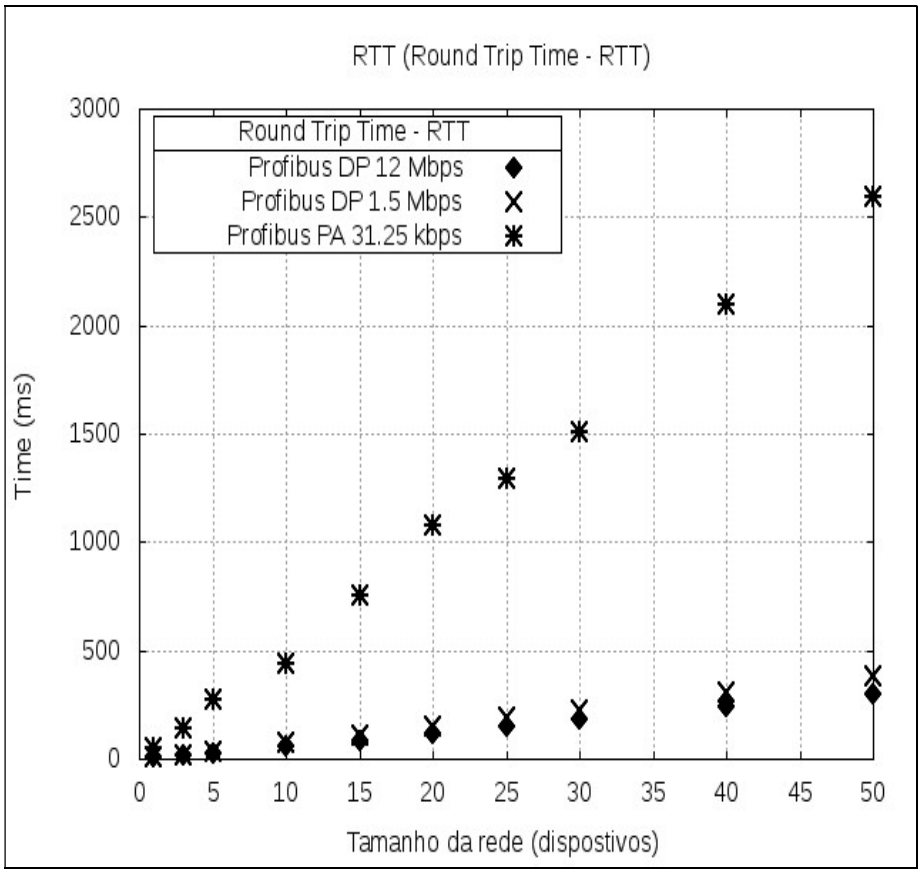

Figura 18 - *RTT* para redes até 50 dispositivos.
Fonte: o próprio autor.

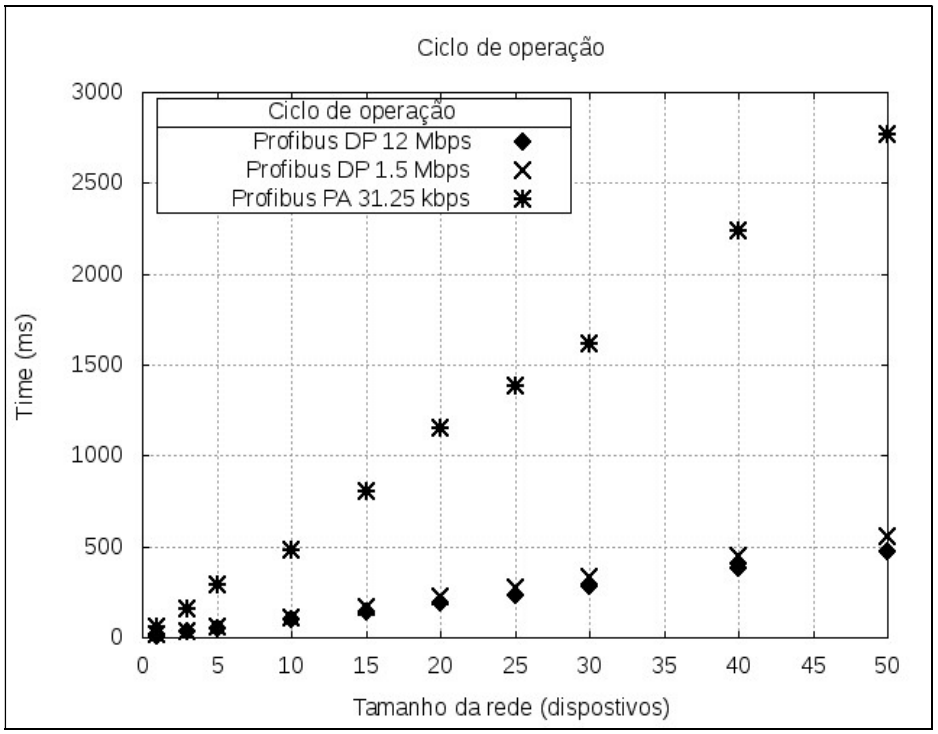

Figura 19 - Ciclo de operação para redes até 50 dispositivos.
Fonte: o próprio autor.

Os tempos *RTT* e dos ciclos de operação de redes até 50 dispositivos e a projeção em milissegundos. Estes gráficos mostram um comportamento linear em relação ao tamanho da rede. Isto se deve ao método de acesso por *polling* que evita congestionamento na rede e impede que as filas de transmissão e recepção produzam atrasos aleatórios. Este dado é importante para programar eventos em uma linha de produção associados à atuação de dispositivos atuadores / sensores da rede.

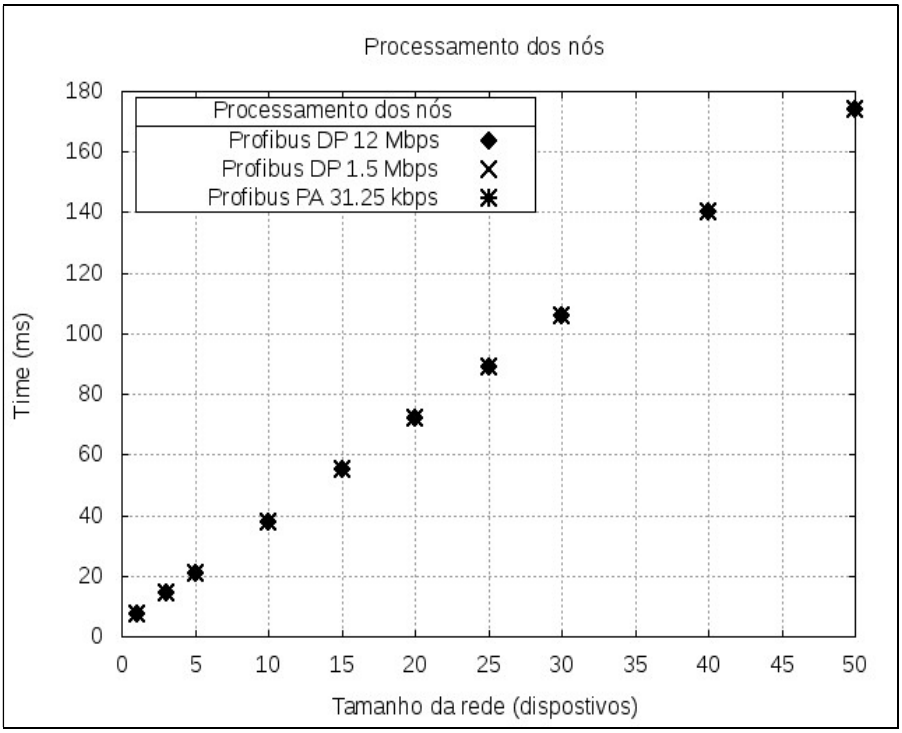

Figura 20 - Tempo de processamento para redes até 50 dispositivos.
Fonte: o próprio autor.

Outro fato observável é o comportamento do processamento dos nós, que se manteve linear mesmo com a mudança do tipo de rede e configuração dos parâmetros da rede (meio físico). Isto mostra uma incerteza na simulação, pois o comportamento de cada dispositivo deve ter uma componente aleatória, devido à sua dependência dos atuadores / sensores a eles conectados. Desta feita, há a necessidade de se construir *scripts* levando-se em conta os fatores que influenciam o processamento dos dados pelos dispositivos.

Apesar de não ser o objetivo do presente trabalho o aprofundamento sistêmico de outros assuntos que não seja relacionado diretamente à simulação de redes, optou-se por mostrar como foram obtidos os gráficos mostrados anteriormente. Nesse aspecto, o *Gnuplot* é ferramenta utilizada para visualização e interpretação de gráficos que serão associados às saídas geradas pelo *NS*-2 (TROTA, 2007). Porém, apesar da infinidade de recursos que este excelente visualizador oferece, utilizar-se-á todos quanto necessários para o desenvolvimento de *script* e dos relatórios.

Uma função muito útil e a que foi utilizada é a de plotar um gráfico a partir de dados contidos num arquivo de texto.

A seguir, nos quadros 5 a 8, estão representados os *scripts* do Gnuplot e os respectivos dados para a geração dos gráficos apresentados neste trabalho.

Foram consideradas as modalidades de redes comumente utilizadas no mercado, a saber, Profibus DP / 1,5 Mbps, Profibus DP / 12 Mbps e Profibus PA / 31,25 kbps.

```
#Script para o arquivo rtt.gnu
reset
set terminal png
set title " RTT (Round Trip Time) "
set xlabel "Tamanho da rede (dispostivos)"
set ylabel "Time (ms)"
set key left box 1
show key
set key title "Round Trip Time"
set grid
set output "rtt.png"
plot "rtt_1_5M.dat" u ($1):($2)  with lp ls 8 lw 2 title
"Profibus DP 1.5 Mbps", "rtt_12M.dat" u ($1):($2) with lp ls 6
lw 2 title "Profibus DP 12 Mbps", "rtt_31_25k.dat" u ($1):($2)
with lp ls 1 lw 2 title "Profibus PA 31.25 kbps"
replot
#Fim
```

Quadro 5 - *Script* para geração do gráfico do *RTT* para as modalidades de redes Profibus com velocidades de 1.5 Mbps, 12 Mbps e 31,25 kbps.

```
#Rede Profibus DP 1.5Mbps

n nós      RTT(ms)    Jobs(ms)   Proc (ms)

  1          6.0        13.4       7.4
  3         19.61       33.8      14.2
  5         34.5        55.5      21.0
 10         71.5       109.5      38.0
 15        110.9       165.9      55.0
 20        149.8       221.8      72.0
 25        186.6       275.6      89.28
 30        226.5       332.5     106.0
 40        309.8       449.8     140.0
 50        383.4       557.4     174.0
```

Quadro 6 - Arquivo de dados para a rede Profibus DP / 1,5 Mbps.

```
#Rede Profibus DP 12Mbps

n nós      RTT(ms)    Jobs(ms)   Proc (ms)

  1          4.3        11.7       7.4
  3         14.9        29.1      14.2
  5         25.6        46.6      21.0
 10         57.7        95.7      38.0
 15         86.9       141.9      55.0
 20        115.1       187.1      72.0
 25        145.5       234.5      89.0
 30        178.8       284.8     106.0
 40        243.2       383.2     140.0
 50        301.6       475.6     174.0
```

Quadro 7 - Arquivo de dados para a rede Profibus DP / 12 Mbps.

```
#Rede Profibus PA 31.25Mbps

n nós      RTT(ms)    Jobs(ms)   Proc (ms)

  1         53.1        60.5       7.4
  3        143.1       157.3      14.2
  5        271.7       292.7      21.0
 10        442.7       480.7      38.0
 15        752.4       807.4      55.0
 20       1079.6      1151.6      72.0
 25       1293.1      1382.1      89.0
 30       1506.8      1612.8     106.0
 40       2099.3      2239.3     140.0
 50       2589.9      2763.9     174.0
```

Quadro 8 - Arquivo de dados para a rede Profibus PA / 31,25 kbps.

Como ilustração, nas figuras 21 e 22 tem-se as simulações apresentadas graficamente pelo *NAM*. O nó zero representa o mestre (CLP) e o nó 1 é o dispositivo repetidor.

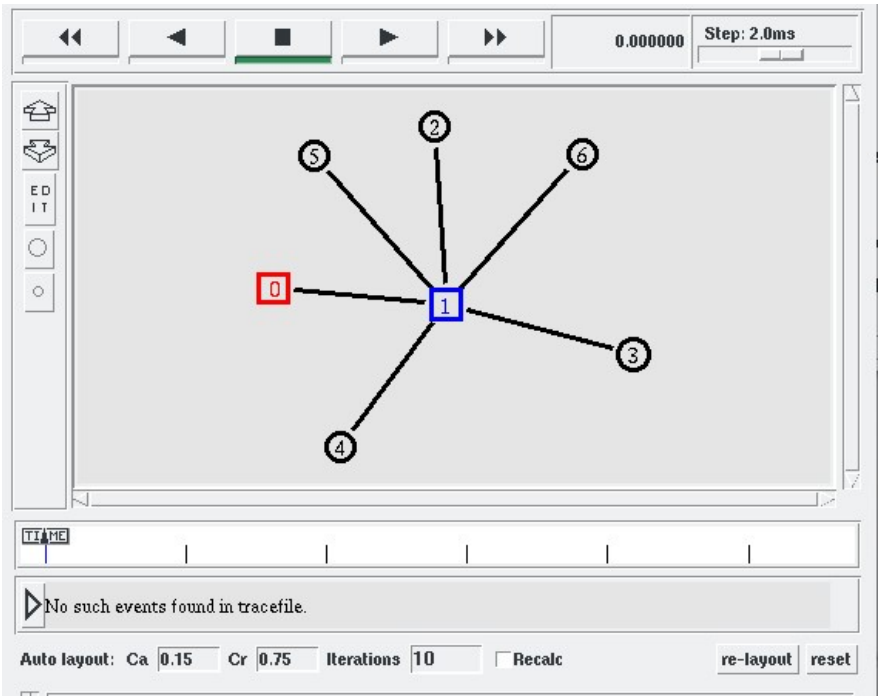

Figura 21 - Simulação gráfica pelo *NAM*, rede com 5 dispositivos.
Fonte: o próprio autor

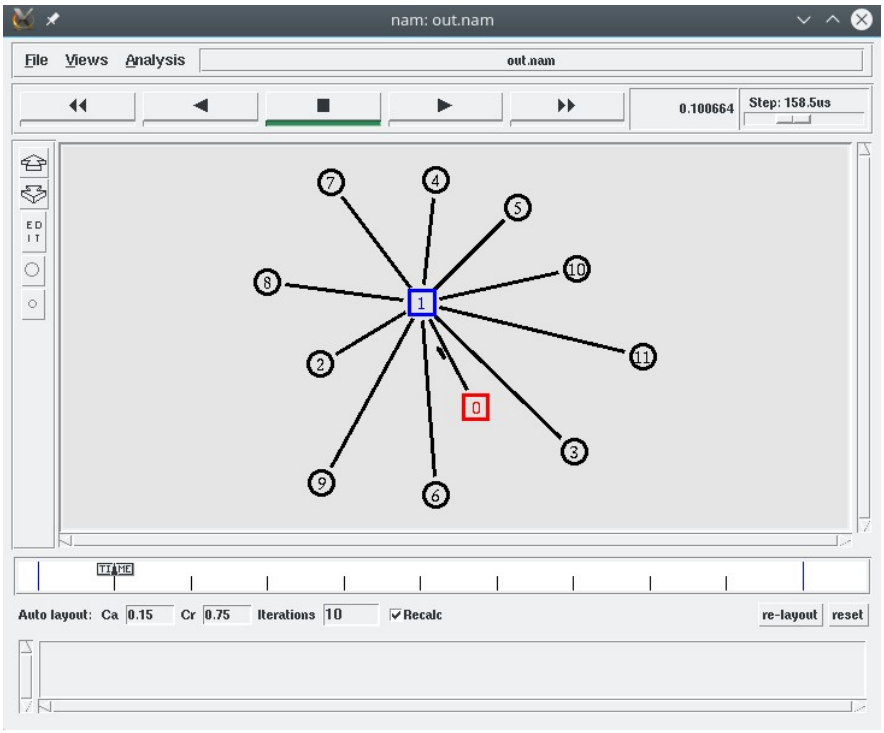

Figura 22 - Simulação gráfica pelo *NAM*, rede com 10 dispositivos.
Fonte: o próprio autor

8 RECURSOS UTILIZADOS

8.1 COMPUTADOR

Após serem realizadas todas as pesquisas, foi providenciado um computador que tivesse uma versão de software compatível com o programa a ser testado de maneira confiável e prática.

No decorrer de um ano aproximadamente, tempo este em que a idéia sobre o projeto em questão foi pautado entre orientador e orientado, a análise da viabilidade como Pré – proposta, a aquisição dos *softwares*, instalação, validação, ensaios, manutenção em equipamentos danificados, tempo á disposição, viagens para orientação, dentre as quais muitas foram circunstâncias adversas enfrentadas para a obtenção dos resultados plotados neste artigo.

Após várias tentativas frustradas, pelo fato de não ser a mais apropriada para simulações, até mesmo por suas características construtivas, mas que realizou bem os ensaios foi instalado em uma máquina da marca Positivo, Modelo *Unique S1990*, com processador *Intel Celeron B800* @ 1,5 GHz (Giga hertz), com memória *RAM* de 4GB e 500 GB de memória física, os *softwares*: sistema operacional *Linux (Opensuse Leap versão 42.1*) e o programa *NS-2* (*Network Simulator* versão 2.35) com partição para o Windows 7.

8.2 NETWORK SIMULATOR 2

A versão do programa em questão é a 2.35, revisada em 4 de novembro de 2011. Está versão está dotada dos seguintes sistemas:

- *Tcl/Tk*: Interpretador para *TCL* (interface para usuário).
- *Otcl: Tcl* com orientação a objetos.
- *Tclcl:* Implementação de classes para *Tcl*.
- *Ns-2:* Classes do simulador.

- *NAM-1:* Visualizador e animador gráfico.
- *X-graph:* Ferramenta de plotagem de gráficos.
- *Cweb e SGB*: Bibliotecas.
- *Gt-itm e gt-itm*, *Sgb2-ns*: Geradores de topologias.
- *Zlib:* Ferramenta para compressão de arquivos.
- *NAM: Network Animator* v. 1.11 (animador de dados)
- *Gnuplot:* Visualizador de gráficos

8.3 GNUPLOT

A versão do Gnuplot utilizada foi a 5.0.

9 CONSIDERAÇÕES FINAIS

Esta pesquisa se propôs testar a viabilidade do uso do *NS*-2 como ferramenta de *software* aberto para a simulação de redes industriais, especificamente a Profibus e suas topologias básicas. O *NS*-2 é um *software* muito conhecido e amplamente utilizado no meio acadêmico, principalmente entre os estudantes das áreas de redes e computação, também por ser um grande objeto de estudo, análise e simulação. Entretanto, até esta data, muito pouco material é encontrado na literatura a respeito da utilização deste *software* voltado as redes industriais, incluindo a Profibus, escolhida como objeto de simulação neste trabalho.

Sendo assim, sem dúvida alguma, a importância empregada neste mérito, trará avanços e agregará valores às pesquisas e projetos futuros relacionados ás áreas de redes industriais juntamente ao meio acadêmico, quer seja de desenvolvimento tecnológico comercial ou não.

Os objetivos foram atingidos, pois este trabalho trata de propor novos rumos a outras pesquisas relacionadas ao tema.

Todas as dificuldades concernentes a este trabalho foram as mesmas de qualquer outro tópico, ou seja, a primeira impressão ao novo, ao desconhecido, que por sinal e devido ao grande interesse despertado, logo foi colocado por terra, adotando-se uma prática comum e interessante.

A respeito da execução dos ensaios, apesar das dificuldades já esperadas, outras inesperadas serviram de desafios bem como uma determinada motivação e alento para o prosseguimento com a pesquisa, que resultou em um trabalho sucinto, interessante e de base sólida para futuros desenvolvimentos de pesquisas correlacionados, até mesmo com o objetivo de correções.

Foi utilizada a versão 2.35, pois se tratando de *software* com o código fonte aberto, além de adotar-se a prática de utilizar sempre uma versão anterior, pelo fato de nesse caso a maioria dos "*bugs*" ou defeitos já puderam ser analisados e resolvidos, todos os recursos estão habilitados e/ou aperfeiçoados, ao contrário do *NS*–3, no qual, por exemplo, até esta data, tem uma versão do *NAM* que ainda não está disponível para uso aberto, vetado apenas a desenvolvedores associados.

Recomendações para trabalhos futuros:

- Estabelecer modelos estatísticos de tráfego, para representar Redes Industriais mais realísticas, com variações do fluxo de dados, correção de erros, retransmissões, descarte de pacotes, congestionamentos;

- Considerar o gerenciamento das filas (*queue*) nas extremidades dos enlaces, considerando o nível de utilização das filas, perdas de pacotes, entre outros;

- Considerar topologias com difusão de pacotes (*multicasting*), roteamento e priorização de pacotes;

- Construção de agentes (*Agen*t) específicos para representar aplicações em redes industriais. Isto implica na construção de bibliotecas com códigos em C++ (C++ source);

- Testar a versão mais atual do *NS* (*NS*-3) para se obter maior poder, facilidade e flexibilidade nas configurações.

REFERÊNCIAS

A FERRAMENTA DE SIMULAÇÃO *NS-2*: NETWORK SIMULATOR. 2015. Disponível em: <http://www.harpia.eng.br/pesquisa/tfc_EngElet_Rafael.pdf>. Acesso em 21/08/2015.

ARISTÓTELES. Metafísica, "**Livro A, cap. I**". Coleção Os Pensadores. Editora Abril, São Paulo, 1979 (orig. século IV A.C.).

ASSOCIAÇÃO PROFIBUS BRASIL. **Descrição Técnica 2009.** São Paulo, agosto de 2009, p 3 – 71.

BACON, F. **Novum Organum. "Aforismo XIX"**. Coleção Os Pensadores. Nova Cultural, São Paulo, 1988 (orig. 1620).

BARROS, A. J. S. e LEHFELD, N. A. S. **Fundamentos de Metodologia: Um Guia para a Iniciação Científica**. 2 Ed. São Paulo: Makron Books, 2000.

CARVALHO, Luís Vidal A. **Datamining: A mineração de dados no marketing, medicina, economia, engenharia e administração**. São Paulo: Editora Érica, 2001.

CASSIOLATO, C. **Redes Industriais – Parte 1**. Revista Saber Eletrônica. São Paulo. Editora Saber, 2012. Edição 461. Páginas 24 a 32.

CHOWDHURY, Dhiman D. **Projetos avançados de redes IP, roteamento, qualidade de serviço e voz sobre IP**. Leiden. Editora Elsevier. 2001. *NS-2* - THE NETWORK SIMULATOR. Disponível em: <http://www.isi.edu/nsnam>. Acesso em 19/08/2015.

COUTINHO, Mário M. **Network Simulator, Guia básico para iniciantes.** Tutorial. 2003. Disponível em: <http://www.cin.ufpe.br/~sfd/universo/sim/nsr1>. Acesso em 29/11/2015.

FERREIRA, Aurélio B. H**. Aurélio, O minidicionário da língua portuguesa.** 4 edição, revista e ampliada do minidicionário Aurélio, 7 impressão. Rio de janeiro: Editora Positivo. 2002.

FORMATAÇÃO de trabalhos acadêmicos. Disponível em: <http://www.utfpr.edu.br/cornelioprocopio/biblioteca-e-producao-academica/acesso-usuario/arquivos/modelo_normal_trab_acad/view> Acessado em: 10/09/14.

GARCIA, Mário A. **Métodos para diagnósticos em redes Profibus.** Trabalho de conclusão de curso. Câmpus Cornélio Procópio. Biblioteca UTFPR-CP, 2011.

GREIS, Marc. **Simulator *NS-2***. Tutorial. 2011. Disponível em: < **http://www.isi.edu/nsnam/ns/tutorial/index.html** >. Acesso em 11/05/2016

JAIN, R. **The art of computer systems performance analysis**. John Wiley & Sons, 1991 apud BRITO, Sergio de Figueiredo et al. **Simulation tool and usage strategy – a practical and pragmatic approach.** In Anais do V INDUSCON 2002 (IEEE – Industry Applications Society), Salvador, BA, 03 jul. 2002.

LUGLI, Alexandre B.; SANTOS, Max M. Dias. **Redes Industriais para a automação industrial: AS-I, Profibus e Profinet.** São Paulo. Editora Saraiva, 2010.

MARTINS, Joberto Sérgio Barbosa; FLEMING, Paulo Victor; RAMOS, João Carlos Rodrigues; ARAÚJO, Rafael Gonçalves Bezerra; PORTNOI, Marcos. **Simulation tool and usage strategy – a practical and pragmatic approach.** In Anais do V INDUSCON 2002 (IEEE – Industry Applications Society), Salvador, BA, 03 Jul. 2002.

MECATRÔNICA ATUAL, Simulador de redes industriais, 2007. Disponível em: <http://www.mecatronicaatual.com.br/educacao/815-simulador-de-redes-industriais> Acesso em 15/08/2015.

MERLIN, Paulo. **Correção técnica instalação de rede de comunicação Profibus DP/PA.** Trabalho de conclusão de curso. Câmpus Cornélio Procópio. Biblioteca UTFPR-CP, 2011.

NETO, João Amato**. Manufatura Classe Mundial: Conceitos, estratégias e aplicação**. São Paulo. Editora Atlas, 2001.

***NS* 2 – SIMULADOR DE REDE.** Disponível em: <http://laplace.eletrica.ufpr.br/ns2.html>. Acesso em 11/05/2016.

PRÉ-TEXTO, Divisão de Bibliotecas e Documentação, 2006. PUC-Rio. Disponível em: <www2.dbd.puc-rio.br/tesesabertas>. Acesso em 09/06/2016.

SITE OFICIAL DO *NS*-2. Disponível em: <http://www.isi.edu/nsnam/ns> Acesso em 06/09/2015.

SITE OFICIAL DO OPENSUSE LEAP 42.1. Disponível em: <https://software.opensuse.org/421/pt_BR>. Acesso em 03/05/2016.

SMAR. **Profibus, A estrutura dos Frames.** 2010. Disponível em: <http://www.smar.com/brasil2/artigostecnicos> Acesso em 18/06/2016.

TROTA, Leonardo de S**. GNUPLOT. Manual simplificado para iniciantes.** Apostila. Câmpus Rio Claro. Universidade Estadual Paulista Júlio Mesquita Filho.2007 Disponível em**: <** http://www.ebah.com.br/content/ABAAAgUS4AF/gnuplot-manual-simplificado-iniciantes> Acesso em 18/05/2016.

VASQUES, Alan Tamer; ESTEVES, Rafael Pereira; ABELEM, Antônio Jorge Gomes. **Simulação de redes de Computadores utilizando o Network Simulator. XI Semana Paraense de informática da UFPA. SEMINF 2004**. WEKA – The University of Waikato. Disponível em: <http://www.cs.waikato.ac.nz/weka>. Acesso em 24/08/2015.

ANEXO A

```
#Script ping-2.tcl
#
#TCC2 do aluno: Giancristian Franzoso
#
#Create a simulator object
set ns [new Simulator]

#Open a trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Parâmetros

set Ndisp 10
set NCORE [expr 2+$Ndisp]
set Tpclp 1                 ;#Tempo (ms) de processamento do CLP
set Tpd   3.4               ;#Tempo (ms) de processamento do
dispositivo
set Tph   3                 ;#Tempo (ms) de processamento do hub
set Tsim  [expr 0.3*$Ndisp] ;#Tempo de simulação em função do tamanho
da rede
puts "NCORE=$NCORE"
puts "Origem  Destino    Dados(Bytes)     RTT (ms)    RTTacumulado
(ms)"

#Create three nodes
set n(0) [$ns node]
set n(1) [$ns node]
$n(0) color "red"
$n(1) color "blue"
$n(0) shape "square"
$n(1) shape "square"
$ns duplex-link $n(0) $n(1) 1.5Mb 1ms DropTail

for {set i 2} {$i < $NCORE} {incr i} {

    set Delay($i) [uniform 1 3]
    set n($i) [$ns node]
    #Connect the nodes with two linksputs
    $ns duplex-link $n(1) $n($i) 1.5Mb $Delay($i)ms DropTail

}

#Create two ping agents and attach them to the nodes n0 and n2
set p(0) [new Agent/Ping]
$p(0) set packetSize_ 36
$ns attach-agent $n(0) $p(0)

for {set i 2} {$i < $NCORE} {incr i} {
    set Pkt($i) [expr int([uniform 36 246])]
    set p($i) [new Agent/Ping]
    Agent/Ping set packetSize_ $Pkt($i)
    #$p($i) set packetSize_ 246
    set p($i) [new Agent/Ping]
```

```
    $ns attach-agent $n($i) $p($i)
}

#Define a 'recv' function for the class 'Agent/Ping'
set Trtt 0
Agent/Ping instproc recv { from rtt } {
        global  ns j NCORE Ndisp Trtt Tpclp Tpd Tph Tcons Pkt
  $self instvar node_
  set dest $j
  set Trtt [expr $rtt + $Trtt]
  set Tcons [expr $Trtt + $Ndisp*$Tpd + $Tpclp + $Tph]
  puts "  [$node_ id]       $dest           $Pkt($dest)            $rtt
$Trtt"
}

set cont 1
proc dump { interval } {
  global  ns j p NCORE Ndisp cont

  #puts " Ndisp=$Ndisp    $interval  [$ns now]."
  $ns at [expr [$ns now]+$interval] "dump $interval"
  #Connect the two agents

        set j [expr 1+$cont]
    if {$j < $NCORE} {
        $ns connect $p(0) $p($j)
        $ns at [expr [$ns now]] "$p(0) send"
        set cont  $j
        #puts "Destino=$j"
    }
    return
}

#Schedule events

$ns at 0.1  "dump 0.1"

$ns at [expr $Tsim]  "finish"

#Define a 'finish' procedure
proc finish {} {
        global ns nf Trtt Tcons NCORE Tpd
        $ns flush-trace
        close $nf

        puts "  "
        puts "  Trtt = $Trtt ms   Ciclo = $Tcons ms "
        puts "  "
        #puts " $Trtt   $Tcons    [expr $Tcons-$Trtt]" >> RTT.txt

        puts "running nam..."
        exec nam out.nam &
        exit 0
}
#Run the simulation
$ns run
#eof
```