SERGIO ROBERTO DE MELLO CANOVAS

INTEGRAÇÃO ENTRE REDES LONWORKS E REDES IP: APLICAÇÕES, REQUISITOS E SOLUÇÕES

São Paulo
2006

SERGIO ROBERTO DE MELLO CANOVAS

Integração entre redes LonWorks e redes IP: aplicações, requisitos e soluções

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

Área de Concentração:
Sistemas Digitais

Orientador:
Prof. Livre-Docente
Carlos Eduardo Cugnasca

São Paulo
2006

DEDICATÓRIA

Aos meus pais e professores da Escola Politécnica, que permitiram e sempre incentivaram minha formação profissional e aperfeiçoamento técnico e humano.

AGRADECIMENTOS

A todos que, direta ou indiretamente, colaboraram na execução deste trabalho, principalmente aos professores Miguel dos Santos Alves Filho e Carlos Eduardo Cugnasca.

Às empresas TAC, Schneider Electric, PureChoice e Conceito Tecnologia pelo fornecimento de módulos e componentes utilizados em todo o estudo e aprendizado necessário para este trabalho.

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), que através do apoio ao projeto TIDIA permitiu a aquisição de equipamentos utilizados neste trabalho.

RESUMO

CANOVAS, S.R.M. **Integração entre redes LonWorks e redes IP: aplicações, requisitos e soluções**. 2006. 193 p. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2006.

As redes de controle tendem a ser cada vez mais utilizadas nos sistemas de automação. Sua interconexão com a Internet também apresenta uma interessante alternativa, gerando uma demanda para novos estudos. Este trabalho considera a interconexão entre redes de controle LonWorks e redes baseadas no *Internet Protocol* (IP) sob o ponto de vista de quatro categorias de aplicações: monitoração remota de redes LonWorks via rede IP; gerenciamento remoto de redes LonWorks via rede IP; utilização de *backbone* IP para interligar diferentes redes LonWorks; e utilização de *backbone* IP para interligar diferentes redes LonWorks com aplicações de requisitos de tempo real. Quatro questões delimitam o escopo do estudo: que tipos de soluções existem para a interconexão entre redes LonWorks e IP; que tipos de parâmetros de rede devem ser considerados para uma solução baseada em redes LonWorks/IP e como eles podem ser utilizados para estabelecer requisitos de uma certa aplicação; a quais requisitos cada uma das quatro categorias de aplicações referidas está associada; quais tipos de soluções, dentre as apresentadas, melhor atendem a cada uma das quatro categorias de aplicações. As respostas a essas questões foram embasadas tanto em estudos teóricos quanto em experimentos. Através do levantamento do estado da arte dessa área, reunindo o conhecimento atual, procurou-se uniformizar a nomenclatura associada e apresentar uma série de elementos e conceitos que servem como base teórica para avaliação e projeto de soluções de interconexão entre redes LonWorks e IP. Introduziu-se também novas explicações, exemplos e considerações sobre o assunto. Desenvolveu-se um aplicativo que permite emular uma situação de laço de controle sobre redes LonWorks e IP, obtendo-se uma ferramenta prática para a realização de testes ou mesmo para auxiliar na elaboração de projetos de novas redes LonWorks interligadas a redes IP. Como contribuições do trabalho, destacam-se: levantamento de parâmetros de rede quantitativos para estabelecimento de requisitos de aplicações LonWorks e IP; desenvolvimento de um aplicativo para experimento de controle sobre LonWorks/IP; análise do conhecimento atual sobre a tecnologia LonWorks, harmonizando as diversas visões; elucidação dos relacionamentos entre as soluções tecnológicas para interconexão LonWorks/IP com as categorias de aplicações da tecnologia.

Palavras-chave: Redes de controle, LonWorks, LonTalk, Controle sobre IP, EIA709, EIA852

ABSTRACT

CANOVAS, S.R.M. **LonWorks and IP networks interconnection: applications, requirements and solutions**. 2006. 193 p. Master thesis - Escola Politécnica, Universidade de São Paulo, São Paulo, 2006.

Control networks utilization on automation systems has been increasing. Their interconnection with the Internet is also an interesting option, creating the necessity of new studies. This work considers the interconnection between LonWorks control networks and IP-based networks. Four categories of applications are the focus: remote monitoring of LonWorks networks thorugh IP networks; remote management of LonWorks networks through IP networks; use of an IP network as a backbone to interconnect more than one LonWorks network; and the same use of an IP network for applications with real time requirements. Four questions establish the escope of this study: what kinds of solutions are available to provide interconnection between LonWorks and IP networks; what kinds of network parameters should be considered in a LonWorks/IP based solution and how can they be used to determine requirements for an application; which are the most common requirements for each one of the four categories of applications; and which solution does best fit to each category.

The answers of these questions were based on theoretical studies and experiments. The notation used in different references was unified, and basic concepts and elements about the topic were presented. New explanations, examples and considerations were also introduced. A software application to emulate a control loop running over LonWorks/IP networks has been developed. This tool can be used for tests and even as a start point for real projects. The main contributions of this work are: identification of quantitative network parameters for establishement of requirements of LonWorks/IP applications; creation of a software application to run a control loop over LonWorks/IP networks; analysis of current knowledge about LonWorks technology; and identification of relationships between technological solutions for LonWorks/IP interconnection and categories of application of the technology.

Keywords: Control networks, LonWorks, LonTalk, Control over IP, EIA709, EIA852

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACK	Acknowledge
ANSI	American National Standards Institute
API	Application Programming Interface
ARPA	Advanced Reserarch Projects Agency
ARPANET	Advanced Reserarch Projects Agency Network
BI	Business Intelligence
bps	bits por segundo
CAN	Controller Area Network
CCS	Centralized Control System
CLP	Controlador Lógico Programável
CoS	Classification of Service
CRC	Cyclic Redundancy Check
CRM	Customer Relationship Management
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Access
DCS	Distributed Control System
DDE	Dynamic Data Exchange
DNS	Domain Name System
DoD	Department of Defense
EIA	Electronic Industries Alliance
ERP	Enterprise Resource Planning
E/S	Entradas e Saídas
FAN	Field Area Network
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
Hz	Hertz
ID	Identifier

IHM	Interface Homem-Máquina
IP	Internet Protocol
ISO	International Organization for Standardization
ITU	International Telecommunications Union
kbps	1000 bits por segundo
LAN	Local Area Network
LNS	LonWorks Network Services
LON	Local Operation Network
MAC	Medium Access Control
Mbps	1.000.000 bits por segundo
MAN	Metropolitan Area Network
MIB	Management Information Base
NI	Network Interface
NBCS	Network Based Control Systems
NCS	Networked Control Systems
NCP	Network Control Protocol
NI	Network Interface
NIC	Network Interface Card
NV	Network Variable
OID	Object Identifier
OSI	Open System Interconnection
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCLTA	PC LonTalk Adapter
PCMCIA	Personal Computer Memory Card International Association
PDU	Protocol Data Unit
POP	Post Office Protocol
PPP	Point-to-Point Protocol
QoC	Quality of Control
QoS	Quality of Service

RLP	RemoteLon Protocol
RPC	Remote Procedure Call
RTT	Round Trip Time
SCADA	Supervisory Control and Data Acquisition
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SNVT	Standard Network Variable Type
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TI	Tecnologia da Informação
UCP	Unidade Central de Processamento
UDP	User Datagram Protocol
UDDI	Universal Description, Discovery and Integration
USB	Universal Serial Bus
VPN	Virtual Private Network
WAN	Wide Area Network
WSDL	Web Services Description Language
WWW	World Wide Web
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS E MOTIVAÇÃO

Sistemas de automação industrial têm evoluído nos últimos anos. Seus desenvolvedores vêm buscando criar uma nova geração de arquitetura de controle com a intenção de obter maior flexibilidade e produtividade (MAHALIK; LEE, 2002). O cenário tradicional com respeito à arquitetura de controle industrial é baseado em controle centralizado, isto é, o processamento da automação da planta é realizado por uma ou poucas unidades centralizadas. Os dispositivos de campo (sensores e atuadores) possuem algum tipo de ligação ponto-a-ponto com o controlador centralizado (Figura 1). A tendência é que redes de dispositivos inteligentes que se comunicam digitalmente através de protocolos apropriados substituam os tradicionais sistemas de processamento centralizado com comunicação analógica baseada em sinais de corrente de 4 - 20 mA (MAHALIK; LEE, 2002) (RAJI, 2002).

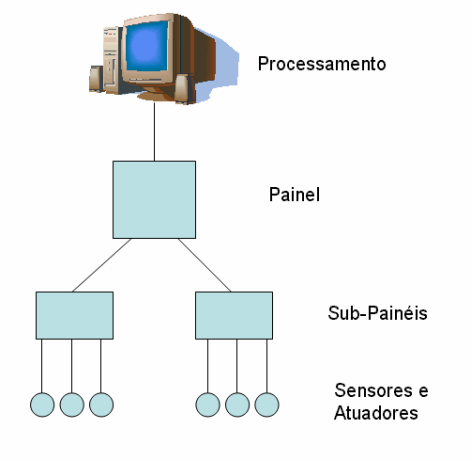


Figura 1. Sistema de controle tradicional centralizado.
Extraído de Canovas e Chermont (2003).

O conceito de redes de controle pode ser explicado através de uma comparação com redes de dados. Redes de dados se destinam a interconectar computadores de modo que eles possam trocar dados. Basicamente, computadores são conectados a um ou mais meios físicos interligados através de roteadores, sobre os quais trafegam mensagens codificadas de acordo com uma pilha de protocolos de comunicação. No caso das redes de dados, o padrão *de facto*

para protocolos de comunicação é a arquitetura TCP/IP, sendo as redes Ethernet um dos meios físicos mais usuais. Redes de dados são otimizadas para trafegar grandes quantidades de informação e normalmente certos atrasos na entrega de pacotes são aceitáveis. As redes de controle, por sua vez, servem para interconectar dispositivos de controle (em geral, sensores, atuadores e unidades de processamento), e possuem requisitos diferentes das redes de dados, tais como: custo, tamanho, facilidade de instalação, tempo de resposta, imunidade a ruído, etc. Sendo assim, o projeto dessas redes considera outros fatores de modo a desenvolvê-la propriamente para atender esses requisitos específicos de maneira otimizada.

LonWorks® é uma tecnologia de especificação aberta baseada no protocolo definido pela norma americana ANSI/EIA 709.1, a qual possibilita a criação de redes de controle descentralizadas. Esta tecnologia foi criada pela empresa Echelon (ECHELON, 1999) (ECHELON, 2005e), que a tornou aberta no final da década de 1990. LonWorks é uma implementação do conceito de redes de controle, ou *fieldbus*, com processamento distribuído. De acordo com Bbdsoft (2005), *fieldbus* é um termo genérico usado para descrever protocolos de comunicação de sistemas de controle e instrumentação de campo. Apesar de alguns protocolos e tecnologias terem sido padronizados para utilização em sistemas de controle, a indústria de uma maneira geral ainda não concordou ou adotou um padrão único para *fieldbus*, como seria o caso do TCP/IP para redes de dados.

LonWorks é um padrão aberto para soluções em automação industrial, predial, residencial, sistemas de transporte e redes de controle gerais (ECHELON, 1999) (ECHELON, 2005b). Apesar de não existir um padrão único e bem definido para *fieldbus*, esta tecnologia vem sendo considerada bastante promissora, podendo vir a se tornar este padrão de grande adoção (KIM; CHO; PARK, 2000). Inúmeras instalações baseadas em LonWorks podem ser encontradas ao redor do mundo. Exemplos de aplicações são apresentados em Alves Filho, Cugnasca e Dias (2004), Mahalik e Lee (2002) e Tse, Chan e Lai (2003). Apesar de ter sido desenvolvida como uma solução completa também para aplicações industriais e em transportes, foi na área de automação residencial e principalmente predial que LonWorks ganhou destaque. A diferença entre a automação residencial e predial é que a primeira se refere a residências, enquanto a última diz respeito a edifícios e ambientes corporativos.

Segundo informações encontradas no site da consultoria americana ARC Advisory Group (ARCWEB, 2002) sobre a movimentação financeira do mercado de automação predial no mundo (baseado no estudo “Building Automation Systems Worldwide Outlook” de 2002), pôde-se observar o crescimento do volume financeiro do setor entre 2001 e 2002 (de US$ 19,1 bi para US$ 19,6 bi), com previsões até 2006 (chegando à US$ 24.4 bi). No Brasil, essa tendência de crescimento se confirma. Segundo Vieira (2002), o mercado de automação

predial no Brasil cresce no ritmo de 10% a 15% ao ano e movimenta perto de US$ 220 milhões.

Sendo uma tecnologia que apresenta produtos e ferramentas que contemplam todos os níveis da automação, desde a instrumentação (sensores e atuadores) até sistemas de informação, a tecnologia LonWorks pode ser encontrada em sistemas de ventilação, ar condicionado, iluminação, segurança, elevadores, detecção de incêndio, controle de acesso, monitoração de energia, etc. (ECHELON, 2005f) (SNOONIAN, 2003).

Em 7 de Fevereiro de 2005, o protocolo de comunicação das redes LonWorks foi oficialmente adotado como padrão oficial europeu para as redes de controle prediais (padrão prEN 14908) (ECHELON, 2005a).

Em outra frente, o surgimento da Internet e seu crescimento explosivo nas décadas de 1990 e 2000 vêm possibilitando e oferecendo inúmeras aplicações cada vez mais comuns e fundamentais no cotidiano de seus usuários. De acordo com Raji (2002), as aplicações mais utilizadas da Internet, a World Wide Web (WWW) e o correio eletrônico (E-Mail) constituem uma maneira econômica para conectar pessoas a pessoas através da publicação de conteúdo e troca de informação.

Na última década, foi possível observar a tendência de conectar as redes locais das empresas, ou *Local Area Networks* (LANs) à Internet. Isso possibilitou mais aplicações e maior agilidade nos negócios sob o ponto de vista de integração de sistemas, automação de transações, eficiência nas comunicações, troca eletrônica de documentos, etc. Com o surgimento das redes de controle, a seguinte pergunta aparece de maneira natural: por que não também conectá-las à Internet (ou intranets) de modo que a informação relativa a dados e controle possa circular através de uma área globalizada?

Assim como as LANs foram integradas à Internet, naturalmente surge a tendência de conectar as redes de controle às primeiras. Raji (2002) chama as redes de controle locais de *infranets* (Figura 2).

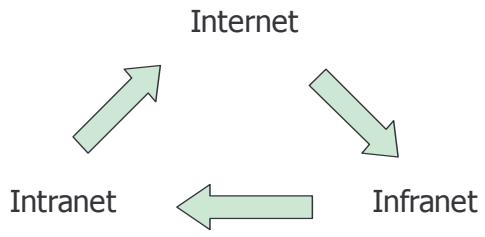


Figura 2. A infra-estrutura completa da informação.
Extraído de Raji (2002).

Uma outra designação de uma *infranet* é *Field Area Network* (FAN), utilizada em Kunes e Sauter (2001).

É preciso considerar que as redes locais e a Internet são redes de dados com requisitos semelhantes. Além disso, o atual padrão de protocolo de comunicação em uso, a arquitetura TCP/IP, é o mesmo para ambas, o que torna sua interligação muito mais fácil. Integrá-las com redes de controle, que apresenta requisitos diferentes, requer que muitos cuidados sejam tomados.

1.2 OBJETIVO

Dadas as razões e tendências apresentadas na seção 1.1, escolheu-se como referência de rede de controle a tecnologia LonWorks.

Neste trabalho, apresenta-se um estudo sobre a integração entre redes de controle LonWorks e redes de dados baseadas em IP.

Ao analisar quais os ganhos e aplicações que poderiam ser obtidos ao interligar redes de controle LonWorks com redes IP, é possível verificar logo de início que toda a infra-estrutura já existente para redes de dados (intranets e Internet) poderia ser aproveitada como um meio de comunicação entre redes de controle distantes. Além desta, outras aplicações também passariam a ser possíveis. Pode-se dividi-las em quatro categorias básicas:

1. monitorar ou enviar comandos para a rede LonWorks através de uma estação ligada à rede de dados (sistema supervisório);
2. gerenciar e configurar a rede LonWorks através de uma estação ligada à rede de dados;
3. utilizar a rede de dados como meio para interconectar redes LonWorks geograficamente distantes para aplicações sem requisitos fortes de tempo real;
4. utilizar a rede de dados como meio para interconectar redes LonWorks geograficamente distantes para aplicações com requisitos fortes de tempo real, tendo em vista principalmente a implementação de laços de controle onde o controlador e a planta não estão conectados diretamente ou na mesma rede.

Observa-se que todos os casos acima estendem a aplicação das redes de dados IP não só para conectar pessoas a pessoas, mas também pessoas a dispositivos e dispositivos a

dispositivos (RAJI, 2002). Neste trabalho, a integração entre redes de controle LonWorks e redes de dados IP é abordada com foco nessas quatro categorias de aplicação.

1.3 CONTEÚDO E ORGANIZAÇÃO

Esta dissertação está dividida em três grandes tópicos: redes IP, redes LonWorks e a integração entre as mesmas.

O Capítulo 2 apresenta uma breve revisão sobre redes IP. Considerando a grande disponibilidade de literatura sobre o assunto, apenas uma curta revisão será apresentada, visando embasar os demais capítulos.

O Capítulo 3 provê uma introdução às redes LonWorks, mostrando onde as mesmas se inserem no contexto da automação e apresentando os conceitos básicos relativos a esta tecnologia.

O Capítulo 4 apresenta uma análise das redes LonWorks com maior profundidade. Apesar de a especificação ANSI/EIA/709.1 ser aberta, não é fácil encontrar uma literatura bem consolidada sobre este protocolo. A maior parte da informação técnica foi encontrada em manuais da Echelon, a qual foi complementada pelos conhecimentos adquiridos através da experimentação do autor em trabalhos acadêmicos anteriores (CANOVAS; CHERMONT, 2003) e atividades profissionais.

O Capítulo 5 apresenta os tipos de abordagens existentes para implementar interconexão entre redes LonWorks e IP, oferecendo bases para o entendimento e a avaliação de como cada abordagem atende cada tipo de aplicação.

O Capítulo 6 introduz o conceito de sistemas de tempo real e estabelece uma lista de parâmetros de qualidade de serviço, ou *Quality of Service* (QoS), fundamentais para identificar requisitos de uma aplicação e para medir se uma determinada infra-estrutura de rede LonWorks / IP os atende.

O Capítulo 7 trata efetivamente da interligação entre as redes LonWorks e IP, passando por cada um dos quatro tópicos citados na seção 1.2. As principais abordagens de soluções serão discutidas no trabalho, elucidando como os problemas são resolvidos e quais os fatores a serem considerados para o sucesso da aplicação em questão. Além dos aspectos teóricos, visa-se também a aplicabilidade prática, citando-se ferramentas e equipamentos existentes no mercado. Um experimento relacionado a um laço de controle executado sobre redes LonWorks e IP é realizado e apresentado.

O Capítulo 8 termina o trabalho apresentando as considerações finais, discussões e conclusões.

Como informações adicionais, o Apêndice A oferece alguns detalhes básicos sobre a arquitetura TCP/IP, tais como formatos de pacotes. O Apêndice B provê detalhes equivalentes para o protocolo LonTalk (servindo como um aprofundamento do Capítulo 4).

2 **REDES IP**

2.1 INTRODUÇÃO

Redes de comunicação podem ser divididas em dois grupos: redes de comutação de circuitos (orientadas a conexão) e redes de comutação de pacotes (não orientadas a conexão, ou, do inglês, *connectionless*) (TANENBAUM, 1996).

Redes de comutação de circuitos operam através do fechamento de uma conexão dedicada entre dois pontos. O melhor exemplo para este tipo é a rede telefônica. Ao fazer uma chamada, o telefone de origem estabelece uma conexão com uma central local, que, por sua vez se conecta a uma central remota via linhas-tronco, e, por fim, a conexão é levada até o telefone de destino. A voz é transmitida entre os dois telefones, seja através de técnicas analógicas ou digitais (essas últimas, mais modernas), sendo garantida a banda necessária para a entrega do sinal na outra ponta. A principal vantagem deste método está na garantia da banda. Uma desvantagem está no custo, que é fixo para o circuito, e costuma ser cobrado mesmo se as duas pontas não participam de conversações (COMER, 2000).

Redes de comutação de pacotes possuem seu grande nicho de aplicação nas redes de computadores, e utilizam uma abordagem diferente das redes de comutação de circuitos. Na comutação de pacotes, a informação a ser transmitida entre dois pontos é quebrada em pequenos pedaços denominados pacotes. Cada pacote carrega informação suficiente para identificar a qual elemento da rede o mesmo deve ser entregue, além da própria informação que se deseja transmitir. O canal de comunicação é compartilhado por todos os computadores conectados à rede. Como normalmente não são utilizadas freqüências diferentes para modular o sinal correspondente a cada transmissão de pacote, todos eles compartilham a mesma banda. Desta maneira, um pacote não pode ser enviado ao canal de comunicação enquanto houver um outro pacote trafegando. Quando isto ocorre, diz-se que houve uma colisão. As tecnologias físicas de redes de computadores devem ser capazes de lidar com este problema.

Uma grande vantagem das redes de comutação de pacotes é que múltiplas comunicações entre computadores podem ocorrer concorrentemente, já que os "pedaços" de cada comunicação, isto é, os pacotes, são intercalados no mesmo meio físico. Pode-se perceber, por outro lado, que isto está relacionado a uma desvantagem: quando o tráfego na rede está elevado, um certo par de computadores que deseja se comunicar enxerga uma menor capacidade efetiva da rede. Apesar desta desvantagem, as redes de comutação de pacotes se tornaram extremamente populares devido a duas principais motivações: custo e desempenho

(COMER, 2000). Como a banda é compartilhada por diversos computadores, não havendo a necessidade de se manter conexões dedicadas, o custo é mantido baixo. Como as tecnologias de hardware e meios físicos deste tipo de rede normalmente são de grande velocidade, o compartilhamento da banda não costuma ser um problema quando se fala em comunicação de dados.

É relevante considerar duas observações fundamentais sobre o projeto de sistemas de comunicação, conforme destaca Comer (2000):

1. nenhuma tecnologia de hardware de rede pode satisfazer todos os requisitos por si só;
2. usuários desejam interconexão universal.

A primeira observação pode ser facilmente verificada na prática, dado que estão disponíveis no mercado inúmeras tecnologias de hardware para a implementação de redes de computadores. Cada tecnologia considera requisitos específicos, como as redes locais, redes de longa distância, redes sem fio, etc. Além do ponto de vista técnico, cada tecnologia também é concebida para atender parâmetros de custo de onde serão aplicadas. Inicialmente, isso pode parecer conflitante com a segunda observação, a qual diz que os usuários desejam sistemas de comunicação que não são limitados pelas fronteiras das redes individuais que se pode construir com uma certa tecnologia.

O objetivo que se pode extrair dessas observações é o de construir um sistema de interconexão de redes unificado que suporta serviços de comunicação universais. Em cada rede, computadores irão utilizar a tecnologia física sobre a qual estão conectados. Novas camadas de software, inseridas nesses computadores, irão abstrair os detalhes de baixo nível a respeito de como a comunicação ocorre e oferecer às aplicações uma interface comum de utilização, permitindo que redes de diferentes tecnologias, quando interconectadas, se apresentem em alto nível como uma única rede maior para os usuários.

Ainda de acordo com Comer (2000), esta idéia segue uma prática de projeto freqüente em sistemas de comunicações: desenvolvedores imaginam um padrão comum de alto nível e, a partir das tecnologias de baixo nível já existentes, constroem camadas sucessivas de software até obter um sistema que implementa, por completo, as interfaces de alto nível imaginadas inicialmente.

Nesse contexto, surgem as redes IP. O protocolo IP está um nível acima da tecnologia física da rede, independentemente de qual seja, e permite um sistema unificado de endereçamento e entrega de pacotes do ponto de vista do usuário pois esconde os mecanismos

das camadas de baixo. Dessa maneira, dois aplicativos sendo executados em computadores conectados a redes de tecnologias distintas, porém interligadas utilizando o protocolo IP, se comunicam da mesma maneira que seria se esta comunicação via IP estivesse ocorrendo dentro da mesma rede. Observa-se, portanto, que a interconexão das redes é transparente. Este conceito será melhor detalhado na seção 2.2.

2.2 ENDEREÇAMENTO E ROTEAMENTO

Cada pacote IP carrega em seu cabeçalho o endereço do computador ao qual está destinado. Um endereço é composto por quatro bytes e é denotado usualmente por quatro números decimais separados por ponto, cada um representando um dos bytes. Exemplos: 192.168.0.1, 10.0.100.45, etc. Esses endereços são chamados de endereços IP; e, quando o contexto não deixa dúvidas de que se está referindo a um endereço, eles são chamados apenas de IP. No entanto, não se deve confundi-los com o protocolo IP. Redes IP são as redes que se baseiam no protocolo IP acima dos protocolos dependentes da tecnologia física. A cada computador ou dispositivo conectado a uma rede IP, é atribuído um endereço IP único.

Uma parte de cada endereço IP designa o endereço da rede que contém o computador de destino. A outra parte identifica o computador dentro da rede. Assim, é possível interpretar cada endereço IP como um par ordenado (*id_rede*, *id_host*), onde *id_rede* é o identificador da rede e *id_host* é o identificador do computador na rede. No entanto, como cada endereço IP é composto por quatro bytes, então deve existir alguma forma de mapeá-los no par (*id_rede*, *id_host*).

Originalmente, esse mapeamento dependia da classe do endereço IP. Os endereços IP são agrupados em classes conforme os primeiros bits de seu primeiro byte, de acordo com a especificação da Figura 3.

As classes de IP utilizadas para endereçamento *unicast*, isto é, que endereçam um único computador em todo o conjunto de redes interconectadas, são as classes A, B e C. A classe D serve para endereçamento *multicast* (mais de um computador de destino endereçados por um mesmo IP), e a classe E está reservada para uso futuro. Essas duas últimas não serão consideradas aqui.

Escrevendo os endereços através da sua notação decimal separada por pontos, é possível mapear os bytes de um endereço IP no par (*id_rede*, *id_host*). Considerando que o IP é denotado por *w.x.y.z*, onde *w*, *x*, *y* e *z* são inteiros de 8 bits sem sinal arbitrários, tem-se a Tabela 1.

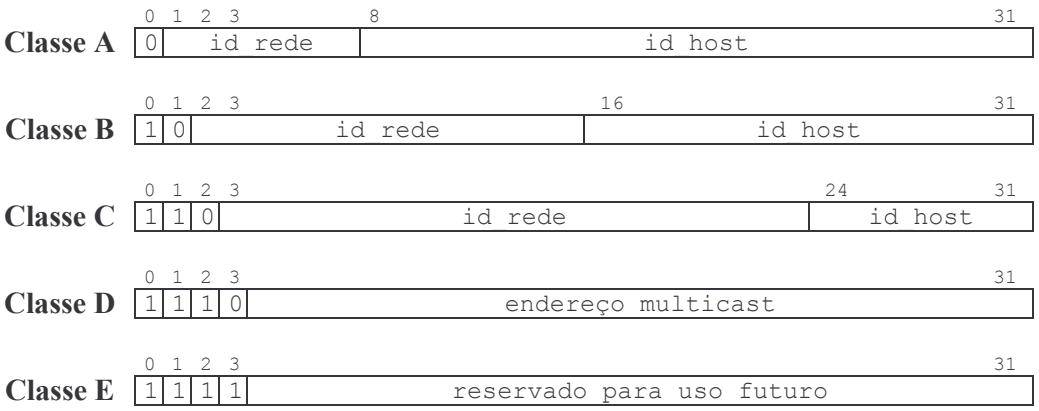


Figura 3. Classes de endereços IP.
Extraído de Comer (2000).

Na verdade, as variáveis *w*, *x*, *y* e *z* da Tabela 1 apresentam algumas restrições. Por exemplo, a situação em que *id_host* é nulo corresponde a um endereço IP que se refere à rede identificada por *id_rede* por completo, e não pode ser utilizada no endereçamento IP. Por outro lado, para fazer um *broadcast* em uma certa rede, utiliza-se um *id_rede* válido e o *id_host* com todos os seus bits iguais a um. Há também faixas de IP reservadas para uso em redes locais, cujos IPs podem repetir em LANs distintas, além de outras restrições. No entanto, faz parte deste trabalho apresentar apenas uma breve revisão do mecanismo de endereçamento de pacotes utilizado no protocolo IP. Maiores detalhes podem ser encontrados em outras referências como Comer (2000) e Tanembaum (1996).

Tabela 1 – Faixas de endereços IP por classes.

Classe	Menor Endereço	Maior Endereço	*id_rede*	*id_host*
A	1.0.0.0	126.0.0.0	*w*	*x.y.z*
B	128.1.0.0	191.255.0.0	*w.x*	*y.z*
C	192.0.1.0	223.255.255.0	*w.x.y*	*z*

Observa-se que, de maneira geral, o conjunto de IPs da classe A pode endereçar poucas redes com muitos computadores, enquanto o da classe B é capaz de identificar um número médio de redes com um número médio de computadores. O conjunto da classe C, por sua vez, endereça muitas redes com poucos computadores.

Quando os projetistas criaram a divisão dos endereços IP em classes, eles não previram que a Internet, a rede mundial de computadores baseada no protocolo IP, iria crescer

aos níveis atuais. Um dos problemas causados por este crescimento é a exaustão dos endereços IP, que não seriam suficientes para acomodar o número de redes e computadores existentes. Comer (2000) afirma que, em particular, os prefixos para *id_rede* dos endereços IP de classe B são insuficientes para acomodar todas as redes de tamanho médio existentes na Internet. Como uma maneira de flexibilizar o agrupamento em classes e permitir que um mesmo prefixo de endereço IP servisse para identificar mais de uma rede, foram criados diversos mecanismos. O mais usado atualmente é o endereçamento de sub-rede.

A título de exemplo, uma organização dona de um *id_rede* de endereço IP classe B pode utilizar este mesmo *id_rede* para suas diversas redes físicas internas. Do ponto de vista da Internet, ela possui uma única rede grande, mas internamente suas redes utilizam parte do espaço do que seria o *id_host* da classe B para identificar a rede interna dentro da grande rede observada externamente.

Supondo que a organização fosse dona do endereço de rede 128.10.0.0, ela poderia atribuir 128.10.1.0, 128.10.2.0 e 128.10.3.0 a três redes internas. O IP 128.10.1.1 e 128.10.3.5 corresponderiam a, respectivamente, computadores da primeira e da terceira rede. A especificação de que o terceiro byte é um identificador de rede é feita através do que se chama de máscara de sub-rede (*subnet mask*). De acordo com o agrupamento original de classes, esse terceiro byte faz parte do *id_host*, mas nesse caso ele é interpretado como parte do *id_rede* das redes internas.

Uma máscara de sub-rede é um inteiro de quatro bytes. Quando se faz uma operação AND binária entre um endereço IP e uma máscara de sub-rede, obtém-se o endereço da rede interna ao qual o IP pertence. Na Figura 4, correspondente ao exemplo acima, está-se aplicando a máscara de sub-rede 255.255.255.0 em um endereço IP de um computador da organização. Essa máscara de sub-rede especifica que o terceiro byte dos endereços IP faz parte do *id_rede*.

	Notação decimal	**Notação binária**
IP do computador:	128.10.3.5	10000000 00001010 00000011 00000101
Máscara de sub-rede:	255.255.255.0	11111111 11111111 11111111 00000000
IP **AND** Máscara:	128.10.3.0	10000000 00001010 00000011 00000000

Considerado pela Internet

Endereço da rede interna

Figura 4. Exemplo de aplicação de máscara de sub-rede.

Até aqui, apresentou-se como conectar computadores em uma mesma rede e como eles se endereçam de maneira transparente em redes diferentes do ponto de vista de software utilizando o protocolo IP. Convém mencionar agora como é feita a interconexão física dessas redes de tecnologias diferentes. Essa interconexão é realizada através de um computador especial denominado roteador (*router*) ou *gateway*. Um roteador possui interfaces de comunicação compatíveis com cada tecnologia física de rede às quais está conectado.

Os termos roteador e *gateway* são comumente utilizados para designar uma mesma categoria de dispositivos destinados à interconexão de redes. No entanto, uma definição mais precisa estabelece que um roteador interliga redes baseadas na mesma tecnologia física, enquanto os *gateways* interligam redes de tecnologias distintas (TANENBAUM, 1996). Neste Capítulo, esses termos não serão rigorosamente diferenciados.

A Figura 5 apresenta um roteador interconectando duas redes.

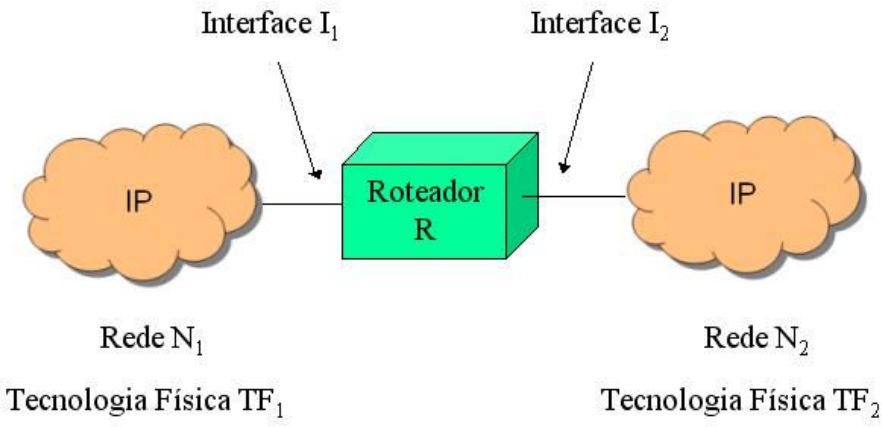


Figura 5. Exemplo de duas redes interconectadas por um roteador.

O roteador R interconecta as redes N_1 e N_2. Fisicamente, ele possui as interfaces de comunicação I_1 e I_2, compatíveis respectivamente com as tecnologias físicas TF_1 e TF_2. Quando R recebe um certo pacote IP através da interface I_1, encapsulado em quadros especificados pela tecnologia TF_1, ele verifica se o endereço de destino está na rede N_2. Em caso positivo, o pacote IP é extraído e reencapsulado em um quadro compatível com TF_2, sendo então lançado à rede N_2 através de I_2.

Se R identifica que o destino do pacote IP está na própria rede onde foi originado, então este não o retransmite pela outra interface.

TF_1 e TF_2 não precisam ser tecnologias distintas. Quando $TF_1 = TF_2$, o roteador está sendo utilizado para interconectar duas redes baseadas em uma mesma tecnologia.

O mecanismo pelo qual R decide se o pacote proveniente de uma interface deve ser retransmitido à outra requer uma descrição bastante detalhada sobre diversos conceitos envolvendo tabelas, tipos e protocolos de roteamento. Esses detalhes fogem ao escopo deste trabalho. Além disso, um roteador pode interligar mais do que duas redes, possuindo três, quatro, cinco ou até mais interfaces de comunicação. No contexto deste trabalho não é necessário considerar os detalhes de funcionamento interno de um roteador. Apresentou-se até então, de forma geral, como os pacotes IP chegam corretamente às redes dos computadores aos quais estão destinados. Uma vez na rede correta, o computador de destino obtém os pacotes IP endereçados a si. Na prática, há ainda alguns outros conceitos que podem ser considerados, como a obtenção do endereço físico da interface de rede do computador de destino em função do endereço IP. Porém, para a abstração empregada neste trabalho, é suficiente considerar que quando um pacote IP chega à sua rede de destino, o computador de destino é capaz de obtê-lo.

2.3 TCP E UDP

Conforme apresentado, o protocolo IP abstrai a tecnologia física de rede empregada, oferecendo uma interface de alto nível para as aplicações de software e tornando transparente os meios pelos quais redes distintas, porém interconectadas, utilizam. No entanto, como mencionado, a informação é fracionada em pedaços (pacotes) para ser transmitida entre dois pontos. Cada pacote é roteado de sua origem até seu destino, podendo passar através de diversos roteadores e redes. Alguns problemas que podem ocorrer nesse caminho, tais como falha em roteadores, podem causar perdas de pacotes. Portanto, simplesmente quebrar a informação em pacotes e enviá-las através do protocolo IP não é um método confiável de comunicação.

Um outro problema está associado ao fato de existirem redes que roteiam pacotes dinamicamente, ou seja, entre uma origem e um destino, pacotes diferentes podem passar por rotas diferentes. Isto é possível quando há mais de um caminho entre dois pontos, conforme mostra o exemplo da Figura 6. Se uma das rotas está mais congestionada que as outras, pacotes que passam por ela podem chegar ao seu destino após os pacotes que passam pela rede mais livre, mesmo que tenham sido disparados anteriormente. Nesse caso, eles chegam fora da ordem correta.

Esses problemas devem ser considerados a ponto de não causarem falhas graves nos aplicativos de software que contam com a infra-estrutura de rede IP para comunicação.

Como os aplicativos de software normalmente precisam trocar grandes quantidades de dados, seria necessário implementar uma série de rotinas que verificassem se os pacotes IP recebidos estão na ordem correta, ou mesmo se ficou faltando algum pacote que se perdeu na rede. Implementar algo deste tipo em todo aplicativo aumenta o custo e o tempo de desenvolvimento, além de tornar a manutenção mais difícil. Seria interessante, portanto, se houvesse alguma camada de software pronta e padronizada entre a aplicação e o protocolo IP, a qual tornasse transparente ao programador a confiabilidade na entrega da informação. Esta camada de software garantiria a confiabilidade através da detecção e tratamento dos problemas de transmissão de pacotes que podem ocorrer. Se um pacote não chegar ao seu destino, ele deve ser enviado novamente. Se pacotes chegam fora de ordem, eles devem ser reordenados. Se um pacote atrasado é dado como não recebido e o mesmo é reenviado, pode ocorrer uma duplicidade na recepção quando o pacote atrasado (que foi dado como perdido) efetivamente chegar. Para suprir esta necessidade, foi criado o *Transmission Control Protocol* (TCP), protocolo que isola a aplicação dos detalhes da transmissão correta de pacotes.

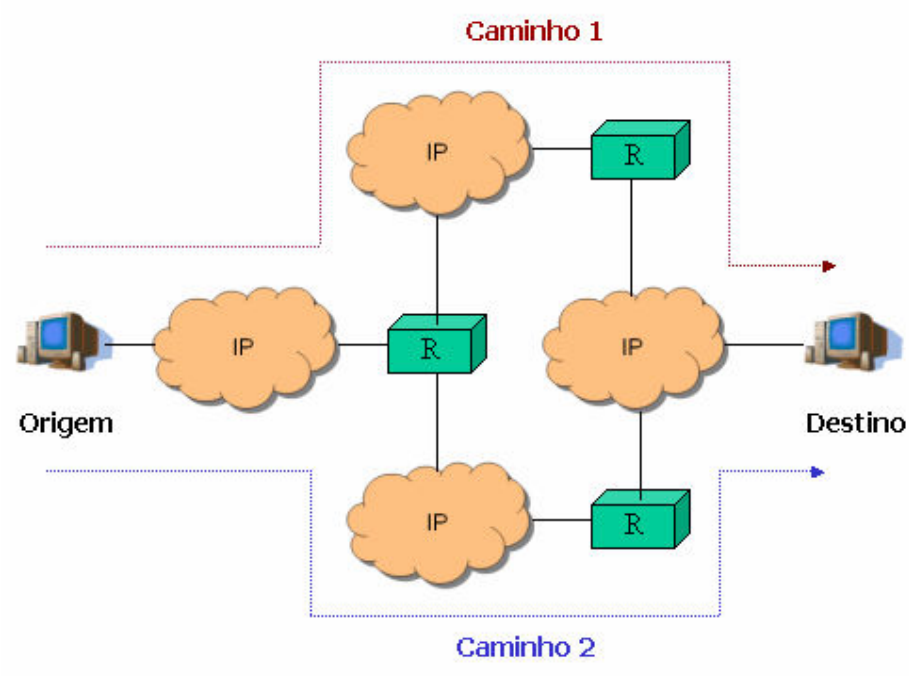


Figura 6. Exemplo de duas rotas entre uma origem e um destino.

Uma aplicação que utiliza o protocolo TCP é capaz de enviar um fluxo de bytes (*stream*) a outra aplicação em outro computador sem se preocupar com a quebra em pacotes e nem com o gerenciamento de sua correta entrega. Isso é tarefa do próprio TCP, que, por sua vez, utiliza a camada IP. Assim, as aplicações baseadas em TCP nunca acessam a camada IP diretamente.

O TCP é orientado a conexão. Ao contrário de uma rede de comutação de circuitos, o conceito de conexão TCP não é baseado em um fechamento físico de um circuito entre a origem e o destino (até porque ele utiliza o IP, que é completamente atrelado à comutação de pacotes). Uma conexão TCP é um conceito lógico simplesmente constituído por um quarteto ordenado (*ip_origem*, *porta_origem*, *ip_destino*, *porta_destino*).

Quando um aplicativo deseja transmitir informações a uma aplicação em um computador remoto, estabelece-se uma conexão TCP. O computador remoto é consultado se é possível estabelecer uma conexão com um certo aplicativo, que por sua vez é identificado por número de 16 bits sem sinal denominado porta. Se a requisição for aceita, a camada de software que implementa o protocolo TCP nas duas pontas irá registrar a existência de uma conexão estabelecida e identificada pelo quarteto (*ip_origem*, *porta_origem*, *ip_destino*, *porta_destino*), onde *ip_origem* e *ip_destino* são os endereços IP da origem e destino, respectivamente, e *porta_origem* e *porta_destino* identificam as aplicações que participam da conexão no computador de origem e destino, também respectivamente.

Enquanto a conexão TCP estiver estabelecida, os aplicativos podem trocar fluxos de bytes de modo *full-duplex*. O protocolo TCP provê aos aplicativos um meio de comunicação confiável orientado a conexões construído sobre uma estrutura não confiável baseada na comutação de pacotes.

Um computador pode participar de diversas conexões ao mesmo tempo. Quando a transmissão de fluxos de dados de diferentes conexões precisa ocorrer concorrentemente, o TCP cuida de intercalar o envio de pacotes de cada conexão na rede. Isso fica transparente para o aplicativo, que enxergará apenas a sua conexão. O fato de existirem identificadores de aplicações (portas) permite também que diversas conexões sejam estabelecidas ao mesmo tempo entre diferentes aplicativos do mesmo par de computadores.

Para que o TCP funcione, ele implementa mecanismos de retransmissão de pacotes e confirmação de recebimento, que por sua vez torna necessária a adição de informações de controle aos pacotes. Mesmo existindo por uma razão nobre, o processamento de controle e a carga de dados extra (também conhecida como *overhead*) pode ser prejudicial em aplicações onde o tempo de entrega é fundamental. Um exemplo típico é a transmissão de voz em tempo real, onde o tempo envolvido na espera de uma confirmação de recebimento de pacote e sua

retransmissão pode ser prejudicial à aplicação. Ao mesmo tempo, a transmissão de voz tolera a perda de alguns pacotes sem prejuízo ao ouvinte, e, portanto, o protocolo TCP não seria indicado para este tipo de aplicação. Nesse caso, o sinal de voz digitalizado poderia ser colocado diretamente dentro dos pacotes IP. Por outro lado, o protocolo IP não implementa o conceito de portas para identificar diferentes aplicativos e, por isso, não seria possível realizar outras conexões TCP e nem mesmo outros fluxos de voz concorrentemente.

Para resolver este problema, foi criado o *User Datagram Protocol* (UDP). Basicamente, o protocolo UDP é uma camada acima do IP que apenas repassa os pacotes para ele, adicionando a implementação do conceito de portas. Ao contrário de uma aplicação baseada em TCP, uma aplicação que usa o UDP deve ser responsável por realizar a quebra em pacotes e gerenciar por si só, conforme conveniente, a entrega correta da informação na outra ponta.

2.4 MODELOS TCP/IP E OSI

Como pôde ser observado a partir das seções anteriores, os desenvolvedores construíram um esquema de interconexão de redes baseado em camadas de protocolos. No nível mais baixo, tem-se a tecnologia física, a qual é constituída pelo meio físico de comunicação (camada física) e pela maneira como os bits são organizados em quadros, também chamados de *frames*, para serem transmitidos sobre esse meio físico (camada de enlace). Acima deste nível, aparece o protocolo IP, que abstrai a camada física e trata do endereçamento e roteamento de pacotes (camada de rede). Em um nível mais acima, tem-se o TCP e o UDP, que são responsáveis por abstrair a camada IP e fornecer maneiras de transporte de fluxos de dados através de, respectivamente, um meio confiável orientado a conexões e um meio não confiável não orientado a conexões (camada de transporte).

As aplicações acessam diretamente apenas o TCP e o UDP, que tornam transparentes todas as camadas abaixo. A Figura 7 oferece uma ilustração deste esquema em camadas. Este modelo é conhecido como o modelo TCP/IP.

Os fluxos de dados transmitidos por TCP ou UDP não possuem formatos definidos, sendo de livre escolha da aplicação em questão. Normalmente, cada aplicação utiliza seu próprio protocolo de comunicação. Esse é o caso das duas principais aplicações da Internet: a *web* e o correio eletrônico. A *web* é executada com base no *Hyptertext Transfer Protocol* (HTTP), protocolo que funciona sobre o TCP. Como a responsabilidade da entrega confiável dos dados é do TCP, o HTTP acaba por ser um protocolo relativamente simples que se preocupa apenas com a solicitação e fornecimento de *home-pages*, sem se preocupar com o

modo pelo qual essa informação trafega na rede e com o modo através do qual a garantia de entrega é oferecida.

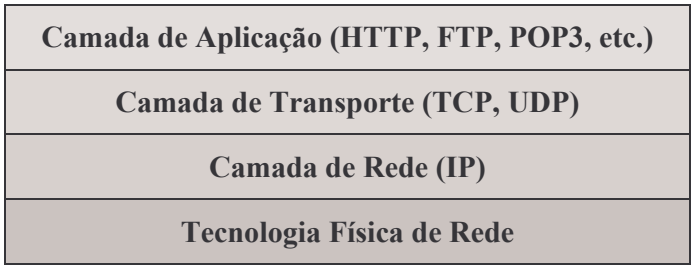


Figura 7. Modelo TCP/IP.

Um outro modelo de referência bastante conhecido é o modelo *Open Systems Interconnection* (OSI), desenvolvido pela *International Standards Organization* (ISO) como um passo na direção da padronização de pilhas de protocolos de comunicação (TANENBAUM, 1996). Ao contrário do modelo TCP/IP, que especifica os protocolos IP e TCP respectivamente para as camadas de rede e transporte, o modelo OSI apenas especifica as responsabilidades de cada camada, permitindo que possa ser implementada por qualquer protocolo que siga as especificações. O modelo OSI é constituído por sete camadas, ao contrário do TCP/IP, que é composto por quatro. No entanto, há um mapeamento entre as camadas do modelo OSI e do TCP/IP. A Tabela 2 apresenta um resumo de cada camada do modelo OSI, enquanto a Figura 8 mostra as sete camadas do modelo OSI mapeadas no TCP/IP.

2.5 A INTERNET

Na década de 1960, em pleno período de Guerra Fria, os militares americanos identificaram a necessidade de uma rede de comunicações que pudesse sobreviver a um ataque nuclear. O sistema telefônico era considerado muito vulnerável, já que a perda de uma linha ou central encerraria todas as conversações que passassem por elas e particionaria o sistema (TANENBAUM, 1996).

Tabela 2 – Modelo OSI de arquitetura de protocolos de rede.

	Camada OSI	**Objetivo**	**Serviços oferecidos**
7	Aplicação	Compatibilidade de aplicação	Objetos e tipos padrões, propriedades de configurações, transferência de arquivos e serviços de rede.
6	Apresentação	Interpretação de dados	Variáveis de rede e mensagens de aplicação.
5	Sessão	Controle	Autenticação.
4	Transporte	Confiabilidade ponto-a-ponto	Confirmação de recebimento de pacotes fim-a-fim. Seqüenciamento de pacotes. Detecção de pacotes duplicados.
3	Rede	Endereçamento e Roteamento	Endereçamento dos dispositivos. Roteamento de pacotes.
2	Enlace	Empacotamento e acesso ao meio físico	Empacotamento, codificação dos dados, detecção de erros por código de erros, detecção de colisões. Prioridades.
1	Física	Conexões elétricas	Interfaces elétricas para o meio físico de comunicações.

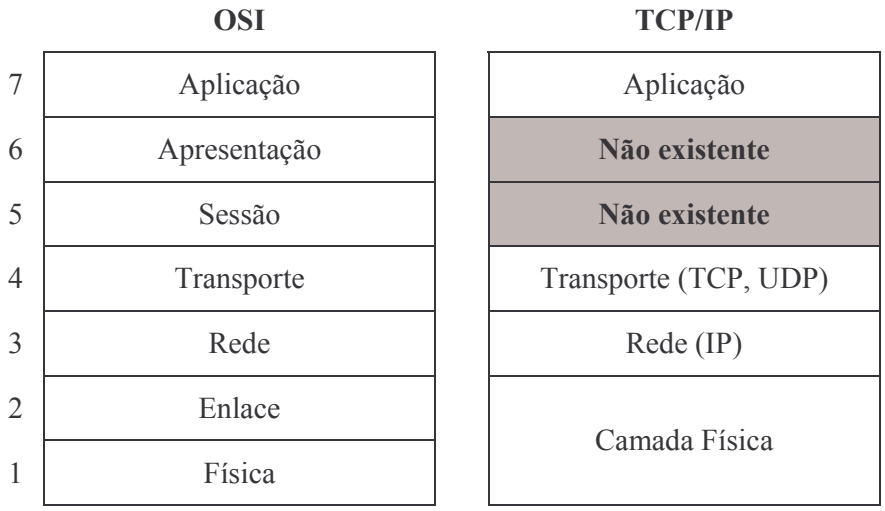


Figura 8. Relação entre os modelos OSI e TCP/IP.
Baseado em Tanembaum (1996).

A *Advanced Reserarch Projects Agency* (ARPA), ligada ao *Department of Defense* (DoD) do governo dos Estados Unidos, decidiu que essa rede deveria ser baseada em

comutação de pacotes, até então uma idéia radical. Através de parcerias e contratos com universidades e empresas, essa rede, que ganhou o nome de *Advanced Research Projects Agency Network* (ARPANET), foi desenvolvida e obteve grande sucesso. Ao longo do tempo, impulsionada pelo meio acadêmico, a rede foi crescendo tanto em abrangência geográfica quanto em número de usuários. Percebeu-se que os protocolos de comunicação utilizados não eram adequados em algumas situações, o que culminou com a invenção do TCP/IP. Voltado para a comunicação entre redes interconectadas, sua importância aumentava à medida que mais redes eram adicionadas à ARPANET. Tornou-se o único protocolo oficial em 1º de Janeiro de 1983, substituindo o anterior *Network Control Protocol* (NCP).

O sucessivo crescimento da ARPANET originou o que hoje se conhece como Internet. Tanembaum (1996) apresenta um histórico detalhado e considera a seqüência cronológica de eventos.

Comer (2000) define a Internet, a grande rede mundial, como "a coleção de redes e roteadores que abrange mais de 200 países e utiliza os protocolos TCP/IP para formar uma única e cooperativa rede virtual". Segundo a definição de Tanembaum (1996), "uma máquina está na Internet se roda a pilha de protocolos TCP/IP, possui um endereço IP atribuído, e tem a habilidade de endereçar datagramas IP para todas as outras máquinas conectadas à Internet".

Tradicionalmente, a Internet possuía quatro aplicações principais: correio eletrônico, *newsgroups*, *login* remoto e transferência de arquivos. Até o início da década de 1990, a maioria dos usuários da Internet eram pesquisadores acadêmicos, industriais, e do governo (TANENBAUM, 1996). A aplicação que revolucionou esse contexto, permitindo o crescimento explosivo do uso comercial, foi a *World Wide Web* (WWW). Através de um programa conhecido como *browser*, os usuários da Internet poderiam acessar milhares de páginas contendo informações na forma de texto, imagens, sons e até mesmo vídeos. A WWW é baseada no conceito de hipertexto, através do qual páginas são interligadas por *links*, permitindo que o usuário navegue interativamente pela informação. A WWW passou por muitos avanços e melhorias até chegar no que se conhece hoje. Os softwares e as possibilidades de conteúdo evoluíram sobremaneira, desta vez impulsionados mais pelo uso comercial do que pelo uso acadêmico.

Assim como a infra-estrutura da rede telefônica inspirou muitas outras aplicações além da transmissão de voz (a própria Internet é um exemplo, pois é possível usar a linha telefônica para conectar um computador à Internet), a crescente expansão e disponibilidade da Internet nos prédios, escritórios, casas, indústrias, fazendas, etc. inspiram outras aplicações. Em especial, neste trabalho, deseja-se considerar o seu uso como meio para acesso e interligação de redes LonWorks geograficamente distantes.

3 INTRODUÇÃO ÀS REDES LONWORKS

3.1 HISTÓRICO DOS SISTEMAS DE CONTROLE TRADICIONAIS

Com a Revolução Industrial no século XVIII, a força humana passou a ser substituída por máquinas. Inicialmente, essas máquinas apenas aumentavam a escala do que já era possível de se fazer com o trabalho humano, sem agregar novas possibilidades naquilo que já era conhecido. Os processos eram originalmente controlados e supervisionados manualmente e, com o tempo, algumas dessas atividades passaram a ser automatizadas graças à criação de novos equipamentos. Esses equipamentos, por sua vez, foram viabilizados pelo surgimento e crescimento em separado da indústria de equipamentos de controle (ASTRÖM, 1985).

Inicialmente, os sistemas de controle eram primitivos. O controle do tipo *on-off* foi amplamente utilizado na década de 1920. Apesar de o controle proporcional já ser bem conhecido desde o século XVIII, apenas no fim dessa mesma década que o mesmo começou a ser aplicado em maior escala (ASTRÖM, 1985).

Após a Segunda Guerra Mundial, iniciou-se a aplicação de instrumentos eletrônicos para controle de processos. Padrões para transmissão de sinais foram estabelecidos, como por exemplo, a conhecida faixa de 4-20 mA para sinais baseados em corrente elétrica e 3-15 psi para sinais pneumáticos.

No final da mesma década, foi introduzida a utilização de computadores nos sistemas de controle de processos. Segundo Aström (1985), em março de 1959 as companhias americanas TRW e Texaco inauguraram um sistema controlado pelo computador RW-300 em uma unidade de polimerização na refinaria de Port Arthur, Texas. Suas funções essenciais eram minimizar a pressão do reator, determinar uma distribuição ótima entre a alimentação de cinco reatores, controlar o fluxo de água quente baseado na mensuração da atividade catalisadora e determinar a recirculação ótima.

Pouco tempo depois, a IBM desenvolveu um computador digital especializado em controle de processos, o IBM 1700, o qual foi instalado na American Oil, na Standard Oil of California e na Du Pont, em 1961. Ainda no mesmo ano, a IBM anunciou um novo modelo de computador para essas aplicações, o IBM 1710.

No início, esses computadores eram grandes, lentos, caros e não confiáveis. Para justificar uma instalação, era necessário atribuir muitas tarefas à estrutura que seria criada. Devido à baixa confiabilidade, os computadores eram utilizados principalmente para executar sistemas supervisórios, deixando o controle principal da planta para os já conhecidos

equipamentos analógicos. Uma mudança drástica nesta abordagem foi realizada pela Imperial Chemical Industries Ltd. na Inglaterra, em 1962. Um sistema completo de instrumentação analógica para controle de processos foi substituído por um único computador digital realizando as mesmas funções. Este fato, a substituição de todo um conjunto de instrumentos analógicos por um painel baseado em um *display* digital e alguns botões, por trás do qual havia um computador *Ferranti Argus*, marcou o início de uma nova era na automação (ASTRÖM, 1985).

Apesar de a baixa confiabilidade dos computadores digitais ter impedido um maior progresso de suas aplicações em automação por um bom tempo, os avanços nas tecnologias de circuitos integrados possibilitaram a construção de computadores mais confiáveis, mais baratos, menores e mais rápidos. Para essas novas máquinas, surgiu o termo minicomputador. Os minicomputadores possibilitaram aplicações de controle de processos mais eficientes.

Embora o desenvolvimento tenha sido grande, ainda havia muitos problemas que os minicomputadores não resolviam. Apenas o pequeno tamanho e o baixo custo dos microcomputadores, que apareceram no início da década de 1970, permitiram que as tecnologias usadas nas funções onde os minicomputadores não podiam ser aplicados pudessem ser substituídas. Nesta ocasião, surgiram os controladores lógicoprogramáveis (CLPs) (ASTRÖM, 1985).

Um CLP é um computador digital especializado em aplicações de controle, que obtém mensurações de campo provenientes de sensores, realiza o processamento dessas informações com base em um programa pré-definido e envia sinais de saída a atuadores que interferem no estado da planta. Os CLPs se baseiam na centralização do processamento da lógica de controle da planta, ou seja, os sinais coletados por sensores na planta são levados aos CLPs que, após processamento, enviam sinais de comando a atuadores.

Conforme Castrucci e Moraes (2001), um CLP é constituído basicamente de:

- fonte de alimentação;
- unidade central de processamento (UCP);
- memórias dos tipos fixo e volátil;
- dispositivos de entrada e saída;
- terminal de programação.

3.2 SISTEMAS DE CONTROLE TRADICIONAIS

Segundo Mahalik e Lee (2002), os sistemas de controle podem ser divididos em duas categorias: os centralizados, *Centralized Control System* (CCS), e os distribuídos, Distributed Control System (DCS). Os sistemas tradicionais são baseados em arquitetura centralizada, na qual um ou poucos dispositivos realizam o processamento relativo à lógica de controle da planta. Usualmente, esses dispositivos são os CLPs.

Esses sistemas comumente possuem natureza proprietária, isto é, os equipamentos que os compõem são projetados e desenvolvidos pelo mesmo fabricante, que tem como responsabilidade fornecer todo o hardware e software necessário. As soluções de cada fabricante normalmente são incompatíveis com os produtos de outros fabricantes.

Esse modelo de negócio prende o usuário ao fornecedor, pois dificulta o processo de troca ou integração com outros sistemas. O consumidor é indiretamente forçado, por razões técnicas e econômicas, a continuar com o seu fornecedor original de equipamentos de automação ao longo da vida de seu sistema.

Nessa arquitetura tradicional de sistemas de controle, os sensores e atuadores presentes na planta são conectados por cabos a sub-painéis, que por sua vez são conectados a painéis controladores centrais. O controlador implementa a lógica de controle de todos os pontos de entrada e saída. Os sensores e atuadores tipicamente são dispositivos “burros” sem nenhum processamento interno. Castrucci e Moraes (2001) definem os sistemas centralizados como aqueles em que um computador gerencia um processo constituído por remotas, onde todo o processamento é realizado em uma única máquina. A informação percorre a direção vertical através de uma estrutura hierarquizada, como mostra a Figura 1, já comentada.

Historicamente, sempre foi difícil interconectar controladores de diferentes fabricantes (ECHELON, 1999). A incompatibilidade dos protocolos de comunicação força o encaminhamento de soluções de integração para sistemas baseados em relês e portas RS-232 programadas. Essas soluções não são transparentes e normalmente são muito limitadas, restringindo os eventuais benefícios. Além disso, nem sempre são compensatórias em termos de custo e confiabilidade.

Além da dificuldade de integração com outros sistemas, a centralização do processamento da automação em uma ou poucas unidades gera gargalos, que podem comprometer o sistema como um todo em caso de falhas.

Como pôde ser visto na seção 3.1, este tipo de arquitetura nasceu no ambiente industrial e ainda é muito utilizado atualmente. No entanto, percebe-se uma nova tendência, como será discutido adiante.

3.3 A ABORDAGEM LONWORKS PARA SISTEMAS DE CONTROLE

No final da década de 1980, surgiu a necessidade de sistemas de baixo custo de tempo real baseados em rede para dispositivos de automação (KIM; CHO; PARK, 2000). A comunicação dos dispositivos dos sistemas de automação industrial até então eram baseadas nos consagrados sinais analógicos de 4-20 mA e outros padrões. Existem diversas desvantagens no uso de sinais analógicos elétricos, tais como a necessidade de cabeamento complexo e baixa imunidade a ruído (ao menos quando comparados aos sinais digitais). Visando uma maior flexibilidade e produtividade, os sistemas de automação industrial iniciaram sua evolução na busca da implementação de uma nova geração de arquitetura de controle (MAHALIK; LEE, 2002). Para satisfazer esses requisitos, diversos padrões de redes de controle surgiram e vêm surgindo até hoje. A principal evolução que pôde ser observada nas duas últimas décadas foi a substituição dos sinais analógicos pelos digitais, os quais trafegam em pacotes de acordo com o protocolo utilizado pela rede de controle em questão.

Uma rede de controle é, de maneira geral, um conjunto de dispositivos (sensores e atuadores em geral) conectados entre si através de uma rede de comutação de pacotes. Um protocolo ou pilha de protocolos de comunicação rege as regras e detalhes da comunicação sobre esta rede. Cada dispositivo deve possuir alguma capacidade de processamento que seja suficiente ao menos para tratar de sua interface com a rede e, portanto, da comunicação que é capaz de realizar. Por possuir capacidade de processamento, seja pouca ou muita, esses dispositivos costumam ser designados por dispositivos inteligentes.

Redes de controle são a base dos sistemas de automação distribuídos. Castrucci e Moraes (2001) definem esses sistemas como um conjunto de máquinas autônomas alocadas ao longo da planta, transparentes ao operador, que, em conjunto, realizam o controle da planta. As remotas deixam de ser executoras para assumirem participação no processamento.

Verifica-se que redes de controle possuem requisitos específicos que muitas vezes não coincidem com os requisitos das redes de dados. Kim, Cho e Park (2000) citam três requisitos para a implementação de redes de controle industriais, a saber:

- tempo de resposta previsível;
- dispositivos de baixo custo;
- compatibilidade com sistemas existentes.

Redes de dados já existiam bem antes das redes de controle. Echelon (1999) frisa que os protocolos de comunicação empregados nas redes de dados eram desenvolvidos e

projetados para trafegar grandes quantidades de informação entre computadores com foco no processamento em lote (*batch*). Aplicações de controle normalmente requerem a transmissão de mensagens curtas para processamento em tempo real ou, ao menos, com tempos de resposta curtos e previsíveis. A referência destaca quatro requisitos próprios das redes de controle:

- comunicação entre dispositivos freqüente, confiável e segura;
- mensagens curtas;
- comunicação ponto-a-ponto (*peer-to-peer*) entre dispositivos;
- dispositivos pequenos e de baixo custo.

Dispositivos de redes de controle também são chamados de nós (*nodes*). A Figura 9 ilustra uma rede de controle.

O meio físico que conecta os nós da rede de controle é chamado de canal de comunicação (*communication channel*). O elemento de um nó responsável pela sua interface com o canal de comunicação é denominado transceptor (*transceiver*). Uma rede de controle não necessariamente se restringe a um único canal de comunicação. LonWorks, por exemplo, permite que haja diversos meios físicos distintos para uma mesma rede. Analogamente às redes de dados, o elemento que faz essa interconexão entre tipos diferentes de redes é chamado de roteador. No exemplo da Figura 9, tem-se dois canais interconectados: par trançado e rede elétrica.

Redes de controle podem variar completamente de tamanho. Podem possuir poucos nós como também algumas centenas ou milhares. A comunicação pode ser ponto-a-ponto ou mestre-escravo.

O desempenho e confiabilidade do sistema podem ser aumentados quando as funções do processamento de controle são distribuídas entre os nós (ECHELON, 1999) (KIM; CHO; PARK, 2000), ou seja, quando se utiliza a abordagem DCS.

A tecnologia LonWorks foi criada no início da década de 1990 pela empresa americana Echelon. De acordo com a abordagem DCS, é baseada em um protocolo de comunicação padronizado denominado LonTalk (norma ANSI/EIA 709.1) que permite integrar em rede diversos nós baseados na tecnologia. Os nós se comunicam de maneira ponto-a-ponto, descentralizando o processamento de controle da planta.

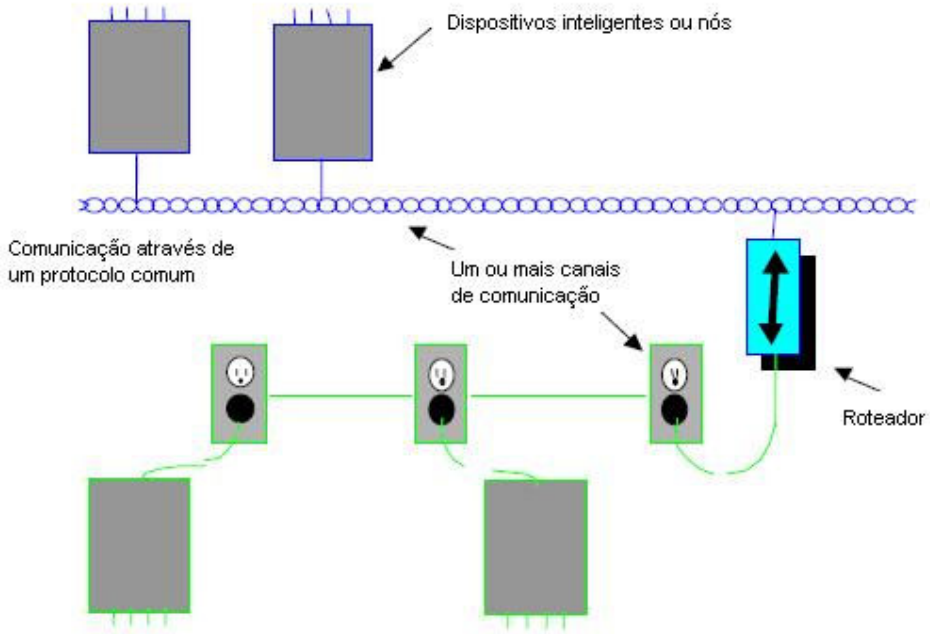


Figura 9. Exemplo de rede de controle.
Baseado em Echelon (1999).

No final da década de 1990, LonWorks se tornou uma tecnologia aberta, isto é, qualquer fabricante poderia construir produtos compatíveis com a norma ANSI/EIA 709.1 sem a exigência do pagamento de *royalties* à Echelon.

3.4 O PROTOCOLO LONTALK

Especificado pela norma americana ANSI/EIA 709.1, o protocolo LonTalk é a parte central da tecnologia LonWorks. Conforme explica Echelon (1999), ele fornece um conjunto de serviços de comunicações que possibilitam que os programas aplicativos executados nos nós enviem e recebam mensagens sem a necessidade de conhecer a topologia da rede, nomes ou endereços dos outros nós.

O protocolo LonWorks é dividido em camadas e é baseado em pacotes. Ele adere ao modelo OSI de arquitetura de redes.

Apesar de a implementação de todas as camadas do modelo OSI não ser um requisito obrigatório para um protocolo de comunicação, LonTalk implementa as sete. O objetivo desta

seção é apenas o de apresentar de forma geral o seu contexto e sua maneira de funcionar, sendo fornecidos maiores detalhes no Capítulo 4 e no Apêndice B.

Pode-se resumir as características do LonTalk nos seguintes tópicos:

- suporta diversos tipos de meios de comunicação. Basta utilizar o transceptor (transceiver) apropriado;
- suporta redes compostas por diferentes tipos de canais com diferentes taxas de transmissão. Para isso devem ser utilizados roteadores;
- suporta entrega eficiente de mensagens curtas, otimizando o uso da rede para aplicações de sistemas de controle;
- oferece tempos de resposta independentes do tamanho da rede;
- suporta implementações de baixo custo para dispositivos, ferramentas e aplicações;
- flexível na reconfiguração dos dispositivos;
- permite comunicação ponto-a-ponto (peer-to-peer), possibilitando seu uso tanto em arquiteturas distribuídas como centralizadas;
- provê um eficiente mecanismo de interoperabilidade de produtos de diferentes fabricantes.

O LonTalk implementa o conceito inovador de variáveis de rede (*network variables,* ou NVs). As variáveis de rede são as grandes responsáveis por permitir fácil interoperabilidade entre os nós produzidos por empresas diferentes. Uma variável de rede é um item de dados qualquer representando uma variável do mundo real (temperatura, pressão, estado de uma chave, etc.) que um dispositivo em particular espera receber de outro dispositivo para tomar uma decisão, ou então que pretende disponibilizar na rede para outros nós utilizarem.

Uma variável de rede utilizada como entrada para o processamento de um nó é denominada de variável de entrada, enquanto aquelas fornecidas à rede são as variáveis de saída.

O conceito de variáveis de rede caracteriza o sistema como baseado em informação, ao contrário dos sistemas baseados em comandos. Nos sistemas baseados em informação, os nós tomam decisões baseadas em processamento sobre as informações recebidas, ao contrário dos sistemas baseados em comandos, onde os dispositivos recebem um comando de atuação já pronto.

O programa de aplicação de cada dispositivo LonWorks, isto é, o software que é executado nos mesmos, não precisa conhecer onde as variáveis de rede de entrada são

geradas, e nem saber para onde as variáveis de rede de saída são enviadas. Quando um dispositivo altera o valor de uma de suas variáveis de saída, seu novo valor é informado aos outros nós via rede, de acordo com o protocolo LonTalk. Através de um processo denominado *binding*, que ocorre durante a instalação da rede, cada nó é configurado com os endereços de todos os outros nós que utilizam esta variável, enviando-a, portanto, apenas aos dispositivos de interesse. O processo de *binding* cria conexões lógicas entre variáveis de saída de um nó e variáveis de entrada de outros nós. Como sugere Echelon (1999), essas conexões podem ser vistas como "cabos virtuais".

Uma maneira simples de entender esses conceitos é considerar cada nó LonWorks, do ponto de vista lógico, como um bloco com entradas e saídas, que são as suas variáveis de rede ou NVs. O processo de *binding* consiste em conectar uma NV de saída de um nó a uma NV de entrada de outro nó (ou do mesmo), de modo que, quando a saída é alterada, o protocolo LonTalk se encarregue automaticamente de enviar uma mensagem ao outro nó, informando que houve uma alteração no valor da variável. Isso é semelhante a um circuito impresso, onde trilhas conectam saídas de um *chip* a entradas de outro *chip*. A diferença é que as variáveis de rede LonWorks podem ser tipos de dados complexos, como inteiros, *strings*, temperatura, pressão, etc, enquanto em um circuito cada trilha carrega apenas um sinal de tensão. Além disso, a trilha é uma ligação física entre a saída e a entrada, enquanto o *binding* é uma ligação lógica configurável e alterável por software.

O programa de aplicação do nó é responsável por processar os valores das variáveis de entrada e gerar as variáveis de saída. Na verdade, as saídas não são necessariamente geradas apenas com base nas NVs de entrada. Inclusive, um nó pode possuir exclusivamente variáveis de saída ou de entrada. Nós sensores, por exemplo, costumam possuir apenas variáveis de saída. Um nó sensor de temperatura comumente obtém a mensuração da grandeza fisicamente através de um transdutor, e o disponibiliza a outros nós através de uma NV de saída. Nesse caso, nenhuma NV de entrada é necessária.

Canovas et al. (2004) fornecem o clássico exemplo da lâmpada e do interruptor LonWorks: imaginando-se que um interruptor LonWorks seja um dispositivo com uma NV de saída denominada "NVO_ESTADO", a qual possa assumir os valores "0" (desligado) ou "1" (ligado); Admitindo-se que a lâmpada possua uma NV de entrada (*input*) "NVI_ESTADO" que também assuma os valores "0" ou "1" com os mesmos significados, o interruptor passará a controlar a lâmpada a partir do momento que for realizado um *binding* que "conecte" a NV "NVO_ESTADO" na NV "NVI_ESTADO". Quando alguém altera o estado da chave do interruptor para a posição ativa, seu programa de aplicação atualiza a variável "NVO_ESTADO" de "0" para "1". Como existe uma conexão lógica (*binding*) entre

"NVO_ESTADO" do interruptor e "NVI_ESTADO" da lâmpada, o valor desta última também é atualizado, através da rede, de "0" para "1". O programa de aplicação da lâmpada obtém então essa informação e a faz acender. A Figura 10 ilustra este exemplo.

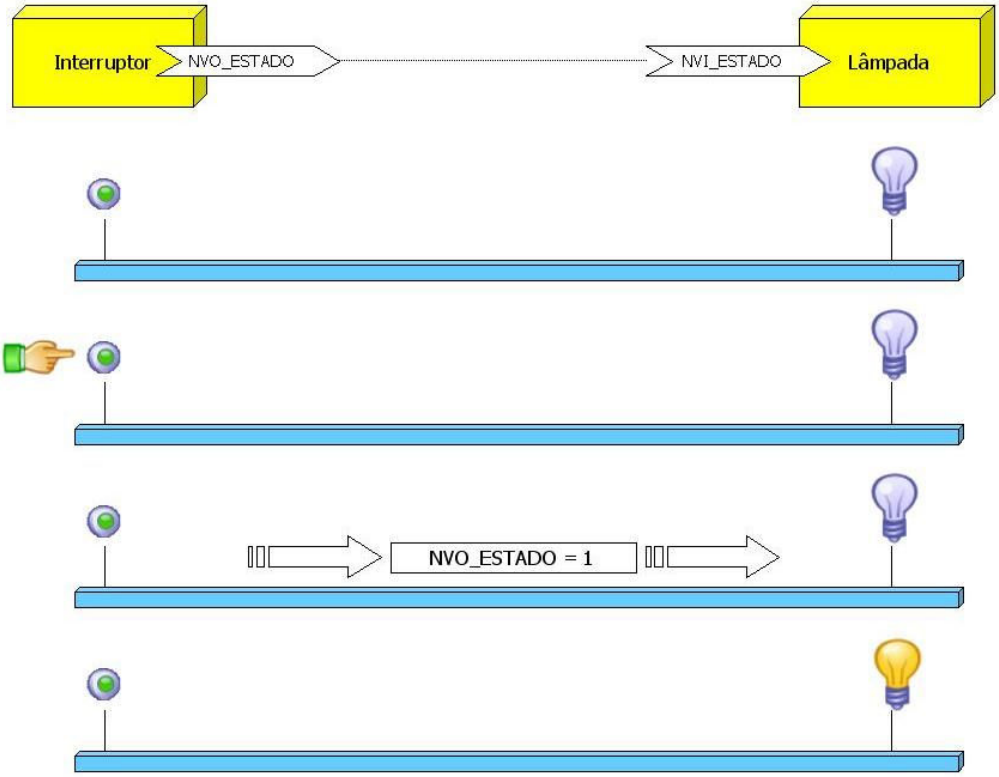


Figura 10. Exemplo de comunicação de nós LonWorks através de binding.

Tanto para o interruptor quanto para a lâmpada, é indiferente conhecer a outra ponta dos *bindings* de suas variáveis. O programa de aplicação se preocupa apenas em realizar sua tarefa. Na seção 3.7, descreve-se como o *binding* é implementado de maneira transparente aos programas de aplicações dos nós.

Os *bindings* são efetuados pelas chamadas ferramentas de gerenciamento de rede. Um software executado em um computador conectado à rede LonWorks apresenta opções para o usuário informar os *bindings* que deseja realizar. As informações fornecidas pelo usuário são convertidas em mensagens de gerenciamento, como é apresentado na seção 3.7, que são lançadas à rede LonWorks para se efetuar os *bindings*.

Nós LonWorks são endereçados por diversas maneiras. As duas principais são por Neuron ID e por domínio / sub-rede / nó. O Neuron ID corresponde a um endereço físico do nó, análogo ao endereço *Medium Access Control* (MAC) das redes Ethernet. O domínio / sub-rede / nó é um endereço lógico análogo ao endereço IP das redes IP. O Neuron ID é uma seqüência de seis bytes própria do nó e não pode ser alterado, enquanto o trio domínio / sub-rede / nó corresponde a inteiros sem sinal que podem ser configurados e alterados através de ferramentas de gerenciamento.

Outras formas de endereçamento para o envio de mensagens também são permitidas pelo LonTalk: *broadcast* (mensagens destinadas a todos os nós da rede) e *multicast* (mensagens destinadas a um grupo específico de nós).

Maiores detalhes sobre o endereçamento de nós LonWorks são apresentados no Capítulo 4.

3.5 O ÓRGÃO LONMARK

LonMark é uma associação formada para promover a interoperabilidade entre nós LonWorks de diferentes fabricantes. Como mencionado no item anterior, os dispositivos se comunicam através da troca de informações contidas nas variáveis de rede, conectadas através do processo de *binding*. O órgão LonMark vem padronizando diversos tipos de variáveis de rede, especificando seu tamanho, formato e unidade (no caso de grandezas físicas). Dispositivos que são construídos para disponibilizar e ler variáveis padronizadas pelo LonMark acabam tendo um grande potencial para se tornarem interoperáveis, já que as informações são trabalhadas da mesma forma. Os tipos de variáveis de rede padronizados pela LonMark são denominados *Standard Network Variable Types* (SNVTs).

Qualquer empresa interessada pode ser membro da LonMark. Diferentes papéis são atribuídos aos fabricantes, integradores de sistemas e usuários finais.

O órgão LonMark também especifica perfis funcionais (*functional blocks*), que descrevem em detalhe a camada de aplicação de cada tipo de dispositivo, como por exemplo, sensores de luminosidade, sensores de temperatura, etc. Um perfil funcional de uma classe de dispositivos estabelece quais variáveis de rede são obrigatórias, as propriedades de configuração e valores *default*. Também são permitidas variáveis e funcionalidades adicionais colocadas pelo próprio fabricante, além das obrigatórias.

Os dispositivos certificados pela associação LonMark podem carregar seu logotipo, indicando aos consumidores que se trata de um nó baseado em um perfil funcional. A

interoperabilidade, portanto, passa a ser garantida para nós certificados, uma vez que a LonMark estabelece perfis funcionais coerentes e padrões para todos os produtores.

A LonMark também estabelece padrões para *transceivers* e meios de comunicação.

3.6 NEURON CHIP

Uma das grandes chaves do sucesso da tecnologia LonWorks é o Neuron Chip. Conforme já mencionado, dispositivos LonWorks precisam ter o protocolo LonTalk implementado, já que é através deste que as NVs são recebidas e passadas à rede. A implementação do protocolo em um nó gera mais custos e aumenta o tempo de desenvolvimento de produtos dos fabricantes. O LonTalk é um meio para se realizar automação distribuída, e não uma finalidade em si. Os fabricantes devem se concentrar nas aplicações finais dos produtos que criam, e focar o mínimo possível naquilo que é meio.

Os Neuron Chips, fabricados pela Toshiba e Cypress, são pastilhas com três Unidades Centrais de Processamento (UCPs) internas que incorporam todas as camadas do protocolo LonTalk implementadas. São produzidos em grande escala e apresentam baixo custo (aproximadamente US$ 3,00 por unidade) Echelon (1999). A primeira UCP do Neuron Chip é chamada de processador de acesso ao meio físico (MAC), e é responsável por enviar e receber pacotes na rede, bem como calcular e checar o código de detecção de erros *Cyclic Redundancy Check* (CRC). A segunda UCP é chamada de processador de rede, que tem por finalidade realizar atividades de endereçamento, confirmações de entrega de pacotes fim-a-fim (*acknowledgement*), detecção de duplicidade de pacotes, dentre outras. A terceira e última UCP é chamada de processador de aplicação e executa o software desenvolvido pelo fabricante do nó. Trata-se de uma UCP simples de 8 bits que, mesmo assim, permite a implementação de uma série de aplicações de controle. No entanto, se o nó a ser desenvolvido requer capacidade de processamento não contemplada pelo Neuron Chip, é possível executar a aplicação em um processador mais avançado e conectá-lo ao Neuron através de pinos de entradas e saídas (E/S), tornando este um co-processador que cuida apenas da parte de comunicação com a rede.

O Neuron Chip consiste, portanto, em uma plataforma pronta para desenvolvimento de nós LonWorks. O fabricante de um nó específico não precisa se preocupar com a implementação do protocolo LonTalk e pode focar na aplicação do produto que deseja desenvolver. Sendo assim, reduz-se o custo e o tempo de desenvolvimento. Esta foi uma das principais razões para a grande adoção da tecnologia LonWorks pelo mercado.

Os programas aplicativos que são executados nos Neuron Chips são construídos através de uma linguagem de programação de alto nível denominada Neuron C (ECHELON, 1995). A Echelon oferece ao mercado a ferramenta NodeBuilder, que é um compilador da linguagem Neuron C.

Os Neuron Chips não são fisicamente conectados ao meio físico da rede LonWorks. Para isso, existem os transceptores (*transceivers*), que fazem a ligação entre ambos. A velocidade de comunicação e o tipo de meio desejados determinam qual o transceptor a ser aplicado. A tecnologia LonWorks suporta diversos tipos de transceptores: par trançado (mais usual), rede elétrica, cabo coaxial, fibra ótica e outros.

3.7 FERRAMENTAS

Esta seção cita algumas ferramentas de software conhecidas para trabalhar com redes LonWorks. Todas elas são comerciais e fornecidas pela Echelon.

3.7.1 **LNS**

O *LonWorks Network Services* (LNS) é uma plataforma de ferramentas para desenvolvimento, instalação, configuração, manutenção e monitoração de redes LonWorks. Consiste na base de operação de outras ferramentas, como o LonMaker e NodeBuilder.

Um banco de dados denominado "LNS database" contém informações sobre todos os nós de uma instalação LonWorks, as quais são utilizadas pelas outras ferramentas baseadas em LNS (ECHELON, 2004).

3.7.2 **LonMaker**

O LonMaker permite criar redes LonWorks, e como se baseia na plataforma de software da Echelon, possibilita a criação de bancos de dados LNS. Através de uma interface de fácil utilização, os usuários podem desenhar redes, criar *bindings*, configurar endereços, e realizar muitas outras tarefas de criação, instalação e gerenciamento de redes LonWorks (ECHELON, 2003a).

3.7.3 **NodeBuilder**

O NodeBuilder é um compilador de Neuron C. Permite que um desenvolvedor de nós crie programas de aplicação para serem executados nos Neuron Chips (ECHELON, 2003b).

3.7.4 **LNS DDE Server**

Visando fornecer um mecanismo fácil para a criação de aplicativos Windows que interagem com redes LonWorks, a Echelon desenvolveu o LNS DDE Server (ECHELON, 2001).

Dynamic Data Exchange (DDE) é um recurso do Microsoft Windows que permite que aplicações compartilhem informações entre si (CANOVAS; CHERMONT, 2003). Quando duas aplicações compartilham informações por DDE, diz-se que elas estão estabelecendo uma conversação DDE. Essa conversação tem um início, meio e fim bem definidos. Para iniciar uma conversação, uma aplicação (cliente DDE) solicita a outra aplicação (servidor DDE) para abrir um canal de comunicações (refere-se a um canal virtual entre duas aplicações do Windows). Uma vez que a conversação foi estabelecida, o cliente pode enviar e receber dados do servidor DDE através do canal.

O LNS DDE Server é um servidor DDE que fornece valores de NVs da rede LonWorks, permitindo também que um cliente DDE solicite alterações nos valores dessas NVs.

3.7.5 **Open LDV**

No início de 2004 a Echelon disponibilizou o OpenLDV em seu site na Internet. Trata-se de uma camada de software que provê uma interface para enviar e receber pacotes LonTalk através da família de produtos de interfaces de rede LonWorks da Echelon. Permite um acesso a informações de nível mais baixo que o LNS DDE Server, possibilitando maior flexibilidade aos desenvolvedores de aplicações Windows que interagem com redes LonWorks (ECHELON, 2005c).

O OpenLDV é livre de pagamento de *royalties* exclusivamente para uso com a família de produtos Echelon.

3.8 CONSIDERAÇÕES SOBRE O CAPÍTULO

Neste capítulo foi introduzido o conceito de redes LonWorks e sua posição no contexto da automação. Os princípios básicos do funcionamento da tecnologia, juntamente com o significado dos principais termos, tais como LonTalk, LonMark, NV, e outros, foram explicados.

No próximo capítulo, a tecnologia LonWorks é abordada com mais profundidade de modo a fornecer base para o restante do texto.

4 REDES LONWORKS E O PROTOCOLO LONTALK

4.1 O NEURON CHIP INTERNAMENTE

O Neuron Chip, introduzido no Capítulo 3, está disponível no mercado basicamente em duas versões: 3120 e 3150. Ambas são altamente integradas e, conforme já mencionado, apresentam três UCPs de 8-bits internas:

- UCP 1 - Controle de Acesso ao Meio Físico (*Media Access Control CPU*): Responsável por implementar as camadas um e dois do protocolo LonTalk (detalhado na seção 4 do Apêndice B), comandando o transceptor e executando o algoritmo de acesso ao meio físico. A UCP 1 se comunica com a UCP 2 através de *buffers* de rede localizados em uma memória compartilhada;
- UCP 2 - UCP de Rede (*Network CPU*): Executa as rotinas que implementam as camadas três a seis do protocolo LonTalk (detalhado na seção 4 do Apêndice B), sendo responsável por codificar e decodificar os pacotes que são enviados e recebidos através da rede LonWorks. Conforme as atribuições dessas camadas, esta UCP gerencia o processamento das NVs, o endereçamento, as transações, autenticação, e outros elementos. A UCP 2 se comunica com a UCP 3 através de *buffers* de aplicação;
- UCP 3 - UCP de Aplicação (*Application CPU*): Executa o programa de aplicação do usuário, realizando tarefas tais como aquisição de sinais de entrada, processamento, tomada de decisões e geração de saídas. Entende-se por usuário aquele que utiliza o Neuron Chip diretamente, ou seja, o desenvolvedor ou fabricante de nós LonWorks.

A Figura 11 ilustra a organização interna do Neuron Chip. Basicamente, as diferenças entre os modelos 3120 e 3150 são três: memória, número de temporizadores e número de pinos. Enquanto o 3120 possui 10K de ROM, 1K de RAM e 512 bytes de EEPROM, o 3150 possui 2K de RAM e requer ROM externa. Até 64K de memória externa pode ser adicionada. O 3120 apresenta um contador/temporizador de 16 bits, enquanto o 3150 possui dois deles. O chip 3120 tem 32 pinos, enquanto o 3150 tem 64.

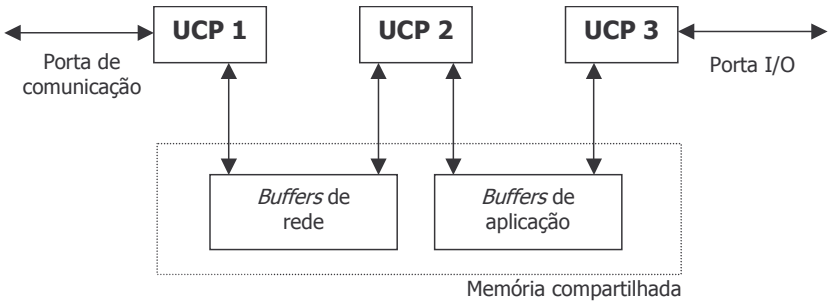


Figura 11. Organização interna do Neuron Chip.
Baseado em Toshiba (1995).

Após verificar a distribuição da implementação do protocolo LonTalk internamente ao Neuron Chip, serão apresentados nas próximas seções os serviços e características deste protocolo.

4.2 SERVIÇOS E CARACTERÍSTICAS DO PROTOCOLO LONTALK

4.2.1 **Suporte a múltiplos canais e meios físicos**

Como visto na seção anterior, o acesso ao meio físico não é realizado diretamente pelo Neuron Chip, mas sim através de um transceptor. O tipo do transceptor é o que determina o meio físico que o nó LonWorks utilizará para a comunicação. Sendo assim, um mesmo Neuron Chip suporta a utilização de diversos tipos de meios físicos, bastando empregar o transceptor adequado.

Múltiplos canais de comunicação em uma só rede também são suportados. Um canal de comunicação é o caminho físico em que os dados são transportados na rede. Em outras palavras, o canal é a implementação de um tipo de meio físico. Analogamente aos roteadores IP, roteadores LonWorks servem para interconectar canais de rede LonWorks, formando uma única rede LonWorks maior. Esse conceito é análogo à formação da Internet através da interconexão de redes IP através de roteadores IP. Cada canal de uma rede LonWorks pode ser de um tipo de meio físico diferente e, nesse caso, o roteador que os interconecta deve possuir os transceptores adequados.

4.2.2 **Taxas de comunicação**

Cada canal LonWorks pode utilizar uma dentre diversas taxas de transmissão permitidas. Uma rede LonWorks, por sua vez, pode ser formada por canais com taxas distintas. É papel dos roteadores que interconectam canais, ao repassar pacotes LonTalk de um para outro, interagir com cada canal na taxa de comunicação correta.

LonWorks permite as seguintes taxas: 0,6; 1,2; 2,4; 4,9; 9,8; 19,5; 39,1; 78,1; 156,3; 312,5; 625 e 1.250 kbps (TOSHIBA, 1995).

4.2.3 **Endereçamento e roteamento**

Analogamente aos computadores conectados às redes Ethernet que utilizam o protocolo IP, os nós LonWorks também apresentam dois tipos de endereçamento: o físico e o lógico.

O endereço físico de um nó LonWorks corresponde a um identificador de 48 bits denominado Neuron ID. Esse endereço é análogo ao endereço MAC das placas Ethernet e é atribuído ao Neuron Chip durante sua fabricação. O Neuron ID é único para cada Neuron Chip e não pode ser alterado. O endereçamento de pacotes LonTalk através do Neuron ID é comumente utilizado em situações de instalação e configuração, quando, por exemplo, os nós que se deseja configurar ainda não possuem endereços lógicos atribuídos.

O endereço lógico de um nó LonWorks é análogo ao endereço IP de um computador, podendo ser configurado pelo programa de aplicação do próprio nó, ou via rede, através de ferramentas de gerenciamento que serão apresentadas adiante. Esse tipo de endereço apresenta uma hierarquia que envolve três níveis: domínios, sub-redes (*subnets*) e nós (*nodes*).

O primeiro nível hierárquico, o domínio, permite que diferentes aplicações sejam implementadas em uma mesma rede LonWorks. Nesse caso, têm-se aplicações distintas operando independentemente sobre a mesma instalação física. A cada aplicação, ou domínio, é atribuído um identificador de 0, 1, 3 ou 6 bytes. Cada nó pode ser membro de um ou dois domínios simultaneamente. Nós de domínios diferentes não podem enviar mensagens diretamente um a outro através de endereçamento lógico, mesmo que estejam conectados a um mesmo canal LonWorks. Para isso, deve-se utilizar endereçamento físico através do Neuron ID.

O segundo nível hierárquico, as sub-redes ou *subnets*, são agrupamentos lógicos de nós do mesmo domínio. Nós de uma sub-rede podem estar espalhados por diferentes canais

da rede. Um roteador LonWorks inteligente opera neste nível, determinando quais sub-redes estão presentes em cada uma de suas portas. Com essa informação, os roteadores são capazes de manter o tráfego localizado apenas nos canais necessários, evitando que pacotes se propaguem para canais onde nunca encontrarão seu destino. Cada domínio pode possuir até 255 sub-redes.

O terceiro e último nível hierárquico são os nós. Cada sub-rede pode possuir até 127 deles, e, portanto, cada domínio pode ter no máximo 255 x 127 = 32.385 nós. Como cada nó pode estar em dois domínios simultaneamente, um nó com programa de aplicação apropriado pode servir como um elemento de roteamento entre domínios. Além disso, um nó sensor simples pode transmitir os valores de suas leituras para dois domínios diferentes, permitindo que duas aplicações distintas os utilizem.

Dessa maneira, o endereço lógico de um nó pode ser representado por um trio (domínio, sub-rede, nó), ou (*domain*, *subnet*, *node*).

Além do agrupamento lógico de nós estabelecido pelas sub-redes, os nós também podem compor grupos (*groups*). Os grupos formam uma maneira alternativa de se agrupar nós logicamente, já que cada grupo pode possuir nós de sub-redes diferentes, mas sempre dentro do mesmo domínio. Como um domínio pode se espalhar por diversos canais, então os grupos também podem ser compostos por nós de canais diferentes. Cada domínio pode possuir 256 grupos, cujo número de nós permitido varia conforme o tipo de serviço utilizado. Para os serviços *acknowledged* ou *request/response*, até 64 nós podem pertencer a um grupo, enquanto que para o serviço *unacknowledged*, não há limites para o número de nós dentro de um grupo. Um nó pode ser membro de 15 grupos simultaneamente. Os tipos de serviço previstos pelo protocolo LonTalk são descritos na seção 4.2.4.

Dentro de um grupo, para uso nos serviços *acknowledged* e *request/response*, cada nó possui um identificador único, o *member id*.

O endereçamento de pacotes LonTalk por grupo permite a aplicação do conceito de *multicast*, através do qual uma determinada informação pode ser transmitida a diversos dispositivos LonWorks através do envio de uma única mensagem na rede.

4.2.4 **Serviços de entrega de mensagens**

O protocolo LonTalk oferece quatro tipos de serviço de entrega de mensagens, a saber:

- *Acknowledged* (ACKD): Uma mensagem LonTalk pode ser enviada para um nó ou grupo de nós. Após enviar a mensagem, o remetente dispara um temporizador e

aguarda por uma confirmação de recebimento (ACK). Se essa confirmação não chegar até o temporizador expirar, o pacote é retransmitido. O tempo do temporizador e o número máximo de retransmissões podem ser configurados. Se esse número máximo for atingido e o ACK não for recebido, o software que implementa o LonTalk informa um erro ao programa de aplicação do dispositivo;

- *Request / Response* (REQUEST): Uma mensagem LonTalk contendo uma requisição pode ser enviada para um nó ou grupo de nós e exigir que cada um deles retorne uma resposta baseada na mesma. Esse tipo de serviço é similar ao *acknowledged* (o temporizador e as retransmissões estão presentes da mesma maneira), com a diferença de o ACK retorna juntamente com alguma resposta processada pelo nó de destino. Portanto, este serviço é adequado para a implementação do conceito de *Remote Procedure Call* (RPC);
- *Repeated* (UNACKD_RPT): Uma mensagem LonTalk pode ser enviada para um nó ou grupo de nós repetidas vezes, sem que, no entanto, o remetente exija o recebimento de um ACK. O objetivo do envio repetido da mesma mensagem é aumentar a probabilidade de que ela realmente seja entregue corretamente ao seu destino final. Esse serviço é tipicamente utilizado nos casos em que os ACKs gerados pelo serviço *acknowledged* podem degradar a taxa de transmissão total da rede. O número de retransmissões é configurável a cada envio;
- *Unacknowledged* (UNACKD): Similar ao serviço *repeated*, mas nesse caso a mensagem é enviada uma única vez, sem repetição. Aplicações que utilizam esse serviço não podem ser sensíveis a perdas ocasionais de pacotes, já que se trata de um serviço de entrega não confiável (análogo ao IP ou UDP).

Assim como no caso das redes IP, as redes LonWorks estão sujeitas à duplicidade de pacotes. Essa duplicidade pode ocorrer através da perda de ACKs. Considerando-se o caso em que o receptor de uma mensagem efetivamente a recebeu, mas o remetente não foi notificado pois o ACK foi perdido, haverá a retransmissão do pacote. O protocolo LonTalk prevê a detecção de mensagens duplicadas.

A cada mensagem transmitida por um nó, é associado um identificador de quatro bits denominado *tag*. Esse identificador é incrementado a cada nova mensagem enviada. Após atingir seu valor máximo, isto é, 15, seu valor retorna a zero. Quando uma retransmissão deve ser feita após o temporizador de espera de um ACK expirar, seja nos serviços *acknowledged*, *request/response* ou *repeated*, a mensagem é retransmitida com o mesmo *tag*, permitindo que o receptor identifique que se trata de uma mensagem duplicada.

O *tag* também serve para que o nó transmissor consiga associar corretamente os ACKs recebidos às mensagens enviadas, já que diversas mensagens podem ser enviadas enquanto seus respectivos ACKs ainda não chegaram. Quando eles chegarem, o nó será capaz de estabelecer qual ACK corresponde a qual mensagem, pois o mesmo retorna com uma cópia do *tag* originalmente enviado.

4.2.5 **Autenticação**

O protocolo LonTalk suporta autenticação, isto é, um mecanismo que permite os receptores determinarem se os transmissores realmente estavam autorizados a enviar as mensagens que enviaram.

A autenticação do protocolo LonTalk é baseada na distribuição de chaves de 48 bits durante a instalação da rede. A cada domínio onde se deseja usar autenticação, deve-se atribuir uma chave. Um receptor que deseja comprovar a autenticidade do transmissor envia de volta um desafio (*challenge*) quando recebe uma mensagem. O desafio, por sua vez, consiste em um pacote contendo um número aleatório. O transmissor, ao receber o desafio, calcula uma resposta baseada na chave de autenticação, no número aleatório e nos dados enviados na mensagem original. Essa resposta é denominada transformação. O desafio é respondido ao receptor, que compara essa transformação com os seus próprios cálculos. Se a resposta for válida, o transmissor é autenticado e é dado prosseguimento à transação.

O algoritmo que gera uma resposta com base na chave, no número aleatório e nos dados, é elaborado de tal maneira que seja muito difícil e demorado descobrir qual é a chave a partir de uma amostra de número aleatório, dados, e resposta.

O uso de autenticação pode ser ativado ou desativado individualmente por *binding*. Esse recurso ajuda a impedir que nós não autorizados enviem comandos de controle para a instalação LonWorks.

4.2.6 **Prioridade**

O protocolo LonTalk oferece um mecanismo de prioridade que garante que pacotes críticos, mais prioritários, sejam enviados à rede imediatamente após o término da transmissão que está ocorrendo (se houver), passando à frente de pacotes menos prioritários.

Maiores detalhes podem ser encontrados no Apêndice B.

4.3 LONTALK E O MODELO OSI

O protocolo LonTalk adere ao modelo OSI (conforme já apresentado na Tabela 2), possuindo implementação para as sete camadas. De acordo com este modelo, uma camada oferece serviços para a camada imediatamente superior, mantendo escondidos desta todos os detalhes existentes abaixo.

A correspondência entre o LonTalk e o modelo OSI pode ser conferida a seguir. Maiores detalhes estão presentes no Apêndice B.

- Camada 1 (Física): Relacionada a aspectos elétricos, de conexão e cabeamento. A tecnologia LonWorks especifica diversas taxas de transmissão e diversos meios físicos cujo acesso está implementado nos transceptores ligados aos Neuron Chips;
- Camada 2 (Enlace): Define regras de acesso ao meio físico e o formato do quadro (*frame*) LonTalk. Entre os serviços oferecidos, tem-se:

    - detecção de erros através de CRC;
    - mecanismos de prioridade;
    - garantia de uso da capacidade máxima da rede mesmo quando sobrecarregada (*predictive p-persistent* CSMA);
    - tratamento de colisões.

- Camada 3 (Rede): Trata do endereçamento e roteamento dos pacotes LonTalk. Os roteadores LonWorks operam neste nível, repassando os pacotes adequadamente entre os canais aos quais estão conectados, baseando-se, para isso, no endereço de destino de cada mensagem;
- Camada 4 (Transporte): Provê entrega de pacotes fim a fim. Relaciona-se aos tipos de serviços descritos na seção II.4 do Apêndice B. Os serviços *acknowledged* e *request/response* são confiáveis (análogos ao TCP), enquanto *repeated* e *unacknowledged* não são confiáveis (análogos ao UDP);
- Camada 5 (Sessão): O serviço *request/response*, já descrito, implementa a camada 5 pois provê comunicação entre aplicações através de requisições e respostas. Como o ACK é embutido em uma resposta, automaticamente se realiza o serviço da camada 4, que é a garantia de entrega fim-a-fim;
- Camada 6 (Apresentação): A camada 6 é conceitualmente responsável pela tradução dos dados entre o formato que circula na rede e o formato utilizado pela

aplicação. A tecnologia LonWorks provê a transmissão correta de dados entre programas de aplicação através da padronização da comunicação pelas variáveis de rede de entradas e saídas.

- Camada 7 (Aplicação): A compatibilidade e a garantia efetiva do entendimento dos dados trocados entre diferentes programas de aplicação de diferentes nós é oferecida pela tecnologia LonWorks através da padronização dos tipos de variáveis de rede (SNVTs) e perfis funcionais, que são atribuições do órgão LonMark.

4.4 TRATAMENTO DE COLISÕES

Em alguns sistemas de comunicação, os protocolos empregados garantem que não haverá colisões. Esses protocolos, conhecidos como livres de colisão (*collision-free protocols*), garantem, através de sua definição, que dois ou mais transmissores nunca irão causar uma colisão. Arcnet e ARINC-629 são exemplos desses protocolos (KOOPMAN, 1996). Outro exemplo bastante conhecido são as redes baseadas no *Controller Area Network* (CAN).

Por outro lado, existem protocolos que permitem a ocorrência de colisões simplesmente por elas se tratarem de uma das poucas maneiras práticas de acomodar eficientemente um grande número de nós em um mesmo meio de comunicação (KOOPMAN, 1996). Uma colisão ocorre quando mais de um transmissor acessa o meio físico para enviar dados. Colisões aparecem com mais freqüência quando o tráfego na rede está alto ou há situações de rajada (*burst*). São eventos normais que não inviabilizam o sistema de comunicação, contanto que existam mecanismos eficientes responsáveis por tratar delas.

A maneira com que a tecnologia LonWorks trata das colisões é apresentada no Apêndice B.

4.5 O PACOTE LONTALK

Os serviços descritos na seção 4.3 são oferecidos pelo protocolo LonTalk através do conjunto das atribuições de todas as camadas[1]. Assim como no modelo OSI e na arquitetura TCP/IP, cada camada do LonTalk define um *Protocol Data Unit* (PDU) que é encapsulado na área de dados do PDU da camada inferior. Isso é feito sucessivamente até compor o quadro

[1] Esses serviços são as atribuições e possibilidades gerais do protocolo, não correspondendo aos tipos de serviços de transporte de mensagens descritos na seção 4.2.4).

(*frame*) completo que, por sua vez, é enviado como um pacote ao canal físico de comunicação através dos serviços da camada 1 (física).

Os pacotes LonTalk podem carregar diversos tipos de mensagens: gerenciamento, diagnóstico, atualização de valor de NV, requisição de valor de NV, e outros. O tipo de cada mensagem e a informação que carregam estão determinados pelo conteúdo dos campos que compõem o pacote.

O detalhamento dos PDUs das camadas do LonTalk e seus respectivos formatos podem ser encontrados no Apêndice B.

4.6 GERENCIAMENTO E DIAGNÓSTICO DE REDES LONWORKS

Entende-se por gerenciamento de rede LonWorks (*network management*) todas as atividades de instalação e configuração de nós através da própria rede. A atribuição de endereços lógicos a cada dispositivo (domínio, sub-rede, nó), realizada através de uma ferramenta de software em execução em um PC conectado à rede, é um exemplo de atividade de configuração que costuma ser realizada como parte do processo de instalação da rede. O conceito de diagnóstico é relacionado às atividades de obtenção de status, estatísticas e configurações dos nós. Essas informações podem ser utilizadas para testes do funcionamento da rede, engenharia reversa (obter um modelo da rede e de seus dispositivos a partir da instalação física em funcionamento) e como parâmetros para a inferência de diversos tipos de conclusões sobre o funcionamento da rede LonWorks.

O protocolo LonTalk define diversas mensagens de gerenciamento e diagnóstico, cada uma com propósitos específicos. Normalmente, essas mensagens são enviadas através do serviço *request/response*, já que costumam requerer algum tipo de informação de retorno do nó que se deseja configurar ou diagnosticar. Entretanto, há exceções.

A implementação do tratamento das mensagens de gerenciamento e diagnóstico já é parte do *firmware* do Neuron Chip, e, portanto, não é necessário que o desenvolvedor de nós LonWorks precise estudar, conhecer, e nem implementar rotinas de tratamento dessas mensagens.

Mensagens de gerenciamento e diagnóstico são comumente enviadas por softwares desenvolvidos para PCs conectados à rede LonWorks. Essas ferramentas tipicamente possuem interfaces gráficas que permitem o usuário configurar e instalar uma rede LonWorks de maneira bastante simples, inclusive contemplando a criação de *bindings*. Exemplos de ferramentas desse tipo são Echelon LonMaker (ECHELON, 2003a) e IEC ICELAN (PRESSBOX, 2002). Apesar de não ser comum, nós LonWorks também são capazes de trocar

mensagens de gerenciamento e diagnóstico com outros. Não é o objetivo deste trabalho descrever todas as mensagens de gerenciamento e diagnóstico suportadas pela tecnologia LonWorks com seus respectivos formatos de dados de requisição e resposta. No entanto, a lista completa consta no Apêndice B e um exemplo é apresentado mais adiante. O Apêndice B de Toshiba (1995) oferece uma referência detalhada dessas mensagens.

Como o Neuron Chip possui capacidade tratar as mensagens de gerenciamento e diagnóstico que recebe, sem nenhuma interação com o programa de aplicação, pode-se concluir que este circuito integrado possui estruturas de dados internas para armazenar as configurações e estados envolvidos nas requisições e respostas dessas mensagens. Existem diversas dessas estruturas, sendo que a maioria é armazenada na memória EEPROM presente tanto no modelo 3120 quanto no modelo 3150. O Apêndice A de Toshiba (1995) apresenta as descrições completas de cada uma dessas estruturas. O programa de aplicação que é executado no Neuron Chip também tem a possibilidade de ler e alterar essas estruturas, o que é útil, por exemplo, no caso de um nó que deve ser capaz de se autoconfigurar. Entretanto, essas configurações são tipicamente realizadas através de mensagens de gerenciamento enviadas por softwares executados em PCs.

Como exemplo, mostra-se aqui o processo de *binding*, que é o responsável por uma NV de saída conseguir comunicar às NVs de entrada conectadas que houve uma alteração de valor. As mensagens LonTalk que configuram *bindings* nos nós da rede são mensagens de gerenciamento.

Duas das estruturas internas do Neuron Chip para armazenamento de configurações são a tabela de configuração de variáveis de rede (*network variable configuration table*) e a tabela de endereços (*address table*), ambas envolvidas no processo de *binding*. Como já foi mencionado, essas tabelas ficam armazenadas na memória EEPROM interna do Neuron Chip e são gerenciadas pelo seu *firmware*, isto é, não são de responsabilidade do programa de aplicação (apesar de que esses também possuem acesso a essas tabelas).

A tabela de configuração de variáveis de rede define os atributos configuráveis das variáveis de rede de um nó. Há uma entrada nessa tabela para cada NV definida no dispositivo. Cada entrada, ou linha da tabela, é composta por sete campos. Omitindo os detalhes, que podem ser conferidos no Apêndice A de Toshiba (1995), para este exemplo nos interessa considerar apenas dois campos: “seletor” e “índice da tabela de endereços”.

A tabela de endereços de um nó LonWorks armazena endereços para os quais o nó pode enviar mensagens através do chamado endereçamento implícito. Na prática, significa que essa tabela armazena endereços de nós com os quais existem *bindings* de NVs com as NVs do nó em questão.

Essas duas tabelas podem ser alteradas remotamente através de mensagens de gerenciamento, isto é, solicitações específicas que trafegam pela rede LonWorks. O protocolo LonTalk prevê as mensagens de gerenciamento denominadas “Update Net Variable Config” e “Update Address”, que são responsáveis respectivamente por encaminhar pedidos de alteração em registros da tabela de configuração de NVs e da tabela de endereços.

Retoma-se aqui o exemplo do interruptor e da lâmpada da seção 3.4. De acordo com o que foi especificado, o interruptor possui apenas uma NV de saída denominada “NVO_ESTADO”, enquanto a lâmpada possui a NV de entrada “NVI_ESTADO”. A tabela de configuração de variáveis de rede de ambos os nós possui uma linha cada, as quais contêm as configurações de cada NV.

Uma conexão lógica entre duas variáveis, o *binding*, ocorre quando o valor do campo “seletor” das duas NVs são iguais. Além disso, o campo “índice da tabela de endereços” deve indicar um registro na tabela de endereços que corresponda ao endereço do nó que possui a NV da outra ponta do *binding*.

Supõe-se que o interruptor esteja configurado como nó 8 da sub-rede 5, enquanto a lâmpada é o nó 9 da sub-rede 18. (É necessário também supor que eles estão no mesmo domínio, não importando qual seja). Para que haja um *binding* entre NVO_ESTADO e NVI_ESTADO, as tabelas de configuração de variáveis de rede e de endereços de cada nó devem apresentar o conteúdo da Figura 12.

Interruptor (Subnet 5 / Node 8):

Tabela de configuração de NVs

#	Seletor	Índice da Tabela de Endereços
1	37	1

Tabela de endereços

#	Subnet	Node
1	18	9

(a) Tabelas do interruptor

Lâmpada (Subnet 18 / Node 9):

Tabela de configuração de NVs

#	Seletor	Índice da Tabela de Endereços
1	37	1

Tabela de endereços

#	Subnet	Node
1	5	8

(b) Tabelas da lâmpada

Figura 12. Exemplo de tabelas de configuração de NV e de endereços em um binding.

Quando alguém pressiona a chave do interruptor, há uma atualização em NVO_ESTADO. O *firmware* do Neuron Chip, que implementa o LonTalk, saberá que deve

ser enviada uma mensagem de atualização de variável para o nó cujo endereço é nó 9 / sub-rede 18. A lâmpada, ao receber esta mensagem, identifica através do número do seletor à qual variável de entrada a atualização de valor corresponde. Nesse caso, há apenas uma variável, NVI_ESTADO, mas se o nó apresentasse também outras NVs de entrada, estaria clara a utilidade do seletor.

O processo de *binding* costuma ser efetuado através de ferramentas de software de gerenciamento de redes LonWorks. A mais conhecida é o aplicativo LonMaker, da Echelon. O LonMaker deve ser executado em um PC conectado a uma interface de rede LonWorks. Ele apresenta uma interface amigável onde os nós da rede LonWorks são visualizados graficamente pelo usuário na forma de blocos. Este, por sua vez, clica e arrasta o mouse para realizar conexões entre as NVs. O LonMaker, quando solicitado para efetuar o comissionamento, envia pela rede as mensagens de gerenciamento apropriadas ("Update Net Variable Config" e "Update Address") de forma a alterar convenientemente as tabelas de configuração de variáveis de rede e de endereços. Quando essas mensagens de gerenciamento são recebidas por um nó LonWorks, são tratadas pelo *firmware* do Neuron Chip, que independe do programa de aplicação que é executado no nó. Apesar de o exemplo em questão considerar apenas duas mensagens, esse conceito é válido para todas as mensagens de gerenciamento do LonTalk. Assim, verifica-se claramente que o fabricante de nós que adota o Neuron Chip consegue focar exclusivamente em sua aplicação, sem custos e tempo de desenvolvimento com funções relativas ao protocolo LonTalk.

Nesse exemplo, foi utilizado o seletor 37 (Figura 11). Ele foi escolhido aleatoriamente para ilustrar o *binding*. Basta que esse número corresponda nas tabelas dos nós envolvidos na conexão. Na prática, conforme detalha o Apêndice A de Toshiba (1995), há alguns valores reservados para indicar que a NV não participa de *bindings*.

4.7 CONSIDERAÇÕES SOBRE O CAPÍTULO

Após a introdução do Capítulo 3, este capítulo apresentou a tecnologia LonWorks com a profundidade necessária para o entendimento do restante do texto.

O Capítulo 5 reúne os conceitos de redes LonWorks e redes IP, apresentando os tipos de abordagem existentes para a interconexão das mesmas.

5 ABORDAGENS PARA INTERCONEXÃO ENTRE REDES LONWORKS E REDES IP

5.1 INTRODUÇÃO

Este capítulo tem por objetivo mostrar como as categorias de aplicações levantadas na seção 1.2 podem ser implementadas com sucesso. Relembrando as categorias de aplicações apresentadas na seção 1.2, tem-se:

1. monitorar ou enviar comandos para a rede LonWorks através de uma estação ligada à rede de dados (sistema supervisório);
2. gerenciar e configurar a rede LonWorks através de uma estação ligada à rede de dados;
3. utilizar a rede de dados como meio para interconectar redes LonWorks geograficamente distantes para aplicações sem requisitos fortes de tempo real (*soft real time*);
4. utilizar a rede de dados como meio para interconectar redes LonWorks geograficamente distantes para aplicações com requisitos fortes de tempo real (*hard real time*), tendo em vista principalmente a implementação de laços de controle onde o controlador e a planta não estão conectados diretamente ou na mesma rede, mas sim através de uma rede IP.

Busca-se atingir o objetivo deste e dos próximos capítulos através da apresentação de respostas para as seguintes perguntas, compreendendo em sua totalidade o escopo do trabalho:

a. Que tipos de soluções existem para a interconexão entre redes LonWorks e IP?
b. Que tipos de parâmetros de rede devem ser considerados para uma solução baseada em redes LonWorks / IP? Como eles podem ser utilizados para estabelecer requisitos de uma certa aplicação?
c. A quais requisitos cada uma das quatro categorias de aplicação acima está associada?
d. Quais tipos de soluções, dentre as apresentadas, melhor atendem a cada uma das quatro categorias de aplicações?

Na seção 5.2, apresenta-se as principais abordagens para se conectar redes LonWorks a redes IP (questão a). Introduz-se então, na seção 6.1, alguns tipos de aplicações, desta vez sob o ponto de vista de sistemas de tempo real. Essa introdução é relevante para este trabalho na medida em que as categorias 3 e 4 da lista acima citam explicitamente requisitos de tempo real. Será apresentado que 1 e 2 também podem ou não apresentar esse tipo de requisito.

Na seção 6.2, mostra-se parâmetros relacionados à qualidade de serviço da rede (QoS) e como eles podem ser utilizados formalmente para estabelecer restrições impostas por uma determinada aplicação (questão b).

Por fim, para cada uma das quatro categorias de aplicações, serão vistos nas seções 7.1 e 7.2 os requisitos associados (questão c) e as soluções, dentre aquelas apresentadas como respostas à questão a, que melhor os atendem (questão d).

Em razão da similaridade, as categorias 1 e 2 serão abordadas de maneira conjunta. A diferença fundamental entre supervisão e gerenciamento remotos de redes LonWorks está nos comandos tratados pela rede. Basicamente, no primeiro caso tem-se comandos de leitura e alteração de variáveis, enquanto no segundo tem-se leitura e alteração de configurações. Dado que os requisitos para integração com redes IP são semelhantes, este agrupamento de tópicos torna-se conveniente.

Novamente, devido à similaridade, os tópicos 3 e 4 também serão abordados conjuntamente. A única diferença entre eles está no requisito de tempo real, como pôde ser observado nas descrições dos mesmos. O tópico 4 é um extenso assunto e é alvo de muita pesquisa atualmente (OVERSTREET; TZES, 1999), sendo merecedor de um outro trabalho dedicado a ele. Este tópico será abordado apenas de maneira simplificada, abrindo caminho para uma futura continuidade desta pesquisa.

5.2 ABORDAGENS PARA INTERCONEXÃO ENTRE REDES LONWORKS E REDES IP

5.2.1 **Introdução**

Existem quatro abordagens básicas para interligar uma rede LonWorks a uma rede IP (GAW; MARSH, 1997), a saber:

1. interface de rede LonWorks baseada em IP;

2. roteador LonWorks / IP;
3. *gateway*;
4. acesso remoto.

As seções seguintes fornecem mais detalhes sobre cada uma das quatro opções. Em decorrência da similaridade entre os itens 3 e 4 (acesso remoto usualmente é implementado por um *gateway*), eles serão tratados conjuntamente. Nas seções 7.1 e 7.2 são mostradas quais dessas opções melhor se encaixam em cada tipo de aplicação.

5.2.2 **Interface de rede LonWorks baseada em IP**

O método convencional para conectar um PC a uma rede LonWorks baseia-se na instalação de uma interface de rede LonWorks, também conhecida no contexto desta tecnologia apenas como interface de rede, ou *Network Interface* (NI). Uma interface de rede consiste em uma placa ou dispositivo que deve ser instalado no PC e que apresenta um conector físico para ligação à rede LonWorks. Pode-se fazer uma analogia a uma placa de rede convencional, a qual é instalada em um PC e apresenta conector Ethernet.

Uma interface de rede LonWorks possui, internamente, um Neuron Chip e um transceptor. Naturalmente, o Neuron Chip detém um Neuron ID único e, portanto, a interface pode ser endereçada como qualquer outro dispositivo na rede LonWorks. O tipo de conector LonWorks depende do transceptor utilizado na construção da interface, sendo o compatível com par trançado a 78kbps o mais usual (TP/FT-10). Ferramentas de software como LonMaker e NodeBuilder requerem que o PC apresente uma interface de rede LonWorks para se comunicar com a mesma.

No mercado, estão disponíveis diversos tipos de interfaces de rede LonWorks, tais como placas compatíveis com os padrões *Peripheral Component Interconnect* (PCI) e *Universal Serial Bus* (USB), além de cartões de padrão *Personal Computer Memory Card International Association* (PCMCIA) para *notebooks*. A Figura 13 ilustra um PC conectado a uma rede LonWorks através de uma placa PC LonTalk Adapter (PCLTA), que é uma interface de rede comercializada pela empresa Echelon. Apesar da representação externa, a PCLTA é instalada internamente no gabinete do PC através de conexão ao barramento PCI.

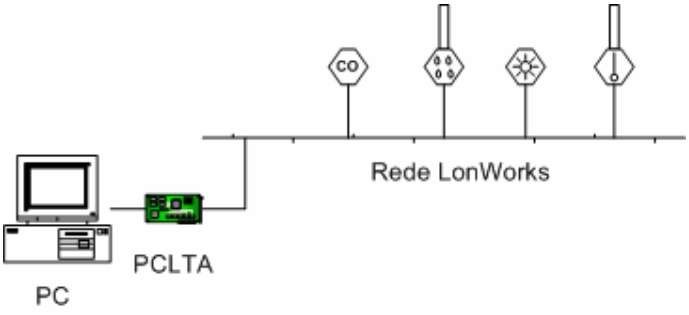


Figura 13. PC conectado a uma rede LonWorks através de uma PCLTA.

Juntamente com o hardware, seja PCI, USB, PCMCIA ou outros, deve ser instalado o *driver* apropriado fornecido pelo fabricante. O *driver* encapsula as rotinas de baixo nível de acesso à placa. Sendo assim, os aplicativos podem acessar a rede LonWorks através de chamadas a *Application Programming Interfaces* (APIs) de nível mais alto, ficando os detalhes de acesso ao hardware transparentes à aplicação. Ao mesmo tempo, isso também torna esses aplicativos independentes de modelos e fabricantes das interfaces.

Apesar da simplicidade desta solução convencional para conectar um PC a uma rede LonWorks, ela apresenta um inconveniente: é necessário que o PC esteja próximo ao cabeamento LonWorks, o que nem sempre ocorre. Considerando que as empresas normalmente já possuem sua instalação física de rede de dados local (LAN), e que o padrão *de facto* para essas redes é baseado em Ethernet e IP, conforme já discutido, uma nova solução foi criada: interfaces de rede LonWorks baseadas em IP. Esses dispositivos possuem um conector Ethernet para a LAN e outro para a rede LonWorks. O *driver* desse dispositivo deve ser instalado no PC que se deseja conectar à rede LonWorks. O *driver* fornece aos aplicativos uma API idêntica àquela das interfaces diretamente conectadas ao PC (PCI, USB, PCMCIA, etc.), mantendo compatibilidade com os aplicativos LonWorks já existentes (LonMaker, NodeBuilder, etc.). No entanto, o *driver* usa a convencional placa de rede Ethernet, normalmente já disponíveis nos computadores, para se comunicar com a interface de rede LonWorks, que por sua vez implementa o protocolo LonTalk. A Figura 14 fornece uma ilustração dessa solução. Assim, não é mais necessário que o PC esteja localizado próximo ao cabeamento LonWorks. O dispositivo que precisa atender a esse requisito passa a ser a interface de rede LonWorks baseada em IP, também conhecida como Network Interface Card – IP (NIC-IP).

Uma vantagem deste tipo de solução é que alguns modelos de NICs-IP são capazes de oferecer acesso simultâneo à rede LonWorks para vários PCs na rede IP. Desta maneira,

comparando-se com o caso da Figura 13, é como se os vários PCs possuíssem cada um a sua própria placa PCLTA instalada. Este tipo de NIC-IP multiplexa entre os diversos PCs, de maneira transparente, a sua única conexão física com a rede LonWorks, fazendo como se cada um deles pudesse acessar uma interface inteiramente a eles dedicada, como se o computador estivesse diretamente conectado ao cabeamento LonWorks. A Figura 15 ilustra essa situação. Um exemplo de fornecedor no mercado para este tipo de NIC é a empresa austríaca Loytec (LOYTEC, 2005).

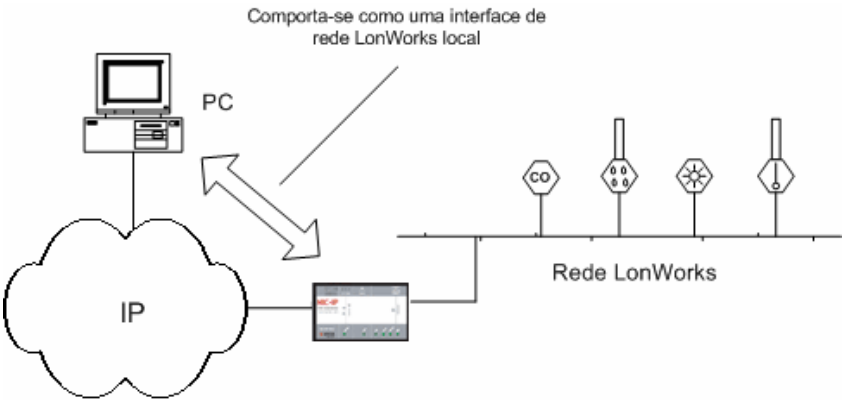


Figura 14. PC conectado a uma rede LonWorks através de uma NIC-IP.

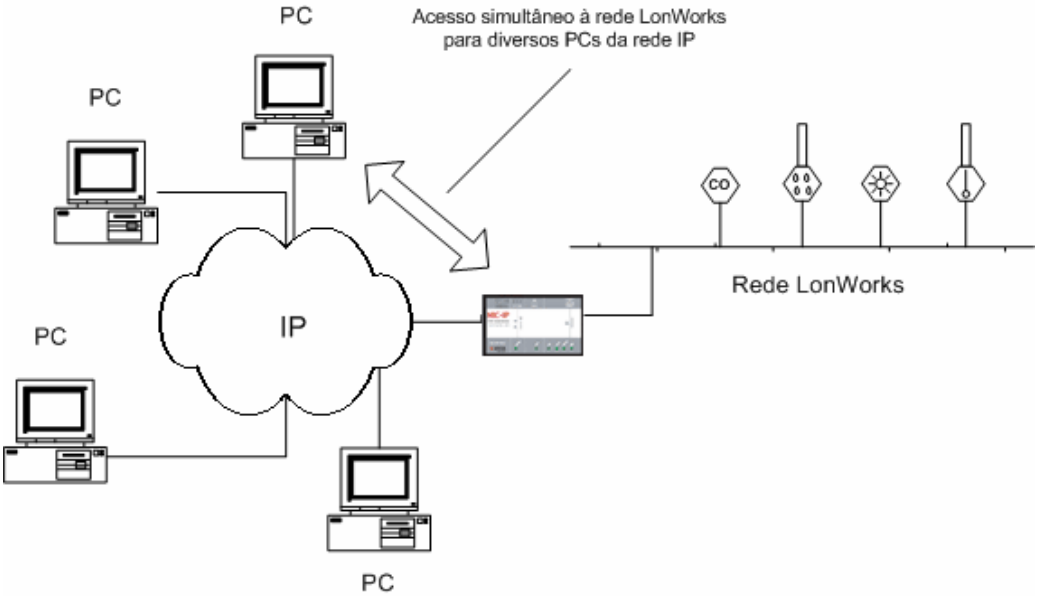


Figura 15. PCs conectados simultaneamente a uma rede LonWorks através de uma NIC-IP.

A comunicação entre os PCs e a NIC-IP é implementada pelo *driver* do fabricante (que deve ser executado em cada PC) e pela própria NIC-IP, sendo totalmente transparente para a aplicação.

O protocolo utilizado para a comunicação entre o PC e a NIC-IP pode ser proprietário ou padronizado, dependendo da opção de projeto do fabricante. Um exemplo de padronização é a norma EIA-852, muito utilizada para roteadores LonWorks/IP. Neste caso específico, torna-se mais fácil implementar um equipamento que pode funcionar tanto como NIC-IP ou como roteador. A seção 5.2.3 apresenta maiores detalhes sobre essa padronização.

5.2.3 **Roteador LonWorks/IP**

Roteadores LonWorks/IP são dispositivos cuja função é encapsular pacotes LonTalk em datagramas IP e transmiti-los através de uma rede IP. Sua principal aplicação é a interconexão de diferentes canais LonWorks, normalmente separados geograficamente. Os pacotes LonTalk gerados em uma rede LonWorks são encapsulados em datagramas IP e enviados às outras redes LonWorks através da rede de dados. Desta maneira, a rede IP pode ser considerada, do ponto de vista das redes LonWorks, como se fosse mais um meio físico disponível para sua implementação, assim como existe o par trançado, rede elétrica, fibra ótica e outros. Sob este ponto de vista, as redes IP são conhecidas como canais IP (*IP Channels*). Os roteadores LonWorks / IP tornam esta interconexão totalmente transparentes aos dispositivos LonWorks. Dispositivos de um canal podem endereçar dispositivos localizados em outro canal, sem precisar ter o conhecimento de como eles estão interconectados.

A arquitetura desta solução pode ser visualizada na Figura 16. Ela é comumente chamada de abordagem de tunelamento (*tunneling approach)*.

A maneira mais freqüente para implementar o tunelamento de pacotes LonTalk pelas redes IP é através da utilização de datagramas UDP como protocolo de transporte, o qual apresenta vantagens sobre o TCP/IP (SOUCEK; SAUTER, 2004). Ou seja, apesar de usualmente se falar em encapsulamento de pacotes LonTalk sobre IP, na prática eles são encapsulados em datagramas UDP, e estes sim sobre o IP. A Figura 17 oferece uma ilustração.

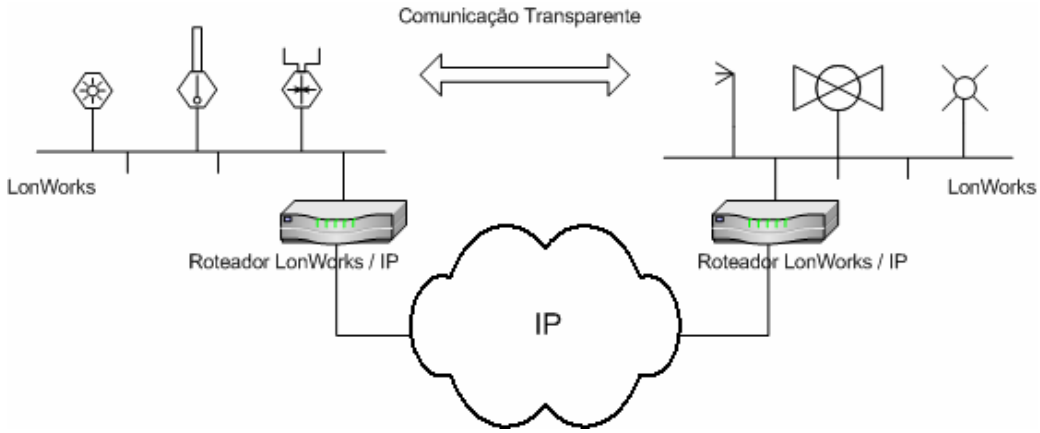


Figura 16. Rede IP servindo como meio para tunelamento de pacotes LonTalk.

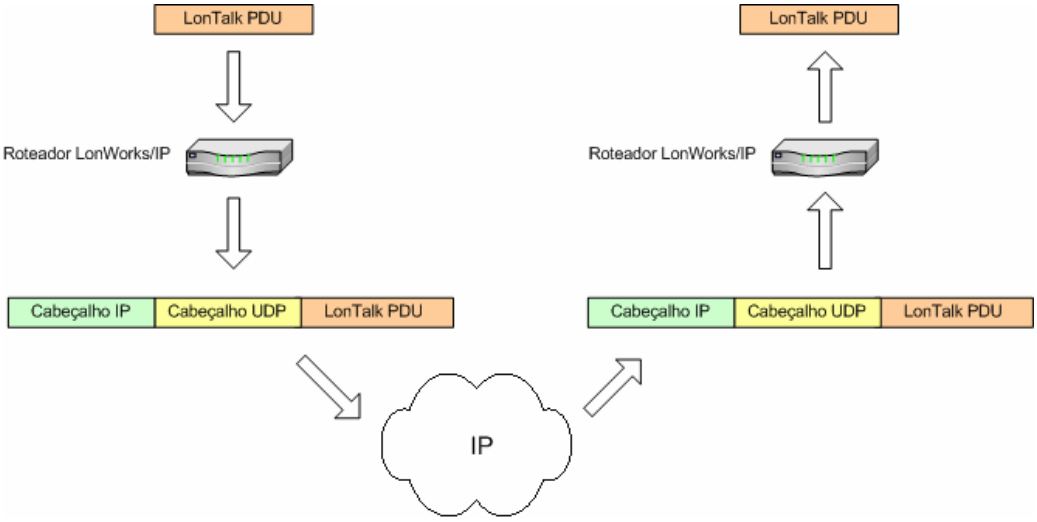


Figura 17. Tunelamento de pacotes LonTalk através de UDP/IP.

Soucek e Sauter (2004) destacam três vantagens do uso do UDP em relação ao TCP para executar o transporte de pacotes LonTalk via redes IP. A primeira vantagem refere-se à natureza não orientada à conexão das redes de controle. Usar o TCP no lugar do UDP aumentaria desnecessariamente o *overhead*, como por exemplo, durante o estabelecimento de conexões TCP. A segunda vantagem, também relacionada ao aumento de *overhead*, diz respeito às retransmissões que o TCP utiliza para garantir a entrega correta dos dados. Aplicações de controle que possuem requisitos críticos de tempo real requerem que a entrega de pacotes ocorra em intervalos máximos de tempo. Quando uma retransmissão ocorre,

provavelmente esse tempo máximo já foi esgotado e, portanto, a retransmissão seria inútil. A terceira vantagem aparece devido ao fato de que os protocolos de redes de controle geralmente já possuem o seu próprio esquema de retransmissões, normalmente mais adequados para aplicações de controle que o esquema do TCP. Este é justamente o caso do LonTalk, como pode ser conferido no Apêndice B. Sendo assim, já que o próprio protocolo LonTalk se encarrega das retransmissões, não é necessário usar o mecanismo do TCP para isso.

A norma EIA-852, sob o título *Tunneling of Component Network Data Over IP Channels*, foi definida com o objetivo de especificar um padrão para o tunelamento de pacotes de redes *fieldbus* sobre redes IP. Entre essas redes *fieldbus*, encaixa-se a norma EIA-709, isto é, a tecnologia LonWorks. EIA-852 descreve os componentes de sistema, o protocolo de comunicação e requisitos de gerenciamento para estabelecer um canal IP com dispositivos de controle e roteadores. Os dispositivos de um sistema baseado na EIA-852 podem ser puramente baseados em IP (apresentando conector Ethernet, usualmente) ou serem nativos de alguma rede de controle (apresentar conector para par trançado LonWorks). A rede de controle é conectada à rede IP através de roteadores.

Esta idéia viabiliza a implementação de dispositivos LonWorks com conector Ethernet (meio físico usual para redes IP) e sem conector para meios físicos convencionais (par trançado, rede elétrica, etc.), reforçando o fato de as redes IP serem consideradas meios físicos. Echelon (2002) apresenta uma relação de todos os meios físicos suportados pela tecnologia LonWorks. Cada meio é identificado por uma sigla, como por exemplo, TP/FT-10 para par trançado de topologia livre a 78kbps. Nesse contexto, os canais IP para redes LonWorks são conhecidos oficialmente pela sigla IP-852.

É possível implementar uma solução LonWorks puramente IP-852 (Figura 18) ou mista (Figura 19), além de, obviamente, apenas com os meios físicos tradicionais sem envolver canais IP.

Na Figura 19, enquanto parte dos dispositivos estão fisicamente conectados a redes IP (IP-852), outros estão conectados a redes LonWorks TP/FT-10. Os roteadores LonWorks/IP implementam o tunelamento definido pela norma EIA-852 e garantem a transparência da solução do ponto de vista dos nós.

Cada dispositivo LonWorks baseado em IP-852 possui, obviamente, um endereço IP. No entanto, nós LonWorks devem se endereçar pelas maneiras definidas pelo protocolo LonTalk (conforme apresentado na seção 4.2.3 e no Apêndice B). Sendo assim, esses dispositivos também possuem seus próprios Neuron IDs e estão configurados em algum

domínio, sub-rede e nó. A Figura 19 ilustra este fato através da indicação de endereço ao lado de cada dispositivo IP-852.

É papel dos roteadores LonWorks/IP converter os possíveis tipos de endereços previstos pelo LonTalk, como por exemplo o Neuron ID, no correspondente endereço IP de um nó IP-852. Isso é necessário para entregar correta e transparentemente os pacotes LonTalk encapsulados em datagramas UDP/IP. Para desempenhar esta tarefa, os roteadores LonWorks/IP implementam tabelas internas e capacidade de processamento, como ocorre em qualquer tipo de roteador de qualquer tipo de rede.

O processamento envolvido na descoberta de endereços e na transmissão de pacotes entre portas de um roteador introduz latência (atraso) na comunicação ponto-a-ponto entre dispositivos, a qual deve ser levada em conta no projeto do sistema (GAW; MARSH, 1997).

O encapsulamento de pacotes LonTalk em datagramas UDP/IP contempla desde o cabeçalho de nível 2 (*Layer 2 Header*) até o campo de CRC (inclusive). Não inclui o preâmbulo, *BitSync*, *ByteSync*, nem a violação *Line-Code* (ECHELON, 2002). Maiores detalhes sobre esses campos são apresentados no Apêndice B e Toshiba (1995).

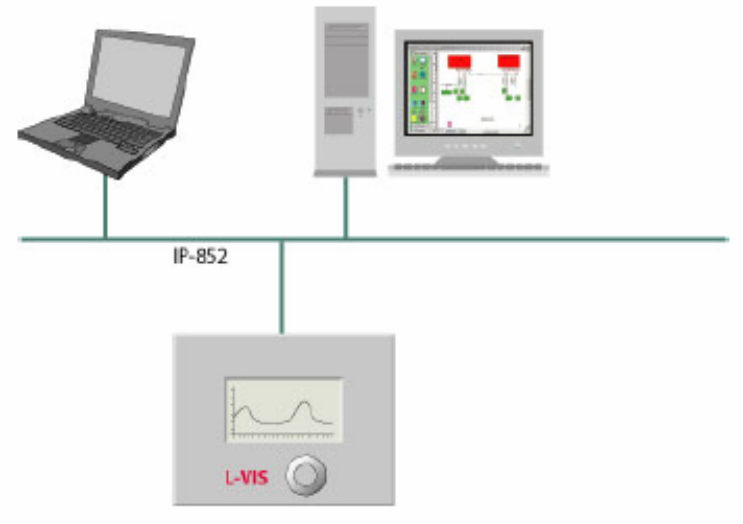


Figura 18. Solução LonWorks puramente baseada em IP-852.
Extraído de Loytec (2003).

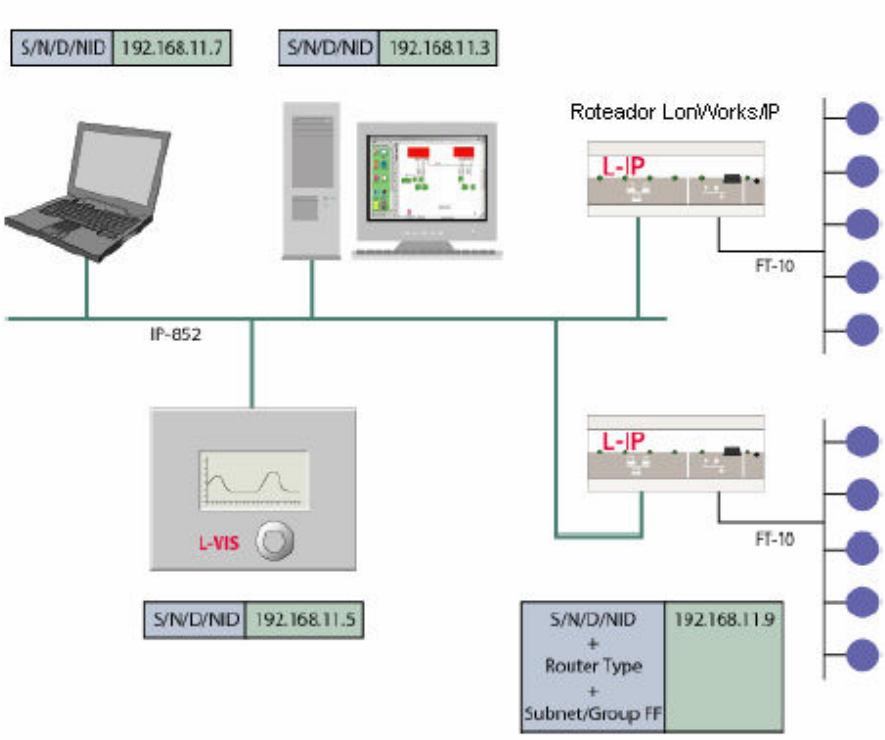


Figura 19. Solução LonWorks baseada em IP-852 e TP/FT-10.
Baseado em Loytec (2003).

Echelon (2002) ilustra o exemplo da inserção de uma informação de uma NV de temperatura em um pacote LonTalk, e este, por sua vez, em um datagrama IP. Esta ilustração pode ser encontrada na Figura 20 e funciona como elemento didático, não mostrando todos os detalhes e nem mesmo explicitando o protocolo UDP.

Dispositivos LonWorks baseados em IP-852, inclusive os próprios roteadores LonWorks/IP, usam as portas padronizadas 1628/udp e 1628/tcp para a comunicação na rede IP. Todos os dispositivos devem suportar o UDP, conforme a norma EIA-852. O suporte ao TCP é opcional para certos tipos de mensagem, e funciona em adição ao UDP (ECHELON, 2002).

Mesmo adotando o UDP no lugar do TCP, o *overhead* introduzido no tunelamento de pacotes LonTalk ainda é grande. O tamanho típico de um pacote LonTalk gira em torno de 10 bytes. A adição dos cabeçalhos UDP, IP e Ethernet gera mais de 30 bytes de *overhead*, isto é, mais que o triplo da informação efetiva que se deseja transmitir. Uma das maneiras de reduzir este *overhead* é através do conceito de *packet bunching*, isto é, introduzir mais de um pacote LonTalk em um mesmo datagrama UDP.

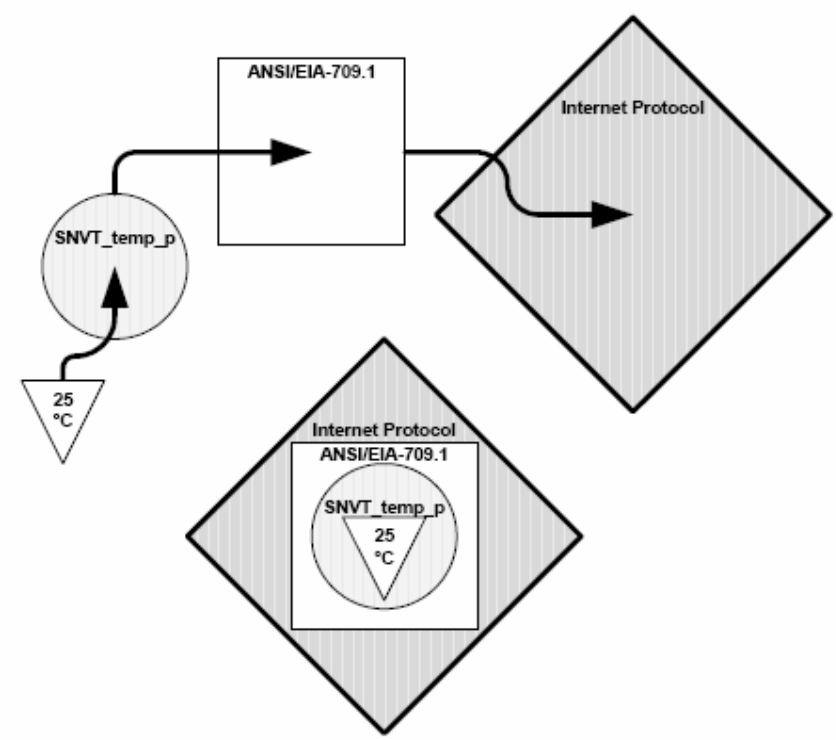


Figura 20. Encapsulamento de valor de NV em datagrama IP.
Extraído de Echelon (2002).

O *packet bunching*, por sua vez, exige que o roteador espere um certo período de tempo após a chegada de um pacote para verificar se novos pacotes chegarão, de forma a conseguir agregá-los em um único datagrama. Isso introduz mais atraso na comunicação e deve ser considerado no projeto de aplicações.

5.2.4 **Gateway**

Enquanto roteadores oferecem conectividade em nível de pacotes do protocolo do *fieldbus* (LonTalk nesse caso), os *gateways* provêem acesso através de outro formato (GAW; MARSH, 1997). Um *gateway* LonWorks pode ser definido como um dispositivo que converte o protocolo LonTalk para um outro protocolo. A definição de Soucek e Sauter (2004), mais refinada, estabelece genericamente que um *gateway* rede de controle / IP traduz os elementos da camada de aplicação (camada 7) do protocolo da rede de controle em um protocolo

proprietário ou padrão, também da camada 7, utilizável em redes IP e na Internet. Um *gateway* também pode ser capaz de realizar a tradução inversa. Considerando os infinitos graus de liberdade para se projetar abstrações e escolher protocolos, muitas soluções específicas têm sido propostas.

Considerando-se o modelo conceitual do roteador LonWorks/IP definido na seção 5.2.3, tem-se que ele encapsula pacotes LonTalk em datagramas UDP/IP, disponibilizando e transmitindo informações sobre uma rede LonWorks para elementos receptores de uma rede IP, sejam nós IP-852 ou outros roteadores LonWorks/IP. O receptor deve possuir o protocolo LonTalk implementado para entender essas informações e conseguir trabalhar com as mesmas. Imagine-se agora que o roteador, ao invés de simplesmente encapsular os pacotes LonTalk sobre IP, passasse a disponibilizar acesso à rede LonWorks através de outros protocolos, tais como em forma de *home-pages* elaboradas na linguagem *HyperText Markup Language* (HTML) acessíveis por HTTP. Nesse caso, passa-se a ter um *gateway* e não mais um roteador, uma vez que haveria conversão entre protocolos de camada de aplicação. Usuários de computadores ligados ao *gateway* através de uma rede IP, seja LAN ou WAN, poderiam acessar a rede LonWorks de uma maneira totalmente transparente, bastando digitar o endereço IP do *gateway* em seu navegador de Internet. Não haveria necessidade de instalação ou configuração de software por parte do usuário. As configurações apropriadas deveriam ser realizadas apenas no próprio *gateway*, sendo tarefa do responsável pela instalação e manutenção da rede LonWorks.

A abordagem dos *gateways* é comumente utilizada para integrar redes LonWorks com sistemas de informação da empresa, ao contrário da abordagem dos roteadores, cuja principal aplicação é prover conectividade entre redes LonWorks geograficamente distantes.

As configurações que devem ser realizadas nos *gateways* normalmente são mais complicadas e trabalhosas que as dos roteadores. Isso porque é necessário definir um mapeamento entre o protocolo LonTalk e o formato ou protocolo utilizado no lado da rede IP. Além do *gateway* LonWorks/HTTP, um outro exemplo seria o *gateway* LonWorks/SOAP. Informações sobre nós da rede seriam acessíveis através de Web Services (ANANTHAMURTHY, 2004), padrão de integração conhecido e aceito no mundo dos sistemas corporativos. Os Web Services disponibilizados por esses *gateways* seriam uma boa opção para a implementação de soluções de acesso a redes LonWorks por parte de sistemas de informação, cujos desenvolvedores não precisariam ter conhecimento sobre o protocolo LonTalk.

Gateways LonWorks também são muito utilizados para aplicações de monitoração e supervisão à distância (acesso remoto), onde um protocolo mais especializado e conveniente

faz a comunicação entre o *gateway* e as estações de monitoramento. Um exemplo pode ser conferido na Figura 21, cujo *gateway* LonWorks/IP é implementado através de um software em um computador com uma placa PCLTA. Uma estação remota é capaz de visualizar o valor lido por um sensor de temperatura localizado na rede LonWorks, disponibilizado por uma NV. O *gateway* possui um banco de dados no qual foi configurado previamente um mapeamento que associa a NV em questão a um componente gráfico a ser exibido em uma página HTML.

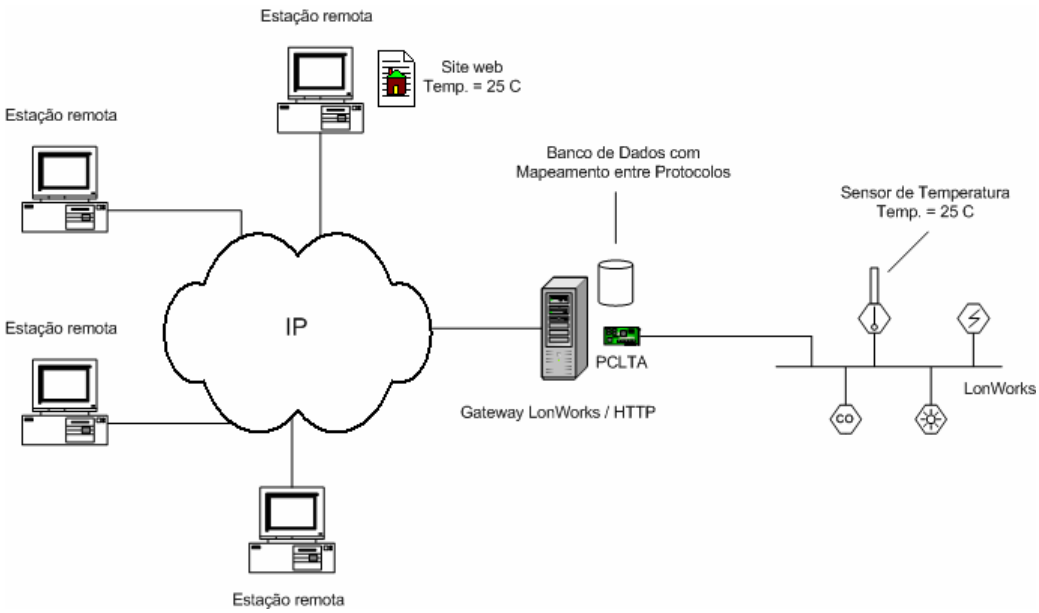


Figura 21. Monitoração remota de rede LonWorks via HTTP.

Utilizar um roteador torna necessário que essas estações possuam o protocolo LonTalk implementado e, portanto, as atribui uma certa carga de processamento devido ao fato de as mesmas terem que processar e interpretar os pacotes da rede de controle. Ou seja, devem funcionar como funcionam os dispositivos IP-852. Com a utilização da abordagem de *gateway*, usa-se um protocolo mais adequado, tal como o HTTP (no qual interfaces e telas são enviadas à estação de monitoramento através de HTML) e apenas a informação útil para o monitoramento trafega pela rede IP. Em geral, as estações remotas apenas carregam e apresentam telas, avisos e alarmes ao usuário, podendo também fornecer opções para que este envie algum comando de atualização de NV ou de gerenciamento. O protocolo LonTalk não

precisa mais, portanto, ser implementado pelas estações remotas. Na verdade, essas não precisam nem ter o conhecimento de que a solução é implementada com uma rede LonWorks.

Uma outra funcionalidade interessante que pode ser implementada em *gateways* para acesso remoto é a possibilidade de acesso discado. Para aplicações onde não há uma conexão IP permanentemente disponível, o *gateway* pode ser capaz de aceitar uma chamada discada e executar, por exemplo, o *Point-to-Point Protocol* (PPP) como camada de enlace para o protocolo IP, recaindo então nos exemplos já apresentados. Alternativamente, o *gateway* acessado por linha discada pode até suportar outros protocolos que não baseados em IP, mais otimizados para as características da conexão disponível. Esse tipo de *gateway* pode não apenas receber chamadas, mas também realizá-las. Ele pode periodicamente (ou em situações consideradas anormais) discar para uma central e transmitir as devidas informações sobre a rede LonWorks. Assim, tem-se uma solução adequada para uma aplicação onde não se dispõe de uma conexão IP permanente, sendo capaz de, autonomamente, informar situações de emergência a uma central sem requerer que seja acessada periodicamente pela mesma.

O *gateway* apropriado deve ser escolhido convenientemente conforme o tipo de aplicação. Vale destacar que não existem apenas *gateways* para interconectar redes LonWorks e redes IP. Essa abordagem também é aplicada para interconectar diferentes tipos de redes de controle. Gaw e Marsh (1997) citam o exemplo de um *gateway* LonWorks/Modbus, o qual apresenta um conector físico para rede LonWorks (ex.: TP/FT-10) e outro para Modbus. Os elementos de dados lógicos e básicos da tecnologia LonWorks, isto é, as variáveis de rede (NVs), são mapeados em endereços Modbus. NVs podem ser lidas ou alteradas por elementos Modbus através de comandos de leitura e escrita de registradores. Apesar da possível dificuldade e complexidade na realização da configuração do *gateway* (mapeamento entre protocolos), uma vez que ela foi realizada, dispositivos LonWorks podem se comunicar com dispositivos Modbus transparentemente. Permite-se o desenvolvimento de soluções híbridas e, principalmente, o aproveitamento de instalações legadas.

5.3 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo apresentou os tipos de abordagem existentes para implementar a interconexão entre redes LonWorks e redes IP. Antes de prosseguir com a análise das quatro categorias de aplicação citadas no Capítulo 1, é necessário introduzir alguns conceitos e definições de parâmetros de qualidade de serviço, o que é feito no Capítulo 6.

6 SISTEMAS DE TEMPO REAL E PARÂMETROS DE QoS

6.1 SISTEMAS DE TEMPO REAL

A computação de tempo real diz respeito a sistemas de hardware e software que possuem restrições de tempo para execução de tarefas. A partir da ocorrência de um certo evento, há um intervalo máximo limite de tempo para que o sistema dê sua resposta. Diz-se que a computação falhou se a mesma não conseguiu cumprir esse período máximo, que também é chamado de *deadline*. Os *deadlines* são independentes da carga de uso do sistema, isto é, não importa se o sistema ainda apresenta capacidade ociosa ou se está plenamente ocupado: os *deadlines* não são função desta carga.

Um exemplo de sistema de tempo real é o freio ABS (*Anti-lock Braking System*) de um carro. A restrição de tempo real é que toda a computação envolvida termine antes do tempo em que os freios devem ser liberados para evitar que as rodas travem. Um servidor de rede, por outro lado, é um sistema em que se deseja respostas rápidas, mas não há *deadline*. Se o mesmo estiver com alta carga de uso, a resposta ficará mais lenta, mas não há falha no serviço. No caso do freio ABS, se o *deadline* não for cumprido, haverá uma falha grave no serviço, possivelmente catastrófica.

Computação de tempo real é comumente confundida com computação de alto desempenho. No entanto, os mesmos exemplos servem para mostrar a diferença. Servidores de rede com baixa carga de programas em execução podem utilizar praticamente todo o seu potencial para dar respostas extremamente rápidas, mas ainda assim não precisam cumprir *deadlines*. O sistema do freio ABS, no entanto, pode não ser tão poderoso quanto o servidor de rede, mas foi especialmente desenvolvido para cumprir o *deadline* envolvido.

Apesar da diferença conceitual, sistemas de tempo real podem ser implementados através de sistemas de alto desempenho com limite de carga. Desta maneira, conhecendo-se a carga máxima à qual o sistema será submetido, é possível garantir tempos de respostas máximos para os eventos em questão. Certificando-se de que esses tempos máximos são menores ou iguais que os *deadlines* requeridos, tem-se então um sistema de tempo real.

Sistemas de tempo real podem ser classificados em duas categorias básicas: *hard real time* e *soft real time*. Na primeira, incluem-se os sistemas de tempo real que ocasionam falhas críticas quando o *deadline* não é cumprido. Na segunda, o não cumprimento do *deadline* causa uma degradação no serviço, mas não gera uma falha crítica.

Marca-passos de coração e controladores de determinados processos industriais são, por razões óbvias, exemplos de sistemas com requisito *hard real time*. Por outro lado, uma interface de rede de computador é um exemplo de sistema *soft real time*. Se a interface não conseguir tratar a fila de pacotes em seu *buffer* em um tempo adequado, ela pode perder novos pacotes que chegam da rede. Esses pacotes são retransmitidos pelos protocolos que se costumam utilizar e, portanto, serão recebidos novamente. O efeito geral é uma degradação na qualidade do serviço, mas não acontecem falhas graves.

Do ponto de vista de redes, é conveniente considerar dois subconjuntos dos sistemas *soft real time* para este trabalho, conforme Soucek e Sauter (2004). O primeiro subconjunto corresponde às redes com entrega garantida (*guaranteed delivery*), isto é, aquelas em que o atraso na entrega fim-a-fim de um pacote é irrelevante. O importante é que o pacote seja recebido corretamente em seu destino. O segundo subconjunto é chamado de melhor esforço (*best effort*). Aqui, a rede tenta entregar os pacotes ao seu destino o mais rápido possível, mas sem garantir o tempo em que isso será feito ou mesmo se o pacote será entregue com sucesso.

Aplicações *hard real time* que dependem da comunicação oferecida por redes de comutação de pacotes não determinísticas tipicamente impõem um limite máximo permitido para o atraso fim-a-fim (SOUCEK; SAUTER, 2004). Seja D a variável aleatória representando este atraso (*delay)* e $f(D)$ sua função de densidade de probabilidade, tem-se:

$$f(D) \equiv 0 \,, \qquad \text{para } D \geq D_{\text{máx}} \tag{1}$$

Ou seja, a probabilidade de ocorrer um certo atraso $D \geq D_{\text{máx}}$ é nula.

No caso de aplicações *soft real time* de entrega garantida, não há restrição para a função densidade de probabilidade $f(D)$. Não é necessário que a mesma possua um $D_{\text{máx}}$ a partir do qual seja nula. Se possuísse, recairia na definição de *hard real time*.

Para o subconjunto de redes *soft real time* de melhor esforço, deve-se tratar $f(D)$ como uma distribuição condicional. Sendo X o evento que denota a entrega com sucesso de um pacote, pode-se considerá-lo independente da distribuição do atraso. Portanto:

$$f(D) = f(D \mid X)\, p(X) \tag{2}$$

Onde $p(X)$ é a probabilidade de ocorrência do evento X. Integrando-se ambos os lados de (2), tem-se:

$$\int_0^{+\infty} f(D)dD = p(X)\int_0^{+\infty} f(D \mid X)dD \qquad (3)$$

Porém, a integral que aparece no segundo membro da equação (3) é igual a 1, já que se o evento *X* ocorre, então o pacote chegou e obrigatoriamente o atraso está entre 0 e +∞. Tem-se, portanto:

$$\int_0^{+\infty} f(D)dD = p(X) \qquad (4)$$

Quando há algum risco de perda de pacote (cenário do esquema de melhor esforço), então $p(X) < 1$ e, portanto, a probabilidade de o pacote ser perdido está implicitamente incluída na função de densidade de probabilidade *f(D)*.

Cada aplicação baseada em redes LonWorks integradas com redes IP apresenta seus próprios requisitos específicos e individuais. No entanto, esses requisitos podem ser encaixados de maneira geral em um dos três grupos apresentados aqui: *hard real time*, *soft real time* de entrega garantida e *soft real time* de melhor esforço.

O desempenho das redes LonWorks e IP implementadas como solução para uma certa aplicação, que inclui o desempenho dos elementos que promovem sua interconexão, é fundamental na determinação do sucesso da aplicação como um todo. Após a apresentação dessas definições de requisitos de aplicações de tempo real, a seção 6.2, a seguir, mostra definições de parâmetros de Qualidade de Serviço, ou *Quality of Service* (QoS) que possibilitam estabelecer formalmente requisitos a serem atendidos pelas redes e elementos que compõem a solução implementada. Nas seções 7.1 e 7.2, apresenta-se quais desses requisitos aparecem em cada uma das quatro categorias de aplicações levantadas na seção 1.2. Essas relações entre aplicações e requisitos servem como um caminho para o projeto de uma solução de sucesso.

6.2 PARÂMETROS DE QUALIDADE DE SERVIÇO (QoS)

6.2.1 **Introdução**

Parâmetros de qualidade de serviço (QoS) vêm sendo muito relevantes em aplicações de transporte de conteúdo multimídia pela Internet. Na verdade, ganharam atenção especial justamente devido a esse tipo de aplicação. Esses parâmetros servem para mensurar a qualidade com que as transmissões são feitas e o impacto no sucesso da aplicação, haja vista as dificuldades e problemas de se utilizar uma rede de comutação de pacotes não determinística para transmitir dados com características de fluxo contínuo (*streams*) que apresentam fortes restrições relacionadas principalmente a perdas, atraso e variação de atraso.

Em sistemas de controle baseados em redes, têm-se problemas semelhantes às aplicações multimídia. Conexões que antes eram feitas diretamente agora estão sendo substituídas por redes de comutação de pacotes (SOUCEK; SAUTER; KOLLER, 2003), como por exemplo LonWorks e LonWorks sobre Ethernet/IP. A semelhança com as características de transporte de dados multimídia torna-se visível quando se considera aplicações de laços de controle, onde controladores devem constantemente enviar sinais de atuação para atuadores, e sensores devem realimentar os controladores com informações obtidas da planta. A característica de transmissão de um fluxo contínuo de dados (*stream*) com fortes requisitos de tempo é evidente. A diferença fundamental é que em aplicações multimídia essas transmissões usualmente envolvem grandes quantidades de dados por unidade de tempo e mensagens longas, e, em controle, tem-se uma menor quantidade de dados e mensagens curtas.

Apresenta-se, a seguir, definições de parâmetros de QoS utilizadas em Soucek e Sauter (2004), Soucek, Sauter e Koller (2002), e Soucek, Sauter e Rauscher (2001). As seções 7.1 e 7.2 irão posteriormente relacionar esses parâmetros com as categorias de aplicações da seção 1.2.

Considera-se que o cenário ao qual essas definições se aplicam é um conjunto com um número arbitrário de redes LonWorks e IP interconectadas por roteadores ou *gateways*. Não se faz restrição há um meio físico específico, tanto para as redes LonWorks quanto para as redes IP. Como caso particular, as definições se aplicam também em cenários onde existem apenas redes IP ou apenas redes LonWorks.

6.2.2 **Atraso fim-a-fim (delay)**

Seja P um caminho entre um ponto de origem e um ponto de destino. P é composto por enlaces (*links*) físicos e por roteadores que os interconectam, denotados respectivamente por $E(P)$ e $V'(P)$. A razão desta nomenclatura ficará clara na seção 7.2. Considera-se que um pacote p_k é gerado no ponto de origem e deve trafegar até o ponto de destino através do caminho P. Seja $d_{k,e}$ o atraso no enlace e e $s_{k,n}$ o tempo total de espera em fila mais processamento do pacote p_k no elemento n, pode-se então representar através da equação (5) o atraso fim-a-fim D_k da transmissão de p_k.

$$D_k = \sum_{e \in E(P)} d_{k,e} + \sum_{n \in V'(P)} s_{k,n} \tag{5}$$

Se o elemento n converter o pacote p_k para um outro protocolo antes de repassá-lo adiante, então trata-se de um *gateway* (e não de um roteador), e, nesse caso, considera-se que o tempo de processamento envolvido na conversão de protocolos no elemento n está incluído em $s_{k,n}$.

Observa-se que a primeira parcela do segundo membro de (5) é a parte determinística do atraso fim-a-fim, enquanto que o segundo termo é estatístico e varia conforme a carga nos roteadores. Esta, por sua vez, depende principalmente da taxa de utilização da rede.

Já foi visto na equação (1) um possível requisito para este parâmetro de qualidade de serviço. Para aplicações *hard real time*, o atraso não pode ser maior que um certo atraso máximo $D_{\text{máx}}$. De (1), pode-se concluir (6), onde p representa a probabilidade.

$$p(D_k > D_{\text{máx}}) = 0 \tag{6}$$

No entanto, quando se trabalha com redes em que não se pode garantir um limite máximo de tráfego, deve-se considerar que elas podem ficar congestionadas arbitrariamente e, portanto, o tempo de fila dos pacotes nos roteadores pode aumentar indefinidamente. Dessa forma, não se pode garantir com certeza a condição da equação (6) e, por conseqüência, aplicações *hard real time* ficam comprometidas. Isso acontece quando se utiliza redes públicas como a Internet, ou redes de terceiros, onde não se tem controle completo sobre o tráfego nos elementos de $V'(P)$.

Para aplicações *soft real time*, pode ser desejável estabelecer o requisito da equação (7).

$$p(D_i \leq D_{máx}) \geq Z_{min} \quad (7)$$

Isso mostra que um certo atraso máximo $D_{máx}$ nem sempre precisa ser respeitado para um pacote qualquer p_i, mas é imposta uma probabilidade mínima Z_{min} para que o atraso não supere $D_{máx}$. Em outras palavras, um atraso maior que $D_{máx}$ é tolerado se ocorrer com probabilidade menor que $1 - Z_{min}$. Pode-se notar que o caso particular no qual $Z_{min} = 1$ é equivalente à equação (6).

O requisito da equação (7) implica no estabelecimento de um perfil para a função densidade de probabilidade do atraso *f(D),* que deve ser cumprido pela rede que implementa a aplicação em questão. Sabendo-se que:

$$p(D_i \leq D_{máx}) = \int_0^{D_{máx}} f(D)dD \quad (8)$$

Então, através da substituição de termos em (7), tem-se:

$$\int_0^{D_{máx}} f(D)dD \geq Z_{min} \quad (9)$$

A atribuição de valores às variáveis $D_{máx}$ e Z_{min} de (9) estabelece um requisito para o atraso da rede, que pode ser utilizado tanto para verificar se uma rede já existente o atende quanto para se projetar uma nova infra-estrutura adequada à aplicação.

No primeiro caso, é necessário levantar a distribuição do atraso na rede utilizada. Não é uma tarefa simples e, além do que, essa distribuição pode variar no tempo, sendo muito comum, por exemplo, a variação sazonal ao longo dos dias da semana e ao longo dos horários de um mesmo dia em redes IP corporativas. Diversos autores propõe técnicas para esse levantamento, como por exemplo Tsang, Coates e Nowak (2003). Apesar de ser possível a obtenção de resultados específicos que descrevem satisfatoriamente redes privadas (onde muitos de seus aspectos podem ser controlados, tais como número de usuários, topologia, protocolos permitidos, etc.), é extremamente difícil descrever a Internet em termos de modelos e distribuições de probabilidades (PAXTON; FLOYD, 1997). Mesmo assim, aproximações podem ser feitas para situações particulares.

A Figura 22 apresenta um exemplo de função de densidade de probabilidade do atraso em uma certa rede, supondo que não há perdas. Pacotes são enviados da origem ao destino em intervalos de tempo T. Há um atraso mínimo D_{min} antes do qual a rede jamais conseguirá entregar o pacote. Após o mesmo, o pacote pode chegar em qualquer instante. Observa-se que há um pico e que, em seguida, a densidade de probabilidade tende a zero quando o atraso tende a infinito, formando uma "cauda".

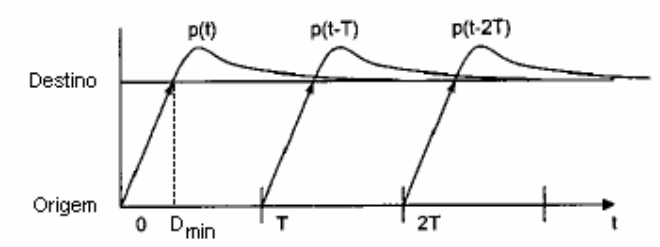


Figura 22. Exemplo de função de densidade de probabilidade do atraso.
Baseado em Soucek, Sauter e Rauscher (2001).

6.2.3 **Variação do atraso (jitter)**

A variação do atraso fim-a-fim, também conhecida como *jitter* e denotada por J_k, é comumente definida como a parcela estatística de D_k, isto é:

$$D_k = D_{min} + J_k \quad (10)$$

$$\therefore J_k = D_k - D_{min} \quad (11)$$

A equação (10) foi obtida reescrevendo-se a equação (5) com a substituição de sua parte determinística por D_{min} e da parte estatística por J_k, a qual refere-se ao *jitter* do pacote p_k.

O *jitter* é função de diferentes tempos de processamento nos elementos de *V'(P)* para diferentes pacotes que por ali passam em diferentes instantes de tempo. Essa diferença de tempo de processamento decorre principalmente dos diferentes tamanhos de fila que são encontrados nos roteadores a cada instante, que por sua vez são função da taxa de utilização da rede.

Caso se considere redes com topologias onde existem múltiplos caminhos entre uma certa origem e um certo destino, diferentes pacotes de uma mesma seqüência transmitida

podem tomar diferentes rotas dependendo das políticas e decisões dos roteadores. Um pacote p_{k+1} pode seguir por uma certa rota e chegar rapidamente ao seu destino, enquanto o pacote imediatamente anterior da seqüência transmitida, p_k, pode ter sido encaminhado por uma rota mais congestionada e chegar ao destino somente após p_{k+1}.

A equação (10) veio de (5), cuja formulação considerou um certo caminho P. Porém, é possível manter a definição de (10) mesmo quando existem múltiplas rotas entre uma origem e um destino. Para isso, faz-se uma abstração: o receptor não tem conhecimento sobre qual rota um pacote tomou até atingi-lo. Se existirem m caminhos P_1, P_2, P_3, ... , P_m entre a origem e o destino, então tem-se m valores para o termo $D_{\min}$ da equação (10), um para cada caminho P_i (com $1 \leq i \leq m$). Eles serão denotados por $D_{\min,1}$, $D_{\min,2}$, $D_{\min,3}$, ... , $D_{\min,m}$. Um determinado pacote p_k será roteado por um algum caminho P_i e apresentará o atraso total D_k, ocasionando o *jitter* J_k:

$$J_k = D_k - D_{\min,i} \tag{12}$$

J_k depende, através do termo $D_{\min,i}$, do caminho P_i pelo qual p_k foi roteado. Para facilitar o raciocínio, está-se considerando que o conjunto de caminhos $\{P_1, P_2, P_3, \ldots, P_m\}$ pode ser abstraído e encarado como um único caminho "lógico" P_L que liga a origem ao destino, cujo termo $D_{\min,L}$ é igual a $min(D_{\min,1}, D_{\min,2}, D_{\min,3},\ldots, D_{\min,m})$. Esta abstração pode ser visualizada através da Figura 23, que ilustra um exemplo com três caminhos possíveis entre nós de duas redes.

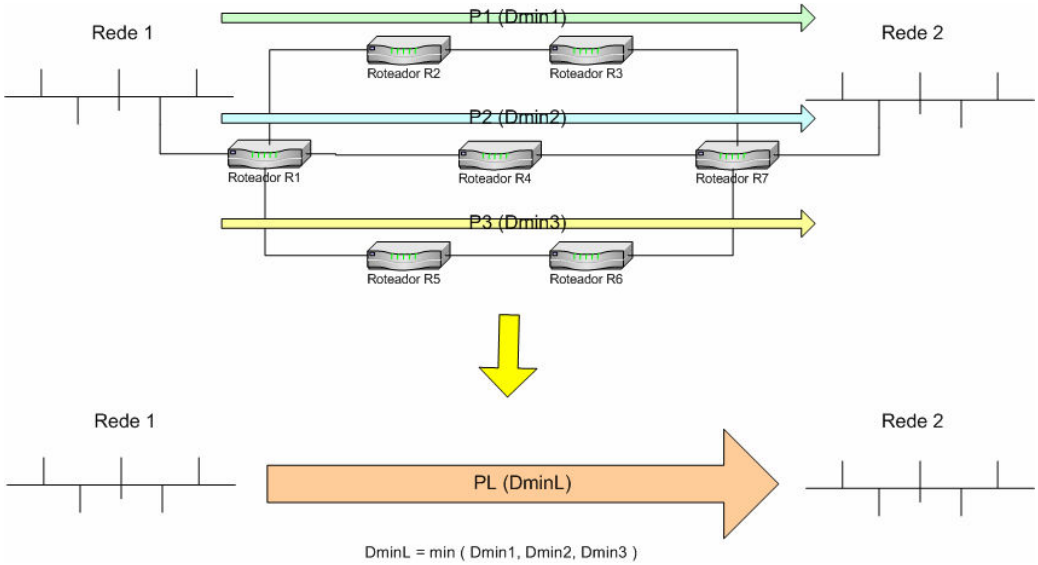


Figura 23. Exemplo de abstração de um caminho representando múltiplos caminhos.

Do ponto de vista do receptor, não importa por qual caminho P_i o pacote p_k chegou. Basta considerar que todo pacote chega através do único caminho lógico P_L existente entre a origem e o destino onde as equações (10) e (11) se aplicam com $D_{\min} = D_{\min,L}$. Como este é igual ao menor valor do conjunto $\{D_{\min,1}, D_{\min,2}, D_{\min,3},..., D_{\min,m}\}$, então pode-se considerar que o termo J_k da equação (10) absorve a diferença entre $D_{\min,i}$ e $D_{\min,L}$ quando o caminho utilizado P_i não corresponde ao $D_{\min,i}$ mínimo.

Aplicações podem apresentar o requisito para limitar o *jitter* a um valor máximo. No entanto, pelas mesmas razões já explicadas para o atraso fim-a-fim, pode ser impossível garantir um *jitter* máximo em todas as situações. Busca-se então, se a aplicação assim o permitir, aliviar este requisito através do estabelecimento de um *jitter* máximo que é respeitado com uma probabilidade mínima $U_{\min}$.

$$p(J_i \leq J_{máx}) \geq U_{\min} \tag{13}$$

Através de (11), o requisito da equação (13) pode ser convertido em um requisito para o atraso. De (11)**,** pode-se encontrar a relação entre um *jitter* qualquer J_i e um certo atraso D_i, lembrando que $D_{\min}$ é uma constante dependente do caminho P. No caso de múltiplos caminhos existentes, basta admitir que $P = P_L$ e $D_{\min} = D_{\min,L}$. Substituindo-se em (13), tem-se:

$$p(D_i - D_{\min} \leq J_{máx}) \geq U_{\min} \tag{14}$$

$$\therefore p(D_i \leq J_{máx} + D_{\min}) \geq U_{\min} \tag{15}$$

Por outro lado, considerando-se que:

$$p(D_i \leq J_{máx} + D_{\min}) = \int_0^{J_{máx}+D_{\min}} f(D)dD \tag{16}$$

Então:

$$\int_{0}^{J_{máx}+D_{\min}} f(D)dD \geq U_{\min} \qquad (17)$$

Pode-se, portanto, concluir que o requisito definido em (13) implica em uma nova restrição para o perfil da função densidade de probabilidade do atraso dada por (17), independentemente do fato de a mesma já possuir alguma outra eventual restrição, como por exemplo proveniente da equação (9).

A título de exemplo, será considerada uma aplicação que impõe os seguintes requisitos na comunicação entre dois certos nós N_1 e N_2:

$D_{máx}$ = 200 ms
$Z_{\min}$ = 0,95
$J_{máx}$ = 60 ms
$U_{\min}$ = 0,85

Supõe-se também que existe um único caminho de roteamento entre N_1 e N_2, no qual $D_{\min}$ = 50 ms. De acordo com (9) e (17), é imposto um perfil à função de densidade de probabilidade do atraso entre N_1 e N_2, o qual é apresentado na Figura 24.

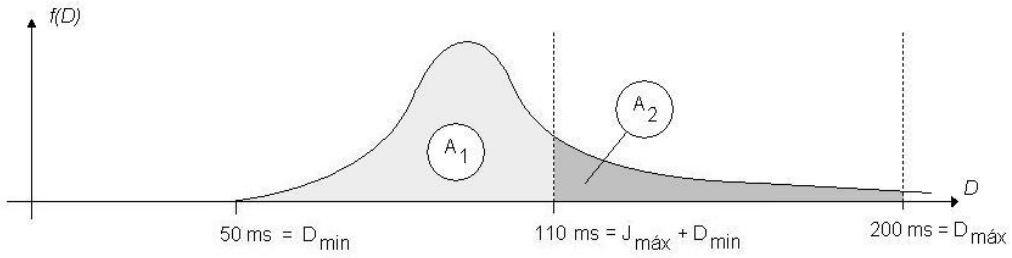


Figura 24. Perfil de função de densidade de probabilidade do atraso de uma aplicação.

A_1 e A_2 representam as áreas indicadas abaixo da curva de densidade de probabilidade. Os requisitos são atendidos se $A_1 \geq 0{,}85$ e $A_1 + A_2 \geq 0{,}95$ (ou $A_2 \geq 0{,}1$).

Quando mais de uma rota existe entre os nós, observa-se que a função de densidade de probabilidade do atraso apresenta mais de um pico (SOUCEK; SAUTER; KOLLLER, 2002). Em um cenário onde há dois caminhos possíveis, os pacotes irão atravessar a rede através de P_1 e P_2, cada um com sua respectiva função de densidade de probabilidade. A função resultante para o caminho P_L é uma combinação apresentando dois picos, não

necessariamente de mesma amplitude. A razão entre as amplitudes depende da proporção da quantidade de pacotes que é roteada entre os dois caminhos. A Figura 25 apresenta um exemplo de duas funções combinadas onde dois caminhos similares são selecionados com a mesma probabilidade.

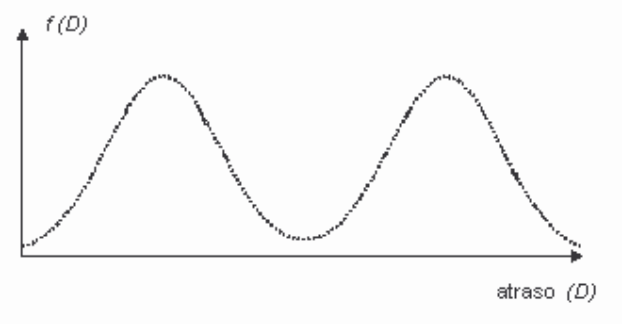


Figura 25. Perfil de função de densidade de probabilidade combinada para dois caminhos.

6.2.4 **Taxa de transmissão (throughput) e largura de banda (bandwidth)**

A taxa de transmissão ou *throughput* do caminho *P* pode ser definida como a capacidade total que ele tem em processar e transmitir dados durante um determinado período de tempo. Matematicamente, o *throughput* Θ no intervalo $(t,t+\Delta t)$ pode ser calculado por (18), onde *A(t)* denota a quantidade total de dados transferidos de um instante inicial t_0 até t.

$$\Theta(t, t+\Delta t) = \frac{A(t+\Delta t) - A(t)}{\Delta t} \qquad (18)$$

A taxa de transmissão é importante para aplicações que transmitem fluxos de dados contínuos com peridiocidade bem definida entre pacotes, como é o caso de aplicações de controle e de multimídia em tempo real.

O *throughput* instantâneo $\Theta(t)$ é definido como o limite de (18) com Δt tendendo a zero. Matematicamente, este limite não existe, pois a transmissão digital não é um fenômeno contínuo. Entretanto, pode-se abstrair e admitir que Δt é suficientemente pequeno comparado ao período de tempo de observação mas suficientemente grande que seja possível medir $\Theta(t)$ através de (18).

Seguindo a mesma linha dos requisitos definidos para os parâmetros anteriores, uma aplicação pode impor uma probabilidade mínima ζ_{min} para o *throughput* se manter acima de um limiar inferior Θ_{min}, o que é denotado por (19).

$$P(\Theta \geq \Theta_{min}) \geq \zeta_{min} \quad (19)$$

Commweb (2002) define as diferenças entre os termos *bandwidth* (largura de banda) e *throughput*. O primeiro se refere à capacidade total de transmissão do canal, enlace ou circuito de comunicações. Originário do contexto analógico, onde é medido em Hz, no contexto digital ele é usualmente medido em bps (por exemplo, 100Mbps para Fast-Ethernet). O segundo termo, *throughput*, diz respeito à taxa de transmissão efetivamente ocorrida na comunicação entre dois nós da rede. Essa grandeza tem por objetivo medir a quantidade real de dados úteis que trafegam em uma comunicação ponto-a-ponto por unidade de tempo.

O *throughput* nunca pode ser maior que a largura de banda. Na prática, ele é sempre menor e essa diferença é determinada por diversos fatores, dentre os quais pode-se citar alguns:

- todos os tipos pacotes, tanto de redes de dados quanto de redes de controle, envolvem algum *overhead* relacionado a cabeçalhos e *trailers* que os compõem. Esse *overhead* consome largura de banda na transmissão, sem, no entanto, contabilizar *throughput* (não são dados úteis para a aplicação);
- muitos protocolos envolvem *overhead* adicional ligado a confirmações de recebimento, estabelecimento de conexões lógicas e outros tipos de mensagens de controle. O LonTalk, por exemplo, envolve transmissão de ACKs nos serviços *acknowledged* e *request/response*;
- pacotes corrompidos na transmissão são descartados e podem requerer retransmissões, as quais consomem largura de banda mas não representam dados úteis adicionais a serem contabilizados no *throughput*;
- em momentos de grande utilização da rede, os tamanhos das filas nos elementos de comutação crescem. Quando não há mais memória nessas filas, pacotes são descartados e, dependendo do protocolo, podem requerer retransmissões.

Orlando (2004) estabelece uma relação entre a largura de banda B_e de um enlace e e o atraso $d_{k,e}$ que um pacote p_k leva para percorrê-lo. Seja l_e o atraso de propagação do sinal no

enlace, que depende do meio físico e da distância, e sz_k o tamanho do pacote (em bits), tem-se:

$$d_{k,e} = l_e + \frac{sz_k}{B_e} \tag{20}$$

A interpretação de (20) é que o atraso em um enlace é a soma do tempo de propagação do sinal com o tempo que o transmissor leva para colocar o sinal físico correspondente ao pacote no canal.

A equação (20), quando substituída em (1) permite relacionar um requisito de atraso à largura de banda:

$$D_k = \sum_{e \in E(P)} \left(l_e + \frac{sz_k}{B_e} \right) + \sum_{n \in V'(P)} s_{k,n} \tag{21}$$

O seguinte exemplo será considerado: dada uma infra-estrutura de rede, os termos l_e e B_e são conhecidos. Conhecendo-se as características dos elementos de comutação, é possível também determinar os $s_{k,n}$ mínimos (que ocorrem quando as filas nos elementos $n \in V'(P)$ estão vazias). Utilizando-se então um valor máximo coerente para sz_k (o valor máximo representa o pior caso), que depende da aplicação em questão, é possível calcular D_k através de (21). D_k é o atraso mínimo a que um pacote p_k é submetido, fim-a-fim, para o pior caso (maior tamanho de pacote). Se este atraso mínimo for superior a um certo requisito $D_{\text{máx}}$ (ver equação (7)), então a infra-estrutura de rede analisada não consegue atender a aplicação.

Para simplificar (21), em alguns casos pode-se desprezar alguns termos. Por exemplo, se os enlaces que compõem a infra-estrutura de rede forem baseados apenas em redes locais Fast-Ethernet, os termos l_e podem ser desprezados. Por outro lado, se a rede contar com algum enlace de satélite, então não se pode desprezá-los.

6.2.5 **Taxa de perdas**

Pacotes podem ser perdidos ao trafegarem de uma certa origem a um certo destino por um determinado caminho P. Essa perda pode ser causada por erros de comunicação seguidos de descartes propositais, como acontece, por exemplo, quando ocorrem erros de *checksum* em quadros de uma rede Ethernet. No entanto, a razão mais comum para a perda de pacotes é o

congestionamento das redes, que provoca grandes filas nos roteadores e ocasiona o descarte de novos pacotes que chegam e não encontram espaço no *buffer* da fila.

A fração dos pacotes que são entregues ao seu destino com sucesso, independentemente do atraso, é denotada por w. A taxa de perdas é definida como 1 - w.

Uma aplicação pode exigir uma probabilidade mínima $W_{\min}$ para que um pacote seja devidamente entregue ao seu destino sem ser perdido.

$$P(\text{pacote recebido}) \geq W_{\min} \tag{22}$$

6.2.6 **Seqüência de pacotes**

Algumas aplicações podem requerer que a seqüência de pacotes recebida por um destino seja obrigatoriamente a mesma seqüência gerada pela origem. Seja $\langle p_k \rangle$ a seqüência transmitida e $\langle p'_k \rangle$ a seqüência recebida, este requisito pode ser representado por (23).

$$\langle p_k \rangle = \langle p'_k \rangle \qquad \forall k \tag{23}$$

A diferença da ordem dos pacotes entre a seqüência recebida e a seqüência transmitida decorre do *jitter*. A existência de múltiplos caminhos entre uma origem e um destino pode possibilitar que pacotes gerados seqüencialmente sejam enviados por diferentes rotas, dependendo das decisões e políticas dos roteadores. Um pacote p_{k+1} pode seguir por uma certa rota e chegar rapidamente ao seu destino, enquanto o pacote imediatamente anterior da seqüência transmitida, p_k, pode ter sido encaminhado por uma rota mais congestionada e chegar ao destino somente após p_{k+1}.

Para relacionar a seqüência de pacotes com o *jitter*, está-se considerando dois pacotes consecutivos, p_k e p_{k+1}, enviados de uma origem a um destino. Seja t_k o instante em que p_k é enviado e $t_{k+1} = t_k + T_s$ o instante de envio de p_{k+1}. De acordo com (10), eles chegam ao destino respectivamente nos instantes $a_k = t_k + D_{\min} + J_k$ e $a_{k+1} = t_{k+1} + D_{\min} + J_{k+1}$. Os pacotes p_k chegarão fora de ordem quando $a_{k+1} < a_k$, ou seja:

$$t_{k+1} + D_{\min} + J_{k+1} < t_k + D_{\min} + J_k$$
$$\therefore\ t_k + T_s + D_{\min} + J_{k+1} < t_k + D_{\min} + J_k$$
$$\therefore\ \cancel{t_k} + T_s + \cancel{D_{\min}} + J_{k+1} < \cancel{t_k} + \cancel{D_{\min}} + J_k$$

$$\therefore\ T_s + J_{k+1} < J_k$$
$$\therefore\ J_k - J_{k+1} > T_s \qquad (24)$$

A equação (24) estabelece, portanto, uma condição para que a seqüência de pacotes enviados chegue quebrada no destino. No caso mais desfavorável para que essa quebra ocorra, tem-se que $J_{k+1} = 0$. Na verdade, J_{k+1} sempre será maior que zero pois os termos $s_{k,n}$ da equação (5) sempre envolvem algum tempo de processamento mesmo que a fila no elemento *n* esteja vazia. Entretanto, supõe-se J_{k+1} desprezível a ponto de simplificar a análise e igualá-lo a 0 no pior caso. Substituindo este valor na equação (24), obtém-se:

$$J_k > T_s \qquad (25)$$

A equação (25) representa um requisito para o *jitter* para que não ocorra quebra na seqüência de pacotes recebidos. Se já há algum outro requisito para o *jitter* proveniente de (13), esses requisitos devem ser combinados de modo que a solução atenda a ambos.

6.3 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo apresentou uma série de definições e conceitos relacionados com sistemas de tempo real e parâmetros de QoS, necessários para embasar o texto subseqüente. Apresentaram-se definições úteis de sistemas de tempo real e estabeleceu-se uma série de parâmetros de QoS comuns que podem ser utilizados para definir requisitos de aplicações que envolvem interconexão entre redes LonWorks e IP.

Tendo já visto as abordagens de interconexão entre redes LonWorks e redes IP no Capítulo 5, o Capítulo 7 relacionará as quatro categorias de aplicações referidas no Capítulo 1 com as abordagens mencionadas e os parâmetros aqui apresentados.

7 REQUISITOS E APLICAÇÕES DA INTERCONEXÃO ENTRE REDES LONWORKS E REDES IP

7.1 SISTEMAS SUPERVISÓRIOS E GERENCIAMENTO DE REDES LONWORKS VIA REDES IP

7.1.1 **Introdução**

Sistemas supervisórios, também designados pela sigla SCADA (*Supervisory Control and Data Acquisition*), são softwares conectados a uma instalação de automação que tem por objetivo apresentar a um operador humano uma Interface Homem-Máquina (IHM) informando status e valores de variáveis relacionadas a instrumentos na planta. Também pode possibilitar a interferência do operador humano através da mesma interface, através de elementos de atuação (CASTRUCCI; MORAES, 2001).

Os sistemas supervisórios modernos são os substitutos naturais dos painéis sinópticos, isto é, painéis com indicadores luminosos, *displays* e botões, cuja função também é prover um meio de supervisão e atuação na planta por parte de operadores humanos. Apesar disso, não se observa uma substituição completa dos painéis sinópticos pelos sistemas supervisórios. Os painéis são ainda muito utilizados, e, em alguns casos, até preferidos. Por exemplo, costuma-se usar painéis sinópticos de fácil acesso e visualização para monitorar sistemas de detecção de incêndio, já que em situações de emergência seria mais difícil se o bombeiro tivesse que procurar por um computador executando um software supervisório para saber sobre a situação do local monitorado.

Apesar de ser possível encontrar no mercado implementações conceitualmente incorretas, a idéia é que a automação da planta jamais esteja implementada no sistema supervisório. O processamento da automação deve ser realizado pelos controladores centralizados ou pelos nós distribuídos, conforme o paradigma adotado (o último é mais condizente com a filosofia LonWorks). Assim, o computador que executa o software supervisório pode ser tranqüilamente desligado que a automação da planta permanece funcionando. Se o processamento da automação fosse incorporado no software supervisório, um simples travamento do PC poderia causar a interrupção do funcionamento de todo o sistema.

Sistemas supervisórios para redes LonWorks seguem a mesma definição, com a restrição apenas de que se referem a aplicações de supervisão e atuação sobre redes LonWorks.

Os sistemas supervisórios atuais são baseados em IHMs de fácil operação, eficientes e ergonômicas (CASTRUCCI; MORAES, 2001), cuja necessidade surgiu para facilitar o trabalho da equipe encarregada da operação do sistema. O operador de um sistema supervisório observa sistematicamente os indicadores da interface, obtendo uma visualização sintética sobre o estado geral do processo ou de parte dele. Nem todos os parâmetros são observados com a mesma freqüência, pois:

- alguns parâmetros são mais sintéticos do que os outros, fornecendo informações mais gerais;
- alguns aparelhos são mais estáveis que outros, os quais se desregulam ou quebram com maior freqüência;
- algumas desregulagens são mais críticas que outras, implicando que os parâmetros associados devem receber maior atenção;
- alguma unidade específica está em uma fase de operação particular, como por exemplo em situações de conserto, o que pode requerer atenção diferenciada do operador.

O conhecimento e experiência que o operador possui em relação ao funcionamento da planta condicionam diretamente sua forma de vigilância através do sistema supervisório. Esses aspectos podem ser levados em conta no projeto do sistema para garantir melhor usabilidade.

Uma parte importante dos sistemas supervisórios são os alarmes. Alarmes são notificações de que certos eventos, possivelmente críticos, ocorreram na planta monitorada. Normalmente, requerem atenção prioritária por parte do operador. Os alarmes são configurados nos processos de instalação e manutenção do sistema supervisório, devendo-se levar em conta as condições de acionamento, escolha e notificação de operadores, envio de mensagens e providência de ações.

A ocorrência de alarmes costuma implicar no surgimento de indicadores bem destacados na tela do operador, com o intuito de chamar sua atenção. Recursos animados (como por exemplo, sinais piscantes) ou até mesmo sinalizações sonoras são aplicadas para indicar de maneira rápida e clara o seu nível de importância e prioridade.

Ainda de acordo com Castrucci e Moraes (2001), os alarmes são chamados de normais (ou pré-alarmes) quando não requerem necessidade alguma de intervenção do operador. Os alarmes críticos usualmente requerem um procedimento de intervenção que pode estar incluído dentre os seguintes:

- supressão de sinal sonoro indicando o reconhecimento do alarme;
- intervenção diretamente na tela do terminal do software supervisório, também indicando reconhecimento por parte do operador;
- aceitação do alarme, indicando que o operador sabe da existência do problema mas no momento não pode realizar ação alguma relacionada com esse fato;
- não reconhecimento por parte do operador.

Conforme introduzido na seção 4.6 e melhor detalhado no Apêndice B e em Toshiba (1995), o conceito de gerenciamento e diagnóstico de redes LonWorks é definido através das mensagens e funcionalidades estabelecidas e suportadas pelo protocolo LonTalk. Assim como um sistema supervisório em operação normal possibilita a leitura de NVs de uma instalação LonWorks, bem como a alteração de valores de NVs de entrada (seja local ou remotamente, conforme será melhor explicado na seção 7.1.2), pode-se imaginar que o gerenciamento e diagnóstico é possível por mecanismos semelhantes. Enquanto em um sistema supervisório se recebe e envia valores de NVs, em um aplicativo de gerenciamento se recebe e envia mensagens de gerenciamento e diagnóstico. Em razão desta similaridade, sistemas supervisórios e de gerenciamento LonWorks via rede IP foram agrupados neste trabalho. A seção 7.1.6 aborda mais especificamente o gerenciamento.

7.1.2 **Arquitetura da solução**

Sistemas supervisórios de redes LonWorks que funcionam através de redes IP podem ser implementados tanto com a abordagem de roteador quanto com a abordagem de *gateway*.

Na abordagem de roteador, o sistema supervisório deve ser capaz de se comunicar diretamente pelo protocolo LonTalk, ou seja, o computador que executa o software supervisório é enxergado externamente como um nó IP-852. O software supervisório interpreta os pacotes LonTalk recebidos e extrai destes as informações úteis a serem exibidas ao usuário. A arquitetura desta solução pode ser visualizada na Figura 26.

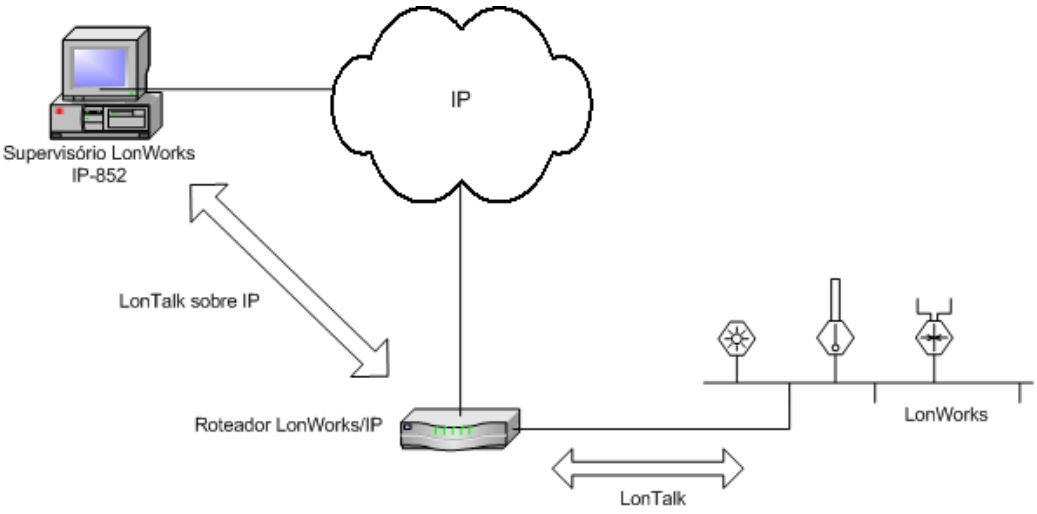


Figura 26. Sistema supervisório com arquitetura baseada em roteador.

É fácil observar que um software supervisório, por mais básico que seja, precisa possuir um banco de dados ou algum outro meio para gravar configurações. Dentre essas configurações, é interessante que o usuário possa criar suas próprias IHMs e determinar quais valores devem ser exibidos em cada uma delas.

Normalmente, deseja-se também que o software supervisório seja capaz de interferir na rede LonWorks, como por exemplo ao permitir que o usuário solicite alterações de valores de NV (como por exemplo, *set-point* de um controlador). Nesse caso, o software supervisório deve gerar pacotes LonTalk adequados e os enviar via rede IP de acordo com a EIA-852.

O banco de dados do software supervisório também serve como meio para o armazenamento de histórico de valores de NVs, de informações de diagnóstico da rede monitorada, de intervenções de operadores (sejam comandos simples ou respostas a alarmes), entre outros.

Apesar da possibilidade desta solução, normalmente é mais conveniente implementar sistemas supervisórios com a abordagem de *gateway*. Enquanto os roteadores LonWorks/IP provêem conectividade em nível de pacotes do protocolo LonTalk, os *gateways* apresentam a vantagem de oferecer conectividade a uma rede LonWorks através de um protocolo de mais alto nível (GAW; MARSH, 1997). Esta é uma característica interessante para sistemas supervisórios, pois o tráfego na rede IP pode ser otimizado se apenas a informação útil para este tipo de aplicação for enviada e recebida.

Conforme já adiantado na seção 5.2.4 e já ilustrado na Figura 21, nesta solução as estações de monitoração ligadas na rede IP acessam um dispositivo *gateway* conectado à rede

LonWorks. A Figura 27 provê outra ilustração. A sigla *XYZ* indica um protocolo qualquer utilizado pelo *gateway* e pelo sistema supervisório.

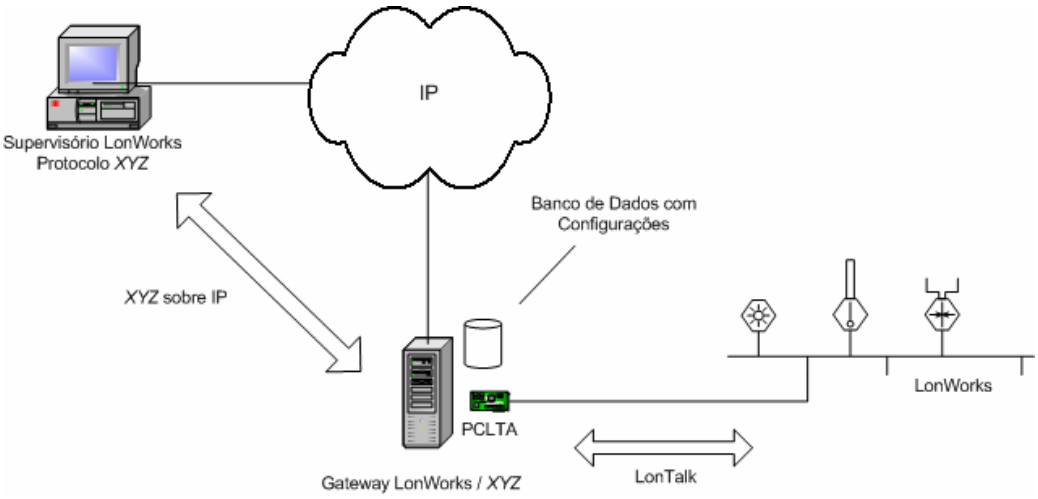


Figura 27. Sistema supervisório com arquitetura baseada em gateway.

De acordo com Soucek e Sauter (2004), a abordagem *gateway* traduz um protocolo de camada de aplicação (camada 7) de uma rede de controle (LonWorks, no caso) para um protocolo também de camada 7 utilizado na Internet e em Intranets. Dado o alto grau de liberdade para se criar abstrações, desenvolver e escolher protocolos, inúmeras soluções baseadas em *gateway* têm sido propostas. As seções 7.1.3 e 7.1.4 apresentam dois exemplos.

7.1.3 **RemoteLon**

O sistema RemoteLon, desenvolvido no trabalho Canovas e Chermont (2003), é um software supervisório básico para redes LonWorks compatível com a plataforma Microsoft Windows. Apresenta arquitetura baseada na abordagem de *gateway*. É composto por três aplicativos: o RemoteLon Server, o RemoteLon Client e o RemoteLon Server Administrator.

O RemoteLon Client é o aplicativo que apresenta a IHM para o usuário que deseja monitorar uma rede LonWorks. Ele pode ser executado em qualquer PC que seja capaz de endereçar e acessar o computador que execute o RemoteLon Server através de uma rede IP.

O RemoteLon Server é o aplicativo *gateway* em si. Deve ser executado em um PC com interface de rede IP (usualmente Ethernet) e uma interface de rede LonWorks. O

RemoteLon Client, que é o *front-end* do software supervisório, abre uma conexão TCP em uma porta pré-configurada com o RemoteLon Server. Sobre esta conexão TCP, é executado o protocolo RemoteLon Protocol (RLP), cuja especificação também faz parte do trabalho Canovas e Chermont (2003). Do lado da rede IP, portanto, a comunicação entre o RemoteLon Client e o RemoteLon Server se dá através do protocolo RLP. Do lado da rede LonWorks, o PC se comunica com os nós através do próprio protocolo LonTalk, utilizando para isso uma interface de rede LonWorks.

Todas as configurações do sistema, sejam aquelas relativas à conectividade LonWorks ou aquelas referentes às configurações de usuários (telas, senhas, permissões, etc.) ficam armazenadas em um banco de dados de exclusivo acesso do RemoteLon Server. Este se baseia nessas configurações para, através do protocolo RLP, encaminhar às estações RemoteLon Client apenas o conteúdo necessário.

Deve-se notar que se a rede IP do PC que executa o RemoteLon Server possuir uma NIC-IP instalada, o mesmo PC não precisa ter sua própria interface de rede LonWorks. Isso porque a NIC-IP funciona como se fosse esta interface (seção 5.2.2). Sendo assim, é possível executar o RemoteLon Server em um PC apenas com placa de rede Ethernet instalada e devidamente configurado com relação à NIC-IP presente na rede.

O RemoteLon Server é considerado um *gateway* pois realiza as devidas interpretações e conversões entre os protocolos RLP e LonTalk.

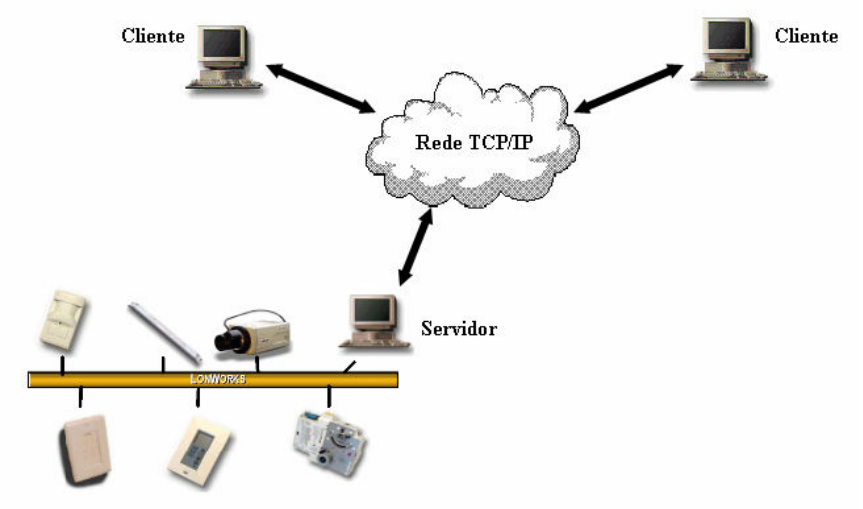


Figura 28. Arquitetura de rede do RemoteLon.
Extraído de Canovas e Chermont (2003).

O RemoteLon Server Administrator funciona da mesma maneira que o RemoteLon Client. Porém, seu objetivo é apresentar uma interface gráfica para configurações gerais do sistema, e não para supervisão dos elementos da rede LonWorks. Como já visto, essas configurações ficam em um banco de dados exclusivo do RemoteLon Server, que gerencia todo o acesso ao mesmo.

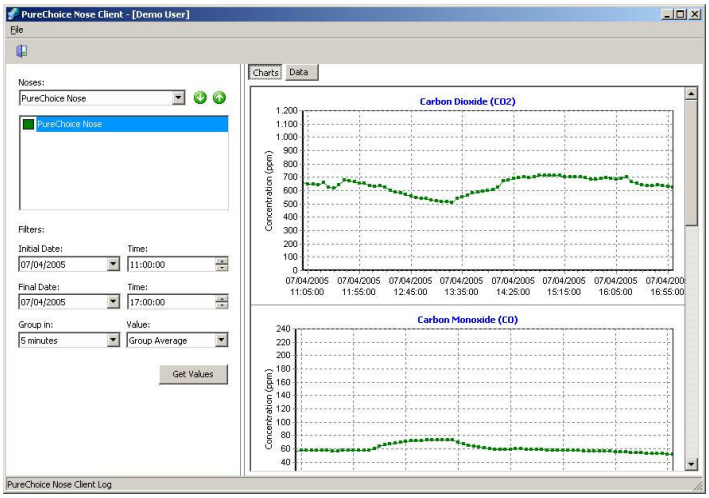
Canovas, Chermont e Cugnasca (2005) apresentam um exemplo de aplicação de monitoração remota de qualidade do ar com o RemoteLon. Algumas telas desta aplicação podem ser conferidas na Figura 29.

7.1.4 **TAC Vista**

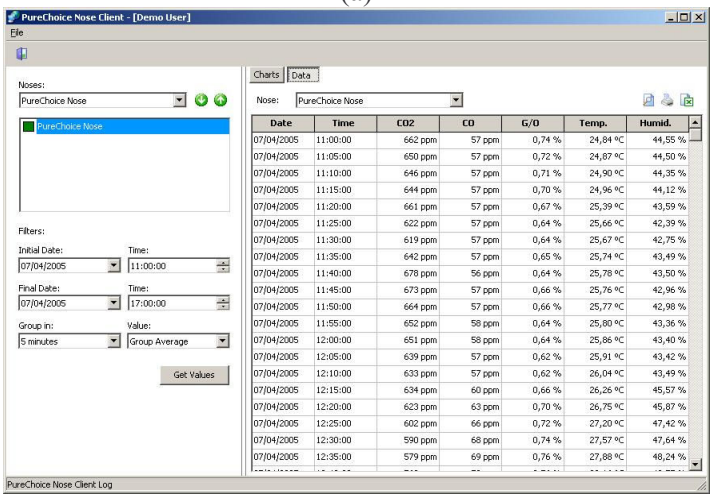
O sistema TAC Vista, da empresa TAC, também é um software supervisório baseado em plataforma Microsoft Windows. Permite a monitoração e operação de dispositivos de redes LonWorks através de interfaces gráficas. Possui foco na área de automação predial, sendo fácil a configuração e integração principalmente com os módulos comerciais da linha "Xenta" fabricados pela própria TAC (controladores, nós de entrada e saída, roteadores, etc.), todos compatíveis com LonWorks. Com estrutura modular, é flexível e adaptável em relação ao tamanho e funcionalidades da instalação (TAC, 2005).

O software supervisório TAC Vista funciona com a abordagem de *gateway* similar ao RemoteLon, mas é bem mais completo e apresenta inúmeras funcionalidades a mais. Mantidas as devidas proporções, o aplicativo TAC Vista Server corresponderia ao RemoteLon Server e o TAC Vista Workstation ao RemoteLon Client e RemoteLon Server Administrator.

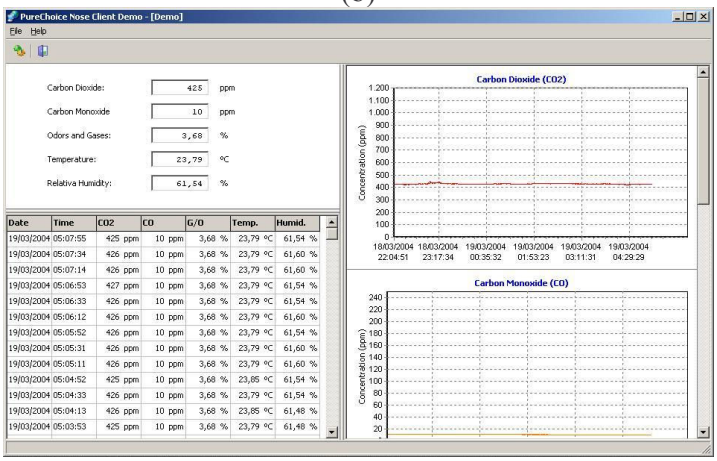
A comunicação entre o TAC Vista Server e o TAC Vista Workstation é realizada através de um protocolo proprietário. Um exemplo de tela do TAC Vista Workstation pode ser visto na Figura 30.



(a)



(b)



(c)

Figura 29. Telas do RemoteLon para aplicação de monitoração de qualidade do ar
Extraído de Canovas, Chermont e Cugnasca (2005)

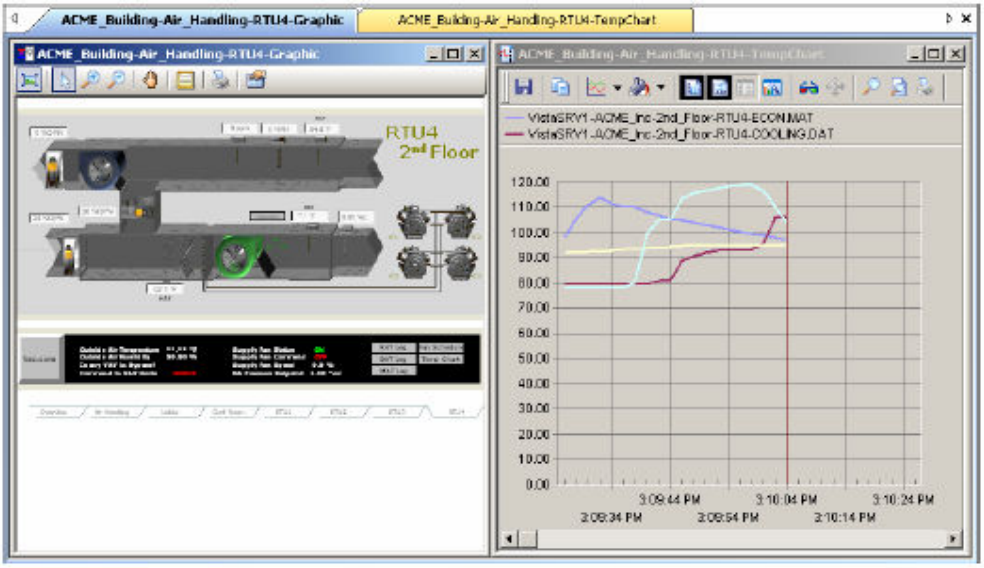


Figura 30. Tela de operação do TAC Vista Workstation.

7.1.5 **Web Services aplicados a sistemas supervisórios**

As arquiteturas orientadas a serviço, também designadas pela sigla SOA (*Service Oriented Architecture*), podem ser definidas como "arquiteturas de aplicações em que todas as funcionalidades são definidas como serviços independentes com interfaces bem definidas, as quais podem ser chamadas em seqüências pré-estabelecidas para formar processos de negócios" (CHATARJI, 2004). De uma maneira menos técnica, uma arquitetura orientada a serviço é formada por uma combinação de serviços, que são funcionalidades bem definidas e que não dependem do contexto ou estado de outros serviços.

A tecnologia dos Web Services, bastante falada atualmente, consiste em um conjunto de tecnologias, tais como *eXtensible Markup Language* (XML), *Simple Object Access Protocol* (SOAP), *Web Services Description Language* (WSDL) e *Universal Description, Discovery and Integration* (UDDI); elas são discutidas em mais detalhes em Ananthamurthy (2004), Chatarji (2004) e Papazoglou e Georgakopoulos (2003). Todas essa tecnologias viabilizam a implementação de sistemas baseados em arquitetura orientada a serviço, ou *Service Oriented Architecture* (SOA). O termo Web Services, além de designar esse conjunto de tecnologias, também serve para referenciar os próprios serviços baseados nelas. Web Services, portanto, é uma solução que implementa SOA, e não SOA em si, que se trata de um conceito teórico.

As mensagens de requisição e resposta de Web Services são baseadas no protocolo SOAP, que pode ser executado dentro de mensagens HTTP, FTP, SMTP. O mais comum é o HTTP.

Os Web Services podem ser utilizados para implementar um supervisório LonWorks. Na verdade, trata-se um caso particular da abordagem de *gateway*. As funcionalidades que o *gateway* de um software supervisório pode oferecer, tais como leitura de valor de NV, alteração de valor de NV, leitura do estado de um nó, etc. podem ser acessíveis através de Web Services (por exemplo, o SOAP via HTTP) padronizados.

A empresa Echelon disponibiliza a plataforma Panoramix (ECHELON, 2005d), que é uma API baseada em Web Services. Uma aplicação que deseja acessar alguma informação de uma rede LonWorks deve fazer uma requisição a um Web Service apropriado (exemplo: ler valor de NV), como se fosse uma chamada a uma rotina de API convencional.

A utilização de Web Services permite não só a construção de sistemas supervisórios LonWorks, mas também torna mais fácil a integração dessas redes com outros aplicativos, tais como aplicativos corporativos de *Enterprise Resource Planning* (ERP), *Customer Relationship Management* (CRM), *Business Intelligence* (BI), etc.

7.1.6 **Gerenciamento de redes LonWorks via IP**

Conforme já mencionado neste trabalho, a mesma arquitetura de *gateway* utilizada pelos exemplos RemoteLon e TAC Vista poderia ser empregada para a construção de sistemas de gerenciamento e diagnóstico remoto de redes LonWorks via redes IP. De maneira bem simplificada, bastaria considerar um software supervisório convencional e adaptá-lo de maneira a habilitá-lo a trabalhar com mensagens de gerenciamento e diagnóstico LonWorks. No caso do RemoteLon, o protocolo RLP passaria a incluir novos comandos e respostas relacionados ao suporte dessas mensagens. Uma adaptação adequada do RemoteLon Client e RemoteLon Server Administrator incluiria a criação de novas interfaces de usuário voltadas para gerenciamento e diagnóstico.

Na seção 4.6, foi visto que as mensagens "*Update Net Variable Config*" e "*Update Address*" do protocolo LonTalk entram em cena durante o processo de estabelecimento de um *binding* entre NVs. São exemplos de mensagens de gerenciamento suportadas pelo protocolo LonTalk. O protocolo RLP, se possuísse mecanismos adequados para dar suporte a essas funcionalidades, em conjunto com as interfaces do RemoteLon Client ou RemoteLon Server

Administrator devidamente adaptadas, poderiam prover uma funcionalidade para que um operador distante, via rede IP, implementasse *bindings* entre NVs.

Muitos aplicativos de gerenciamento remoto de redes LonWorks funcionam desta maneira. Se o *gateway* em questão for compatível com HTTP e disponibilizar interfaces *web* em HTML, funcionalidades de gerenciamento podem ser oferecidas para usuários remotos através de navegadores de Internet comuns. Os Web Services também são aplicáveis aqui, e pode-se construir serviços de gerenciamento de redes LonWorks baseados em SOAP.

Por outro lado, no mundo IP, já existe um protocolo bastante conhecido e utilizado para gerenciamento, o *Simple Network Management Protocol* (SNMP). Baseando-se neste protocolo e uma visão de gerenciamento de rede integrada, Kunes e Sauter (2001) propõem uma arquitetura de um *gateway* para gerenciamento de redes *fieldbus* em geral (não necessariamente LonWorks).

Até hoje o SNMP é o único protocolo para gerenciamento de redes e de seus elementos que atingiu grande aceitação entre diversas soluções proprietárias (KUNES; SAUTER, 2001). Originalmente, foi desenvolvido como uma solução de curto prazo até que outros protocolos se tornassem disponíveis. Entretanto, devido à sua grande simplicidade e baixa necessidade de recursos computacionais, ainda é o protocolo da maioria de dispositivos de redes locais que precisam prover ao menos algumas informações sobre seu *status*.

O SNMP estabelece um relacionamento cliente/servidor entre dispositivos. O gerente (*manager*) se comunica com diversos agentes (*agents*) para obter dados ou estabelecer novos valores em um dispositivo gerenciado. Nesse contexto, o gerente é o cliente e o agente é o servidor. Todos os dados acessíveis nos agentes são armazenados em uma estrutura com forma de árvore denominada *Management Information Base* (MIB), a qual fica localizada no próprio agente. Todos os dados que podem ser endereçados e manipulados por aplicações de gerenciamento devem fazer parte da MIB, na qual cada elemento é univocamente identificado por um código conhecido como *Object Identifier* (OID).

Se um elemento da rede não pode ser diretamente endereçável pelo protocolo SNMP, um agente *proxy* (*proxy agent*) provê a conversão entre os protocolos proprietários e o ambiente SNMP. Um exemplo desta situação é a própria tecnologia LonWorks. Dispositivos de uma rede LonWorks não podem ser diretamente endereçados por endereços IPs (redes IP são o ambiente natural do SNMP). Porém, um *gateway* LonWorks/IP pode ser endereçado neste contexto, e através de mensagens apropriadas consegue-se receber e enviar informações a um nó da rede LonWorks.

O SNMP é um protocolo que se encaixa na camada de aplicação da arquitetura TCP/IP (Figura 7). Foi desenvolvido para utilizar o protocolo UDP como meio de transporte. Apenas

cinco mensagens são definidas na comunicação entre o gerente e o agente pelo SNMP: *GetRequest, GetNextRequest* e *SetRequest* são enviadas pelo gerente e respondidas pelo agente com *GetResponse*. A mensagem *Trap* é enviada pelo agente e serve para que o mesmo comunique ao gerente a ocorrência imediata de algum evento importante sem que este último precise ficar periodicamente solicitando valores (abordagem *polling*). Uma mensagem *Trap* pode, portanto, ser comparada a uma interrupção de um microprocessador. Em instalações grandes e complexas, seu uso é melhor que o da abordagem *polling* no que diz respeito ao atraso e utilização de largura de banda (KUNES; SAUTER, 2001). A Figura 31 provê uma ilustração dessas mensagens.

Como é utilizado o UDP, e não o TCP, perdas de pacotes podem ocorrer e interferir no funcionamento do SNMP. Essas perdas usualmente são tratadas através de mecanismos de *timeout* e retransmissão pela aplicação de gerenciamento.

Uma hierarquia baseada em camadas foi definida pela *International Telecommunications Union* (ITU) na recomendação M.3100. Esta hierarquia apresenta definições para quatro categorias de gerenciamento de redes, conforme a pirâmide da Figura 32 (KUNES; SAUTER, 2001).

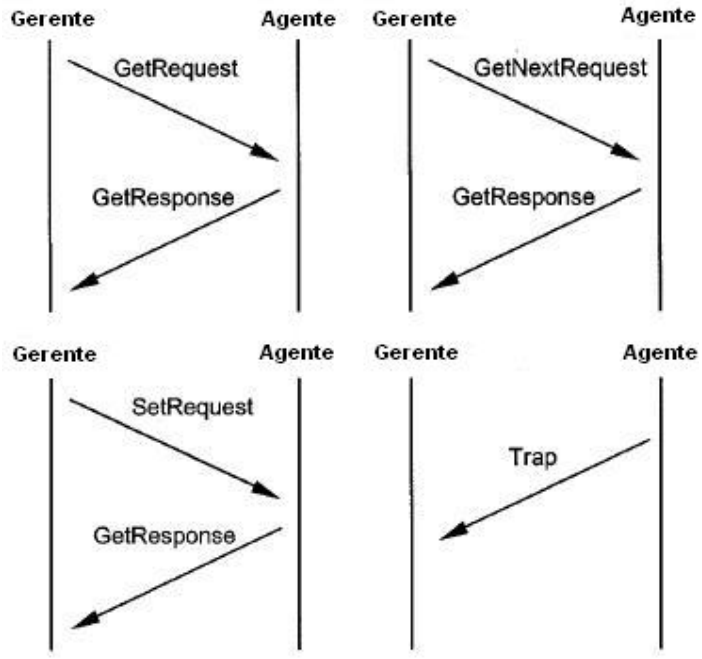


Figura 31. Mensagens do Protocolo SNMP.
Baseado em Kunes e Sauter (2001)

Figura 32. Hierarquia de gerenciamento definida pelo ITU.
Baseado em Kunes e Sauter (2001).

Enquanto a camada mais inferior refere-se ao gerenciamento de nós individuais da rede, a camada imediatamente superior diz respeito ao tratamento de uma rede como um todo. O gerenciamento de serviço combina informações em uma visão geral e se preocupa com o estado dos serviços oferecidos pela rede (por exemplo, impressão de documentos, autenticação de usuários, etc.). Por fim, o gerenciamento de negócio corresponde a uma abstração mais geral dos processos de uma empresa.

Ferramentas de gerenciamento específicas para redes *fieldbus* normalmente consideram apenas as duas camadas mais inferiores da Figura 32. A proposta de Kunes e Sauter (2001) consiste em tratar uma rede *fieldbus* (por exemplo, LonWorks) como um elemento individual de uma infra-estrutura de rede IP maior. O *fieldbus* é interconectado à rede IP através de um *gateway*, sendo considerado como um único nó da rede IP. Assim, através de ferramentas de gerenciamento SNMP já existentes, é possível gerenciar o *fieldbus* através da rede IP. A Figura 33 ilustra esta situação e exemplifica com um *fieldbus* LonWorks. Nesse caso, o *gateway* deve funcionar como um agente SNMP que disponibiliza acesso às NVs dos nós da rede (e outras de suas propriedades) em sua MIB. Mais especificamente e de acordo com a definição mencionada, o *gateway* é um agente *proxy*.

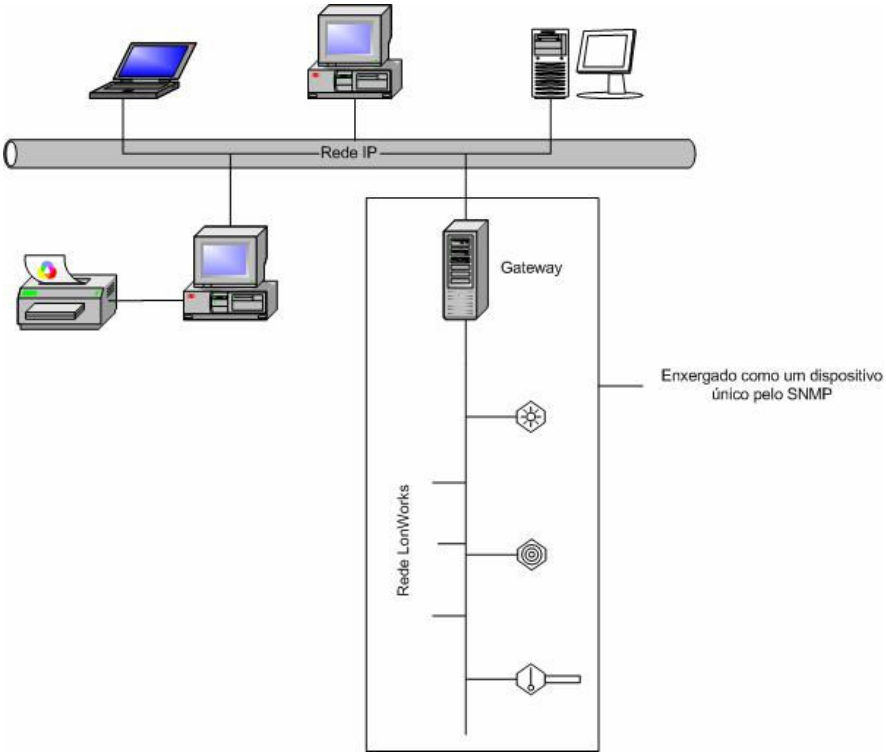


Figura 33. Rede LonWorks e gateway enxergados como nó individual pelo SNMP.

Na verdade, buscando simplificar a implementação, Kunes e Sauter (2001) propõem que o agente SNMP, implementado pelo *gateway*, seja dividido em dois: o agente mestre (*master agent*) e o subagente (*subagent*). Essa divisão é suportada pelo SNMP e tem como objetivo tornar a estrutura mais modular (por isso os agentes implementados dessa maneira são conhecidos como agentes modulares). A proposta do artigo é genérica em relação às redes *fieldbus* (não fala apenas sobre LonWorks). A intenção é que o agente mestre seja o agente principal SNMP visível pelos gerentes, enquanto cada subagente fica responsável pela interface com um tipo diferente de *fieldbus* e se comunique apenas com o agente mestre. A comunicação entre o mestre e os subagentes pode ser feita através de protocolos padronizados para agentes modulares, mas, no caso deste artigo, foi desenvolvido um protocolo específico pelo fato de os já existentes não atenderem às necessidades dos autores (KUNES; SAUTER, 2001). Essa estrutura modular pode ser conferida na Figura 34, juntamente com agentes convencionais (não modulares).

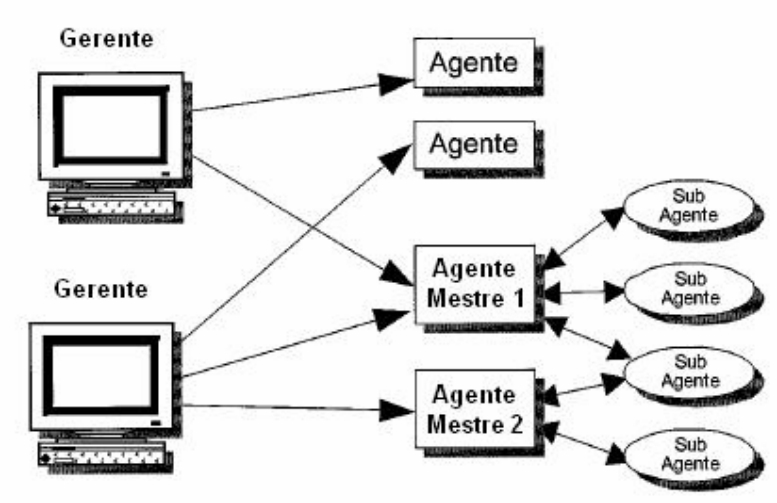


Figura 34. Agentes SNMP convencionais e modulares.
Baseado em Kunes e Sauter (2001).

A simplificação trazida por esta modularização dos agentes está no fato de o agente mestre poder ser o mesmo para diversos subagentes. Toda a implementação da comunicação SNMP com os gerentes fica no agente mestre. Quando for desejada a implementação de compatibilidade com um novo *fieldbus*, basta implementar um novo subagente. O desenvolvedor não precisa reimplementar o SNMP e pode focar em sua atividade específica, que corresponde aos detalhes de comunicação com o novo *fieldbus*. Maiores detalhes sobre essa arquitetura (tais como a localização e estrutura das MIBs envolvidas), bem como exemplos de implementações para alguns tipos de *fieldbus* podem ser encontrados em Kunes e Sauter (2001).

7.1.7 **Requisitos de QoS**

Para estabelecer quais requisitos de QoS uma determinada aplicação pode requerer, é necessário estudar seus objetivos, necessidades dos usuários, impactos em caso de falhas, e diversos outros fatores. Nesta seção considera-se especificamente sistemas supervisórios e de gerenciamento de redes LonWorks via redes IP.

Para os sistemas supervisórios, tipicamente alguns parâmetros de um processo são selecionados para serem visualizados em uma estação remota de monitoramento via rede IP, onde eles podem ser acompanhados e registrados. Esses parâmetros, em redes LonWorks, são NVs de nós distribuídos pela rede. Esse tipo de aplicação normalmente envolve interações

com usuários humanos, o que limita eventuais restrições de desempenho mínimo que refletiriam em requisitos fortes de QoS. Por exemplo, se o valor de uma certa NV existente em um processo cujas variações são da ordem de minutos demora 1 s ou 2 s para chegar ao terminal de um usuário, dificilmente será notada alguma diferença ou haverá prejuízo para a aplicação. No entanto, esta mesma mudança de 100% poderia ser significativa se a variável monitorada apresentasse variações mais rápidas e se o software que recebe a informação à distância não fosse um simples software supervisório, mas sim um elemento chave de uma aplicação crítica que deve emitir alguma resposta em um tempo máximo (esse tipo de situação será considerado em mais detalhes na seção 7.2).

Sistemas de gerenciamento usualmente não são críticos e também envolvem interações com humanos. Por exemplo, quando um operador de manutenção de uma rede LonWorks deseja efetuar uma operação de leitura de estatísticas de um certo nó para analisar o número de erros de CRC ocorridos (pode ser feito pela mensagem "*Read Memory*"), certamente ele não necessitará da resposta em poucos milissegundos, mas pode esperar até poucos segundos sem que transtornos sejam causados e sem que a usabilidade seja comprometida. Se o sistema de gerenciamento passa por uma rede IP, não deve ser necessário impor requisitos fortes a essa rede.

Uma atividade de gerenciamento bastante comum é a carga de atualizações de softwares em dispositivos. A tecnologia LonWorks permite que o programa de aplicação que é executado em um nó, bem como parte de seu *firmware*, possam ser carregados ou atualizados via própria rede LonWorks. Para isso, mensagens do tipo "*Write Memory*" devem ser enviadas contendo o código executável a ser carregado. Obviamente, essas mensagens também podem ser enviadas via rede IP por uma estação remota de gerenciamento até chegar ao nó de destino, seja através de soluções baseadas na abordagem de roteador ou de *gateway*. Para esse tipo de aplicação, dificilmente se exige um alto desempenho de rede em termos de atraso e *throughput*. Em vez disso, é imprescindível a entrega garantida de pacotes. Essa situação é análoga à transferência de arquivos via FTP nas redes de dados. Apesar de, obviamente, o usuário desejar pouco atraso e alto *throughuput*, o grande objetivo do FTP é a transferência correta e completa de arquivos entre dois pontos da rede. Por outro lado, pode haver situações em que os tempos envolvidos no gerenciamento ou atualização são relevantes, ocasionando então maiores exigências de QoS.

Apesar de não ser o caso geral, eventualmente alguma aplicação de supervisão ou gerenciamento pode impor um atraso pequeno quando comparado aos tolerados em aplicações menos críticas. Um software supervisório que monitora o sistema de detecção de incêndio de um prédio é um exemplo. Se um sinal sonoro de alerta tiver que ser acionado pelo

software após a chegada da informação de ocorrência de um alarme, provavelmente será desejado que o atraso na rede seja limitado a um valor considerado pequeno.

Considerando os cenários e as discussões acima, os requisitos de QoS para aplicações de monitoramento e registro podem variar desde *soft real time* de melhor esforço, onde inclusive uma certa taxa de perdas pode ser tolerada (equação (22)) até *hard real time* (equações (1) e (6)) (SOUCEK; SAUTER, 2004) (SOUCEK; SAUTER; RAUSCHER, 2001).

Aplicações de gerenciamento determinam, pelo menos, requisitos *soft real time* de entrega garantida, haja vista que as configurações estabelecidas para os nós e até mesmo atualizações de códigos executáveis devem obrigatoriamente chegar completa e corretamente ao seu destino (SOUCEK; SAUTER, 2004) (SOUCEK; SAUTER; RAUSCHER, 2001).

A Tabela 3 relaciona tipos de aplicações (linhas) com classes de sistemas de tempo real (colunas). Sinais "+" indicam forte associação, enquanto os símbolos "-" representam fraca ou nenhuma associação. O sinal "±" indica que pode haver forte ou fraca associação conforme a aplicação em questão. As colunas representam: *hard real time* (RT), *soft real time* de entrega garantida (GD) e *soft real time* de melhor esforço (BE).

Tabela 3 – Tipos de aplicações e possíveis requisitos de tempo real.

Baseado em Soucek e Sauter (2004).

Aplicação	*RT*	*GD*	*BE*
Monitoração / Registro	+	+	±
Gerenciamento	±	+ +	-

7.2 UTILIZANDO REDES IP COMO CANAL LONWORKS

7.2.1 **Introdução**

Em instalações típicas com vários canais LonWorks TP/FT-10 (par trançado a 78kbps) que precisavam ser interconectados para formar uma única rede LonWorks maior, costumava-se criar um *backbone* LonWorks TP/XF-1250 (par trançado a 1,25Mbps) para interconectar os diversos canais através de roteadores (ADEPT, 2002). Esta topologia pode ser visualizada na Figura 35.

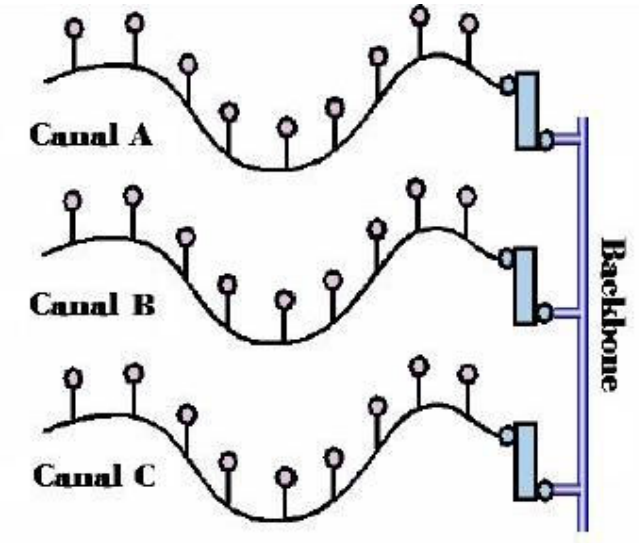


Figura 35. Canais LonWorks TP/FT-10 interligados por backbone TP/XF-1250.
Baseado em Adept (2002).

Esta topologia é adequada para muitas instalações básicas, mas observa-se que o canal TP/XF-1250 apresenta algumas limitações quando serve como *backbone*. Adept (2002) destaca três delas.

A primeira limitação refere-se ao comprimento máximo do canal, isto é, 130 metros, que pode ser um problema para interconectar diversas redes distribuídas em uma grande área (por exemplo, prédios de um campus de universidade). Repetidores podem ser instalados para estender o comprimento máximo do canal, mas isso requer planejamento, cabeamento, manutenção e suporte adicionais.

Outra limitação diz respeito à banda disponível. Quando a tecnologia LonWorks foi introduzida, um canal de 1,25Mbps (que apresenta largura de banda aproximada de 800 pacotes por segundo), era suficiente para praticamente todos os projetos que se tinha em mente. Com o aumento da complexidade e do tamanho dos sistemas baseados em redes de controle, bem como a fatia de largura de banda cada vez maior tomada pelas ferramentas de gerenciamento, monitoração, instalação e configuração, este tipo de canal deixou de ser suficiente em muitas situações.

Por fim, a terceira limitação refere-se às dificuldades de instalação de um *backbone* TP/XF-1250 em um edifício já construído. Um planejamento adequado é necessário para garantir que os roteadores sejam instalados em locais estratégicos para respeitar as especificações de comprimento máximo de segmentos e de banda disponível. Novos cabos

devem ser passados, seja dentro de um mesmo edifício ou entre edifícios distribuídos em uma certa área. Nesse último caso, o custo pode inviabilizar o projeto.

Três dos fatores que tornam as redes IP uma boa alternativa para servir como *backbones* LonWorks são: velocidade, grande presença em diversos ambientes (principalmente corporativos) e o baixo custo de equipamentos e infra-estrutura (ADEPT, 2002).

Um *backbone* IP é uma rede IP utilizada para transportar pacotes LonTalk entre diferentes canais LonWorks, interconectando-os de forma transparente e formando uma única rede LonWorks maior. A Figura 36 provê uma ilustração que utiliza a Internet como parte do *backbone*.

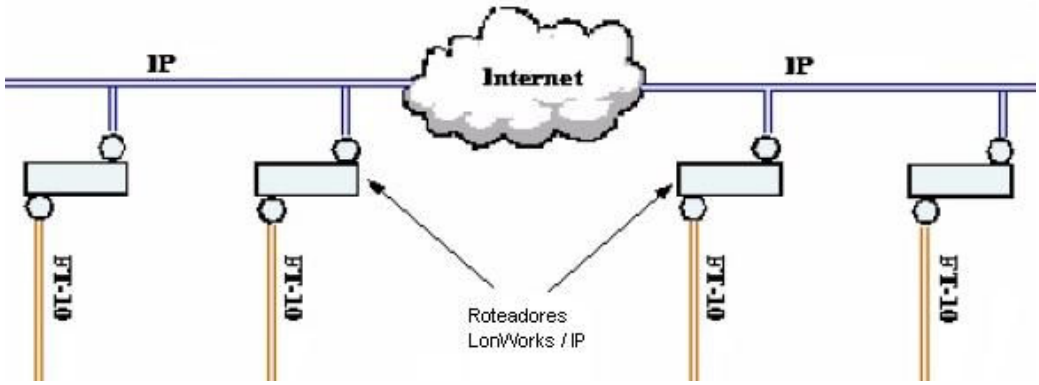


Figura 36. Canais LonWorks TP/FT-10 interligados por backbone IP.
Baseado em Adept (2002).

A distância que pode ser coberta por um *backbone* IP é ilimitada. Quando utilizada a Internet, pode-se interligar canais LonWorks presentes em até mesmo cidades diferentes a um baixo custo. Por outro lado, nesse caso é até intuitivo perceber que crescem significativamente as dificuldades para se atender a requisitos de parâmetros de qualidade de serviço. Sendo assim, nem toda aplicação pode ser implementada com sucesso utilizando-se a Internet. Porém, as intranets privadas atendem a esses requisitos com mais facilidade e, conseqüentemente, podem ser a solução para muitos problemas, tais como o já mencionado exemplo da interligação de redes de prédios de uma universidade.

Quando se diz que diversos canais LonWorks são interconectados transparentemente por um *backbone* IP, como é o caso da Figura 36, está-se considerando que os dispositivos de um canal podem enviar pacotes LonTalk para dispositivos de outro canal utilizando as

maneiras de endereçamento permitidas pela tecnologia LonWorks (apresentado na seção 4.3) sem se preocupar com a sua localização física. Em outras palavras, seria indiferente se o nó endereçado estivesse no mesmo ou em outro canal.

Dessa maneira é possível, por exemplo, efetuar *bindings* entre NVs de nós presentes em canais diferentes, conforme ilustração da Figura 37.

É aqui que se encaixa a terceira categoria de aplicação levantada na seção 1.2, ou seja, a utilização de uma rede de dados IP como meio para interconectar redes LonWorks geograficamente distantes para aplicações sem requisitos fortes de tempo real. Pela Figura 37, o nó A e o nó B estão transparentemente interligados via rede IP, mas as características e condições instantâneas dessa rede podem interferir no desempenho da comunicação. De acordo com os requisitos de QoS da aplicação em questão, pode-se verificar se a infra-estrutura IP atende às necessidades ou, por outro lado, projetar uma infra-estrutura especialmente para atendê-las. As situações que apresentam requisitos fortes de tempo real ou referem-se a um laço de controle serão abordadas posteriormente.

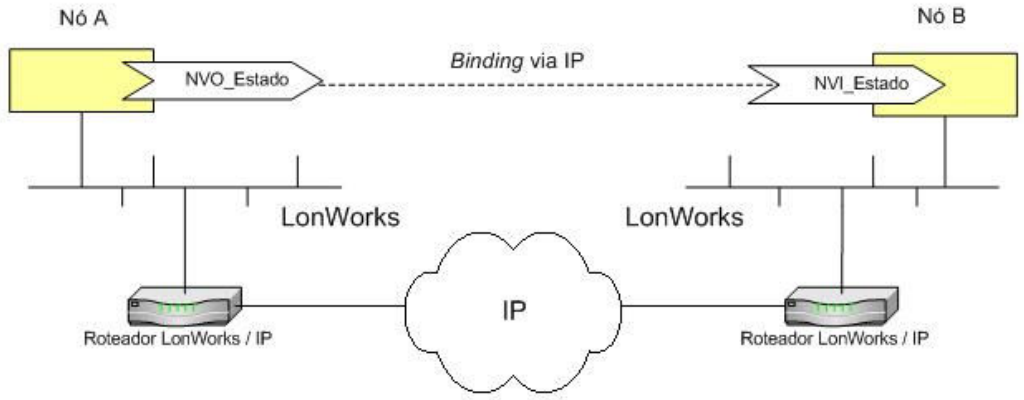


Figura 37. Binding via IP entre NVs presentes em canais diferentes.

7.2.2 **Arquitetura da solução**

Como já adiantado na seção 5.2.3 e ilustrado nas Figuras 16, 36 e 37, o método mais natural e utilizado para se interconectar canais LonWorks via redes IP é através do tunelamento de pacotes LonTalk em datagramas UDP de acordo com a norma EIA-852. Do ponto de vista das redes LonWorks, esse tunelamento é transparente e a rede IP é enxergada como mais um meio físico disponível para redes LonWorks (conhecido como IP-852).

Os roteadores LonWorks/IP devem ser previamente configurados para ter conhecimento sobre outros roteadores LonWorks/IP na rede. Isso é necessário, no mínimo, para que eles saibam para onde enviar os datagramas UDP com pacotes LonTalk encapsulados. Esses roteadores podem ser implementados de maneira básica, simplesmente repassando todos os pacotes LonTalk para todos os outros roteadores na rede IP (repetidor), ou podem apresentar implementação mais elaborada em que a correspondência entre endereços de nós LonWorks e de endereços IP de roteadores é automaticamente aprendida (*learning router*) (GAW; MARSH, 1997).

Na rede IP, podem estar diretamente conectados nós IP-852. Conforme já visto, são dispositivos que implementam o protocolo LonTalk mas, ao invés de apresentar conector para algum meio físico LonWorks tradicional (par trançado, rede elétrica, etc.), já apresentam conector Ethernet e implementam nativamente a norma EIA-852. Nós LonWorks convencionais localizados "atrás" de roteadores LonWorks/IP são capazes de endereçar os nós IP-852 normalmente através das formas de endereçamento previstas pela tecnologia LonWorks.

A Figura 38 apresenta uma ilustração de canais LonWorks interligados através de uma rede IP, a qual por sua vez conta com alguns nós IP-852.

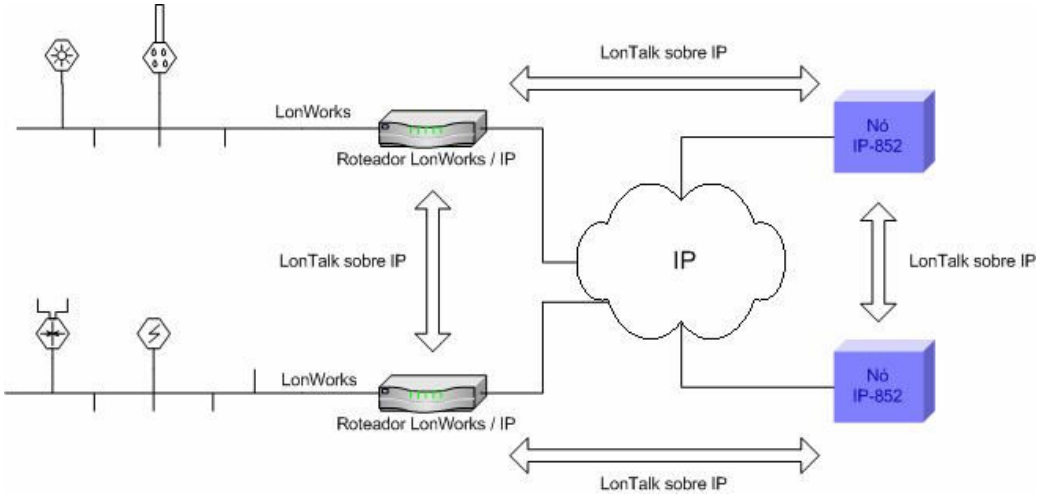


Figura 38. Canais LonWorks e nós IP-852 interligados por rede IP.
Baseado em Soucek, Sauter e Koller (2003).

Conceitualmente, também é possível implementar soluções para o mesmo problema com a abordagem de *gateways*. Ao invés de se encapsular pacotes LonTalk em datagramas

UDP, converte-se o LonTalk para algum outro protocolo na rede IP conforme a Figura 39. A sigla *XYZ* indica um protocolo qualquer utilizado pelos *gateways* sobre a rede IP.

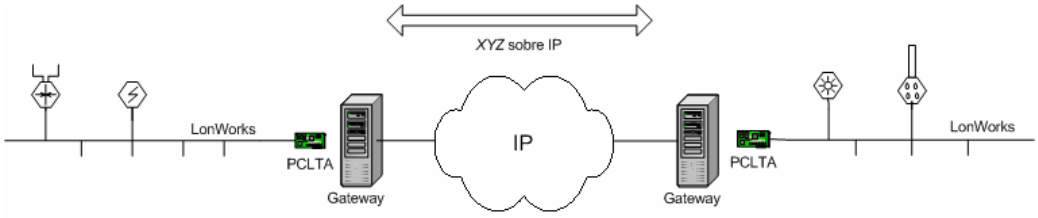


Figura 39. Canais LonWorks interligados por rede IP através de gateways.

No entanto, não se costuma utilizar esta abordagem na prática porque ela apenas acrescenta um *overhead* de processamento na conversão de protocolos. O protocolo que é executado no lado da rede IP deve ser novamente convertido em LonTalk quando uma mensagem chega a um roteador de destino, e, por isso, acaba tendo pouca utilidade. Além do mais, não haveria compatibilidade com nós IP-852 ou roteadores LonWorks / IP, já padronizados pela norma EIA-852.

Por outro lado, casos especiais devem ser considerados. Pode ser desejado, por exemplo, criar um protocolo para transmitir mensagens LonTalk na rede IP com certos recursos de segurança (criptografia, por exemplo). Nesse caso, provavelmente a necessidade desses recursos é mais importante que a queda de desempenho causada pelo aumento de *overhead* de processamento. Uma outra possível justificativa para o uso de um *gateway* nessa aplicação seria a utilização de um protocolo no meio IP que tunelasse pacotes LonTalk compactados, reduzindo a utilização da rede. Todavia, esses pacotes costumam ser muito pequenos e, para que a compactação gerasse resultados significativos, seria necessária a agregação de pacotes (*packet bunching*), o que introduziria mais atraso.

7.2.3 **O equipamento i.Lon 600**

O i.LON 600 é um roteador LonWorks/IP fabricado pela criadora da tecnologia, a Echelon. Compatível com a norma EIA-852, provê também uma interface serial para configuração de endereço IP, máscara de sub-rede, usuário, senha, etc. Pode ser configurado

como um *learning router*, aprendendo automaticamente a localização dos dispositivos LonWorks e montando sozinho suas tabelas de roteamento internas.

O i.LON 600 também oferece acesso via web (HTTP) para configuração. Por segurança, o mesmo pode ser desabilitado. Também provê acesso via FTP e suporta a configuração de usuário e senha.

Até 256 equipamentos i.LON 600 podem compor um canal IP (seção 7.2.6).

7.2.4 **O equipamento L-IP**

O L-IP, fabricado pela empresa austríaca Loytec, é outra alternativa de roteador LonWorks/IP compatível com a norma EIA-852. Também pode ser utilizado como uma interface LonWorks remota, bastando instalar o *driver* apropriado no PC. É um exemplo de equipamento que funciona tanto como NIC-IP quanto roteador, conforme mencionado pela seção 5.2.2. Possui compatibilidade com ferramentas de diagnóstico da Loytec.

7.2.5 **Controle via rede IP**

Esta seção iniciará o tratamento da quarta categoria de aplicação levantada na seção 1.2, isto é, utilizar a rede de dados IP como meio para interconectar redes LonWorks geograficamente distantes para aplicações com requisitos fortes de tempo real (*hard real time*), tendo em vista principalmente a implementação de laços de controle onde o controlador e a planta não estão conectados diretamente ou na mesma rede, mas sim através de uma rede IP.

Na situação tradicional, os laços de controle são implementados localmente, ou seja, o controlador fica localizado geograficamente próximo aos sensores e atuadores, e a largura de banda de comunicação entre os mesmos é dedicada e conhecida. Desta maneira, consegue-se projetar e implementar o laço de controle com certa facilidade, pois já se conhece de antemão todas as características da comunicação, como por exemplo o atraso e o *jitter*. Este último é, inclusive, normalmente nulo, pois usa-se um meio físico dedicado e não uma rede de comutação de pacotes com roteadores intermediários.

No início, logo após o surgimento dos *fieldbus*, sua aplicação em laços de controle era limitada a poucas áreas e, ainda assim, utilizavam-se apenas alguns poucos tipos de rede especialmente projetados para atender os fortes requisitos de tempo real exigidos pela

engenharia de controle clássica. Entretanto, no passado recente, a atitude da engenharia de controle tem mudado e redes de controle mais gerais passaram a receber atenção adicional. Nos sistemas de controle baseados em rede, ou *Network-Based Control Systems* (NBCS), nem sempre é mais a rede que tem que obedecer as regras de projeto de sistema, mas sim o projeto de sistema que deve considerar e lidar com as imperfeições da rede de comunicação disponível. Tudo isso sem desconsiderar a busca por sistemas robustos (SOUCEK; SAUTER; KOLLER, 2003).

A aplicação de redes *fieldbus* em laços de controle tem sido foco de muita pesquisa. Um passo adicional seria a sua utilização de maneira integrada com as redes IP, que é justamente o tema desta seção. Abordou-se anteriormente soluções para integrar canais LonWorks distantes através de rede IP (especialmente via tunelamento de pacotes LonTalk conforme a norma EIA-852). A questão que surge é por que utilizar essa estrutura como parte de um laço de controle, já que laços de controle são convencionalmente implementados através de estruturas dedicadas onde os controladores se localizam próximos à planta controlada. O natural seria usufruir das redes IP como uma maneira de configurar e parametrizar remotamente o laço de controle, mas não como parte fundamental dele.

A Figura 40 apresenta o diagrama de blocos de um laço de controle tradicional, ilustrando também como seria se o mesmo fosse implementado via rede IP.

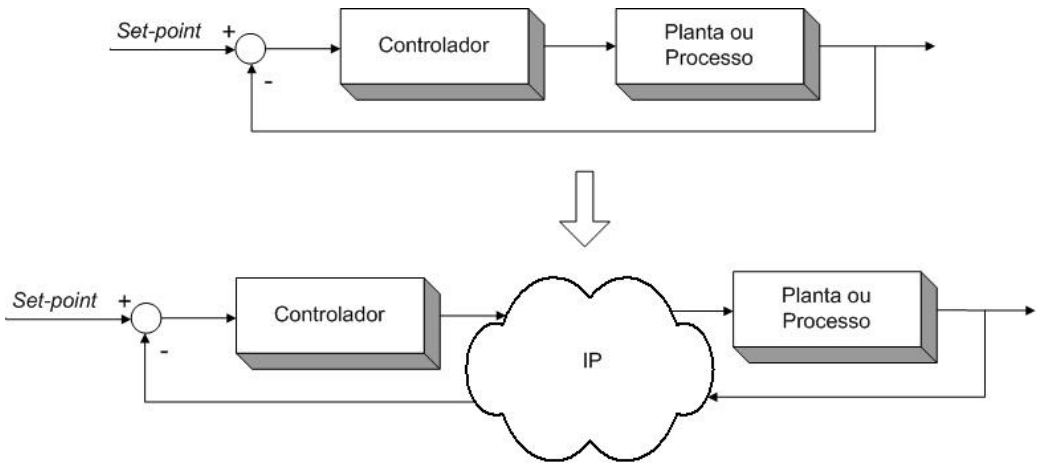


Figura 40. Controle tradicional e controle via rede IP.

Uma das razões é que as redes IP estão cada vez mais onipresentes em todo tipo de ambiente e, mercadologicamente, observa-se sua crescente inserção no mundo das redes

fieldbus. Em automação predial, por exemplo, há a tendência de restringir redes de controle a salas individuais para conectar sensores e atuadores e utilizar o cabeamento estruturado já existente para interconectar essas "ilhas" de redes de controle.

A Internet é um caso particular de rede IP que, apesar das contestações, apresenta potencial para servir como meio de comunicação dentro de um laço de controle. Overstreet e Tzes (1999) apresentam uma solução para a implementação de um laboratório virtual em uma universidade com acesso remoto pela Internet. Apesar de não ser baseada em tecnologia LonWorks, serve para ilustrar aplicações interessantes e mostrar resultados de experimentos que certamente seriam semelhantes para qualquer tecnologia específica aplicada. O cenário deste problema é composto por um servidor ligado a um experimento físico através de placas de aquisição de sinais. Uma máquina cliente acessa esse servidor via rede, seja local ou Internet, permitindo que um usuário monitore e obtenha dados à distância. Nas soluções tradicionais, o algoritmo de controle costuma ser executado localmente no servidor. Parâmetros gerais e sinais de referência de controle são enviados em lote (*batch*) do cliente para o servidor, fazendo com que o experimento tenha que ser reiniciado para a carga nos novos parâmetros quando se deseja realizar algum ajuste. A solução proposta por este artigo transfere a carga de processamento do algoritmo de controle para o cliente. O cliente passa a enviar ao servidor em tempo real, via rede IP, um sinal de comando baseado no sinal medido do experimento. Este último é enviado do servidor ao cliente. Os autores desenvolveram um conjunto de softwares que implementam esta proposta. Diversos testes foram realizados e os resultados são apresentados pelo artigo. Para o envio dos sinais em tempo real, pode ser utilizado tanto o UDP quanto o TCP. Um protocolo próprio de aplicação é executado acima dessas camadas. Obviamente, os tempos de transmissão ficam sujeitos às condições instantâneas da rede, que influencia diretamente os resultados. Apesar das desvantagens, a transferência do processamento do algoritmo de controle para a máquina cliente, que fica "do outro lado" da Internet em relação ao local do experimento, também viabiliza uma série de vantagens:

- habilidade de utilizar clientes computacionalmente poderosos para a implementação de leis de controle complexas, que não precisam ser transportados até o laboratório;
- habilidade de implementar controle avançado apesar de limitações no lado do servidor (conseqüência do item anterior);
- privacidade do código do controlador implementado, já que ele é executado na máquina do cliente. Não é necessário carregá-lo no servidor;

- maior flexibilidade no processo de desenvolvimento do controlador. O usuário é livre para implementar qualquer tipo de controlador no lado do cliente;
- maior segurança contra vírus ou programas com mal-funcionamento. Deve-se lembrar que o servidor está acessível aos usuários via rede, e há a possibilidade de programas desse tipo terem sido implantados voluntária ou involuntariamente.

O artigo apresenta diversos resultados, já que muitos parâmetros são testados. Dentre as conclusões principais, cabe destacar que a resposta do sistema é mais lenta quando se aumenta o período de envio de amostras do servidor para o cliente. Isso é esperado, haja vista que o sinal de comando enviado pelo controlador do cliente contém um certo atraso, já que a medida recebida contém um atraso de sincronização gerado pelo servidor. Quando a carga na rede é elevada e a banda disponível é pequena, atrasos ainda maiores são observados entre a resposta do sistema e o sinal de referência ou comando.

7.2.6 **O modelo de um canal IP**

Para as seções seguintes, convém definir formalmente o conceito de canal IP conforme Soucek e Sauter (2004) e Soucek, Sauter e Koller (2003).

Em geral, redes IP podem ser modeladas através de um grafo conectado $G(V,E)$ onde o conjunto de vértices V representa o conjunto de dispositivos IP conectados à rede e o conjunto de arestas E é o conjunto de enlaces (*links*) de comunicação. A Figura 41 oferece um exemplo desse tipo de representação.

Observe que os nós V_{10}, V_{11}, V_{12} e V_{13} foram descartados no grafo pelo fato de se ter-se interesse apenas nos dispositivos IP.

Seja $C(V',E')$ um subgrafo de G que abrange exclusivamente os dispositivos de controle existentes na rede IP, bem como os enlaces, *switches* e roteadores que os interconectam. O grafo $C(V',E')$ representa uma rede lógica que é, por definição, conhecida como canal IP. Seja $V''(C)$ o subconjunto de vértices que denota os dispositivos de controle IP (ex.: sensores, atuadores e controladores IP-852), e $V'''(C)$ o subconjunto dos nós intermediários (roteadores e *switches*) que interligam os elementos de $V''(C)$, tem-se que $V''(C) \cup V'''(C) = V'(C)$.

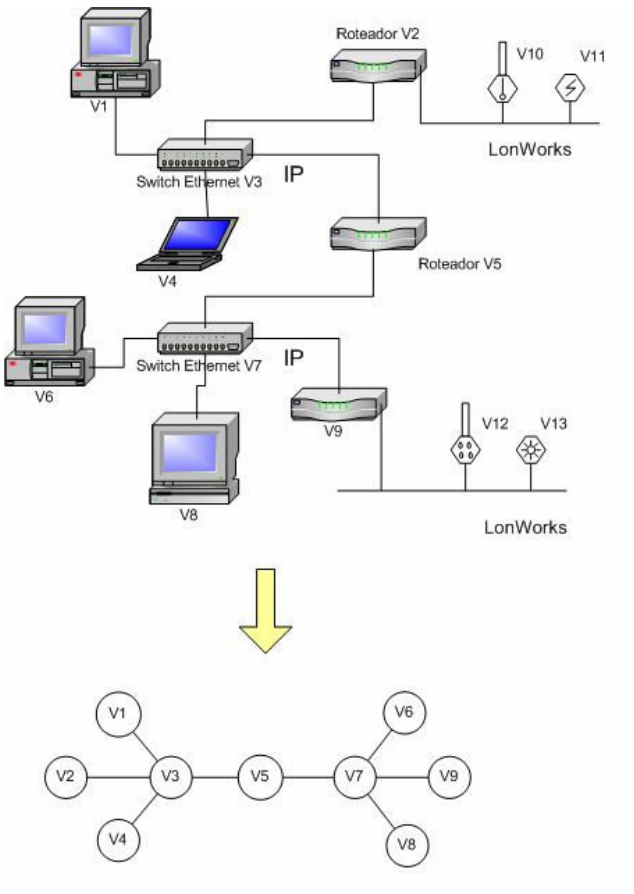


Figura 41. Conjunto de redes IP / LonWorks representada por um grafo conectado.

V''(C) é chamado de conjunto de membros do canal IP, enquanto *V'''(C)* e *E'(C)* são denominados de infra-estrutura de rede do canal IP.

Para implementar um canal IP que possibilite a troca de dados entre seus membros, é necessário um protocolo especial que o gerencie e permita a diferenciação entre os pacotes relacionados às aplicações de controle e aqueles relacionados às aplicações de dados. Esses pacotes trafegarão na mesma rede caso a infra-estrutura não tenha sido construída para ser de uso exclusivo das aplicações de controle, isto é, caso o canal IP seja de uso compartilhado entre aplicações de controle e aplicações de dados. Essa é exatamente a situação da Internet quando utilizada como canal IP.

Foi destacado que a norma EIA-852 é um padrão para tunelamento de pacotes de redes de controle através de redes IP, sendo inclusive bastante adotada por produtos de mercado. Assim, está-se considerando a partir de então que os canais IP são sempre implementados através do protocolo estabelecido pela norma EIA-852.

Em geral, as redes LonWorks baseadas em IP podem ser implementadas de duas maneiras (SOUCEK; SAUTER, 2004):

- todos os membros do canal IP são dispositivos IP-852, isto é, geram pacotes LonTalk e estão conectados diretamente à rede IP;
- alguns membros do canal IP são roteadores LonWorks/IP que interconectam uma rede LonWorks nativa à rede IP.

Em ambos os casos, a norma EIA-852 está sendo aplicada. No exemplo da Figura 40, tem-se que $V''(C) = \{V_2, V_9\}$ e $V'''(C) = \{V_3, V_5, V_7\}$. Não há dispositivos IP-852 além dos roteadores LonWorks / IP.

Chama-se de tráfego de controle (*control network data traffic*) todo o tráfego de pacotes injetado pelos elementos de $V''(C)$ no subgrafo C. O tráfego gerado pelos outros elementos de V que passam pelos enlaces de $E'(C)$ é chamado de tráfego cruzado no canal IP (*cross traffic*). Observando o tráfego de controle em um canal IP, alguns pacotes estão associados a um subconjunto $V_{cl}(C)$ de $V''(C)$. Considerando-se que $V_{cl}(C)$ é composto exclusivamente por nós envolvidos em um laço de controle (*control loop*) escolhido para estudo, denomina-se esses pacotes de tráfego de laço de controle (*control loop traffic*), e todos os outros de tráfego paralelo (*parallel traffic*).

De acordo com essas definições, o *tráfego de laço de controle* mais o *tráfego paralelo* compõem o *tráfego de controle*. O tráfego total na rede é formado pelo *tráfego de controle* mais o *tráfego cruzado*. O tráfego cruzado nada mais é do que aquele proveniente das aplicações de dados que compartilham os membros e infra-estrutura do canal IP.

7.2.7 **Requisitos de QoS**

Uma vez que foram definidos os conceitos de controle sobre IP e de canal IP, serão vistos agora quais requisitos de QoS de rede devem ser considerados para aplicações de controle.

Quando se deseja avaliar se um certo conjunto de parâmetros de QoS atende uma determinada aplicação, é necessário conhecer os parâmetros da aplicação que precisam ser atingidos. Estudando-se a relação entre os parâmetros de QoS e os parâmetros da aplicação, pode-se projetar uma rede que ofereça a QoS necessária ou verificar se a QoS de uma rede existente é suficiente para a aplicação.

Ao se analisar aplicações de sistemas supervisórios, viu-se que, por exemplo, o atraso não pode exceder o tempo máximo em que uma mensagem de alarme deve trafegar do servidor à estação de trabalho do operador. Trata-se de uma relação direta entre um requisito da aplicação e um requisito de QoS.

Quando se tem em mente a implementação de laços de controle, é necessária a adoção de parâmetros mensuráveis que permitam avaliar o sucesso ou não da implementação. Nesse contexto, surge o conceito de qualidade de controle, ou *Quality of Control* (QoC), adotado por Soucek e Sauter (2004) e Soucek, Sauter e Koller (2003). Entre possíveis parâmetros de QoC, são citadas variáveis bastante comuns no mundo do controle, tais como o sobressinal (*overshoot*), tempo de acomodação e margem de estabilidade.

Segundo Soucek, Sauter e Koller (2003), os parâmetros de QoS mais relevantes para a QoC são o atraso, o *jitter* e a taxa de perda de pacotes. Define-se, portanto, o vetor **q**, chamado de vetor de QoS.

$$\mathbf{q} = (D_{\min}, J_k, 1 - w) \tag{26}$$

De acordo com o mesmo artigo, as origens do *jitter* e da perda de pacotes relativos ao tráfego de laço de controle podem ser discutidas como uma função do tráfego cruzado e do tráfego paralelo no canal IP. Pode-se escrever o vetor **q** como uma soma de duas componentes: o vetor $\mathbf{q}_N = (D_{\min,N}, J_{k,N}, 1 - w_N)$ induzido pela rede (*network induced*) e o vetor $\mathbf{q}_P = (D_{\min,P}, J_{k,P}, 1 - w_P)$ induzido pelo protocolo (*protocol induced*).

$$\mathbf{q} = \mathbf{q}_N + \mathbf{q}_P \tag{27}$$

O tráfego cruzado costuma ser a maior fonte de *jitter* de um canal IP. Os nós de *V'''* do grafo *C* devem servir filas de pacotes que provêm de todo o tipo de aplicação que utiliza a infra-estrutura da rede IP, resultando em tempos de serviço variáveis e em perda de pacotes de acordo com o volume do tráfego cruzado. Uma vez que esses efeitos são induzidos pela topologia, tráfego, recursos e equipamentos da rede, eles são chamados de induzidos pela rede.

O tráfego paralelo também aumenta a quantidade de tráfego que os nós de *V'''* devem servir, mas este é tipicamente pequeno quando comparado ao tráfego cruzado e pode ser desprezado (SOUCEK; SAUTER; KOLLER, 2003).

Efeitos mais significativos são originados por dois mecanismos suportados pela norma EIA-852 conhecidos como agregação e seqüenciamento. Por ter origem no protocolo, e não na rede, esses efeitos são chamados de induzidos pelo protocolo.

A agregação consiste em coletar pacotes LonTalk de diferentes origens para um mesmo destino durante um intervalo de tempo t_A. Neste cenário, os nós de origem e de destino estão localizados em redes LonWorks distantes, conectadas por roteadores LonWorks / IP via rede IP. A coleta é feita pelo roteador LonWorks/IP conectado à rede LonWorks de origem, que encapsula todos pacotes obtidos no mesmo datagrama UDP. Portanto, após sucessivos intervalos de tempo com duração t_A, um único datagrama UDP é enviado ao roteador que provê interconexão com a rede de destino. Este datagrama contém todos os pacotes LonTalk endereçados ao nó em questão. O roteador de destino é o responsável por extrair cada pacote LonTalk do datagrama UDP e dispará-los corretamente na rede LonWorks. A agregação diminui o *overhead* na rede IP, mas aumenta o atraso ponto a ponto em até t_A (despreza-se os tempos de processamento adicionais nos roteadores para compor e separar os datagramas UDP).

O seqüenciamento é um mecanismo implementado nos roteadores LonWorks/IP responsável por garantir que os pacotes sejam recebidos na mesma seqüência em que foram enviados. Constatou-se que, em uma rede IP, os pacotes enviados de uma origem a um destino podem chegar fora de ordem quando ocorre a condição dada pela equação (24). Se uma aplicação apresenta o requisito dado pela equação (23), isto é, se a seqüência de pacotes não pode ser alterada, então este mecanismo adquire extrema importância. Contudo, como será visto a seguir, o seqüenciamento também é uma fonte de *jitter* e de perda de pacotes induzida pelo protocolo, podendo comprometer outros requisitos da aplicação.

Seja S a seqüência de pacotes recebidos, onde $p_i(t_i)$ indica que o pacote p_i foi observado pelo receptor no instante t_i. Para que a seqüência enviada seja preservada na recepção, deve-se ter:

$$S = \langle p_0(t_0), p_1(t_1), p_2(t_2), \ldots, p_n(t_n) \rangle, \qquad t_n > \ldots > t_1 > t_0 \tag{28}$$

O mecanismo de seqüenciamento no roteador de destino tem duas opções de implementação. A primeira é descartar imediatamente todos os pacotes fora de ordem recebidos pela rede IP (*drop sequencer*). A segunda é armazenar esses pacotes em um *buffer* de memória por um intervalo de tempo máximo t_E (*escrow sequencer*). Se os pacotes faltantes não chegarem até t_E, os recebidos fora de ordem são descartados. Se chegarem, são colocados

em ordem e só então o roteador os repassa. Repare que o caso *drop sequencer* é um caso particular de *escrow sequencer* onde $t_E = 0$.

O esquema *drop sequencer* é uma fonte de aumento da taxa de perda de pacotes, enquanto o *escrow sequencer* introduz *jitter* em até t_E. Outro exemplo será considerado. Seja S_R uma certa seqüência de pacotes recebidos.

$$S_R = \langle p_0(t_0), p_2(t_1), p_3(t_2), p_1(t_3) \rangle, \qquad t_3 > \ldots > t_0 \tag{29}$$

O pacote p_1 sofreu um grande atraso e foi recebido somente depois de p_2 e p_3. O esquema *drop sequencer* produziria a seqüência S_D.

$$S_D = \langle p_0(t_0), p_2(t_1), p_3(t_2) \rangle, \qquad t_2 > \ldots > t_0 \tag{30}$$

O esquema *escrow sequencer*, ao aguardar pelo pacote faltante, produziria a seqüência S_E, supondo que $t_3 - t_1 < t_E$.

$$S_E = \langle p_0(t_0), p_1(t_3), p_2(t_3), p_3(t_3) \rangle, \qquad t_3 > \ldots > t_0 \tag{31}$$

O *jitter* introduzido é igual a $t_3 - t_1$ para p_1 e $t_3 - t_2$ para p_2.

Soucek, Sauter e Koller (2003) estudam através de simulações os efeitos do *jitter* e da perda de pacotes em um sistema de controle baseado em rede. São investigadas as influências do vetor de QoS **q** na QoC. O cenário proposto é ilustrado na Figura 42.

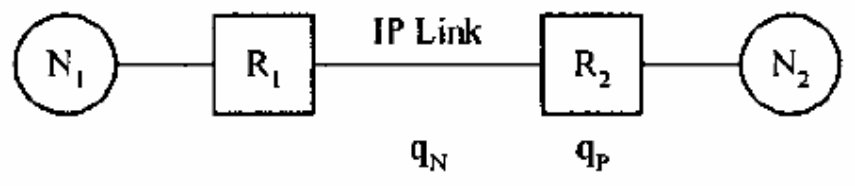


Figura 42. Cenário de simulação do efeito da QoS na QoC.
Extraído de Soucek, Sauter e Koller (2003).

N_1 e N_2 são nós LonWorks conectados via rede IP através dos roteadores R_1 e R_2 conforme a norma EIA-852. Esses roteadores implementam agregação e seqüenciamento.

Considera-se que N_1 gera pacotes a uma taxa fixa com período T_S, enviando-os a N_2 através do serviço *Repeated* (seção 4.2.4) com n retransmissões. R_1 está com a agregação desligada ($t_A = 0$) e assume-se que a rede IP possui largura de banda suficiente. R_2 emprega seqüenciamento com t_E. A componente induzida pelo protocolo do vetor de QoS, isto é, $\mathbf{q}_P$, não é um parâmetro de entrada direto da simulação, mas sim uma conseqüência de n, T_S e t_E. Esses três, por sua vez, são parâmetros da simulação. T_S foi fixado em 10 ms.

Por outro lado, a componente de QoS induzida pela rede, $\mathbf{q}_N$, é um parâmetro direto da simulação. Foi utilizado $D_{\min,N} = 20$ ms, $1 - w_N = 0$ (considerou-se que não há perda de pacotes induzida pela rede) e um *jitter* aleatório segundo a distribuição de probabilidade exponencial indicada por (32).

$$f_J(J) = \lambda \exp(-\lambda J) \tag{32}$$

Na simulação, um vetor com muitos valores aleatórios para o *jitter* foi previamente gerado de acordo com (32). Para cada pacote enviado de N_1 para N_2, um certo elemento do vetor foi tomado. Foi utilizado $1/\lambda = 5$ ms.

Apesar de ser muito difícil atribuir uma distribuição de probabilidade ao atraso ou *jitter* de uma rede IP, principalmente quando se trata da Internet (PAXTON; FLOYD, 1997), é necessário utilizar algum modelo para a simulação. Além disso, justamente pela complexidade ou impossibilidade de se prever o atraso e o *jitter*, o projeto de um controlador robusto não deve se apoiar em uma distribuição específica dessas grandezas. Sendo assim, a distribuição de (32) acaba servindo como um caso de teste qualquer para a simulação, e não como uma distribuição do *jitter* fiel à realidade.

O laço de controle simulado consiste em uma planta discreta e um controlador também discreto que respeita o atraso médio da rede. A Figura 43 apresenta uma visualização deste laço, enquanto as equações (33) e (34) descrevem a planta e o controlador, respectivamente, através de sua função de transferência.

Assim como o controlador (nó N_1) gera pacotes em intervalos T_S, a planta também amostra os valores recebidos da rede em intervalos T_S. É suposto que ambos não possuem fila de entrada, ou seja, o último valor recebido da rede é utilizado no próximo ciclo. Se a planta recebe do controlador dois valores da rede dentro de um mesmo intervalo de amostragem, apenas o último é enxergado e o primeiro é descartado.

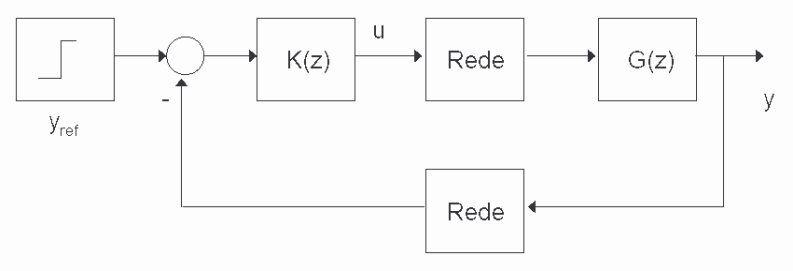


Figura 43. Laço de controle simulado.
Baseado em Soucek, Sauter e Koller (2003).

$$G(z) = \frac{-0{,}0286z - 0{,}0273}{z^2 - 1{,}836z + 0{,}872} \tag{33}$$

$$K(z) = \frac{-0{,}133z^2 + 0{,}302z - 0{,}190}{z^2 - 1{,}368z + 0{,}368} \tag{34}$$

Inicialmente, utilizou-se $n = 0$ e seqüenciamento desligado. Isso significa que o roteador R_2 não reordena nem descarta os pacotes fora de seqüência. Assim, a troca de ordem ou perda de pacotes causada pelo *jitter* é enxergada pelo controlador e pela planta como ruído no sinal.

Como parâmetro para avaliar a QoC, foi escolhida a resposta ao degrau unitário *H(t)*, que é apresentada na Figura 44.

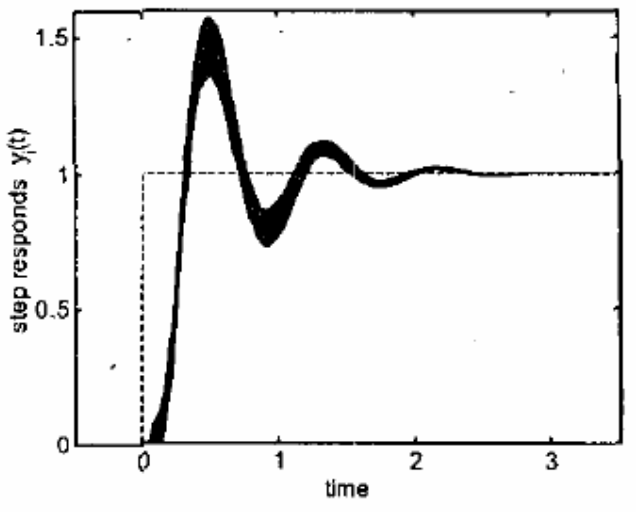


Figura 44. Resposta ao degrau unitário.
Extraído de Soucek, Sauter e Koller (2003).

O trabalho de Soucek, Sauter e Koller (2003) investiga potenciais melhoras na QoC alterando alguns parâmetros da rede. Com a utilização de seqüenciamento com $t_E = 0$ (*drop sequencer*), elimina-se o problema de trabalhar com dados antigos mesmo quando há novos dados disponíveis. Apesar de este método possuir a aparente desvantagem do aumento da taxa de perda de pacotes, verifica-se que a QoC melhora devido ao fato de que apenas os pacotes desatualizados são descartados. O resultado desta nova simulação é apresentado na Figura 45.

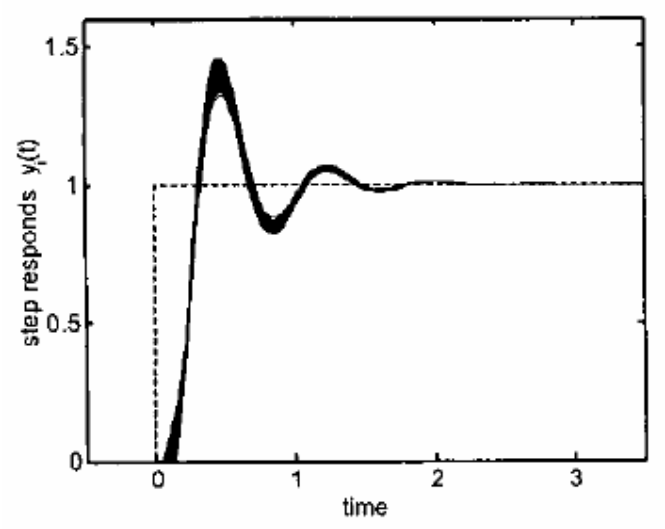


Figura 45. Resposta ao degrau unitário com drop sequencer.
Extraído de Soucek, Sauter e Koller (2003).

A utilização do esquema *escrow sequencer* no lugar de *drop sequencer* não melhora a situação pois nem o controlador nem a planta conseguem tirar vantagem da correta seqüência de pacotes. Repare na equação (30) que os pacotes p_1, p_2 e p_3 são observados conjuntamente no mesmo instante, t_3, resultando no efeito de como se p_1 e p_2 tivessem sido descartados pela sobreposição de p_3.

Uma possível maneira de tirar vantagem do *escrow sequencer* seria a utilização de uma fila de entrada no controlador e na planta aliada à introdução de algum atraso entre a recepção do valor da rede e o seu repasse adiante. Esse atraso possibilitaria a recomposição da seqüência na fila sem que os pacotes reordenados fossem repassados no mesmo instante. Essa idéia é semelhante ao conceito dos *buffers* no receptor utilizados na transmissão de áudio e vídeo em tempo real pela Internet. Em contrapartida, deve-se avaliar se o atraso introduzido não prejudica de outras maneiras os parâmetros de QoC relevantes para a aplicação.

A sobreamostragem (*oversampling*) é uma outra possibilidade para melhorar a QoC. Ela pode ser realizada transmitindo-se o valor amostrado diversas vezes dentro de um mesmo período, sendo conhecida, nesse caso, como sobreamostragem de rede (*network oversampling*). No exemplo desta simulação, isso seria equivalente a transmitir a mesma amostra quatro vezes em intervalos de 2,5 ms dentro do período de T_S = 10 ms. Do ponto de vista da rede LonWorks, isso pode ser conseguido utilizando-se n = 3, isto é, três retransmissões para cada valor. Isso combate os efeitos da perda de pacotes e do *jitter*, mas multiplica por quatro o tráfego injetado na rede. Se a infra-estrutura e os tráfegos cruzado e paralelo não permitirem que o *throughput* necessário seja atingido, não haverá sucesso.

A sobreamostragem de sistema (*system oversampling*) consiste em amostrar todo o sistema em uma taxa maior que a necessária. No esquema deste exemplo, ajustando-se T_S para 2,5 ms, adaptando-se o controlador e refazendo-se a simulação, obtém-se a resposta da Figura 46.

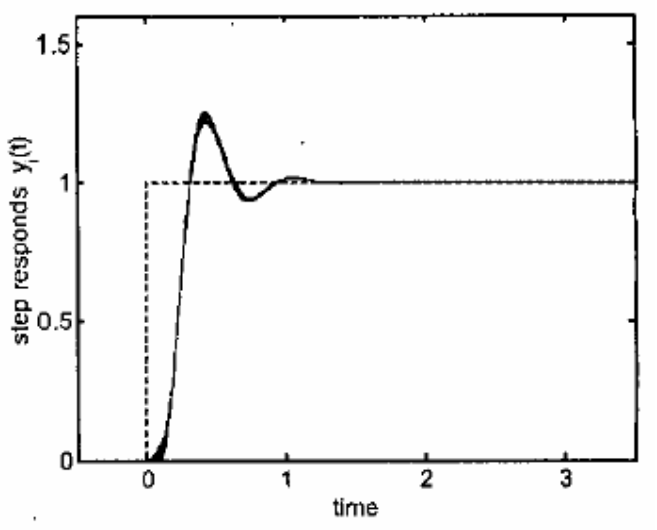


Figura 46. Resposta ao degrau unitário com sobreamostragem de sistema.
Extraído de Soucek, Sauter e Koller (2003).

Para maiores detalhes, pode-se consultar Soucek, Sauter e Koller (2003). Neste artigo, também são abordados critérios de estabilidade e o cálculo do erro do controlador devido à perda de pacotes, itens que não são discutidos neste trabalho.

Também podem ser encontrados na literatura outras análises e resultados. Soucek, Sauter e Koller (2002) consideram como parâmetro de QoC o sobressinal da resposta ao

degrau unitário, denotado por Δh. Adotando um cenário semelhante ao da Figura 42, é suposto que a rede apresenta distribuição de probabilidade do atraso dada por uma função similar a uma gaussiana com média μ e desvio padrão σ. Uma série de simulações é realizada com o objetivo de relacionar μ e σ ao Δh médio, e por esta razão denota-se este último como uma função $\Delta h_{méd}(\mu,\sigma)$. Considerando o cenário descrito e detalhado pelo artigo, em um primeiro momento fixa-se σ e varia-se μ, obtendo-se o gráfico da Figura 47 para $\Delta h_{méd}(\mu,\sigma)$. Cada curva é referente a um desvio padrão diferente. Em uma segunda simulação, fixa-se μ e varia-se σ, resultando no gráfico da Figura 48.

Observa-se, portanto, que $\Delta h_{méd}$ cresce com o aumento da média da distribuição do atraso. Também verifica-se que a inclinação da curva de $\Delta h_{méd}$ em função do desvio padrão do atraso, fixando-se a média, depende da média escolhida.

Além dos laços de controle em malha fechada, também existem as aplicações de controle em malha aberta. Elas aparecem onde as características do processo e seus distúrbios são bem conhecidos, fazendo com que a realimentação da Figura 40 não seja necessária. Nesses casos, a rede deve prover ao menos a transferência confiável de dados, de modo que o sinal de comando do controlador possa chegar corretamente ao atuador. Os requisitos de tempo podem variar desde sem tempo real até *hard real time*. Controles de malha aberta envolvendo diversas variáveis podem apresentar requisitos adicionais de atraso devido a dependências entre variáveis. Tipicamente, os dados de controle são enviados periodicamente, assim como no caso de malha fechada. Isso faz com que o *overhead* para prover entrega garantida, como seria o caso das retransmissões, possam ser evitados. Exemplos de aplicação de controle de malha aberta são interruptores de iluminação, tomadas inteligentes ou *displays* de cristal líquido.

A Tabela 4 relaciona tipos de aplicações de controle (linhas) com classes de sistemas de tempo real (colunas). Sinais "+" indicam forte associação, enquanto os símbolos "-" representam fraca ou nenhuma associação. O sinal "±" indica que pode haver forte ou fraca associação conforme a aplicação em questão. As colunas representam: *hard real time* (RT), *soft real time* de entrega garantida (GD) e *soft real time* de melhor esforço (BE).

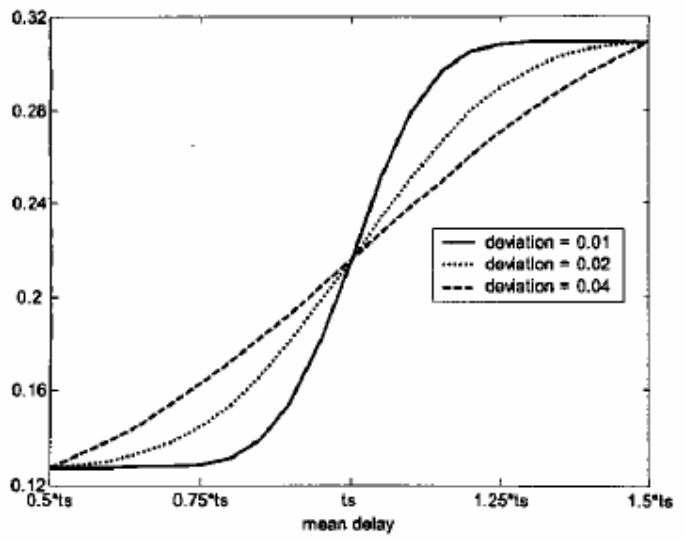


Figura 47. Sobressinal médio com desvio padrão fixo e média variando.
Extraído de Soucek, Sauter e Koller (2002).

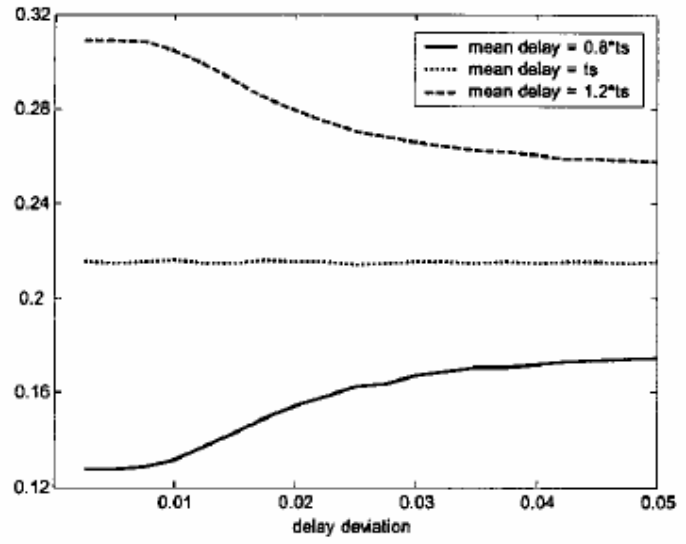


Figura 48. Sobressinal médio com média fixa e desvio padrão variando.
Extraído de Soucek, Sauter e Koller (2002).

Tabela 4 – Tipos de aplicações de controle e possíveis requisitos de tempo real

Baseado em Soucek e Sauter (2004).

Aplicação	*RT*	*GD*	*BE*
Controle em Malha Aberta	±	+ +	+
Controle em Malha Fechada	+	-	+

Os sistemas de controle baseados em rede, ou NBCS, fazem parte de um assunto muito extenso que é alvo de muita pesquisa atualmente. Quanto a este tema, o objetivo deste trabalho é apenas oferecer uma introdução e indicar relações básicas entre parâmetros de QoS e QoC. O leitor que desejar mais informações sobre o tema, principalmente do ponto de vista teórico envolvendo modelos e análises, pode consultar Lian (2001).

7.2.8 **Controle sobre LonWorks / IP experimentalmente**

Para estudos experimentais de aplicações de laços de controle utilizando redes LonWorks e IP, pode-se tanto tomar uma determinada rede IP e levantar seus parâmetros de QoS (ORLANDO, 2004) (TSANG; COATES; NOWAK, 2003) para relacioná-los com parâmetros de QoC, como construir uma situação em que os primeiros são controlados, e aí consegue-se cenários desejados para testes de aplicações de laços de controle. Para este segundo caso, propõe-se um experimento com arquitetura baseada no software NetDisturb (PDS, 2005).

O NetDisturb é um software para emulação de rede IP que pode gerar os parâmetros de QoS desejados (atraso, *jitter*, largura de banda limitada e taxa de perda de pacotes) sob configuração do usuário. É possível, portanto, criar perturbações em um fluxo de datagramas IP para estudar o comportamento de aplicações. O NetDisturb deve ser instalado em um computador com duas placas de rede Ethernet. Os pacotes são roteados entre as duas placas respeitando-se os parâmetros configurados (ex.: introdução de atraso e *jitter* conforme certa distribuição de probabilidade), e, deste modo tem-se uma rede IP emulada com as características desejadas (Figura 49).

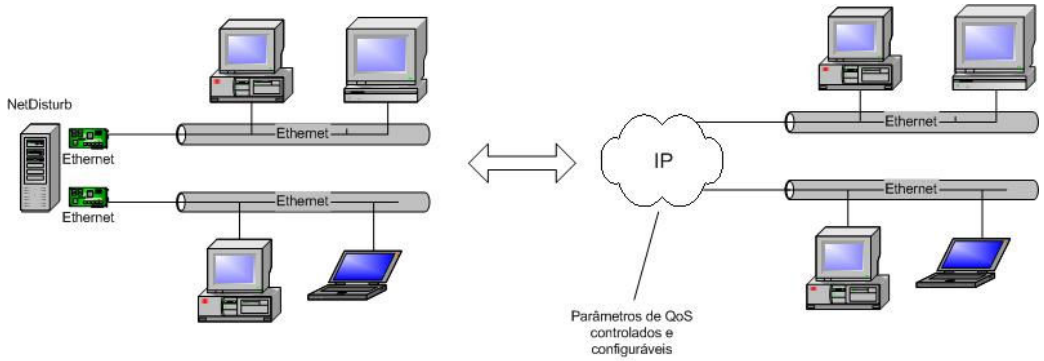


Figura 49. Software NetDisturb.

Para medir parâmetros de QoC resultantes de uma aplicação de laço de controle sobre uma rede IP, pode-se aplicar o NetDisturb para emular uma rede IP *backbone* de duas redes LonWorks, configurando-se os parâmetros de QoS desejados. Implementa-se um nó controlador e um nó "planta", que executam as funções de transferência desejadas. Esses nós podem tanto implementar as funções matematicamente, para testes, como constituírem uma aplicação real. Nesse caso, o nó "planta" será na verdade um conjunto de pelo menos um nó sensor e um nó atuador. O nó "planta" faz sentido quando se deseja simular a planta controlada, e não quando se tem uma planta real. A Figura 50 oferece uma ilustração. Formas de medir os parâmetros de QoC também devem ser implementadas nos nós.

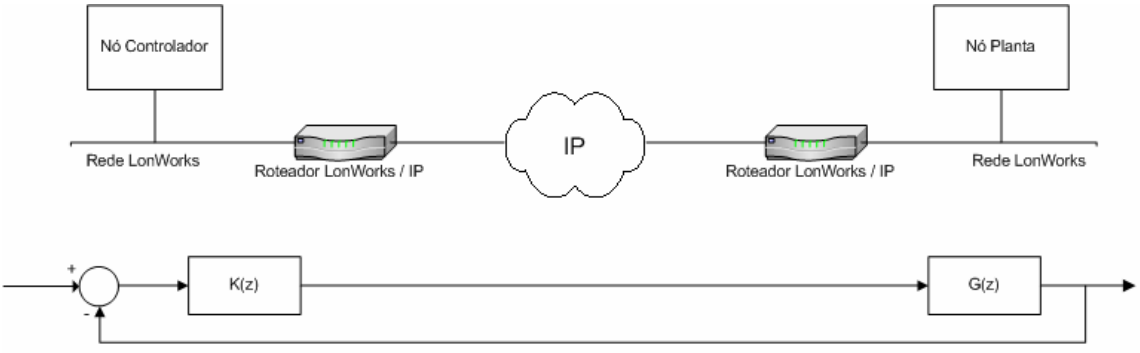


Figura 50. Experimento de laço de controle sobre redes LonWorks e IP.

Com o objetivo de verificar experimentalmente uma aplicação de controle sobre LonWorks / IP, utilizou-se dois PCs com interfaces LonWorks, dois roteadores LonWorks /

IP e um terceiro PC com duas interfaces de rede Ethernet para montar a arquitetura da Figura 50.

Em um dos PCs com interface LonWorks (PC 1) foi instalado um aplicativo específico, desenvolvido como parte deste trabalho[2], para se construir o nó controlador. Este aplicativo executa a função de transferência *K(z)* da equação (34) e envia as saídas via rede LonWorks em intervalos de tempo T_S. Cada saída é enviada em um pacote LonTalk carregando uma mensagem explícita de aplicação (TOSHIBA, 1995) através serviço *unacknowledged* sem retransmissões ($n = 0$). No outro PC com interface LonWorks (PC 2), um aplicativo similar foi instalado. A única diferença em relação ao primeiro é que este executa a função *G(z)* da equação (33). O PC 2 é, portanto, o nó "planta".

O terceiro PC mencionado não possui interface LonWorks, mas sim duas placas de rede Ethernet. O software NetDisturb é executado para rotear pacotes entre as duas placas, sendo possível configurar o atraso, *jitter* e taxa de perda de pacotes. A Figura 51 ilustra o esquema experimental construído.

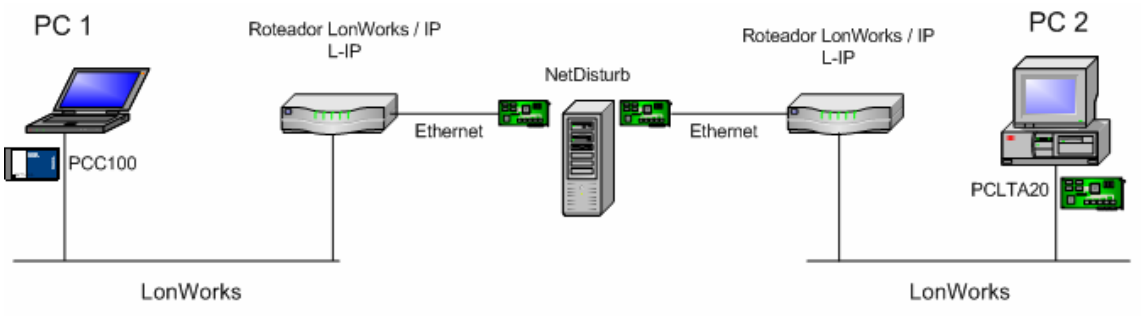


Figura 51. Esquema experimental para controle sobre LonWorks / IP

No PC 1, utiliza-se uma interface LonWorks PCC100 da Echelon (ECHELON, 2005e). A PCC100 é um cartão PCMCIA, já que se utilizou um *notebook*. O PC 2 possui uma PCLTA-20, também da Echelon. Trata-se de uma placa PCI conectada à placa mãe do computador. Os roteadores LonWorks / IP são equipamentos L-IP fabricados pela empresa Loytec (LOYTEC, 2005).

O PC 1 e o PC 2 também são diretamente conectados por uma outra rede Ethernet, não mostrada na Figura 51, que serve apenas para a sincronização do relógio. Quando a execução dos aplicativos dos nós controlador e planta é disparada, o instante inicial de ambos é

[2] O autor disponibiliza o código-fonte do aplicativo, desenvolvido em Delphi 7, através de solicitação pelo e-mail sergio.canovas@poli.usp.br.

sincronizado. A sincronização de relógio foi implementada através de um datagrama UDP enviado do PC 1 ao PC 2 quando o usuário solicita o início da execução (instante t_E). O PC 2, ao receber a mensagem de sincronização do PC 1 (instante t_R), devolve uma outra mensagem em outro datagrama UDP indicando confirmação de sincronização. O PC 2 considera o instante inicial t_0 sendo o instante de recebimento do datagrama UDP (t_R) mais um período T_S. Ou seja:

$$t_0 = t_R + T_S \tag{35}$$

Quando PC 1 receber a resposta, é necessário ter uma maneira de calcular t_0 para sincronizar este instante. T_S é conhecido (parâmetro da aplicação), e então bastaria saber t_R para aplicá-los na equação (35). Apenas o PC 2 tem o conhecimento de t_R, cujo valor poderia ser enviado no mesmo datagrama UDP que é encaminhado a PC 1. Porém, esta informação só faria sentido se os relógios dos dois PCs estivessem perfeitamente sincronizados, o que não podemos assumir como hipótese. Sendo assim, é ncessário estimar t_R.

O PC 1 tem condições de calcular o *round trip time* (RTT). O RTT é o intervalo de tempo entre o envio da primeira mensagem e o recebimento da resposta de PC 2. PC 1 estima que o atraso envolvido na entrega do primeiro datagrama a PC 2 é igual ao RTT dividido por 2. Portanto:

$$t_R \approx t_E + \frac{RTT}{2} \tag{36}$$

E, conseqüentemente:

$$t_0 \approx t_E + \frac{RTT}{2} + T_S \tag{37}$$

Desvios na sincronização do instante inicial podem influir nos resultados da aplicação de controle. No experimento considerado, os pacotes com informação de controle são enviados periodicamente em intervalos T_S tanto pelo controlador quanto pela planta a partir do instante inicial t_0. Na Figura 52, supõe-se que os instantes iniciais do PC 1 e PC 2 estão perfeitamente sincronizados. Uma simples comparação visual com a Figura 53 permite identificar a diferença nos instantes de recepção dos pacotes p_{ck} (enviados pelo nó controlador) e p_{pk} (enviados pelo nó planta) emitidos nos instantes t_k.

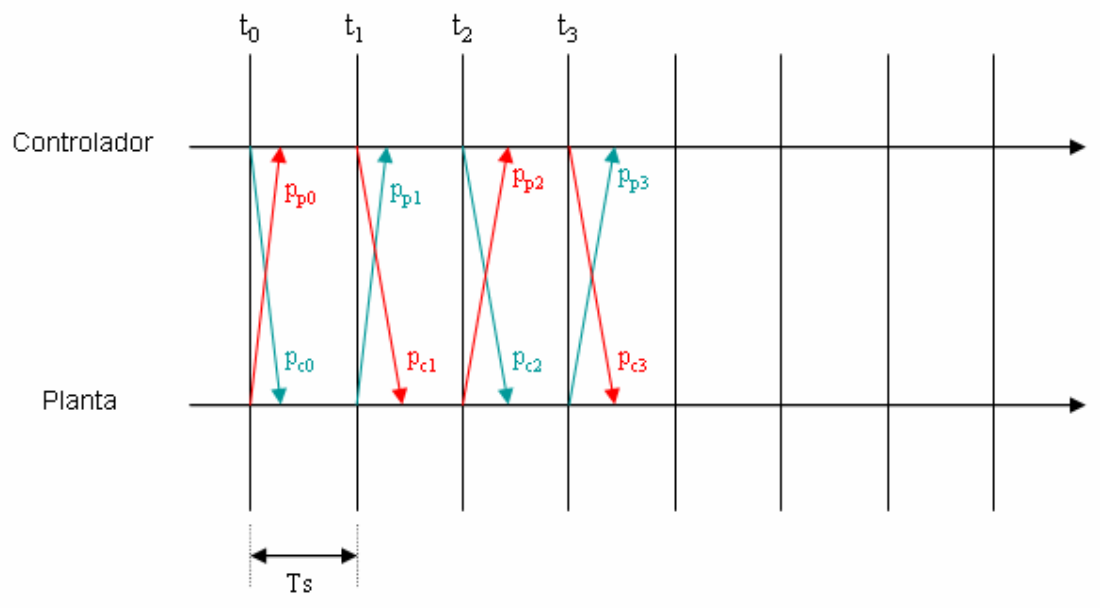


Figura 52. Instantes iniciais do controlador e da planta perfeitamente sincronizados

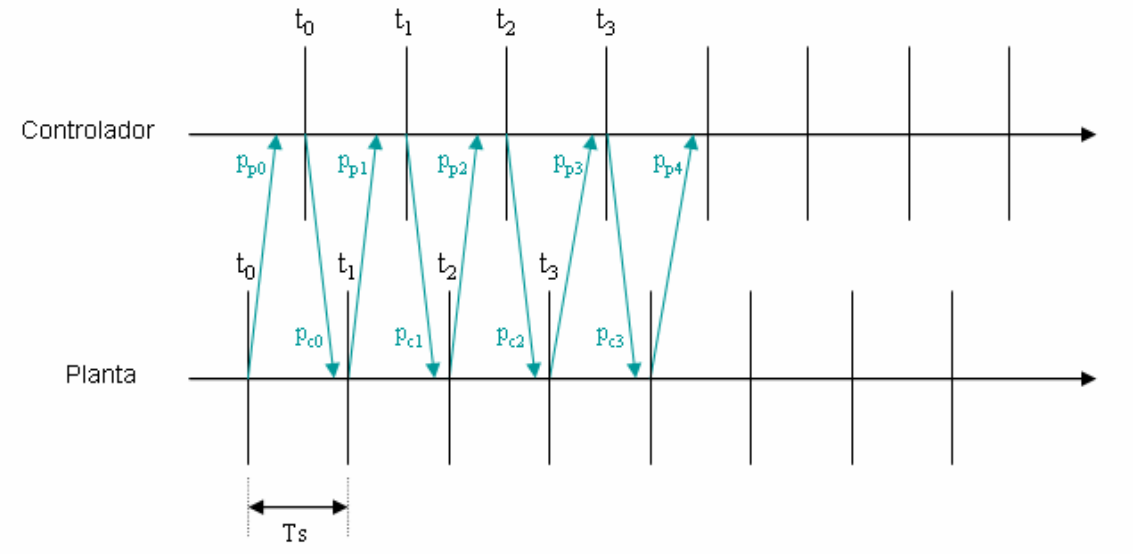


Figura 53. Instantes iniciais do controlador e da planta com defasagem de sincronização

Na Figura 52, o pacote p_{p1}, que chega ao controlador entre t_1 e t_2, inclui em seu valor alguma influência de p_{c0}, enviado em t_0. Na Figura 53, o mesmo pacote p_{p1} chega antes de t_1, e quando p_{c1} é emitido, no instante t_1, já carregará influência de p_{p1}. Na Figura 52, será somente p_{c2}, emitido em t_2, que conterá influência de p_{p1}. Observa-se, portanto, que o intervalo de tempo entre os instantes iniciais de cada nó interfere de alguma maneira na aplicação de controle. No experimento realizado, quanto mais próxima da realidade for a estimativa RTT/2 para $t_R - t_E$, mais próximos entre si estarão os instantes iniciais de cada nó.

O fato de quanto mais próximo da sincronização perfeita estiverem os instantes iniciais não quer dizer que o controle é melhor. Pelo contrário, a Figura 53 sugere, intuitivamente, que o controle ocorre de forma mais rápida. A influência da defasagem de sincronização de t_0 no controle não será estudada aqui. Observou-se na prática, para os experimentos realizados, que os instantes t_0 do controlador e da planta estão suficientemente sincronizados de modo a se obter o cenário da Figura 52.

Os aplicativos que rodam no PC 1 e PC 2 são, na verdade, o mesmo. Ele apresenta uma tela simples onde se pode selecionar o nome do dispositivo de interface LonWorks instalado na máquina (Figura 54). Uma outra caixa de seleção determina o modo de execução: controlador ou planta. O endereço LonWorks do nó planta foi previamente configurado para domínio 1, subrede 1 e nó 126. O do nó controlador, por sua vez, foi ajustado para domínio 1, subrede 2 e nó 126. Quando o usuário pressiona o botão “Iniciar”, o modo de execução selecionado é verificado. Se foi escolhido o modo controlador, automaticamente o endereço para onde os pacotes LonTalk são enviados é ajustado para o endereço do nó planta, e vice-versa. Antes de enviar o primeiro pacote da aplicação de controle, PC 1 envia o datagrama UDP sinalizando início a PC 2. O instante inicial em cada nó acontece de acordo com as equações (35) e (37), respectivamente na planta e no controlador. Supõe-se que estão significativamente sincronizados. Os intervalos periódicos T_S passam então a ser computados em cada nó. Nos instantes $t_k = kT_S$, com k = 0, 1, 2, 3 ... , tanto o nó controlador quanto o nó planta enviam novo pacote LonTalk com o último valor calculado através de sua função de transferência.

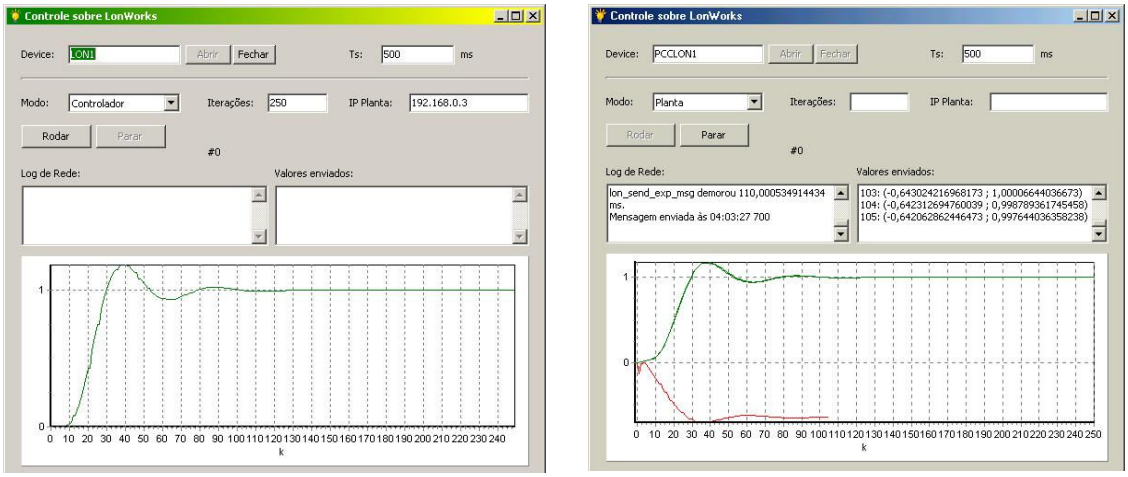


Figura 54. Aplicativo desenvolvido para realização de experimento de controle

Para implementação dos aplicativos que emulam o controlador (execução em PC 1) e planta (execução em PC 2), foi utilizada uma biblioteca de acesso a redes LonWorks da empresa P2S (P2S, 2005), construída com base na *OpenLDV* da Echelon (ECHELON, 2005e), que por sua vez utiliza uma outra biblioteca de baixo nível de acesso às interfaces LonWorks desta empresa. Nos computadores PC 1 e PC 2, constatou-se que o tempo de execução de uma instrução de envio de mensagem LonTalk com serviço *unacknowledged* via interface LonWorks está em torno de 110 ms. Sendo assim, com essas bibliotecas, não é possível realizar o experimento com T_S abaixo deste valor, como seria necessário para reproduzir a simulação de Soucek, Sauter e Koller (2003) onde T_S = 10 ms. Todavia, para este experimento, pode-se aumentar a escala de tempo e considerar um T_S maior (500 ms por exemplo), desde que se aumente proporcionalmente o atraso médio e o *jitter* introduzido pelo software NetDisturb. Se essas grandezas aumentarem proporcionalmente a T_S, os efeitos sobre a aplicação de controle são os mesmos, pois o que afeta o cálculo das funções de transferência não são os instantes absolutos em que os pacotes chegam, mas sim seu instante relativo com respeito aos instantes $t_k = kT_S$, com k = 0, 1, 2, 3, etc. Sendo assim, os resultados do controle podem ser analisados em função dos inteiros k, e não dos instantes t_k. Se T_S mudar, espera-se os mesmos resultados desde que o atraso médio e o *jitter* mudem na mesma escala.

Para a implementação de um controlador real em um PC, provavelmente seria necessário otimizar essas bibliotecas ou então construir nós LonWorks mais rápidos que um PC com elas.

Para visualizar os efeitos de uma rede IP no caminho entre o nó controlador e o nó planta, o experimento foi executado com duas montagens:

A. PC 1 e PC 2 conectado diretamente na mesma rede LonWorks, sem rede IP intermediária (Figura 51 sem roteadores e sem NetDisturb, mas sim com PC 1 e PC 2 ligados na mesmo canal físico LonWorks);
B. PC 1 e PC 2 de acordo com esquema da Figura 51.

A execução com a montagem B foi repetida oito vezes, utilizando-se os parâmetros abaixo. T_S e $1/\lambda$ foram mantidos constantes, com D_{min} variando. Essas execuções são referidas por B_1, B_2, B_3, B_4, B_5, B_6, B_7 e B_8.

$T_S = 500$ ms

$f_J(J) = \lambda \exp(-\lambda J)$, com $1/\lambda = 250$ ms

D_{min} = 0 ms (B_1), 250 ms (B_2), 500 ms (B_3), 1000 ms (B_4), 1500 ms (B_5), 2000 ms (B_6), 2500 ms (B_7), 3000 ms (B_8)

Em todos os casos, foi aplicada a função degrau unitário no *set-point* do controlador[3].

Na montagem A, pelo fato de ambos os nós envolvidos na comunicação estarem no mesmo canal LonWorks, sem roteadores intermediários e sem outros nós compartilhando a banda, tem-se a situação de atraso mínimo (apesar de não ser nulo, pode ser considerado nulo quando comparado à ordem de grandeza dos atrasos da montagem B). O experimento com a montagem A foi executado 10 vezes. As curvas obtidas encontram-se sobrepostas na Figura 55.

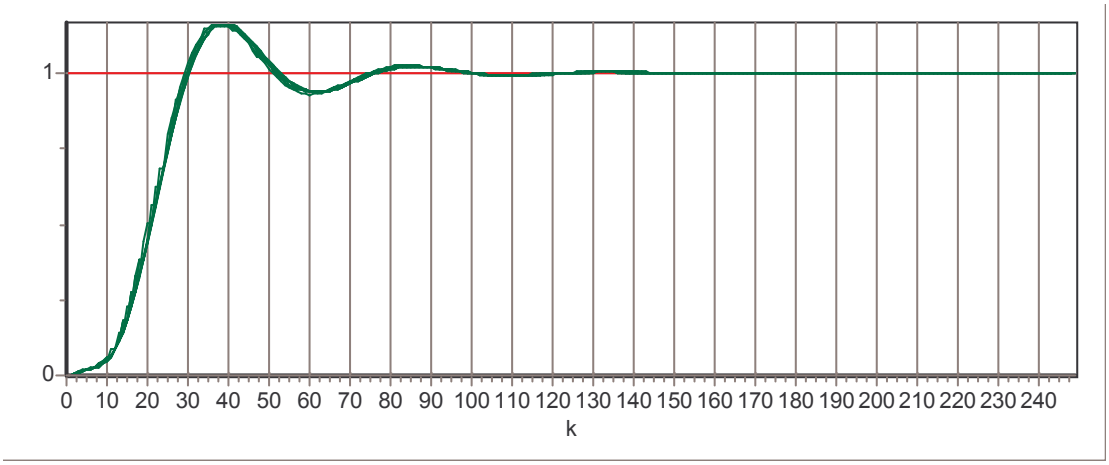


Figura 55. Controle sobre LonWorks sem rede IP intermediária

[3] O autor disponibiliza uma planilha com os pontos de todas as curvas de resultado, das montagens A e B, através de solicitação pelo e-mail sergio.canovas@poli.usp.br.

No caso B_4, todos os valores são proporcionais àqueles aplicados na simulação apresentada por Soucek, Sauter e Koller (2003). A taxa de perda de pacotes foi configurada como nula no NetDisturb, isto é, $1 - w = 0$. As funções de transferência correspondem às equações (33) e (34). Os mecanismos de agregação e seqüenciamento foram desligados nos roteadores LonWorks / IP.

Para cada conjunto de parâmetros da montagem B, o experimento foi realizado 10 vezes com k variando de 0 a 299, isto é, 300 iterações. Tem-se um total de 80 curvas de resultado. Todas elas foram sobrepostas na Figura 56. As curvas correspondentes a cada conjunto de parâmetros estão diferenciadas por cores. O conjunto de B_1 apresenta o menor sobressinal, o de B_2 o segundo menor, e assim por diante. B_8 é a curva mais lenta.

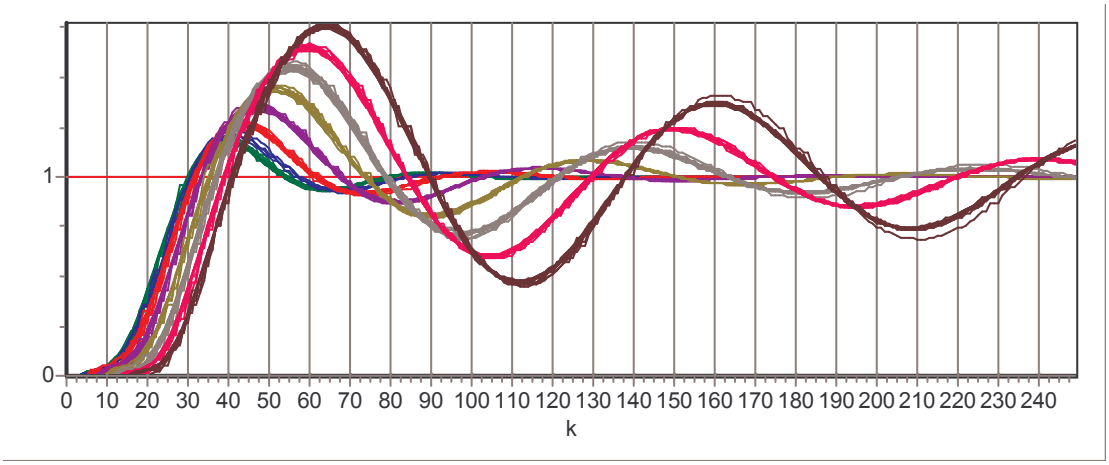


Figura 56. Controle sobre LonWorks/IP – influência do atraso da rede no sinal controlado

Assim como Soucek, Sauter e Koller (2002), toma-se o sobressinal médio (Δh) como parâmetro de QoC. Observa-se que Δh cresce com o aumento de D_{min}, mantendo λ constante, conforme a Tabela 5 e a Figura 57.

Tabela 5 – Relação entre Δh e D_{min} para o experimento na configuração B_1.

D_{min} (ms)	D_{min} / T_S	Δh
0	0	0,1829
250	0,5	0,2142
500	1	0,2714
1000	2	0,3556
1500	3	0,4469
2000	4	0,5532
2500	5	0,6529
3000	6	0,7614

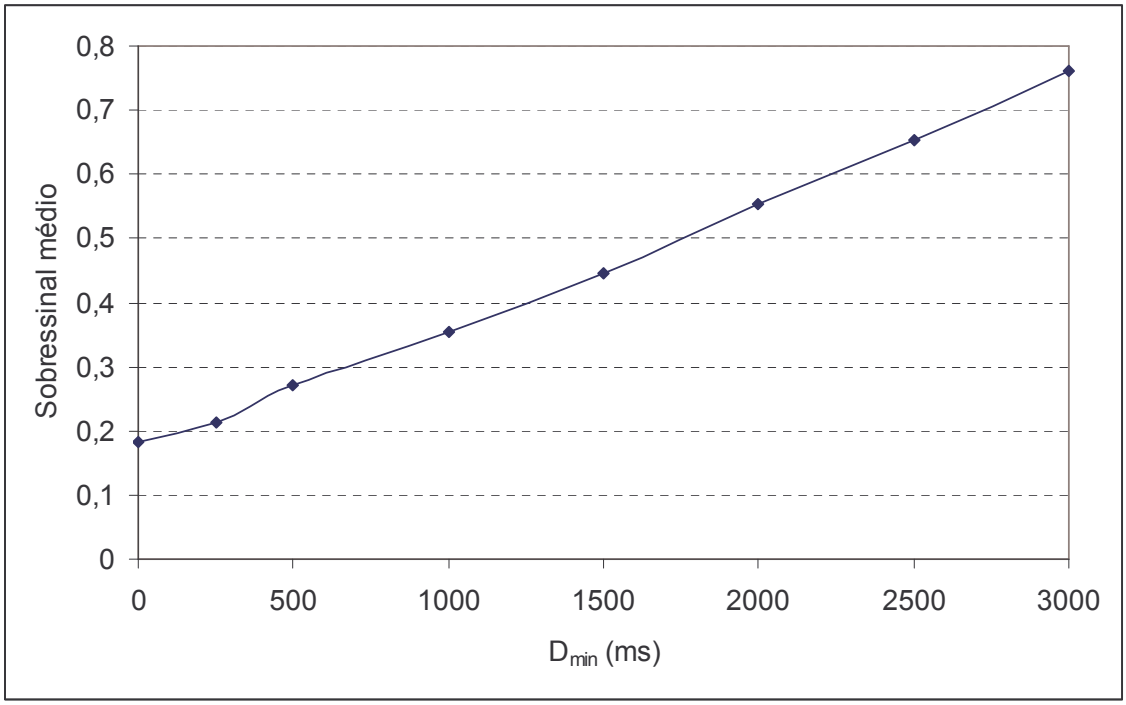


Figura 57. Sobressinal médio em função do atraso mínimo para o experimento realizado

O gráfico da Figura 57 sugere uma relação linear entre $D_{\min}$ e Δh na faixa estudada (0 a 3000 ms) para *jitter* com distribuição exponencial e $1/\lambda = 250$ ms. Não é possível comparar quantitativamente esses resultados com os da simulação de Soucek, Sauter e Koller (2002) (Figura 47), pois neste artigo foram utilizadas outras funções de transferência e outra distribuição de atraso e *jitter*. Todavia, observa-se coerência no fato de ambos os casos apresentarem relação crescente entre atraso médio e sobressinal médio. Isso pode ser explicado pelo fato de que, quanto maior o atraso, maior o tempo que o controlador demora para responder à planta. Quando o sinal controlado ultrapassa o valor 1 durante a primeira subida, o controlador precisa de mais tempo para conseguir inverter a tendência de crescimento e retornar em direção ao 1. Esse tempo adicional possibilita que o sinal atinja um valor máximo maior, aumentando Δh.

Grandes atrasos usualmente estão relacionados a caminhos em redes com diversos segmentos, enlaces e roteadores, como por exemplo a Internet. Por outro lado, e de maneira independente, *jitter* elevado comumente representa uma situação de rede congestionada, submetida à alta carga de uso. A combinação adequada do atraso mínimo e da distribuição do *jitter* como parâmetros do NetDisturb é de fundamental importância para emular o tipo de rede desejada (LAN, MAN ou WAN) na situação de carga de interesse.

Existem muitas outras possibilidades para realizar experimentos de controle sobre LonWorks / IP: testar outras funções de transferência, fixar $D_{\min}$ e variar λ, verificar outras distribuições de probabilidade para o *jitter*, utilizar outros parâmetros para QoC além de Δh, etc. O escopo deste trabalho é apresentar este assunto de maneira introdutória, ilustrando-o com resultados de simulações obtidos na literatura e com um experimento desenvolvido como parte deste trabalho. As referências citadas fornecem uma base adicional.

Uma possibilidade de se obter mais resultados com o mesmo experimento é executá-lo com outros valores de $D_{\min}$ não múltiplos de T_S, ou seja, onde $D_{\min}/T_S$ não são inteiros. Aqui, apenas o caso B_2 ($D_{\min}$ = 250ms) respeita esta condição. É possível que a relação linear da Figura 57 não seja tão perfeita, ou até mesmo apresente um perfil diferente ao considerar esses pontos intermediários. Também é provável que as curvas obtidas para um mesmo conjunto de parâmetros não sejam tão próximas umas das outras como na Figura 56, onde é possível distinguir plenamente cada conjunto de curvas mesmo que não estivessem indicados por cores diferentes. Isso porque quando $D_{\min}$ é múltiplo de T_S, a tendência é que a curva de probabilidade do atraso total de um pacote ($D_{\min} + J$) se "encaixe" com algum dos períodos T_S (Figura 52), fazendo que com que a maioria dos pacotes cheguem após um número fixo de intervalos T_S a partir de sua geração. Quando $D_{\min}/T_S$ não é inteiro, as curvas de densidade de probabilidade do instante de chegada de cada pacote não mais se "encaixam" nos intervalos T_S, fazendo que com que esta probabilidade fique mais divida entre dois períodos T_S consecutivos. Portanto, em uma mesma execução, pacotes consecutivos passam a chegar ao destino não mais após um número fixo de intervalos T_S, mas sim após um número de intervalos que depende da posição relativa entre a curva de densidade de probabilidade mencionada e os instantes t_k, impactando na forma e distribuição da curva da variável controlada.

7.3 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo relacionou as quatro categorias de aplicações apresentadas no Capítulo 1 com as abordagens de interconexão entre redes LonWorks e IP do Capítulo 5. Também foram relacionados os conceitos e parâmetros definidos no Capítulo 6. Do início do Capítulo 5 até aqui, buscou-se responder as quatro questões apresentadas na seção 5.1.

Um experimento relacionado ao tópico de controle sobre LonWorks / IP foi implementado e serve como ponto de partida para a realização de novas investigações ou até mesmo para o projeto de uma aplicação real. Construiu-se um aplicativo para ser executado

em um PC com interface de rede LonWorks (por exemplo, utilizando uma PCLTA) que atua como um nó em dois modos possíveis: controlador ou planta. Ao executar este aplicativo simultaneamente em dois PCs conectados por redes conforme a montagem da Figura 51, foi possível, com auxílio do software NetDisturb, avaliar quantitativamente a relação entre parâmetros de QoS de uma rede IP e um parâmetro de QoC previamente estabelecido para uma aplicação de laço de controle.

8 CONSIDERAÇÕES FINAIS

Este trabalho apresentou a integração entre redes LonWorks e redes IP sob diversos aspectos. Aplicações, requisitos e soluções foram mostrados e discutidos, destacando-se alguns resultados. Considerou-se quatro tipos de aplicações para a integração entre redes LonWorks e redes IP, a saber:

1. Monitoração e registro (supervisório);
2. Gerenciamento;
3. Interconexão de redes LonWorks via *backbone* IP;
4. Interconexão de redes LonWorks via *backbone* IP para laço de controle (presença de requisitos fortes de tempo real).

Tendo em vista essas aplicações, buscou-se responder quatro perguntas. Resumidamente, as seguintes respostas foram dadas:

a. Que tipos de soluções existem para a interconexão entre redes LonWorks e IP?

Resposta: Interfaces de redes LonWorks baseadas em IP, roteadores e *gateways*. Para os dois primeiros, pode ser usado o padrão da norma EIA-852.

b. Que tipos de parâmetros de rede devem ser considerados para uma solução baseada em redes LonWorks / IP? Como eles podem ser utilizados para estabelecer requisitos de uma certa aplicação?

Resposta: Os sistemas podem ser classificados de acordo com categorias de aplicações de tempo real. Para este trabalho, consideraram-se três delas: *hard real time*, *soft real time* com entrega garantida e *soft real time* de melhor esforço. Essas categorias oferecem uma indicação inicial da busca por requisitos mais precisos baseados em parâmetros de QoS de rede, que podem ser estabelecidos pelas diversas equações estudadas no Capítulo 5. Dentre esses requisitos, foram abordados o atraso fim-a-fim (*delay*), a variação do atraso (*jitter*), a taxa de transmissão, largura de banda (*throughput* e *bandwidth*), taxa de perdas e seqüência de pacotes. Usualmente, a categoria de tempo real determina um requisito para o atraso, o qual está relacionado com o *jitter* e largura de banda por (17) e (21). A taxa de perdas está

relacionada com a capacidade da infra-estrutura de rede e o quanto ela está sendo exigida em um determinado instante. O requisito de entrega de pacotes na seqüência correta relaciona-se com o *jitter* através de (25). O conhecimento desses requisitos de QoS permite tanto verificar se uma determinada infra-estrutura de rede atende a uma aplicação quanto projetar e construir uma nova infra-estrutura para atendê-la.

c. A quais requisitos cada uma das quatro categorias de aplicações acima está associada?

Resposta: A Tabela 6 condensa as informações das Tabelas 3 e 4, associando tipos de aplicações a categorias de sistemas de tempo real. Os requisitos exatos para os parâmetros de QoS dependem de uma análise precisa da aplicação em questão (por exemplo,.: verificação de tempo máximo para recepção de alarmes do software supervisório, sobressinal médio desejado para uma resposta a degrau unitário de um sistema de controle, etc.)

Tabela 6 – Resumo de tipos de aplicações e possíveis requisitos de tempo real.
Baseado em Soucek e Sauter (2004).

Aplicação	*RT*	*GD*	*BE*
Monitoração / Registro	+	+	±
Gerenciamento	±	+ +	-
Controle em Malha Aberta	±	+ +	+
Controle em Malha Fechada	+	-	+

d. Quais tipos de soluções, dentre as apresentadas, melhor atendem a cada uma das quatro categorias de aplicações?

Resposta: Para monitoramento e gerenciamento, as soluções tipicamente mais adequadas são baseadas na abordagem de *gateway*. Para interconexão de redes LonWorks via rede IP com ou sem requisitos de tempo real, as mais adequadas são baseadas na abordagem de roteador.

8.2 CONTRIBUIÇÕES

Este trabalho apresenta algumas contribuições relevantes para o contexto de pesquisa do Laboratório de Automação Agrícola da USP, relacionando-se também com outros trabalhos e projetos vinculados a aplicações da tecnologia LonWorks.

Dentre as principais contribuições, podem-se citar:

- Uso de parâmetros de rede quantitativos para estabelecimento de requisitos de aplicações LonWorks/IP: visando prover alguma ferramenta para projeto e avaliação de soluções baseadas em redes LonWorks e redes IP, este texto menciona algumas definições de sistemas de tempo real e estabelece uma série de parâmetros de rede comuns que são utilizados para compor requisitos de aplicações. Esses requisitos podem servir tanto para construir uma infra-estrutura de rede e equipamentos que implementem uma determinada aplicação como para avaliar se uma infra-estrutura já existente atende adequadamente uma situação. Foi realizado significativo esforço no sentido de reunir, organizar e harmonizar as informações, bem como padronizar nomenclatura e notação de diversas referências. Novas explicações, detalhamentos, exemplos e ilustrações foram introduzidos visando o aprimoramento e enriquecimento do conteúdo.

- Criação de aplicativo para experimento de controle sobre LonWorks/IP: baseando-se em simulações encontradas nas referências, ferramentas de software e equipamentos disponíveis no mercado, foi proposto e realizado um experimento para relacionar parâmetros de QoS de rede com parâmetros de QoC de um laço de controle. Para isso, foi criado um pequeno aplicativo que funciona em dois modos: controlador e planta. Quando executado em dois computadores dispostos em uma configuração na qual entre eles existe um simulador de rede IP (implementado através do software NetDisturb), cada qual rodando em um dos modos, é possível medir o sobressinal médio como parâmetro de QoC. Pode-se variar os parâmetros de QoS da rede IP simulada através do NetDisturb. Dessa maneira, obtém-se uma relação entre os dois. Mais importante que os resultados obtidos, que consistem na medição de apenas um parâmetro de QoC e se relacionam a um laço de controle com funções de transferência bastante específicas, o aplicativo desenvolvido pode ser utilizado como ponto de partida para novos experimentos, seja para testar

outros laços, medir outros parâmetros, ou ambos. Pode ser utilizar também como ponto de partida para a implementação de um controlador LonWorks real.

- Reunião de referências sobre a tecnologia LonWorks: existe muita disponibilidade de referências sobre a tecnologia LonWorks do ponto de vista comercial, que explica a tecnologia de maneira básica e foca em produtos específicos de determinados fabricantes. O conhecimento mais detalhado e técnico da tecnologia como um todo, na profundidade de maior interesse para um pesquisador ou engenheiro, encontra-se espalhado em diversos manuais técnicos e normas de difícil acesso e compreensão. Com o conhecimento adquirido pelo autor ao longo da execução deste trabalho e de trabalhos anteriores que incluem experiências práticas, desenvolvimento de softwares, levantamento de referências em vários artigos e manuais, entre outros, este trabalho reúne diversos conceitos técnicos relacionados à tecnologia LonWorks de maneira organizada e objetiva, principalmente com respeito ao protocolo LonTalk, indicando também outras referências apropriadas para cada tópico. O material aqui apresentado se destina tanto como um elemento de aprendizagem para o leitor iniciante quanto como uma referência bastante útil para o leitor já conhecedor da tecnologia. Destaque para o Apêndice B, que mostra o protocolo LonTalk sob o ponto de vista de pacotes e os explica campo a campo.

- Relacionamento entre a interconexão LonWorks/IP e categorias de aplicações: a interconexão entre redes LonWorks e redes IP pode ser implementada de diversas maneiras, cada qual apropriada para um determinado tipo de aplicação. Este trabalho reúne o conhecimento espalhado por diversas referências, divide as soluções de interconexão em categorias mais gerais e as relaciona objetivamente com os principais tipos de aplicações existentes. O trabalho constitui-se em uma referência interessante para o pesquisador ou engenheiro que precisa construir ou avaliar uma solução baseada em redes LonWorks e redes IP.

8.3 TRABALHOS FUTUROS E MELHORIAS

Muitas questões importantes foram apenas citadas, tratadas de maneira introdutória ou nem mesmo mencionadas. Novos trabalhos de pesquisa, como continuidade deste, poderiam abordá-las mais profundamente. Algumas delas são:

- Segurança: questões de segurança relacionadas à interconexão de redes LonWorks e redes IP não foram tratadas neste trabalho. A aplicação de criptografia dos dados de redes de controle que trafegam sobre um *backbone* IP e o uso de chaves para autenticação de roteadores são itens interessantes cujo impacto nos requisitos de aplicações poderiam ser mais bem estudados;

- Gerenciamento por SNMP: foi abordado de maneira básica um modelo de gerenciamento de redes *fieldbus* através de redes IP baseado em SNMP segundo Kunes e Sauter (2001). No entanto, os detalhes deste tipo de implementação, tais como o esquema de uma MIB, podem ser mais bem avaliados e definidos. Entre os benefícios desta pesquisa, podem-se viabilizar que o gerenciamento das redes de controle de uma instalação predial corporativa, por exemplo, possa ser realizado em conjunto com o gerenciamento das redes de dados sob responsabilidade da equipe de TI. Tudo isso utilizando ferramentas já existentes, conforme propõe um trabalho em desenvolvimento no Laboratório de Automação Agrícola da USP[4], além dos autores já citados.

- Método para levantamento de requisitos de QoS: ao longo deste trabalho, foram citados parâmetros de QoS aplicáveis como requisitos para diversas aplicações. Também foram mostrados alguns exemplos. No entanto, não foi apresentado um método formal para levantamento de requisitos de QoS com base em uma determinada aplicação. A elaboração de um método, mesmo que restrito para alguns casos é uma sugestão de trabalho futuro. Soucek, Sauter e Rauscher (2001) apresentam uma tentativa para aplicações de controle.

- Método para avaliação ou construção de redes que atendam determinados requisitos de QoS: este tópico representa o caminho inverso do tópico anterior. Conhecendo-se requisitos de QoS a serem atendidos, podem-se desejar construir uma infra-estrutura de rede compatível ou próxima das necessidades. A elaboração de um método sistemático para essa construção ou então para avaliação de uma rede existente, pode constituir um bom trabalho de pesquisa. Georges, Rondeau e

[4] CHERMONT, M.G. Gerenciamento de uma rede de automação LonWorks® através de um agente SNMP. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo. São Paulo, 2006. /Em elaboração/

Divoux (2005) associam a implementação de classes de serviço, ou *Classification of Service* (CoS), para que uma rede atenda ou se aproxime de requisitos de QoS.

- Controle sobre IP: Neste trabalho, este tópico foi apresentado apenas de maneira introdutória utilizando-se resultados baseados em trabalhos já existentes e um experimento básico. Outros experimentos para relacionar parâmetros de QoS e de QoC podem ser realizados, servindo tanto como testes de viabilidade para aplicações específicas quanto como bases para construção de teorias. Lian (2001) aborda aspectos teóricos com alguma profundidade.

- Melhorias no experimento realizado: Mencionaram-se os efeitos da má sincronização dos relógios dos computadores utilizados no experimento do Capítulo 7. Esta influência poderia ser estudada de modo a melhor avaliar e compreender os resultados. Uma outra sugestão é realizar o experimento em uma nova configuração, onde as duas instâncias do aplicativo desenvolvido (controlador e planta) fossem executadas no mesmo computador ao invés de dois. Nesse caso, nesse único computador precisariam estar instaladas duas placas de rede LonWorks (por exemplo, PCLTAs). É como se o PC 1 e o PC 2 da Figura 51 fossem "mesclados" em um único PC. Nesse caso, as duas instâncias do aplicativo teriam o mesmo relógio como referência, já que a execução ocorre na mesma máquina, e não haveria mais o problema da sincronização.

Como consideração final, destaca-se que a área de automação tem passado por diversas evoluções tecnológicas, através das quais surgem novos conceitos, tecnologias, equipamentos, fabricantes, padrões, normas, etc. Portanto, as alternativas de elementos para construção de soluções são cada vez maiores e abre-se espaço constantemente para discussões e debates. Muitas vezes a melhor solução técnica acaba inviabilizada por questões econômicas ou até mesmo por pressões comerciais de concorrentes. Mesmo que o futuro divirja das tendências observadas no Capítulo 1, este trabalho analisa e baseia-se em alternativas tecnológicas comuns atualmente, considerando também o contexto de pesquisa do Laboratório de Automação Agrícola da USP, seus projetos e perspectivas.

REFERÊNCIAS

Referências citadas:

ADEPT SYSTEMS INC. **Using IP as a backbone for ANSI 709.1 Networks.** 2002. Disponível em: <http://www.adeptsystemsinc.com/pdf/IPBackbone1.1.pdf>. Acesso em: 18 de Outubro de 2005.

ALVES FILHO, M.S.; CUGNASCA, C.E.; DIAS, E.M. LonWorks: Inovação Tecnológica para Automação Residencial, Predial e Industrial. In: CONAI – CONGRESSO E EXPOSIÇÃO INTERNACIONAL DE AUTOMAÇÃO, 11., São Paulo. 2004. **Programação Temária do Congresso.** São Paulo: Conai, 2004. Disponível em: <http://www.conai.com.br/palestras2004/13_05_04_15h30_Miguel.zip>. Acesso em: 20 de Fevereiro de 2005.

ANANTHAMURTHY, L. Introduction to Web Services. **Developer.com.** 2004. Disponível em: <http://www.developer.com/services/article.php/1485821> Acesso em: 27 de Novembro de 2004.

ARCWEB. $ 24 BILLION BUILDING AUTOMATION MARKET – INTEGRATED FACILITIES MANAGEMENT IS KEY. **Informações sobre o mercado norte-americano de automação predial.** 2002. Disponível em: <http://www.arcweb.com/Community/arcnews/arcnews.asp?ID=272>. Acesso em: 21 de Fevereiro de 2005.

ASTRÖM, K.J. Process Control – Past, Present, and Future. **IEEE Control Systems Magazine**, v.5, n.3, p. 3-10, 1985.

BBDSOFT. AUTOMATION AND REAL-TIME COMPUTING TERMS. **Glossário de termos relacionados à automação e computação de tempo real.** Disponível em: <http://www.bbdsoft.com/glossary.html>. Acesso em: 20 de Fevereiro de 2005.

CANOVAS, S.R.M.; CHERMONT, M.G. **RemoteLon: Sistema Supervisório Remoto para Redes LonWorks.** 2003. 101p. Projeto de Formatura – Escola Politécnica, Universidade de São Paulo. São Paulo, 2003.

CANOVAS, S.R.M.; CHERMONT, M.G.; ALVES FILHO, M.S.; CUGNASCA, C.E.; Sistema Supervisório para Redes LonWorks. In: CONAI – CONGRESSO E EXPOSIÇÃO INTERNACIONAL DE AUTOMAÇÃO, 11., São Paulo. 2004. **Programação Temária do Congresso.** São Paulo: Conai, 2004. Disponível em: <http://www.conai.com.br/palestras2004/13_05_04_15h_Marlon.zip>. Acesso em: 20 de Fevereiro de 2005.

CANOVAS, S.R.M.; CHERMONT, M.G.; CUGNASCA, C.E.; Remote Monitoring and Actuation Based on LonWorks® Technology. In: 2005 EFITA/WCCA JOINT CONGRESS ON IT IN AGRICULTURE, 2005, Vila Real, Portugal. Proceedings of 2005 EFITA/WCCA Joint Congress on IT in Agriculture. Vila Real, Portugal: Universidade de Trás-os-Montes e Alto Douro, 2005. v. 1, p. 812-819.

CASTRUCCI, P.L.; MORAES, C.C. **Engenharia de Automação Industrial.** 1.ed. Rio de Janeiro: LTC Editora, 2001. 295p.

CHATARJI, J. Introduction to Service Oriented Architecture (SOA). **Devshed.com.** Outubro de 2004. Disponível em: <http://www.devshed.com/c/a/Web-Services/Introduction-to-Service-Oriented-Architecture-SOA/> Acesso em: 27 de Novembro de 2004

COMER, D.E. **Internetworking with TCP/IP – Volume 1 - Principles, Protocols and Architectures**. 4.ed. Upper Saddle River: Prentice Hall, 2000. 750p.

COMMWEB. BANDWIDTH, THROUGHPUT, AND GOODPUT. **Definições e diferenças entre os termos bandwidth, throughput e goodput.** 2002. Disponível em: <http://www.commweb.com/8706356>. Acesso em: 15 de Novembro de 2005.

ECHELON CORPORATION. ECHELON TECHNOLOGY ADOPTED AS EUROPEAN BUILDING CONTROL STANDARD. **Matéria sobre a adoção de LonWorks como padrão europeu.** (a) Disponível em: <http://www.echelon.com/about/press/ahr2005.htm>. Acesso em: 03 de Abril de 2005.

_____. **Introduction to the LONWORKS® System.** Palo Alto, Echelon, 1999 (Relatório 078-0183-01A).

_____. **LNS DDE Server User's Guide.** Palo Alto, Echelon, 2001 (Relatório 078-0170-01C).

_____. **LNS Programmer's Guide.** Palo Alto, Echelon, 2004 (Relatório 078-0177-01F).

_____. **LonMaker User's Guide.** Palo Alto, Echelon, 2003 (a) (Relatório 078-0168-02G).

_____. **LonMark Layer 1-6 Interoperability Guidelines.** Palo Alto, Echelon, 2002 (Relatório 078-0120-01F).

_____. **Neuron C programmer's guide.** Palo Alto, Echelon, 1995 (Relatório 078-0002-02G).

_____. **NodeBuilder User's Guide.** Palo Alto, Echelon, 2003 (b) (Relatório 078-0141-01E).

_____. OPEN SYSTEMS OVERVIEW. **Fornece visão geral sobre a arquitetura de redes LonWorks.** (b) Disponível em: <http://www.echelon.com/solutions/opensystems>. Acesso em: 21 de Fevereiro de 2005.

_____. **OpenLDV Programmer's Guide.** Palo Alto, Echelon, 2005 (c) (Relatório 078-0275-01C).

_____. PANORAMIX ENTERPRISE PLATFORM 2.0. **Fornece visão geral sobre a plataforma Panoramix.** (d) Disponível em: <http://www.echelon.com/products/middleware/panoramix/documents/Panoramix_DS.pdf>. Acesso em: 18 de Outubro de 2005.

_____. **Site da empresa Echelon Corporation.** (e) Disponível em: <http://www.echelon.com>. Acesso em: 20 de Fevereiro de 2005.

_____. SOLUTIONS USING LONWORKS TECHNOLOGY. **Exemplos de soluções e aplicações da tecnologia LonWorks.** (f) Disponível em: <http://www.echelon.com/solutions>. Acesso em: 03 de Abril de 2005.

GAW, D.; MARSH, A. **Architectural issues related to Ethernet TCP/IP Connectivity to LonWorks.** Disponível em: <http://www.coactive.com/pages/wp9702.html>. Acesso em: 03 de Outubro de 2005.

GEORGES, J.P.; RONDEAU, G.E.; DIVOUX, T. Classification of Service for Networked Control Systems. **1st Workshop on Networked Control System and Fault Tolerant Control.** Disponível em: <http://www.strep-necst.org/rubrique.php3?id_rubrique=45>. Acesso em: 08 de Dezembro de 2005.

KIM, B.H.; CHO, K.H.; PARK, K.S. Towards LonWorks Technology and its applications to automation. Science and Technology, 2000. KORUS 2000. **Proceedings**. The 4th Korea-Russia International Symposium on, Junho/Julho/2000. v.2. p.197 –202.

KOOPMAN, P.J. Lost Messages and System Failures. **Embedded Systems Programming**, v.11, n.9, p. 38-52, Outubro de 1996.

KUNES, M.; SAUTER, T. Fieldbus-Internet Connectivity: The SNMP Approach. **IEEE Transactions on Industrial Electronics**, v. 48, n. 6, p. 1248-1256, Dezembro de 2001.

LIAN, F.L. **Analysis, Design, Modeling, and Control of Networked Control Systems.** 2001. 179p. Tese (Doutorado) – University of Michigan. Dearborn, 2001. Disponível em: <http://www.eecs.umich.edu/~impact/Publications/200105DissertationLian.pdf>. Acesso em: 3 de Dezembro de 2005.

LOYTEC. **Networking IP-852 Nodes: How to use Ethernet/IP as Communication Channel to Network IP-852 Nodes**. Disponível em: <www.lonworldexpo.com/archive/2003/conference/presentations/HansJoergSchweinzer/LW2003LoytecPresentation.pdf>. Acesso em: 09 de Outubro de 2005.

_____. **Site da empresa Loytec.** Disponível em: <http://www.loytec.com>. Acesso em: 05 de Outubro de 2005.

MAHALIK, N.G.P.C.; LEE, S.K. A study on production line automation with LonWorks™ control networks. **Computer Standards & Interfaces**, n.24, p.21-27, 2002.

ORLANDO, E. **Applicability of Network Tomography in LAN and WAN Environments.** 2004. 99p. Tese (Doutorado) – Universidade de Piza. Piza, 2004. Disponível em: <http://etd.adm.unipi.it/theses/available/etd-06242004-163624>. Acesso em: 15 de Novembro de 2005.

OVERSTREET, J.W.; TZES, A. An Internet-Based Real-Time Control Engineering Laboratory. **IEEE Control Systems Magazine**, v.19, n.5, p. 19-34, Outubro de 1999.

P2S TECNOLOGIA. **Site da empresa P2S Tecnologia.** Disponível em: <http://www.p2s.com.br>. Acesso em: 18 de Abril de 2005.

PAPAZOGLOU, M.P.; GEORGAKOPOULOS, D. Service-Oriented Computing. **Communications of the ACM**, v. 46, n. 10, p. 25-28, Outubro de 2003.

PAXTON, V.; FLOYD, S. Why we don't know how to simulate the Internet. **Proceedings of the 1997 Winter Communication Conference**, p. 1037-1044, Dezembro de 1997.

PDS – PACKET DATA SYSTEMS LTD. **Site do produto NetDisturb da empresa Packet Data Systems Ltd.** Disponível em: <http://www.pds-test.co.uk/products/netdisturb.html>. Acesso em: 8 de Dezembro de 2005.

PRESSBOX. IEC RELEASES ICELAN 2000 VERSION 5.0 LONWORKS NETWORK MANAGEMENT TOOL. **Artigo sobre o lançamento da ferramenta ICELAN 2000 v.5.0.** Disponível em: <http://www.pressbox.co.uk/Detailed/6302.html>. Acesso em: 15 de Abril de 2006.

RAJI, R.S. Control Networks and the Internet. In: MAHALIK, N.P. (Ed.) **Fieldbus Technology: The Digital Control Networking System for Automation and Control Applications**. German: Springer-Verlag GmbH & Co., 2002. p. 171-182. ISBN 3-540-40183-0

SNOONIAN, D. Smart Buildings. **IEEE Spectrum**, v. 40, n. 8, p. 18-23, Agosto de 2003.

SOUCEK, S.; SAUTER, T. Quality of Service Concerns in IP-Based Control Systems. **IEEE Transactions on Industrial Electronics**, v. 51, n. 6, p. 1249-1258, Dezembro de 2004.

SOUCEK, S.; SAUTER, T.; KOLLER, G. Effect of Delay Jitter on Quality of Control in EIA-852-based Networks. **Industrial Electronics Society, 2003. IECON'03. The 29th Annual Conference of the IEEE**, v. 2, p. 1431-1436, Novembro de 2003.

_____. Impact of QoS Parameters on Internet-Based EIA 709.1 Control Applications. **Industrial Electronics Society, 2002. IECON'02. The 28th Annual Conference of the IEEE**, v. 4, p. 3176-3181, Novembro de 2002.

SOUCEK, S.; SAUTER, T.; RAUSCHER, T. A Scheme to Determine QoS Requirements for Control Network Data over IP. **Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE**, v. 1, p. 153-158, 2001.

TAC VISTA PRESENTATION SYSTEM. **Introdução sobre o sistema supervisório TAC VISTA para redes LonWorks.** Disponível em: <http://www2.tac.com/Navigate?node=1441>. Acesso em: 03 de Abril de 2005.

TANENBAUM, A.S. **Computer Networks.** 3.ed. Upper Saddle River: Prentice Hall, 1996. 813p.

TOSHIBA. Neuron Chip Data Book. Toshiba, EUA, 1995.

TSE, W.L.; CHAN, W.L.; LAI, S.S. Emergency lighting monitoring system using LonWorks. **Automation in Construction**, n.12, p.617-629, 2003.

VIEIRA, S. Conai 2002 revela área de serviços das empresas, automação predial e integração com dispositivos Web. **Revista Controle & Instrumentação**, São Paulo, n.71, Julho de 2002.

TSANG, Y.; COATES, M.; NOWAK, R.D. Network Delay Tomography. **IEEE Transactions on Signal Processing**, v. 51, n. 8, p. 2125-2136, Agosto de 2003.

Outras referências utilizadas:

AIRIKKA, P. The PID Controller: Algorithm and Implementation. **IEEE Computing and Control Engineering Journal**, v.14, n.6, p. 6-11, Dezembro de 2003 / Janeiro de 2004.

ANANTHAMURTHY, L.; TIWARI, P.; CHITNIS, M. Introduction to Web Services – Part 2: Architecture. **Developer.com.** 2004 (a). Disponível em: <http://www.developer.com/services/article.php/1495021> Acesso em: 27 de Novembro de 2004.

_____. Introduction to Web Services - Part 3: Understanding XML. **Developer.com.** 2004 (b). Disponível em: <http://www.developer.com/services/article.php/1495021> Acesso em: 27 de Novembro de 2004.

BOLZANI, C.A.M. **Residências Inteligentes.** 1.ed. São Paulo: Editora Livraria da Física, 2004. 332p.

CHERVET, A. **Considerations for Using XML Web Services for Device-to-device Communication**. 2004. Disponível em: <http://www.echelon.com/solutions/opensystems/papers/XML-Considerations.pdf>. Acesso em: 20 de Fevereiro de 2005.

CONCEITO TECNOLOGIA. **Site da empresa Conceito Tecnologia.** Disponível em: <http://www.conceitotecnologia.com.br>. Acesso em: 20 de Fevereiro de 2005.

CORDER, R. Web-Enabled Communication Applications for Remote Access and Equipment Monitoring. **Sensorsmag**. Peterborough, NH, EUA. Abril de 2004. Disponível em: <http://www.sensorsmag.com/articles/0404/14>. Acesso em: 15 de Março de 2005.

DUNBAR, M. Plug-and-Play Sensors in Wireless Networks. **IEEE Instrumentation & Measurement Magazine**, p.19-23, Março de 2001.

ECHELON CORPORATION. LonWorks Host Application Programmer's Guide – 1993 (078-0016-01B). Echelon, EUA, 1993.

GAW, D. **Advances in Internetworking Technology and the Future of Control Systems.** Disponível em: <http://www.coactive.com/media/wp9811adv.pdf>. Acesso em: 03 de Outubro de 2005.

_____. **Connecting LonWorks and TCP/IP Enterprise Networks – Real Application Successes.** 1997 (a) Disponível em: <http://www.coactive.com/pages/wp9710.html>. Acesso em: 03 de Outubro de 2005.

_____. **LonWorks over the Internet: Technical Issues and Applications.** 1997 (b) Disponível em: <http://www.coactive.com/media/wp9705.pdf>. Acesso em: 03 de Outubro de 2005.

GERSHENFELD, N.; KRIKORIAN, R.; COHEN, D. The Internet of Things. **Scientific American**, p. 76-81, Outubro de 2004.

HILL, J.L. Mica: A Wireless Platform for Deeply Embedded Networks. **IEEE Micro**, p.12-24, Novembro-Dezembro de 2002.

HIRATA, A.; SILVA, D.F.. **Sistema de Controle e Supervisão de Redes LonWorks via Internet.** 2003. 69p. Projeto de Formatura – Escola Politécnica, Universidade de São Paulo. São Paulo, 2003.

INTILLE, S.S. Designing a Home of the Future. **IEEE Pervasive Computing**, v.1, n.2, p. 76-82, Abril-Junho de 2002.

KINGERY, P. Digital X10. **Home Toys**. Fevereiro de 1999. Disponível em <http://www.hometoys.com/htinews/feb99/articles/kingery/kingery13.htm>. Acesso em: 19 de Abril de 2005.

LAA - LABORATÓRIO DE AUTOMAÇÃO AGRÍCOLA DA ESCOLA POLITÉCNICA DA USP. **Site do LAA.** Disponível em: <http://laa.pcs.usp.br>. Acesso em: 20 de Fevereiro de 2005.

LEE, K.B.; SCHNEEMAN, R.D. Internet-Based Distributed Measurement and Control Applications. **IEEE Instrumentation & Measurement Magazine**, p.23-27, Junho de 1999.

PUTNAM, F.A. Internet-Based Data Acquisition and Control. **Sensors**, v.16, n.10, Novembro de 1999. Disponível em: <http://www.sensorsmag.com/articles/1199/60_1199/main.shtml>. Acesso em: 20 de Fevereiro de 2005.

SIEMENS. TALON™ OPEN SYSTEMS TECHNOLOGY. **Descreve o sistema supervisório TALON para redes LonWorks.** Disponível em: <http://www.sbt.siemens.com/hvp/staefa/bacs/talon.asp>. Acesso em: 21 de Fevereiro de 2005.

TRIVEDI, R. Web Services Tutorial: Understanding XML and XML Schema – Part 1. **Developer.com.** 2004. Disponível em: <http://www.developer.com/services/article.php/2195981> Acesso em: 27 de Novembro de 2004.

WEAVER, A. C. Factory Monitoring and Control Using the Internet. In: IECON'01: THE 27th ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY, 27, 2001. **Proceedings**. p. 1639-1645.

APÊNDICE A – Detalhes sobre protocolos utilizados na Internet

Este apêndice tem como objetivo fornecer uma breve descrição sobre os protocolos nos quais a Internet se baseia, entre eles o IP, TCP, UDP e, de maneira geral, os protocolos de aplicação. Serve como um guia de consulta rápida para o leitor que necessitar dessas informações.

Este apêndice foi baseado principalmente na referência Comer (2000).

A.1 ARQUITETURA DA INTERNET

TCP e IP são protocolos de comunicação que implementam o modelo conceitual sobre o qual a Internet foi construída. Verifica-se aqui a estrutura desse modelo e como os protocolos mencionados o implementam.

Apesar da existência de outras tecnologias e modelos, a arquitetura TCP/IP é atualmente o grande padrão *de facto* para as redes de computadores, independentemente da tecnologia de rede física utilizada e de sua abrangência geográfica (LANs, MANs e WANs).

Conceitualmente, o modelo utilizado pela arquitetura TCP/IP é baseado em três conjuntos de serviços interdependentes. São interdependentes no sentido de que um conjunto de serviços utiliza serviços fornecidos por outro conjunto, mas cada um pode ser abstraído e estudado isoladamente. A Figura A.1 ilustra esses três conjuntos de serviços e sugere sua interdependência. Cada camada necessita dos serviços da camada imediatamente inferior para realizar suas tarefas.

Figura A.1. Camadas conceituais do modelo TCP/IP.

De acordo com Comer (2000), uma das vantagens mais significativas desta separação conceitual em camadas é a possibilidade de substituir a implementação de um serviço por

outra, sem perturbar as outras camadas. Isso porque a interdependência está nos serviços oferecidos, e não em suas implementações. Dessa maneira, é possível realizar estudos, pesquisas, e desenvolvimento para as três camadas de maneira paralela e independente.

Apesar dessa possibilidade de substituição, observa-se que o mundo convergiu quase que em sua totalidade para a utilização dos protocolos da arquitetura TCP/IP.

A.2 O PROTOCOLO IP

A.2.1 **Introdução**

A camada mais inferior da Figura A.1, "Serviço de Entrega de Pacotes Não Confiável", é a camada mais fundamental da Internet e consiste em um sistema de entrega de pacotes não confiável, não orientada a conexão, e de melhor esforço (*best-effort*). Esta camada é a mais próxima da tecnologia física da rede, utilizando-se de seus serviços de comunicação. Se estivesse representada na Figura A.1, a rede física estaria representada como uma camada abaixo desta.

O serviço de entrega de pacotes que esta camada provê tem por objetivo funcionar sobre uma infra-estrutura de redes interconectadas por roteadores conforme explicado no Capítulo 3 (Figuras 5 e 6).

As características mencionadas no primeiro parágrafo têm os seguintes significados:

- não confiável: a entrega de pacotes não é garantida. Cada pacote pode ser perdido, duplicado, atrasado ou entregue fora da ordem correta;
- não orientado a conexão: cada pacote é tratado de maneira independente pela rede. Uma seqüência de pacotes enviada de um computador para outro pode trafegar por caminhos diferentes, perdendo-se alguns e entregando outros corretamente;
- melhor esforço: Os softwares que implementam esse serviço na Internet fazem a tentativa de entrega de pacotes o mais rápido possível. Pacotes não são descartados arbitrariamente, mas sim apenas quanto os recursos estão esgotados ou falhas na rede ocorrem. É daí que surge a não confiabilidade.

O protocolo IP (*Internet Protocol*) consiste em uma implementação desta camada de serviço de entrega não confiável de pacotes. É tão fundamental que as redes construídas com

a arquitetura TCP/IP são comumente chamadas apenas de redes IP ou de redes baseadas na tecnologia IP (*IP-based technology*).

O protocolo IP apresenta três atribuições essenciais:

1. definição de uma unidade básica de transferência de dados;
2. os softwares que o implementam realizam a função de roteamento dos pacotes, isto é, das unidades básicas de transferência de dados;
3. o protocolo IP define um conjunto de regras que caracterizam a entrega não confiável de pacotes.

Do ponto de vista do usuário de rede, isto é, do utilizador dos serviços, há uma forte analogia entre uma rede TCP/IP e uma rede física (Ethernet, FDDI, etc). Em uma tecnologia física de rede, a unidade fundamental de transferência de dados é o quadro (*frame*). Um quadro é dividido em duas partes: cabeçalho e dados. O cabeçalho é a parte que carrega, entre outras informações, o endereço físico de origem e de destino. A área de dados carrega as informações propriamente ditas que se deseja transmitir. Nas redes TCP/IP, este papel é o do pacote IP, ou seja, da unidade básica de transferência de dados definida pelo protocolo IP. Os pacotes IP também são conhecidos como datagramas IP (*IP datagram*) ou simplesmente datagramas. Assim como um quadro, um datagrama também é dividido em duas partes: cabeçalho e dados. O cabeçalho de um datagrama IP carrega uma série de informações úteis para identificação e roteamento, entre elas os endereços IP de origem e de destino do pacote.

A.2.2 **O datagrama IP**

O formato do datagrama IP pode ser conferido na Figura A.2. Cada linha representa um conjunto de 32 bits, ou seja, 4 bytes. As seis primeiras linhas compõem o cabeçalho do datagrama, enquanto a informação carregada (*payload*) vem a partir da sétima linha.

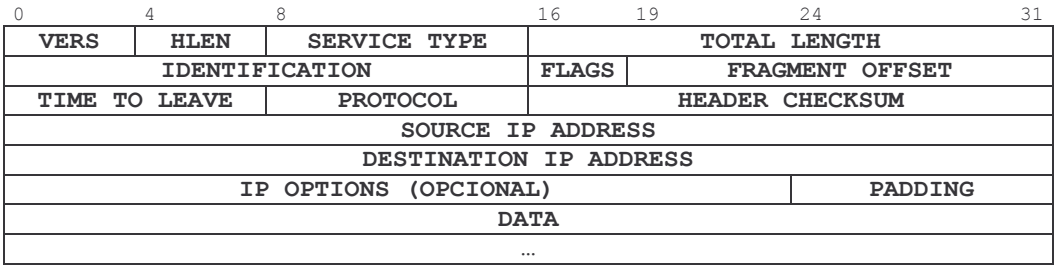


Figura A.2. Formato do datagrama IP.

Cada campo do datagrama IP possui um significado específico, os quais podem ser conferidos brevemente a seguir:

- VERS: versão do protocolo IP utilizada para criar o datagrama. Atualmente, o IP está em sua versão 4, também conhecida como *IPv4*. O *IPv6* é bastante comentado atualmente, e deve ser o substituto do *IPv4*;
- HLEN: contém o tamanho do cabeçalho IP medido em palavras de 32 bits. Todos os campos do cabeçalho IP têm tamanho fixo, exceto IP OPTIONS e PADDING. Se o cabeçalho não contiver valores para esses campos, terá 20 bytes e, portanto, o campo HLEN conterá o valor 5;
- SERVICE TYPE: também chamado de TOS (*Type of Service*), este campo especifica como o datagrama deve ser tratado pelos roteadores pelos quais passar ao longo do caminho entre origem e destino final. Será detalhado mais adiante;
- TOTAL LENGTH: tamanho do datagrama IP medido em bytes, incluindo cabeçalho e dados. Dessa forma, o tamanho da área de dados, também em bytes, pode ser computado pela fórmula TOTAL LENGTH – (HLEN * 4). Pelo fato de este campo possuir 16 bits, o tamanho máximo teórico de um datagrama IP é 2^{16} ou 65.535 bytes;
- IDENTIFICATION: identificador do datagrama utilizado para recomposição de datagramas fragmentados. O conceito de fragmentação será visto mais adiante;
- FLAGS: bits de controle de fragmentação. Será visto mais adiante;
- FRAGMENT OFFSET: em caso de datagramas fragmentados, indica em qual posição o fragmento deve se encaixar no processo de reconstituição do datagrama. Será visto mais adiante;
- TIME TO LEAVE (TTL): especifica a duração do datagrama IP na rede. Tabelas de roteamento inconsistentes em roteadores rede podem fazer com que datagramas tomem caminhos circulares e nunca cheguem ao seu destino final. Para garantir que um datagrama não tenha vida eterna trafegando pela rede, este campo é um contador que é decrementado por cada roteador pelo qual o datagrama passa. Quando este campo atinge o valor zero, o datagrama é descartado e a origem é avisada com uma mensagem de erro. A própria origem é a responsável por especificar um TTL inicial;
- PROTOCOL: especifica o formato do conteúdo da área de dados do datagrama. A cada protocolo de nível mais alto utilizado sobre o IP é atribuído um inteiro

identificador. Esses identificadores são administrados por uma autoridade central na Internet;

- HEADER CHECKSUM: garante a integridade do cabeçalho IP. Este *checksum* é calculado interpretando-se o cabeçalho como uma seqüência de inteiros de 16 bits, somando-os, e tomando o complemento de um do resultado. O próprio campo HEADER CHECKSUM não entra no cálculo do *checksum*, isto é, supõe-se que ele contém o valor zero durante a soma;
- SOURCE IP ADDRESS: contém o endereço IP do computador de origem, isto é, aquele que gerou o datagrama;
- DESTINATION IP ADDRESS: contém o endereço IP do computador de destino;
- IP OPTIONS: este campo é opcional e possui tamanho variável. Contém opções que podem ser especificadas pelo computador de origem. O protocolo IP suporta várias opções. As opções aparecem continuamente no campo IP OPTIONS, sem nenhum separador. Cada uma delas consiste em um byte identificador da opção e um número variável de bytes contendo parâmetros específicos, que pode inclusive ser zero. O byte de código da opção é dividido em três sub-campos: COPY (1 bit), OPTION CLASS (2 bits) e OPTION NUMBER (5 bits). O campo COPY controla como os roteadores tratam as opções na fragmentação, que será vista mais adiante. OPTION CLASS e OPTION NUMBER especificam a opção propriamente dita. OPTION CLASS representa uma classe de opções e OPTION NUMBER é a opção desejada dentro da classe. A lista das opções definidas pelo protocolo IP, seus significados e parâmetros podem ser encontrados na literatura especializada, tais como Comer (2000);
- PADDING: o tamanho do cabeçalho do protocolo IP deve ser um múltiplo de 4 bytes (32 bits). Como o campo IP OPTIONS tem tamanho variável que não é necessariamente múltiplo de 4 bytes, o campo PADDING serve para adicionar bits nulos após o IP OPTIONS até que o tamanho total do cabeçalho complete um valor válido;
- DATA: corresponde à área de dados do datagrama IP, cujo tamanho é variável e depende daquilo que se deseja enviar.

A.2.3 **Type of Service**

Originalmente, o campo SERVICE TYPE era dividido em cinco sub-campos conforme a Figura A.3.

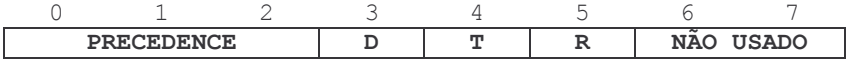


Figura A.3. Campo SERVICE TYPE do datagrama IP.

Os significados desses campos são:

- PRECEDENCE: especifica uma prioridade variando de 0 a 7. Quando maior o número, maior a prioridade atribuída ao datagrama pelos roteadores. A prioridade 0 também é chamada de precedência normal (*normal precedence*), enquanto a prioridade 7 é conhecida como controle de rede (*network control*). Essa nomenclatura torna mais claro o fato de que informações de controle da rede podem circular com prioridade maior, tomando a frente do tráfego de dados mesmo quando a rede está congestionada. Apesar disso, na prática, muitos roteadores ignoram este campo;
- D / T / R : especificam o tipo de transporte desejado para o datagrama. D significa baixo delay, T corresponde a alta taxa de transmissão (*throughput*) e R requisita alta confiabilidade.

Dependendo da situação corrente da rede, pode não ser possível garantir as condições solicitadas pelo campo SERVICE TYPE. Dessa maneira, ele acaba servindo apenas como uma orientação aos roteadores, os quais podem tomar decisões de roteamento de maneira mais adequada ao valor especificado para SERVICE TYPE com base em seu conhecimento sobre as redes às quais está conectado. No entanto, não são fornecidas garantias de que os datagramas serão entregues sob as condições solicitadas.

No final da década de 1990, a definição do campo SERVICE TYPE foi alterada para acomodar um conjunto de serviços diferenciados, ou DS (*differentiated services*), o que alterou sua interpretação para os campos indicados pela Figura A.4.

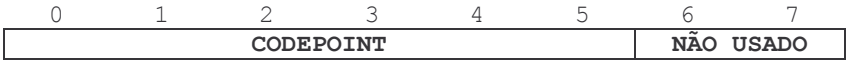


Figura A.4. Campo SERVICE TYPE do datagrama IP após redefinição.

- CODEPOINT: contém um inteiro correspondente a uma definição padronizada de serviço. Por ter 6 bits, é possível definir 2^6 ou 64 serviços. No entanto, os desenvolvedores recomendam que cada roteador implemente apenas alguns serviços, e que a cada um deles sejam mapeados diversos valores de CODEPOINT.

Para manter compatibilidade com a interpretação original do campo SERVICE TYPE, os últimos três bits de CODEPOINT são verificados. Se eles possuírem valor zero, os bits anteriormente correspondentes ao campo PRECEDENCE são mapeados em oito classes de serviço que obedecem às mesmas diretrizes da definição original: quanto maior o número, maior a prioridade atribuída pelo roteador ao datagrama.

A.2.4 **Fragmentação**

O conceito de fragmentação está ligado ao encapsulamento dos datagramas IP nos quadros da tecnologia física de rede utilizada. Quando um datagrama é enviado pela rede, ele é colocado dentro da área de dados do quadro definido pela rede física. A Figura A.5 fornece uma ilustração.

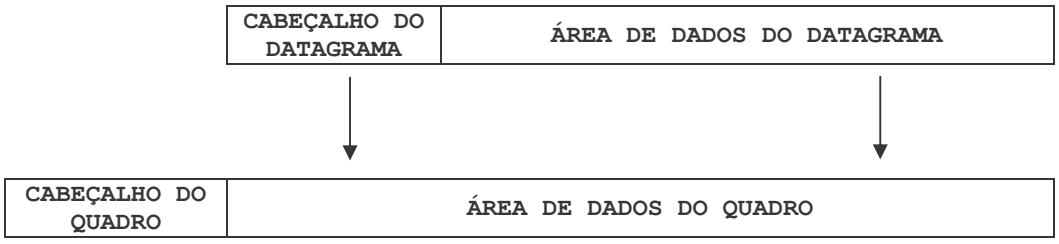


Figura A.5. Encapsulamento de datagramas IP em quadros.
Baseado em Comer (2000).

O ideal, para tornar a comunicação simples e eficiente, é encapsular cada datagrama em um único quadro. Quando se conhece o tamanho do quadro definido pela tecnologia da

rede física utilizada, pode-se limitar o tamanho dos datagramas ao tamanho da área de dados do quadro, permitindo que o primeiro "caiba" no segundo. Por outro lado, quando se está lidando com uma rede formada pela interconexão de outras redes através de roteadores, como é o caso da Internet, nem sempre se pode saber de antemão qual o tamanho máximo do quadro de cada rede física pelo qual o datagrama passará até chegar em seu destino final. A situação é clara quando os roteadores estabelecem caminhos dinâmicos para os datagramas. Sendo assim, surge o seguinte problema: o que um roteador deve fazer com um datagramas maior que a área de dados do quadro da rede para o qual ele será encaminhado? A resposta para este problema chama-se fragmentação (*fragmentation*).

A fragmentação consiste em quebrar o datagrama em partes menores de modo que cada parte caiba individualmente no quadro em questão. Cada parte é denominada fragmento (*fragment*) e é enviada individualmente como se fosse um datagrama isolado. Uma vez fragmentado, o datagrama completo nunca é reconstituído na própria rede, mas sim apenas em seu destino final, que também recebe os fragmentos como se fossem datagramas individuais.

Para que este conceito possa ser aplicado com sucesso, é necessário adicionar um novo cabeçalho a cada fragmento, o qual consiste no cabeçalho do datagrama original com algumas modificações. É neste ponto que entram os campos `IDENTIFICATION`, `FLAGS` e `FRAGMENT OFFSET` indicados na seção I.2.2.

Quando uma seqüência de datagramas enviados de uma certa origem a um certo destino passam por um caminho que causa fragmentação, o computador de destino necessita reconstituir todos os datagramas através da concatenação dos fragmentos. Porém, a rede trata cada fragmento como um datagrama individual e, portanto, eles podem ser perdidos, chegar fora de ordem, ou de maneira duplicada. O primeiro problema é estabelecer quais fragmentos correspondem a quais datagramas. Para isso, existe o campo `IDENTIFICATION` do cabeçalho IP, que carrega um inteiro identificando o datagrama a qual o fragmento pertence. Cada datagrama é enviado pela origem com um identificador único, o qual é replicado no cabeçalho de cada fragmento durante o processo de fragmentação. Assim, o destino final tem uma maneira de saber exatamente quais fragmentos correspondem a quais datagramas originais.

Fragmentos podem ser entregues fora de ordem e, para isso, o destino final também deve contar com alguma maneira para saber qual a ordem em que eles devem ser recompostos para recuperar o datagrama original. O campo `FRAGMENT OFFSET` tem essa finalidade. Ele contém um número de seqüência correspondente aos bytes que a área de dados do fragmento carrega. Se o primeiro fragmento de um datagrama contiver 100 bytes (0 a 99), então o `FRAGMENT OFFSET` do segundo fragmento será 100.

Finalmente, fragmentos podem ser perdidos individualmente. Dessa forma, um datagrama não pode ser reconstruído se um de seus fragmentos não chegar ao destino, mesmo que os outros sejam entregues corretamente. Nesse caso, os fragmentos recebidos são descartados o datagrama por completo é dado como perdido. Se um fragmento intermediário não chega, é relativamente fácil para o destino final da mensagem identificar sua falta. Isso poderia ser feito pela verificação dos valores de `FRAGMENT OFFSET` e do tamanho de cada fragmento, que indicariam lacunas na recomposição do datagrama. Por outro lado, o fato de o último fragmento de um datagrama se perder pode aparentar ao destino final que todos os fragmentos foram recebidos, já que não há lacunas entre eles. O bit menos significativo do campo `FLAGS` do cabeçalho IP é chamado de bit de mais fragmentos (*more fragments bits*). Quando um roteador fragmenta um datagrama, todos os fragmentos, com exceção do último, devem carregar o valor 1 neste bit, enquanto o último leva o valor 0. Isso informa ao destino final se o último fragmento da seqüência recebida, já colocada em ordem, corresponde realmente ao último fragmento que compõe o datagrama ou se ainda há algum por chegar.

Um outro bit do campo `FLAGS` serve para informar aos roteadores se é permitido fragmentar o datagrama. Este bit é denominado não fragmente (*do not fragment bit*). Datagramas em que esse bit contém o valor zero nunca chegarão ao seu destino final se a fragmentação for realmente necessária ao longo do caminho percorrido. Um roteador, ao receber um datagrama que precisa ser fragmentado mas que não possui o bit de autorização necessário, descarta o mesmo e encaminha uma mensagem de erro à origem. A não permissão de fragmentação é comumente utilizada em situações de testes e depuração da rede.

A.3 O PROTOCOLO TCP

A.3.1 **Introdução**

A camada do meio da Figura A.1, “Serviço de Transporte Confiável”, corresponde ao segundo serviço mais importante desta arquitetura. Sua intenção é prover a entrega confiável de um fluxo de dados (*stream*) de uma origem a um destino final. Para isso, utiliza-se dos serviços prestados pela camada inferior “Serviço de Entrega de Pacotes Não Confiável”.

De acordo com as seções anteriores, o protocolo IP provê a entrega de datagramas fim-a-fim sem que o usuário precise se preocupar com os caminhos que o datagrama tomará na rede e como o roteamento é efetivamente realizado. Porém, essa entrega não é garantida e está sujeita a alguns problemas: perda de datagramas, chegada fora de ordem e duplicação.

Os programas de aplicação que são executados nos computadores das redes freqüentemente desejam transferir grandes quantidades de dados. Fazer isso através de um serviço de entrega de pacotes não confiável torna necessário o desenvolvimento de rotinas para tratar dos problemas que podem ocorrer, o que não é trivial e requer um esforço considerável. Cada desenvolvedor teria então um enorme custo e tempo de desenvolvimento aplicado nisso, que, apesar de complexo, nada mais é do que uma atividade auxiliar para a comunicação, e usualmente não consiste no foco do seu aplicativo ou negócio.

Sendo assim, um dos objetivos da pesquisa de protocolos de rede é encontrar soluções de propósito geral para o problema de prover um serviço de entrega confiável de dados. Ter um protocolo padrão que implementa esse serviço isola os programas de aplicação dos detalhes de controle de erros da rede, além de permitir a definição de uma interface única para a transferência fim-a-fim de fluxos de dados.

Entende-se por serviço de entrega confiável aquele que fornece a garantia de que os dados serão entregues na outra ponta sem perdas e na ordem correta. O tratamento de erros que está por trás desse serviço deve ser transparente ao usuário.

Na arquitetura TCP/IP, a camada que provê esse serviço é implementada pelo protocolo TCP (*Transmission Control Protocol*). Ela utiliza os serviços da camada inferior, "Serviço de Entrega de Pacotes Não Confiável", implementada pelo protocolo IP. A maneira pela qual um serviço confiável é construído sobre um serviço não confiável será vista mais adiante.

Pode-se destacar cinco características do serviço prestado pelo protocolo TCP (COMER, 2000):

1. orientado a fluxo (*stream*): os dados trocados entre dois aplicativos via rede são vistos como fluxos de bits, divididos em grupos de 8 (bytes). O destino final dos dados recebe exatamente a mesma seqüência de bytes enviadas pela origem;
2. conexão baseada em circuito virtual: para que uma transferência de um fluxo de bytes seja realizada, o TCP requer que seja feita uma conexão entre os dois aplicativos que participarão do processo. Essa conexão é análoga a uma chamada telefônica: um dos aplicativos solicita uma conexão ao outro, e o segundo deve aceitar o pedido. Uma vez que a conexão é estabelecida, abre-se o caminho para que os dados trafeguem entre os dois aplicativos. No entanto, deve ficar claro que não se trata de uma conexão física baseada em comutação de circuitos como no caso da rede telefônica, afinal, as redes de computadores são baseadas em comutação de pacotes. Essa conexão é o modo como o usuário enxerga a

comunicação e é implementada via software. Na prática, ela é implementada através de pacotes que circulam na rede. Por isso, atribui-se o termo "circuito virtual". Os processos de conexão e desconexão também são tratados pelo protocolo TCP;

3. *buffers* de transferência: quando um aplicativo deseja transmitir a outro uma seqüência de bytes, ele a passa para o software que implementa o protocolo TCP na máquina. O tamanho dessa seqüência é arbitrário e pode variar de um único byte a vários megabytes. No entanto, visando uma maior eficiência na comunicação, o software do TCP é livre para dividir o fluxo de dados em pacotes com tamanhos independentes dos fornecidos pela aplicação. Se a seqüência fornecida pela aplicação for muito pequena, o software do TCP pode aguardar que o aplicativo envie mais bytes até que o *buffer* atinja um tamanho suficiente. O TCP também provê um serviço que permite ao aplicativo forçar o envio dos dados, mesmo que o *buffer* de transmissão ainda esteja inferior ao desejado;
4. *stream* não estruturado: o fluxo de dados que um aplicativo passa para o TCP transmitir não tem nenhum formato. A seqüência é totalmente livre e não estruturada, sendo entregue à outra ponta na mesma ordem em que foi enviada. Como conseqüência, os programas de aplicação devem concordar sobre o entendimento das seqüências a serem transmitidas antes do estabelecimento de uma conexão;
5. *full duplex*: as conexões oferecidas pelo protocolo TCP permitem o envio de dados nas duas direções, isto é, uma máquina transmissora pode ao mesmo tempo ser uma máquina receptora. As seqüências de dados enviadas nos dois sentidos podem ser completamente independentes.

O serviço de entrega de dados confiável é construído sobre o serviço não confiável de entrega de datagramas através de uma técnica conhecida como confirmação positiva com retransmissão (*positive acknowledgement with retransmission*). A seqüência de dados que um aplicativo deseja transmitir é quebrada em diversas partes, e cada uma delas é encapsulada em um datagrama IP. Os datagramas são então enviados da origem ao destino, na ordem, mas esse envio deve obedecer algumas regras. A técnica estabelece que o destino final de um datagrama deve enviar de volta à origem outro datagrama contendo uma confirmação de recebimento (*acknowledgement*, ou, abreviadamente, ACK). A origem mantém uma cópia do datagrama enviado e inicializa um temporizador logo após o envio do mesmo. Se a confirmação de recebimento chegar antes do temporizador expirar, a cópia pode ser

descartada e o datagrama é dado como entregue. O temporizador é então cancelado. Se ele expirar antes do ACK chegar, o pacote é considerado como perdido na rede e, portanto, é retransmitido. Nesse caso, um novo temporizador é inicializado e o processo se repete.

O processo mais simples de confirmação positiva com retransmissão é aquele em que apenas um pacote é enviado por vez. O próximo só pode ser enviado quando a confirmação do anterior chegar. A Figura A.6 ilustra a técnica descrita. O caso (a) é aquele em que o pacote IP chega corretamente ao seu destino e é confirmado. No caso (b), supõe-se que o datagrama foi perdido na rede. O eixo vertical representa o tempo, que é crescente de cima para baixo. A faixa intermediária entre origem e destino, onde estão representadas setas, representa a rede. Para o TCP, é indiferente a maneira pela qual os datagramas são roteados ao seu destino. As setas representam os datagramas IP trafegando da origem ao destino e vice-versa.

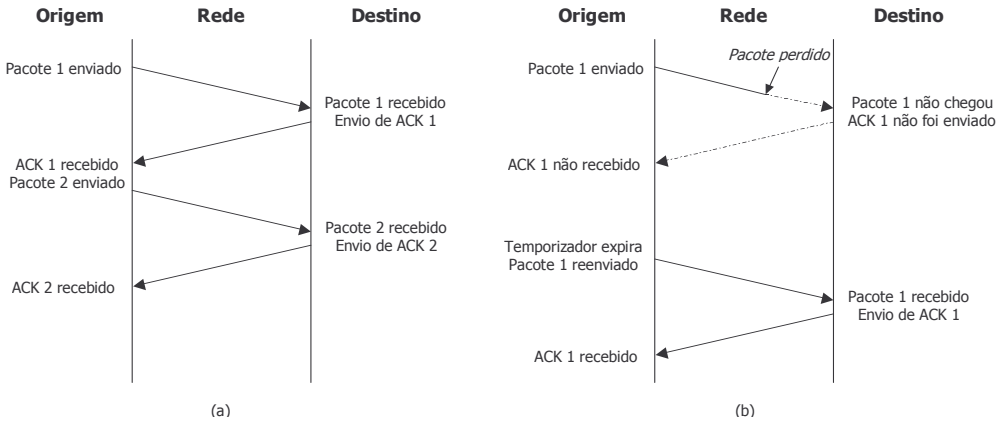


Figura A.6. Confirmação positiva com retransmissão.
(a) Sem perda de pacote. (b) Com perda de pacote e retransmissão.
Baseado em Comer (2000).

Pode-se observar que esta implementação da técnica de confirmação positiva com retransmissão garante que os datagramas chegarão ao seu destino final. Se houver perda, ele é reenviado. Um número excessivo de retransmissões sem sucesso pode indicar que a rede está com problemas, permitindo à origem o cancelamento da conexão. Além disso, os datagramas são entregues na ordem correta pois a transmissão do próximo sempre aguarda a confirmação de recebimento do anterior.

Alguns detalhes a mais devem ser considerados, mas são facilmente resolvidos através de implementação de software adequada. Se a rede está lenta, um pacote corretamente

recebido pode ser dado como perdido se a confirmação de recebimento demorar para retornar à origem. Nesse caso, o datagrama será retransmitido sem necessidade, e o destino final irá recebe-lo duas vezes. O software que implementa o TCP deve ser inteligente o suficiente para detectar essa duplicidade e saber que se trata de uma retransmissão, e não de um novo datagrama. Para isso, um número de seqüência que identifica o datagrama está presente no cabeçalho TCP. Esse número permite identificar a posição do datagrama dentro do *stream* de dados que está sendo transmitido e, conseqüentemente, se o datagrama recebido é realmente um novo ou se é apenas uma retransmissão.

O leitor deve observar que a situação de duplicidade também pode ocorrer quando o datagrama chega corretamente ao destino mas o ACK é perdido na rede.

A.3.2 **Janelas deslizantes**

Mostrou-se que a técnica de confirmação positiva com retransmissão permite criar um serviço de entrega confiável de *stream* de dados sobre um serviço não confiável de entrega de pacotes. Por outro lado, observa-se que surge uma ineficiência: o transmissor fica em espera aguardando por um ACK enquanto a rede pode estar ociosa. A espera por um ACK é necessária para garantir a confiabilidade do protocolo, mas deve-se verificar se existe outra solução em que é possível enviar mais datagramas enquanto se aguarda por um ACK. Assim, o tempo de espera não é desperdiçado e a rede continua sendo utilizada. Nesse contexto, entra o conceito das janelas deslizantes (*sliding windows*).

O conceito de janelas deslizantes é um pouco mais complicado que o exemplo fornecido para a transmissão de um pacote por vez. A melhor maneira de entender esse conceito, que deu origem ao seu próprio nome, é visualizando uma seqüência de pacotes a serem transmitidos limitados por uma janela. Toma-se a situação em que o software do TCP quebrou o *stream* de dados que o programa de aplicação deseja transmitir em vários datagramas IP. Cada datagrama deve agora ser transmitido, mas foi visto que é ineficiente aguardar um ACK por vez. Aplicando-se uma abstração, considera-se que o protocolo posiciona uma "janela" com um tamanho pré-determinado sobre essa seqüência, indicando quais pacotes podem ser transmitidos sem aguardar pelo ACK dos pacotes anteriores. A Figura A.7 provê uma ilustração supondo uma janela de tamanho igual a três.

O protocolo transmite todos os datagramas presentes dentro da janela, independentemente do recebimento de ACKs. Quando os ACKs chegam, a janela é deslizada, incluindo novos datagramas a serem transmitidos.

No exemplo da Figura A.7, os datagramas 1, 2 e 3 são enviados. Quando o ACK do pacote 1 chega, a janela é deslizada, englobando também o pacote 4 (Figura A.7 b), que já pode ser enviado. Enquanto o pacote 4 é enviado, o ACK do pacote 2 pode estar chegando, e assim por diante.

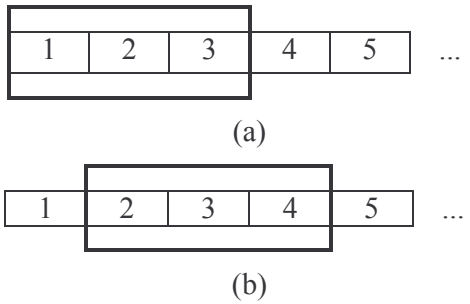


Figura A.7. Janela deslizante de tamanho três.
(a) Antes de receber o primeiro ACK. (b) Deslocamento após receber primeiro ACK.
Baseado em Comer (2000).

Um temporizador individual é disparado para cada datagrama após seu envio. Se ele expira, apenas o datagrama em questão é reenviado, ou seja, mantém-se a individualidade na retransmissão. A janela é deslizada quando os ACKs correspondentes aos pacotes mais anteriores são recebidos.

Mesmo que intuitivamente, observa-se que a eficiência foi aumentada. A Figura A.8 oferece uma imagem da dinâmica deste processo.

A.3.3 **Portas**

O TCP é orientado a conexão e foi construído com base no conceito de circuitos virtuais. Um circuito virtual, do ponto de vista lógico, é um caminho de comunicação entre dois programas de aplicação. É, portanto, natural dizer que os pontos finais (*endpoints*) de uma conexão TCP são os aplicativos que estão conectados. Conceitualmente, isso não deixa de estar correto, mas a definição formal e precisa de ponto final é um par ordenado (*ip*, *porta*), onde *ip* corresponde ao endereço IP da máquina em questão e *porta* é um inteiro de 16 bits sem sinal que permite que uma mesma máquina participe simultaneamente de diversas conexões. Se o conceito de porta não existisse, não haveria como separar os datagramas que chegam da rede em grupos correspondentes às conexões TCP das quais a máquina participa. Apenas uma única conexão por vez seria possível.

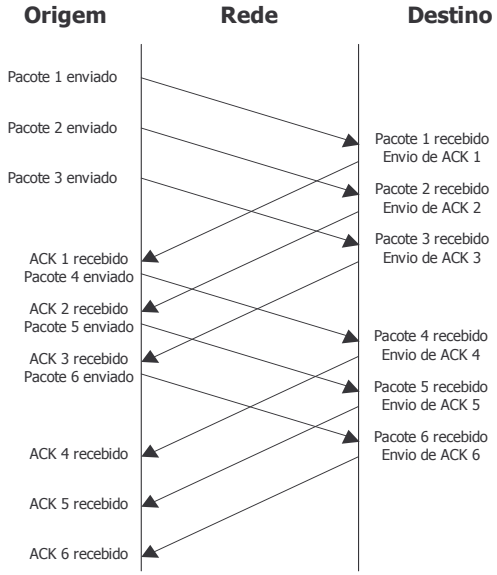


Figura A.8. Transmissões e ACKs com janela igual a três.
Baseado em Comer (2000).

Uma conexão TCP pode ser identificada por dois pontos finais, e, portanto, pode ser representada por um quarteto ordenado (*ip_host_1*, *porta_host_1*, *ip_host_2*, *porta_host_2*).

No estabelecimento de uma conexão TCP, o requisitante deve especificar a porta em que deseja se conectar no computador de destino (além de, obviamente, próprio endereço IP do mesmo). Existe um mapeamento padronizado que relaciona uma faixa de portas a aplicações comuns da Internet, permitindo que o programa de aplicação que requisita a conexão consiga contatar corretamente o aplicativo desejado em um determinado servidor. Por exemplo, a porta 80 é padronizada para a *World Wide Web*. Quando um usuário digita um endereço em seu *web browser*, este solicita uma conexão na porta 80 na máquina que possui o endereço digitado. Após o estabelecimento da conexão, uma *home page* é solicitada através do protocolo HTTP (*Hyper Text Transfer Protocol*), que se encaixa na camada "Serviços de Aplicação" da Figura A.1 e utiliza o serviço de entrega confiável de dados do TCP. Isso será melhor detalhado na seção I.5.

A.3.4 **O segmento TCP**

Foi visto que o fluxo de dados que um programa de aplicação entrega ao protocolo TCP é dividido em partes, e cada parte é encaixada em um datagrama para ser enviado através do protocolo IP. Verificou-se também que algumas informações de controle precisam constar

nesses datagramas, entre elas o número de seqüência que identifica a posição do conteúdo do datagrama em relação a todo o *stream*. Uma outra informação de controle é a própria confirmação de recebimento que o destino final devolve à origem, que também deve incluir o número de seqüência do datagrama que está sendo confirmado.

Por outro lado, o formato do datagrama IP já foi estudado, e, pôde-se observar que não existem campos em seu cabeçalho que contemplam essas necessidades. É justamente aí que entra o protocolo TCP. Ele define um formato de pacote conhecido como segmento TCP, ou simplesmente segmento, que inclui um cabeçalho próprio onde são colocadas todas as informações de controle necessárias para o funcionamento do TCP.

O segmento é o *Protocol Data Unit* (PDU) da camada TCP. PDU é um nome genérico para designar o bloco de dados gerado por uma certa camada de uma pilha de protocolos. O datagrama IP é o PDU da camada IP e o quadro Ethernet, por exemplo, é o PDU da camada física das redes Ethernet.

As partes em que o *stream* de dados foi divido são, portanto, colocadas na área de dados de segmentos TCP. Cada segmento é, por sua vez, colocado em um datagrama IP, conforme a Figura A.9.

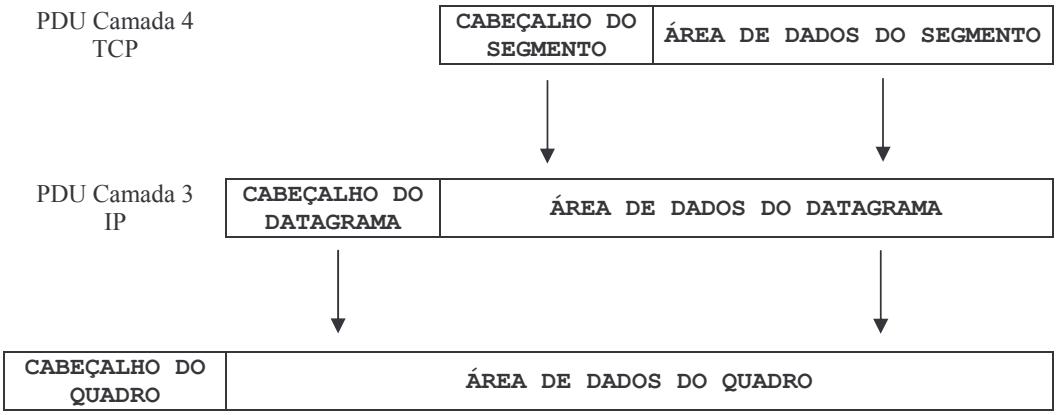


Figura A.9. Encapsulamento entre pacotes de camadas diferentes.

Na Figura A.10, consta o formato do segmento TCP.

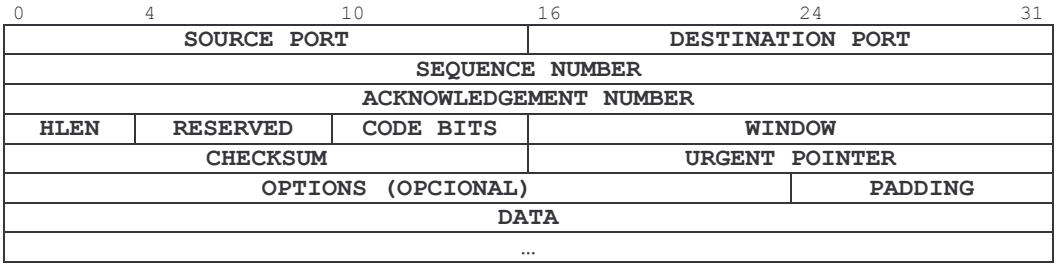


Figura A.10. Formato do segmento TCP.

Os significados de cada campo são:

- SOURCE PORT: identifica a porta da conexão na máquina de origem;
- DESTINATION PORT: identifica a porta da conexão na máquina de destino. Juntamente com SOURCE PORT, SOURCE ADDRESS e DESTINATION ADDRESS (esses dois últimos provenientes do cabeçalho do datagrama IP), forma o quarteto ordenado que identifica a conexão TCP à qual o segmento corresponde;
- SEQUENCE NUMBER: número de seqüência correspondente à posição dos dados que o segmento carrega em relação ao *stream* transmitido. Análogo ao FRAGMENT OFFSET do cabeçalho IP. Por exemplo, se o segmento anterior continha 100 bytes de dados e possuía número de seqüência 3687, então o próximo número de seqüência será 3787. O número de seqüência é incrementado individualmente em cada conexão TCP;
- ACKNOWLEDGEMENT NUMBER: os ACKs são enviados através deste campo. Ele contém o número da posição do último byte recebido com sucesso pelo transmissor. Esse campo refere-se à posição no *stream* recebido, que é independente do *stream* transmitido. Isso permite que os ACKs relativos aos pacotes recebidos sejam enviados juntamente com os dados a serem transmitidos, economizando recursos da rede enquanto se realiza comunicação *full duplex*;
- HLEN: contém um inteiro que especifica o tamanho do cabeçalho TCP medido em múltiplos de 4 bytes (32 bits). Similar ao HLEN do cabeçalho IP;
- RESERVED: reservado para uso futuro;
- CODE BITS: alguns segmentos carregam apenas um ACK, enquanto outros carregam dados. Além disso, eles podem carregar dados e ACK ao mesmo tempo (campo ACKNOWLEDGEMENT NUMBER), bem como uma série de outras informações

presentes e necessárias no protocolo TCP, tais como requisições de conexão. Este campo de seis bits carrega um código que determina o propósito do conteúdo do segmento, dizendo ao software como interpretar os outros campos do cabeçalho. Esses seis bits são ativados com o valor 1 e desativados com 0. Do mais significativo para o menos significativo, são:

- URG: Indica que o campo URGENT POINTER deve ser considerado (ver adiante);
- ACK: Indica que o campo ACKNOWLEDGEMENT NUMBER contém um valor válido, isto é, que o segmento carrega um ACK;
- PSH: Indica que o segmento requisita um *push*, isto é, o aplicativo forçou o envio de dados;
- RST: Indica um *reset* de conexão;
- SYN: Sincroniza os números de seqüência;
- FIN: Indica que o transmissor não tem mais dados para enviar;

Para um entendimento mais detalhado, recomenda-se a consulta de referências especializadas, como Comer (2000);

- WINDOW: toda vez que um segmento é enviado, o software do TCP sinaliza neste campo a quantidade de dados que está preparado para receber. Assim como o ACK, essa sinalização pode ser enviada juntamente com os dados transmitidos, melhorando a utilização de recursos da rede. Na prática, corresponde ao tamanho da janela deslizante, mas é medida em bytes e não em número de pacotes. Apesar de o exemplo mostrado na seção I.3.2 possuir um tamanho de janela fixo e pré-determinado, o protocolo TCP permite que esse tamanho seja ajustado dinamicamente ao longo de cada conexão, seguindo algumas regras específicas para otimizar a eficiência da comunicação. Esse campo serve para esse ajuste dinâmico;
- CHECKSUM: *checksum* do segmento, calculado de maneira similar ao HEADER CHECKSUM do IP. Considera um pseudo-cabeçalho, isto é, uma extensão "virtual" do cabeçalho que não é transmitida, para feito de cálculos. Maiores detalhes em Comer (2000);
- URGENT POINTER: o protocolo TCP permite que o programa de aplicação transmissor sinalize que determinados dados devem ser tratados como urgentes, significando que o aplicativo que os recebe deve ser notificado de sua chegada o

mais rápido possível. Essa indicação de urgência é feita através deste campo em conjunto com o bit URG de CODE BITS. Quando o bit URG está ligado, o URGENT POINTER especifica a posição no segmento em que os dados urgentes terminam;

- OPTIONS: analogamente às opções do cabeçalho IP, o TCP também apresenta este campo opcional de tamanho variável que permite especificar diversas opções. Maiores detalhes podem ser encontrados em literatura mais especializada;
- PADDING: bits preenchidos com zero adicionados ao cabeçalho para que seu tamanho total seja um múltiplo de 32 bits. É análogo ao PADDING do cabeçalho IP;
- DATA: corresponde à área de dados do segmento TCP, cujo tamanho é variável.

O protocolo TCP também define algumas regras importantes que determinam seu comportamento e, conseqüentemente, seu desempenho. Um exemplo é a capacidade de resposta a congestionamento na rede. Quando a rede está sobrecarregada, os computadores enfrentam grandes atrasos e efetuam muitas retransmissões. O excesso de retransmissões simplesmente piora a situação, já que lança mais datagramas à rede. Para evitar um colapso, o TCP reduz a taxa de transmissão quando ocorre congestionamento de acordo com os algoritmos *slow-start* e *multiplicative decrease*. Detalhes sobre esses algoritmos e outras atribuições do TCP podem ser encontrados em Comer (2000).

A.4 O PROTOCOLO UDP

O protocolo TCP implementa um serviço confiável de entrega de fluxos de dados, ou *streams*. Em alguns casos, um determinado programa de aplicação pode desejar implementar seu próprio mecanismo sem aproveitar as funcionalidades fornecidas pelo TCP. Em outros casos, onde alguma perda de informação é tolerável, a entrega de dados de maneira confiável não é tão necessária e pode até atrapalhar a comunicação. Um exemplo para este caso é a transmissão de voz em tempo real, onde se deseja enviar um *stream* de dados com alta taxa (*throughput*) e baixo *delay*. A sobrecarga (*overhead*) introduzida pelo TCP através dos mecanismos de confirmação de recebimento e retransmissão pode se tornar indesejável, ao mesmo tempo em que eventuais perdas de pacotes não causariam grandes transtornos ao ouvinte.

Para esses casos, existe o protocolo UDP (*User Datagram Protocol*). Ele se encaixa em uma camada com o mesmo nível hierárquico da camada “Serviço de Transporte

Confiável" da Figura A.1, pois se utiliza dos serviços da camada inferior ("Serviço de Entrega de Pacotes Não Confiável") e é utilizado pela camada superior ("Serviços de Aplicação").

O UDP é bastante simples e pode ser enxergado de maneira bastante semelhante ao IP. Também é um serviço de entrega de pacotes não confiável e não orientado a conexão. A única funcionalidade adicionada em relação ao IP é a utilização do número de portas, que permite que diversos *streams* simultâneos sejam transmitidos via UDP para uma mesma máquina. A porta permite identificar a qual *stream* o pacote pertence, possibilitando que o software do protocolo UDP entregue os dados ao aplicativo correto. O pacote do protocolo UDP é denominado datagrama UDP (*UDP datagram*) e, assim como o segmento TCP, é encapsulado na área de dados do datagrama IP para ser enviado pela rede.

O datagrama UDP também é dividido em duas partes: cabeçalho e dados, e, como pode ser visto na Figura A.11, é bastante simples.

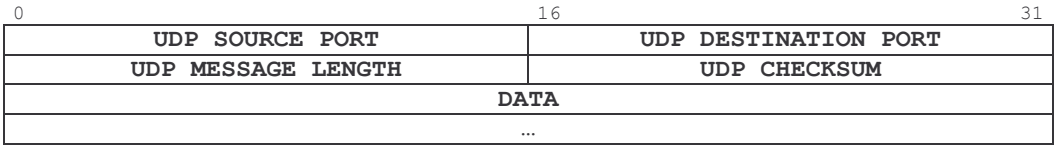


Figura A.11. Formato do datagrama UDP.

As descrições de cada campo podem ser conferidas a seguir:

- `UDP SOURCE PORT`: inteiro de 16 bits correspondente à porta de origem;
- `UDP DESTIONATION PORT`: inteiro de 16 bits correspondente à porta de destino, utilizado juntamente com `UDP SOURCE PORT` para demultiplexar corretamente todos os datagramas UDP que chegam a um destino final;
- `UDP MESSAGE LENGTH`: tamanho total do datagrama UDP, em bytes, somando as áreas de cabeçalho e dados;
- `UDP CHECKSUM`: *checksum* do datagrama, calculado de maneira similar ao `CHECKSUM` do TCP. Considera um pseudo-cabeçalho, isto é, uma extensão "virtual" do cabeçalho que não é transmitida, para feito de cálculos. Maiores detalhes em Comer (2000);
- `DATA`: Área de dados com tamanho variável.

A.5 PROTOCOLOS DE APLICAÇÃO

As seções anteriores serviram para explicar a infra-estrutura que a arquitetura TCP/IP provê para os programas de aplicação que desejam realizar comunicação via rede. Uma das aplicações mais comuns da Internet, a *World Wide Web*, pode ser tomada como exemplo para explicar o conceito de protocolo de aplicação.

A *World Wide Web* consiste na obtenção de *home pages* via Internet para serem visualizadas na máquina de um usuário através de um programa denominado *browser* (Internet Explorer, Mozilla Firefox, etc).

Quando o usuário digita um endereço na barra de navegação, o *browser* utiliza o protocolo HTTP (*Hyper Text Transfer Protocol*) para solicitar e obter uma *home page* no servidor identificado pelo endereço digitado.

Um mecanismo não mencionado até aqui entra em ação para converter o endereço digitado pelo usuário, por exemplo, http://usp.br, em um endereço IP correspondente. Isso é implementado através de um conceito chamado *Domain Name System* (DNS), cuja explicação foge ao escopo deste trabalho e pode ser encontrada na literatura especializada, como por exemplo Comer (2000). Considera-se aqui que há uma maneira transparente de converter os endereços baseados em nomes para endereços IP.

Após obter o endereço IP do servidor que hospeda a *home page* solicitada pelo usuário, o *browser* monta então uma requisição HTTP conforme as especificações deste protocolo, solicitando ao servidor de destino que envie o conteúdo da *home page* hospedada. O HTTP funciona sobre os serviços do TCP, sendo a porta 80 o padrão estabelecido para servidores *web*. Portanto, uma conexão TCP na porta 80 é solicitada pelo *browser*. Após o estabelecimento da conexão, tem-se um circuito virtual disponível para comunicação confiável entre o *browser* e o servidor *web*. É através dele que a requisição HTTP é encaminhada. É tarefa do software do protocolo TCP, e não do *browser*, dividir essa requisição em segmentos se for necessário (na prática, requisições HTTP costumam ser pequenas e não precisam ser segmentadas). O software do TCP, por sua vez, passa esses segmentos ao software da camada IP, que os encapsula em datagramas IP e os transmite via rede. Na outra ponta, a camada IP desencapsula o(s) segmento(s) TCP dos datagramas IP, repassa-os à camada TCP, que por sua vez reconstrói o *stream* original e o entrega ao software da camada de aplicação. Nesse exemplo, esse software é o servidor *web*. Como já visto, é função do TCP detectar a perda de pacotes e retransmiti-los se for necessário.

O servidor *web*, ao receber a requisição, obtém a *home page* solicitada em seu disco rígido e a devolve ao *browser* através do mesmo circuito virtual. A página é exibida na tela do usuário e a conexão encerrada.

Observe que tanto o desenvolvedor do *browser* quanto o do servidor web não precisam se preocupar com a implementação da detecção de perda de pacotes e retransmissão. Mais ainda, não precisam fazer nenhuma suposição a respeito do funcionamento da rede em que as máquinas estão conectadas.

Esse exemplo, apesar de simplificado, permite visualizar a robustez que a separação conceitual em camadas da Figura A.1 e sua implementação através dos protocolos TCP e IP dão a toda infra-estrutura formada.

Na Internet, existem inúmeras aplicações. Outras bastante comuns que podem ser citadas são a transferência de arquivos (protocolo FTP) e correio eletrônico (protocolos POP e SMTP). Normalmente, um número de porta é padronizado para cada aplicação, permitindo que os softwares que solicitam conexões (clientes) saibam como encontrar os aplicativos que provêem os serviços que estão aptos a utilizar (servidores). Cada aplicação define o seu próprio protocolo e estabelece se ele deve funcionar sobre o TCP ou UDP.

APÊNDICE B – Detalhes sobre o protocolo LonTalk

Este apêndice tem como objetivo fornecer uma descrição do protocolo LonTalk com detalhes sobre cada camada (com exceção da camada física), servindo como um guia de consulta e aprendizado para o leitor que necessitar dessas informações.

B.1 CAMADA FÍSICA

A tecnologia LonWorks suporta diversos meios físicos de comunicação, entre eles o par trançado, a rede elétrica e fibra ótica. Cada meio físico possui características de custo, confiabilidade e desempenho próprios, aceitando taxas de transmissão distintas. Os requisitos de aplicação, instalação, manutenção e custo, entre outros, são os fatores determinantes da escolha do meio físico mais apropriado.

É responsabilidade do transceptor realizar a interface entre o Neuron Chip e o canal de comunicação. Uma rede LonWorks pode ser formada por diversos canais baseados em diferentes meios físicos, os quais são interligados através de roteadores.

Os detalhes sobre cada meio físico fogem ao escopo desse texto e não serão tratados aqui.

B.2 CAMADA DE ENLACE

A camada de enlace do protocolo LonTalk define as regras de acesso ao meio físico e o formato do quadro (*frame*) LonTalk.

LonTalk utiliza o conceito de *Carrier Sense Multiple Access* (CSMA), o qual estabelece que um transmissor deve escutar o canal antes de iniciar o envio de um pacote. Se o canal estiver ocupado, isto é, se já houver uma transmissão ocorrendo, o transmissor deve aguardar até a liberação do canal e tentar novamente. Dois transmissores provocam uma colisão quando iniciam uma transmissão em instantes de tempo muito próximos. Isso pode ocorrer pois há um tempo τ de propagação do sinal no meio físico entre os dois transmissores. Supondo um canal livre onde o primeiro transmissor inicia uma transmissão no instante T_0, o segundo transmissor detectará o canal como livre até o instante $T_0 + \tau$. Se o segundo iniciar

uma transmissão nesse intervalo, uma colisão ocorrerá. Após $T_0 + \tau$, o segundo transmissor identificaria o canal como ocupado, e, portanto, aguardaria para transmitir.

Quando uma colisão ocorre, as mensagens envolvidas são perdidas e devem ser retransmitidas.

A tecnologia LonWorks utiliza um mecanismo baseado em *slots* de tempo para evitar colisões. Consiste em uma extensão do CSMA denominada *predictive p-persistent CSMA*. Quando a transmissão de um quadro LonWorks termina, a rede fica ociosa (isto é, nenhum nó transmite) por um intervalo de tempo pré-fixado denominado *Beta 1 slot*. Após o *Beta 1 slot*, há 16 *slots* de tempo denominados *Beta 2 slots*. A Figura B.1 provê uma ilustração.

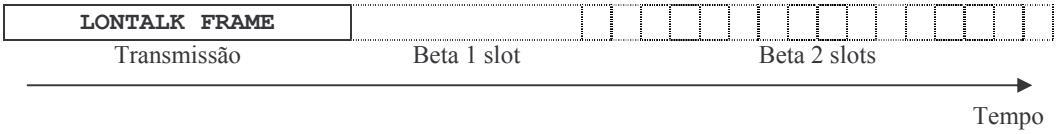


Figura B.1. Transmissão de quadro LonTalk e Beta slots.

Cada nó que aguardava pela liberação do canal para enviar um quadro seleciona aleatoriamente um dos 16 *Beta 2 slots*. O *slot* selecionado é aquele no qual a transmissão será iniciada. Cada *slot*, portanto, pode ter sido escolhido por pelo menos um transmissor ou então por nenhum. No primeiro *slot* com algum transmissor associado, um ou mais quadros serão enviados. Se o *slot* em questão tiver sido selecionado por um único transmissor, ele conseguirá enviar seu quadro sem colisões (os transmissores que escolheram *slots* subseqüentes detectarão que a rede está ocupada e aguardarão até o fim da transmissão do pacote). Se o *slot* tiver sido escolhido por mais de um transmissor, mais de um quadro será enviado à rede simultaneamente e, portanto, uma colisão ocorrerá.

O protocolo LonTalk permite, na verdade, que o número de *Beta 2 slots* seja maior que 16 e possa variar ao longo do tempo. Isso é bastante útil quando o tráfego na rede aumenta, situação na qual a quantidade de quadros a serem enviados por unidade de tempo é maior. O aumento do número de *Beta 2 slots* representa um maior conjunto de opções para escolha aleatória, diminuindo a probabilidade de um mesmo *slot* ser selecionado por mais de um transmissor. A probabilidade de colisão, portanto, é reduzida. O número de *Beta 2 slots* é denotado por R e pode variar de 16 a 1008. Ele é calculado com base em outra variável, conhecida como *backlog* (BL), através da fórmula R = 16 x BL. BL pode assumir os valores de 1 a 63.

Existe mais um importante detalhe, relacionado à priorização de quadros, que deve ser considerado. Entre o instante final do *Beta 1 slot* e o primeiro *slot* disponível para transmissão, existem *slots* de prioridade. Nós que desejam transmitir quadros prioritários e que possuem um número pré-configurado de *slot* de prioridade irão transmitir neste *slot*. O uso de prioridade reduz a probabilidade de colisão e favorece o envio antecipado dos quadros mais urgentes. O número de *slots* de prioridade é denotado por P e é pré-determinado para cada canal, podendo estar na faixa de 0 a 127. Esses *slots* também fazem parte do grupo de *Beta 2 slots*, e, portanto, o número de *Beta 2 slots* é igual a R + P. A Figura B.2 oferece uma visualização gráfica.

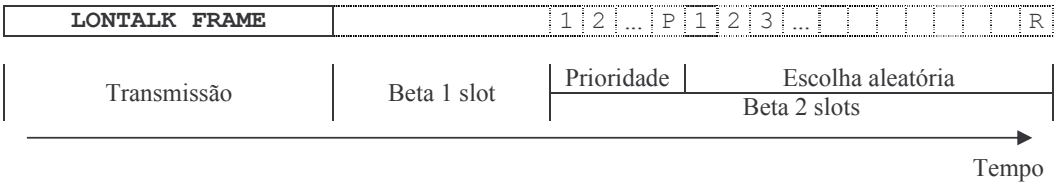


Figura B.2. Transmissão de quadro LonTalk e Beta slots de prioridade.

A definição do formato do quadro LonTalk, ou *Link PDU* (LPDU), também faz parte da camada de enlace.

O LPDU é composto por três partes: o cabeçalho, a área de dados e um CRC para verificação de erros, conforme indicado na Figura B.3. A área de dados do quadro LonTalk abriga um NPDU (*Network PDU*), isto é, o PDU da camada de rede. Isto é análogo ao fato de a área de dados do quadro Ethernet abrigar um datagrama IP.

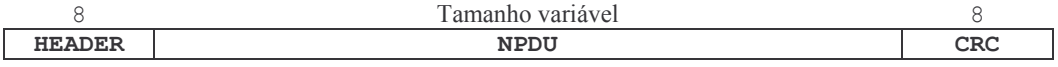


Figura B.3. LPDU do protocolo LonTalk

Os campos presentes na Figura B.3 são:

- HEADER: contém o cabeçalho do quadro, com 8 bits de tamanho, cujo formato está indicado na Figura B.4;

- NPDU: área de dados do quadro. Serve para abrigar um NPDU (PDU da camada de rede), que será detalhado mais adiante;
- CRC: código de CRC de 16 bits gerado pelo transmissor. O receptor faz o seu próprio cálculo e compara o resultado com este campo. Se não forem iguais, o pacote é dado como incorreto e, portanto, descartado. O CRC é calculado sobre o conteúdo de HEADER e NPDU. O polinômio utilizado é $x^{16} + x^{12} + x^5 + 1$.

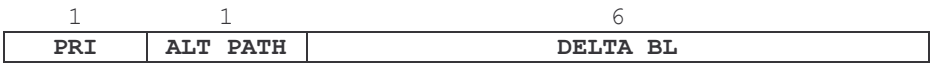


Figura B.4. Campo HEADER do LPDU do protocolo LonTalk.

Os sub-campos de HEADER são:

- PRI: indicador de prioridade. Quadros prioritários possuem PRI = 1, enquanto os normais utilizam PRI = 0;
- ALT PATH: caminho alternativo. Alguns transceptores LonWorks podem ser conectados a dois canais simultaneamente. Este campo especifica para qual canal o quadro deve ser transmitido;
- DELTA BL: se este campo for diferente de zero, indica aos receptores que o *backlog* (BL) deve ser incrementado. O incremento do *backlog*, como já mencionado, resulta em um aumento do número de *Beta 2 slots*, pois R = 16 x BL. Um algoritmo definido pelo protocolo LonTalk, que não será detalhado aqui, é responsável por calcular o novo *backlog* com base nesse campo e na atividade da rede. Uma situação interessante em que DELTA BL pode ser utilizado é quando um nó faz uma transmissão *multicast* com serviço *acknowledged* ou *request/response*. Ao iniciar a transmissão, o dispositivo já sabe que deverá esperar por um certo número de ACKs igual ao número de membros do grupo de destino. Em outras palavras, o nó sabe que sua mensagem terá como conseqüência uma rajada (*burst*) de tráfego na rede. Nesse caso, ele pode sinalizar a necessidade de aumento dos *Beta 2 slots* através de um valor apropriado em DELTA BL.

B.3 CAMADA DE REDE

A camada de rede é responsável por endereçar e encaminhar pacotes entre nós de um mesmo domínio. Entretanto, nenhuma precaução é tomada no sentido de corrigir erros de comunicação. Essa atribuição é da camada de transporte.

O formato do NPDU pode ser conferido na Figura B.5.

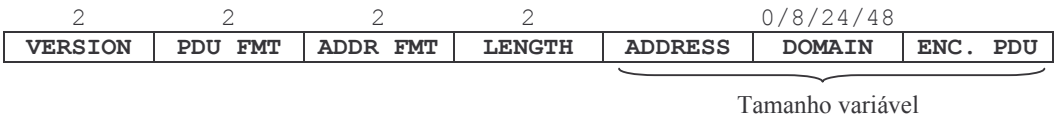


Figura B.5. NPDU do protocolo LonTalk.

Os campos do NPDU são:

- VERSION: versão do protocolo utilizada na geração do pacote. É representada por um identificador de zero a três;
- PDU FMT: indica o formato dos dados contidos na área ENCLOSED PDU (ENC. PDU). É utilizado para que as camadas superiores do protocolo saibam como interpretar esses dados. Os valores possíveis são:
  - 0: indica que o campo ENCLOSED PDU contém um PDU de transporte, ou TPDU (*Transport PDU*);
  - 1: indica que o campo ENCLOSED PDU contém um PDU de sessão, ou SPDU (*Session PDU*);
  - 2: indica que o campo ENCLOSED PDU contém um PDU de autenticação, ou AuthPDU (*Authentication PDU*);
  - 3: indica que o campo ENCLOSED PDU contém um PDU de aplicação, ou APDU (*Application PDU*).
- ADDR FMT: especifica o tipo de endereçamento a ser utilizado para o encaminhamento do pacote ao seu destino final. O endereço em si é especificado nos campos ADDRESS e DOMAIN. Os valores possíveis são:

- 0: *broadcast*. envia a mensagem para todos os nós de um domínio ou para todos os nós de uma sub-rede específica. Aceita apenas os serviços *unacknowledged* e *repeated*;
- 1: *multicast*. envia a mensagem para todos os membros de um grupo. Aceita todos os tipos de serviço;
- 2: *unicast*. envia a mensagem para um determinado nó através de endereçamento por domínio / sub-rede / nó ou domínio / grupo / *member id*. Aceita todos os tipos de serviço;
- 3: *unicast*: envia a mensagem para um determinado nó através de seu Neuron ID. Não é necessário que o remetente e o destinatário estejam no mesmo domínio, ao contrário dos casos acima. Aceita todos os tipos de serviço;

- LENGTH: especifica o tamanho do campo DOMAIN. A tecnologia LonWorks suporta identificadores de domínios de 0, 1, 3 ou 6 bytes. O domínio de 0 bytes representa um domínio sem identificador. Os valores possíveis são:

  - 0: domínio de 0 bytes;
  - 1: domínio de 1 byte;
  - 2: domínio de 3 bytes;
  - 3: domínio de 6 bytes.

- ADDRESS: armazena os endereços de origem e destino do pacote. O endereço de origem é sempre armazenado no formato domínio / sub-rede / nó. O de destino varia conforme especificação do campo ADDR FMT. As sub-divisões do campo ADDRESS serão vistas mais adiante. O domínio, que é o mesmo para o remetente e destinatário, é armazenado no campo DOMAIN, e não dentro de ADDRESS;
- DOMAIN: armazena o domínio da origem e do destino do pacote, complementando o campo ADDRESS. Possui tamanho variável especificado em LENGTH. Este campo é nulo se o domínio utilizado for de 0 bytes.

O campo ADDRESS é dividido em sub-campos, cuja interpretação varia conforme o valor de ADDR FMT. A Figura B.6 mostra a subdivisão de ADDRESS.

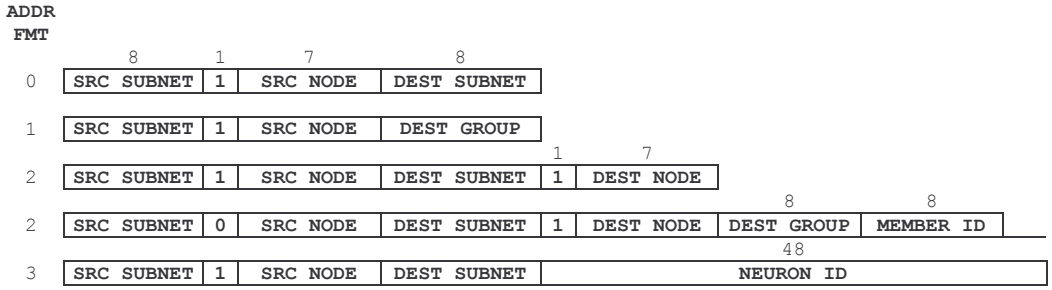


Figura B.6. Campo ADDRESS do NPDU do protocolo LonTalk.

Os sub-campos de ADDRESS são:

- SRC SUBNET: sub-rede do nó de origem;
- SRC NODE: número do nó de origem;
- DEST SUBNET: sub-rede de destino. Em caso de *broadcast* (ADDR FMT = 0) para todas as sub-redes do domínio, DEST SUBNET deve conter o valor 0;
- DEST GROUP: grupo de destino usado em endereçamento *multicast* (ADDR FMT = 1);
- DEST NODE: número do nó de destino;
- MEMBER ID: identificador do membro de destino dentro de um grupo;
- NEURON ID: neuron ID do nó de destino. Quando esse tipo de endereçamento é utilizado, o campo DEST SUBNET deve conter 0 para que a mensagem passe por todos os roteadores.

B.4 CAMADA DE TRANSPORTE

A camada de transporte do protocolo LonTalk é responsável por prover os serviços de entrega *acknowledged, unacknowledged* e *repeated*. O formato do PDU de transporte, ou TPDU (*Transport PDU*), está ilustrado na Figura B.7. Os TPDUs são colocados na área ENCLOSED PDU do NPDU (Figura A.5), que se trata do PDU da camada de rede. Nesse caso, o campo PDU FMT, também do NPDU, contém o valor 0.

Figura B.7. TPDU do protocolo LonTalk.

Os campos presentes na Figura B.7 são os seguintes:

- AUTH: bit identificando se a transação requer autenticação;
- TPDU TYPE: especifica o tipo de serviço de entrega e, ao mesmo tempo, o formato do campo TPDU CONTENT, ou seja, como este deve ser interpretado. Os valores possíveis são:

    - 0: especifica que o pacote em questão utiliza o serviço *acknowledged*. O receptor deve enviar um ACK de volta ao transmissor.
    - 1: especifica que o pacote em questão utiliza o serviço *unacknowledged* ou *repeated*. A diferença entre ambos está na ausência ou presença de repetições de transmissão. O pacote LonTalk não carrega a informação do número de repetições empregadas no serviço *repeated*, que fica exclusivamente a critério do transmissor. O receptor encara as repetições de pacotes como pacotes duplicados, e essa identificação é feita pelo TRANSACTION NUMBER (*tag*).
    - 2: indica que o pacote carrega um ACK. Nesse caso, TPDU tem tamanho nulo;
    - 4: indica que o pacote carrega um lembrete (*reminder*). Um lembrete é utilizado em transmissões *multicast* com os serviços *acknowledged* e *request/response*. Nesses casos, um ACK é esperado de cada receptor. Quando o temporizador de reenvio do transmissor expira, pode ser que ACKs de alguns receptores tenham chegado e outros não. Se a mensagem for reenviada, muitos dos receptores que a receberam corretamente na primeira transmissão, e inclusive já responderam com um ACK, iriam enviar novos ACKs desnecessariamente na rede. Uma mensagem *reminder* antecede uma retransmissão de pacote com endereçamento *multicast*, e serve para avisar aos receptores envolvidos de quais deles o transmissor ainda não recebeu um ACK;
    - 5: similar ao caso anterior, mas nesta situação o lembrete segue no mesmo pacote que a retransmissão da mensagem, e não em pacotes separados.

- TRANSACTION NUMBER: número *tag* que identifica a transação. Permite que o transmissor associe ACKs recebidos a pacotes enviados, além de possibilitar a detecção de mensagens duplicadas por parte do receptor;
- TPDU CONTENT: área de dados do TPDU, cujo formato e tamanho varia conforme valor do campo TPDU TYPE (Figura B.8).

O APDU é o PDU da camada de aplicação, o qual será detalhado mais adiante. A Figura B.8 indica que o TPDU carrega um APDU em sua área de dados para TPDU TYPE igual a 0, 1 ou 5.

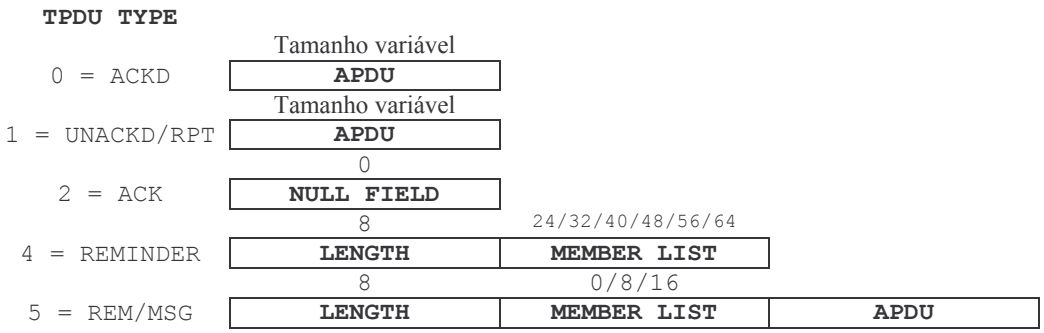


Figura B.8. Campo TPDU CONTENT do TPDU do protocolo LonTalk.

Os campos que aparecem na situação em que TPDU TYPE indica um lembrete, ou seja, 4 para *reminder* ou 5 para *reminder/message*, são:

- LENGTH: contém o número de nós envolvidos na transmissão *multicast*, isto é, o número de membros do grupo para o qual a mensagem original foi endereçada. Determina também, indiretamente, o número de bits do campo MEMBER LIST, que é igual ao menor múltiplo de 8 maior ou igual ao número de membros do grupo. Por exemplo, se LENGTH = 7, então MEMBER LIST terá 8 bits. Já para LENGTH = 42, MEMBER LIST terá 48 bits;
- MEMBER LIST: de acordo com a explicação de LENGTH, este campo tem sempre tamanho suficiente para conter uma seqüência de bits onde cada bit corresponde a um membro do grupo para o qual se está enviando o lembrete. O primeiro bit corresponde ao membro 1, o segundo corresponde ao membro 2, e assim por diante. Se um bit for igual a 0, então o nó transmissor ainda está aguardando um

ACK do membro correspondente. Os receptores cujo bit correspondente em `MEMBER LIST` é igual a 1 não enviam outro ACK ao transmissor.

Se o número de membros do grupo for menor ou igual a 16, é possível retransmitir a mensagem enviada originalmente dentro do mesmo pacote que sinaliza o lembrete. Nesse caso, utiliza-se o valor 5 em `TPDU TYPE` (Figura B.8). Se o número de membros do grupo for maior que 16, o lembrete e a retransmissão da mensagem são enviados em pacotes separados. Para isso, utiliza-se o valor 4 em `TPDU TYPE` (Figura B.8).

Para exemplificar o funcionamento da camada de transporte do protocolo LonTalk, a Figura B.9 ilustra o diagrama de tempos de uma comunicação *acknowledged* sem perdas, enquanto a Figura B.10 mostra uma transmissão *acknowledged* com endereçamento *multicast* e perda de pacotes.

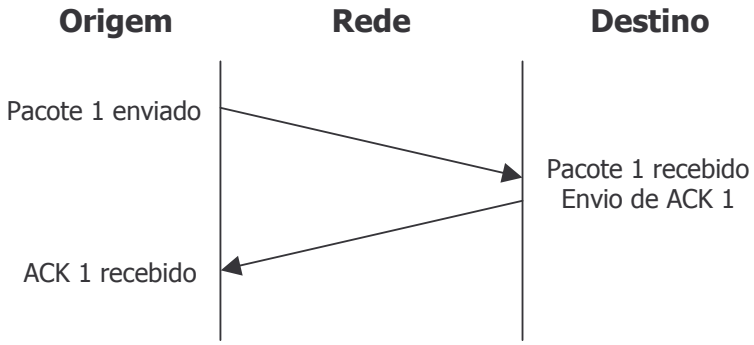


Figura B.9. Transação com serviço acknowledged sem perdas e envio unicast.

Na Figura B.10, foi suposto que o grupo para o qual a transmissão foi endereçada possui mais de 16 membros. Caso contrário, o lembrete e o reenvio da mensagem estariam no mesmo pacote.

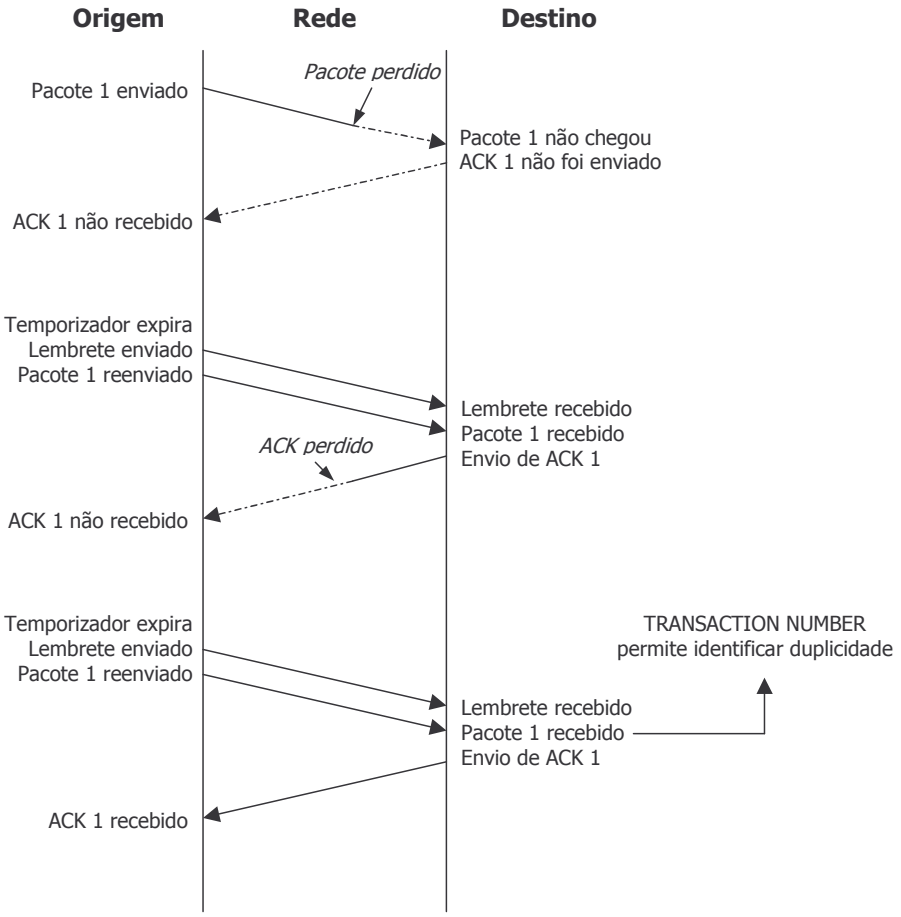


Figura B.10. Transação com serviço acknowledged com perdas e envio multicast.

B.5 CAMADA DE SESSÃO

A camada de sessão do protocolo LonTalk oferece o serviço *request/response*, similar ao *acknowledged* mas com a possibilidade de o programa de aplicação do receptor processar a requisição do transmissor e devolver uma resposta juntamente com o ACK. O PDU desta camada é o SPDU (*Session PDU*), que assume o formato indicado na Figura B.11. Os SPDUs são colocados na área `ENCLOSED PDU` do NPDU (Figura A.1), que se trata do PDU da camada de rede. Nesse caso, o campo `PDU FMT`, também do NPDU, contém o valor 1.

Figura B.11. SPDU do protocolo LonTalk.

Os campos presentes na Figura B.11 são os seguintes:

- AUTH: idêntico ao campo AUTH do TPDU;
- SPDU TYPE: especifica o tipo do conteúdo. O formato do campo SPDU CONTENT, ou seja, como este deve ser interpretado, é uma conseqüência do valor de SPDU TYPE. Os valores possíveis são:

    - 0: especifica que o pacote em questão carrega uma requisição (*request*). É o pacote enviado pelo transmissor que deve ser processado e respondido pelo receptor;
    - 2: especifica que o pacote em questão carrega uma resposta (*response*). Funciona também como um ACK;
    - 4: indica que o pacote carrega um lembrete (*reminder*). Tem o mesmo objetivo do lembrete da camada de transporte, conforme a explicação anterior do campo TPDU TYPE. No lugar do ACK, aqui lê-se *response*;
    - 5: indica que o pacote carrega um lembrete junto com uma retransmissão de mensagem. Vide explicação correspondente para o campo TPDU TYPE da camada de transporte.

- TRANSACTION NUMBER: mesmo significado do campo TRANSACTION NUMBER do TPDU;
- SPDU CONTENT: área de dados do SPDU, cujo formato e tamanho varia conforme valor do campo SPDU TYPE. Veja a Figura B.12.

A maioria das transações de gerenciamento e diagnóstico de rede costuma utilizar o serviço *request/response*, já que normalmente o nó gerenciado ou diagnosticado precisa retornar um valor ao solicitante.

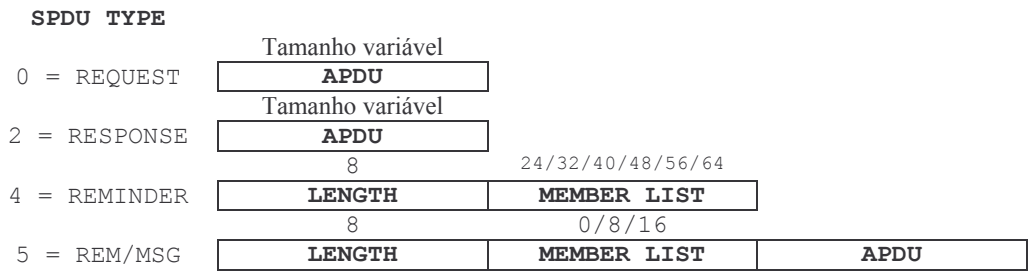


Figura B.12. Campo SPDU CONTENT do SPDU do protocolo LonTalk.

B.6 CAMADAS DE APRESENTAÇÃO E APLICAÇÃO

As camadas de apresentação e aplicação do protocolo LonTalk são bastante ligadas uma à outra. Suas implementações se dão, de certa forma, conjuntamente.

A camada de apresentação, cujo objetivo é receber e entregar dados à camada de aplicação em um formato com o qual esta seja capaz de lidar, é implementada através do conceito das variáveis de rede (NVs). Os programas de aplicação de nós LonWorks são comumente desenvolvidos para realizar comunicação através de NVs, que, conforme já explicado, são entradas e saídas lógicas que podem ser conectadas a NVs de outros nós através de um processo denominado *binding*. Um *binding* só pode ser realizado entre NVs do mesmo tipo, ou seja, que apresentam a mesma codificação da informação (sintaxe) e o mesmo significado (semântica). A organização LonMark padroniza tipos de NVs. Os tipos padronizados são conhecidos como *Standard Network Variable Types* (SNVTs).

Embora exista o conceito das NVs, programas de aplicação de dispositivos LonWorks também podem se comunicar através da troca de dados em formatos proprietários trafegando dentro de pacotes LonTalk. As mensagens dessa categoria são chamadas de mensagens explícitas (*explicit messages*). Nesse caso, a responsabilidade da codificação, decodificação e interpretação do conteúdo dessas mensagens é inteiramente do programa de aplicação dos nós envolvidos, ao contrário das NVs, cujas rotinas estão implementadas no *firmware* do Neuron Chip (camada de apresentação do protocolo LonTalk).

Mensagens de gerenciamento e diagnóstico são outra possibilidade de comunicação entre nós LonWorks. Essas mensagens são tipicamente geradas por ferramentas de software executadas em PCs conectados na rede, que podem ser encarados como nós com grande capacidade de processamento.

Por fim, as redes LonWorks também permitem que mensagens geradas por dispositivos não LonWorks trafeguem pela rede. Elas normalmente são introduzidas por um

gateway que interliga a rede LonWorks com outro tipo de rede. As mensagens dessa categoria são chamadas de *foreign frames*.

O PDU da camada de apresentação / aplicação do protocolo LonTalk é denominado APDU (*Application PDU*). Seu formato pode ser visto na Figura B.13. O APDU aparece como conteúdo da área de dados do TPDU e SPDU, como foi mencionado anteriormente.

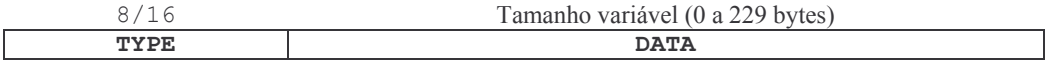


Figura B.13. APDU do protocolo LonTalk.

Os campos presentes na Figura B.13 são os seguintes:

- TYPE: determina o tipo da mensagem e o significado do conteúdo do campo DATA. Pode ter 8 ou 16 bits, conforme indica o seu bit mais significativo. Se este for igual a 1, então terá 16 bits e, caso contrário, 8. Os primeiros bits de TYPE determinam não só como DATA deve ser interpretado, mas também como ele próprio deve ser lido. A Figura B.14 ilustra as possibilidades para esse campo.

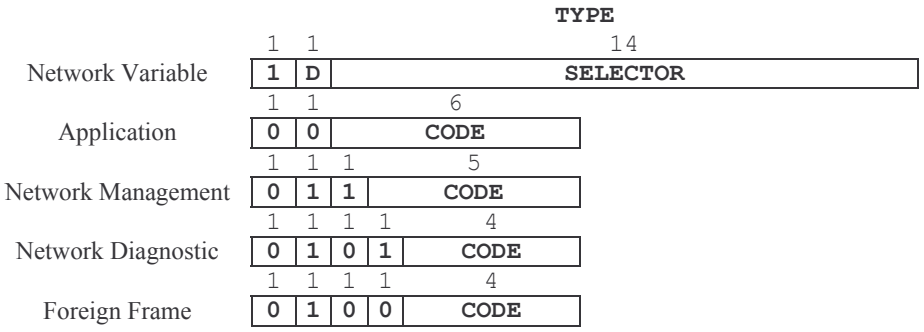


Figura B.14. Campo TYPE do APDU do protocolo LonTalk.

Como demonstra a Figura B.14, o campo TYPE indica qual o tipo de mensagem que o pacote LonWorks carrega:

- *Network Variable*: mensagem contendo uma solicitação ou atualização de valor de NV. É comumente utilizada na comunicação entre nós com NVs ligadas por *bindings*, quando a alteração do valor de uma NV de saída implica na atualização do valor de uma NV de entrada de outro nó. Nesse

caso, o campo TYPE possui 16 bits. Em todos os outros, possui 8. O campo DATA do APDU (Figura B.13) contém o valor da NV, e o campo SELECTOR (Figura B.14), identifica a qual NV do nó a mensagem se refere. O formato em que o valor da NV é codificado depende de seu tipo. Para SNVTs, eles são padronizados e publicados pelo órgão LonMark. O campo D (Figura A.10) é uma abreviação para DIRECTION e estabelece se a NV é de saída (D = 1) ou de entrada (D = 0);

- *Application*: mensagem explícita cujo formato deve ser conhecido pelos programas de aplicação envolvidos na comunicação. O campo CODE permite que o transmissor atribua um código de identificação a cada tipo de mensagem, bastante útil em aplicações onde diversos tipos de mensagens explícitas podem ser enviados;
- *Network Management*: mensagem de gerenciamento. Os valores possíveis para CODE e os nomes de cada mensagem podem ser encontrados na Tabela B.1 (os valores da coluna CODE incluem os bits iniciais 011 da Figura B.14). Cada mensagem de gerenciamento possui também um código de resposta indicando sucesso e outro indicando fracasso, que são devolvidos através do serviço *request/response*. Esse O Apêndice B de Toshiba (1995) apresenta uma descrição completa de cada mensagem de gerenciamento;
- *Network Diagnostic*: mensagem de diagnóstico. Os valores possíveis para CODE e os nomes de cada mensagem podem ser encontrados na Tabela B.2 (os valores da coluna CODE incluem os bits iniciais 0101 da Figura B.14). Cada mensagem de diagnóstico possui também um código de resposta indicando sucesso e outro indicando fracasso, que são devolvidos através do serviço *request/response*. O Apêndice B de Toshiba (1995) apresenta uma descrição completa de cada mensagem de diagnóstico;
- *Foreign Frame*: mensagem não LonWorks que trafega pela rede. Normalmente são geradas por outras redes e inseridas na rede LonWorks através de um *gateway*. O campo CODE permite atribuir um identificador de 4 bits ao tipo de *foreign frame*.

- DATA: conteúdo da mensagem. Em caso de transmissão de valor de NV, contém o valor da NV propriamente dita codificado segundo o seu tipo. Em caso de mensagem de gerenciamento ou diagnóstico, contém os dados envolvidos na requisição ou no retorno da transação, próprios de cada tipo de mensagem. Para

mensagens explícitas e *foreign frames*, DATA carrega o conteúdo da mensagem em formato proprietário, não definidos pela tecnologia LonWorks.

Tabela B.1 – Mensagens de gerenciamento do protocolo LonTalk.
Extraído de Toshiba (1995).

Mensagem	CODE (hex)	Código de sucesso	Código de fracasso
Query Status	51	31	11
Proxy Command	52	32	12
Clear Status	53	33	13
Query XCVR Status	54	34	14

Tabela B.2 – Mensagens de diagnóstico do protocolo LonTalk.
Extraído de Toshiba (1995).

Mensagem	CODE (hex)	Código de sucesso	Código de fracasso
Query ID	61	21	01
Respond to Query	62	22	02
Update Domain	63	23	03
Leave Domain	64	24	04
Update Key	65	25	05
Update Address	66	26	06
Query Address	67	27	07
Query Net Variable Config	68	28	08
Update Group Address Data	69	29	09
Query Domain	6A	2A	0A
Update Net Variable Config	6B	2B	0B
Set Node Mode	6C	2C	0C
Read Memory	6D	2D	0D
Write Memory	6E	2E	0E
Checksum Recalculate	6F	2F	0F
Wink	70	30	10
Memory Refresh	71	31	11
Query SNVT	72	32	12
Network Variable Fetch	73	33	13
Device Escape Code	7D	3D	1D